

## Функция Abs( )

### Назначение:

Возвращает абсолютное значение числа.

### Синтаксис:

**Abs(*num\_expr*)**

где

*num\_expr* – численное выражение.

### Величина, полученная в результате:

Число с плавающей запятой. Величина типа Float.

### Описание:

Функция **Abs(\_)** возвращает абсолютное значение числа, полученного в результате вычисления выражения *num\_expr*.

Если результат выражения больше или равен нулю, **Abs(\_)** возвращает это число без изменений.

Если же результат выражения меньше нуля, **Abs(\_)** возвращает значение выражения, умноженное на минус единицу.

### Пример:

```
Dim x, y As Float
x = -2.5
y = Abs(x)
'
' y равно 2.5
```

### Смотрите также:

**Sgn(\_)**

### Функция `Acos( )`

#### Назначение:

Возвращает арккосинус числа.

#### Синтаксис:

**`Acos(num_expr)`**

где

*num\_expr* – численное выражение, результат которого должен находиться в диапазоне от единицы до минус единицы включительно.

#### Величина, полученная в результате:

Число с плавающей запятой. Величина типа Float.

#### Описание:

Функция **`Acos( )`** вычисляет арккосинус числа, полученного в результате вычисления выражения *num\_expr*. Другими словами, **`Acos( )`** возвращает величину угла (в радианах), косинус которого равен параметру *num\_expr*.

Диапазон значения угла – между 0 и  $\pi/2$  радиан (число  $\pi$  равно приблизительно 3.141593, и  $\pi/2$  радиан равно 90 градусам).

Для перевода градусов в радианы число необходимо умножить на число `DEG_2_RAD`. Для обратного конвертирования используется коэффициент `RAD_2_DEG`. Для того, чтобы Ваша программа могла использовать эти коэффициенты конвертирования, она должна содержать оператор **`Include "MAPBASIC.DEF"`**.

Так как значения косинуса могут находиться в диапазоне от минус единицы до единицы, то и результат вычисления выражения *num\_expr* должен быть не менее минус единицы и не более единицы.

#### Пример:

```
Include "MAPBASIC.DEF"
Dim x, y As Float
x = 0.5
y = Acos(x) * RAD_2_DEG
'
' y будет равен 60,
' так как косинус от 60 градусов равен 0.5
'
```

#### Смотрите также:

**`Asin( ), Atn( ), Cos( ), Sin( ), Tan( )`**

## Add Cartographic Frame

Оператор **Create Cartographic Frame** позволяет добавлять разделы к существующей картографической легенде, созданной оператором **Create Cartographic Legend**.

### Синтаксис:

#### Add Cartographic Frame

```
[Window legend_window_id ]
[Custom]
[Default Frame Title [def-frame_title] [Font... ] ]
[Default Frame Subtitle [def-frame_subtitle] [Font... ] ]
[Default Frame Style [def-frame_style] [Font... ] ]
[Default Frame Border Pen... pen-expr]
Frame From Layer [map-layer_id | map-layer_name]
  [ Position (x, y) [Units paper-units] ]
  [ Using
    [ Column { column | object } ]
    [ Label { expression | default } ]
  [ Title [frame_title] [Font... ] ]
  [ SubTitle [frame_subtitle] [Font... ] ]
  [ Border Pen... ]
  [ Style [Font... ] [NoRefresh ]
    [ Text { style_name } { Line Pen... | Region Pen... Brush...
      | Symbol Symbol... } ]
    [, ... ]
  ]
[ ... ]
```

*legend\_window\_id* - это целочисленный идентификатор окна, который Вы можете получить при вызове функций FrontWindow() и WindowId().

*def-frame\_title* - это строковая величина, которая по умолчанию определяет заголовок раздела легенды. Может включаться специальный символ "#", который может заменяться именем текущего слоя.

*def\_frame\_subtitle* - это строковая величина, которая по умолчанию определяет заголовок подраздела легенды. Может включаться специальный символ "#", который может заменяться именем текущего слоя.

*def\_frame\_style* - это строковая величина, которая определяет каждый символ в каждом разделе легенды. Символ "#" будет замещаться именем слоя. Символ % будет замещаться текстом "Line", "Point", "Region", как это определено для символа. Например, "% of #" будет означать "Region of States" для слоя states.tab.

*pen\_expr* - это выражение Pen, то есть MakePen( ширина, тип, цвет). Если по умолчанию линия рамки определена, то она и останется по умолчанию такой же для раздела легенды. Если предложение pen для рамки существует для раздела легенды, то оно будет определять тип линии для рамки вместо заданного по умолчанию.

*map\_layer\_id* или *map\_layer\_name* определяют слой карты; задается коротким целочисленным типом (например, используйте 1 для определения самого верхнего слоя, не считая косметического) или строковая переменная, соответствующая имени таблицы, отображенной на карте. Для

## Add Cartographic Frame

---

тематического слоя необходимо определять параметр *map\_layer\_id*.

*frame\_title* - это строковая переменная, определяющая заголовок раздела легенды. Если будет определено предложение заголовка раздела, то будет использоваться эта величина вместо величины *def\_frame\_title*.

*frame\_subtitle* - это строковая переменная, определяющая подзаголовок раздела легенды. Если это предложение определено, то оно будет использоваться вместо величины *def\_frame\_subtitle*.

*Column* это атрибутивное имя колонки из раздела слоя таблицы или колонка 'object' (означает, что стили легенды базируются на уникальном стиле в mapfile). По умолчанию это 'object'.

*Label* это полноценное выражение или 'по умолчанию' (означает, что используется по умолчанию стиль раздела шаблона, когда создается каждый стиль текста, если предложение стиля не содержит текст).

*style\_name* - это строковая переменная, которая указывает к какому типу: символу, линии или области относится объект в разделе легенды.

### Описание

Если ключевое слово **Custom** включено в описание, то каждый раздел легенды должен включать в себя предложение **Position**. Если **Custom** пропущено и легенда отображается в виде книжной или альбомной ориентации, то разделы легенды будут добавлены в самый конец.

Предложение **Position** контролирует положение раздела в окне легенды. Верхний левый угол окна легенды имеет позицию 0, 0. Величины позиционирования используют бумажные единицы измерения, такие, как "in" (дюймы) или "cm" (сантиметры). MapBasic имеет текущие единицы измерения, по умолчанию это дюймы; программа MapBasic может менять эти единицы, используя оператор **Set Paper Units**. Вы можете изменить эти настройки единиц измерения, включив подпредложение **Units** в предложение **Position**.

По умолчанию в этом операторе настройки применяются только к разделам, созданным в этом операторе. Они не действуют на существующие разделы. Разделы, используемые по умолчанию в **Create Cartographic Legend** или ранее, не действуют на разделы, созданные в этом операторе.

Предложение **Style** и ключевое слово **NoRefresh** позволяют Вам создавать собственные разделы легенды, которые не будут изменяться при обновлении легенды. Если ключевое слово **NoRefresh** используется в предложении **Style**, то таблица не проверяется на предмет используемых стилей.

Вместо этого, предложение **Style** должно содержать Ваш собственный список стилей, используемых в разделе. Для этого определяются значения **Text** и предложения **Line**, **Region** или **Symbol**.

### Смотри также:

**Create Cartographic Legend, Set Cartographic Legend, Alter Cartographic Frame, Remove Cartographic Frame**

## Оператор Add Column

### Назначение:

Добавляет новую временную колонку в открытую таблицу или обновляет уже созданную колонку данными из другой таблицы.

### Синтаксис:

```
Add Column table ( column [ datatype ] )
    {Values const [, const ... ] |
    From source-table
    Set To expression
    [ Where { dest-column = source-column | Within | Contains | Intersects } ]
    [ Dynamic ] }
```

где

*table* – имя открытой таблицы;

*column* – имя новой колонки;

*datatype* – тип данных колонки, который может быть одним из следующих:

**Char** (*width*), где *width* задает максимальное количество символов в строке,

**Float** (этот тип используется по умолчанию), **Integer**, **SmallInt**,

**Decimal**(*width*, *decimal\_places*), где *width* задает ширину поля,

а *decimal\_places* – позицию десятичной точки, **Date** или **Logical**;

*const* – константа или выражение, результат которого заносится в создаваемую колонку;

*source\_table* – имя другой открытой таблицы, таблицы-источника;

*expression* – выражение, извлекающее данные из второй таблицы для возможного обобщения;

*source\_column* – имя колонки-образца из таблицы *source\_table*, таблицы-источника;

*dest\_column* – имя колонки изменяемой таблицы (*table*).

**Dynamic** – назначает колонку динамически вычисляемой: если оператор имеет это ключевое слово, то последующие изменения данных в таблице-источнике (*source\_table*) повлекут немедленное изменение в таблице с колонкой, созданной оператором.

### Описание:

Оператор **Add Column** используется для создания новой **временной** колонки в открытой таблице MapInfo. Созданная колонка становится постоянной только после сохранения таблицы на диске. Однако если временная колонка создана на базе данных из постоянных колонок другой таблицы и Вы сохранили Рабочий Набор, то последний будет включать информацию о временной колонке, которая позволит Вам вызвать вновь таблицу с этой колонкой, загрузив Рабочий Набор. Для создания постоянной колонки в таблице используйте команды **Alter Table** и **Update**.

### Явное задание данных для новой колонки

Для этого используется предложение **Values**. После ключевого слова задайте список постоянных значений, разделенных запятыми. В следующем примере временная колонка таблицы WARDS заполняется некоторыми числами:

```
Open Table "wards"
Add Column wards(percent_dem)
Values 31,17,22,24,47,41,66,35,32,88
```

## Оператор Add Column

---

### Заполнение новой колонки данными из другой таблицы

Вместо предложения **Values** используйте предложение **From**, которое задает имя другой таблицы (*source\_table*), данные из которой будут использоваться в новой колонке. Обе таблицы должны быть открыты.

MapBasic предоставляет возможность объединить две таблицы в одну на основе совпадения значений колонок. Если предложение, начинающееся со слова **Where**, в операторе отсутствует, MapBasic сам выбирает метод объединения. Поэтому для точного задания метода объединения, задавайте его в явном виде. Вариант предложения

**Where колонка = колонка**

задает объединение двух таблиц на основе совпадения значений колонок из разных таблиц. Этот метод приемлем, если колонка Вашей таблицы имеет величины, сходные с величинами в колонке из другой таблицы (например, Вы добавляете колонку в таблицу STATES и другая таблица также имеет колонку с именами штатов).

Если обе таблицы имеют графические объекты, то предложением **Where** можно задать географическое объединение. Так, если оператор включает предложение **Where Contains**, MapInfo производит объединение, проверяя, могут ли объекты из таблицы *source\_table* содержать объекты из изменяемой таблицы.

Следующий оператор добавляет колонку в таблицу STORES; новая колонка будет содержать имена округов, которые выделяются из таблицы COUNTY:

```
Add Column
stores(county char(20)) ' новая колонка "county" в таблице STORES
From counties           ' использует данные из таблицы COUNTIES
Set to cname            ' из колонки "cname"
Where Contains          ' объединение: округа, содержащие города.
```

Метод создания колонки с ключом **Where Contains** возможен в том случае, если таблица имеет точечные объекты и вторая таблица имеет объекты, которые могут содержать эти точки.

В следующем фрагменте таблице STATES добавляется временная колонка. Значения для нее извлекаются из таблицы CITY\_1K, содержащей данные о крупных городах США. После выполнения оператора **Add Column**, каждая строка таблицы STATES будет содержать количество крупных городов в каждом штате.

```
Open Table "states" Interactive
Open Table "city_1k" Interactive
Add Column states(num_cities)
From city_1k           ' данные используются из таблицы CITY_1K
Set To Count(*)        ' количество городов в каждом штате
Where Within           ' объединение:
                        ' города, принадлежащие штату.
```

Предложение **Set To** задает правило обобщения данных функцией **Count(\*)**. Описание функций обобщения приводится ниже.

### Заполнение уже существующей колонки данными из другой таблицы

Для того, чтобы обновить содержимое уже существующей колонки вместо того, чтобы создать новую временную колонку, надо опустить параметр *datatype* и использовать предложение **From** вместо

предложения **Values**. При обновлении MapBasic будет игнорировать ключевое слово **Dynamic**.

## Обобщение данных в новой колонке

Если Вы используете предложение **From**, то Вы можете вычислить значения для новой колонки, обобщая данные из другой таблицы (*source\_table*). Для определения правила обобщения данных используется предложение **Set To**, включающее в себя функции обобщения. Следующая таблица приводит список используемых в операторе **Add Column** функций обобщения:

Функция	Что будет в новой колонке
<b>Avg</b> ( <i>col</i> )	средняя величина всех значений поля <i>col</i> из строк таблицы-источника, которые участвуют в объединении;
<b>Count</b> (*)	число строк в таблице-источнике, которые используются в объединении;
<b>Max</b> ( <i>col</i> )	наибольшая величина в колонке <i>col</i> из строк таблицы-источника, которые участвуют в объединении;
<b>Min</b> ( <i>col</i> )	наименьшая величина в колонке <i>col</i> из строк таблицы-источника, которые участвуют в объединении;
<b>Sum</b> ( <i>col</i> )	сумма значений из колонки по всем строкам таблицы-источника, которые используются в объединении;
<b>WtAvg</b> ( <i>col</i> , <i>weight_col</i> )	взвешенная средняя величина значений поля <i>col</i> из строк таблицы-источника; колонки с большим значением <i>weight_col</i> вносят больший вклад в результат;
<b>Proportion Avg</b> ( <i>col</i> )	среднее число, вычисленное с учетом перекрытия одних областей другими;
<b>Proportion Sum</b> ( <i>col</i> )	сумма, вычисленная с учетом взаимного перекрытия областей;
<b>Proportion WtAvg</b> ( <i>col</i> , <i>weight_col</i> )	взвешенная средняя величина с учетом перекрытия областей.

В основном функции обобщения работают только с неграфическими данными таблицы. Но последние три функции (**Proportion Sum**, **Proportion Avg**, **Proportion WtAvg**) выполняют вычисления с учетом взаимного расположения объектов. Это лучше проиллюстрировать примером. Допустим, Вы имеете таблицу TOWNS, содержащую границы города и демографическую информацию (например, население) для каждого города. Существует еще таблица RISK, которая содержит области. Объекты таблицы RISK представляют зоны различной степени риска, например, риска возможного затопления при разливе близлежащей реки.

Пользуясь данными этих двух таблиц, Вы хотите рассчитать численность населения, живущего в зонах риска. Если половина города перекрыта зоной риска, то будем считать, что половина населения попадает в зону риска; если треть города перекрыта зоной риска, то мы полагаем, что при наводнении опасность угрожает трети населения и т.д.

Следующий пример показывает, как с помощью функции обобщения **Proportion Sum** вычисляется новая колонка "population\_at\_risk", представляющая количество населения, подвергаемое риску:

```
Add Column Risk(population_at_risk Integer)
From towns
Set To Proportion Sum(town_pop)
Where Within
```

## Оператор Add Column

---

Для каждого города, часть которого попадает в зону возможного затопления, MapInfo вычисляет соответствующую долю населения и все это суммирует.

**Внимание:** Результат функции **Proportion Sum** достоверен с учетом допущения, что обобщаемые данные равномерно распределены по территориям. На практике часто оказывается, что данные распределяются по территории неравномерно, например, большая часть населения штата Нью-Йорк живет в городе Нью-Йорк. Чтобы добиться достоверности, разбивайте большие области на меньшие области, в которых данные распределены равномерно.

Функция **Proportion Avg** вычисляет средний результат, учитывая взаимное перекрытие объектов. Продолжая пример с риском наводнения, представим, что в таблице TOWNS содержится колонка "median\_age", которая содержит значения среднего возраста жителей каждого города. Следующий оператор вычисляет средний возраст живущих в зоне риска:

```
Add Column Risk(age Float)
From Towns
Set To Proportion Avg(median_age)
Where Within
```

Для каждой записи в таблице TOWNS MapInfo вычисляет отношение зоны риска к городской площади. В результате получается число от нуля до единицы включительно. MapInfo умножает это число на средний возраст жителей города ("median\_age") и затем суммирует результаты. Так, если средний возраст равен 50 и в зоне риска находится 10% площади города, то MapInfo добавит в общую сумму число 5, то есть 10% от числа 50.

Функция **Proportion WtAvg** работает так же, как и **Proportion Avg**, но позволяет еще и извлекать весовые коэффициенты из отдельной колонки данных.

### Использование функций Proportion... с объектами иного типа, чем "область"

Если Вы используете одну из трех функций **Proportion** и таблица *source\_column* содержит объекты типа "область", MapInfo вычисляет процент перекрытия одной области другой. Однако, если к записям таблицы-источника присоединены объекты другого графического типа, MapInfo считает каждый объект лежащим либо внутри, либо снаружи обрабатываемой области (объект считается лежащим внутри области, если его центроид лежит внутри области).

### Динамически вычисляемая колонка

Если при создании новой колонки Вы использовали ключевое слово **Dynamic**, то новая колонка будет динамически вычисляемой. То есть изменения данных в таблице-источнике (*source\_table*) повлекут немедленное изменение в таблице с колонкой, созданной оператором **Add Column...**

#### From...

Если Вы создадите динамически вычисляемую колонку и потом закроете таблицу-источник, то значения в колонке будут зафиксированы (то есть колонка больше уже не будет динамически вычисляемой). Аналогичный эффект срабатывает при географическом объединении: если была создана динамически вычисляемая колонка и Вы закрыли какую-либо карту, используемую в географическом объединении, то значения в колонке зафиксируются.

### Смотрите также:

**Alter Table, Update**



## Оператор Add Map

### Значение:

Добавляет слой в окно Карты.

### Синтаксис:

**Add Map** [ Window *window\_id* ] [ Auto ] Layer *table* [, *table* ... ] [Animate]

где

*window\_id* – идентификатор окна Карты, целое число;

*table* – имя открытой таблицы, объекты которой будут добавлены в окно Карты. Эта таблица должна иметь разрешение на присоединение графических объектов (mappable).

### Описание:

Оператор **Add Map** добавляет новый слой, который содержит данные открытой таблицы, в окно Карты. Оператор может добавить сразу несколько слоев. MapInfo после выполнения оператора автоматически перерисует картинку в окне Карты, если это не запрещено операторами **Set Event Processing Off** или **Set Map... Redraw Off**.

Идентификатор открытого окна для параметра *window\_id* можно получить с помощью функций **FrontWindow()** или **WindowID()**. Если параметр *window\_id* опущен, то слой будет добавлен в самое верхнее открытое окно.

Ключевое слово **Auto** позволяет MapInfo автоматически подобрать порядковый номер слоя. Растровые таблицы и таблицы с объектами типа "область" помещаются в самый конец списка слоев, а таблицы с точечными объектами – в начало списка. Если слово **Auto** было опущено, то слой с объектами таблицы *table* будет самым верхним некосметическим слоем в окне, другими словами, при перерисовке изображения в окне объекты этого слоя выводятся на экран последними. Вы можете изменить порядок слоев оператором **Set\_Map**.

### Добавление слоев с разными проекциями

Если добавляется слой с растровой таблицей и Карта еще не содержит растровых слоев, то Карта воспринимает координатную систему и проекцию растрового изображения. Если Карта уже имеет два или более растровых слоев, то проекция окна заменяется на проекцию того растрового изображения, которое занимает большую часть окна.

Если добавляемый слой не является растровым, то MapInfo продолжает показывать окно Карты с использованием той проекции и координатной системы, которые действовали до применения оператора **Add Map**, даже если таблица *table* имела свою собственную, другую проекцию и координатную систему. Если собственные проекции таблицы не совпадают с проекциями карты, MapInfo динамически преобразует координаты, отображая данные таблицы в слое карты. При обновлении окна Карты, содержащей такие слои, перерисовка изображения будет замедлена, так как MapInfo производит математические вычисления для проецирования в слой таблицы с собственными проекциями, отличными от имеющихся.

### Анимационные слои и перерисовка Карты

Если в операторе **Add Map** присутствует слово **Animate**, то добавляемый слой становится анимационным. Когда объект на анимационном слое перемещается, окно Карты перерисовывается очень быстро, потому что MapInfo перерисовывает только слой с анимацией.

## Оператор Add Map

---

Пример работы анимационного слоя можно найти в программе ANIMATOR.MB из комплекта поставки.

Эффект анимации полезен в приложениях, отображающих процессы реального времени, в которых объекты Карты должны часто и быстро перерисовываться. Например, пусть Вы разрабатываете прикладную систему управления группой грузовых автомобилей, в которой каждый грузовик представлен точечным объектом. Информацию о положении грузовика Вы получаете с помощью устройства спутникового позиционирования GPS (Global Positioning Satellite), и эта информация должна незамедлительно отражаться в окне Карты. В задачах подобного типа, когда объекты на Карте постоянно перемещаются, их лучше размещать на анимационном слое, а не на обычном.

Следующие операторы открывают таблицу (ГРУЗОВИК) и делают соответствующий слой анимационным:

```
Open Table "грузовик" Interactive
Add Map Layer грузовик Animate
```

Если с помощью оператора **Add Map** задано несколько слоев со атрибутом **Animate**, то только первый такой слой становится анимационным, а остальные слои добавляются как обычные.

Чтобы прекратить анимацию, примените оператор **Remove Map ... Layer Animate**.

Анимационные слои подчиняются специальным ограничениям. Например, пользователь не может применить инструмент Информация к объекту анимационного слоя. Каждое окно Карты может иметь только один анимационный слой. Более подробные сведения об анимационных слоях Вы можете найти в главе 8 *Руководства пользователя MapBasic*.

### Пример:

```
Open Table "world"
Map From world
Open Table "cust1992" As customers
Open Table "lead1992" As leads
Add Map Auto Layer customers, leads
```

### Смотрите также:

**Map, Remove Map Layer, Set Map**

## Функция **AEExecute( )**

### Назначение:

Посылает событие DoScript программе-серверу Apple Event.

### Предупреждение:

Использование этой функции возможно только в MapInfo для Macintosh.

### Синтаксис:

**AEExecute( *channel*, *command* )**

где

*channel* – номер канала Apple Event, предварительно открытого функцией **AEInitiate( )**, целое число;

*command* – строковая величина для отправки в качестве события DoScript.

### Величина, полученная в результате:

Величина типа Integer. Целое число, являющееся стандартным для Apple Event кодом ошибки.

Функция возвращает 0 (ноль) в случае удачного завершения.

### Описание:

После того как, Вы выполнили функцию **AEInitiate( )** и открыли канал Apple Event, Вы можете посылать с помощью функции **AEExecute( )** событие DoScript.

Более подробно об Apple Events можно прочитать в файле ..\MAPBASIC\DOC\MAC.DOC, появляющемся после установки пакета MapBasic.

### Смотрите также:

**AEInitiate( )**

## Функция **AEInitiate( )**

---

### Функция **AEInitiate( )**

#### Назначение:

Устанавливает дескриптор канала Apple Event, который будет использован для связи функцией **AEExecute( )**.

#### Предупреждение:

Использование этой функции возможно только в MapInfo для Macintosh.

#### Синтаксис:

**AEInitiate(channel)**

*channel* – целочисленная величина, используемая как номер канала.

#### Величина, полученная в результате:

Величина типа Integer:

0	если канал открыт;
1	если пользователь отменил PPC browser;
-x	код ошибки Apple Event.

#### Описание:

Вызов функции **AEInitiate(\_)** открывает канал связи Apple Event.

Более подробно об Apple Events можно прочитать в файле `..\MAPBASIC\DOC\MAC.DOC`, появляющемся после установки пакета MapBasic.

#### Смотрите также:

**AEExecute( )**

## Оператор **Alter Button**

### Назначение:

Управляет выбором и доступностью кнопки на инструментальных панелях.

### Синтаксис:

```
Alter Button { handler | ID button_id }  
    [ { Enable | Disable } ]  
    [ { Check | Uncheck } ]
```

где

*handler* – обработчик, который уже назначен существующей кнопке (обработчиком может быть имя процедуры MapBasic и стандартный командный код из файла MENU.DEF, например, M\_TOOLS\_RULER или M\_WINDOW\_LEGEND);

*button\_id* – идентификатор кнопки.

### Описание:

Если оператор **Alter Button** использует обработчик (например, имя процедуры), то изменения затронут все кнопки, вызывающие этот обработчик. Если используется предложение **ID**, то MapInfo изменяет только кнопку с номером *button\_id*.

Ключевое слово **Disable** делает кнопку недоступной. Кнопка закрашивается серым цветом и не реагирует на указание мышкой. Ключевое слово **Enable** возвращает кнопку в активное состояние.

Слова **Check** и **Uncheck** нажимают или отжимают кнопку типа **ToggleButton** (кнопку, для которой состояние выбора фиксируется). Ключевое слово **Check** назначает кнопке состояние нажатия; на экране эта кнопка будет зафиксирована нажатой. Например, для кнопки "Показать окно статистики":

```
Alter Button M_WINDOW_STATISTICS Check
```

**Замечание:** нажатие кнопки таким образом не означает автоматическое выполнение действий, закрепленных за кнопкой; так, например, выбор кнопки Показать/скрыть Статистику оператором **Alter Button** не открывает окно "Статистика", а только показывает эту кнопку нажатой на экране. Для выполнения действий кнопки в данном случае надо использовать соответствующие операторы, то есть открыть окно "Статистика" оператором **Open Window Statistics**.

Вы можете использовать слово **Check** для изменения состояния только для типа кнопок **ToolButton**. Однако, изменение состояния нажатия на кнопку типа **ToolButton** не то же самое, что и выбор инструмента этой кнопкой. Для выбора инструмента после изменения состояния кнопки Вы можете использовать оператор **Run Menu Command**, например:

```
Run Menu Command M_TOOLS_RULER
```

Имя кода M\_TOOLS\_RULER, использованное в этом примере, определено в файле MENU.DEF – файле стандартных определений для системы меню MapInfo.

Для того, чтобы активизировать созданный инструмент, используется оператор **Run Menu Command ID IDnum**.

### Смотрите также:

**Alter ButtonPad, Create ButtonPad, Run Menu Command**

### Оператор Alter ButtonPad

#### Назначение:

Показывает/скрывает инструментальную панель или добавляет/переопределяет в ней кнопку.

#### Синтаксис:

```
Alter ButtonPad { pad_name | ID pad_num }  
    [ Add button_definition [ button_definition ... ] ]  
    [ Remove { handler_num | ID button_id } [, ... ] ]  
    [ Title new_title ]  
    [ Width w ]  
    [ Position ( x, y ) [ Units unit_name ] ]  
    [ { Show | Hide } ]  
    [ ToolbarPosition ( row, column ) ]  
    [ { Show | Hide } ]  
    [ { Fixed | Float } ]  
    [ Destroy ]
```

где

*pad\_name* – имя панели (например, "Операции");

*pad\_num* – идентификатор инструментальной панели (1 – для панели "Операции", 2 – для панели "Пенал", 3 – для панели "Программы", 4 – "Команды", 5 – "ODBC");

*button\_id* – уникальный идентификатор для новой кнопки;

*handler\_num* – целочисленный код обработчика (например, M\_TOOLS\_RULER), определенный в файле MENU.DEF;

*new\_title* – заголовок инструментальной панели; пользователь его видит, когда панель показывается в виде вспомогательного окна;

*w* – ширина панели, измеряется количеством кнопок;

*x*, *y* – координаты верхнего левого угла панели в "бумажных" единицах;

*unit\_name* – имя "бумажной" единицы (например, "in" – дюйм, "cm" – сантиметр).

*row*, *column* – задает положение инструментальной панели, когда она находится в состоянии строки инструментов (docked) (например, 0, 0 задает расположение строки инструментов, прижатой к левому краю, и самой верхней строкой, а 0, 1 – расположение второй строкой сверху).

Каждый параметр *button\_definition* является либо ключевым словом **Separator**, либо группой предложений следующего синтаксиса:

```
{ PushButton | ToggleButton | ToolButton }  
    Calling { procedure | menu_code | OLE methodname | DDE server, topic }  
    [ ID button_id ]  
    [ Icon n [ File file_spec ] ]  
    [ Cursor n [ File file_spec ] ]  
    [ DrawMode dm_code ]  
    [ HelpMsg msg ]  
    [ ModifierKeys { On | Off } ]  
    [ { Enable | Disable } ]  
    [ { Check | Uncheck } ]
```

*procedure* – имя процедуры-обработчика, вызываемой при нажатии на кнопку.

*menu\_code* – стандартный в MapInfo командный код из файла MENU.DEF (например, M\_FILE\_OPEN); MapInfo начнет выполнение соответствующей команды при нажатии на кнопку.

*methodname* – строковая величина, задающая имя метода OLE. Синтаксис смотрите в описании оператора **Create ButtonPad**.

*server, topic* – строковая величина, задающая сервер DDE-связи и имя раздела (topic). Синтаксис смотрите в описании оператора **Create ButtonPad**.

*button\_id* – назначает уникальный номер для кнопки. В дальнейшем **этот номер** можно использовать **как**: число, используемое при вызове Справки (Tag in Help); как идентификатор кнопки в ее процедуре-обработчике, когда один и тот же обработчик вызывают несколько кнопок; как параметр в операторе **Alter Button**.

Предложение **Icon** *n* задает пиктограмму, которая будет на кнопке. Здесь *n* может быть одним из специальных кодов файла ICONS.DEF (например, MI\_ICON\_RULER). Подпредложение **File** *file\_spec* задает файл ресурсов изображений; в этом случае параметр *n* должен быть целочисленным идентификатором одного из ресурсов файла *file\_spec*.

Предложение **Cursor** *n* задает картинку указателя, которая появится после выбора кнопки. Здесь *n* может быть одним из специальных кодов из файла ICONS.DEF (например, MI\_CURSOR\_ARROW). Это предложение может входить только в описание кнопки инструмента (тип **ToolButtons**). Подпредложение **File** *file\_spec* задает имя файла ресурсов изображений; в этом случае параметр *n* должен быть целочисленным идентификатором одного из ресурсов файла *file\_spec*.

Предложение **DrawMode** *dm\_code* задает возможность инструмента рисовать (использование возможности передвигать мышку с нажатой клавишей) или только указывать (использование только возможности нажимать на клавишу мышки), при этом параметр *dm\_code* должен быть одним из специальных кодов из файла ICONS.DEF (например, DM\_CUSTOM\_LINE). Это предложение может входить в описание кнопки инструмента (тип **ToolButtons**).

*msg* – строковая величина, задает текст подсказки, появляющейся в строке сообщений при указании на кнопку, а также может задавать текст для плавающей подсказки **ToolTip**. Первая часть строки *msg* используется строкой сообщений. Если величина *msg* включает в себя литеры \n, то текст, следующий за ними, используется подсказкой **ToolTip**.

Предложение **ModifierKeys** управляет использованием клавиш SHIFT и CTRL в режиме рисования, сопровождающемся прорисовкой образа объекта ("rubber-band"), инструментом кнопки типа **Tool-Button**. По умолчанию применяется режим **Off**, не использующий клавиши SHIFT и CTRL.

### Описание:

Оператор **Alter ButtonPad** используется для изменения состояния отображения на экране инструментальных панелей и их изменения. Более подробная информация об инструментальных панелях содержится в 6 главе *Руководства пользователя MapBasic*.

Для того, чтобы показать панель, в операторе используется ключевое слово **Show**; для того, чтобы ее скрыть, – ключевое слово **Hide**. Пользователь может показывать и убирать с экрана инструментальные панели с помощью диалога команды НАСТРОЙКА > ИНСТРУМЕНТАЛЬНЫЕ ПАНЕЛИ.

Ключевые слова **Fixed** и **Float** задают состояние панели, в котором она показывается на экране: закрепленной к верхнему краю окна или плавающей в виде вспомогательного окна.

Предложение **Position** задает расположение панели на экране, если она находится в плавающем состоянии. Предложение **ToolbarPosition** задает расположение панели, прикрепленной к верхнему

краю рабочего окна.

С помощью ключевого слова **Destroy** Вы можете убрать панель насовсем, то есть имя этой панели больше не будет отображаться в списке диалога команды НАСТРОЙКА > ИНСТРУМЕНТАЛЬНЫЕ ПАНЕЛИ.

Оператор **Alter ButtonPad** может добавлять кнопки в инструментальные панели Операции и Пенал. Существуют три типа кнопок, которые можно добавить оператором:

тип **PushButton** – кнопка, нажатие на которую приводит к определенному действию, например, к открытию диалога;

тип **ToggleButton** – кнопка с фиксированным выбором, то есть после нажатия на кнопку она остается в состоянии выбора (на панели она "утоплена"), для отмены выбора надо нажать на нее снова;

тип **ToolButton** – кнопка инструмента, который пользователь может использовать только в окне Карты или Отчета.

Если Вы создаете кнопку с ключом **Disable**, новая кнопка будет недоступна для выбора (недоступная кнопка закрашивается серым). Доступной кнопку можно сделать следующим оператором **Alter Button**. Однако, если обработчиком (*handler*) кнопки определена стандартная в MapInfo команда, MapInfo автоматически изменит доступность кнопки в зависимости от возможности выполнения этой команды.

Если Вы создаете кнопку типа **ToolButton** или типа **ToggleButton** с ключом **Check**, то в начале работы с панелью кнопка уже "утоплена".

Если пользователь нажал на кнопку типа **ToolButton**, MapInfo автоматически вызывает соответствующий обработчик инструмента, который будет действовать до тех пор, пока пользователь не отменит его (например, клавишей ESC). В процедуре обработчика может быть использована функция **CommandInfo()** для определения места, на которое пользователь указал мышкой. Если две или более инструментальные кнопки вызывают один обработчик, то функция **CommandInfo()** поможет Вам определить идентификатор кнопки, вызвавший его.

### Встроенные картинки для кнопок и указателя мышки

Предложение **Icon** определяет пиктограмму, которую пользователь увидит на кнопке. Если опущено подпредложение **File**, то параметр *n* должен быть одним из кодов, имена которых определены в файле ICONS.DEF (например, MI\_ICON\_RULER).

**Совет:** MapInfo имеет много встроенных иконок для кнопок, не являющихся частью нормального пользовательского интерфейса. Просмотреть эти иконки Вы можете, запустив программу ICON-DEMO.MBX. Эта программа позволяет просмотреть иконки и скопировать код иконки в почтовый ящик Windows (clipboard) для дальнейшего использования в Вашей программе.

В Windows подпредложение **File** *file\_spec* ссылается на DLL-файл, который содержит ресурсы растровых изображений, а параметр *n* в предложении **Icon** является идентификатором **ID** одного из них. Более подробно создание новых кнопок в Windows описано в 11 главе *Руководства пользователя MapBasic*.

Для кнопок типа **ToolButton** Вы также можете определить картинку, которую примет указатель мышки в то время, пока активен инструмент, вызванный этой кнопкой. Имена возможных кодов для этого определены в файле ICONS.DEF (например, MI\_CURSOR\_CROSSHAIR или MI\_CURSOR\_ARROW).



Правила использования собственных ресурсов для указателя в предложении **Cursor** такие же, как при определении пиктограммы кнопки в предложении **Icon**. За тем исключением, что файл стандартных кодов для указателя в системе Macintosh называется CURS.

### Режимы рисования

Определение **ToolButton** может включать предложение **DrawMode**, которое задает возможность указывать, перемещая инструмент (например, рисование линии) или только одним нажатием на клавишу мыши (например, рисование точечного объекта). В следующей таблице перечислены возможные режимы рисования. Имена для режимных кодов определены в файле ICONS.DEF.

Значения <i>dm_code</i>	Описание режима
DM_CUSTOM_POINT	Инструмент может только указывать, а не рисовать.
DM_CUSTOM_LINE	При рисовании инструментом от указателя тянется прямая линия, закрепленная другим концом в точке, в которой была нажата клавиша мышки.
DM_CUSTOM_RECT	При рисовании инструментом указатель растягивает прямоугольник, закрепленный противоположным по диагонали углом в точке, в которой была нажата клавиша мышки.
DM_CUSTOM_CIRCLE	При рисовании указатель растягивает окружность.
DM_CUSTOM_ELLIPSE	При рисовании инструментом указатель растягивает эллипс. Если добавлено предложение <b>ModifierKeys</b> , то при нажатой клавише SHIFT будет растягиваться окружность.
DM_CUSTOM_POLYGON	Рисует многоугольник (полигон). Чтобы узнать, какой объект был нарисован пользователем, надо вызвать функцию: <b>Command-Info(CMD_INFO_CUSTOM_OBJ)</b> .
DM_CUSTOM_POLYLINE	Рисует ломаную (полилинию). Чтобы узнать, какой объект был нарисован пользователем, надо вызвать функцию: <b>Command-Info(CMD_INFO_CUSTOM_OBJ)</b> .

Все режимы рисования, исключая режим DM\_CUSTOM\_POINT, поддерживают режим автопрокрутки, который позволяет автоматически перемещать окно Карты или Отчета при операциях выбора и рисования с участием мыши. Отключить автопрокрутку можно оператором **Set Window**.

**Замечание:** Переключение автопрокрутки не доступно пользователю MapInfo. Его можно переключать только из приложения MapBasic. Если создаваемый инструмент имеет следующее предложение:

**Calling M\_TOOLS\_SEARCH\_POLYGON**

то инструмент может рисовать многоугольник. Когда пользователь дважды укажет мышкой для закрытия полигона, MapInfo выбирает все объекты (с доступных слоев Карты), попавшие в многоугольник. Многоугольник не сохраняется.

### Пример 1:

Следующие операторы показывают инструментальную панель "Операции" и скрывают панель

## Оператор Alter ButtonPad

---

"Пенал":

```
Alter ButtonPad "Операции" Show
Alter ButtonPad "Пенал" Hide
```

### Пример 2:

Следующий пример прикрепляет панель "Операции" к верхнему краю рабочего окна:

```
Alter ButtonPad "Операции" Fixed ToolbarPosition(0,0)
```

### Пример 3:

Теперь переведем панель "Операции" в плавающее состояние и поместим ее на расстоянии полдюйма от верхнего и левого краев экрана.

```
Alter ButtonPad "Операции" Float Position(0.5,0.5) Units "in"
```

### Пример 4:

Прикладная программа из стандартной поставки SCALEBAR содержит оператор **Alter ButtonPad**, который добавляет кнопку типа **ToolButton** на инструментальную панель "Программы" (идентифицируемая "ID 3").

```
Alter ButtonPad ID 3
Add
  Separator
  ToolButton
    Icon MI_ICON_CROSSHAIR
    HelpMsg "Рисует масштабную линейку в окне Карты\nМасштабная линейка"
    Cursor MI_CURSOR_CROSSHAIR
    DrawMode DM_CUSTOM_POINT
    Calling custom_tool_routine
  Show
```

Заметьте, что ключевое слово **Separator** вставляет пустое пространство между последней кнопкой панели "Программы" и новой "+" кнопкой.

Использование ID-номеров вместо названий панелей позволяет создавать приложения, не зависящие от языковой версии MapInfo.

### Смотрите также:

**Alter Button, ButtonPadInfo(\_), Create ButtonPad**

## Оператор Alter Cartographic Frame

### Назначение:

Оператор **Alter Cartographic Frame** изменяет положение, заголовок, подзаголовок, рамку и стиль существующего раздела легенды, созданной оператором **Create Cartographic Legend**. (Для изменения размера, позиции или заголовка окна легенды, используйте оператор **Set Window**.)

### Синтаксис:

```
Alter Cartographic Frame
[ Window legend_window_id ]
Id { frame_id }
    [ Position ( x , y ) [ Units paper_units ] ]
    [ Title [ frame_title ] [ Font... ] ]
    [ SubTitle [ frame_subtitle ] [ Font... ] ]
    [ Border Pen... ]
    [ Style [ Font... ]
        [ ID { id } Text { style_name } ] [Line Pen... | Region Pen... Brush...
        | Symbol Symbol... ] ]
    [ , ... ]
```

*legend\_window\_id* - это целочисленный идентификатор окна, который Вы можете получить при вызове функций **FrontWindow()** и **WindowId()**.

*frame\_id* - это индекс ID раздела легенды. Вы не можете использовать здесь имя слоя. Например, три раздела легенды могут иметь индексы ID 1, 2 и 3.

*frame\_title* - это строковая величина, определяющая заголовок раздела легенды.

*frame\_subtitle* - это строковая величина, определяющая подзаголовок раздела легенды.

*id* - это положение внутри списка стилей для данного раздела. В настоящее время нет функций MapBasic, которые могут дать информацию о номере стиля в разделе легенды.

*style\_name* - это строковая величина, которая отображает следующий за каждым символом для раздела с указанным индексом ID. Символ "#" будет замещаться именем слоя. Символ % будет замещаться текстом "Линия", "Точка", "Полигон", в соответствии с типом символа. Например, "% of #" будет заменено на "Region of States" для раздела легенды, соответствующей слою states.tab.

### Описание:

Если предложение **Window** не определено, MapInfo будет использовать самое верхнее окно легенды.

Предложение **Position** контролирует положение раздела в окне легенды. Верхний левый угол окна легенды имеет позицию 0, 0. Положение измеряется в "бумажных" единицах, таких как "in" (дюймы) или "cm" (сантиметры). MapBasic имеет по умолчанию установку в дюймах; программа MapBasic может поменять единицы, используя оператор **Set Paper Units**. Оператор **Alter Cartographic Legend** может изменить единицы измерения с помощью подпредложения **Units** в предложении **Position**.

Предложения **Title** и **SubTitle** вводят новый текст, новый шрифт или и то и другое.

Предложение **Style** должно содержать список определений для стилей, отображающихся в разделе. Вы можете только обновлять **Style** для собственного стиля. Вы можете обновлять **Text** для любого стиля. Нет возможности добавлять или удалять стили для любых типов разделов легенды.

### Смотрите также:

**Create Cartographic Legend, Set Cartographic Legend, Add Cartographic Frame, Remove Cartographic Frame**

# Оператор Alter Control

### Назначение:

Изменяет состояние элемента диалога, составленного приложением.

### Синтаксис:

```
Alter Control id_num  
    [ Title { title | From Variable array_name } ]  
    [ Value value ]  
    [ { Enable | Disable } ]  
    [ { Show | Hide } ]  
    [ Active ]
```

где

*id\_num* – целочисленный идентификатор одного из элементов активного диалога

в активном диалоге;

*title* – новый заголовок для элемента диалога, строковая величина;

*array\_name* – имя или массив величин, используемый для элементов типа **ListBox**, **MultiListBox**, **RadioGroup** и **PopupMenu**;

*value* – новое значение для элемента диалога.

### Предупреждение:

Вы не можете использовать этот оператор в окне MapBasic.

### Описание:

Оператор **Alter Control** изменяет атрибуты элемента активного диалога, окно которого было открыто оператором **Dialog**. Применение оператора возможно только пока диалоговое окно открыто, т. е. в специальной подпрограмме, называемой процедурой-обработчиком элемента диалога, вызов которой учитывается при создании диалога. Если на экране находятся два или более диалоговых окон, то оператор воздействует на активное окно, которое лежит поверх остальных.

Параметр *id\_num* определяет элемент диалога, который будет изменяться. Значение параметра соответствует значению одноименного параметра в операторе **Dialog** при создании элемента (предложение **ID id\_num**).

Изменение состояния и атрибутов элемента диалога производится при помощи предложений **Title**, **Value**, **Enable/Disable**, **Hide/Show**, **Active**. Оператор может использовать либо одно из этих предложений, либо одновременно несколько, либо все. То есть одновременно оператор **Alter Control** может изменить имя, значение и режим доступа элемента диалога.

Однако, не все атрибуты могут меняться для каждого типа элемента диалога. Например, для элемента **StaticText** не может быть изменен режим доступа, или присвоено значение, так как этот элемент таких атрибутов не имеет. Элемент типа **Button**, напротив, эти атрибуты имеет, поэтому они могут меняться оператором **Alter Control**.

Предложение **Title** назначает текст для большинства элементов (исключение составляют элементы типа **Picker** и **EditText**; текст элемента **EditText** определяется значением через предложение **Value**). Если Вы меняете текстовый атрибут для элементов типа **ListBox**, **MultiListBox**, **RadioGroup** или **PopupMenu**, предложение **Title From Variable** может читать новое содержимое элемента из строкового массива переменных.

Ключевое слово **Active** используется только для элемента **EditText**. Оператор **Alter Control... Active** помещает курсор в текстовое окошко элемента.

Ключевое слово **Hide** прячет элемент, оставляя пустое место в окне диалога на его месте. Показать вновь элемент можно, используя ключевое слово **Show**.

Для полной отмены выбора в списке элемента **MultiListBox** определите значение элемента (параметр *value*) равным нулю. Для того, чтобы добавить к текущему выбору в списке элемента **MultiListBox** еще одну строку, выполните оператор **Alter Control** с положительным значением, соответствующим номеру строки в списке.

### Пример:

Следующая программа создает диалог с двумя флажками и кнопками "ОК" и "Отмена" ("Cancel"). Когда диалог открывается, кнопка "ОК" не активна (окрашена серым). Кнопка становится доступной пользователю, когда он установит один или оба флажка.

```
Include "mapbasic.def"
Declare Sub Main
Declare Sub checker
Sub Main
    Dim browse_it, map_it As Logical
    Dialog
        Title "Показать файл"
        Control CheckBox
            Title "Показать в окне Списка"
            Value 0
            Calling checker
            ID 1
            Into browse_it
        Control CheckBox
            Title "Показать в окне Карты"
            Value 0
            Calling checker
            ID 2
            Into map_it
        Control CancelButton
        Control OKButton
            ID 3
            Disable
    If CommandInfo(CMD_INFO_DLG_OK) Then
        ,
        ' ... действие при нажатии кнопки "ОК"...
        ,
    End If
End Sub

Sub checker
    ' Если хотя бы один из флажков установлен,
    ' кнопка "ОК" становится активной; иначе она неактивна.
```

## Оператор Alter Control

---

```
If ReadControlValue(1) Or ReadControlValue(2) Then
    Alter Control 3 Enable
Else
    Alter Control 3 Disable
End If
End Sub
```

Смотрите также:

Dialog, Dialog Preserve, ReadControlValue( )

## Оператор Alter MapInfoDialog

### Назначение:

Делает недоступными, прячет или присваивает значение элементу стандартного диалогового окна в MapInfo.

### Предупреждение:

Идентификаторы, используемые этим оператором, могут быть изменены в будущих версиях MapInfo. В результате программы, работающие в 4 версии MapInfo, могут некорректно работать в следующих версиях.

### Синтаксис (вариант 1 - присвоение нестандартных установок):

```
Alter MapInfoDialog dialog_ID
                  Control control_ID
                  { Disable | Hide | Value new_value } [ , { Disable ... } ]
                  [ Control ... ]
```

### Синтаксис: (вариант 2 - восстановление стандартных установок)

```
Alter MapInfoDialog dialog_ID Default
```

где

*dialog\_ID* – целое число, идентификатор изменяемого диалогового окна MapInfo;

*control\_ID* – целое число от 1 и более, идентификатор изменяемого элемента диалога;

*new\_value* – новое значение элемента диалога.

### Описание:

Оператор позволяет делать недоступными, скрытыми и присваивать значения элементам стандартных диалогов MapInfo - кнопкам, флажкам и т. п. Надо отметить, что оператор **Alter MapInfoDialog** работает только со стандартными диалогами в MapInfo. Для работы с элементами диалогов, построенных оператором **Dialog**, используется оператор **Alter Control**.

### Определение идентификатора

Для определения идентификатора диалога необходимо загрузить MapInfo для Windows командой:

```
mapinfow.exe -helpdiag
```

После того, как программа MapInfo будет запущена с аргументом -helpdiag, выведите нужное диалоговое окно на экран и нажмите на кнопку Справки в нем. Обычно, эта кнопка вызывает окно Справочной системы, но в этом случае MapInfo показывает идентификатор данного диалога.

**Замечание:** Существуют разные общесистемные диалоги (такие как диалоги открытия и сохранения) в разных версиях Windows. Если Вы хотите внести изменения в общесистемный диалог, и если Ваше приложение используется в разных версиях Windows, Вам необходимо приготовить две версии оператора **Alter MapInfoDialog** - по одному для каждой версии Windows.

Каждый элемент диалога также имеет идентификатор. Например, большинство кнопок "ОК" имеют идентификатор 1, а кнопка "Отмена" ("Cancel") - идентификатор 2. Определить идентификатор элемента диалога можно только с помощью программы, не входящей в состав пакета MapInfo, такой, как программа Microsoft Spy++ из пакета компилятора С.

Изменное состояние элементов диалога и значения, присвоенные оператором **Alter MapInfoDialog**, сохраняются, только если пользователь закрыл диалог с подтверждением (кнопкой "ОК" или "Да").

## Оператор Alter MapInfoDialog

Например, Вы используете оператор **Alter MapInfoDialog** для помещения адреса в диалог "Найти", но MapInfo не выполнит поиск, пока Вы не откроете диалог и пользователь не нажмет на кнопку "ОК".

### Возможные виды изменений

Ключевое слово **Disable** используется для того, чтобы сделать элемент недоступным для пользователя (в диалоге элемент закрашивается серым цветом).

Ключевое слово **Hide** прячет элемент, делает его невидимым.

Предложение **Value** изменяет значение элемента.

Если Вы вносите изменения в общесистемный диалог (например, "Открыть таблицу"), то можете изменить выбор в поле со списком (combo box), или Вы можете изменить текст подписи (static text), кнопки или в текстовом окошке. Вы можете изменить положение элементов в диалоге "Настройка печати"; кнопкам "Книжная" и "Альбомная" соответствуют числа 1056 и 1057 соответственно. Однако, в 16-битной версии MapInfo, если пользователь подключает разные принтеры, диалог "Настройка печати" может быть изменен некорректно.

Если Вы изменяете другой диалог MapInfo, то следующий список приводит виды изменений, которые Вы можете в нем сделать:

**Кнопка, подпись (static text), текстовое окошко, окошко со списком (editable combo box):** можно менять текст, задавая новый текст параметром *new\_value*.

**Список (list box), окошко со списком (combo box):** можно менять выбор в списке, задавая номер элемента списка в параметре *new\_value*.

**Флажок:** можно устанавливать (значение *new\_value* равно 1) или сбрасывать (значение *new\_value* равно 0).

**Переключатель:** можно менять выбор кнопки, задавая номер в параметре *new\_value* 0 (не выбрана) или 1 (выбрана).

**Кнопка стиля символа:** можно менять установку стиля символа (например, используя вызов функции **MakeSymbol()**).

**Кнопка стиля линии:** можно менять установку стиля линии.

**Кнопка стиля штриха:** можно менять установку стиля заливки.

**Кнопка стиля шрифта:** можно менять установку стиля шрифта.

**Кнопка стиля оформления области (линия/штрих):** можно задать новое значение стиля линии или стиля штриха.

### Пример:

Следующий фрагмент вносит в диалог "Найти" текстовую строку с адресом ("Волхонка 13") в поле первого окна и прячет кнопку "Снова".

```
If SystemInfo(SYS_INFO_MIVERSION) = 400 Then
  Alter MapInfoDialog 2202
  Control 5 Value "Волхонка 13"
  Control 12 Hide
End If
Run Menu Command M_ANALYZE_FIND
```

Идентификатор окна "Найти" – 2202. **Control 5** ссылается на текстовое поле, в которое вводится



адрес для поиска. **Control 12** – ссылается на кнопку "Снова". Все эти номера в будущей версии MapInfo будут изменены. Поэтому используется функция **SystemInfo(\_)** для определения версии программы, в которой выполняется это приложение.

### Смотрите также:

**Alter Control, Dialog**

## Оператор Alter Menu

### Назначение:

Добавляет элемент в список одного из существующих меню в окне MapInfo или убирает существующий элемент из списка меню.

### Синтаксис (вариант 1):

```
Alter Menu { menuname | ID menu_id }  
Add menudef [, menudef... ]
```

где

*menuname* – заголовок меню;

*menu\_id* – целочисленный идентификатор меню от одного до пятнадцати, где единица представляет меню ФАЙЛ;

*menudef* – идентификатор элемента списка меню, который имеет следующий синтаксис:

```
newmenuItem  
[ ID menu_item_id ]  
[ HelpMsg help ]  
[ { Calling handler | As menuname } ]
```

где

*newmenuItem* – строковая величина, имя нового элемента списка меню;

*menu\_item\_id* – целочисленный идентификатор элемента меню, который может быть использован в дальнейшем оператором **Alter Menu Item**;

*help* – строковая величина, текст которой будет показываться в строке сообщений, когда элемент меню будет выбран (подсвечен);

*handler* – имя sub-процедуры обработчика или командный код, имена для которых определены в файле MENU.DEF (например, M\_FILE\_NEW) или же специальный код обработки события выбора меню механизмами OLE или DDE. Особенности последнего способа обработки см. в описании оператора **Create Menu**.

### Синтаксис (вариант 2):

```
Alter Menu { menuname | ID menu_id }  
Remove { handler | submenuname | ID menu_item_id }  
[, { handler | submenuname | ID menu_item_id } ... ]
```

где

*menuname* – заголовок уже определенного меню;

*menu\_id* – целочисленный идентификатор меню от 1 до 22, где единица представляет меню ФАЙЛ;

*handler* – имя sub-процедуры обработчика или код для стандартной команды MapInfo;

*submenuname* – заголовок подменю, иерархически подчиненного *menuname*;

*menu\_item\_id* – целочисленный идентификатор элемента меню, который может быть использован в дальнейшем оператором **Alter Menu Item**.

### Описание:

Оператор **Alter Menu** добавляет или убирает элемент в списке меню (или подменю) *menuname*.

Элемент меню в операторе может быть идентифицирован его именем (например, "File" или "Файл").

Набор стандартных имен зависит от языковой версии MapInfo; оператор **Alter Menu** использует имена, соответствующие языковой версии. Поэтому, если Вы планируете использовать Вашу

прикладную программу в MapInfo другой языковой версии, используйте вместо имени номер, задаваемый предложением ID. Значения параметра *menu\_id* от одного до двадцати двух соответствуют заголовкам стандартного меню MapInfo: 1 соответствует меню ФАЙЛ и т. д.

Ниже приводится таблица с именами стандартной системы меню MapInfo и соответствующими им значениями идентификатора *menu\_id*. Внимание: меню с 16 по 22 – это быстрые меню, которые появляются при нажатии на правую кнопку мыши. Быстрые меню доступны только в Windows.

Имя меню	Описание и идентификатор
"Файл"	Стандартное меню ФАЙЛ ( <b>ID 1</b> )
"Правка"	Стандартное меню ПРАВКА ( <b>ID 2</b> )
"Объекты"	Стандартное меню ОБЪЕКТЫ ( <b>ID 14</b> )
"Запрос"	Стандартное меню ЗАПРОС ( <b>ID 3</b> )
"Таблица"	Стандартное меню ТАБЛИЦА ( <b>ID 15</b> )
"Настройки"	Стандартное меню НАСТРОЙКИ ( <b>ID 5</b> )
"Окно"	Стандартное меню ОКНО ( <b>ID 6</b> )
"Справка"	Стандартное меню СПРАВКА ( <b>ID 7</b> )
"Список"	Меню СПИСОК ( <b>ID 8</b> ). Используется, когда активно окно Списка. Смотрите имя "WinSpecific".
"Карта"	Меню КАРТА ( <b>ID 9</b> ). Используется, когда активно окно Карты.
"График"	Меню ГРАФИК ( <b>ID 11</b> ). Используется, когда активно окно Графика.
"Отчет"	Меню ОТЧЕТ ( <b>ID 10</b> ). Используется, когда активно окно Отчета.
"Геогруппы"	Меню РАЙОНИРОВАНИЕ ( <b>ID 13</b> ). Используется, когда активно окно <b>Районирование</b> .
"MapBasic"	Меню MAPBASIC ( <b>ID 12</b> ). Используется, когда активно окно MapBasic.
"Программы"	Меню ПРОГРАММЫ ( <b>ID 4</b> ). Предлагается для помещения в него команд, задаваемых в программах, таких как SCALEBAR.
"WinSpecific"	Имя, обозначающее меню, соответствующее открытому окну: "Карта", "График", "Список", "Отчет", "MapBasic" или "Справка".
"Растр"	Это меню в стандартном варианте иерархически подчинено меню ТАБЛИЦА.
"Изменить"	Это меню в стандартном варианте иерархически подчинено меню ТАБЛИЦА.
"DefaultShortcut"	Стандартное быстрое меню. Это меню появляется при нажатии правой кнопки мыши на окне, не имеющем своего быстрого меню ( <b>ID 16</b> ) .
"MapperShortcut"	Быстрое меню для окна Карты ( <b>ID 17</b> ) .

## Оператор Alter Menu

---

“BrowserShortcut”	Быстрое меню для окна Списка (ID 18) .
“LayoutShortcut”	Быстрое меню для окна Отчета (ID 19) .
“GrapherShortcut”	Быстрое меню для окна Графика (ID 20) .
“CmdShortcut”	Быстрое меню для окна MapBasic (ID 21) .
“RedistrictShortcut”	Быстрое меню для окна Районирование (ID 22) .

### Пример 1:

Добавим в список меню ФАЙЛ новую команду:

```
Alter Menu "Файл" Add  
"Специальная команда" Calling sub_procedure_name
```

### Пример 2:

Следующий фрагмент делает то же самое, что и предыдущий пример, только используется идентификатор меню.

```
Alter Menu ID 1 Add  
"Специальная команда" Calling sub_procedure_name
```

### Пример 3:

Теперь добавим назначение идентификатора для команды, который позже можно будет использовать в операторе **Alter Menu Item**.

```
Alter Menu ID 1 Add  
"Специальная команда" ID 300 Calling sub_procedure_name
```

### Пример 4:

Следующий пример удаляет определенный элемент списка, который был создан ранее.

```
Alter Menu ID 1 Remove sub_procedure_name
```

### Пример 5:

Программа TEXTBOX использует оператор **Create Menu** для создания меню “Рамка,” и оператор **Alter Menu** для добавления его в качестве иерархически подчиненного в меню “Программы”:

```
Alter Menu "Программы" Add  
" (-",  
"Рамка" As "Рамка"
```

### Пример 6:

В следующем примере добавляется новая команда в быстрое меню для окна Карты (это меню появляется при нажатии на вторую кнопку мыши).

```
Alter Menu ID 17 Add  
"Поиск ближайшего города" Calling sub_procedure_name
```

### Смотрите также:

**Alter Menu Bar, Alter Menu Item, Create Menu, Create Menu Bar**

## Оператор Alter Menu Bar

### Назначение:

Добавляет или удаляет заголовки меню в строку меню окна MapInfo.

### Синтаксис:

```
Alter Menu Bar { Add | Remove }
               { menuname | ID menu_id }
               [, { menuname | ID menu_id } ... ]
```

где

*menuname* – имя меню (например, "Файл");

*menu\_id* – целочисленный идентификатор меню от 1 до 22, где единица представляет меню ФАЙЛ.

### Описание:

Оператор **Alter Menu Bar** добавляет или убирает один или более заголовков меню в строку меню. Заголовок меню – это слово, представляющее группу команд MapInfo. Указав на это слово, пользователь открывает список меню, состоящий из команд, имен подменю и элементов оформления (например, черта, разделяющая список на части).

Параметр *menuname* может быть строкой или выражением строкового типа, результатом которого является имя одного из стандартных меню в MapInfo (например, "Файл" или "Правка"). Параметр *menuname* может быть также именем меню, созданного оператором **Create Menu** (смотрите примеры).

Меню может быть также определено специальным идентификатором. Например, предложение **ID 2** задает меню ПРАВКА.

Заметим, что набор стандартных имен зависит от языковой версии; оператор **Alter Menu** использует имена, соответствующие языковой версии. Поэтому, если Вы планируете использовать Вашу прикладную программу в MapInfo другой языковой версии, используйте вместо имени номер, задаваемый предложением **ID**. Значения параметра *menu\_id* от 1 до 22 соответствуют заголовкам стандартного меню MapInfo: 1 соответствует меню ФАЙЛ и т. д.

Список имен стандартной системы меню MapInfo и соответствующих им значений идентификатора *menu\_id* приведен в разделе, описывающем оператор **Alter Menu**.

### Как добавить меню

Оператор **Alter Menu Bar Add...** добавляет заголовок меню в строку меню с правого края. Если Вам необходимо вставить меню в определенное место в строке меню, то для переопределения строки меню выполните оператор **Create Menu Bar**.

В Windows, если Вы добавите много заголовков меню в строку меню, то в ней появляется вторая строчка.

### Как убрать меню

Оператор **Alter Menu Bar Remove...** удаляет заголовок из строки меню. При этом меню не пропадает и может быть в любой момент восстановлено так, как показано ниже. Следующая пара операторов сначала удаляет меню ЗАПРОС из строки меню, а затем помещает ее снова туда крайним справа:

```
Alter Menu Bar Remove "Запрос"
Alter Menu Bar Add "Запрос"
```

## Оператор Alter Menu Bar

---

После того, как оператор **Alter Menu Bar Remove...** удалит меню, MapInfo отменяет все клавишные сокращения, ранее назначенные командам, которые находились в удаленном списке меню.

Например, если Вы удалите меню ФАЙЛ, то клавишное сокращение Ctrl+O больше не будет вызывать команду ОТКРЫТЬ ТАБЛИЦУ.

### Пример:

Создается меню ДАННЫЕ, содержащее три команды, и затем оператором **Alter Menu Bar Add** добавляется в строку заголовков меню MapInfo.

```
Declare Sub addsub
Declare Sub editsub
Declare Sub delsub
Create Menu "Данные" As
    "Добавить" Calling addsub,
    "Правка" Calling editsub,
    "Удалить" Calling delsub
' Удаляются меню Окно и Справка...
Alter Menu Bar Remove ID 6, ID 7
' Добавляется меню Данные, а затем восстанавливаются
' меню Окно и Справка
Alter Menu Bar Add "Данные", ID 6, ID 7
```

Перед тем как поместить заголовок созданного меню в строку меню, эта программа сначала удаляет меню ОКНО (идентификатор 6) и меню СПРАВКА (идентификатор 7). Затем в правый конец строки меню добавляется сначала заголовок меню ДАННЫЕ, а затем меню ОКНО и СПРАВКА. Тем, что в примере меню сначала удаляется, а потом восстанавливается, соблюдается соглашение, что меню Справочной системы всегда должно быть последним справа.

### Смотрите также:

**Alter Menu, Alter Menu Item, Create Menu, Create Menu Bar, Menu Bar Hide, Menu\_Bar Show**

## Оператор **Alter Menu Item**

### Назначение:

Изменяет состояние элемента списка меню.

### Синтаксис:

```
Alter Menu Item { handler | ID menu_item_id }
                { [ Check | Uncheck ] |
                  [ Enable | Disable ]   |
                  [ Text itemname ]     |
                  [ Calling handler | As menuname ] }
```

где

*handler* – имя sub-процедуры или код для стандартной команды MapInfo;

*menu\_item\_id* – целочисленный идентификатор элемента меню, который задается при создании списка меню (оператором **Create Menu** или **Alter Menu**);

*menuname* – заголовок списка меню;

*itemname* – новый текст для элемента меню (может содержать управляющие коды).

### Описание:

Оператор **Alter Menu Item** изменяет значения атрибутов одного или более элементов списка меню. Например, оператор может сделать команду недоступной для выбора (на экране она закрашивается серым).

Элемент меню может задаваться либо именем обработчика *handler*, который запускается при выборе элемента в списке меню, либо идентификатором в предложении **ID**. Заметим, что один и тот же обработчик могут вызывать разные элементы меню. Поэтому, если оператор **Alter Menu Item** использует имя процедуры-обработчика *handler*, то MapInfo будет менять все элементы, вызывающие этот обработчик, из всех меню. Если Вы используете предложение **ID**, то MapInfo изменит атрибуты только одного элемента меню.

Оператор **Alter Menu Item** может использовать идентификатор только для тех элементов, для которых при создании списка меню он был определен. Приложение MapBasic не может использовать идентификатор, который был задан другим приложением MapBasic.

Оператор **Alter Menu Item** позволяет вносить изменения в систему меню MapInfo. Если меню, подвергаемое изменению, уже находится в строке заголовков, то Вы можете увидеть изменения немедленно, открыв соответствующее меню.

Если задан режим **Check**, то при выборе элемента меню в списке меню напротив его имени устанавливается галочка. Это возможно, если при создании этого элемента оператором **Create Menu** он был назначен как избираемый ("checkable"). Режим **Uncheck** убирает галочку.

Режимы **Disable** и **Enable** определяют доступность выбора элемента меню. Недоступные элементы закрашиваются серым цветом. Заметим, что MapInfo автоматически делает некоторые элементы меню доступными и недоступными в соответствии с текущим состоянием в среде программы MapInfo. Например, команда **ФАЙЛ > ЗАКРЫТЬ** становится серой, если не открыто ни одной таблицы. Поэтому, приложение MapBasic не может изменять доступность стандартного элемента MapInfo.

Вы можете обращаться к инструментальным средствам как к элементам меню (например, **M\_TOOLS\_RULER** в **MENU.DEF**), но не можете сделать их недоступными с помощью оператора

## Оператор Alter Menu Item

---

### Alter Menu Item.

В предложении **Text** можно изменить имя элемента.

Предложение **Calling** задает имя процедуры-обработчика, вызываемой элементом меню. Если пользователь выберет этот элемент в меню, то MapInfo запустит на выполнение эту процедуру.

#### Пример 1:

Создается меню ДАННЫЕ, содержащее четыре команды, и затем добавляется в строку заголовков меню MapInfo.

```
Declare Sub addsub
Declare Sub editsub
Declare Sub delsub
Create Menu "Данные" As
    "Добавить" Calling addsub,
    "Правка" Calling editsub,
    "Удалить" ID 100 Calling delsub,
    "Удалить все" ID 101 Calling delsub
' Удаляется меню Справка...
Alter Menu Bar Remove ID 7
' Добавляется меню Данные, а затем восстанавливается меню Справка
Alter Menu Bar Add "Данные", ID 7
```

#### Пример 2:

Следующий оператор **Alter Menu Item** переименовывает команду ПРАВКА в команду ПРАВКА...

```
Alter Menu Item editsub Text "Правка..."
```

#### Пример 3:

Следующий оператор делает команду УДАЛИТЬ ВСЕ недоступной.

```
Alter Menu Item ID 101 Disable
```

#### Пример 4:

Следующий оператор делает недоступными две команды: УДАЛИТЬ ВСЕ и УДАЛИТЬ, так как они используют один и тот же обработчик.

```
Alter Menu Item delsub Disable
```

#### Смотрите также:

**Alter Menu, Alter Menu Bar, Create Menu**



## Оператор Alter Object

### Назначение:

Изменяет форму, положение или графический тип существующего объекта.

### Синтаксис:

```
Alter Object obj
{ Info object_info_code , new_info_value |
  Geography object_geo_code , new_geo_value |
  Node { Add [ Position polygon_num , node_num ] ( x , y ) |
        Set Position polygon_num , node_num ( x , y ) |
        Remove Position polygon_num , node_num
      }
}
```

где

*obj* – переменная типа Object;

*object\_info\_code* – целое число, код, возвращаемый функцией **ObjectInfo(\_)**

(например, OBJ\_INFO\_PEN);

*new\_info\_value* – новое значение для кода *object\_info\_code* (например, новая величина типа Pen);

*object\_geo\_code* – целое число, код, возвращаемый функцией **ObjectGeography(\_)**;

(например, OBJ\_GEO\_POINTX);

*new\_geo\_value* – новое значение для кода *object\_geo\_code* (например, новая X-координата);

*polygon\_num* – короткое целое число, идентификатор для одного полигона в объекте регион (область);

*node\_num* – короткое целое число, идентификатор для одного узла в полилинии или полигоне;

*x*, *y* – X- и Y-координаты узла.

### Описание:

Оператор **Alter Object** изменяет форму, местоположение, графический стиль существующего объекта. Эффект действия оператора **Alter Object** зависит от того, какое предложение используется в конструкции оператора: **Info**, **Node**, или **Geography**.

Если оператор использует предложение **Info**, то MapBasic изменяет графический стиль оформления объекта (например, стиль линии и штриха). Если оператор использует предложение **Node**, то MapBasic добавляет, удаляет или передвигает узлы объекта типа "полилиния" или "область". Если оператор включает в себя предложение **Geography**, то MapBasic изменяет географические атрибуты всех объектов, не являющихся полилиниями и областями (например, X- или Y-координата точечного объекта).

### Предложение Info

Оператор **Alter Object** с предложением **Info** изменяет стиль оформления объекта (например, стиль линии и штриха). В предложении изменяются атрибуты, значения которых можно получить от функции **ObjectInfo(\_)**. Например, Вы можете определить текущий стиль штриха (величину типа Brush), вызвав функцию **ObjectInfo(\_)**:

```
Dim b_fillstyle As Brush
b_fillstyle = ObjectInfo(Selection.obj, OBJ_INFO_BRUSH)
```

И, наоборот, следующий оператор **Alter Object** восстанавливает прежнее значение стиля штриховки:

## Оператор Alter Object

```
Alter Object obj_variable_name  
Info OBJ_INFO_BRUSH, b_fillstyle
```

Заметьте, что Вы используете один и тот же код (OBJ\_INFO\_BRUSH) в функции **ObjectInfo()** и в операторе **Alter Object**.

В следующей таблице в первой колонке приводятся имена кодов для использования в предложении **Info** в качестве параметра *obj\_info\_code*. Имена присвоены целочисленным кодам для удобства использования их в операторе. Эти определения находятся в файле MAPBASIC.DEF и, если Вы хотите использовать имена, включите в начало Вашей программы оператор **Include "MAPBASIC.DEF"**.

Значения <i>obj-info-code</i>	Результат выполнения Alter Object
OBJ_INFO_PEN	Изменяется стиль линии или контура. Параметр <i>new-info-value</i> должен иметь значение типа Pen.
OBJ_INFO_BRUSH	Изменяется стиль штриховки объекта. Параметр <i>new-info-value</i> должен иметь значение типа Brush.
OBJ_INFO_TEXTFONT	Изменяется стиль шрифта. Параметр <i>new-info-value</i> должен иметь значение типа Font.
OBJ_INFO_SYMBOL	Изменяется стиль символа. Параметр <i>new-info-value</i> должен иметь значение типа Symbol.
OBJ_INFO_SMOOTH	Изменяется режим сглаживания углов для полилиний. Параметр <i>new-info-value</i> должен иметь значение логического типа (TRUE или FALSE).
OBJ_INFO_FRAMEWIN	Меняет содержимое рамки на изображение другого окна. Параметр <i>new-info-value</i> должен быть целочисленным идентификатором окна.
OBJ_INFO_FRAMETITLE	Заменяет заголовок в рамке. Параметр <i>new-info-value</i> должен иметь значение типа String.
OBJ_INFO_TEXTSTRING	Меняет текст в текстовом объекте. Параметр <i>new-info-value</i> должен иметь значение типа String.
OBJ_INFO_TEXTSPACING	Изменяет расстояние между строками в текстовом объекте. Параметр <i>new-info-value</i> должен иметь значение типа Float равным 1, 1.5, или 2.
OBJ_INFO_TEXTJUSTIFY	Изменяет значение выравнивания для текстовых объектов. Параметр <i>new-info-value</i> должен иметь одно из следующих целочисленных значений: 0 – выравнивание влево, 1 – центрирование, 2 – выравнивание вправо.

OBJ\_INFO\_TEXTARROW

Изменяет вид указки для текстового объекта. Параметр *new-info-value* должен иметь одно из следующих целочисленных значений:  
 0 – нет указки,  
 1 – только линия,  
 2 – линия со стрелкой.

## Предложение Geography

Оператор **Alter Object** с предложением **Geography** изменяет расположение объекта. Предложение **Geography** действительно для всех типов объектов за исключением полилиний и областей. Для изменения расположения объектов последних двух типов используйте предложение **Node**, которое описано ниже. В предложении **Geography** изменяются атрибуты, текущие значения которых можно получить от функции **ObjectGeography()**. Например, Вы можете получить координаты конца объекта "линия":

```
Dim o_cable As Object
Dim x, y As Float
x = ObjectGeography(o_cable, OBJ_GEO_LINEENDX)
y = ObjectGeography(o_cable, OBJ_GEO_LINEENDY)
```

Оператор **Alter Object** изменяет координаты концов линейного объекта:

```
Alter Object o_cable
  Geography OBJ_GEO_LINEENDX, x
Alter Object o_cable
  Geography OBJ_GEO_LINEENDY, y
```

Заметим, что используется один и тот же код (OBJ\_GEO\_LINEENDX) в функции **ObjectGeography()** и в операторе **Alter Object**.

В следующей таблице в первой колонке приводятся имена кодов для использования в предложении **Geography** в качестве параметра *obj\_geo\_code*. Имена присвоены целочисленным кодам для удобства использования их в операторе. Эти определения находятся в файле MAPBASIC.DEF и, если Вы хотите использовать имена, включите в начало Вашей программы оператор **Include "MAPBASIC.DEF"**.

Значения <i>obj_geo_code</i>	Результат выполнения Alter Object
OBJ_GEO_MINX	Изменяет X-координату верхнего левого угла минимального прямоугольного покрытия (МПП).
OBJ_GEO_MINY	Изменяет Y-координату верхнего левого угла МПП.
OBJ_GEO_MAXX	Изменяет X-координату нижнего правого угла МПП.
OBJ_GEO_MAXY	Изменяет Y-координату нижнего правого угла МПП.
OBJ_GEO_ARCBEGANGLE	Изменяет начальный угол дуги.
OBJ_GEO_ARCENDANGLE	Изменяет конечный угол дуги.
OBJ_GEO_LINEBEGX	Изменяет X-координату начальной точки линии.
OBJ_GEO_LINEBEGY	Изменяет Y-координату начальной точки линии.

## Оператор Alter Object

OBJ_GEO_LINEENDX	Изменяет X-координату конечной точки линии.
OBJ_GEO_LINEENDY	Изменяет Y-координату конечной точки линии.
OBJ_GEO_POINTX	Изменяет X-координату точечного объекта.
OBJ_GEO_POINTY	Изменяет Y-координату точечного объекта.
OBJ_GEO_ROUNDRAIDUS	Изменяет радиус закругления для объекта типа "скругленный прямоугольник".
OBJ_GEO_TEXTLINEX	Изменяет координату по оси X конца текстовой строки объекта.
OBJ_GEO_TEXTLINEY	Изменяет координату по оси Y конца текстовой строки объекта.
OBJ_GEO_TEXTANGLE	Изменяет угол поворота текстового объекта

### Предложение Node

Оператор **Alter Object** с предложением **Node** добавляет, перемещает или убирает узлы полилиний и многоугольников (полигонов), составляющих области.

**Add** используется для создания в объекте нового узла. **Remove Position** удаляет заданный узел. **Set Position** меняет местоположение узла.

Обычно оператор **Alter Object** с предложением **Node** используется в связке с операторами **Create Pline** или **Create Region** (создание полилинии и полигона). Операторы **Create** позволяют создать объекты полилинии или полигона из образца. Операторы **Create** требуют точного задания количества узлов объекта на этапе компиляции; но в ряде случаев Вы не можете знать, сколько узлов будет содержать объект при работе программы.

Если Ваша программа не знает точное количество узлов, Вы можете, используя операторы **Create Region** или **Create PLine**, создать "пустые" объекты (т. е. количество узлов объекта будет нулевым). Когда понадобится, Вы с помощью оператора **Alter Object... Node Add** можете добавить к созданному объекту любое количество узлов.

Параметры *polygon\_num* и *node\_num* могут принимать значения от 1 и более. Параметр *polygon\_num* определяет номер изменяемого многоугольника в области. Параметр *node\_num* – номер узла в многоугольнике и в ломаной (полилинии). Если объект является полилинией, то параметр *polygon\_num* должен быть всегда единицей.

### Пример:

```
Dim myobj As Object, i As Integer
Create Region Into Variable myobj 0
For i = 1 to 10
    Alter Object myobj
        Node Add (Rnd(1) * 100, Rnd(1) * 100)
Next
```

### Объекты Группа точек и Коллекция

Оператор **Alter Object** расширен поддержкой следующих новых типов объектов.

Группа точек: устанавливает символ группы точек , как показано ниже:

## Alter Object *obj\_variable\_mpoint*

**Info** OBJ\_INFO\_SYMBOL, *NewSymbol*

Коллекция: Используя оператор Alter Object с предложением Info, можно переустановить части коллекции (регион, полилиния или группа точек) внутри объекта "коллекция".

Предложение Info позволяет видоизменять те же атрибуты, которые Вы можете запрашивать через функцию ObjectInfo( ). Например, можно определить часть регионов объекта коллекция вызывом функции ObjectInfo( ):

```
Dim ObjRegion As Object
ObjRegion = ObjectInfo(Selection.obj, OBJ_INFO_REGION)
```

Таким образом, следующий оператор **Alter Object** позволяет переустановить часть регионов, входящих в коллекцию:

```
Alter Object obj_variable_name
Info OBJ_INFO_REGION, ObjRegion
```

Внимание: Вы используете тот же самый код ( OBJ\_INFO\_REGION) и в функции **ObjectInfo( )** и в операторе **Alter Object**.

К оператору **Alter Object** добавлена поддержка, позволяющая вставлять и удалять узлы из объектов типа "группа точек".

## Alter Object *obj Node* .

Чтобы вставить узлы в группу точек:

```
Dim mpoint_obj as object
Create Multipoint Into Variable mpoint_obj 0
Alter Object mpoint_obj Node Add (0,1)
Alter Object mpoint_obj Node Add (2,1)
```

Внимание: Узлы для группы точек всегда добавляются в конец таблицы.

Чтобы удалить узлы из группы точек:

## Alter Object *mpoint\_obj Node Remove Position polygon\_num, node\_num*

*mpoint\_obj* - объект типа "группа точек".

*polygon\_num* - игнорируется для группы точек, рекомендуется установить значение 1.

*node\_num* - число удаляемых узлов.

Чтобы установить точки внутри группы точек:

## Alter Object *mpoint\_obj Node Set Position polygon\_num, node\_num (x,y)*

*mpoint\_obj* - объект типа "группа точек".

*polygon\_num* - игнорируется для группы точек, рекомендуется установить значение 1.

*node\_num* - число узлов, которые будут изменены.

*(x,y)* - новые координаты узла *node\_num*.

В новой версии также можно определять замкнутые области, не являющиеся полигонами - для этой цели используется оператор **Objects Check**.

## Objects Check From *tablename*

[SelfInt [*Symbol Clause*] ]

[Overlap [*Pen Clause*] [*Brush Clause*] ]

[Gap Area [*Unit Units*] [*Pen Clause*] [*Brush Clause*] ] ]

## Оператор Alter Object

---

Пустоты (бреши) - это замкнутые области, которые не образуют полигонов. В таблице с границами, полигоны должны иметь общие границы. В идеальном случае в таблице не должны встречаться перекрытия полигонов и пустоты между ними. В некоторых случаях пустоты между полигонами имеют смысл и право на существование. Например, Великие озера на карте мира являются "пустотой" между Канадой и США. Однако, в большинстве случаев пустоты являются результатом плохого согласования общих границ между полигонами. Такие бреши обычно имеют малые размеры.

Чтобы успешно отделить допустимые пустоты (например, Великие озера) от заведомо ненужных брешей, используется предложение *Area*. Любые пустоты больше заданной площади будут оставляться без изменения. Единицы измерения площади *Area* задаются предложением *Units*. Если подпредложение *Units* не задано, то площадь брешей *Area* будет измеряться в текущих единицах измерения MapBasic.

Пустоты выделяются и превращаются в полигоны, линии границ и заливка которых определяются предложениями **Pen** и **Brush**, следующими за ключевым словом **Gap**. По умолчанию, эти полигоны изображаются с синей заливкой и тонкой черной границей.

### Пример

Этот пример запускает оператор **Objects Check** для таблицы "TestFile" и сохраняет результат в таблице "DumpFile". Также применяется параметр **Overlap** и изменяются стандартные стили точечных и площадных объектов.

```
objects check from TestFile into table Dumpfile Overlap
  SelfInt Symbol (67, 16711680, 28)
  Overlap Pen (1,2,0) Brush (2, 16776960,0)
  Gap 100000 Units "sq mi" Pen (1,2,0) Brush (2,255,0)
```

### Смотрите также:

Create PLine, Create Region, ObjectGeography(\_), ObjectInfo(\_)

## Оператор Alter Table

### Назначение:

Изменяет структуру открытой таблицы. Не может быть применен к связанным таблицам.

### Синтаксис:

```
Alter Table table (
    [Add columnname columnntype [, ... ] ]
    [Modify columnname columnntype [, ... ] ]
    [Drop columnname [, ... ] ]
    [Rename oldcolumnname newcolumnname [, ... ] ]
    [Order columnname, columnname [, ... ] ]
)
```

где

*table* – имя открытой таблицы;

*columnname* – имя колонки (поля) в открытой таблице, длина которой не должна превышать 31 символ и состоит из букв, цифр и символа подчеркивания и не может начинаться с цифры;

*columnntype* – тип данных колонки (поля) в таблице (включая ширину поля, если необходимо);

*oldcolumnname* – старое имя колонки (поля) для переименования;

*newcolumnname* – новое имя колонки (поля) для переименования.

### Описание:

Оператор **Alter Table** используется для изменения структуры открытой таблицы. Можно добавлять и удалять колонки, изменять у существующих колонок ширину или тип данных, переименовывать колонки и менять порядок их расположения. Надо заметить, что, если в таблице были изменения, то перед тем как выполнить оператор **Alter Table**, Вы должны сохранить их на диске или восстановить таблицу с диска, удалив изменения.

Параметр *columnntype* может иметь следующие значения:

Integer – целое число (4 байта);

SmallInt – короткое целое число (2 байта);

Float – десятичное число с плавающей точкой;

Decimal(*size, decplaces*) – десятичное число фиксированной длины (*size*) и числом знаков после запятой (*decplaces*);

Char(*size*) – строка символов, где *size* – максимальное количество символов в поле;

Date – дата;

Logical – логическая величина.

Предложение **Add** используется для добавления новой колонки в Вашу таблицу. Предложение **Modify** используется для изменения типа данных колонки. Предложение **Drop** используется для удаления колонки. Предложение **Rename** изменяет имя колонки. Предложение **Order** позволяет Вам задать свой порядок расположения колонок. Один оператор **Alter Table** одновременно может включать в себя все пять предложений, и каждое предложение может обращаться к списку колонок. Поэтому одним оператором **Alter Table** Вы можете сделать все структурные изменения в таблице (смотрите пример).

## Оператор Alter Table

---

Предложение **Order** работает только с колонками и не влияет на порядок записей в таблице. Перечислите через запятую имена полей в нужном Вам порядке. Первое имя в списке соответствует самой левой колонке в окне Списка. В таком же порядке будут расположены поля в окне "Информация" – в верхней строке будет значение из первой колонки таблицы и так далее по порядку.

Если программа применяет оператор **Alter Table** к таблице, которая имеет мемо-поля, то последние будут утеряны. Сообщений об этом не выводится.

Когда Вы производите аналогичные действия, используя интерфейс MapInfo (команда ТАБЛИЦА > ИЗМЕНИТЬ > ПЕРЕСТРОИТЬ), то программа предупредит о потере мемо-полей. Когда программа MapBasic меняет структуру таблицы, то предупреждений не будет.

Оператор **Alter Table** может послужить причиной удаления слоев из окна Карты, а следовательно, привести к потере объектов тематических слоев или объектов с Косметического слоя. Если Вы используете ключевое слово **Interactive**, то MapInfo предложит пользователю сохранить тематические или/и косметические объекты.

### Пример:

Таблица GCPOR.TAB состоит из следующих колонок: "pop\_88", "metsize", "fipscod" и "utmcode".

Оператор **Alter Table** делает следующее:

- в предложении **Rename** меняет имя колонки "pop\_88" на "Население";
- в предложении **Drop** удаляет колонки "metsize", "fipscod", "utmcode";
- в предложении **Add** создает колонку "Школа" для 2-байтовых значений целого типа (SmallInt) и колонку "Субсидии" вещественного типа;
- в предложении **Order** заново определяет порядок колонок в таблице.

```
Open Table "gcpor"
Alter Table pop
(Rename pop_88 Население
 Drop metsize, fipscod, utmcode
 Add Школа SmallInt, Субсидии Float
 Order Школа, Население, Субсидии)
```

### Смотрите также:

**Add Column, Create Index, Create Map, Create Table, Drop**



## Функция ApplicationDirectory\$( )

### Назначение:

Возвращает имя каталога, в котором находится файл выполняющегося приложения.

### Синтаксис:

**ApplicationDirectory\$( )**

### Величина, полученная в результате:

Строка. Величина типа String.

### Описание:

Вызов функции **ApplicationDirectory\$( )** из приложения поможет Вам определить каталог (или папку), из которого была запущена прикладная программа. Если в данный момент никаких приложений не загружено (Вы вызвали функцию в окне MapBasic), то функцией возвращается пустая строка.

Для определения каталога, в котором установлена программа MapInfo, используйте функцию **ProgramDirectory\$( )**.

### Пример:

```
Dim sAppPath As String
sAppPath = ApplicationDirectory$( )
' С этого момента переменная sAppPath равна строке:
'
'      "C:\MAPBASIC\CODE\"
```

### Смотрите также:

**ProgramDirectory\$( )**

### Функция Area( )

#### Назначение:

Возвращает географическую площадь объекта.

#### Синтаксис:

**Area(obj\_expr, unit\_name)**

где

*obj\_expr* – выражение, результат которого есть величина типа Object;

*unit\_name* – единицы измерения площади (например, "sq km" – квадратные километры).

#### Величина, полученная в результате:

Десятичное число с плавающей запятой. Величина типа Float.

#### Описание:

Функция **Area(\_)** возвращает площадь географического объекта, определенного параметром *obj\_expr*, в единицах измерения, указанных во втором параметре. Смотрите описание оператора **Set Area Units**, в котором приведен список возможных имен единиц измерений.

Площадь имеют только области, эллипсы, прямоугольники и скругленные прямоугольники.

Результат применения функции к точечным, текстовым объектам, а также к прямой линии, дуге и полилинии будет равен нулю. Для скругленного прямоугольника функция **Area(\_)** возвращает приблизительное значение. MapBasic вычисляет площадь этого объекта так, как если бы он был обычным прямоугольником.

#### Пример 1:

В этом примере демонстрируется, как используется функция для вычисления площади географического объекта. Объект "states.obj" представляет географический объект, соответствующий текущей строке в определенной таблице.

```
Dim sq_miles As Float
Open Table "states.tab"
Fetch First From states
sq_miles = Area(states.obj, "sq mi")
```

#### Пример 2:

Вы можете также использовать функцию **Area(\_)** в операторе **Select** для формирования SQL-запроса. В следующем примере из таблицы STATES выбирается некоторая информация и помещается в таблицу запроса. Эта таблица также имеет колонку для результатов вычисления площади выбранных штатов.

```
Select state, abbr, Area(obj, "sq km")
From states
Into results
```

#### Смотрите также:

**ObjectLen(\_), Perimeter(\_), Set Area Units**

## Функция AreaOverlap( )

### Назначение:

Вычисляет площадь пересечения двух замкнутых объектов.

### Синтаксис:

**AreaOverlap** (*object1*, *object2*)

где

*object1* и *object2* – два замкнутых (имеющих площадь) объекта.

### Величина, полученная в результате:

Десятичное число с плавающей запятой. Величина типа Float.

### Описание:

Функция возвращает величину площади пересечения двух объектов *object1* и *object2* в единицах измерения площадей, текущих для выполняемой программы MapBasic.

### Смотрите также:

**Overlap( ), ProportionOverlap( ), Set Area Units**

### Функция Asc( )

#### Назначение:

Возвращает код первого символа строки.

#### Синтаксис:

**Asc(*string\_expr*)**

где

*string\_expr* – выражение, результат которого есть символьная строка.

#### Величина, полученная в результате:

Целое число. Величина типа Integer.

#### Описание:

Функция **Asc( )** возвращает код первого символа строки *string\_expr*. В Windows это ANSI-код.

Если строка пустая, то функция **Asc( )** возвращает ноль.

Операционные оболочки Windows и Macintosh работают с разными системами кодов символов.

Поэтому результат одного и того же вызова функции **Asc( )** в разных системах может различаться.

Однако обе системы имеют общие коды от 32 (пробел) до 126 (тильда). В этот диапазон входят все цифры и буквы латинского алфавита. Буквы русского алфавита в разных системах имеют разные коды.

В системах, поддерживающих двубайтовые коды символов (например, Windows Japanese): если первый символ в строке *string\_expr* однобайтовый, то функция **Asc( )** вернет код от 0 до 255; если первый символ в строке *string\_expr* двубайтовый, то функция **Asc( )** вернет код от 256 до 65535.

В системах, не поддерживающих систему двубайтовых кодов, результат функции находится в диапазоне от 0 до 255.

#### Пример:

```
Dim code As SmallInt
code = Asc("Afghanistan")
' code равно 65,
' так как 65 код символа A
```

#### Смотрите также:

**Chr\$( )**

## Функция Asin( )

### Назначение:

Возвращает арксинус.

### Синтаксис:

**Asin(*num\_expr*)**

где

*num\_expr* – численное выражение, результат которого должен находиться в диапазоне от единицы до минус единицы включительно.

### Величина, полученная в результате:

Десятичное число с плавающей запятой. Величина типа Float.

### Описание:

Функция **Acos( )** вычисляет арксинус числа, полученного в результате вычисления выражения *num\_expr*. Другими словами, **Acos( )** возвращает величину угла в радианах, синус которого равен параметру *num\_expr*.

Результатом вычисления является угол, значение которого возвращается в радианах. Диапазон значения угла находится между  $-\pi/2$  и  $\pi/2$  радиан (число  $\pi$  равно приблизительно 3.141593, и  $\pi/2$  радиан равно 90 градусам).

Для перевода градусов в радианы число необходимо умножить на число DEG\_2\_RAD. Для обратного конвертирования используется коэффициент RAD\_2\_DEG. Чтобы Ваша программа могла использовать эти коэффициенты конвертирования, она должна содержать оператор **Include "MAPBASIC.DEF"**.

### Пример:

```
Include "MAPBASIC.DEF"
Dim x, y As Float
x = 0.5
y = Asin(x) * RAD_2_DEG
' y равно 30,
' так как синус от 30 градусов равен 0.5
```

### Смотрите также:

**Acos( ), Atn( ), Cos( ), Sin( ), Tan( )**

### Функция Ask( )

#### Назначение:

Показывает диалоговое окно с сообщением или вопросом и предлагающее подтвердить или отменить предложение.

#### Синтаксис:

**Ask(prompt, ok\_text, cancel\_text)**

где

*prompt* – текст сообщения, заданный в кавычках;

*ok\_text* – текст на кнопке подтверждения (например, "Да" или "Продолжить");

*cancel\_text* – текст на кнопке отмены (например, "Отменить" или "Нет").

#### Величина, полученная в результате:

Логическая. Величина типа Logical.

#### Описание:

Функция **Ask( )** формирует и выводит на экран диалоговое окно, в котором пользователю предлагается ответить "да" или "нет". В соответствии с этим функция возвращает TRUE или FALSE.

Параметр *prompt* содержит текст сообщения или вопроса, на который надо ответить пользователю, например – "Файл уже существует. Заменить на новый?". Текст сообщения должен содержать не более 300 символов.

Диалоговое окно имеет две кнопки: одна для положительного ответа, другая для отрицательного. Параметры *ok\_text* и *cancel\_text* задают текст для этих кнопок (например, "ОК" или "Да" – для положительного ответа, "Нет" или "Стоп" – для отрицательного). Если пользователь нажмет на кнопку с надписью *ok\_text*, то функция вернет значение TRUE. Кнопка с надписью *cancel\_text* выдает значение FALSE. Нажатие на клавишу ESC равносильно выбору кнопки отмены в диалоге; функция **Ask( )** примет значение FALSE. Текст для кнопок должен быть лаконичен и сжат. Если Ваша фраза настолько велика, что не помещается в пределах стандартной кнопки, то лучше воспользуйтесь оператором **Dialog** вместо функции **Ask( )**.

Кнопка *ok\_text* выбирается по умолчанию, то есть вместо нажатия мышкой эту кнопку можно выбрать клавишей ENTER.

#### Пример:

```
Dim more As Logical  
more = Ask("Вы согласны удалить объекты?", "Да", "Нет")
```

#### Смотрите также:

**Dialog, Note( ), Print**

## Функция Atn( )

### Назначение:

Возвращает арктангенс.

### Синтаксис:

**Atn(*num\_expr*)**

где

*num\_expr* – выражение, результатом которого является число.

### Величина, полученная в результате:

Десятичное число с плавающей запятой. Величина типа Float.

### Описание:

Функция **Atn()** возвращает арктангенс от числа, полученного в результате вычисления выражения *num\_expr*. Другими словами, **Atn()** возвращает угловое значение в радианах, при котором тангенс равен числу *num\_expr*.

Результатом вычисления является угол, значение которого возвращается в радианах. Диапазон значения угла находится между  $-\pi/2$  и  $\pi/2$  радиан (число  $\pi$  равно приблизительно 3.141593, и  $\pi/2$  радиан равно 90 градусам).

Для перевода градусов в радианы число необходимо умножить на число DEG\_2\_RAD. Для обратного конвертирования используется коэффициент RAD\_2\_DEG. Чтобы Ваша программа могла использовать эти коэффициенты конвертирования, она должна содержать оператор **Include "MAPBASIC.DEF"**.

### Пример:

```
Include "MAPBASIC.DEF"
Dim val As Float
val = Atn(1) * RAD_2_DEG
' val равен 45, потому что
' арктангенс от 1 равен 45 градусам
```

### Смотрите также:

**Acos()**, **Asin()**, **Cos()**, **Sin()**, **Tan()**

### Оператор AutoLabel

#### Назначение:

Размещает подписи объектов в окне Карты на Косметическом слое.

#### Синтаксис:

```
AutoLabel
  [ Window window_id ]
  [ { Selection | Layer layer_id } ]
  [ Overlap [ { On | Off } ] ]
  [ Duplicates [ { On | Off } ] ]
```

где

*window\_id* – идентификатор окна, целое число;

*layer\_id* – имя таблицы или идентификатор слоя, целое число.

#### Описание:

Оператор **AutoLabel** создает подписи для объектов (объекты типа “текст”) в окне Карты.

Подписываются только те объекты, которые в данный момент видны в окне Карты.

**Замечание:** Подписи, созданные этим оператором, действительно просто текстовые объекты, а не динамические подписи, обновление содержания которых автоматически поддерживает MapInfo версии 4.0. Для управления показом динамических подписей используйте оператор **Set Map**.

Предложение **Window** определяет окно Карты. Если его нет, MapBasic обращается к самому верхнему окну Карты.

Предложение **Selection** определяет действие оператора на выборку, а предложение **Layer\_** – на указанный слой. Если ни слой, ни выборка не заданы, то подписываются все слои.

Предложения **Overlap** и **Duplicates** задают режимы, соответствующие выбору флажков “Подписи могут накладываться” и “Подписей может быть много” в диалоге соответствующей команды Map-Info. По умолчанию режимы имеют значение **Off**, если слова не участвуют в операторе, и значение **On**, если Вы написали слово **Overlap** или **Duplicates** без определения режима.

Оператор **AutoLabel** использует текущие установки шрифта и расположение подписей.

Пользователь может изменять эти установки в диалоге команды КАРТА > УПРАВЛЕНИЕ СЛОЯМИ.

Изменить установки шрифта и расположение подписей из приложения MapBasic можно, используя оператор **Set Map**.

#### Пример:

```
Open Table "world" Interactive
Open Table "worldcap" Interactive
Map From world, worldcap
AutoLabel
  Window FrontWindow( )
  Layer world
```

#### Смотрите также:

**Set Map**



## Оператор Beep

### Назначение:

Подает звуковой сигнал.

### Синтаксис:

**Beep**

### Описание:

Оператор **Beep** посылает команду на динамик Вашего компьютера для подачи звука.

# Оператор Browse

### Назначение:

Открывает окно Списка.

### Синтаксис:

```
Browse expression_list From table  
    [ Position ( x , y ) [ Units paperunits ] ]  
    [ Width window_width [ Units unitname ] ]  
    [ Height window_height [ Units unitname ] ]  
    [ Row n ]  
    [ Column n ]  
    [ Min | Max ]
```

где

*expression\_list* – выражения, задающие через запятую колонки, или звездочка (\*);

*table* – имя открытой таблицы;

*unitname* – строковая величина, задающая единицу измерения листа или "бумажные" единицы (например, "mm" – миллиметры);

*x*, *y* – координаты верхнего левого угла окна Списка в "бумажных" единицах;

*window\_width* и *window\_height* – определяют размер окна Списка в "бумажных" единицах;

*n* – положительное целое число.

### Описание:

Оператор **Browse** открывает новое окно Списка для открытой таблицы.

Список может показать все поля таблицы, если параметр *expression\_list* равен звездочке, или же вид окна Списка может быть определен списком выражений. Выражением может быть имя колонки, оператор, функция, число, которое определяет одну колонку окна Списка. Имена колонок, показываемые в самой верхней строке Списка, полностью зависят от параметра *expression\_list*. Если, например, Вы определили вычисляемую колонку выражением вроде **НАСЕЛЕНИЕ** / **area(obj, "acre")**, то оно и будет именем колонки. Смотрите пример ниже.

Предложение **Position** задает расположение окна на экране. Координаты *x* и *y* определяют верхний левый угол окна Списка относительно верхнего левого угла окна MapInfo. Предложения **Width** и **Height** определяют ширину и высоту окна Списка. Если предложений **Width** и **Height** нет в операторе, MapInfo самостоятельно определит размер следующим образом: площадь окна Списка приблизительно равна четверти рабочего окна MapInfo, так, чтобы строки и колонки были показаны в окне полностью.

Параметры *x*, *y*, *window\_width*, *window\_height* задаются в единицах, определенных в предложениях **Units**. Если это предложение в какой-либо из трех конструкций опущено, соответственные параметры будут пониматься в "бумажных" единицах, определенных в Вашей программе (смотрите описание оператора **Set Paper Units**).

Если оператор **Browse** включает в себя ключевое слово **Max**, то окно будет открыто полностью на всю рабочую область окна MapInfo. Аналогично, если оператор **Browse** включает в себя ключевое слово **Min**, то окно будет свернуто в икону. Но последнее не будет возможно, если приложение запустить в MapInfo для Macintosh.

Предложение **Row** используется для определения, какая строка будет самой верхней в окне Списка.

По умолчанию это будет первая строка.

Предложение **Column** используется для определения, какая колонка будет самой левой в окне Списка. По умолчанию это будет первая колонка.

### Пример 1:

Этот пример демонстрирует, как открытую таблицу WORLD показать в окне Списка.

```
Open Table "world.tab"  
Browse * From world
```

### Пример 2:

Данные из той же таблицы можно показать в окне Списка по-другому, явно задавая колонки Списка и используя выражения для них.

```
Open Table "world"  
Browse  
  страна,  
  население,  
  население/area(obj, "sq km") "Плотность"  
From world
```

Результатом будет окно Списка из трех колонок. Первые две колонки содержат данные так, как они хранятся в файле таблицы WORLD. Третья колонка является вычисляемой, и ей присваивается псевдоним ("Плотность"), который пользователь увидит в окне в строке заголовков колонок.

### Смотрите также:

**Set Browse, Set Window**

### Предложение Brush

#### Назначение:

Задаёт стиль штриховки графических объектов.

#### Синтаксис:

**Brush** *brush\_expr*

где

*brush\_expr* – выражение, результат которого есть величина типа Brush. Например, вызов функции **MakeBrush**(*pattern*, *fgcolor*, *bgcolor*).

#### Описание:

Предложение **Brush** не является отдельным оператором, а входит в состав операторов, в которых необходимо задавать стиль штриха для следующих объектов: многоугольник (полигон), область (регион), прямоугольник и эллипс. В понятие стиля входят установки цвета штриха и фона, типа штриха.

Предложение **Brush**, например, используется в операторе **Create Ellipse**, который создаёт новый объект типа "эллипс". Предложение **Brush** задаёт стиль штриховки объекта. Если оператор не использует это предложение, то будет использована текущая настройка этого стиля в MapInfo.

Параметр *brush\_expr* должен быть величиной типа Brush и может задаваться переменной, например:

**Brush** *br\_var*

Или параметр может задаваться значением, например, полученным вызовом функций **CurrentBrush**(\_) или **MakeBrush**(\_):

**Brush** **MakeBrush**(64, CYAN, BLUE)

В некоторых операторах (таких как **Set Map**) предложением **Brush** стиль задаётся непосредственно набором из трёх целочисленных параметров (*pattern*, *foreground\_color*, *background\_color*), заключённым в скобки, например:

**Brush**(64, CYAN, BLUE)

Некоторые операторы MapBasic используют выражения типа стиля штриха в качестве параметра (например, переменная типа Brush) не используя при этом предложения **Brush**. Одним из примеров является оператор **Alter Object**.

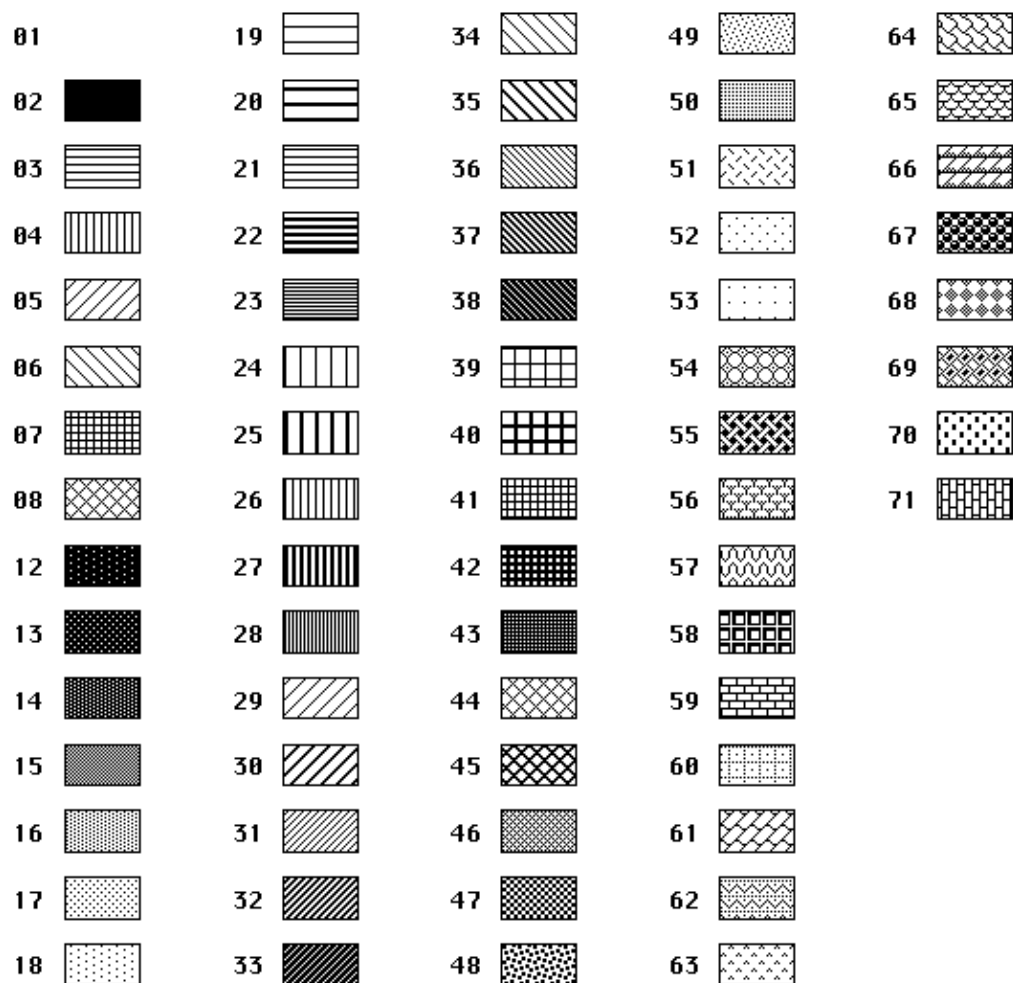
В таблице приводится определение компонент стиля штриховки замкнутых объектов:

Компонента стиля	Описание
pattern	Целое число от 1 до 8 и от 12 до 71. Задаёт рисунок штриха. Смотрите следующую таблицу.

foreground color	<p>Целочисленный код цвета для рисунка штриха. Может быть заменен вызовом функции <b>RGB()</b>. Вы также можете использовать имена для кодов цвета (как в примере выше), если в тексте Вашей программы есть ссылка на файл стандартных определений MAPBASIC.DEF. В нем определены имена для следующих цветов:</p> <p>BLACK — черный; WHITE — белый; RED — красный; GREEN — зеленый; BLUE — синий; CYAN — голубой; MAGENTA — фиолетовый; YELLOW — желтый.</p>
background color	<p>Целочисленный код цвета фона штриха. Может быть заменен вызовом функции <b>RGB()</b>. Вы также можете использовать имена для кодов цвета (как в примере выше), если в тексте Вашей программы есть ссылка на файл стандартных определений MAPBASIC.DEF.</p> <p>Заметим, что для значений параметра <i>pattern</i>, задающего вид штриха, равных 1 (прозрачный) или 2 (ровная заливка), параметр <i>background_color</i> не влияет на штриховку. Для задания прозрачного фона штриховки в предложении <b>Brush</b> следует опускать третий параметр (например, функция <b>Brush(5,_BLUE)</b> закрасит объект синими диагональными полосками с прозрачным фоном).</p> <p>Соответственно, в диалоге стиля флажок рядом с цветом фона будет опущен. Для задания прозрачного фона штриховки функцией <b>Make-Brush()</b> третий параметр должен иметь значение минус единица.</p>

Ниже приводится таблица кодов и соответствующих им рисунков штриховок. Заметим, что для штриха номер 1 (прозрачный) или 2 (ровная заливка), параметр *background\_color* не влияет на заливку.

## Предложение Brush



Смотрите также:

`CurrentBrush()`, `MakeBrush()`, `Pen`, `Font`, `Symbol`

## Функция **Buffer( )**

### Назначение:

Возвращает объект типа "область", представляющий собой буферную зону вокруг выбранного объекта (область, граница которой отстоит от границы объекта на заданное расстояние).

### Синтаксис:

**Buffer(*inputobject*, *resolution*, *width*, *unit\_name*)**

где

*inputobject* – объектное выражение (выражение, результат которого есть величина типа Object);

*resolution* – число узлов многоугольника, принимаемого как окружность (число типа SmallInt);

*width* – радиус буфера, число типа Float;

*unit\_name* – имя единицы измерения расстояний (например, "mi" – миля, "km" – километр).

### Величина, полученная в результате:

Область. Величина типа Object.

### Описание:

Функция **Buffer( )** возвращает буферную зону (объект типа "полигон") вокруг объекта, заданного параметром *inputobject*.

Функция **Buffer( )** может создавать буферные зоны только вокруг одного объекта. Если Вам необходимо создать буфер вокруг группы объектов, используйте оператор **Create Object As Buffer**.

Если значение параметра *width* отрицательно и объект *inputobject* является замкнутым, то буфер будет меньшим по площади по сравнению с *inputobject*.

### Пример:

Следующий фрагмент программы создает объект "прямая линия". Затем создается буферная зона вокруг линии.

```
Dim o_line, o_region As Object
o_line = CreateLine(-73.5, 42.5, -73.6, 42.8)
o_region = Buffer( o_line, 20, 10, "mi")
```

### Смотрите также:

**Create Object**

## Функция ButtonPadInfo( )

### Функция ButtonPadInfo( )

#### Назначение:

Возвращает информацию о состоянии инструментальной панели

#### Синтаксис:

**ButtonPadInfo(*pad\_name*, *attribute*)**

где

*pad\_name* – строковая величина, представляющая имя инструментальной панели; например, “Операции”, “Пенал”, “Программы” или “Команды” для стандартных панелей или то имя, которое было определено при создании новых панелей.

*attribute* – целочисленный код, управляющий типом результата функции.

#### Величина, полученная в результате:

Зависит от значения параметра *attribute*.

#### Описание:

В зависимости от значения *attribute* функция **ButtonPadInfo( )** вернет следующую информацию об инструментальной панели *pad\_name*:

#### Значения *attribute*

BTNPAD\_INFO\_  
FLOATING

BTNPAD\_INFO\_NBTNS  
BTNPAD\_INFO\_WIDTH

BTNPAD\_INFO\_WINID  
BTNPAD\_INFO\_X

BTNPAD\_INFO\_Y

#### Результат ButtonPadInfo( )

Логическая величина (Logical). Возвращается значение TRUE, если панель плавающая, и значение FALSE, если панель находится в прикрепленном состоянии, то есть вытянута вдоль верхнего края рабочего окна.

Целое число типа SmallInt, количество кнопок на панели.

Целое число типа SmallInt, ширина панели, причем единица измерения равна одной кнопке (не включая разделитель).

Целое число типа Integer, идентификатор панели.

X-координата верхнего левого угла инструментальной панели. Если панель находится в прикрепленном состоянии, то возвращается целое число от 0 и более, если панель плавающая, то число типа Float (при этом значение представляется в “бумажных” единицах).

Y-координата верхнего левого угла инструментальной панели. Смотрите описание предыдущего кода.

Вы можете использовать приведенные имена кодов, если в тексте Вашей программы есть ссылка на файл стандартных определений MAPBASIC.DEF.

#### Пример:

```
Include "mapbasic.def"
If ButtonPadInfo("Операции", BTNPAD_INFO_FLOATING) Then
    '...если панель Операции плавающая,
    '    то следующий оператор прикрепит ее.
    Alter ButtonPad "Операции" ToolbarPosition(0,0) Fixed
End If
```

#### Смотрите также:

**Alter ButtonPad**



## Оператор Call

### Назначение:

Вызывает sub-процедуру или внешнюю процедуру (DLL, XCMD или RPC).

### Предупреждение:

Вы не можете использовать этот оператор в окне MapBasic.

### Синтаксис:

**Call subproc [ ( [parameter] [, ... ] ) ]**

где

*subproc* – имя подпрограммы;

*parameter* – параметр, передающий подпрограмме значение из основного модуля.

### Описание:

Оператор **Call** используется для вызова подпрограммы. Подпрограммой может быть процедура, написанная на MapBasic, первым оператором которой является оператор **Sub**. Так же с помощью оператора **Call** в Windows может быть вызвана процедура из динамической библиотеки (DLL). Аналогично, прикладная программа в MapInfo для Macintosh может вызывать внешние XCMD-команды. Смотрите описание оператора **Declare Sub** для подробной информации о внешних процедурах.

Если вызывается sub-процедура, MapBasic начинает выполнять операторы подпрограммы, пока не встретит оператор **End Sub** или **Exit Sub**. Выполнение возвращается в программу, откуда был произведен вызов, к следующему оператору после **Call**. Оператор **Call** может обращаться только к тем процедурам, которые являются частью текста программы.

Программа MapBasic должна в главной процедуре содержать оператор **Declare Sub**, объявляющий имя процедуры и список ее параметров перед ее вызовом. Это требование должно выполняться как по отношению к процедурам, организованным оператором **Sub**, так и к внешним процедурам (DLL и XCMD).

### Передача значений параметров

Подпрограмма может не иметь параметров, тогда вызов такой процедуры может выглядеть так:

**Call subroutine**

или

**Call subroutine(\_)**

Если sub-процедура имеет параметры, то они могут быть объявлены двумя способами: ссылкой ("by reference") или значением ("by value"). По умолчанию параметр объявляется как "ссылка". Если параметр подпрограммы был определен таким образом, то при вызове sub-процедуры соответствующим параметром вызова должна быть переменная. Такой параметр не только передает значение переменной в процедуру, но и возвращает значение обратно. В процедуре значение параметра может быть изменено, и это измененное значение будет присвоено переменной, выступавшей в роли параметра вызова, после того, как оператор **Call** вернет управление в вызывавшую программу.

Объявление величины параметра как значения осуществляется при помощи ключевого слова **ByVal** перед именем параметра в списке операторов **Sub** и **Declare Sub**. Если один из параметров

## Оператор Call

---

подпрограммы был определен таким образом, то при вызове sub-процедуры соответствующим параметром вызова может быть как постоянная величина, так и переменная или выражение. Однако в этом случае новое значение параметра не может быть возвращено обратно той же переменной.

Sub-процедуры могут пересылать как простые параметры, так и массивы переменных. При этом в списке вызова Вы пишете только имя массива без скобок.

### Вызов внешних процедур

Если оператор **Call** вызвал процедуру DLL, то MapBasic выполняет процедуру до тех пор, пока она не вернет управление. Сама процедура находится в отдельном файле (например, "KERNEL.EXE"). Этот файл с внешней процедурой должен быть доступен в то время, как MapBasic выполняет внешний вызов.

Аналогично, если оператор **Call** вызывает XCMD, то файл, содержащий XCMD, должен быть доступным.

В вызове XCMD-команд не могут участвовать массивы переменных и переменные сложных типов, составленных оператором **Type**, в качестве параметров.

### Пример:

Подпрограмма **Cube** вычисляет куб (третью степень) числа. Она имеет два параметра: первый содержит само число, второй – результат.

```
Declare Sub Cube(ByVal original As Float, cubed As Float)
Dim x, result As Float
Call Cube( 2, result)
' переменная result равна 8 (2 x 2 x 2)
x = 1
Call Cube( x + 2, result)
' переменная result равна 27 (3 x 3 x 3)
End Program

Sub Cube (ByVal original As Float, cubed As Float)
    ' Параметр "original" – число,
    ' параметр "cubed" – результат вычисления.
    cubed = original ^ 3
End Sub
```

### Смотрите также:

**Declare Sub, Exit Sub, Global, Sub... End Sub**

## Функция **CartesianArea( )**

### Назначение:

Возвращает площадь, используя вычисления в системе координат Широта/Долгота. Используется декартовый алгоритм вычислений.

### Синтаксис:

**CartesianArea( *expr*, *unit\_name* )**

*unit\_name* это строковая величина, имя единиц измерения площади. (sq km")

### Возвращаемое значение:

Величина с плавающей запятой

### Описание:

Функция **CartesianArea( )** возвращает площадь географического объекта, указанного выражением *obj\_expr*.

Функция возвращает измеренную площадь, в единицах, определенных параметром *unit\_name*; например, чтобы получить площадь в акрах, укажите "acre" в качестве параметра *unit\_name*. Смотрите описание оператора **Set Area Units**, где описаны возможные единицы измерения.

Функция **CartesianArea( )** всегда будет возвращать значение площади, рассчитанное по декартовым алгоритмам. Величина -1 будет возвращаться для данных в плановых координатах, поскольку такие данные не могут быть конвертированы в Широту/Долготу. Возвращаются приближенные результаты в том случае, если используется скругленный прямоугольник. MapBasic рассчитывает площадь скругленного прямоугольника как если бы объект был настоящим прямоугольником.

Только полигоны, эллипсы, прямоугольники и скругленные прямоугольники имеют площади. По определению, значение функции **CartesianArea( )** для точки, дуги, текста, линии или полилинии это ноль. Функция **CartesianArea( )** возвращает приблизительные результаты, когда применяется к скругленным прямоугольникам. MapBasic вычисляет площадь скругленного прямоугольника как если бы объект был настоящим прямоугольником.

### Примеры:

Следующие примеры показывают, как функция **CartesianArea( )** может вычислять площадь одиночного картографического объекта. Обратите внимание, что выражение *tablename.obj* (как в *states.obj*) представляет географический объект текущей строки в указанной таблице.

```
Dim f_sq_miles As Float
Open Table "states"
Fetch First From states
f_sq_miles = CartesianArea(states.obj, "sq mi")
```

Вы можете также использовать функцию **CartesianArea( )** внутри оператора SQL Select как показано в следующем примере.

```
Select lakes, CartesianArea(obj, "sq km")
From lakes Into results
```

### Смотрите также:

Функция **Area ( )**, функция **SphericalArea ( )**

## Функция **CartesianBuffer( )**

---

### Функция **CartesianBuffer( )**

#### Назначение:

Возвращает объект типа полигон, представляющий буферную зону (площадь внутри указанного расстояния от существующего объекта).

#### Синтаксис:

**CartesianBuffer ( *inputobject*, *resolution*, *width*, *unit\_name* )**

*inputobject* это выражение объекта

*resolution* это короткое целое, представляющее число узлов для круга в каждом углу

*width* это величина с плавающей запятой, представляющая радиус буфера; если ширина отрицательна, и если входящий объект является закрытым, то возвращаемый объект будет по размерам меньше исходного

*unit\_name* это имя единиц измерения расстояния (например, "mi" для миль, "km" для километров) используемых для измерения ширины

#### Возвращаемое значение:

Объект типа полигон

#### Описание:

Функция **CartesianBuffer( )** возвращает полигон, представляющий буферную зону.

Функция **CartesianBuffer( )** оперирует с одним объектом одновременно. Для создания буфера вокруг ряда объектов, используйте оператор **Create Object As Buffer**.

Функция **CartesianBuffer( )** будет рассчитывать буферную зону, в предположении, что объект спроецирован на плоскость и используя ширину *width* для расчета декартового расстояния буферной зоны вокруг объекта. Если *inputobject* в проекции Широта/Долгота, то сферические вычисления будут использоваться независимо от того, какая функция, связанная с буфером будет применяться. Если *inputobject* в плановых координатах, то будут использоваться декартовы вычисления, независимо от того, какая буферная функция вызывается.

#### Пример :

Следующая программа создает линейный объект, затем создает буфер вокруг него. Буферная зона занимает 10 миль во всех направлениях вокруг линии.

```
Dim o_line, o_region As Object
o_line = CreateLine(-73.5, 42.5, -73.6, 42.8)
o_region = CartesianBuffer( o_line, 20, 10, "mi")
```

#### Смотрите также:

**Функция Buffer ( ). Создание объектов. Функция SphericalBuffer ( )**

## Функция CartesianDistance ( )

### Назначение:

Возвращает расстояние между двумя точками.

### Синтаксис:

**CartesianDistance ( *x1* , *y1* , *x2* , *y2* , *unit\_name* )**

*x1* и *x2* это x-координаты (долгота)

*y1* и *y2* это y-координаты (широта)

*unit\_name* это строковая величина, соответствующая имени единиц измерения расстояния (например, "km")

### Возвращаемое значение :

Вещественное

### Описание:

Функция **CartesianDistance( )** вычисляет расстояние между двумя точками.

Функция возвращает измеренное расстояние в единицах, указанных параметром *unit\_name*; например, что бы получить расстояние в милях, укажите "mi" как параметр *unit\_name* . Смотрите оператор **Set Distance Units**, там полный список возможных единиц измерения.

Функция **CartesianDistance( )** всегда возвращает значение, используя для расчетов декартовый алгоритм. Будет возвращено значение -1 для данных в системе координат Широта/Долгота, поскольку система Широта/Долгота не проективная и не декартовая.

Параметры x- и y-координат должны использовать текущую систему координат MapBasic. По умолчанию, MapInfo использует координатную систему долгота/широта. Вы можете поменять систему координат MapBasic используя оператор **Set CoordSys**.

Если текущая система координат географическая, **CartesianDistance( )** возвращает расстояние между двумя точками, измеренное по большой дуге. Расстояние по большой дуге это кратчайшее расстояние на сфере между двумя точками.

Если текущая система координат плановая, то **CartesianDistance( )** возвращает декартовое расстояние.

### Пример:

```
Dim dist, start_x, start_y, end_x, end_y As Float
Open Table "cities"
Fetch First From cities
start_x = CentroidX(cities.obj)
start_y = CentroidY(cities.obj)
Fetch Next From cities
end_x = CentroidX(cities.obj)
end_y = CentroidY(cities.obj)
dist = CartesianDistance(start_x,start_y,end_x,end_y,"mi")
```

### Смотрите также:

Математические функции,  
Функция CartesianDistance ( ), Функция Distance ( )

## Функция **CartesianObjectLen()**

---

### Функция **CartesianObjectLen()**

#### Назначение:

Возвращает географическую длину линии или полилинии.

#### Синтаксис:

**CartesianObjectLen( *expr* , *unit\_name* )**

*obj\_expr* выражение объекта

*unit\_name* это строковая величина, представляющая имя единиц измерения расстояния (**например**, "km")

#### Возвращаемое значение:

Вещественное

#### Описание:

Функция **CartesianObjectLen()** возвращает длину объекта. Обратите внимание, что только объекты типа линия и полилиния имеют длину большую чем ноль; для измерения периметра полигонов, эллипсов и прямоугольников, используйте функцию **Perimeter()**.

Функция **CartesianObjectLen()** всегда будет возвращать значение, используя декартовый алгоритм. Величина -1 будет возвращаться для данных в системе Широта/Долгота, поскольку Широта/Долгота не проективна и не декартова.

Функция **CartesianObjectLen()** возвращает длину, измеренную в единицах длины, определенных параметром *unit\_name*; например, для получения длины в милях, укажите "mi" как параметр *unit\_name*. Смотрите описание оператора **Set Distance Units** там есть список возможных типов единиц измерения длины.

#### Пример:

```
Dim geogr_length As Float
Open Table "streets"
Fetch First From streets
geogr_length = CartesianObjectLen(streets.obj, "mi")
' geogr_length now represents the length of the
' street segment, in miles
```

#### Смотрите также:

Запросы к объектам карты, функция **SphericalObjectLen()**, функция **CartesianObjectLen()**, функция **ObjectLen()**

## Функция **CartesianPerimeter( )**

### Назначение

Возвращает периметр графических объектов.

### Синтаксис

**CartesianPerimeter( *obj\_expr* , *unit\_name* )**

*obj\_expr* это выражение для объекта

*unit\_name* это строковая величина - имя единицы измерения расстояния (например, "km")

### Возвращаемое значение

Вещественное

### Описание

Функция **CartesianPerimeter( )** вычисляет периметр объекта *obj\_expr* object. Функция **Perimeter( )** определена для следующих типов объектов: эллипсов, прямоугольников, скругленных прямоугольников и полигонов. Другие типы объектов имеют периметр равный нулю.

Функция **CartesianPerimeter( )** всегда будет возвращать значение, вычисленное по декартовому алгоритму. Величина -1 будет возвращаться для данных в системе Широта/Долгота, поскольку Широта/Долгота не проективна и не декартова.

Возвращаемое значение длины периметра осуществляется в единицах длины, определенных параметром *unit\_name*; например, для получения длины в милях, укажите "mi" в качестве параметра *unit\_name*. Смотрите описание оператора **Set Distance Units** там есть полный список возможных единиц измерения длины.

Приближенный результат возвращается при измерении периметра скругленного прямоугольника. MapBasic вычисляет периметр скругленного прямоугольника как если бы он не был скругленным.

### Пример

Следующий пример показывает как Вы можете использовать функцию **CartesianPerimeter( )** для определения периметра географического объекта.

```
Dim perim As Float
Open Table "world"
Fetch First From world
perim = CartesianPerimeter(world.obj, "km")
' The variable perim now contains the perimeter of the polygon that's
attached to
' the first record in the World table.
```

Вы можете также использовать функцию **CartesianPerimeter( )** внутри оператора **SQL Select**.

Следующий оператор **Select** выбирает информацию из таблицы States, и хранит результаты во временной таблице, называемой **Results**. Поскольку оператор **Select** включает функцию **CartesianPerimeter( )**, таблица Results будет включать колонку, показывающую периметр каждого штата.

```
Open Table "states"
Select state, CartesianPerimeter(obj, "mi")
From states
Into results
```

### Смотрите также:

Функция **CartesianPerimeter( )**, функция **SphericalPerimeter( )**, функция **Perimeter( )**

### Функция Centroid( )

#### Назначение:

Возвращает центральную точку объекта (центроид).

#### Синтаксис:

**Centroid(*obj\_expr*)**

где

*obj\_expr* – объектное выражение

#### Величина, полученная в результате:

Точка. Величина типа Object.

#### Описание:

Функция **Centroid()** возвращает точечный объект, расположенный в центре объекта, представленного параметром *obj\_expr*.

**Замечание:** В MapInfo области не всегда имеют центроид в центре объекта. Вы всегда можете передвинуть центроид области в нужное место на изменяемом слое Карты в режиме изменения формы. От расположения центроида области зависит расположение подписей, полученных в результате автоподписывания, точек геокодирования и расположения графиков и диаграмм на тематическом слое Карты.

Если объект *obj\_expr* является точечным, то функция его и вернет.

Если объект – линия, то в результате получится точка между ее концами. Если параметр *obj\_expr* представляет объект типа "полилиния", то **Centroid()** возвращает середину среднего сегмента полилинии.

Для других типов объектов функция возвращает точку с координатами действительного центроида. Если объект прямоугольник, дуга, текст или эллипс, то точка будет равноудалена между верхней и нижней границами, и между правой и левой границами. Для объекта типа "область" центроид всегда лежит внутри объекта, но не обязательно находится в геометрическом центре объекта.

#### Пример :

```
Dim pos As Object
Open Table "world"
Fetch First From world
pos = Centroid(world.obj)
```

#### Смотрите также:

**CentroidX( ), CentroidY( )**



## Функция CentroidX( )

### Назначение:

Возвращает координату центральной точки (центроида) по оси X.

### Синтаксис:

**CentroidX(*obj\_expr*)**

где

*obj\_expr* – объектное выражение.

### Величина, полученная в результате:

Вещественное число. Величина типа Float.

### Описание:

Функция **CentroidX( )** возвращает X-координату (или долготу) центральной точки объекта – точечного объекта, который можно получить, используя функцию **Centroid( )**.

Координаты возвращаются относительно действующей координатной системы, которая была назначена до этого оператором **Set CoordSys**. По умолчанию MapBasic использует систему широта/долгота.

### Пример 1:

Функция **CentroidX( )** возвращает долготу одного географического объекта.

```
Dim x As Float
Open Table "world.tab"
Fetch First From world
x = CentroidX(world.obj)
Note "Долгота: " + x + " град."
```

### Пример 2:

В этом примере продемонстрировано использование функций **CentroidX( )** и **CentroidY( )** в операторе **Select** при формировании SQL-запроса. В результате нижеприведенных операторов получается временная таблица из трех колонок, где первая содержит названия стран, а следующие две содержат X- и Y-координаты (долготу и широту) центроида каждой страны.

```
Open Table "world"
Select country, CentroidX(obj), CentroidY(obj)
From world Into results
```

### Смотрите также:

**Centroid( ), CentroidY( ), Set CoordSys**

### Функция CentroidY( )

#### Назначение:

Возвращает координату центральной точки (центроида) по оси Y.

#### Синтаксис:

**CentroidY(*obj\_expr*)**

где

*obj\_expr* – объектное выражение.

#### Величина, полученная в результате:

Вещественное число. Величина типа Float.

#### Описание:

Функция **CentroidY( )** возвращает Y-координату (или широту) центральной точки объекта – точечного объекта, который можно получить, используя функцию **Centroid( )**.

Координаты возвращаются относительно действующей координатной системы, которая была назначена перед этим оператором **Set CoordSys**. По умолчанию MapBasic использует систему широта/долгота.

#### Пример:

Функция **CentroidY( )** возвращает широту одного географического объекта.

```
Dim y As Float
Open Table "world"
Fetch First From world
y = CentroidY(world.obj)
```

#### Смотрите также:

**Centroid( ), CentroidX( ), Set CoordSys**

## Предложение CharSet

### значение:

Определяет набор кодов, используемый в MapBasic для интерпретирования символов.

### Синтаксис:

**CharSet** *char\_set*

где

*char\_set* – строковая величина, содержащая имя кода, например, "ANSI".

### Описание:

Предложение **CharSet** используется для перекодировки символов в операциях чтения из файла или записи в файл или таблицы. Предложение не является самостоятельным оператором и входит в состав таких операторов, работающих с файлами, как **Export** и **Open File**.

### Для чего нужна система кодов?

Каждому символу клавиатуры соответствует свой численный код. Например, буква латинского алфавита "А" имеет код 65. Системой кодов называют набор соответствий кодов символам, используемый компьютером для представления этих символов в текстовых данных.

Существуют несколько систем кодов, используемых в разных операционных системах. Например, Windows версии для Северной Америки и Западной Европы использует код 176 для представления знака градуса (°), тогда как Windows другой языковой версии может использовать этот код для другого символа. Тем более различаются системы кодов в операционных системах разных компьютерных платформ. Знак градуса в системе Macintosh имеет код 161.

Большинство программ DOS используют расширенную систему кодов ASCII для представления символов. В среде Windows используется ANSI-стандарт. Одним из следствий существования нескольких наборов символов является необходимость перекодировки данных при переносе их с одной вычислительной платформы на другую.

Для определения системы кодов символов используйте вызов функции **System-Info(SYS\_INFO\_CHARSET)**.

### Как кодировка символов влияет на программу MapBasic?

В большинстве кодировок часть символов представлена одинаковыми числами, например, символ английской буквы А имеет один и тот же код 65 в средах Windows, UNIX и Macintosh. Начиная с 126 символа размещаются символы различных языков.

Если Ваш файл включает текст, в котором используются символы, имеющие коды от 32 (пробел) до 126 (тильда), то Вам не надо беспокоиться о конфликтах, которые могут произойти из-за некорректности перевода данных.

Даже если Вы использовали символы из интернациональной клавиатуры, но не выходите за пределы своей операционной среды (например, Вы работаете только в Windows), то тоже нет причин для беспокойства.

Однако, если файл включает интернациональные символы и Вы хотите его перенести в другую операционную среду, которая использует другую кодировку, Вам не обойтись без предложения **CharSet** для сохранения целостности Ваших данных.

## Предложение CharSet

---

Например, приложение MapBasic, выполняющееся в MapInfo для Windows, может использовать операторы **Open File** и **Line Input** для открытия файла и чтения из него текста. Если этот текстовый файл был создан в операционной среде Macintosh, то оператору **Open File... For Input** необходимо иметь предложение **CharSet** для того, чтобы текст был правильно прочитан:

```
Open File "cogo23.txt" For INPUT As #2 CharSet "MacRoman"
```

Аналогично, если программа записывает в этот файл, оператор **Open File... For Output** должен иметь такое же предложение **CharSet**, чтобы текстовый файл можно было прочитать в Macintosh.

Предложение **CharSet** имеет только один строковый параметр, возможные значения которого приведены в следующей таблице:

Character Set	Комментарии
"Neutral"	нет преобразования кодов
"ISO8859_1"	ISO 8859-1 (Unix)
"ISO8859_2"	ISO 8859-2 (Unix)
"ISO8859_3"	ISO 8859-3 (Unix)
"ISO8859_4"	ISO 8859-4 (Unix)
"ISO8859_5"	ISO 8859-5 (Unix)
"ISO8859_6"	ISO 8859-6 (Unix)
"ISO8859_7"	ISO 8859-7 (Unix)
"ISO8859_8"	ISO 8859-8 (Unix)
"ISO8859_9"	ISO 8859-9 (Unix)
"PackedEUCJapanese"	Unix, японский стандарт
"WindowsLatin1"	Windows, стандарт США и Западной Европы
"WindowsLatin2"	Windows, стандарт Восточной Европы
"WindowsArabic"	Windows, стандарт арабских языков
"WindowsCyrillic"	Windows, стандарт русского языка
"WindowsGreek"	Windows, стандарт греческого языка
"WindowsHebrew"	Windows, стандарт иврита
"WindowsTurkish"	Windows, стандарт турецкого языка
"WindowsTradChinese"	Windows, стандарт традиционный китайский
"WindowsSimpChinese"	Windows, стандарт упрощенный китайский
"WindowsJapanese"	Windows, стандарт японского языка
"WindowsKorean"	Windows, стандарт корейского языка

"MacRoman"	Macintosh, стандарт США и Западной Европы
"MacArabic"	Macintosh арабский
"MacGreek"	Macintosh греческий
"MacHebrew"	Macintosh-иврит
"MacCentralEuropean"	Macintosh для Центральной Европы
"MacCroatian"	Macintosh для Хорватии
"MacCyrillic"	Macintosh русский
"MacIcelandic"	Macintosh исландский
"MacThai"	Macintosh тайландский
"MacTurkish"	Macintosh турецкий
"MacTradChinese"	Macintosh китайский традиционный
"MacSimpChinese"	Macintosh китайский упрощенный
"MacJapanese"	Macintosh японский
"MacKorean"	Macintosh корейский
"CodePage437"	DOS Code Page 437 = расширенная кодировка ASCII для IBM
"CodePage850"	DOS Code Page 850 = многоязычная
"CodePage852"	DOS Code Page 852 = Восточная Европа
"CodePage855"	DOS Code Page 855 = Кириллица
"CodePage857"	
"CodePage860"	DOS Code Page 860 = Португальская
"CodePage861"	DOS Code Page 861 = Исландская
"CodePage863"	DOS Code Page 863 = Французско-Канадская
"CodePage864"	DOS Code Page 864 = Арабская
"CodePage865"	DOS Code Page 865 = Норвежская
"CodePage869"	DOS Code Page 869 = Современная Греческая
"LICS"	Кодировка Lotus версии 1,2
"LMBCS"	Кодировка Lotus версии 3,4

Заметим, что в операторе **Open Table** нет необходимости использовать предложение **CharSet**, так как файл таблицы содержит информацию о той системе, где она была создана и MapInfo автоматически подбирает необходимую кодировку для перевода.

Предложение **CharSet** может понадобиться в операторе **Commit Table... As**, если Вы хотите

## Предложение CharSet

---

сохранить данные в таблице в определенной кодировке.

### Синтаксис предложения в языке MapBasic версии 2.x

MapBasic версии 2.x располагает только тремя именами систем кодов: "XASCII", "ANSI" и "MAC". Тексты программ, написанные во времена этой версии, и которые используют эти имена систем кодов, могут быть откомпилированы и запущены в MapBasic версии 3.0. Однако, использовать имена систем кодов из версии 2.x не рекомендуется:

Предложение **CharSet "XASCII"** соответствует предложению **CharSet "CodePage437"**.

Предложение **CharSet "MAC"** соответствует предложению **CharSet "MacRoman"**.

Если Вы работаете в Windows, то предложение **CharSet "ANSI"** задает кодировку, которая используется в Windows. Если программа запускается в UNIX или Macintosh, то **CharSet "ANSI"** задает такую же кодировку, как предложение **CharSet "WindowsLatin1"**.

### Пример:

Текстовый файл PARCEL.TXT был создан в DOS, поэтому в следующем примере используется код "CodePage437".

```
Open File "parcel.txt"  
For INPUT As #1  
CharSet "CodePage437"
```

### Смотрите также:

**Commit Table, Create Table, Export, Open File, Register Table**

## Функция ChooseProjection\$()

### Назначение

Показывает диалог выбора проекции и возвращает координатную систему, выбранную пользователем.

### Синтаксис

**ChooseProjection\$( *initial\_coordsys*, *get\_bounds* )**

*initial\_coordsys* - это строковая величина из предложения **Coordsys**. Она используется для установки той координатной системы, которая первый раз выбирается в диалоге. Если *initial\_coordsys* является пустой или предложение соответствует неправильной координатной системе, то по умолчанию в первом указании координатной системы используется система широта/долгота.

*get\_bounds* - это логическая величина, которая определяет, какие границы ввести пользователю при использовании плановых координат. Если *get\_bounds* - истинно, то появляется диалог, в котором надо определить границы карты. Если это выражение ложно, то диалог не появляется и используются границы, заданные по умолчанию.

### Описание

Эта функция отображает диалог выбора проекции и возвращает выбранную систему координат в виде строковой величины. Возвращаемая строковая величина имеет тот же формат, что и предложение **CoordSys**. Используйте эту функцию, если Вы хотите позволить пользователю установить проекцию внутри Вашего приложения.

### Пример

```
Dim strNewCoordSys As String

strNewCoordSys = ChooseProjection$( "", True)
strNewCoordSys = "Set " + strNewCoordSys
Run Command strNewCoordSys
```

### Функция Chr\$( )

#### Назначение:

Возвращает символ, соответствующий заданному коду символа.

#### Синтаксис:

**Chr\$(*num\_expr*)**

где

*num\_expr* – целочисленное выражение.

#### Величина, полученная в результате:

Строка. Величина типа String.

#### Описание:

Функция **Chr\$( )** возвращает строковое значение длиной в один символ, которое соответствует коду, полученному в результате вычисления выражения *num\_expr*. Используется система кодов той операционной платформы, в которой вызвана функция. В Windows это система ANSI-кодов. Для большинства систем параметр *num\_expr* должен быть положительным целым числом от 0 до 255. В операционных платформах, поддерживающих систему двубайтовых кодов (например, Windows Japanese), параметр *num\_expr* должен быть числом от 0 до 65535.

Windows и Macintosh используют различные системы кодов, но для значений от 32 (пробел) до 126 (тильда) результат функции **Chr\$( )** будет одинаков для всех систем. Например, функция **Chr\$(34)** будет равна двойной кавычке и в Windows, и в Macintosh. Для других значений параметра *num\_expr* результат функции будет уникален для каждой системы кодов.

Если в результате *num\_expr* получается дробное число, то MapBasic округляет его до целого.

12-й символ в Windows соответствует переводу на новую страницу (form-feed). Это значение функции удобно использовать в операторе **Print** для очистки окна "Сообщения" перед выводом строки. 34-й символ в Windows соответствует двойной кавычке ("). Когда строка включает Chr\$(34), MapBasic вставит в текст кавычки.

#### Ошибки:

Функция вернет код ошибки ERR\_FCN\_ARG\_RANGE, если значение аргумента выходит за пределы, заданные при его определении.

#### Пример:

```
Dim s_letter As String * 1
s_letter = Chr$(65)
Note s_letter ' Этот пример показывает символ "A"
Note "A это сообщение " + Chr$(10) + "в две строки."
```

#### Смотрите также:

**Asc( )**



## Оператор Close All

### Назначение:

Закрывает все открытые таблицы.

### Синтаксис:

**Close All [ Interactive ]**

### Описание:

Выполняя оператор **Close All**, MapBasic закрывает все открытые таблицы, включая те, которые изменялись. При этом все изменения после закрытия теряются, и предупреждения пользователю не выводятся.

Если Вы не хотите потерять текущие изменения в таблицах, используйте слово **Interactive** для вывода на экран диалога, предлагающего пользователю сохранить или игнорировать изменения.

### Смотрите также:

**Close Table**

### Оператор Close File

#### Назначение:

Закрывает открытый файл.

#### Синтаксис:

**Close File** [**#**]*filenum*

где

*filenum* – целое число, обозначающее открытый файл.

#### Описание:

Оператор **Close File** используется для закрытия файлов, открытых оператором **Open File**.

Операторы **Open File** и **Close File** работают с файлами, но не с таблицами MapInfo. Для обращения к таблицам MapBasic имеет другие операторы; например, оператор **Open Table** используется для открытия таблицы.

#### Пример:

```
Open File "cxdata.txt" For INPUT As #1
'
' файл открыт, теперь его можно закрыть:
'
Close File #1
```

#### Смотрите также:

**Open File**

## Оператор Close Table

### Назначение:

Закрывает открытую таблицу.

### Синтаксис:

**Close Table** *table* [ **Interactive** ]

где

*table* – имя открытой таблицы.

### Описание:

Оператор **Close Table** используется для закрытия таблицы *table*. Для закрытия сразу всех открытых таблиц используется оператор **Close All**.

Когда таблица показывается в одном или нескольких окнах Графиков или Списков, то при закрытии таблицы эти окна также закрываются. При закрытии таблицы из окон Карт удаляются только соответствующие закрываемой таблице слои. Окно Карты автоматически будет закрыто только тогда, когда будет закрыта последняя, отображаемая в окне Карты, таблица.

Если применить оператор **Close Table** к связанной таблице, которая имеет несохраненные изменения, то MapInfo запомнит изменения до следующего сеанса работы.

### Сохранение изменений в таблице

Когда оператор **Close Table** применяется к таблице, в которой были произведены изменения и которые не были сохранены на диске, то при закрытии они будут утеряны. Сообщений об этом не выводится. Если Вы не хотите потерять текущие изменения в таблицах, используйте слово **Interactive** для вывода на экран диалога, предлагающего пользователю сохранить или игнорировать изменения.

Если Вы хотите гарантировать несохранение изменений в таблице, используйте оператор **RollBack** перед ее закрытием. Аналогично, если Вы хотите гарантировать сохранение изменений в таблице, выполните оператор **Commit Table** перед оператором закрытия. Для определения, есть ли в таблице изменения, не сохраненные на диске, используется функция **TableInfo(table,TAB\_INFO\_EDITED)**.

### Сохранение тематических и косметических объектов

Если Вы закрываете таблицу, графические объекты которой показываются на единственном некосметическом слое окна Карты, то будет закрыто и это окно. Чтобы сохранить объекты Косметического и/или тематического слоев из этого окна, используйте ключевое слово **Interactive**. Если MapInfo обнаружит косметические и тематические объекты, то перед тем, как закрыть окно Карты, выведет сообщения, предлагающие пользователю сохранить объекты на диске или забыть о них. Сообщений не будет выводиться, если Вы использовали ключевое слово **Interactive**, но на Карте нет тематических слоев и объектов в Косметическом слое.

### Пример:

```
Open Table "world"  
Close Table world
```

### Смотрите также:

**Close All, Commit, Open Table, RollBack, TableInfo(\_)**

# Оператор Close Window

### Назначение:

Закрывает или прячет окна.

### Синтаксис:

**Close Window** *window\_spec* [ **Interactive** ]

где

*window\_spec* – имя окна (такое как **Ruler**), код окна (такой как WIN\_RULER) или целое число, определяющее окно.

### Описание:

Оператор **Close Window** закрывает или прячет окна MapInfo.

Для закрытия документального окна (окна Карты, Списка, Графика и Отчета) необходимо задать его целым числом в параметре *window\_spec*. Это число помогут Вам определить операторы **FrontWindow()** и **WindowID()**.

Чтобы закрыть специальное окно или инструментальную панель, надо в параметре *window\_spec* ввести специальное имя окна или его код. Вы можете окно "Линейка" задать именем **Ruler** или кодом WIN\_RULER. В следующей таблице приводятся имена и коды для параметра *window\_spec*:

Имя окна	Код окна	Описание окна
MapBasic.	WIN_MAPBASIC	Окно MapBasic
Help	WIN_HELP	Окно Справочной системы.
Statistics	WIN_STATISTICS	Окно "Статистика".
Legend	WIN_LEGEND	Окно "Легенда".
Info	WIN_INFO	Окно "Информация", которое открывается инструментом Информация.
Ruler	WIN_RULER	Окно "Линейка", которое открывается инструментом Линейка.
Message	WIN_MESSAGE	Окно "Сообщение", которое открывается оператором <b>Print</b> .

### Сохранение тематических и косметических объектов

Если Вы закрываете Карту, то пользователь может захотеть сохранить объекты с Косметического или тематических слоев. Для вывода сообщений, предлагающих пользователю сохранить эти объекты перед закрытием окна Карты, в операторе **Close Window** используется ключевое слово **Interactive**.

Если ключевого слова нет в операторе, косметические и тематические объекты будут утеряны. Сообщений не будет выводиться, если Вы использовали ключевое слово **Interactive**, но на Карте нет тематических слоев и объектов в Косметическом слое.

### Пример:

**Close Window Legend**

### Смотрите также:

**Open Window, Print, Set Window**

## Функция ColumnInfo( )

### Назначение:

Возвращает информацию о колонке в открытой таблице.

### Синтаксис:

**ColumnInfo({ *tablename* | *tablenum* }, { *colomnname* | "COLn" }, *attribute*)**

где

*tablename* – имя открытой таблицы;

*tablenum* – целое число, идентифицирующее таблицу;

*colomnname* – имя колонки в этой таблице;

*n* – целое число как номер колонки в таблице;

*attribute* – код, управляющий типом результата функции.

### Величина, полученная в результате:

Зависит от значения параметра *attribute*.

### Описание:

Функция **ColumnInfo( )** возвращает информацию об одной колонке в открытой таблице.

Первый параметр функции задает таблицу (именем или идентификатором). Второй параметр определяет колонку. Параметр *attribute* должен принимать значения целочисленного кода, задающего тип возвращаемой функцией информации.

В следующей таблице в первой колонке приводятся имена кодов для использования в качестве параметра *attribute*. Имена присвоены целочисленным кодам для удобства. Эти определения находятся в файле MAPBASIC.DEF и, если Вы будете использовать имена, включите в начало Вашей программы оператор **Include "MAPBASIC.DEF"**.

#### Значения *attribute*

COL\_INFO\_NAME

COL\_INFO\_NUM

COL\_INFO\_TYPE

COL\_INFO\_WIDTH

COL\_INFO\_DECPLACES

COL\_INFO\_INDEXED

COL\_INFO\_EDITABLE.

#### Результат ColumnInfo( ):

Имя колонки (строка).

Номер колонки (короткое целое число).

Тип колонки (короткое целое число). Смотрите вторую таблицу.

Ширина символьного или десятичного поля (короткое целое число). Используется только для символьных и десятичных колонок.

Число знаков после десятичной точки в поле десятичного типа (короткое целое число).

Признак индексирования колонки (логическая величина).

Признак наличия изменений значений колонки (логическая величина)

Если функция **ColumnInfo( )** имеет значение параметра *attribute* равным

## Функция ColumnInfo( )

---

COL\_INFO\_TYPE, то возвращает одно из следующих:

Функция возвращает код	Тип колонки
COL_TYPE_CHAR	Символьный
COL_TYPE_DECIMAL	Десятичный с фиксированной запятой
COL_TYPE_FLOAT	Вещественный
COL_TYPE_INTEGER	Целочисленный (4 байт)
COL_TYPE_SMALLINT	Короткое целое число (2 байт)
COL_TYPE_DATE	Дата
COL_TYPE_LOGICAL	Логический (TRUE или FALSE)
COL_TYPE_GRAPHIC	Специальный тип колонки "Obj" (представляет графические объекты, присоединенные к таблице)

### Ошибки:

Функция вернет следующие коды ошибок:

ERR\_TABLE\_NOT\_FOUND, если не найдена данная таблица,

ERR\_FCN\_ARG\_RANGE, если значение аргумента выходит за пределы, заданные при его определении.

### Пример:

```
Include "MAPBASIC.DEF"  
Dim s_col_name As String, i_col_type As SmallInt  
Open Table "world"  
s_col_name = ColumnInfo("world","col1",COL_INFO_NAME)  
i_col_type = ColumnInfo("world","col1",COL_INFO_TYPE)
```

### Смотрите также:

**NumCols( ), TableInfo( )**

### Функция **Combine( )**

#### Назначение:

Возвращает либо объект типа "область", либо "полилиния", представляющий объединение двух объектов.

#### Синтаксис:

**Combine(*object1,object2*)**

где

*object1, object2* – два объектных выражения, результатом вычисления которых должны быть либо два объекта, имеющих площадь (например, область и окружность), либо два линейных объекта (например, прямая линия и полилиния).

#### Величина, полученная в результате:

Область или полилиния. Величина типа Object.

#### Описание:

Функция **Combine( )** возвращает объект, являющийся результатом географического объединения двух объектов. Объединение двух смежных объектов удаляет границу между ними.

Результат функции **Combine( )** такой же как при выполнении географического объединения командой ОБЪЕКТЫ > КОМБИНАЦИЯ, за тем исключением, что функция создает новый объект, не меняя объектов *object1* и *object2*. Кроме того, функция **Combine( )** объединяет только объекты без данных.

Объект, полученный в результате выполнения функции **Combine( )**, оформляется так же, как объект *object1*.

#### Смотрите также:

**Objects Combine**

### Функция **CommandInfo( )**

#### Назначение:

Возвращает информацию о последних внутрисистемных событиях.

#### Синтаксис:

**CommandInfo(*attribute* )**

где

*attribute* – целочисленный код типа информации, которая будет возвращена.

#### Величина, полученная в результате:

Целое число или число с плавающей запятой, или строка, или логическая величина.

Величина типа Integer, или типа Float, или типа String, или типа Logical.

#### Описание:

Функция **CommandInfo( )** возвращает информацию о последних событиях в MapInfo. Например, какие изменения произошли во временной таблице "Selection", в какое место окна указал пользователь, использовал ли он только мышь или нажал на кнопку мышки в комбинации с клавишей SHIFT и так далее. Тип информации задается целочисленным кодом в параметре *attribute*. Для описания кодов будем использовать имена, которые назначены этим постоянным величинам в файле определений MAPBASIC.DEF. Если Вы хотите использовать имена кодов в Вашей программе, то она должна иметь строчку **Include "MAPBASIC.DEF"**.

#### Вызов после закрытия диалога, построенного приложением

Вызывая функцию **CommandInfo( )** сразу после закрытия окна диалога, Вы можете использовать в качестве параметра *attribute* один из следующих кодов:

Значение <i>attribute</i>	Результат <b>CommandInfo(attribute)</b>
CMD_INFO_DLG_OK	Величина типа Logical: TRUE - если в диалоговом окне была выбрана кнопка <b>OkButton</b> ; FALSE - если пользователь отменил диалог, нажав на кнопку <b>CancelButton</b> или на клавишу ESC. Этот код в функции предназначен только для информации о диалоговых окнах, вызываемых оператором <b>Dialog</b> .
CMD_INFO_STATUS	Величина типа Logical: TRUE – если пользователь позволил процессу, снабженному диалогом со шкалой (смотрите оператор <b>ProgressBar</b> ), завершиться самостоятельно; FALSE – если пользователь нажал на кнопку в этом диалоге, прекращающую выполнение.

#### Вызов из обработчиков новых меню и диалогов

Если приложение строит свои меню с элементами, вызывающими обработчик, или строит диалог, элементы которого вызывают процедуры-обработчики, то в этих процедурах Вы можете использовать функцию **CommandInfo( )** со следующими кодами:

Значение <i>attribute</i>	Результат <b>CommandInfo(attribute)</b>
---------------------------	---



CMD_INFO_MENUITEM	Целое число (Integer), представляющее идентификатор элемента меню, который запустил этот обработчик. Этот код используется только внутри процедуры-обработчика элемента меню.
CMD_INFO_DLG_DBL	Величина типа Logical: TRUE - если пользователь использовал при указании на элемент списка элемент <b>ListBox</b> или <b>MultiListBox</b> двойное нажатие на клавишу мышки. Этот код используется только внутри процедуры-обработчика элемента диалога, построенного приложением MapBasic.

#### Вызов из обработчиков системных событий

Если Ваше приложение имеет процедуры-обработчики системных событий (например, такую как **SelChangedHandler**), то в этих процедурах Вы можете использовать функцию **CommandInfo( )** со следующими кодами:

Значение <i>attribute</i>	Результат <b>CommandInfo(attribute)</b>
Из процедуры <b>Sel-ChangedHandler</b> :	
CMD_INFO_SELTYPE	1 – если строка была добавлена в выборку; 2 – если строка была исключена из предыдущей выборки; 3 – если все строки в таблице выбраны; 4 – если все строки таблицы были исключены из предыдущей выборки.
CMD_INFO_ROWID	Целое число (Integer) – номер строки, которая была добавлена в выбор или исключена из выбора (работает только, если одна строка была выбрана или исключена из выбора).
CMD_INFO_INTERRUPT	Логическая величина (Logical). TRUE – если пользователь прервал выбор клавишей ESC, FALSE – иначе.
Из процедуры <b>RemoteMsgHandler</b> или <b>RemoteQueryHandler( )</b> :	
CMD_INFO_MSG	Строка, представляющая собой “выполняемое” сообщение или имя элемента послания MapInfo от программы-клиента. Детали смотрите в описании процедур <b>RemoteMsgHandler</b> и <b>RemoteQueryHandler( )</b> .
Из процедуры <b>Win-ChangedHandler</b> или <b>Win-ClosedHandler</b> :	

## Функция **CommandInfo( )**

CMD_INFO_WIN	Целое число (Integer), идентификатор окна, в котором было изменение или которое было закрыто. Детали смотрите в описании процедур <b>WinChangedHandler</b> и <b>WinClosedHandler</b> .
Из процедуры <b>Foreground-TaskSwitchHandler</b> :	
CMD_INFO_TASK_SWITCH	Целое число (Integer), указывающее, что MapInfo либо только что стало активным приложением, либо только что перестало быть активным. Результатом может быть один из кодов: SWITCHING_INTO_MAPINFO (если MapInfo получает фокус). SWITCHING_OUT_OF_MAPINFO (если MapInfo теряет фокус).

### Вызов после операции поиска

После выполнения оператора **Find**, параметр функции *attribute* может принимать одно из следующих значений:

Значение <i>attribute</i>	Результат <b>CommandInfo(attribute)</b>
CMD_INFO_FIND_RC	Целое число (Integer), код успеха поиска.
CMD_INFO_FIND_ROWID	Целое число (Integer), номер строки, которой соответствует найденный объект.
CMD_INFO_X или CMD_INFO_Y	Действительное число, координата (по оси X или Y) найденного места.

### Вызов из процедуры-обработчика инструмента

Внутри процедуры-обработчика инструмента ToolButton Вы можете использовать следующие коды:

Значение <i>attribute</i>	Результат <b>CommandInfo(attribute)</b>
CMD_INFO_X	X-координата точки, где была нажата кнопка мыши: на Карте, долгота в установленных единицах и в соответствии с текущей в MapBasic системой координат; в Списке, номер колонки в окне (единица соответствует самой левой колонке в окне); в Отчете, расстояние между левым краем Отчета и указанной точкой в текущих "бумажных" единицах (ноль соответствует точке на левом крае листа Отчета).

CMD_INFO_Y	Y-координата точки, где была нажата кнопка мыши: на Карте, широта в установленных единицах и в соответствии с текущей в MapBasic системой координат; в Списке, номер строки в окне (единица соответствует самой верхней строке в окне); в Отчете, расстояние между верхним краем Отчета и указанной точкой в текущих "бумажных" единицах (ноль соответствует точке на верхнем крае листа Отчета);
CMD_INFO_X2	X-координата точки, в которой пользователь отпустил кнопку мыши. Использование этого кода возможно, если инструменту был назначен режим рисования (например, использовался код DM_CUSTOM_LINE).
CMD_INFO_Y2	Y-координата точки, в которой пользователь отпустил кнопку мыши.
CMD_INFO_SHIFT	Логическая величина: TRUE (логическое "Да"), если пользователь при указании нажал на клавишу SHIFT, FALSE (логическое "Нет"), если эта клавиша при указании не была нажата.
CMD_INFO_CTRL	Логическая величина: TRUE (логическое "Да"), если пользователь при указании нажал на клавишу CTRL, FALSE (логическое "Нет"), если эта клавиша при указании не была нажата.
CMD_INFO_TOOLBTN	Целое число (Integer), идентификатор кнопки инструмента, который вызвал этот обработчик.
CMD_INFO_CUSTOM_OBJ	Объектная величина: полилиния или многоугольник, нарисованный пользователем. Применяется с режимами рисования DM_CUSTOM_POLYLINE или DM_CUSTOM_POLYGON.

#### После вызова Macintosh XCMD

В среде MapInfo для Macintosh Вы можете обратиться к Macintosh XCMD. После вызова XCMD, Вы можете использовать функцию **CommandInfo( )** со следующими кодами:

Значение <i>attribute</i>	Результат <b>CommandInfo(attribute)</b>
CMD_INFO_XCMD	Строка, показывающая, произошла ли ошибка во время выполнения команды XCMD. Более подробно о XCMD можно прочитать в файле ..\MAPBASIC\DOC\MAC.DOC, появляющемся после установки пакета MapBasic.

#### Поддержка Геолинка (Hotlink)

Приложения MapBasic запущенные инструментом Геолинк (Hotlink) могут получить информацию об активном объекте через функцию CommandInfo. В таблице приведен список атрибутов

## Функция **CommandInfo( )**

---

которые могут быть запрошены:

Значение <i>attribute</i>	Результат <b>CommandInfo(attribute)</b>
CMD_INFO_HL_WINDOW_ID	Id окна карты или списка
CMD_INFO_HL_TABLE_NAME	Имя таблицы ассоциированной со слоем карты или окном списка
CMD_INFO_HL_ROWID	Id строки таблицы соответствующая соответствующее объекту карты или записи в списке
CMD_INFO_HL_LAYER_ID	id слоя, если программа была запущена из окна карты.
CMD_INFO_HL_FILENAME	Имя запущенного файла.

**Смотрите также:**

**FrontWindow(\_), SelectionInfo(\_), Set Command Info, WindowInfo(\_)**

## Оператор Commit Table

### Назначение:

Сохраняет последнюю редакцию таблицы на диске или сохраняет ее копию.

Access добавлен к тем типам таблиц, которые могут редактироваться в MapInfo, также как и “родные” (Native) таблицы и таблицы DBF. Если выбран тип таблицы Access, то будет затребован маршрут к базе данных. Если маршрут и/или база данных недоступны, то будет создана новая база данных. К таблице в Access будет добавлена колонка MAPINFO\_ID.

### Синтаксис:

```
Commit Table table
  [ As filespec
    [ Type { NATIVE |
      DBF [ Charset char_set ] |
      Access Database database_filespec [ Version version ] Table tablename
      [ Password pwd ] [ Charset char_set ] |
      QUERY
    } ]
    ODBC Connection ConnectionNumber Table tablename
  [ CoordSys... ]
  [ Version version ] ]
  [ { Interactive | Automatic commit_keyword } ]
```

*table* имя таблицы, которую Вы сохраняете.

*filespec* спецификация файла (включая DOS-маршрут). Это то место, где сохраняется файл .TAB.

*version* выражение, определяющее версию базы данных Microsoft Jet database format используемую для новой базы данных. Допускаются значения 4.0 (для Access 2000) или 3.0 (для Access '95/'97). Если выражение пропущено, то по умолчанию используется версия 4.0. Если база данных, в которой создается таблица уже существует, то определение версии базы данных игнорируется

*char\_set* имя строковой переменной; см. раздел, посвященный функции CharSet.

*database\_filespec* строка, которая определяет имя и DOS-маршрут к доступной базе данных Access. Если такая база не существует, MapInfo создаст новый Access-файл .MDB.

*tablename* строковая переменная, определяющая имя таблицы, которая появится в Access.

*pwd* пароль на уровне базы данных, определяемый при включении защиты базы данных.

**ODBC** определяет копию *Table* которая сохраняется в базе данных, определенной номером удаленного соединения *ConnectionNumber*.

*ConnectionNumber* целое значение, номер соединения с базой данных.

*tablename* строковая переменная, определяющая имя таблицы, которая появится в Access.

**CoordSys** система координат; см. подробнее раздел CoordSys.

*version* величина от 100 (для таблиц, которые могут читаться ранними версиями MapInfo) до 300 (MapInfo 3.0 формат), для не-Access таблиц. Для таблиц Access, версия должна быть 410.

## Оператор Commit Table

---

*commit\_keyword* одно из следующих ключевых слов:: **NoCollision**, **ApplyUpdates**, **DiscardUpdates**

### Описание

Если предложение **As** не определено, оператор **Commit** сохраняет любые изменения в таблице. Это аналогично команде **ФАЙЛ > СОХРАНИТЬ**.

Оператор **Commit** который включает предложение **As** имеет тот же самый эффект, как и команда **ФАЙЛ > СОЗДАТЬ КОПИЮ**. Предложение **As** может быть использовано для сохранения таблицы под другим именем, в другом месте, или в виде другого типа файла, проекции.

Для сохранения таблицы под новым именем, укажите новое имя в строковой переменной *filespec*. Для сохранения таблицы в другом месте, укажите путь в начале строковой переменной *filespec*.

Для сохранения таблицы как файла нового типа, включите предложение **Type** внутри предложения **As**. По умолчанию, тип новой таблицы **NATIVE**, но она также может быть сохранена как **DBF**.

Предложение **CharSet** определяет установку шрифта. Параметр *char\_set* должен быть строковой константой, такой как "MacRoman" или "WindowsLatin1". Если предложение **CharSet** не определено, MapBasic использует по умолчанию тот шрифт, который установлен в Windows в это время. См. так же предложение **CharSet** для большей информации.

Для сохранения таблицы с использованием других систем координат или проекций, включите предложение **CoordSys** в предложение **As**. Обратите внимание, что только геокодируемые таблицы могут иметь систему координат и проекцию.

Для сохранения запроса используйте тип таблицы **QUERY**. Может быть сохранен только запрос сделанный пользователем через интерфейс и запрос, созданный оператором **Run Command** в MapBasic. Оператор **Commit Table** создает файлы **.TAB** и **.QRY**.

Предложение **Version** контролирует формат таблицы. Если Вы укажите **Version 100**, MapInfo сохранит таблицу в формате, читаемом ранними версиями MapInfo. Если Вы укажите **Version 300**, MapInfo сохранит таблицу в формате, используемом MapInfo 3.0. Обратите внимание, что объекты типа полилиния и регион, имеющие более 8,000 узлов и полилинии, состоящие из множества сегментов требуют версию 300. Если Вы пропустите предложение **Version**, то таблица сохранится в формате версии 300.

**Внимание:** Если приложение MapBasic использует оператор **Commit Table...As** действующий на таблицу у которой есть мемо-поля, то эти мемо-поля не сохранятся в новой таблице. Предупреждение об этом на экране не будет. Если таблица сохраняется в виде новой таблицы с помощью команды MapInfo (**ФАЙЛ > СОЗДАТЬ КОПИЮ**), то MapInfo предупредит пользователя о потере мемо полей. Таким образом, при сохранении новой таблицы через программу MapBasic, предупреждения не будет.

### Сохранение связанных таблиц

Сохранение связанной таблицы может породить конфликт, поскольку другие пользователи могут редактировать РСУБД. Следующие предложения позволят Вам контролировать то, что может произойти при конфликте. (Эти предложения не действуют при сохранении обычной таблицы MapInfo).

**Interactive**

В случае конфликта, MapInfo показывает диалог “Разрешение конфликтов”. После успешного выполнения оператора **Commit Table Interactive**, MapInfo показывает диалог обновления.

### Automatic NoCollision

В случае конфликта, MapInfo не выполняет сохранение. (Этот режим используется по умолчанию, то есть в случае, если не используются предложения **Interactive** или **Automatic**.)

### Automatic ApplyUpdates

В случае конфликта, MapInfo сохраняет значения локальной копии связанной таблицы. (Это аналогично полному игнорированию конфликта.)

### Automatic DiscardUpdates

В случае конфликта MapInfo сохраняет значения из РСУБД (локальные изменения отменяются).

Вы можете сохранить копию связанной таблицы, используя предложение **As**; но, полученная в результате таблица не будет связанной и не может быть обновлена с сервера.

## Оператор ODBC Connection

Длина имени таблицы *tablename* варьирует в зависимости от базы данных. Мы рекомендуем применять 14 или менее символов для имени таблицы, в этом случае гарантирована корректная работа с любой базой данных. В операторе установлена максимально возможная длина имени файла - *tablename*, равная 31 символ.

Если используется предложение **AS** и **ODBC** это Type, то копия таблицы будет сохранена в базе данных, указанной номером соединения *ConnectionNumber* а имя таблицы будет *tablename*. Если исходная таблица имеет географические объекты, то к таблице *tablename*, создаваемой в удаленной базе, могут быть добавлены три колонки **Key**, **Object** и **Style**, независимо от того, есть такие колонки в исходной таблице или нет. Если исходная таблица без географических объектов, то только одна колонка, **Key**, может быть добавлена к таблице базы данных, *tablename*, независимо от того, была ли такая колонка в исходных данных. Колонка Key будет использоваться для создания уникального индекса.

Пространственный индекс будет создаваться в колонке Object, если она есть.

Неподдерживаемые типы объектов не будут сохраняться в создаваемой таблице, но часть их атрибутов будет сохранена. Поддерживаемые базы данных это Oracle, SQL Server, IIS (Informix Universal Server) и Microsoft Access. Таким образом, для сохранения таблицы с пространственной геометрией/объектами (включая сохранение таблицы только с точечными объектами), для SQL Server и IUS потребуется SpatialWare/Blade, в дополнение к пространственным настройкам для Oracle. Схема XY не поддерживается в этом операторе.

## Пример

Следующий пример открывает таблицу STATES, затем использует оператор **Commit** чтобы сделать копию этой таблицы под новым именем (ALBERS). Необязательное предложение **CoordSys** приводит к тому, что таблица ALBERS сохранится с равноплощадной проекцией Альберта.

```
Open Table "STATES"
Commit Table STATES
As "ALBERS"
CoordSys Earth
Projection 9,7, "m", -96.0, 23.0, 20.0, 60.0, 0.0, 0.0
```

## Оператор Commit Table

---

Следующий пример иллюстрирует соединение с удаленной базой данных:

```
dim hodbс as integer
hodbс = server_connect("ODBC", "dlg=1")
Open table "C:\MapInfo\USA"
Commit Table USA
as "c:\temp\as\USA"
Type ODBC Connection hodbс Table "USA"
```

**Смотрите также**

Rollback



## Оператор Continue

### Назначение:

Возобновляет выполнение программы MapBasic.

### Синтаксис:

**Continue**

### Предупреждение:

Оператор используется *только* в окне MapBasic и не может быть частью программы.

### Description

Оператор **Continue** используется для возобновления выполнения приложения MapBasic, остановленного оператором **Stop**. Используется для отладочных целей.

Когда программа выполняет оператор **Stop**, она приостанавливается, и в списке меню **Файл** в окне MapInfo команда **ЗАПУСТИТЬ ПРОГРАММУ MAPBASIC** меняется на **ПРОДОЛЖИТЬ ПРОГРАММУ MAPBASIC**. Продолжить выполнение программы также можно, введя в окно MapBasic оператор **Continue** или выбрав команду **Файл > ПРОДОЛЖИТЬ ПРОГРАММУ MAPBASIC**.

## Предложения Control Button / OKButton / CancelButton

### Назначение:

Часть оператора **Dialog**. Отвечает за создание кнопки с текстом.

### Синтаксис:

```
Control { Button | OKButton | CancelButton }  
        [ Position x, y ] [ Width w ] [ Height h ]  
        [ ID control_ID ]  
        [ Calling handler ]  
        [ Title title_string ]  
        [ Disable ] [ Hide ]
```

где

*x, y* – координаты левого верхнего угла кнопки в окне диалога в специальных диалоговых единицах (смотрите подраздел в описании оператора **Dialog**);

*w* – ширина кнопки в диалоговых единицах, по умолчанию – 40

*h* – высота кнопки в диалоговых единицах, по умолчанию – 18

*control\_ID* – целое число, которое должно быть уникальным по отношению к идентификаторам элементов активного диалога;

*handler* – имя процедуры-обработчика, которая запускается при нажатии на кнопку;

*title\_string* – текст на кнопке.

Ключевое слово **Disable** делает кнопку недоступной для выбора (закрашивается серым).

Ключевое слово **Hide** прячет кнопку из диалогового окна.

### Описание:

Ключевые слова **Button**, **CancelButton** и **OkButton** в предложении **Control** оператора **Dialog** позволяют создавать в диалоге кнопки с текстом, нажатие на которые приводит к выполнению определенных действий.

Кнопки, которым соответствуют ключевые слова **OKButton** и **CancelButton**, являются специальными. Нажатие на первую приводит к закрытию диалогового окна с сохранением всех установленных значений в диалоге. Вторая кнопка также закрывает диалог, но измененные пользователем значения не сохраняются. Каждый диалог должен содержать не более одной кнопки подтверждения (**OKButton**) и не более одной кнопки отмены (**CancelButton**).

Для изменения состояния элемента диалога используйте оператор **Alter Control** (например, для показа скрытой кнопки).

### Пример:

```
Control Button  
  Title "&Восстановить"  
  Calling reset_sub  
  Position 10, 190
```

### Смотрите также:

**Alter Control, Dialog**

## Предложение Control CheckBox

### Назначение:

Часть оператора **Dialog**. Отвечает за создание флажка.

### Синтаксис:

```
Control CheckBox
  [ Position x , y ] [ Width w ]
  [ ID control_ID ]
  [ Calling handler ]
  [ Title title_string ]
  [ Value log_value ]
  [ Into log_variable ]
  [ Disable ] [ Hide ]
```

где

*x*, *y* – координаты левого верхнего угла флажка в окне диалога в специальных единицах измерения диалога (смотрите подраздел в описании оператора **Dialog**);

*w* – ширина в диалоговых единицах;

*control\_ID* – целое число, которое должно быть уникальным по отношению к остальным идентификаторам элементов активного диалога;

*handler* – имя процедуры-обработчика, которая запускается при изменении режима;

*title\_string* – текст справа от флажка;

*log\_value* – логическая величина, задающая начальное значение: FALSE – флажок сброшен;

*log\_variable* – имя логической переменной, которой будет присвоено значение элемента после закрытия диалога.

Ключевое слово **Disable** делает флажок недоступным для выбора (закрашивается серым).

Ключевое слово **Hide** прячет флажок из диалогового окна.

### Описание:

Ключевое слово **CheckBox** в предложении **Control** оператора **Dialog** позволяет создавать в диалоге флажок – элемент, который может иметь только два значения. Обычно такой элемент используется для управления установкой режима. Предложение **Value** позволяет присваивать элементу диалога начальное значение. Если предложение опущено или, наоборот, присутствует в конструкции **Control CheckBox** и задает начальное значение TRUE, то флажок при открытии диалогового окна будет установлен. Если предложение **Value** задает значение FALSE, то флажок будет сброшен.

### Пример:

```
Control CheckBox
  Title "Показывать &Легенду"
  Into showlegend
  ID 6
  Position 115, 155
```

### Смотрите также:

Alter Control, Dialog, ReadControlValue(\_)

## Предложение Control DocumentWindow

### Назначение

Часть оператора Dialog; добавляет управление окном документа к диалогу, который может быть порожденным при использовании интегрированной картографии.

### Синтаксис

**Control DocumentWindow**

**[ Position  $x, y$  ]**

**Width  $w$  ] [ Height  $h$  ]**

**[ ID control\_ID ]**

**[ Disable ] [ Hide ]**

$x, y$  определяют позицию контроля в единицах диалога

$w$  определяет ширину контроля в единицах диалога; стандартная ширина 100

$h$  определяет высоту контроля в единицах диалога; стандартная высота 100

*control\_ID* целое; не может совпадать с другими идентификаторами контроля в диалоге

**Disable** делает контроль изначально неактивным

**Hide** изначально скрывает контроль

### Описание

Если оператор Dialog включает в себя предложение Control DocumentWindow, то диалог включает в себя контроль окна документа, который может быть порожден при использовании Set Next Document.

### Пример

Следующий пример создает легенду в диалоге:

```
Control DocumentWindow
  ID ID_LEGENDWINDOW
  Position 160, 20
  Width 120 Height 150
```

Обработчик диалога должен породить окно как показано в следующем примере:

```
Sub DialogHandler
  OnError Goto HandleError
  Dim iHwnd As Integer
  Alter Control ID_LEGENDWINDOW Enable Show
  ' draw the legend
  iHwnd = ReadControlValue(ID_LEGENDWINDOW)
  Set Next Document Parent iHwnd Style WIN_STYLE_CHILD
  Create Legend
  Exit Sub
HandleError:
  Note "DialogHandler: " + Error$()
End Sub
```

## Предложение Control EditText

### Назначение:

Часть оператора **Dialog**. Отвечает за создание текстового окошка ввода.

### Синтаксис:

```
Control EditText
  [ Position x, y ] [ Width w ] [ Height h ]
  [ ID control_ID ]
  [ Value initial_value ]
  [ Into variable ]
  [ Disable ] [ Hide ] [ Password ]
```

где

*x, y* – координаты левого верхнего угла окошка в окне диалога в специальных единицах измерения диалога (смотрите подраздел в описании оператора **Dialog**);

*w* – ширина в единицах измерения диалога;

*h* – высота в единицах измерения диалога;

*control\_ID* – целое число, которое должно быть уникальным по отношению к остальным идентификаторам элементов активного диалога;

*handler* – имя процедуры-обработчика, которая запускается при изменении режима;

*initial\_value* – строковая величина, задающая начальное значение;

*variable* – имя строковой переменной, которой будет присвоен текст из окошка после закрытия диалога кнопкой “ОК” или “Да”.

Ключевое слово **Disable** делает элемент недоступным (закрашивается серым).

Ключевое слово **Hide** прячет элемент из диалогового окна.

Ключевое слово **Password** включает “слепой” режим ввода текста (показывается звездочка вместо каждого введенного пользователем символа), используется для создания окошка ввода пароля.

### Описание:

Ключевое слово **EditText** в предложении **Control** оператора **Dialog** позволяет создавать в диалоге окошко для ввода текста пользователем. Элемент **EditText** может иметь строковое значение длиной до 32767 символов.

Текстовое окошко может быть как однострочным, так и многострочным. Количество строк определяется высотой элемента диалога. Если высота больше 20 единиц, то окошко будет иметь две и более строк. В этом случае текст, который пользователь введет, будет автоматически разбиваться на строки. Пользователь также может разделять текст на строки с помощью символа перевода каретки (line-feed). В Windows для этого надо нажать на клавиши CTRL+ENTER. Так как символ перевода каретки имеет код 10, Вы можете с помощью функции **Chr\$(10)** задать в параметре *str\_value* начальное значение в несколько строк.

Для перемещения фокуса в элемент **EditText** используйте оператор **Alter Control...Active**.

### Пример:

```
Control EditText
  Value "Торговые точки"
  Position 68, 8   Width 90   ID 1
  Into s_map_title
```

### Смотрите также:

**Alter Control, Dialog, ReadControlValue(\_)**

## Предложение Control GroupBox

### Назначение:

Часть оператора **Dialog**. Отвечает за создание прямоугольной рамки с текстом.

### Синтаксис:

```
Control GroupBox  
    [ Position x , y ] [ Width w ] [ Height h ]  
    [ Title title_string ]  
    [ Hide ]
```

где

*x*, *y* – координаты левого верхнего угла рамки в окне диалога в специальных единицах измерения диалога (смотрите подраздел в описании оператора **Dialog**);

*w* – ширина рамки в единицах измерения диалога;

*h* – высота рамки в единицах измерения диалога;

*title\_string* – текст заголовка, который начинается от левого верхнего угла рамки.

Ключевое слово **Hide** не показывает групповую рамку в диалоговом окне.

### Описание:

Ключевое слово **GroupBox** в предложении **Control** оператора **Dialog** позволяет создавать в диалоге прямоугольную рамку с заголовком. Элемент **GroupBox** в окне диалога выполняет оформительскую роль, потому не имеет значения. С помощью рамки Вы можете объединять другие элементы в диалоге в группы.

### Пример:

```
Control GroupBox  
    Title "Уровень детализации"  
    Position 5, 30  
    Height 40    Width 70
```

### Смотрите также:

[Alter Control, Dialog](#)

## Предложения Control ListBox / MultiListBox

### Назначение:

Часть оператора **Dialog**. Отвечают за создание списков.

### Синтаксис:

```
Control { ListBox | MultiListBox }
    [ Position x, y ] [ Width w ] [ Height h ]
    [ ID control_ID ]
    [ Calling handler ]
    [ Title { str_expr | From Variable str_array_var } ]
    [ Value i_selected ]
    [ Into i_variable ]
    [ Disable ] [ Hide ]
```

где

*x, y* – координаты левого верхнего угла окошка списка в окне диалога в специальных единицах измерения диалога (смотрите подраздел в описании оператора **Dialog**);

*w* – ширина окошка в единицах измерения диалога, по умолчанию – 80;

*h* – высота окошка в единицах измерения диалога, по умолчанию – 70;

*control\_ID* – целое число, которое должно быть уникальным по отношению к остальным идентификаторам элементов активного диалога;

*handler* – имя процедуры-обработчика, которая запускается при изменении пользователем выбора в списке или двойным указанием в списке;

*str\_expr* – строковое выражение, которое задает текст строк списка; элементы списка разделены точкой с запятой (;);

*str\_array\_var* – имя массива строковых переменных;

*i\_selected* – короткое целое число, задающее номер элемента списка, который будет выбран при открытии диалога, по умолчанию в списке не будет выбрано ни одной строки;

*i\_variable* – имя переменной типа SmallInt, которая будет использоваться для сохранения значения выбора в списке после закрытия диалога.

Ключевое слово **Disable** делает список недоступным для выбора (закрашивается серым).

Ключевое слово **Hide** прячет список из диалогового окна.

### Описание:

Ключевые слова **ListBox** и **MultiListBox** в предложении **Control** оператора **Dialog** позволяют создавать в диалоге прямоугольную рамку со списком внутри, строки которого может выбирать пользователь. Если в окошко не вмещаются все строки списка, то окошко снабжается справа полосой прокрутки.

Список **MultiListBox** отличается от списка **ListBox** тем, что в первом пользователь может выбрать одновременно несколько строк или не выбрать ни одной. Выбор нескольких элементов осуществляется указанием на строку с нажатой клавишей CTRL или SHIFT (это зависит от вычислительной платформы).

Предложение **Title** задает строки списка. Если за словом **Title** задан список текстов в кавычках, через точку с запятой, то каждый текст будет определять одну строчку списка. Например:

```
Title  "1-й квартал;2-й квартал;3-й квартал;4-й квартал"
```

Также список в предложении **Title** можно задавать массивом. В следующем фрагменте "s\_optionlist" – имя строкового массива.

```
Title From Variable s_optionlist
```

### Чтение значений элемента MultiListBox

Из процедуры-обработчика элемента Вы можете прочесть, сколько и какие именно строки выбраны сейчас в окошке списка при помощи функции **ReadControlValue()**.

Для элемента **MultiListBox** существует специальная процедура чтения значений. В окошке списка этого элемента может быть не выбрано ни одной строки, выбрана одна строка, несколько или весь список. Соответственно, элемент может не иметь значений, а может иметь их несколько. Функция **ReadControlValue()**, вызванная из обработчика элемента **MultiListBox**, может возвращать только одну величину за один вызов. Поэтому, для чтения всех значений необходимо функцию вызывать несколько раз. Первый вызов функции дает номер первой строки, выбранной в списке.

Следующий – второй строки и т. д. Когда все значения будут прочитаны, функция вернет ноль. Если ноль будет получен при первом вызове, то, следовательно, в списке ничего не было выбрано.

### Двойной щелчок мышки в списке

Для списка может быть объявлена процедура-обработчик, которая вызывается всякий раз, когда пользователь указывает на список. Причем это может быть простое указание, а может быть двойное, то есть, если быстро нажать два раза на клавишу мышки. Например, двойное указание в списке аналогично простому указанию и последующему нажатию на кнопку ОК, которая закрывает диалог с подтверждением всех изменений значений элементов в диалоге.

Для определения, применялось ли пользователем двойное указание или нет, используется функция **CommandInfo()** в процедуре обработчика спискового элемента диалога. Пример такой процедуры обработчика приводится ниже:

```
Sub lb_handler
  Dim i As SmallInt
  If CommandInfo(CMD_INFO_DLG_DBL) Then
    ' ... если пользователь использовал двойное указание...
    i = ReadControlValue( TriggerControl() )
    Dialog Remove
    ' теперь i содержит номер выбранного элемента списка...
  ,
  End If
End Sub
```

### Пример:

```
Control ListBox
  Title "1-ого квартала;2-ого квартала;3-его квартала;
        4-ого квартала;текущего месяца;года"
  ID 3
  Value 1
  Into i_quarter
  Position 10, 92 Height 40
```

### Смотрите также:

**Alter Control, Dialog, ReadControlValue()**



## Предложения Control PenPicker / BrushPicker / SymbolPicker / FontPicker

### Назначение:

Часть оператора **Dialog**. Предложения отвечают за создание кнопок выбора стиля: линии, штриха, символа или шрифта.

### Синтаксис:

```
Control { PenPicker | BrushPicker | SymbolPicker | FontPicker }  
    [ Position x, y ] [ Width w ] [ Height h ]  
    [ ID control_ID ]  
    [ Calling handler ]  
    [ Value style_expr ]  
    [ Into style_var ]  
    [ Disable ] [ Hide ]
```

где

*x, y* – координаты левого верхнего угла кнопки в окне диалога в специальных единицах измерения диалога (смотрите подраздел в описании оператора **Dialog**);

*w* – ширина кнопки в единицах измерения диалога, по умолчанию 20;

*h* – высота кнопки в единицах измерения диалога, по умолчанию 20;

*control\_ID* – целое число, которое должно быть уникальным по отношению к остальным идентификаторам элементов активного диалога;

*handler* – имя процедуры-обработчика, которая запускается при нажатии кнопки "ОК" в диалоге стиля;

*style\_expr* – выражение типа Pen, Brush, Symbol или Font, которое задает начальное значение элемента;

*style\_var* – имя переменной типа Pen, Brush, Symbol или Font (тип переменной должен соответствовать типу элемента диалога).

Ключевое слово **Disable** делает кнопку недоступной для выбора (закрашивается серым).

Ключевое слово **Hide** прячет кнопку из диалогового окна.

### Описание:

Ключевые слова **PenPicker**, **BrushPicker**, **SymbolPicker** и **FontPicker** в предложении **Control** оператора **Dialog** позволяют создавать в диалоге кнопку, нажатие на которую приводит к открытию стандартного диалога стиля оформления объектов. Предложение **Control PenPicker** отвечает за стиль линии, предложение **Control BrushPicker** – за стиль штриха, предложение **Control SymbolPicker** – за стиль символа точечного объекта и предложение **Control FontPicker** – за стиль шрифта.

### Пример:

```
Control SymbolPicker  
    Position 140, 42  
    Into sym_storemarker
```

### Смотрите также:

Alter Control, Dialog, ReadControlValue( )

### Предложение Control PopupMenu

#### Назначение:

Часть оператора **Dialog**. Отвечает за создание раскрывающегося меню.

#### Синтаксис:

```
Control PopupMenu
  [ Position x, y ] [ Width w ]
  [ ID control_ID ]
  [ Calling handler ]
  [ Title { str_expr | From Variable str_array_var } ]
  [ Value i_selected ]
  [ Into i_variable ]
  [ Disable ]
```

где

*x, y* – координаты левого верхнего угла окошка меню в окне диалога в специальных единицах измерения диалога (смотрите подраздел в описании оператора **Dialog**);

*w* – ширина окошка в единицах измерения диалога, по умолчанию 80;

*control\_ID* – целое число, которое должно быть уникальным по отношению к остальным идентификаторам элементов активного диалога;

*handler* – имя процедуры-обработчика, которая запускается при изменении выбора в списке;

*str\_expr* – строковое выражение, которое задает текст строк списка меню, где элементы списка разделены точкой с запятой (;);

*str\_array\_var* – имя массива строковых переменных;

*i\_selected* – короткое целое число, задающее номер элемента списка меню, который будет показан в окошке при открытии диалога, по умолчанию будет показан первый элемент;

*i\_variable* – имя переменной типа SmallInt, которая будет использоваться для сохранения номера выбранного в меню элемента после закрытия диалога.

Ключевое слово **Disable** делает меню недоступным для выбора (закрашивается серым).

#### Описание:

Ключевое слово **PopupMenu** в предложении **Control** оператора **Dialog** позволяет создавать в диалоге меню, представляющее собой однострочное окошко с кнопкой справа. Нажатие на кнопку раскрывает комбинированное окошко списка, в котором пользователь может выбрать строчку. Выбранная строка будет отображена в окошке.

Предложение **Title** задает список элементов меню. Если за словом **Title** задан список текстов в кавычках, через точку с запятой, то каждый текст будет определять одну строчку списка. Например:

```
Title "Город;Область;Территория;Регион;Вся страна"
```

Список в предложении **Title** можно также задавать массивом. В следующем фрагменте "s\_optionlist" – имя строкового массива.

```
Title From Variable s_optionlist
```

#### Пример:

```
Control PopupMenu
  Title "Город;Область;Территория;Регион;Вся страна"
  Value 2
```

```
ID 5  
Into mapscope  
Position 10, 150
```

Смотрите также:

Alter Control, Dialog, ReadControlValue( )

## Предложение Control RadioGroup

### Назначение:

Часть оператора **Dialog**. Отвечает за создание кнопок переключателя.

### Синтаксис:

```
Control RadioGroup  
  [ Position x, y ]  
  [ ID control_ID ]  
  [ Calling handler ]  
  [ Title { str_expr | From Variable str_array_var } ]  
  [ Value i_selected ]  
  [ Into i_variable ]  
  [ Disable ] [ Hide ]
```

где

*x, y* – координаты левого верхнего угла прямоугольника, в который вписываются кнопки переключателя, в окне диалога в специальных единицах измерения диалога (смотрите подраздел в описании оператора **Dialog**);

*control\_ID* – целое число, которое должно быть уникальным по отношению к остальным идентификаторам элементов активного диалога;

*handler* – имя процедуры-обработчика, которая запускается при указании на одну из кнопок;

*str\_expr* – строковое выражение, которое задает текст подписей справа от кнопок, где каждая подпись отделена от другой точкой с запятой (;), по количеству подписей определяется количество кнопок в переключателе;

*str\_array\_var* – имя массива строковых переменных;

*i\_selected* – короткое целое число, задающее номер кнопки, которая будет выбрана при открытии диалога, по умолчанию выбирается первая кнопка;

*i\_variable* – имя переменной типа SmallInt, которая будет использоваться для сохранения номера выбранной кнопки после закрытия диалога.

Ключевое слово **Disable** делает переключатель недоступным для выбора (закрашивается серым).

Ключевое слово **Hide** прячет переключатель из диалогового окна.

### Описание:

Ключевое слово **RadioGroup** в предложении **Control** оператора **Dialog** позволяет создавать в окне диалога кнопочный переключатель. Каждая кнопка представляет собой небольшой кружок с подписью справа, при выборе в кружок помещается черная точка. Выбрана может быть только одна кнопка в переключателе.

Предложение **Title** задает список подписей для кнопок. Если за словом **Title** задан список текстов в кавычках, через точку с запятой, то каждый текст будет определять одну кнопку переключателя.

Например:

```
Title  "&Всё;В&мборочно"
```

Список в предложении **Title** можно также задавать массивом. В следующем фрагменте *s\_optionlist* – имя строкового массива.

```
Title  From Variable  s_optionlist
```

### Пример:

```
Control RadioGroup
  Title "&Всё;В&ыборочно"
  Value 2
  ID 2
  Into details
  Calling rg_handler
  Position 15, 42
```

### Смотрите также:

[Alter Control](#), [Dialog](#), [ReadControlValue\( \)](#)

### Предложение Control StaticText

#### Назначение:

Часть оператора **Dialog**. Отвечает за создание текстового элемента в окне диалога, (неизменяемого текста).

#### Синтаксис:

```
Control StaticText
    [ Position x, y ]
    [ Width w ] [ Height h ]
    [ ID control_ID ]
    [ Title title_string ]
    [ Hide ]
```

где

*x, y* – координаты левого верхнего угла прямоугольника, в который можно вписать строку, в окне диалога в специальных единицах измерения диалога (смотрите подраздел в описании оператора **Dialog**);

*w* – ширина подписи в единицах измерения диалога;

*h* – высота подписи в единицах измерения диалога;

*control\_ID* - целое; не может быть таким же как ID другого элемента управления в диалоге

*title\_string* – текст.

Ключевое слово **Hide** прячет текстовый элемент из диалогового окна.

#### Описание:

Ключевое слово **StaticText** в предложении **Control** оператора **Dialog** позволяет создавать тексты в окне диалога. Элемент **GroupBox** в окне диалога выполняет оформительскую роль, и потому не может иметь значения. Однако, если элемент **StaticText** задан в операторе сразу до или после элемента **EditText**, то клавишное сокращение в тексте **StaticText** можно использовать для перехода в окошко элемента **EditText**.

Если Вы хотите, чтобы текст в диалоге занимал несколько строк, то используйте предложения **Width** и **Height**. Если эти предложения не использовать, то текст будет иметь только одну строку.

#### Пример:

```
Control StaticText
    Title "Заголовок Карты:"
    Position 5, 10
```

#### Смотрите также:

**Alter Control, Dialog**

## Функция **ConvertToPline( )**

### Назначение:

Возвращает полилинию, которая описывает заданный объект.

### Синтаксис:

**ConvertToPline(*object*)**

где

*object* – объект для преобразования любого типа, кроме текстового и точечного.

### Величина, полученная в результате:

Полилиния. Величина типа Object.

### Описание:

Функция **ConvertToPline( )** возвращает объект типа "полилиния", которая описывает объект *object*.

Так, если *object* задает область, то функция **ConvertToPline( )** вернет ломаную линию, представляющую границу области, и с тем же количеством узлов.

Результат функции **ConvertToPline( )** такой же, как при выполнении команды ОБЪЕКТЫ > ПРЕВРАТИТЬ В ПОЛИЛИНИИ, за тем исключением, что функция создает новый объект, не меняя объект *object*.

### Функция **ConvertToRegion( )**

#### Назначение:

Возвращает область по форме заданного объекта.

#### Синтаксис:

**ConvertToRegion(*object*)**

где

*object* – объект для преобразования любого типа, кроме текстового и точечного.

#### Величина, полученная в результате:

Область. Величина типа Object.

#### Описание:

Функция **ConvertToRegion(\_)** возвращает область, имеющую такую же форму, какую имеет объект *object*. Так, если параметр *object* задает объект типа "прямоугольник", то функция **ConvertToRegion(\_)** вернет область такой же прямоугольной формы.

Сохраняются все значения стилей оформления объекта. Недостающие атрибуты используются по текущим значениям стилей. В полилинии первый узел не совпадает с последним. При преобразовании полилинии, если первый узел совпадает с последним, то они сливаются. Если первый и последний узлы полилинии не совпадают, то при преобразовании в область они соединяются.

Результат функции **ConvertToRegion(\_)** такой же, как при выполнении команды ОБЪЕКТЫ > ПРЕВРАТИТЬ В ОБЛАСТИ, за тем исключением, что функция создает новый объект, не меняя объект *object*.



## Функция **ConvexHull()**

### Назначение:

Возвращает объект-регион, который представляет собой полигон-контур, созданный на основании узлов входного объекта. Он содержит минимальное количество точек (т.е. точки входного объекта лежат на границах или внутри полигона). Т.е. все внешние углы созданного полигона-контура больше, чем 180 градусов.

### Синтаксис:

**ConvexHull ( *inputobject* )**

*inputobject* - оконтуриваемый объект

### Возвращаемое значение:

**Возвращает объект-регион**

### Описание:

Функция **ConvexHull()** возвращает регион, представляющий контур, охватывающий точки входного объекта. Функция **ConvexHull()** создает один контур **за раз**. Чтобы создать контуры вокруг нескольких объектов, используйте оператор **Create Object As ConvexHull**.

### Пример:

Следующий пример выбирает штат New York из таблицы штатов, затем создает полигон-контур вокруг выборки.

```
Dim Resulting_object as object
select * from States
where State_Name = »New York»
Resulting_object = ConvexHull(selection.obj)
Insert Into States(obj) Values (Resulting_object)
```

### Смотрите также:

**Create Object**

## Предложение CoordSys

### Назначение:

Задаёт систему координат.

### Синтаксис (вариант 1):

```
CoordSys Earth
  [ Projection type,
    datum,
    unitname
    [, origin_longitude ]
    [, origin_latitude ]
    [, standard_parallel_1 [, standard_parallel_2 ] ]
    [, azimuth ]
    [, scale_factor ]
    [, false_easting ]
    [, false_northing ]
    [, range ] ]
  [ Affine Units unitname, A, B C, D, E, F ]
  [ Bounds ( minx,miny ) ( maxx,maxy ) ]
```

### Синтаксис (вариант 2):

```
CoordSys Nonearth
  [ Affine Units unitname, A, B C, D, E, F ]
  Units unitname
  Bounds ( minx, miny ) ( maxx, maxy )
```

### Синтаксис (вариант 3):

```
CoordSys Layout Units paperunitname
```

### Синтаксис (вариант 4):

```
CoordSys Table tablename
```

### Синтаксис (вариант 5):

```
CoordSys Window window_id
```

где

*type* – положительное целое число, представляющее тип координатной системы;

*datum* – положительное целое число, определяющее референс-эллипсоид;

*unitname* – единица измерения расстояний, строковая величина (например, "m" – метры, список единиц приведен в описании оператора **Set Distance Units**);

*origin\_longitude* – долгота точки отсчета (нулевой точки) в градусах, вещественное число (тип Float);

*origin\_latitude* – широта точки отсчета (нулевой точки) в градусах, вещественное число (тип Float);

*standard\_parallel\_1* и *standard\_parallel\_2* – широта в градусах, вещественное число (тип Float);

*azimuth* – азимутальный угол в градусах, вещественное число (тип Float);

*scale\_factor* – коэффициент искажения (масштабный коэффициент), вещественное число (тип Float);

*range* – вещественное число от 1 до 180 (тип Float), определяющее, какая часть Земли видна;

*minx* – минимальная X-координата, вещественное число (тип Float);

*miny* – минимальная Y-координата, вещественное число (тип Float);

- maxx* – максимальная X-координата, вещественное число (тип Float);  
*maxy* – максимальная Y-координата, вещественное число (тип Float);  
*paperunitname* – строковая величина, представляющая имя "бумажной" единицы (например, "in" – дюймы, список единиц приведен в описании оператора **Set Paper Units**);  
*tablename* – имя открытой таблицы;  
*window\_id* – целочисленный идентификатор окна Карты или Отчета.
- A задает масштабирование или растяжение вдоль оси X.  
 B задает поворот или сдвиг вдоль оси X.  
 C задает смещение вдоль оси X.  
 D задает масштабирование или растяжение вдоль оси Y.  
 E задает поворот или сдвиг вдоль оси Y.  
 F задает смещение вдоль оси Y.

#### Описание:

Предложение **CoordSys** не является отдельным оператором, а входит в состав тех операторов, в которых необходимо задавать координатную систему. С помощью этого предложения можно также задавать проекции Карты, которые используются с данной координатной системой. Это предложение, например, может использоваться в операторе **Set Map** для переопределения проекции, используемой в окне Карты.

Первый вариант синтаксиса предложения используется для Карт мира. Параметры предложения **Projection** задают проекцию Карты (если она есть), и должны использоваться в соответствии с координатной системой. Если это предложение опущено, то MapBasic использует координатную систему широта/долгота, представляющую Северную Америку 1927 года (NAD-27).

Второй вариант синтаксиса предложения **CoordSys** используется для координатных систем планов, например, для поэтажного плана или другого CAD-изображения.

Третий вариант синтаксиса предложения (**CoordSys Layout**) используется для задания координатной системы в окне Отчета. Программа MapBasic должна выполнить оператор **Set CoordSys Layout** перед тем, как создавать объекты Отчета и работать с ними. Параметр *unitname* задает имя "бумажной" единицы, которая будет использоваться в окне. Например, следующий оператор **Set CoordSys** определяет дюймы, как единицы измерений в окне Отчета:

```
Set CoordSys Layout Units "in"
```

Четвертый вариант синтаксиса предложения (**CoordSys Table**) используется для задания координатной системы в таблице, данные которой сохраняются на диске.

Пятый вариант синтаксиса предложения (**CoordSys Window**) позволяет установить такую же координатную систему, как в окне *window\_id*.

В операторах **Set Map** и **Set Digitizer**, использующих предложение **CoordSys**, MapBasic игнорирует предложение **Bounds**. Это предложение используется для планов (непроецированных Карт), когда предложение **CoordSys** входит в состав других операторов.

Предложение **Bounds** задает границы, в которых показывается Карта. Объект не может быть создан за пределами заданных этим предложением границ. Если задается координатная система Карты мира (Earth), то Вы должны опустить предложение **Bounds**, так как MapInfo пытается по умолчанию охватить всю Землю.

## Предложение CoordSys

---

**Замечание:** В операторе **Create Map** можно увеличить точность координат на Карте, задав более узкий охват предложением **Bounds**.

Каждая картографическая проекция задается уравнением, имеющим свой индивидуальный набор параметров. Поэтому предложение **CoordSys** может иметь разный набор параметров в предложении **Projection**. Например, уравнение, задающее проекции Робинсона, использует в качестве параметров референс-эллипсоид (Datum), единицы измерения (Units) и начальную широту (Origin Latitude), а уравнение для модифицированной проекции Меркатора – референс-эллипсоид (Datum), единицы измерения (Units), начальную широту (Origin Latitude), начальную долготу (Origin Latitude), коэффициент искажения (Scale Factor), восточное смещение (False Easting) и северное смещение (False Northing).

Для дальнейшей информации о проекциях и координатных системах смотрите документацию Map-Info.

В каждом приложении MapBasic действует своя координатная система. При выполнении одним приложением оператора **Set CoordSys**, для другого приложения эти установки не имеют значения. Установка координатной системы действительна только для того приложения, в котором она была определена.

### Пример 1:

Оператор **Set Map** задает режим представления существующей Карты. Представленный в примере оператор **Set Map** задает режим показа Карты в проекции Робинсона:

```
Set Map CoordSys Earth Projection 12, 12, "m", 0.
```

Первое число 12 определяет проекцию Робинсона; второе число 12 определяет применение этой проекции ко всему Земному шару; параметр "m" задает метры, как единицы измерения; ноль задает нулевую долготу как начальную.

### Пример 2:

Следующий оператор задает показ карты без проекций:

```
Set Map CoordSys Earth
```

### Пример 3:

В следующем примере открывается таблица WORLD, и с помощью оператора **Commit** сохраняется под именем RWORLD в проекции Робинсона.

```
Open Table "world.tab" As World
Commit Table World As "RWORLD.TAB"
CoordSys Earth Projection 12, 12, "m", 0.
```

### Пример 4:

Следующий оператор копирует проекцию из окна Карты, имеющего идентификатор "first\_map\_id", в окно с идентификатором "second\_map\_id".

```
Set Map
Window second_map_winid
CoordSys Window first_map_winid
```

### Пример 5:

Следующий пример определяет систему координат DCS, которая получена из системы

координат UTM Zone 10, путем аффинного преобразования

$$x1 = 1.57x - 0.21y + 84120.5$$

$$y1 = 0.19x + 2.81y - 20318.0$$

Здесь координаты (x1 , y1) представляют полученные координаты для DCS, а (x, y) исходные координаты UTM Zone 10. Если координаты DCS исчислялись в футах, то предложение CoordSys для DCS выглядело бы следующим образом:

`CoordSys Earth`

`Projection 8, 74, »m», -123, 0, 0.9996, 500000, 0`

`Affine Units »ft», 1.57, -0.21, 84120.5, 0.19, 2.81, -20318.0`

Смотрите также:

`Commit Table, Set CoordSys, Set Map`

### Функция Cos( )

#### Назначение:

Вычисляет косинус.

#### Синтаксис:

**Cos**(*num\_expr*)

где

*num\_expr* – численное выражение угла в радианах.

#### Величина, полученная в результате:

Вещественное число. Величина типа Float.

#### Описание:

Функция **Cos( )** вычисляет косинус числа, полученного в результате вычисления выражения *num\_expr*. Угол должен задаваться в радианах. Диапазон возвращаемого значения находится между единицей и минус единицей включительно.

Для перевода градусов в радианы число необходимо умножить на число DEG\_2\_RAD. Для обратного конвертирования используется коэффициент RAD\_2\_DEG. Чтобы Ваша программа могла использовать эти коэффициенты конвертирования, она должна содержать оператор **Include "MAPBASIC.DEF"**:

#### Пример:

```
Include "MAPBASIC.DEF"
Dim x, y As Float
x = 60 * DEG_2_RAD
y = Cos(x)
' y равен 0.5, поскольку
' косинус от 60 градусов равен 0.5
```

#### Смотрите также:

**Acos( ), Asin( ), Atn( ), Sin( ), Tan( )**

## Оператор Create Arc

### Назначение:

Создает объект типа "дуга".

### Синтаксис:

#### Create Arc

```
[ Into { Window window_id | Variable var_name } ]  
(x1, y1) (x2, y2)  
start_angle end_angle  
[Pen... ]
```

где

*window\_id* – идентификатор окна, целое число;

*var\_name* – имя объектной переменной;

*x1, y1* – координаты одного угла минимального прямоугольного покрытия (МПП), прямоугольника, описывающего дугу;

*x2, y2* – координаты противоположного по диагонали угла МПП дуги;

*start\_angle* – значение начального угла дуги, в градусах;

*end\_angle* – значение конечного угла дуги, в градусах;

**Pen** – слово, с которого начинается стандартное предложение для назначения стиля линии.

### Описание:

Результатом действия оператора **Create Arc** является новый объект типа "дуга".

Если оператор использует предложение **Into Variable**, то созданный объект "дуга" объявляется как значение объектной переменной. Если слово **Into** указывает окно, объект помещается на подготовленное место в окне (например, в изменяемый слой). Если **Into** вообще нет в операторе, MapBasic попытается создать дугу в самом верхнем окне. Если это невозможно (например, поверх всех окон лежит окно Графика), то объект не будет создан.

Параметры *x* и *y* являются координатами в той системе координат, которая была объявлена до оператора **Create Arc**, например, оператором **Set CoordSys**. Если система не объявлялась, то координаты будут принимать значения широты и долготы. При этом надо учесть, что MapBasic игнорирует координатную систему самого окна Карты.

Если объект создается для окна Отчета, параметры *x* и *y* интерпретируются как координаты на листе в объявленных ранее единицах измерения листа (смотрите описание оператора **Set Paper Units**).

Так X-координата – это расстояние от левого края листа до точки, и Y-координата – расстояние от верхнего края листа. Перед созданием объекта в окне Отчета не забудьте выполнить оператор **Set CoordSys Layout**.

Если в операторе нет предложения **Pen**, MapBasic, создавая дугу, будет использовать установку соответствующего режима для стиля линии в MapInfo (стиль линии можно изменить командой НАСТРОЙКА > СТИЛЬ ЛИНИЙ).

### Смотрите также:

**Insert, Pen, Update**

### Оператор Create ButtonPad

#### Назначение:

Создает инструментальную панель.

Синтаксис:

```
Create ButtonPad { title_string | ID pad_num }  
    As button_definition [ button_definition ... ]  
        [ Title title_string ]  
        [ Width w ]  
        [ Position (x, y) [ Units unit_name ] ]  
        [ ToolbarPosition ( row , column ) ]  
        [ { Show | Hide } ]  
        [ { Fixed | Float } ]
```

где

*title\_string* – заголовок инструментальной панели;

*pad\_num* – идентификатор стандартной инструментальной панели (1 – для панели "Операции", 2 – для панели "Пенал", 3 – для панели "Программы", 4 – "Команды", 5 – "ODBC");

*w* – ширина панели в количестве кнопок, которые можно разместить по горизонтали;

*x, y* – координаты верхнего левого угла панели в "бумажных" единицах измерения;

*unit\_name* – имя "бумажной" единицы (например, "in" – дюйм, "cm" – сантиметр);

*row, column* – координаты инструментальной панели, когда она находится в прикрепленном состоянии (docked); например, координаты 0, 0 означают расположение строки инструментов прижатой к левому и верхнему краям рабочего окна, а 0, 1 задают положение панели второй строкой).

Каждый параметр *button\_definition* может быть либо ключевым словом **Separator**, либо конструкцией следующего синтаксиса:

```
{ PushButton | ToggleButton | ToolButton }  
    Calling { procedure | menu_code | OLE methodname | DDE server , topic }  
    [ ID button_id ]  
    [ Icon n [ File file_spec ] ]  
    [ Cursor n [ File file_spec ] ]  
    [ DrawMode dm_code ]  
    [ HelpMsg msg ]  
    [ ModifierKeys { On | Off } ]  
    [ Enable | Disable ]  
    [ Check | Uncheck ]
```

где

*procedure* – обработчик, вызываемый нажатием на кнопку. Обработчиком может быть sub-процедура или стандартный в MapInfo код (например, M\_FILE\_OPEN из MENU.DEF);

*menu\_code* – стандартный в MapInfo код команды из MENU.DEF (например, M\_FILE\_OPEN);

MapInfo начнет выполнение команды, как только пользователь нажмет на кнопку;

*methodname* – строковая величина, задающая имя OLE-метода;

*server, topic* – строковая величина, задающая DDE-сервера и имя темы (topic).

Предложение **ID button\_id** задает кнопке уникальный номер, который можно будет потом использовать как ее идентификатор, в случае, если несколько кнопок вызывают один обработчик, а также в операторе **Alter Button**.



Предложение **Icon** *n* задает картинку, которая будет на кнопке. Здесь *n* может быть одним из специальных кодов из файла ICONS.DEF (например, MI\_ICON\_RULER). Подпредложение **File** *file\_spec* задает файл ресурсов изображений, в этом случае параметр *n* должен быть целочисленным идентификатором одного из ресурсов файла *file\_spec*.

Предложение **Cursor** *n* задает форму, которую примет указатель мыши после выбора кнопки. Здесь *n* может быть одним из специальных кодов из файла ICONS.DEF (например, MI\_CURSOR\_ARROW). Это предложение может входить в описание кнопки инструмента (тип **ToolButtons**). Подпредложение **File** *file\_spec* задает файл ресурсов изображений, в этом случае параметр *n* должен быть целочисленным идентификатором одного из ресурсов файла *file\_spec*.

Предложение **DrawMode** *dm\_code* задает возможность инструмента рисовать (использование возможности передвигать мышку с нажатой клавишей) или только указывать (использование только возможности нажимать на клавишу мышки), при этом параметр *dm\_code* должен быть одним из специальных кодов из файла ICONS.DEF (например, DM\_CUSTOM\_LINE). Это предложение может входить в описание кнопки инструмента (тип **ToolButtons**).

Предложение **HelpMsg** *msg* задает текст подсказки, которая появляется в строке сообщений при указании на кнопку, а также может задавать текст для плавающей подсказки **ToolTip**. Первая часть строки *msg* используется строкой сообщений. Если величина *msg* включает в себя литеры \n, то текст, следующий за ними, используется подсказкой **ToolTip**. Параметр *msg* должен иметь тип String.

Предложение **ModifierKeys** управляет использованием клавиш SHIFT и CTRL в режиме рисования, сопровождающемся прорисовкой образа объекта ("rubberband"), инструментом кнопки типа **ToolButton**. По умолчанию используется режим **Off**, не использующий клавиши SHIFT и CTRL.

### Описание:

Оператор **Create ButtonPad** создает новую инструментальную панель. Созданную панель Вы можете по ходу выполнения программы изменять при помощи операторов **Alter Button** и **Alter ButtonPad**.

Инструментальная панель может прятаться. Для создания инструментальной панели в скрытом состоянии используйте ключевое слово **Hide**.

Инструментальная панель может показываться на экране в одном из двух состояний: плавающим ("floating") – в виде вспомогательного окна и строкой ("docked") – вытянутой в полосу и прижатой к верхнему краю рабочего окна. Если Вы хотите, чтобы созданная панель была зафиксирована у верхнего края окна, то используйте ключевое слово **Fixed**. Для показа панели вспомогательным окном используется слово **Float**.

Если панель плавающая, то ее положение на экране задается предложением **Position**. Если панель показывается строкой, то предложением **ToolbarPosition**.

Другую информацию об инструментальных панелях смотрите в описании оператора **Alter ButtonPad**. Инструментальные панели также описаны в 6 главе *Руководства пользователя MapBasic*.

### Режимы предложения Calling

Предложение **Calling** задает, что должно случиться, если пользователь нажмет на кнопку инструментальной панели. Приведем возможные примеры использования этого предложения:

#### Примеры

#### Описание

## Оператор Create ButtonPad

Calling M\_FILE\_NEW

Если за словом **Calling** идет целочисленный код из файла MENU.DEF, MapInfo запускает на выполнение соответствующую стандартную команду MapInfo (например, ФАЙЛ > НОВАЯ ТАБЛИЦА).

Calling my\_procedure

Если задано имя sub-процедуры, MapInfo передает управление этой процедуре. Эта процедура должна быть частью программы MapBasic, создавшей эту инструментальную панель.

Calling OLE "methodname"

Только для Windows. MapInfo управляет событиями, обращаясь с именем метода к объекту OLE Automation, усвоенному SetCallback-методом MapInfo. Детали смотрите в 12 главе *Руководства пользователя MapBasic*.

Calling DDE "server","topic"

Только для Windows. MapInfo управляет событиями, присоединяясь через DDE к паре "сервер|тема" и посылая выполняемое сообщение на DDE-сервер.

В последних двух случаях строка послания для OLE или DDE сервера должна начинаться с трех символов "MI:" для того, чтобы сервер смог определить, что послание пришло от MapInfo. Остальная часть этой строки состоит из разделенного запятыми списка значений возврата функций с **CommandInfo(1)** по **CommandInfo(8)**. Детали читайте в 12 главе *Руководства пользователя MapBasic*.

### Пример:

```
Create ButtonPad "Новые Кнопки" As
PushButton
    HelpMsg "Нажмите на эту кнопку для вывода диалога запроса"
    Calling button_sub_proc
    Icon MI_ICON_ZOOM_QUESTION

ToolButton
    HelpMsg "Используйте этот инструмент для рисования
            нового маршрута"
    Calling tool_sub_proc
    Icon MI_ICON_CROSSHAIR
    DrawMode DM_CUSTOM_LINE

ToggleButton
    HelpMsg "Переключение показа расстояний"
    Calling toggle_prox_check
    Icon MI_ICON_RULER
    Check
    Title "Средства"
    Width 3
    Show
```

### Смотрите также:

**Alter Button, Alter ButtonPad**

## Оператор Create ButtonPads As Default

### Назначение:

Восстанавливает стандартные инструментальные панели в их начальном положении.

### Синтаксис:

**Create ButtonPads As Default**

### Описание:

Оператор убирает все инструментальные панели, построенные приложением, и восстанавливает три стандартные панели в их прежнем виде: Операции, Пенал и Программы.

Использовать оператор **Create ButtonPads As Default** надо осторожно, так как удаляются и те панели, которые были построены другими приложениями MapBasic.

### Смотрите также:

**Alter Button, Alter ButtonPad, Create ButtonPad**

### Оператор Create Cartographic Legend

Новый оператор **Create Cartographic Legend** позволяет создать и отобразить стиль картографических легенд, также как и тематических легенд для активного окна карты. Каждый стиль картографической и тематической легенды будет связан только с одним окном карты, так что одновременно может быть открыто несколько окон легенды.

Вы можете создать раздел легенды для каждого картографического или тематического слоя, который Вы захотите включить в легенду. Картографические и тематические разделы будут включать заголовки и подзаголовки легенды. Картографические разделы легенды показывают стили слоев карты; разделы легенды отражают цвета, символы и их размер для тематических слоев. Вы можете создать разделы, которые имеют стили, основанные на стиле окна карты или создать Ваши собственные разделы легенды.

Ранее в легенда в MapInfo Professional имела вид одного плавающего окна, в котором могло отобразиться только содержимое тематического слоя для активного окна карты. Новое окно легенды размещается теперь вместо текущего окна легенды; таким образом, текущее окно легенды и его функции до сих пор возможны, и вызывается это программным способом, используя существующие операторы MapBasic (например, Create Legend, Set Legend, и др...)

#### Синтаксис:

##### Create Cartographic Legend

```
[ From Window map_window_id ]
[ Behind ]
[ Position ( x , y ) [ Units paper_units ] ]
[ Width win_width [ Units paper_units ] ]
[ Height win_height [ Units paper_units ] ]
[ Window Title { legend_window_title }
[ ScrollBars { On | Off } ]
[ Portrait | Landscape | Custom ]
[ Default Frame Title { def_frame_title } [ Font... ] } ]
[ Default Frame Subtitle { def_frame_subtitle } [ Font... ] } ]
[ Default Frame Style { def_frame_style } [ Font... ] } ]
[ Default Frame Border Pen [ [ pen_expr ]
Frame From Layer { map_layer_id | map_layer_name
[ Using
    [ Column { column | object } ]
    [ Label { expression | default } ]
    [ Position ( x , y ) [ Units paper_units ] ]
    [ Title { frame_title [ Font... ] } ]
    [ SubTitle { frame_subtitle [ Font... ] } ]
    [ Border Pen pen_expr ]
    [ Style [ Font... ] [ Norefresh ]
    [ Text { style_name } { Line Pen... | Region Pen... Brush... | Symbol Symbol... } |
Collection [Symbol ...] [ Line Pen ... ] [ Region Pen... Brush ... ] }
```

Предложение **Style** и ключевое слово **NoRefresh** позволяют создавать собственные разделы, которые не будут перерисовываться при обновлении легенды. Если ключевое слово **NoRefresh** используется в предложении **Style**, то таблица не проверяется на предмет стилей. Вместо этого предложение **Style** будет содержать Ваш собственный список определений для стилей, используемых в разделе легенды. Это делается с предложением **Text** и соответствующими

предложениями **Line**, **Region** или **Symbol**. Объекты типа "группа точек" здесь определяются как точечные объекты.

Объекты типа "коллекция" обрабатываются отдельно. При создании легенды на основе типов объектов сначала обрабатываются точки, потом линии, затем полигоны. Коллекции прорисовываются последними. Внутри коллекции рисуются образцы точек, линий и полигонов. *map\_window\_id* - это целночисленный идентификатор окна, который Вы можете получить при вызове функций **FrontWindow()** и **WindowId()**.

*x* - определяет требуемое расстояние от верхнего края рабочего стола MapInfo до верхнего угла окна легенды.

*y* - определяет требуемое расстояние от левого края рабочего стола MapInfo до левого угла окна легенды.

*paper\_units* - это строковая величина, выраженная в бумажных единицах измерения (например "cm" для сантиметров).

*win\_width* - это ширина окна легенды.

*win\_height* - это требуемая высота окна легенды.

*legend\_window\_title* - это строковое выражение, соответствующее заголовку окна легенды, по умолчанию это "Legend of xxx" где xxx это заголовок окна карты.

*def\_frame\_title* - это строковая величина, определяющая заголовок раздела легенды по умолчанию. Эта величина может включать специальный символ "#", который будет замещаться именем текущего слоя.

*def\_frame\_subtitle* - это строковая величина, определяющая подзаголовок раздела легенды по умолчанию. Эта величина может включать специальный символ "#", который будет замещаться именем текущего слоя.

*def\_frame\_style* - это строковая величина, которая указывает на тип каждого объекта в каждом разделе. Символ "#" будет замещаться именем текущего слоя. Символ % будет замещаться словами "Line", "Point", "Region", которые будут соответствовать типу географических объектов. Например, "% of #" будет соответствовать тексту "Region of States" для слоя states.tab.

*pen\_expr* - это выражение **Pen**, соответствующее **MakePen( ширина, стиль, цвет )**. Если по умолчанию атрибуты линии для рамки определены, то они и будут атрибутами линии для рамки раздела легенды. Если предложение, определяющее атрибуты линии для рамки легенды существует, то эти атрибуты и будут использоваться вместо атрибутов, заданных по умолчанию.

*map\_layer\_id* или *map\_layer\_name* определяют слой карты; это может быть целая величина Smallint (например, используйте 1 для определения самого верхнего слоя, не считая косметического) или строковая величина, соответствующая имени таблицы, отображенной на карте. Для тематического слоя необходимо определить *map\_layer\_id*.

*frame\_title* - это строковая величина, определяющая заголовок раздела легенды. Если эта величина определена, то именно она будет использоваться вместо имени, задаваемого по умолчанию величиной *def\_frame\_title*.

*frame\_subtitle* это строковая величина, определяющая подзаголовок раздела легенды. Если эта величина определена, то именно она будет использоваться вместо имени, задаваемого по умолчанию величиной *def\_frame\_subtitle*.

## Оператор Create Cartographic Legend

---

Column – имя атрибутивной колонки из раздела слоя таблицы, или колонка object (означает, что стили легенды базируются на уникальном стиле). По умолчанию – это object.

Label – это выражение или значение, принятое по умолчанию.

*style\_name* – это строковая величина, которая указывает к какому типу объектов относится географический объект: символу, линии, или полигону.

### Описание:

Как минимум, требуется задать одно предложение **Frame**.

Все предложения, относящиеся ко всей легенде (scrollbars, width, и др.) должны соотноситься с первым предложением **Frame**.

Предложение **From Layer** должно быть первым предложением после **Frame**.

Легенда размещается после окна тематической карты.

Предложение **Position** контролирует положение окна легенды на рабочем столе. Верхний левый угол рабочего стола MapInfo имеет позицию 0, 0. Предложения **Width** и **Height** контролируют размер окна легенды. При позиционировании и указании размеров окна используются бумажные единицы, такие, как “in” (дюймы) или “cm” (сантиметры). MapBasic по умолчанию использует дюймы; в программе MapBasic можно изменить единицы измерения, используя **Set Paper Units**. Оператор **Create Cartographic Legend** может переопределить единицы измерения, для этого надо включить подпредложение Units, используемое в предложениях **Position**, **Width** или **Height**.

Используйте предложение **ScrollBars** чтобы показать или скрыть линейку прокрутки в окне карты.

**Portrait** или **Landscape** описывают ориентировку разделов легенды в окне. Книжная ориентировка – это **Portrait**. Альбомная ориентировка – это **Landscape**.

Если указано **Custom**, то Вы можете задать Ваше собственное предложение **Position** для позиционирования раздела.

Предложение **Position** определяет позиционирование раздела, если определено предложение **Custom**.

Предложения **Position**, **Title**, **SubTitle**, **Border Pen** и **Style** для разделов легенды используются только для слоев карты. Они не используются для тематических слоев. Для тематических слоев вся необходимая информация задается автоматически при их создании.

Предложение **Font** определяет стиль текста. Если по умолчанию заголовок раздела, подзаголовок или имя стиля объекта имеют определенный шрифт, то этот же шрифт по умолчанию будет использоваться и для раздела легенды. Если на уровне раздела определены предложения, описывающие заголовок, подзаголовок и стиль объекта, и в этих предложениях используется предложение **Font**, то будет использоваться шрифт, заданный этим предложением. Если ни на каком уровне шрифт не определен, то будет применяться текущий шрифт с размером букв 10, 9 и 8 для заголовка, подзаголовка и имени стиля соответственно.

Предложение **Style** и ключевое слово **NoRefresh** позволяют Вам создавать собственные разделы легенды, которые не будут изменяться при обновлении легенды. Если ключевое слово **NoRefresh** используется в предложении **Style**, то в таблице не будет производиться поиск стилей. Вместо этого, предложение **Style** должно содержать Ваш список определений для стилей, которые отобразятся в разделе легенды. Это достигается заданием предложения **Text** и соответствующих предложений

Line, Region или Symbol.

**Пример:**

```
Include "C:\Program Files\MapInfo\MapBasic\mapbasic.def"

Open Table "C:\Program
Files\MapInfo\Professional\Data\USA\States.tab"
As states Interactive
Open Table "C:\Program
Files\MapInfo\Professional\Data\USA\City_125.tab"
As cities Interactive
Open Table "C:\Program
Files\MapInfo\Professional\Data\USA\Us_hiway.tab"
As hiway Interactive
Map From hiway, cities, states

Create Cartographic Legend
From Window FrontWindow()
Behind
Position (5, 5)
Width 5
Height 3
Window Title "Legend of cities,hiway,states Map"
ScrollBars On
Portrait
Default Frame Title "Legend Title"
Default Frame Subtitle "Legend SubTitle"
Default Frame Style "frame style"
Font ("Arial", 10, 12, BLACK, BLUE)
Default Frame Border Pen (2,50, BLACK)
Frame From Layer 1
Title "Layer1" Font ("Helv", 10, 20, YELLOW, BLUE)
SubTitle "Layer1 Sub"Font ("Helv", 3, 10, RED, 100)
Border Pen (3, 50, RED)
Style Font ("Arial", 30, 10, RED, YELLOW)
Text "layer1 style1" Line Pen (1, 10, BLACK)
Text "layer1 style2" Region Pen (1, 10, Blue)
Brush (5, 10, Red)
Text "layer1 style3" Symbol (35, RED, 36)
Frame From Layer 2
Title "Layer2"
```

## Оператор Create Cartographic Legend

---

```
SubTitle "Layer2 Sub"
Border Pen (2, 3, BLUE)
Style Font ("Arial", 10, 16, BLUE, RED) Norefresh
  Text  "layer2 style1" Line Pen (1, 100, Blue)
  Text  "layer2 style2" Region Pen (1, 10, RED)
  Brush (5, 10, BLUE)
  Text  "layer2 style3" Symbol (45, BLACK, 24)
Frame From Layer 3
  Title "Layer3"
  SubTitle "Layer3 Sub"
  Border Pen (1, 10, RED)
  Style Font ("Arial", 10, 10, BLACK, RED) Norefresh
    Text  "layer3 style1" Line Pen (2, 5, Red)
    Text  "layer3 style2" Region Pen (1, 10, YELLOW)
    Brush (5, 10, Red)
    Text  "layer3 style3" Symbol (52, Blue, 18)
```

### Смотри также:

[Set Cartographic Legend](#), [Alter Cartographic Frame](#), [Add Cartographic Frame](#), [Remove Cartographic Frame](#), [Create Legend](#), [Set Window](#), [WindowInfo\(\)](#)



## Оператор Create Cutter

### Назначение

Задаёт набор изменяемых объектов и набор полилиний в качестве выбранного объекта, далее оператор создаёт объект регион, который может использоваться как разрезающий объект для операции Разрезать, а также может использоваться в качестве нового набора изменяемых объектов, которые могут быть подмножеством исходных изменяемых объектов.

### Синтаксис

**Create Cutter Into Target**

### Описание

Перед использованием **Create Cutter**, должны быть выбраны один или более объектов полилиний и должен существовать изменяемый объект. Это можно сделать командой ОБЪЕКТЫ > ВЫБРАТЬ ИЗМЕНЯЕМЫЙ ОБЪЕКТ, или использованием оператора **Set Target**. Объекты полилинии содержащиеся в выборке должны быть представлены одним, непрерывным объектом, без разрывов и самопересечений.

Полилиния должна пересекать МОП (минимальный описывающий прямоугольник) изменяемого объекта, который в свою очередь должен соответствовать требованиям операции разрезания. Полилиния, таким образом, сама не пересекает изменяемый объект. Например, изменяемый объект представлен серией островов (Гавайских) и полилиния может использоваться для разделения островной гряды на 2 части, при этом не касаясь ни одного отдельного острова. Если МОП изменяемого объекта не пересекает полилинию, то тогда такой объект будет удален из списка изменяемых объектов.

После выделения изменяемых объектов, рассчитывается минимальный описывающий прямоугольник всех этих объектов, сам МОП представляет пространство, которое будет разделено. Затем полилиния продолжается, если необходимо, так что она попадает на территорию МОП. Это достигается так: берется направление между двумя последними точками на каждом конце полилинии в этом декартовом направлении протягивается полилиния, пока она не пересечется с МОП. Протянутая таким образом полилиния разделяет пространство изменяемых объектов на две части. В результате будет создан и возвращен объект Область, представляющий одну из этих двух частей.

Этот оператор будет возвращать обработанное множество изменяемых объектов и новый обрезанный объект Область. Этот объект область будет вставлен в таблицу с изменяемыми объектами (которая должна быть изменяемой). Исходный объект(ы) полилиния останется, но перестанет быть выделенным. Выделенным станет новый объект Область.

**Внимание:** Обрезанный объект останется в слое с изменяемыми объектами. Можно удалить обрезанный объект вручную из изменяемого слоя.

### Пример

```
Open Table "C:\MapInfo_data\TUT_USA\USA\STATES.TAB"
Open Table "C:\MapInfo_data\TUT_USA\USA\US_HIWAY.TAB"
Map from States, Us_hiway
select * from States where state = "NY"
Set target On
select * from Us_hiway where highway = "I 90"
Create Cutter Into Target
```

## Оператор Create Cutter

---

`Objects Split Into Target`

### Смотрите также

`Set Target`

## Функция CreateCircle( )

### Назначение:

Возвращает объект "окружность".

### Синтаксис:

**CreateCircle(*x, y, radius*)**

где

*x* – X-координата центра окружности (или широта), действительное число;

*y* – Y-координата центра окружности (или долгота), действительное число;

*radius* – действительное число, назначающее радиус окружности.

### Величина, полученная в результате:

Величина типа Object.

### Описание:

Функция **CreateCircle( )** возвращает графический объект типа "окружность".

Параметры *x* и *y* задают координаты центра окружности в той координатной системе, которая была объявлена MapBasic ранее. (Смотрите описание оператора **Set CoordSys.**) Если система не объявлялась, то координаты будут принимать значения широты и долготы. При этом надо учесть, что MapBasic игнорирует координатную систему самого окна Карты.

Параметр *radius* назначается в тех единицах измерения, которые были назначены MapBasic до выполнения этой функции. (Смотрите описание оператора **Set Distance Units.**) Если единицы не были объявлены, то радиус будет измеряться в милях.

Линия и заливка создаваемой окружности будут создаваться в соответствии с выбором стилей линии и штриховки в операторе **Set Style**, который выполняется до функции **CreateCircle( )**. Вы можете также воспользоваться для создания объекта типа "окружность" оператором **Create Ellipse**, в котором могут быть предложения **Pen** и **Brush**, для определения стилей линии и штриховки.

Графический объект, созданный функцией **CreateCircle( )**, может быть присвоен объектной переменной, которая задает значение для уже существующей строки таблицы (оператор **Update**) или вновь созданной (оператор **Insert**).

**Замечание:** Перед созданием объекта в окне Отчета не забудьте выполнить оператор **Set CoordSys Layout**.

### Ошибки:

Функция вернет код ошибки ERR\_FCN\_ARG\_RANGE, если значение аргумента выходит за пределы, заданные при его определении.

### Пример 1:

В примере используется оператор **Insert** для создания новой строки в таблице SITIES. Функция **CreateCircle( )** используется в теле оператора **Insert** для создания объекта "окружность", данные которого будут помещены в этой строке.

```
Open Table "sities.tab"
Insert Into sities (obj)
  Values ( CreateCircle(-72.5, 42.4, 20) )
```

## Функция CreateCircle( )

---

### Пример 2:

В следующем примере используется таблица TOWERS, которая имеет три колонки: "Xcoord", "Ycoord" и "Radius". Колонки "Xcoord" и "Ycoord" содержат значения долготы и широты, где находятся радиостанции, а колонка "Radius" – значения радиусов областей их вещания.

Оператор **Update** использует функцию **CreateCircle(\_)** для построения окружностей для каждой строки таблицы. После выполнения оператора каждой строке таблицы TOWERS будет присоединен соответствующий объект.

```
Open Table "towers"  
Update towers  
Set obj = CreateCircle(xcoord, ycoord, radius)
```

### Смотрите также:

**Create Ellipse, Insert, Update**

## Оператор Create Collection

### Назначение

Объединяет точечные, линейные и площадные объекты в один объект. Коллекция показывается в окне Отчета в виде одной записи.

### Синтаксис

```
Create Collection [ num_parts ]
    [ Into { Window window_id | Variable var_name } ]
Multipoint
    [ num_points ]
    ( x1, y1) ( x2, y2) [ ... ]
    [ Symbol ... ]
Region
    num_polygons
    [ num_points1 (x1, y1) (x2, y2) [ ... ] ]
    [ num_points2 (x1, y1) (x2, y2) [ ... ] ... ]
    [Pen ... ]
    [ Brush ... ]
    [ Center ( center_x, center_y ) ]
Pline
    [ Multiple num_sections ]
    num_points
    ( x1, y1) ( x2, y2) [ ... ]
    [ Pen ... ]
    [ Smooth ... ]
```

*num\_parts* - число непустых частей в коллекции. Это число от 0 до 3 является дополнительным для кода MapBasic (это обязательно для MIF).

### Пример

```
create collection multipoint 2 (0,0) (1,1) region 3 3 (1,1) (2,2) (3,4)
4 (11,11) (12,12) (13,14) (19,20) 3 (21,21) (22,22) (23,24) pline 3
(-1,1) (3,-2) (4,3)

dim a as object
create collection into variable a multipoint 2 (0,0) (1,1) region 1 3
(1,1) (2,2) (3,4) pline 3 (-1,1) (3,-2) (4,3)
insert into test (obj) values (a)

create collection region 2 4 (-5,-5) (5,-5) (5,5) (-5,5) 4 (-3,-3) (3,-
3) (3,3) (-3,3) pline multiple 2 2 (-6,-6) (6,6) 2 (-6,6) (6,-6)
multipoint 6 (2,2) (-2,-2) (2,-2) (-2,2) (4,1) (-1,-4)
```

### Смотрите также

Оператор Create Multipoint

# Оператор Create Ellipse

### Назначение:

Создает эллипсы и окружности.

### Синтаксис:

#### Create Ellipse

```
[ Into { Window window_id | Variable var_name } ]  
(x1, y1) (x2, y2)  
[ Pen... ]  
[ Brush... ]
```

где

*window\_id* – идентификатор окна, целое число;

*var\_name* – имя объектной переменной;

*x1*, *y1* – координаты одного угла прямоугольника, описывающего эллипс;

*x2*, *y2* – координаты противоположного по диагонали угла прямоугольника, описывающего эллипс;

**Pen** – слово, с которого начинается стандартное предложение для назначения стиля линии объекта;

**Brush** – слово, с которого начинается стандартное предложение для назначения стиля штриховки объекта.

### Описание:

Результатом действия оператора **Create Ellipse** является новый объект типа "эллипс". MapBasic создает объект, вписывая его в прямоугольник, задаваемый координатами двух противоположных углов. Такой прямоугольник называется минимальным прямоугольным покрытием объекта (МПП). Если оператор задает квадрат, то будет создана окружность, иначе – эллипс.

Если оператор включает конструкцию **Into Variable**, то созданный объект будет значением объектной переменной *var\_name*. Если одним из параметров предложения **Into** указывается окно, объект помещается на подготовленное место в окне (например, на изменяемый слой). Если предложения **Into** вообще нет в операторе, MapBasic попытается создать эллипс в самом верхнем окне. Если это невозможно (например, поверх всех окон лежит окно Графика), то объект не будет создан.

Параметры *x* и *y* являются координатами в той системе координат, которая была объявлена до оператора **Create Ellipse**. (Смотрите описание оператора **Set CoordSys**.) Если система не объявлялась, то координаты будут принимать значения широты и долготы. При этом надо учесть, что MapBasic игнорирует координатную систему самого окна Карты. Если объект создается для окна Отчета, параметры *x* и *y* – координаты точки на листе в объявленных ранее единицах измерения листа. (Смотрите описание оператора **Set Paper Units**.) Так, X-координата – это расстояние от левого края листа до точки и Y-координата – расстояние от верхнего края листа.

Предложения **Pen** и **Brush** назначают стиль линии и штриховки объекта. Если в операторе не участвует предложение **Pen**, оператор **Create Rect** использует установку соответствующего режима для стиля линии в MapInfo. Стиль линии можно изменить командой НАСТРОЙКА > Стиль линий. (Аналогично тому, что предложению **Brush** в MapInfo соответствует команда НАСТРОЙКА > Стиль ОБЛАСТЕЙ.)

### Смотрите также:

**Brush**, **CreateCircle( )**, **Insert**, **Pen**, **Update**

## Оператор Create Frame

### Назначение:

Создает новый объект "рамка" в окне Отчета.

### Синтаксис:

#### Create Frame

```
[ Into { Window layout_win_id | Variable var_name } ]
(x1, y1) (x2, y2)
[ Pen ... ]
[ Brush ... ]
[ Title title ]
[ From Window contents_win_id ]
[ FillFrame { On | Off } ]
```

где

*x1, y1* – координаты одного угла рамки;

*x2, y2* – координаты другого угла рамки;

*layout\_win\_id* – идентификатор окна Отчета, целое число;

*var\_name* – имя объектной переменной;

*title* – строка, задающая заголовок окна, изображение из которого будет помещено в рамку (не имеет смысла использовать, если в операторе используется предложение **From Window**);

*contents\_win\_id* – идентификатор окна, изображение из которого будет помещено в рамку, целое число.

Слово **Pen** начинает стандартное предложение для назначения стиля линии объекта;

Слово **Brush** начинает стандартное предложение для назначения стиля штриховки объекта.

### Описание:

Результатом действия оператора **Create Frame** является новый объект типа "рамка" в окне Отчета.

**Замечание:** Перед созданием объекта в окне Отчета не забудьте выполнить оператор **Set CoordSys Layout**.

MapInfo запомнит установки в окне Отчета, если поместить оператор **Create Frame** в файл Рабочего Набора. Файл Рабочего Набора является текстовым, и его можно открыть и отредактировать в окне MapBasic.

Предложение **Pen** диктует стиль линии контура рамки, а предложение **Brush** задает стиль раскраски фона рамки.

Если оператор использует предложение **Into Variable**, то созданный объект будет значением объектной переменной. Если предложением **Into Window *layout\_win\_id*** указывается окно Отчета, объект помещается в это окно. Если предложения **Into** вообще нет в операторе, MapBasic попытается создать рамку в самом верхнем окне; если это невозможно (например, окно Отчета не лежит поверх всех окон), то объект не будет создан. Перед созданием объекта в окне Отчета необходимо выполнить оператор **Set CoordSys Layout**.

Предложение **From Window** используется для задания окна, изображение из которого будет показано в рамке. Для этого необходимо знать идентификатор окна. Окно должно быть уже открыто в MapInfo.

Предложение **Title** является альтернативой предложению **From Window** и тоже задает окно,

## Оператор Create Frame

---

изображение из которого будет показано в рамке. Для задания используется заголовок окна. Например, для окна Карты в котором показаны данные таблицы WORLD, предложение будет такое – **Title "WORLD Map"**. Если нет окна для рамки, то параметр *title* должен быть равен пустой строке (""). Оператор создаст пустой объект.

Если Вы включили в оператор сразу два предложения **Title** и **From Window**, то только последнее из них подействует.

При создании рамки для окна Карты, можно применять предложение **FillFrame**, чтобы управлять заполнением рамки Картой: задание **FillFrame On** помещает всю Карту в рамку, **FillFrame Off** (или отсутствие предложения **FillFrame**) задает режим, в котором помещение рамки в окно зависит от соотношения высоты и ширины окна и рамка может не показывать Карту полностью. Эти режимы управляются флажком в диалоге “Рамка”.

### Смотрите также:

**Brush, Insert, Layout, Pen, Set CoordSys, Set Layout, Update**



## Оператор Create Grid

Тематическая растровая поверхность это непрерывный растровый грид, который создается при интерполяции точечных данных. Оператор **Create Grid** берет данные из колонки, которые находятся в таблице с точечными данными и посылает эти точки и их числовые значения в интерполятор. Интерполятор создает растровый грид-файл, который MapBasic отображает в виде растровой таблицы в окне карты.

Оператор **Create Grid** считывает значения (x, y, z) из таблицы, определенной в предложении From. Он получает значения z, которые указываются в предложении With, относящемуся к данной таблице.

Размеры грида (сетки) могут быть определены двумя способами. Первый способ определяет размер ячейки грида, выраженной в единицах расстояния, например, милях. Другой способ заключается в задании количества ячеек грида по ширине и высоте. Например, если Вы хотите получить грид размером не менее 200 ячеек по ширине и 200 ячеек по высоте, то надо будет определить "cell min 200". В зависимости от площади, покрываемой гридом, действительный размер грида будет не менее 200 на 200.

### Синтаксис:

```
Create Grid From tablename With expression [ Ignore value_to_ignore ] Into filespec [ Type
grid_type ] [ Coordsys ... ] [ Clipping { Object obj } | { Table tablename } ] Inflect num_inflections at [
By Percent ] color : inflection_value [ color : inflection_value ... ]
[ Round rounding_factor ] { [ Cell Size cell_size [ Units distance_unit ] ] | [ Cell Min n_cells ] } [ Border
numcells ] Interpolate With interpolator_name Version version_string Using num_parameters
parameter_name : parameter_value [ parameter_name : parameter_value ... ]
```

*tablename* - это "псевдоним" имени открытой таблицы, из которой берутся точки для расчета поверхности (грида).

*expression* - это выражение, которое выделяет необходимую часть таблицы, например, имя колонки.

*value\_to\_ignore* - это значение, которое будет проигнорировано; это всегда ноль. Грид не будет создаваться для строки, если значение в этой строке совпадает со значением, которое игнорируется.

*filespec* определяет полный путь и новое имя грид-файла. У этого нового файла будет расширение .MIG.

*grid\_type* - это строковое выражение, указывающее какой тип файла будет создан. По умолчанию это .MIG файл.

*Coordsys* - это предложение, определяющее, какая координатная система будет использована при создании грида. Если это предложение не используется, то грид файл будет иметь ту же систему координат, что и исходная таблица. Смотрите описание предложения **Coordsys** для более полной информации.

*obj* - это объект, который обрезает ячейки грида. Только часть ячеек внутри такого объекта будет отображена. Если ячеек грида внутри такого объекта нет, то никаких значений для ячеек записано не будет, и запишутся ячейки с нулевыми значениями.

*tablename* - это имя таблицы, содержащей объекты типа полигонов, которые будут объединены в единый полигон и будут использованы для обрезания ячеек грида.

*num\_inflections* - это числовое выражение, определяющее число переломных точек в шкале значений

## Оператор Create Grid

---

и цвета.

*color* - это выражение для обозначения цвета для переломной точки.

*inflection\_value* - это числовое выражение, определяющее величину цвет:числовое значение для переломной точки.

*cell\_size* - это числовое выражение, определяющее размер ячейки грида в единицах расстояния.

*n\_cells* - это числовое выражение, которое определяет высоту и ширину грида в количестве ячеек.

*numcells* - определяет число ячеек, которое будет добавлено вокруг границы грида, увеличивая размер грида сверху, снизу, слева и справа.

*distance\_unit* - это строковое выражение, определяющее единицы измерения размера ячейки. Если эта величина не задана, то используются единицы координатной системы из обрабатываемой таблицы.

*interpolator\_name* - это строковое выражение, определяющее имя интерполятора, который используется для создания грида.

*version\_string* - это строковое выражение, определяющее версию интерполятора.

*num\_parameters* - это числовое выражение, определяющее число параметров интерполятора - число пар имя\_параметра : значение.

*parameter\_name* - это строковое выражение, определяющее имя в паре имя\_параметра : значение.

*parameter\_value* - это числовое выражение значение в паре имя\_параметра : значение.

*By Percent* - это строковое выражение, определяющее имя, для пары имя-значение.

*Round* - это числовое выражение, для пары имя-значение.

### Пример:

```
Open Table "C:\States.tab" Interactive
Map From States
Open Table "C:\Us_elev.tab" Interactive
Add Map Auto Layer Us_elev
set map redraw off
Set Map Layer 1 Display Off
set map redraw on

create grid
  from Us_elev
  with Elevation_FT
  into "C:\Us_elev_grid"
  clipping table States
  inflect 5 at
    RGB(0, 0, 255) : 13
    RGB(0, 255, 255) : 3632.5
    RGB(0, 255, 0) : 7252
    RGB(255, 255, 0) : 10871.5
    RGB(255, 0, 0) : 14491
```

```
cell min 200
interpolate
  with "IDW" version "100"
  using 4
    "EXPONENT": "2"
    "MAX POINTS": "25"
    "MIN POINTS": "1"
    "SEARCH RADIUS": "100"
```

**Смотри также:**

[Set Map](#)

## Оператор Create Index

### Назначение:

Создает индекс для колонки в открытой таблице.

### Синтаксис:

**Create Index On table (column)**

где

*table* – имя открытой таблицы;

*column* – имя колонки в открытой таблице.

### Описание:

Оператор **Create Index** создает индекс для определенной колонки открытой таблицы. MapInfo использует индексированные колонки в команде ЗАПРОС > НАЙТИ и других. Индексы также улучшают выполнение запросов.

**Замечание:** MapInfo не может индексировать колонки в таблице, в которой есть не сохраненные изменения. Перед индексированием сохраните Вашу таблицу на диск (например, оператором **Commit**).

### Пример:

Создается индекс колонки "Столица" в таблице WORLD:

```
Open Table "World"
```

```
Create Index on World(Столица)
```

### Смотрите также:

**Alter Table, Create Table, Drop Index**

## Оператор Create Legend

### Назначение:

Открывает новое окно Легенды, для определенного окна Карты или Графика.

### Синтаксис:

#### Create Legend

```
[ From Window window_ID ]  
[ { Show | Hide } ]
```

где

*window\_ID* – целочисленный идентификатор открытого в рабочем окне MapInfo окна Карты или Графика.

### Описание:

Этот оператор создает дополнительное окно "Легенда" в добавок к стандартному окну легенды в рабочем окне MapInfo. Последнее Вы можете открыть оператором **Open Window Legend**.

Оператор **Create Legend** может быть Вам полезен, если необходимо вывести на экран Легенду Карты, когда окно Карты открыто, но не активно. Этот оператор полезен в приложениях, использующих т.н. "интегрированную картографию", в которых окно MapInfo вставляется в другую программу, например, созданную при участии Visual Basic. Концепция и описание интегрированной картографии содержатся в 12 главе *Руководства пользователя MapBasic*.

Если Вы добавите предложение **From Window**, новая легенда считается порожденной определенным в этом предложении окном; иначе новое окно Легенды считается порожденным последним активным окном Карты.

Если в операторе используется ключевое слово **Hide**, то окно будет создано в скрытом состоянии. Вывести на экран скрытое окно Вы можете использовав оператор **Set Window ... Show**.

После выполнения оператора **Create Legend** определить идентификатор нового окна можно будет вызовом функции **WindowID(0)**. Этим идентификатором Вы можете пользоваться в следующих операторах (таких как **Set Window**).

Новая легенда создается в соответствии с режимами наследования и стилизации, заданными оператором **Set Next Document**.

### Смотрите также:

**Open Window, Set Next Document, Create Cartographic Legend**

### Функция CreateLine( )

#### Назначение:

Возвращает объект типа "прямая линия".

#### Синтаксис:

**CreateLine( *x1* , *y1* , *x2* , *y2* )**

где

*x1* – X-координата начальной точки линии (или долгота), действительное число;

*y1* – Y-координата начальной точки линии (или широта), действительное число;

*x2* – X-координата конечной точки линии, действительное число;

*y2* – Y-координата конечной точки линии, действительное число.

#### Величина, полученная в результате:

Величина типа Object.

#### Описание:

Функция возвращает объект типа "прямая линия".

Параметры *x* и *y* задают координаты концов отрезка прямой линии в той координатной системе, которая была объявлена MapBasic ранее. (Смотрите описание оператора **Set CoordSys**.) Если система не объявлялась, то координаты будут принимать значения широты и долготы. При этом надо учесть, что MapBasic игнорирует координатную систему самого окна Карты.

Функция при создании объекта будет использовать стиль линии, назначаемый оператором **Set Style**, который выполняется до функции **CreateLine( )**. Вы можете воспользоваться для создания объекта "прямая линия" оператором **Create Line**, в котором есть предложение **Pen** для задания стиля линии.

Графический объект, созданный функцией **CreateLine( )**, может быть присвоен объектной переменной, которая определяет значение уже существующей строки в таблице (оператор **Update**) или вновь созданной (оператор **Insert**).

**Замечание:** Перед созданием объекта в окне Отчета не забудьте выполнить оператор **Set CoordSys Layout**.

#### Пример:

В примере используется оператор **Insert** для создания новой строки в таблице ROUTES. Функция **CreateLine( )** используется в теле оператора **Insert** для создания объекта, данные которого будут помещены в этой строке.

```
Open Table "Routes.tab"  
Insert Into routes (obj)  
Values (CreateLine(-72.55, 42.431, -72.568, 42.435))
```

#### Смотрите также:

Create Line, Insert, Update

## Оператор Create Line

### Назначение:

Создает объект "прямая линия".

### Синтаксис:

#### Create Line

```
[ Into { Window window_id | Variable var_name } ]  
(x1, y1) (x2, y2)  
[ Pen... ]
```

где

*window\_id* – идентификатор окна, целое число;

*var\_name* – имя объектной переменной;

*x1, y1* – координаты начала отрезка прямой;

*x2, y2* – координаты конца отрезка прямой линии.

Слово **Pen** начинает стандартное предложение для назначения стиля линии объекта.

### Описание:

Результатом действия оператора **Create Line** является новый графический объект типа "прямая линия".

Если оператор включает предложение **Into Variable**, то созданный объект будет значением объектной переменной. Если параметр после **Into** указывает окно, то прямая помещается на подготовленное место в окне (например, в изменяемый слой). Если предложения **Into** вообще нет в операторе, MapBasic попытается создать прямую в самом верхнем окне. Если это невозможно (например, активно окно Графика), то объект не будет создан.

Параметры *x* и *y* являются координатами в той системе координат, которая была объявлена до оператора **Create Line**. (Смотрите описание оператора **Set CoordSys**.) Если система не объявлялась, то координаты будут принимать значения широты и долготы. При этом надо учесть, что MapBasic игнорирует координатную систему самого окна Карты.

Если объект создается для окна Отчет, параметры *x* и *y* – координаты на листе в объявленных ранее единицах измерения листа. (Смотрите описание оператора **Set Paper Units**.) Так X-координата – это расстояние от левого края листа до точки, и Y-координата – расстояние от верхнего края листа.

**Замечание:** Перед созданием объекта в окне Отчета не забудьте выполнить оператор **Set CoordSys Layout**.

Если в операторе нет предложения **Pen**, оператор **Create Line** использует установку соответствующего режима для стиля линии в MapInfo (стиль линии можно изменить командой НАСТРОЙКА > Стиль линий).

### Смотрите также:

CreateLine( ), Insert, Pen, Update

### Оператор Create Map

#### Назначение:

Изменяет структуру существующей таблицы, разрешая сопоставлять ее записям графические объекты.

#### Синтаксис:

```
Create Map
  For table
    [ CoordSys... ] Using from_table
table имя открытой таблицы
CoordSys... предложение CoordSys, начинает стандартное предложение для назначения
координатной системы.
```

#### Описание:

Оператор **Create Map** присоединяет географические объекты к открытой таблице, после чего их можно видеть в окне Карты.

Этот оператор не открывает новое окно Карты. Чтобы открыть новое окно Карты используйте оператор **Map**.

Не надо применять оператор **Create Map** к таблице которая уже имеет присоединенные географические объекты; поступая так, вы удалите все географические объекты из таблицы. Если таблица уже имеет прикрепленные географические объекты, и Вам надо постоянно изменять проекцию карты, используйте оператор **Commit Table As**. С другой стороны, если надо временно изменить проекцию, в которой отображается карта, используйте оператор **Set Map** с предложением **CoordSys**. Оператор **Create Map** не работает со связанными таблицами. Чтобы присоединить к связанной таблице географические объекты, используйте оператор **Server Create Map**.

#### Определение системы координат

Используйте один из следующих методов для задания системы координат:

Используйте имя уже открытой таблицы с географическими объектами как часть *from\_table* предложения **Using**. В этом случае, используемая система координат будет такой же как и используемая в *from\_table*. Параметр *from\_table* должен быть уже открытой таблицей, причем с географическими объектами, иначе появится сообщение об ошибке.

Точно примените информацию о системе координат используя предложение **CoordSys**.

Если Вы пропускаете и предложение **CoordSys** и предложение **Using**, то таблица будет использовать текущую систему координат MapBasic.

Обратите внимание на то, что предложение **CoordSys** влияет на точность карты. Предложение **CoordSys** включает в себя предложение **Bounds**, которое устанавливает допустимые значения минимальных и максимальных координат, которые могут быть на карте. Если предложение **Bounds** пропущено, MapInfo Professional использует стандартные значения, охватывающие всю Землю (в этом случае, координаты имеют точность - миллионную часть градуса, или приблизительно 4 дюйма). Если у Вас имеется информация, что карта создана в ограниченном районе, то можно увеличить точность координат карты, задав ее границы. Полностью со синтаксисом предложения **CoordSys** можно в описании предложения **CoordSys**.



### Смотрите также

[Commit](#), [CoordSys](#), [Create Table](#), [Drop Map](#), [Map](#), [Server Create Map](#), [Set Map](#)

### Оператор Create Map3D

#### Назначение:

Создает 3D карту с определенными параметрами.

#### Синтаксис:

```
Create Map3D
[ From Window window_id | MapString mapper_creation_string ]
[ Camera [ Pitch angle | Roll angle | Yaw angle | Elevation angle ] |
    [ Position (x,y,z) | FocalPoint (x,y,z) ] |
    [ Orientation (vu_1, vu_2, vu_3, vpn_1, vpn_2, vpn_3, clip_near, clip_far) ] ]
[ Light [ Position (x,y,z) | Color lightcolor ] ]
[ Resolution (res_x, res_y) ]
[ Scale grid_scale ]
[ Background backgroundcolor ]
[ Units unit_name ]
```

*window\_id* - идентификатор окна карты, содержащего слой поверхности. Если слой поверхности не найден, отображается сообщение об ошибке.

*mapper\_creation\_string* - определяет командную строку, которая создает текстуру (отображает объекты векторных слоев) на поверхности.

**Camera** определяет позицию и ориентацию камеры.

*angle* - угол в градусах. Горизонтальный угол в пределах от 0-360 градусов определяет поворот карты вокруг центральной точки поверхности. Вертикальный угол в диапазоне от 0-90 определяет возвышение начальной точки поверхности.

**Pitch** - угловая величина определяет текущее положение точки наблюдения по оси X

**Roll** - угловая величина определяет текущее положение точки наблюдения по оси Z

**Yaw** - угловая величина определяет текущее положение точки наблюдения по оси Y

**Elevation** - определяет поворот камеры (находящейся в точке фокуса) относительно оси X.

**Position** - определяет положение точки наблюдения/источника света.

**FocalPoint** – определяет положение точки наблюдения/точки фокуса.

**Orientation** - определяет ориентацию камеры ViewUp, ViewPlane Normal и Clipping Range (фиксированное отображение).

**Resolution** - количество ячеек поверхности (по X, Y). Эти значения могут быть увеличены до максимального разрешения поверхности (максимального размера ячейки -x,y ). Если поверхность 200x200, то значения разрешения могут лежать только в пределах 200x200.

*Grid scale* - определяет вертикальный масштаб (в направлении Z-координаты). Значение >1 подчеркивает рельеф, значение <1 снижает топологические особенности (в направлении Z-координаты).

*backgroundcolor* - цвет, используемый для фона и определяемый функцией RGB.

**Units** - определяет единицы измерения точек поверхности. Не указывайте их для поверхности, созданной на основании данных температуры или плотности. Этот параметр необходимо указывать во время создания 3D Карты. Если существуют единицы измерения для значений точек поверхности, они должны быть определены во время создания 3D Карты, так как Вы не сможете изменить(настроить) их позже с использованием диалога "Свойства 3D Карты".

### Описание:

В результате создается автономное окно 3D Карты, которое связано с данными тех таблиц (в окне Карты), на основании которых оно было создано. Т.е. если эти таблицы будут изменены и окно 3D Карты будет обновлено (или восстановлено из рабочего набора), все изменения отобразятся на трехмерной поверхности.

Ошибка создания будет иметь место в том случае, если window\_id - не окно карты или карта не содержит слой поверхности. Если окно карты содержит несколько поверхностей, каждое будет представлено в окне 3D Карты.

3D Карта хранит информацию (строку) об окне карты для генерации текстуры. Эта строка будет также "преобладать" в Рабочем наборе, если окно 3D Карты присутствует в данном Рабочем наборе.

### Пример:

```
Create Map3D Resolution(75,75)
```

Создает окно 3D карты для последнего окна Карты. Операция не будет выполнена, если окно карты не содержит слоя поверхности. Другой пример:

```
Create Map3D From Window FrontWindow() Resolution(100,100) Scale 2 Back-  
ground RGB(255,0,0) Units «ft».
```

Создает окно 3D карты с красным фоном, единицы по оси z -футы, масштабный коэффициент - 2, и разрешение 100x100.

### Смотрите также:

**Оператор Set Map3D**

# Оператор Create Menu

### Назначение:

Создает новое меню или переопределяет уже существующее.

### Синтаксис (вариант 1):

```
Create Menu newmenuname [ ID menu_id ] As  
  menuitem [ ID menu_item_id ] [ HelpMsg help ] { Calling handler | As menuname }  
  [, menuitem ... ]
```

### Синтаксис (вариант 2):

```
Create Menu newmenuname As Default
```

где

*newmenuname* – заголовок нового меню или стандартное имя переопределяемого меню, тип String;

*menuitem* – имя элемента списка меню, тип String;

*menu\_id* – идентификатор стандартного элемента меню, число типа SmallInt ID от 1 до 15;

*menu\_item\_id* – целочисленный идентификатор созданного элемента меню;

*menuname* – заголовок меню, которое будет включено в список как подменю;

*help* – текст, который будет показываться в строке сообщений при указании на элемент в списке меню;

*handler* – либо имя процедуры-обработчика, либо код стандартной команды MapInfo, либо строка специального синтаксиса для управления событиями в системе меню через механизмы OLE или DDE; см. ниже “Предложение Calling. Если Вы задаете код команды меню MapInfo (такой как M\_WINDOW\_STATISTICS), то аргумент *menuitem* должен начинаться с восклицательного знака и содержать внутри символ каретки (^), чтобы удовлетворять синтаксису команд типа ПОКАЗАТЬ/СКРЫТЬ\_.

### Описание:

Если параметр *newmenuname* является именем одного из существующих меню (например, "File"), то оператор **Create Menu** переопределяет это меню. Если параметр *newmenuname* имеет какое-то уникальное значение, то оператор **Create Menu** создает новое меню. Список стандартных имен меню MapInfo приводится в описании оператора **Alter Menu**.

Оператор **Create Menu** не показывает новое или переопределяемое меню. Для вывода меню на экран используются операторы **Alter Menu Bar** и **Create Menu Bar**. Однако, если оператор **Create Menu** переопределяет меню, которое есть в строке меню, то изменения в меню отображаются немедленно.

**Замечание:** MapInfo может одновременно поддерживать не более 48 определений меню, включая стандартные в MapInfo ("Файл" и так далее). Число это не зависит от количества меню, показываемых в данный момент на экране.

В параметре *menuitem*, который определяет имя элемента меню, могут использоваться специальные управляющие символы, определяющие состояние данного элемента меню (например, доступность команды в меню). Эти символьные последовательности должны задаваться в начале имени элемента. Смотрите таблицу ниже.

Следующие знаки являются зарезервированными и выполняют специальную роль: прямой слэш (/), обратный слэш(\) и знак меньше (<). Если Вы хотите использовать эти знаки в тексте меню или строке подсказки, то необходимо поставить перед зарезервированным символом обратный слэш.

Например, для помещения в меню "Данные" такой команды как "Клиент\Сервер", надо:

```
Create Menu "Data" As
"Клиент\Сервер" Calling cs_proc
```

Если текст параметра *menuitem* начинается с символа @, то пользовательское меню делится на две колонки. Такой элемент помещается первым во вторую колонку.

### Обработчики для новых команд меню

Если параметр *menuitem* задает команду, то в предложении **Calling handler** может быть задан либо обработчик элемента меню, либо код стандартной команды MapInfo (например, код M\_FILE\_SAVE соответствует команде ФАЙЛ > СОХРАНИТЬ).

Если для элемента определен обработчик и если пользователь выберет в меню этот элемент, то Map-Basic автоматически вызовет процедуру-обработчик. Процедурой-обработчиком меню является заранее объявленная sub-процедура без атрибутов. Если для элемента меню определен код, то Ваша программа должна в начале иметь оператор **Include "MENU.DEF"**, подключающий файл стандартных определений для кодов команд.

С помощью необязательного предложения **ID** Вы можете задать элементу меню уникальный идентификатор. Вы можете связать один и тот же обработчик с разными элементами меню. В этом случае идентификатор может оказаться полезным для определения, какой именно командой была вызвана эта процедура. Определить идентификатор элемента меню из процедуры-обработчика, которую он вызвал, можно функцией **CommandInfo(CMD\_INFO\_MENUITEM)**. Идентификатор элемента меню используется также в операторе **Alter Menu Item**.

Если за параметром *menuname* не задано ни команды, ни подменю, ни вызова процедуры *handler*, то соответствующий элемент меню будет неактивным. То есть при указании на этот элемент ничего не будет происходить. Такие элементы в меню выполняют косметическую роль (например, горизонтальная линия, разделяющая список меню на части).

### Создание иерархически-подчиненного меню

#### Свойства элементов меню

Элемент меню может иметь разные состояния. Команда в меню может быть недоступной для выбора пользователем. В списке меню такие команды показываются серым шрифтом. Элемент меню может также выполнять роль переключателя режима. Такая команда при выборе снабжается специальным маркером (в Windows это галочка). При повторном выборе маркер убирается, а установленный режим выключается. Как уже упоминалось выше, параметр *menuitem* может включать в себя специальные управляющие последовательности:

Управляющий код	Эффект
(-)	Горизонтальная разделительная линия; такой элемент не может иметь обработчик. Пример: " (- "
(	Элемент меню недоступен для выбора. Пример: "(Закреть"

## Оператор Create Menu

---

(\$	Код для представления в меню ФАЙЛ четырех последних открывавшихся файлов. Его можно применять в системе меню один раз и его нельзя использовать в быстрых меню. Чтобы удалить список последних открывавшихся файлов из меню ФАЙЛ, либо удалите этот код из файла MAPINFOW.MNU, либо переопределите меню ФАЙЛ оператором <b>Create Menu</b> .
(>	Код для представления в меню списка открытых окон. Его можно применять в системе меню один раз.
!	Элемент меню при выборе может снабжаться галочкой, но на момент создания галочка отсутствует. Пример: <b>"!Показывать предупреждения"</b>
! ... ^ ...	Синтаксис для задания названия меню с текстом, зависящим от ситуации. Текст после восклицательного знака сменяется на текст после символа "^". Поведение созданного подобным образом элемента похоже на поведение следующего (текст меню снабжается галочкой), однако вместо галочки показывается первый текст, а вместо отсутствия галочки показывается второй текст. Пример: <b>"!Скрыть строку сообщений^Показать строку сообщений"</b>
!+	Элемент меню при выборе может снабжаться галочкой и на момент создания галочка присутствует. Пример: <b>"!+Показывать предупреждения"</b>

В системе Macintosh элементы могут иметь разное написание, то есть для команд может использоваться жирный шрифт, курсив, подчеркнутый, контурный и оттененный шрифты. Для достижения этого эффекта используются следующие управляющие символы в параметре *menuname*:

Управляющий код	Эффект (только для Macintosh)
<B	Жирный шрифт. Пример: <b>"Закреть&lt;B"</b>
<I	Курсив.
<U	Подчеркнутый шрифт.
<O	Контурный шрифт.
<S	Оттененный шрифт.

Заметим, что эти коды ставятся в конце имени элемента, но перед кодами, задающими клавишное сокращение.

### Задание клавишных сокращений

Элементам меню могут быть назначены клавишные сокращения (акселераторы), то есть пользователь может вызвать команду с клавиатуры без использования мыши. Клавишные сокращения бывают двух типов.

Клавишные сокращения первого типа имеют и имена меню, и имена элементов меню. Для того, чтобы вызвать элемент меню, воспользовавшись его клавишным сокращением, надо сначала открыть меню, используя сокращение меню или, как обычно, мышь. Например, для того, чтобы открыть окно Карты, в рабочем окне Map-Info пользователь нажимает сначала клавишное сокращение ALT+O, которым открывается список команд меню ОКНО, и только потом клавишу К (или ALT+K) для запуска команды Карта. На экране в именах меню и команд необходимые литеры подчеркнуты.

Для того, чтобы определить буквы для клавишного сокращения в меню и элементах меню, надо в текстах параметров *newmenuname* и *menuitem* использовать знак амперсанда (&) непосредственно перед нужной буквой. Например, текст "&Карта" задает команду Карта.

Другой тип клавишных сокращений позволяет вызвать команду немедленно, не открывая меню, где она находится. Например, для вызова вышеупомянутой команды КАРТА достаточно только нажать клавишу F3. Эти сокращения Вы можете увидеть в списке меню справа от команд. Управляющие коды, используемые для задания клавишного сокращения второго типа, приведены в следующей таблице.

#### Управляющий код

#### Эффект

Для Windows:

*/W {letter| %number}*

Задаёт клавишное сокращение для Windows; команда активизируется клавишей *letter* или клавишей, заданной ее числовым кодом *number*.

Например: записи "Данные/WД" и "Данные/W%196" равносильны.

*/W# {letter| %number}*

Задаёт клавишное сокращение для Windows; команда активизируется клавишей *letter* или клавишей, заданной ее числовым кодом *number*, с нажатой клавишей SHIFT.

Например: записи "Данные/W#Д" и "Данные/W#%196" равносильны.

*/W@ {letter| %number}*

Задаёт клавишное сокращение для Windows; команда активизируется клавишей *letter* или клавишей, заданной ее числовым кодом *number*, с нажатой клавишей ALT.

Например: записи "Данные/W@Д" и "Данные/W@%196" равносильны.

*/W^ {letter| %number}*

Задаёт клавишное сокращение для Windows; команда активизируется клавишей *letter* или клавишей, заданной ее числовым кодом *number*, с нажатой клавишей CTRL.

Например: записи "Данные/W^Д" и "Данные/W^%196" равносильны.

Для Macintosh:

*/Mletter* Задаёт клавишное сокращение для Macintosh; команда активизируется клавишей *letter* с нажатой клавишей COMMAND.

Например: "Данные/МД".

В Windows для использования клавиш *F<sub>n</sub>* используется возможность задавать клавишные сокращения численным кодом (*%number*). Так, для клавиши F1 численным кодом является 112, для

## Оператор Create Menu

F2 – 113 и так далее.

Заметим, что оператор **Create Menu Bar As Default** убирает и переопределяет *все* пользовательские меню, заданные оператором **Create Menu**. Но если Вам надо вернуть к стандартному виду только один список меню, а не все, то используйте оператор **Create Menu *menuname* As Default**.

### Предложение Calling

Предложение **Calling** задает, что должно случиться, если пользователь в меню выполнит команду, построенную приложением. Приведем возможные примеры использования этого предложения:

Примеры	Описание
Calling M_FILE_NEW	Если за словом <b>Calling</b> идет целочисленный код из файла MENU.DEF, MapInfo запускает на выполнение соответствующую стандартную команду MapInfo (например, Файл > Новая таблица).
Calling my_procedure	Если задано имя sub-процедуры, MapInfo передает управление этой процедуре.
Calling OLE "methodname"	Только для Windows. MapInfo управляет событиями, обращаясь с именем метода к объекту OLE Automation, установленному SetCallback-методом MapInfo. Детали смотрите в 12 главе <i>Руководства пользователя MapBasic</i> .
Calling DDE "server","topic"	Только для Windows. MapInfo управляет событиями, присоединяясь через DDE к паре "сервер тема" и посылая выполняемое сообщение на DDE-сервер.

В последних двух случаях строка послания для OLE или DDE сервера должна начинаться с трех символов "MI:" для того, чтобы сервер смог определить, что послание пришло от MapInfo. Остальная часть этой строки состоит из разделенного запятыми списка значений возврата функций с **CommandInfo(1)** по **CommandInfo(8)**. Детали читайте в 12 главе *Руководства пользователя MapBasic*.

### Пример 1:

Оператор **Create Menu** создает пользовательское меню ДАННЫЕ, которое добавляется в строку меню оператором **Alter Menu Bar** до меню Окно (ID 6) и меню справка (ID 7):

```
Declare Sub Main
Declare Sub addsub
Declare Sub editsub
Declare Sub delsub

Sub Main
  Create Menu "Данные" As
    "Добавить" Calling addsub,
    "Правка" Calling editsub,
    "Удалить" Calling delsub
```



```
Alter Menu Bar Remove ID 6, ID 7
Alter Menu Bar Add "Данные", ID 6, ID 7
End Sub
```

### Пример 2:

В этом фрагменте оператор **Create Menu** создает сокращенную версию меню ФАЙЛ. Управляющий символ (делает команды ЗАКРЫТЬ, СОХРАНИТЬ и ПЕЧАТАТЬ недоступными для выбора). Для команд ОТКРЫТЬ и СОХРАНИТЬ заданы клавишные сокращения второго типа. При определении текста элемента меню используется знак табуляции – Chr\$(9) для отделения клавишного сокращения.

```
Include "MENU.DEF"
Create Menu "Файл" As
"Новый"      Calling M_FILE_NEW,
"Открыть" +Chr$(9)+"Ctrl+O/W^J" Calling M_FILE_OPEN,
"(-",
"(Заккрыть"  Calling M_FILE_CLOSE,
"(Сохранить" +Chr$(9)+"Ctrl+Ы /W^S" Calling M_FILE_SAVE,
"(-",
"(Печатать"   Calling M_FILE_PRINT,
"(-",
"Выход"      Calling M_FILE_EXIT
```

### Пример 3:

Если не хотите, чтобы пользователь мог открывать быстрые меню, используйте оператор **Create Menu** и переопределите эти меню одним только разделителем: “(-”, как показано в следующем примере.

```
Create Menu "MapperShortcut" As "(-"
```

### Смотрите также:

**Create Menu Bar, Alter Menu Item**

## Оператор Create Menu Bar

### Назначение:

Перестраивает строку заголовков меню, используя стандартные и ранее определенные меню.

### Синтаксис (вариант 1):

```
Create Menu Bar As
    { menu_name | ID menu_number }
    [ , { menu_name | ID menu_number } ... ]
```

### Синтаксис (вариант 2)

**Create Menu Bar As Default**

где

*menu\_name* – заголовок стандартного для MapInfo меню или заголовок специально определенного меню, которое было ранее создано при помощи оператора **Create Menu**;

*menu\_number* – номер стандартного меню (например, 1 для меню ФАЙЛ).

### Описание:

Оператор **Create Menu Bar** говорит MapInfo, какие меню должны быть помещены в строку меню и в каком порядке. Если оператор задает не полный список стандартных меню, то результатом будет строка меню с сокращенным списком. Если в список меню включено одно или несколько специально определенных имен меню (они создаются при помощи оператора **Create Menu**), то результатом будет строка меню с расширенным списком.

Меню может задаваться именем (например, "Файл"), как стандартное меню, так и специально определенное. Каждое стандартное меню также имеет номер (идентификатор), который может использоваться при задании. Например, меню ФАЙЛ имеет идентификатор 1.

Список стандартных имен меню MapInfo приводится в описании оператора **Alter Menu**.

После того как система меню была изменена, Вы можете вернуть ее к стандартному виду оператором:

```
Create Menu Bar As Default
```

При этом удаляются все изменения в строке меню, включая и те, которые были созданы другими приложениями. Поэтому следует быть внимательным, употребляя этот оператор.

При формировании своей строки меню существует строгая рекомендация: первым определять меню ФАЙЛ, а вторым меню ПРАВКА. Эти два меню являются стандартным элементом интерфейса Windows, отсутствие этих меню может дезориентировать работу пользователя. Также желательно самым правым меню ставить меню СПРАВКА.

### Пример 1:

Строка меню сокращается до четырех заголовков: "Файл", "Правка", "Анализ" и меню, соответствующее открытому окну ("Карта", "График" и т. п.).

```
Create Menu Bar As
    "Файл", "Правка", "Анализ", "WinSpecific"
```

### Пример 2:

В стандартной строке меню такие меню как КАРТА и СПИСОК не показываются, если окна Карты или Списка соответственно не являются активными. Следующий оператор помещает эти меню в строку

меню так, что они не зависят от того, есть ли на экране окна Карты и Списка. Но при этом, если пользователь откроет меню КАРТА, когда на экране нет активного окна Карты, то он увидит, что все команды этого меню недоступны (показаны серым шрифтом). Аналогично работает меню СПИСОК.

```
Create Menu Bar As  
  "Файл", "Правка", "Запрос", "Карта", "Список"
```

### Пример 3:

В следующем фрагменте оператор **Create Menu** создает пользовательское меню ДАННЫЕ, которое можно будет использовать в операторе **Create Menu Bar**.

```
Declare Sub AddSub  
  Declare Sub EditSub  
  Declare Sub DelSub  
Create Menu "Данные" As  
  "Добавить" Calling addsub,  
  "Правка" Calling editSub,  
  "Удалить" Calling delSub  
Create Menu Bar As  
  "Файл", "Правка", "Данные"
```

### Смотрите также:

**Alter Menu Bar, Create Menu, Menu Bar**

# Оператор Create MultiPoint

### Назначение

Объединяет множество точек в один объект. Все точки имеют один и тот же символ. Объект Multipoint отображается в окне Списка как одна запись.

### Синтаксис:

#### Create Multipoint

```
[ Into { Window window_id | Variable var_name } ]  
[ num_points ]  
( x1, y1 ) ( x2, y2 ) [ ... ]  
[ Symbol ... ]
```

*window\_id* - идентификатор окна

*var\_name* - имя переменной объекта

*x y* - координаты точки

Предложение **Symbol** определяет стиль объекта. Внимание: один символ используется для всех точек, содержащихся в объекте Multipoint.

*num\_points* - число точек в объекте Multipoint.

В настоящее время MapInfo Professional использует четыре варианта синтаксиса для определения символа, используемого для точек. Необходима поддержка всех этих типов для объектов Multipoint:

### Синтаксис 1 (Синтаксис MapInfo 3.0 Symbol)

#### Symbol ( *shape*, *color*, *size* )

*shape* - целое, имеющее значение 31 или более, определяющее, какой символ используется из стандартного набора символов MapInfo. Для создания невидимого символа используйте значение 31. Стандартный набор символов включает символы от 31 до 67, но пользователь может настроить свой собственный набор символов, используя приложение Symbol.

*color* - целое, значение цвета RGB; смотрите функцию RGB( ).

*size* - целое, размер символа в пунктах (точках), от 1 до 48.

### Синтаксис 2 (Синтаксис TrueType Font)

#### Symbol ( *shape*, *color*, *size*, *fontname*, *fontstyle*, *rotation* )

*shape* - целое, имеющее значение 31 или больше, определяющее, какой символ из шрифтов TrueType используется. Для создания невидимого символа используйте значение 31.

*color* - это целое, значение цвета RGB; смотрите функцию RGB( ).

*size* - целое, размер символа в пунктах (точках), от 1 до 48.

*fontname* - строка, имя шрифта TrueType (например, "Wingdings").

*fontstyle* - целое, код, контролирующий атрибуты, например, курсив.

*rotation* - вещественное, угол поворота символа, в градусах.

### Синтаксис 3 (Синтаксис Custom Bitmap File)

#### Symbol ( *filename*, *color*, *size*, *customstyle* )

*filename* - строка длиной до 31 символа, имя файла bitmap. Файл должен находиться в папке CUSTSYMB (если не применен оператор Reload Symbols для указания другой папки).

*color* - целое, цвет RGB; смотрите функцию RGB( ).

*size* - целое, размер символа в пунктах (точках), от 1 до 48.

*customstyle* - целочисленный код, контролирующий цвет и атрибуты фона. См. таблицу ниже.

### Синтаксис 4

**Symbol** *symbol\_expr*

*symbol\_expr* - это выражение для Symbol, которое может быть или именем переменной символа или функцией, которая возвращает значение символа, например, MakeSymbol.

### Пример

```
Create Multipoint 7 (0,0) (1,1) (2,2) (3,4) (-1,1) (3,-2) (4,3)
```

# Оператор Create Object

### Назначение:

Создает один или более регионов при использовании операторов Buffer, Merge, Intersect, Union или Voronoi.

### Синтаксис:

```
Create Object As { Buffer | Union | Intersect | Merge | ConvexHull }  
    From fromtable  
    [ Into { Table intotable | Variable varname } ]  
    [ Width bufferwidth [ Units unitname ] ]  
    [ Resolution smoothness ]  
    [ Data column = expression [ , column = expression ... ] ]  
    [ Group By column | RowID } ] ]
```

где

*fromtable* – имя открытой таблицы, содержащей один или более графических объектов;

*intotable* – имя открытой таблицы, в которую помещается новый объект (или объекты);

*varname* – имя объектной переменной;

*bufferwidth* – положительное число, радиус буферной зоны, в основном число положительное (если это число отрицательное и исходный объект замкнут, то результатом будет объект меньше исходного);

*unitname* – имя единицы измерения расстояния (например, "km" – километры);

*smoothness* – число сегментов для окружности, задающих гладкость границы буферной зоны, число от 2 до 100;

*column* – имя колонки в таблице.

### Описание:

```
Create Object As { Buffer | Union | Intersect | Merge | ConvexHull | Voronoi }  
    From fromtable  
    [ Into { Table intotable | Variable varname } ]  
    [ Width bufferwidth [ Units unitname ] ] [Type { Spherical | Cartesian } ] ]  
    [ Resolution smoothness ]  
    [ Data column = expression [ , column = expression ... ] ]  
    [ Group By { column | RowID } ] ]
```

*fromtable* имя открытой таблицы, содержащей один или более графических объектов

*intotable* имя открытой таблицы, где будут храниться новые объекты

*varname* имя переменной Object где будет храниться новый объект

*bufferwidth* число, определяющее смещение (ширину), используемое в операции Buffer; если число отрицательное, и если исходный объект замкнутый, то результирующий буфер будет меньше чем исходный объект. Если ширина отрицательна и объект линейный (линия, полилиния, дуга) или точка, то будет использоваться абсолютная величина ширины для создания положительного буфера.

*unitname* имя единиц измерения расстояния (например, "km" для километров)

*smoothness* целое, от 2 до 100, определяющее число сегментов в окружности для операции Buffer

*column* имя колонки в таблице

Оператор **Create Object** создает один или более новых объектов регионов, при применении

географических операций (**Buffer**, **Merge**, **Intersect**, **Union**, **ConvexHull** или **Voronoi**) к одному или большему числу существующих объектов.

Предложение **Into** определяет, где будут храниться результаты. Для сохранения результатов в таблице, укажите **Into Table**. Для сохранения результатов в переменной Object, укажите **Into Variable**. Если пропущено предложение **Into**, результаты сохранятся в исходной таблице. Внимание: если используется предложение **Group By** чтобы объединить данные, то надо сохранить результаты в таблице раньше чем в переменной.

Ключевое слово, которое следует за ключевым словом **As** определяет, какой тип объектов будет создан. Укажите **Buffer** для генерации буферных регионов; подробности смотрите в тексте ниже. Укажите **Intersect** для создания объекта, представляющего пересечение других объектов (например, если два региона пересекаются, то пересечение - это область, покрывающая сразу эти два объекта).

Укажите **Merge** для создания объекта, представляющего комбинированную площадь из объектов-источников. Операция **Merge** создает результирующий объект, который содержит все полигоны, относящиеся к исходным объектам. Если исходные объекты перекрываются, то операция слияния (merge) не удаляет перекрытия. Таким образом, если объединяются два перекрывающихся региона (каждый из которых содержит один полигон), то окончательный результат будет в виде одного региона, состоящего из двух перекрывающихся полигонов. Вообще-то в этом случае можно вместо этого оператора использовать **Union**.

Укажите **Union** для осуществления операции комбинирования, которая удаляет любые области перекрытия. Если осуществляется операция объединения (union) двух перекрывающихся регионов (каждый из которых содержит один полигон), то результат будет в виде объекта-региона, содержащего один полигон.

Операции объединения (union) и слияния (merge) похожи, но их поведение сильно различается в том случае, если объекты полностью содержат в себе другие объекты. В этом случае, операция слияния (merge) удаляет область малого объекта из большого объекта, оставляя дыру там, где был малый объект. Операция объединения не будет удалять область малого объекта.

Оператор **Create Objects As Union** похож на оператор Objects Combine. Objects Combine будет удалять исходные объекты и вставлять новый комбинированный объект. Create Objects As Union будет только вставлять новый комбинированный объект, не стирая исходные объекты.

Комбинирование с использованием оператора Target и различных таблиц возможно только при применении Objects Combine. Оператор Combine Objects использующий функциональность Column допустим только при использовании Create Objects As Union вместе с предложением Group By.

Если оператор **Create Object As Union** не включает в себя предложение Group By, MapInfo Professional создает один комбинированный объект для всех объектов в таблице. Если этот оператор включает в себя предложение Group By, то должно быть задано имя колонки в таблице, что позволит MI Pro сгруппировать исходные объекты в соответствии с содержанием колонки и создать комбинированный объект для каждой группы объектов.

Если Вы определяете предложение **Group By**, MI Pro сгруппирует все записи имеющие одинаковые значения и осуществит операцию (слияния) в группу.

Если Вы определяете предложение **Data**, MI Pro осуществит объединение данных. Например, если осуществляется слияние или объединение, может возникнуть желание использовать предложение **Data**, чтобы присвоить значение данным основываясь на функциях объединения **Sum()** или **Avg()**.

## Оператор Create Object

---

Используйте **Type** для определения метода, по которому подсчитывается ширина буферной зоны вокруг объекта. Значение может быть **Spherical** или **Cartesian**. Обратите внимание, что если **Coordsys** в таблице это **NonEarth**, то вычисления будут осуществлены на плоскости, независимо от того, какие настройки сделаны, и если **Coordsys** в таблице это Долгота/Широта, то вычисления будут осуществлены на сфере, независимо от того, какие настройки сделаны.

### Операция **Convex Hull** для оператора **Create Object**

**Create Object As { Buffer | Union | Intersect | Merge | ConvexHull }**

Оператор **Create Object** создает один или более новых объектов-регионов при выполнении географических операций (**Buffer**, **Merge**, **Intersect**, **Union** или **ConvexHull**) на одном или большем числе существующих объектов.

Оператор **ConvexHull** будет создавать полигон, представляющий огибающий объект вокруг набора точек. Огибающий полигон может быть проинтерпретирован как оператор, натягивающий резинку вокруг всех крайних точек. Он будет состоять из минимального количества точек, поскольку большая часть точек обычно оказывается внутри множества. Полигон всегда получается выпуклым.

Точки, используемые для огибающего региона могут быть любыми узлами их **Regions**, **Polylines** или **Points** в таблице **From**. Если оператор **Create Object As ConvexHull** не включает в себя предложение **Group By**, **MI Pro** создаст один огибающий полигон. Если оператор включает в себя предложение **Group By** указывающее имя колонки в таблице, то **MI Pro** группирует исходные объекты исходя из содержания указанной колонки, затем создаст огибающий полигон для каждой группы объектов. Если оператор включает в себя предложение **Group By RowID**, то **MI Pro** создаст один огибающий полигон для каждого объекта в исходной таблице.

### Создание буферных зон

Если оператор **Create Object** осуществляет операцию **Buffer**, то он может включать в себя предложения **Width** и **Resolution**. Предложение **Width** определяет ширину буферной зоны. Добавочное подпредложение **Units** позволяет указать имя единиц измерения расстояния (например, "km" для километров) относящихся к предложению **Width**. Если предложение **Width** не включает в себя подпредложение **Units**, то ширина буферной зоны будет интерпретироваться в **MapBasic** в текущих единицах измерения. По умолчанию, **MapBasic** использует мили для измерения расстояний; чтобы изменить стандартные единицы измерения, смотрите описание оператора **Set Distance Units**.

Дополнительное подпредложение **Type** позволяет указать тип алгоритма вычисления расстояний при создании буферной зоны. Если используется тип алгоритма **Spherical**, то вычисления будут на сфере, а данные должны быть в Долготе/Широте. Если используется тип алгоритма **Cartesian** (декартовский), то измерения расстояний производятся на плоскости. Если предложение **Width** не включает в себя подпредложение **Type**, то по умолчанию используется алгоритм типа **Spherical** (подсчет расстояний на сфере). Если данные в проекции Долгота/Широта, то используется подсчет расстояний на сфере, независимо от подпредложения **Type**. Если данные в картографических проекциях, то используются декартовые вычисления на плоскости, независимо от установок подпредложения **Type**.

Параметр *smoothness* позволяет определить число сегментов из которых состоит каждая окружность на концах буферной зоны. Стандартное значение *smoothness* равно 12. Если задать величину *smooth-*



ness большую чем 12, то получится более сглаженная буферная зона. Обратите внимание, что чем больше значение *smoothness*, тем дольше работает оператор **Create Object**, и больше дискового пространства занимает результат.

Если оператор **Create Object As Buffer** не включает в себя предложение **Group By**, то MI Pro создаст один буферный регион. Если оператор включает в себя предложение **Group By**, которое является именем колонки в таблице, MI Pro сгруппирует исходные объекты в соответствии с содержанием колонки, затем создаст по одному буферному региону для каждой группы объектов. Если оператор включает в себя предложение **Group By RowID**, то MI Pro создаст по одному буферному региону для каждого объекта в исходной таблице.

### Полигоны Вороного, предложение Voronoi

Задайте предложение **Voronoi** для создания регионов, представляющих полигоны Вороного для исходных точек. Значения данных их исходных точек могут быть присвоены результирующему полигону, для этого в предложении надо указать источник таких точек.

### Пример

Следующий пример показывает слияние объектов-регионов из таблицы *Parcels* и хранит результирующие регионы в таблице *Zones*. Если оператор **Create Object** включает в себя предложение **Group By**, то MapBasic сгруппирует регионы таблицы *Parcel*, а затем произведет слияние для каждой группы. Таким образом, таблица *Zones* будет иметь по одному объекту-региону для каждой группы объектов в таблице *Parcels*. Каждая группа будет состоять из тех частей, которые имеют одинаковые значения в колонке *zone\_id*.

Следующая за оператором **Create Object** колонка, подсчитывающая число однородных объектов в таблице *Zones* будет показывать сколько элементов сливаются вместе, образуя одну зону. Колонка *zonevalue* в таблице *Zones* будет показывать сумму элементов, которые включает в себя эта зона.

```
Open Table "PARCELS"
Open Table "ZONES"
Create Object As Merge
  From PARCELS Into Table ZONES Data
  parcelcount=Count(*),zonevalue=Sum(parcelvalue)
  Group By zone_id
```

Следующий пример создает объект-регион, представляющий зону шириной в четверть мили вокруг выбранных объектов. Объект-буфер хранится в переменной *Object corridor*. Последующие операторы **Update** или **Insert** могут затем копировать объект в таблицу.

```
Dim corridor As Object
Create Object As Buffer
  From Selection
  Into Variable corridor
  Width 0.25 Units "mi"
  Resolution 60
```

Следующий объект показывает оконтуривающий полигон, состоящий из многих объектов, созданный оператором **Create Object As**.

```
create object as convex hull from state_caps into table dump_table
```

### Смотрите также

**Buffer( ), Objects Combine, Objects Erase, Objects Intersect, ConvexHull( )**

# Оператор Create Pline

### Назначение:

Создает объект типа "полилиния".

### Синтаксис:

#### Create Pline

```
[ Into { Window window_id | Variable var_name } ]  
num_points  
(x1, y1) (x2, y2) [ ... ]  
[ Pen... ]  
[ Smooth ]
```

где

*window\_id* – идентификатор окна, целое число;

*var\_name* – имя объектной переменной;

*num\_points* – определяет число узлов для полилинии;

*x*, *y* – координаты для одного узла.

Слово **Pen** начинает стандартное предложение для назначения стиля линии объекта.

### Описание:

Результатом действия оператора **Create Pline** является новый объект типа "полилиния".

Вы можете сразу задать в операторе **Create Pline** все узлы полилинии. Вы можете также использовать двухшаговую тактику: сначала постройте объект без узлов оператором **Create Pline**, и далее, используя оператор **Alter Object**, задайте все необходимые атрибуты полилинии.

Если оператор **Create Pline** использует предложение **Into Variable**, то созданный объект "полилиния" присваивается объектной переменной. Если после слов **Into Window** указывается окно, объект помещается на подготовленное место в окне (например, на изменяемый слой). Если предложения **Into** вообще нет в операторе, MapBasic попытается создать полилинию в активном окне; если это невозможно (например, поверх всех окон лежит окно Графика), то объект не будет создан.

Параметры *x* и *y* являются координатами в той системе координат, которая была объявлена до оператора **Create Pline**. (Смотрите описание оператора **Set CoordSys**.) Если система не объявлялась, то координаты будут принимать значения широты и долготы. При этом надо учесть, что MapBasic игнорирует координатную систему самого окна Карты.

Если объект создается для окна Отчета, параметры *x* и *y* интерпретируются как координаты на листе в объявленных ранее единицах измерения листа. (Смотрите описание оператора **Set Paper Units**.) Так X-координата – это расстояние от левого края листа до точки, а Y-координата – расстояние от верхнего края листа.

**Замечание:** Перед созданием объекта в окне Отчета не забудьте выполнить оператор **Set CoordSys Layout**.

Если в операторе не участвует предложение **Pen**, то оператор **Create Arc** использует установку соответствующего режима для стиля линии в MapInfo (стиль линии можно изменить командой НАСТРОЙКА > СТИЛЬ ЛИНИЙ).

Одна полилиния может состоять как из одной ломаной, так и из нескольких фрагментов.

Однофрагментный объект типа "полилиния" может содержать до 32 763 узлов. Для многофрагментных объектов лимит узлов меньше: на каждую ломаную линию надо убавлять по три узла.

### Смотрите также:

**Alter Object, Insert, Pen, Update**

## Функция **CreatePoint( )**

### Назначение:

Возвращает объект типа "точка".

### Синтаксис:

**CreatePoint(*x, y*)**

где

*x* – X-координата точки (или широта), вещественное число;

*y* – Y-координата точки (или долгота), вещественное число.

### Величина, полученная в результате:

Точка. Величина типа Object.

### Описание:

Функция **CreatePoint( )** возвращает объект типа "точка".

Параметры *x* и *y* задают координаты центра окружности в той координатной системе, которая была объявлена MapBasic ранее. Смотрите описание оператора **Set CoordSys**. Если система не объявлялась, то координаты будут принимать значения широты и долготы. При этом надо учесть, что MapBasic игнорирует координатную систему самого окна Карты.

Точечный объект будет создаваться в соответствии с установкой стиля символа для точечных объектов в операторе **Set Style**, который надо выполнить до функции **CreatePoint( )**. Вы можете также воспользоваться для создания объекта оператором **Create Point**, в котором используется предложение **Symbol** для определения стиля символа точки.

Графический объект, созданный функцией **CreateLine( )**, может быть присвоен объектной переменной, которая определяет значение уже существующей строки в таблице (оператор **Update**) или вновь созданной (оператор **Insert**).

**Замечание:** Перед созданием объекта в окне Отчета не забудьте выполнить оператор **Set CoordSys Layout**.

### Пример 1:

В примере используется оператор **Insert** для создания новой строки в таблице SITES. Функция **CreatePoint( )** используется в теле оператора **Insert** для создания объекта типа "точка", данные которого будут помещены в этой строке.

```
Open Table "sites.tab"
Insert Into sites (obj)
Values ( CreatePoint(-72.5, 42.4) )
```

### Пример 2:

В следующем фрагменте используются колонки "Xcoord" и "Ycoord" таблицы SITES, как содержащие координаты для точечных объектов, которые с помощью оператора **Update** и функции **CreatePoint( )** будут присоединены к таблице SITES: каждый к соответствующей строке.

```
Open Table "sites"
Update sites
Set obj = CreatePoint(xcoord, ycoord)
```

## Функция CreatePoint( )

---

### Пример 3:

В приведенном выше примере подразумевается, что колонки "Xcoord" и "Ycoord" содержат значения широты и долготы. Файлы точек MapInfo для DOS содержат координаты в миллионных долях градуса, а не в целых градусах. Кроме того, в большинстве файлов точек MapInfo для DOS принято направление отсчета долгот с Востока на Запад. Поэтому, чтобы оператор **Update** мог корректно преобразовать файл точек из DOS в Windows, нужно разделить обе координаты на миллион и умножить "Xcoord" на минус единицу:

```
Update sites  
Set obj = CreatePoint(-xcoord/1000000,ycoord/1000000)
```

### Смотрите также:

**Create Point, Insert, Update**

## Оператор Create Point

### Назначение:

Создает объект типа "точка".

### Синтаксис:

#### Create Point

```
[ Into { Window window_id | Variable var_name } ]  
(x, y)  
[ Symbol... ]
```

где

*window\_id* – идентификатор окна, целое число;

*var\_name* – имя объектной переменной;

*x*, *y* – координаты точки;

Слово **Symbol** начинает стандартное предложение для назначения стиля символа точечного объекта.

### Описание:

Оператор **Create Point** создает новый точечный объект.

Если оператор включает предложение **Into Variable**, то созданный объект объявляется как значение объектной переменной. Если после слов **Into Window** указывается окно, объект помещается на подготовленное место в окне (например, на изменяемый слой). Если **Into** вообще нет в операторе, MapBasic попытается создать точку в самом верхнем окне; если это невозможно (например, поверх всех окон лежит окно Графика), то объект не будет создан.

Параметры *x* и *y* являются координатами в той системе координат, которая была объявлена до оператора **Create Point**. (Смотрите описание оператора **Set CoordSys**.) Если система не объявлялась, то координаты будут принимать значения широты и долготы. При этом надо учесть, что MapBasic игнорирует координатную систему самого окна Карты.

Если объект создается для окна Отчета, параметры *x* и *y* интерпретируются как координаты на листе в объявленных ранее единицах измерения листа. (Смотрите описание оператора **Set Paper Units**.)

Так X-координата – это расстояние от левого края листа до точки, а Y-координата – расстояние от верхнего края листа. Перед созданием объекта в окне Отчета необходимо выполнить оператор **Set CoordSys Layout**.

Предложением **Symbol** назначается стиль символа, которым точечный объект обозначается на Карте. Если в операторе это предложение не участвует, оператор **Create Point** использует установку соответствующего режима для стиля линии в MapInfo (стиль линии можно изменить командой НАСТРОЙКА > СТИЛЬ СИМВОЛОВ).

### Смотрите также:

**CreatePoint( ), Insert, Symbol, Update**

# Оператор Create PrismMap

## Назначение

Создает Карту-призму.

## Синтаксис

### Create PrismMap

```
[ From Window window_ID |  
  MapString mapper_creation_string ]  
{ layer_id | layer_name }  
With expr  
[ Camera [ Pitch angle | Roll angle | Yaw angle | Elevation angle ] |  
  [ Position (x,y,z) | FocalPoint (x,y,z) ] |  
  [ Orientation (vu_1, vu_2, vu_3, vpn_1, vpn_2, vpn_3,  
    clip_near, clip_far) ] ]  
[ Light Color lightcolor ] ]  
[ Scale grid_scale ]  
[ Background backgroundcolor ]
```

*window\_id* - это идентификатор окна для окна Карты, которое содержит слой полигонов.

Если слой с полигонами не найден, появится сообщение об ошибке.

*mapper\_creation\_string* определяет командную строку, которая создает текстуру для Карты-призмы.

*layer\_id* - это идентификатор слоя Карты

*layer\_name* - это имя слоя Карты.

*Camera* определяет позицию и ориентацию камеры.

*angle* - это угол в градусах. Горизонтальный угол (измеряется в диапазоне 0-360 градусов) определяет вращение карты вокруг центральной точки поверхности (grid). Вертикальный угол изменяется в диапазоне 0-90 и измеряет наклон карты от начальной точки.

*Pitch* настраивает вращение камеры вокруг оси X, при этом точкой вращения считается центральная (начальная) точка камеры.

*Roll* настраивает вращение камеры вокруг оси Z, при этом точкой вращения считается центральная (начальная) точка камеры.

*Yaw* настраивает вращение камеры вокруг оси Y, при этом точкой вращения считается центральная (начальная) точка камеры.

*Elevation* настраивает вращение карты вокруг оси X, при этом точкой вращения считается фокальная точка камеры.

*Position* определяет позицию камеры и/или источника освещения

*FocalPoint* определяет фокальную точку камеры и/или источника освещения.

*Orientation* определяет для камеры значение параметров ViewUp, ViewPlane и Clipping Range (используется для инерции зрительного восприятия).

*grid\_scale* - это масштаб по оси Z. Если значение > 1, то изображение растянется по оси Z, а если значение < 1, то изображение сожмется по оси Z.

*backgroundcolor* - это цвет, используемый для фона и определяемый функцией RGB.

### Описание

Оператор **Create PrismMap** создает окно Карты-призмы. С помощью карты-призмы можно отображать несколько переменных для одного объекта. Например, цвет, ассоциированный с полигоном, может иметь тематическое значение одной колонки, а высота призмы может отражать значение другой колонки. Оператор **Create PrismMap** соответствует команде КАРТА > СОЗДАТЬ КАРТУ-ПРИЗМУ.

Между сеансами работы MapInfo сохраняет настройки карты-призмы, сохраняя оператор **Create PrismMap** в файле рабочего набора. Таким образом, чтобы увидеть пример оператора **Create PrismMap**, создайте карту, выполните команду КАРТА > СОЗДАТЬ ТЕМАТИЧЕСКУЮ КАРТУ, сохраните рабочий набор (например, PRISM.WOR), и проверьте рабочий набор в окне MapBasic. После этого скопируйте оператор **Create PrismMap** в Вашу программу MapBasic. Аналогично можно увидеть примеры операторов **Create PrismMap** при открытии окна MapBasic перед выполнением команды КАРТА > СОЗДАТЬ ТЕМАТИЧЕСКУЮ КАРТУ.

Дополнительное предложение *window\_id* определяет, какая карта послужит основой Карты-призмы; если предложение *window\_id* отсутствует, MapBasic создаст призмы для самого верхнего окна Карты. Оператор **Create PrismMap** указывает, какой слой надо использовать, даже если окно Карты имеет только один слой. Слой может быть идентифицирован по номеру (*layer\_id*), где самый верхний слой карты имеет *layer\_id*, равный 1, следующий слой имеет *layer\_id*, равный 2, и т.д. В то же время оператор **Create PrismMap** может идентифицировать слой карты по имени (например, "world").

Каждый оператор **Create PrismMap** должен определять предложение выражения *expr*. MapInfo оценивает это выражение для каждого объекта в таблице, из которой строятся призмы; обрабатывая действия оператора **Create PrismMap**, MapInfo выбирает стиль отображения каждого объекта, основываясь на значении записи *expr*. Выражение обычно включает в себя имя одной или более колонок из таблицы.

### Пример

```
Open Table "STATES.TAB" Interactive
Map From STATES
Create PrismMap From Window FrontWindow() STATES With Pop_1980
Background RGB(192,192,192)
```

### Смотрите также

Функция PrismMapInfo()

Оператор Set PrismMap

# Оператор Create Ranges

### Назначение:

Вычисляет значения диапазонов для условного (тематического) выделения методом выделения диапазонов и помещает объекты в массив переменных, который можно впоследствии использовать в операторе **Shade**.

### Синтаксис:

#### Create Ranges

```
From table  
With expr  
[ Use { "Equal Ranges" | "Equal Count" | "Natural Break" | "StdDev" } ]  
[ Quantile Using q_expr ]  
[ Number num_ranges ]  
[ Round rounding_factor ]  
Into Variable array_variable
```

где

*table* – имя открытой таблицы, объекты которой будут участвовать в тематическом выделении;

*expr* – выражение, которое вычисляется для каждой записи таблицы;

*q\_expr* – выражение, используемое при квантовании;

*num\_ranges* – задает число диапазонов (по умолчанию 4);

*rounding\_factor* – делитель, по которому округляются диапазоны при разделении (например, значение 10 задает округление до ближайшего десятка);

*array\_variable* – массив численных переменных типа Float, в который будут помещены значения для диапазонов.

### Описание:

Оператор **Create Ranges** вычисляет значения диапазонов, которые могут быть впоследствии использоваться в операторе **Shade**, для создания тематического слоя на Карте методом выделения диапазонов. Подробную информацию о тематических Картах смотрите в документации по MapInfo.

Предложение **Use** задает метод, каким будут разделены данные на диапазоны. Задав **"Equal Ranges"**, Вы разделите диапазоны, исходя из разброса значений (например, 0-25, 25-50, 50-75, 75-100). Задав **"Equal Count"**, Вы разделите диапазоны, исходя из количества записей в таблице, т.е. в диапазоны попадет примерно одинаковое количество записей. Задав **"Natural Break"**, Вы разделите диапазоны, исходя из естественно близких групп значений. Задав **"StdDev"**, Вы сначала делите диапазоны по среднему значению, а затем добавляете один диапазон со значениями выше среднего, но не далее величины дисперсии от среднего, а также еще один диапазон со значениями ниже среднего, но не далее величины дисперсии от среднего.

Предложение **Into Variable** определяет имя массива переменных типа Float, в который будет помещен результат. Не нужно следить за размерами этого массива; MapInfo автоматически изменяет его размер. Размер этого массива будет всегда в два раза больше количества диапазонов, так как в нем помещаются как нижняя, так и верхняя граница диапазона.

После выполнения оператора **Create Ranges** выполняется оператор **Shade**, чтобы создать тематическую карту. В операторе **Shade** можно задать предложение **From Variable**, которое прочитает массив границ диапазонов. Оператор **Shade** обычно использует ту же таблицу и то же



выражение для колонки, что и оператор **Create Ranges**.

### Квантование диапазонов

Предложение **Quantile Using** отключает предложение **Use** и назначает деление диапазонов квантованием, которое задается выражением **Quantile Using**.

Квантование лучше всего проиллюстрировать следующим примером. Оператор вычисляет границы диапазонов покупательной способности населения США (BPI), квантуя их по значениям населения штатов.

```
Create Ranges      From states
  With BPI_1990    Quantile Using Pop_1990
    Number 5
  Into Variable f_ranges
```

В этом примере создается пять диапазонов (Number 5).

Штаты с высокой покупательной способностью населения (**With BPI\_1990**) помещаются в "высшие" диапазоны (темные оттенки цвета), а с низкой – в "низшие" (светлые оттенки цвета).

Границы диапазонов устанавливаются следующим образом: так как предложение **Quantile Using** задает колонку "Pop\_1990", то MapInfo сначала вычисляет общее количество населения США (около 250 миллионов); затем MapInfo делит результат на количество диапазонов (в нашем случае, 5) и получается пятьдесят миллионов. После этого MapInfo делит диапазоны так, чтобы суммарное население штатов, охватываемых диапазоном, приближалось (но не превосходило) пятьдесят миллионов.

MapInfo собирает штаты по порядку возрастания покупательной способности BPI, и в первый, "низший" диапазон попадают штаты с наименьшей покупательной способностью. MapInfo продолжает добавлять штаты в первый диапазон до тех пор, пока суммарное значение населения не достигнет или не приблизится к пятидесяти миллионам; в этот момент MapInfo решает, что первый диапазон готов и приступает к обсчету следующего и т.д.

### Пример:

```
Include "mapbasic.def"

Dim range_limits() As Float, brush_styles() As Brush
Dim col_name As Alias

Open Table "states" Interactive

Create Styles
  From Brush(2, CYAN, 0) 'стиль для "низшего" диапазона
  To Brush (2, BLUE, 0) 'стиль для "высшего" диапазона
  Vary Color By "RGB"
  Number 5
  Into Variable brush_styles

' Присвоим имя колонки переменной типа Alias:
col_name = "Pop_1990"
```

## Оператор Create Ranges

---

```
Create Ranges      From states
  With col_name
  Use "Natural Break"
  Number 5
  Into Variable range_limits

Map From states

Shade states
  With col_name
  Ranges
    From Variable range_limits
    Style Variable brush_styles

' Вывод окна Легенды:
Open Window Legend
```

**Смотрите также:**

**Create Styles, Set Shade, Shade**

## Оператор Create Rect

### Назначение:

Создает объект "прямоугольник".

### Синтаксис:

#### Create Rect

```
[ Into { Window window_id | Variable var_name } ]  
(x1, y1) (x2, y2)  
[ Pen... ]  
[ Brush... ]
```

где

*window\_id* – идентификатор окна, целое число;

*var\_name* – имя объектной переменной;

*x1*, *y1* – координаты начального угла прямоугольника;

*x2*, *y2* – координаты противоположного по диагонали угла прямоугольника.

Слово **Pen** начинает стандартное предложение для назначения стиля линии объекта.

Слово **Brush** начинает стандартное предложение для назначения стиля штриховки объекта.

### Описание:

Оператор **Create Rect** создает новый объект типа "прямоугольник". Если ширина равняется высоте, то объект является квадратом, иначе – прямоугольником.

Предложением **Into Variable** созданный объект объявляется как значение объектной переменной *var\_name*. Если после слов **Into Window** указывается окно, объект помещается на подготовленное место в окне (например, в изменяемый слой). Если предложения **Into** вообще нет в операторе, MapBasic попытается создать прямоугольник в самом верхнем окне; если это невозможно (например, поверх всех окон лежит окно Графика), то объект не будет создан.

Параметры *x* и *y* являются координатами в той системе координат, которая была объявлена до оператора **Create Rect**. (Смотрите описание оператора **Set CoordSys**.) Если система не объявлялась, то координаты будут принимать значения широты и долготы. При этом надо учесть, что MapBasic игнорирует координатную систему самого окна Карты. Если объект создается для окна Отчета, параметры *x* и *y* интерпретируются как координаты на листе в объявленных ранее единицах измерения листа. (Смотрите описание оператора **Set Paper Units**.) Так, X-координата – это расстояние от левого края листа до точки, а Y-координата – расстояние от верхнего края листа.

**Замечание:** Перед созданием объекта в окне Отчета не забудьте выполнить оператор **Set CoordSys Layout**.

Предложения **Pen** и **Brush** назначают стиль линии и штриховки объекта. Если в операторе не участвует предложение **Pen**, оператор **Create Rect** использует установку соответствующего режима для стиля линии в MapInfo. Стиль линии можно изменить командой НАСТРОЙКА > СТИЛЬ ЛИНИЙ. Предложению **Brush** в MapInfo соответствует команда НАСТРОЙКА > СТИЛЬ ОБЛАСТЕЙ.

### Смотрите также:

**Brush, Create RoundRect, Insert, Pen, Update**

# Оператор Create Redistricter

### Назначение:

Открывает окно Районирование.

### Синтаксис:

```
Create Redistricter source_table By district_column
    With
    [ Count ]
    [, Brush ] [, Symbol ] [, Pen ]
    [, { Sum | Percent } (expr) ]
    [, { Sum | Percent } (expr) ... ]
    [ Order { "MRU" | "Alpha" | "Unordered" } ]
```

где

*source\_table* – имя открытой таблицы, объекты которой будут участвовать в районировании;

*district\_column* – имя колонки; начальный набор районов создается на базе значений из этой колонки и в нее же помещается новая структура районов;

*expr* – числовое выражение.

Слово **Count** предопределяет показ в Списке Районов колонки с количеством объектов в каждой группе.

Слово **Brush** назначает показ колонки с образцами штриховки объектов.

Слово **Symbol** назначает показ колонки с образцами символов.

Слово **Pen** назначает показ колонки с образцами линий объектов.

Предложение **Order** задает порядок строк в Списке **Районов** (по алфавиту, произвольно или же все затронутые изменения строки помещаются в начало списка; последний режим (**MRU**) используется по умолчанию).

### Описание:

Оператор **Create Redistricter** начинает сеанс районирования. Этот оператор соответствует команде MapInfo ОКНО > РАЙОНИРОВАНИЕ. Информацию о районировании Вы можете прочитать в документации MapInfo.

Управлять составом районов можно оператором **Set Redistricter**. Закончить сеанс районирования можно, выполнив оператор **Close Window**, закрывающий окно **Районирование**.

Если включить слово **Brush**, то в окно **Районирование** будет добавлена колонка с образцами штриховок каждого района. Обратите внимание на то, что **Brush** является ключевым словом, а не предложением. Так же **Symbol** и **Pen** являются отдельными ключевыми словами. Если в Списке Районов есть колонки с образцами оформления районов, то пользователь может их изменять, указывая на них мышкой.

### Смотрите также:

**Set Redistricter**

## Оператор Create Region

### Назначение:

Создает объект типа "область".

### Синтаксис:

#### Create Region

```
[ Into { Window window_id | Variable var_name } ]
    num_polygons
    [ num_points1 (x1, y1) (x2, y2) [ ... ] ]
    [ num_points2 (x1, y1) (x2, y2) [ ... ] ... ]
    [ Pen... ]
    [ Brush... ]
    [ Center (center_x, center_y) ]
```

где

*window\_id* – идентификатор окна, целое число;

*var\_name* – имя объектной переменной;

*num\_polygons* – число полигонов в области (ноль или более);

*num\_points1* – число узлов в первом полигоне;

*num\_points2* – число узлов во втором полигоне, и т. д.;

*x*, *y* – координаты узла полигона;

*center\_x*, *center\_y* – координаты центра области.

Слово **Pen** начинает стандартное предложение для назначения стиля линии объекта.

Слово **Brush** начинает стандартное предложение для назначения стиля штриховки объекта.

### Описание:

Результатом действия оператора **Create Region** является новый объект типа "область".

Если параметр *num\_polygons*, который определяет количество многоугольников, включенных в область, приравнять нулю, то оператор создаст пустую область. Впоследствии Вы можете, используя оператор **Alter Object**, добавлять все необходимые детали в этот объект.

Создание области можно разделить на два этапа: сначала при помощи оператора **Create Region** создать объект, не имеющий полигонов, а затем по ходу выполнения приложения оператор **Alter Object** добавит в объект необходимые элементы. Такая последовательность может оказаться полезной, когда на момент создания объекта нет полной информации о количестве и расположении многоугольников и узлов в будущей области. Детали смотрите в описании оператора **Alter Object**.

Предложение **Into Variable** объявляет созданный объект как значение объектной переменной *var\_name*. Если слово **Into** указывает окно, объект помещается на подготовленное место в окне (например, в изменяемый слой). Если предложения **Into** вообще нет в операторе, MapBasic попытается создать область в самом верхнем окне. Если это невозможно (например, поверх всех окон лежит окно Графика), то объект не будет создан.

Параметры *x* и *y* являются координатами в той системе координат, которая была объявлена до оператора создания. (Смотрите описание оператора **Set CoordSys**.) Если система не объявлялась, то координаты будут принимать значения широты и долготы. При этом надо учесть, что MapBasic игнорирует координатную систему самого окна Карты. Если объект создается для окна Отчет, параметры *x* и *y* интерпретируются как координаты на листе в объявленных ранее единицах

## Оператор Create Region

---

измерения листа. (Смотрите описание оператора **Set Paper Units**.) Так X-координата – это расстояние от левого края листа до точки, а Y-координата – расстояние от верхнего края листа. Перед созданием объекта в окне Отчета не забудьте выполнить оператор **Set CoordSys Layout**.

Предложения **Pen** и **Brush** назначают стиль линии и штриховки объекта. Если в операторе не участвует предложение **Pen**, оператор **Create Region** использует установку соответствующего режима для стиля линии в MapInfo. Стиль линии можно изменить командой НАСТРОЙКА > СТИЛЬ ЛИНИЙ. Аналогично предложению **Brush**, в MapInfo соответствует команда НАСТРОЙКА > СТИЛЬ ОБЛАСТЕЙ.

Одна область может состоять как из одного полигона (многоугольника), так и из нескольких полигонов. Объект типа "область", состоящая из одного полигона, может содержать до 32 763 узлов. Для многофрагментных объектов лимит узлов меньше: на каждый полигон надо убавлять по три узла.

### Пример:

```
Dim obj_region As Object
Dim x(100), y(100) As Float
Dim i, node_count As Integer

' В массивах x() and y()
' мы имеем координаты узлов области

' Сначала создадим пустой объект :
Create Region Into Variable obj_region 0

' Теперь внесем информацию об узлах:
For i = 1 to node_count

    Alter Object obj_region Node Add ( x(i), y(i) )

Next

' Теперь поместим область в таблицу SITES:
Insert Into Sites (Object) Values (obj_region)
```

### Смотрите также:

**Alter Object, Brush, Insert, Pen, Update**

### Оператор Create Report From Table

#### Назначение:

Создает файл отчета для Crystal Reports из открытой таблицы MapInfo:

#### Синтаксис:

**Create Report From Table** *tablename* [**Into** *reportfilespec*][**Interactive**]

*tablename* - это имя открытой таблицы в MapInfo

*reportfilespec* - это полный путь и имя файла для нового файла отчета.

Ключевое слово **Interactive** означает, что новый отчет будет немедленно загружен в модуль Crystal Report Designer. Режим **Interactive** употребляется, если предложение **Into** пропущено.

Обратите внимание на то, что нельзя создать отчет для растровой таблицы или грид-файла, сразу появится сообщение об ошибке.

#### Смотри также:

**Open Report**

# Оператор Create RoundRect

### Назначение:

Создает объект типа "сглаженный прямоугольник".

### Синтаксис:

#### Create RoundRect

```
[ Into { Window window_id | Variable var_name } ]  
(x1, y1) (x2, y2)  
rounding  
[ Pen... ]  
[ Brush... ]
```

где

*window\_id* – идентификатор окна, целое число;

*var\_name* – имя объектной переменной;

*x1*, *y1* – координаты начального угла прямоугольника;

*x2*, *y2* – координаты противоположного по диагонали угла прямоугольника;

*rounding* – диаметр окружности (величина типа Float), которую можно вписать в дугу закругления угла, в единицах системы координат.

Слово **Pen** начинает стандартное предложение для назначения стиля линии объекта.

Слово **Brush** начинает стандартное предложение для назначения штриховки объекта.

### Описание:

Результатом действия оператора **Create RoundRect** является новый объект типа "сглаженный прямоугольник" (прямоугольник со скругленными углами).

Предложение **Into Variable** объявляет созданный объект как значение объектной переменной *var\_name*. Если слово **Into** указывает окно, объект помещается на подготовленное место в окне (например, в изменяемый слой). Если предложения **Into** вообще нет в операторе, MapBasic попытается создать прямоугольник в самом верхнем окне; если это невозможно (например, поверх всех окон лежит окно Графика), то объект не будет создан.

Если объект создается для окна Отчета, параметры *x* и *y* интерпретируются как координаты на листе в объявленных ранее единицах измерения листа. (Смотрите описание оператора **Set Paper Units**.) Так, X-координата – это расстояние от левого края листа до точки, а Y-координата – расстояние от верхнего края листа. Перед созданием объекта в окне Отчета не забудьте выполнить оператор **Set CoordSys Layout**.

Предложения **Pen** и **Brush** назначают стиль линии штриховки объекта. Если в операторе не участвует предложение **Pen**, оператор **Create RoundRect** использует установку соответствующего режима для стиля линии в MapInfo. Стиль линии можно изменить командой НАСТРОЙКА > СТИЛЬ ЛИНИЙ. Аналогично стиль штриховки объекта в MapInfo можно изменить командой НАСТРОЙКА > СТИЛЬ ОБЛАСТЕЙ.

### Смотрите также:

**Brush, Create Rect, Insert, Pen, Update**



## Оператор Create Styles

### Назначение:

Создает массив из значений стилей линии, штриха или символа.

### Синтаксис:

#### Create Styles

```

From{ Pen... | Brush... | Symbol... }
To { Pen... | Brush... | Symbol... }
Vary
  { Color By { "RGB" | "HSV" } |
    Background By { "RGB" | "HSV" } |
    Size By { "Log" | "Sqrt" | "Constant" }
  }
[ Number num_styles ]
[ Inflect At range_number With { Pen... | Brush... | Symbol... } ]
Into Variable array_variable

```

где

*num\_styles* – число создаваемых значений стиля, по умолчанию 4;

*range\_number* – число типа SmallInt, задающее номер диапазона последнего перед переломом;

*array\_variable* – массив переменных типа Pen, Brush или Symbol.

### Описание:

Оператор **Create Styles** создает множество значений стиля линии, штриха или символа и присваивает его массиву переменных соответствующего типа. Массив этих значений в дальнейшем может быть использован оператором **Shade** при создании условных (тематических) Карт. О создании тематических Карт смотрите документацию MapInfo.

Предложения **From** и **To** задают первое и последнее значения стиля: **Pen**, **Brush** или **Symbol**. Если массив позже будет использован в условном выделении, то стилем из предложения **From** будут выделены объекты самого "нижнего" диапазона, а объекты самого "верхнего" диапазона будут выделены стилем из предложения **To**.

Оператор **Create Styles** строит ряд, множество переходных значений от значения, заданного в предложении **From**, до значения стиля, заданного в предложении **To**. Например, после слова **From** задан стиль **Brush** для морских глубин как насыщенный синий, и после слова **To** задан стиль **Brush** для отмелей как светло-голубой. В этом случае MapInfo построит последовательность значений стиля штриха (тип Brush), где первое будет иметь насыщенный синий цвет, последнее значение стиля – светло-голубой, а промежуточные значения – переходные оттенки.

Предложение **Number** задает количество значений стиля, включая заданные предложениями **From** и **To** первое и последнее значения. Это число должно соответствовать числу диапазонов условного выделения в следующем операторе **Shade**.

Предложение **Vary** задает порядок, согласно которому атрибуты значений стиля будут меняться в создаваемом ряду. Для задания изменения цвета штриховок и линий переднего плана используется подпредложение **Color**. Для задания изменения цвета фона используется подпредложение **Background**. В обоих случаях задается перебор значений по шкале **RGB** или по шкале **HSV**. Если Вы создаете ряд для стиля символа, Вы можете использовать подпредложение **Size** для интерполяции по размеру символа. Для стиля линии (тип Pen) подпредложение **Size** задает изменение толщины

## Оператор Create Styles

---

линии.

Если Вы используете предложение **Inflect At**, то Вы задаете переломную (или пороговую) точку в вычислении ряда значений стилей между значениями, заданными предложением **From**, и значениями, заданными предложением **To**. При этом MapInfo создает две последовательности значений: одну от **From** до значения перелома **Inflect**, другую от **Inflect** до **To**. Например, переломное значение используется при построении условной карты прибыли и убытков. Области, где предприятия получают прибыль, будут окрашены разными оттенками зеленого, а области, где предприятия терпят убытки, будут окрашены разными оттенками красного. Переломное значение можно использовать только при раскраске разными цветами.

Предложение **Into Variable** задает имя массива переменных (без скобок). Вам нет необходимости заботиться о размерности массива, MapBasic автоматически подгонит размерность массива по числу значений, если это необходимо. Тип массива переменных должен соответствовать типу стиля, значения которого задаются предложениями **From** и **To**.

### Пример:

```
Dim b_ranges(_) As Brush

Create Styles
  From Brush(2, CYAN, 0) ' стиль для нижнего диапазона
  To Brush (2, BLUE, 0) ' стиль для верхнего диапазона
  Vary Color By "RGB"
  Number 5
  Into Variable b_ranges
```

Этот оператор **Create Styles** создает ряд из пяти значений стиля штриха. Значения помещаются в ячейки массива "b\_ranges". Следующий оператор **Shade** может создать условную карту, в которой будут использованы стили из массива "b\_ranges". Смотрите также описание оператора **Create Ranges**.

### Смотрите также:

**Create Ranges, Set Shade, Shade**

## Оператор Create Table

### Назначение:

Создание новой таблицы.

### Синтаксис:

```
Create Table table
( column columntype [ , ... ] ) | Using from_table }
[ File filespec ]
[ { Type NATIVE |
    Type DBF [ CharSet char_set ] |
    Type {Access|ODBC} database_filespec [ Version version ] Table tablename
    [ Password pwd ] [ CharSet char_set ]
} ]
[ Version version ]
```

*table* имя для новой таблицы MapInfo.

*column* имя колонки в новой таблице. Имя колонки может быть длиной до 31 символов, и может содержать буквы, числа, и знак подчеркивания. Название колонки не может начинаться с цифры.

*from\_table* имя текущей открытой таблицы, в которой находится та колонка, которую надо сохранить в новой таблице. Таблица *from\_table* должна быть базовой и должна содержать колонку с данными. Таблицы запросов и растровые таблицы не могут использоваться для этих целей, их применение даст сообщение об ошибке. Структура колонки в новой таблице будет идентичной той, которая имеется в исходной таблице.

*filespec* определяет место, где создавать файлы .TAB, .MAP, и .ID (и в случае Access, .AID файлов). Если Вы пропускаете предложение **File**, то файлы создаются в текущей директории.

*char\_set* строковая величина, определяющая кодировку символов в таблице (смотрите описание стандартного предложения **CharSet**).

*database\_filespec* строка, определяющая доступную базу данных Access. Если такая база не существует, MapInfo создаст новый Access-файл .MDB.

*version* это выражение указывающее какая версия базы данных Microsoft Jet будет использоваться при создании новой таблицы. Допустимые значения 4.0 (для Access 2000) или 3.0 (для Access '95/'97). Если значение не указано, то по умолчанию принимается версия 4.0. Если база данных в которой создается новая таблица уже существует, то указанная версия базы данных игнорируется.

*tablename* строковая переменная, определяющая имя таблицы, которая появится в Access.

*pwd* пароль на уровне базы данных, определяемый при включении защиты базы данных.

*version* равно 100 (при создании таблицы, которая может читаться ранними версиями MapInfo) или 300 (формат MapInfo). Этот параметр не применяется при создании таблицы Access; версия для таблицы Access управляется DAO.

*columntype* тип данных, связанных с колонкой. Каждый параметр *columntype* задает тип данных колонки и имеет следующий синтаксис:

```
Char( width ) |
Float |
Integer |
SmallInt |
```

## Оператор Create Table

---

**Decimal( *width* , *decplaces* ) |**  
**Date |**  
**Logical**

*width* определяет максимальный размер каждого поля (применяется не ко всем типам полей).

Символьные поля могут содержать до 254 символов.

*decplaces* определяет число знаков после десятичной точки для поля десятичного типа.

### Описание

Оператор **Create Table** создает новую пустую таблицу размером до 250 колонок. Определите **ODBC** для создания новых таблиц на сервере DBMS.

Предложение **Using** позволяет создавать новую таблицу как часть функциональности "Объединение объектов по колонке". Таблица *from\_table* должна быть базовой, и должна содержать колонку с данными. Таблицы запросов и растровые таблицы не могут здесь использоваться и появится сообщение об ошибке. Структура (колонок) создаваемой новой таблицы будет идентична структуре данной таблице.

Добавочное предложение *filespec* определяет место, где создается новая таблица. Если предложение *filespec* не используется, таблица создается в текущей директории или папке.

Предложение **Type** определяет формат данных таблицы. По умолчанию это формат MapInfo, но может быть также и DBF. Формат MapInfo (NATIVE) занимает меньше места, чем формат DBF, но формат DBF читается в любых dBASE-совместимых системах управления базами данных. Таким образом создаются новые таблицы на серверах DBMS с помощью предложения ODBC Type оператора Create Table.

Предложение **CharSet** определяет установку шрифта. Параметр *char\_set* должен быть строковой постоянной, такой как "MacRoman" или "WindowsLatin1". Если предложение **CharSet** не определено, MapBasic использует по умолчанию шрифт, который установлен в этот момент Windows. См также обсуждение предложения **CharSet** для получения большей информации.

Величина типа SmallInt для колонок резервирует два байта для каждого значения; так, колонка может содержать значения от -32,767 до +32,767. Величина типа целое (Integer) для колонок резервирует четыре байта для каждого значения; так, колонка может содержать значения от -2,147,483,647 до +2,147,483,647.

Предложение **Version** управляет форматом таблицы. Если Вы зададите **Version 100**, MapInfo создаст таблицу в формате, читаемом ранними версиями MapInfo, ранее 3.0. Если зададите **Version 300**, MapInfo создаст таблицу в формате, используемом MapInfo 3.0. Обратите внимание, что объекты типа полилиния и регион, имеющие более 8,000 узлов и полилинии, состоящие из множества сегментов требуют версию 300. Если Вы пропустите предложение **Version**, то таблица сохранится в формате версии 300.

### Пример

Следующий пример показывает, как создать таблицу, названную *Towns*, содержащую 3 поля: символьное поле, названное *townname*, целочисленное поле, названное *population*, и десятичное поле, названное *median\_income*. Файл будет создан в поддиректории C:\MAPINFO\DATA. Поскольку необязательное предложение Type используется, таблица будет создана в формате dBASE.

```
Create Table Towns
( townname Char(30),
  population SmallInt,
```

```
median_income Decimal(9,2) )  
File "C:\MAPINFO\TEMP\TOWNS"  
Type DBF
```

### Смотрите также

Import, Export, Open, Drop, Alter Table, Create Index, Create Map

Оператор Server Create Map

### Оператор Create Text

#### Назначение:

Создает объект типа "текст".

#### Синтаксис:

##### Create Text

```
[ Into { Window window_id | Variable var_name } ]  
  text_string  
  (x1, y1) (x2, y2)  
  [ Font . . . ]  
  [ Label Line { Simple | Arrow } (label_x, label_y) ]  
  [ Spacing { 1.0 | 1.5 | 2.0 } ]  
  [ Justify { Left | Center | Right } ]  
  [ Angle text_angle ]
```

где

*window\_id* – целое число, идентификатор окна Карты или Отчета;

*var\_name* – имя существующей объектной переменной;

*text\_string* – текст длиной до 255 символов (многострочный текст содержит символ **Chr\$(10)**);

*x1*, *y1* – координаты одного угла прямоугольника, заполненного текстом;

*x2*, *y2* – координаты противоположного по диагонали угла прямоугольника;

*label\_x*, *label\_y* – координаты места, к которому прикреплен текст;

*text\_angle* – угол поворота текста в градусах, действительная величина.

Слово **Font** начинает стандартное предложение для назначения стиля шрифта текстового объекта.

#### Описание:

Параметры *x* и *y* являются координатами в той системе координат, которая была объявлена до оператора **Create Text**. (Смотрите описание оператора **Set CoordSys**.) Если система не объявлялась, то координаты будут принимать значения широты и долготы. При этом надо учесть, что MapBasic игнорирует координатную систему самого окна Карты. Если объект создается для окна Отчета, параметры *x* и *y* интерпретируются как координаты на листе в объявленных ранее единицах измерения листа. (Смотрите описание оператора **Set Paper Units**.) Так X-координата – это расстояние от левого края листа до точки, а Y-координата – расстояние от верхнего края листа. Перед созданием объекта в окне Отчета не забудьте выполнить оператор **Set CoordSys Layout**.

Параметры *x1*, *y1*, *x2* и *y2* задают прямоугольник текста. Если текстовый объект создается в окне Карты, текст будет заполнять заданную текстовую область так, чтобы длина строки была равна ширине текстовой области. Размер шрифта, заданный предложением **Font**, будет проигнорирован. В окне Отчета текст будет рисоваться заданного в предложении **Font** размера, при этом координаты верхнего левого угла текстовой области (*x1*, *y1*) будут задавать расположение объекта на листе, а вторая пара (*x2*, *y2*) будет проигнорирована.

Предложение **Font** назначает шрифт для текстового объекта. Если предложения нет в операторе, оператор **Create Text** использует текущую установку шрифта в MapInfo (шрифт можно изменить в диалоге команды НАСТРОЙКА > СТИЛЬ ТЕКСТА).

#### Смотрите также:

**Autolabel, Font, Insert, Update**

## Функция CreateText( )

### Назначение:

Возвращает текстовый объект, созданный в определенном окне Карты.

### Синтаксис:

**CreateText(*window\_id* , *x* , *y* , *text* , *angle* , *anchor* , *offset*)**

*window\_id* – целочисленный идентификатор окна Карты;

*x* , *y* – координаты, задающие закрепленное положение подписи;

*text* – строка с текстом подписи, представляющим текстовый объект;

*angle* – угол поворота подписи в градусах; для для горизонтального текста он равен нулю;

*anchor* – целое число типа Integer от 0 до 8, контролирующее расположение подписи относительно места привязки. Ниже перечислены возможные значения кодов, которые описаны в MAPBA-SIC.DEF.

LAYER\_INFO\_LBL\_POS\_CC (0)  
 LAYER\_INFO\_LBL\_POS\_TL (1)  
 LAYER\_INFO\_LBL\_POS\_TC (2)  
 LAYER\_INFO\_LBL\_POS\_TR (3)  
 LAYER\_INFO\_LBL\_POS\_CL (4)  
 LAYER\_INFO\_LBL\_POS\_CR (5)  
 LAYER\_INFO\_LBL\_POS\_BL (6)  
 LAYER\_INFO\_LBL\_POS\_BC (7)  
 LAYER\_INFO\_LBL\_POS\_BR (8)

Двухбуквенное окончание определяет ориентацию подписи: T=Top, B=Bottom, C=Center, R=Right, L=Left. Например, для размещения подписи ниже и правее места привязки, укажите код LAYER\_INFO\_LBL\_POS\_BR, или укажите значение 8.

*offset* – целое число от 0 до 50, расстояние в точках от подписи до точки привязки на подписываемом объекте; *offset* игнорируется, если значение кода привязки равно 0.

### Величина, полученная в результате

Величина типа Object.

### Описание:

Функция **CreateText( )** возвращает величину типа Object, являющейся текстовым объектом.

Текстовый объект использует текущий стиль текста. Для создания текстового объекта с определенным стилем, используйте оператор **Set Style** перед **CreateText( )**.

В тот момент, когда текст создан, его высота контролируется размером текущего шрифта. Таким образом, после создания текстового объекта его высота зависит от размера окна Карты; изменение масштаба ведет к соответственному изменению размера текста.

Возвращаемый объект имеет тип Object и сохраняется в существующей строке таблицы (используя оператор **Update**) или вставляется в новую строку таблицы (используя оператор **Insert**).

## Функция CreateText( )

---

### Пример:

Следующий пример создает текстовый объект и помещаеь его на Косметический слой Карты (целочисленная переменная i\_map\_id содержит идентификатор окна Карты).

```
Insert Into Cosmetic1 (Obj)  
Values ( CreateText(i_map_id, -80, 42.4, "Sales Map", 0,0,0) )
```

### Смотрите также:

**Autolabel, Create Text, Font, Insert, Update**



## Функция **CurDate( )**

### Назначение:

Возвращает текущее значение даты в формате ГГГГММДД.

### Синтаксис:

**CurDate( )**

### Величина, полученная в результате:

Дата.

### Описание:

Функция **CurDate( )** возвращает текущее значение даты.

Формат всегда будет вида – ГГГГММДД. Для того чтобы возвращаемое значение было в формате локальной системы, используйте функцию **FormatDate\$( )**.

### Пример:

```
Dim d_today As Date  
d_today = CurDate( )
```

### Смотрите также:

**FormatDate\$( ), Day( ), Month( ), StringToDate( ), Timer( ), Weekday( ), Year( )**

### Функция **CurrentBorderPen( )**

#### Назначение

Возвращает текущее значения стиля линии границы региона.

#### Синтаксис

**CurrentBorderPen( )**

#### Возвращаемое значение

Pen

#### Описание

Функция **CurrentBorderPen( )** возвращает текущее значения стиля линии границы региона. MapInfo Professional применяет текущий стиль линии к любому площадному объекту, нарисованному пользователем. Если программа MapBasic создает объект используя оператор типа Create Region, но в нем не задан стиль линии границы предложением Pen, то в объекте будет использован текущий стиль BorderPen.

Возвращаемое значение может быть присвоено переменной Pen, или может быть использовано в качестве параметра внутри оператора, который использует установки Pen в качестве параметра (в таком как Set Map).

Чтобы извлечь специфические атрибуты стиля Pen (такие как color), надо вызвать функцию **StyleAttr( )**.

Более подробная информация об установках Pen находится в описании предложения Pen.

#### Пример

```
Dim p_user_pen As Pen    p_user_pen = CurrentBorderPen( )
```

#### Смотрите также

Pen, CurrentLinePen(), Set Style, StyleAttr( )

## Функция **CurrentBrush( )**

### Назначение:

Возвращает значение установленного на данный момент стиля штриховки.

### Синтаксис:

**CurrentBrush( )**

### Величина, полученная в результате:

Величина типа Brush.

### Описание:

Функция **CurrentBrush( )** возвращает значение текущей установки стиля штриха. В MapInfo это значение изменяется в диалоге команды НАСТРОЙКИ > СТИЛЬ ОБЛАСТЕЙ. Когда Вы рисуете в окне такие объекты, как эллипс, прямоугольник, сглаженный прямоугольник или регион, MapInfo заполняет его установленным штрихом. Когда программа на MapBasic создает такие объекты (например, оператором **Create Region( )** без предложения **Brush**, объект заштриховывается в соответствии с текущей установкой стиля штриха в MapInfo.

Величина, полученная функцией **CurrentBrush( )**, может быть присвоена переменной типа Brush или использована как параметр в операторах, в которых используется установка стиля штриха (такие, как **Set Map** или **Shade**).

Для вывода отдельных характеристик стиля штриха (например, цвета) используется функция **StyleAttr( )**.

Более подробно о стиле можно прочитать в описании предложения **Brush**.

### Пример:

```
Dim b_current_fill As Brush
b_current_fill = CurrentBrush( )
```

### Смотрите также:

**Brush**, **MakeBrush( )**, **StyleAttr( )**

### Функция CurrentFont( )

#### Назначение:

Возвращает значение шрифта, используемого в данный момент в окне Карты.

#### Синтаксис:

**CurrentFont( )**

#### Величина, полученная в результате:

Величина типа Font.

#### Описание:

Функция возвращает значение текущей установки стиля шрифта. В MapInfo это значение изменяется в диалоге команды НАСТРОЙКИ > СТИЛЬ ТЕКСТА. Когда Вы создаете в окне текстовый объект, MapInfo рисует буквы установленным шрифтом. Когда программа MapBasic создает текстовый объект при помощи оператора **Create Text** без слова **Font**, объекту назначается шрифт в соответствии с текущей установкой шрифта в MapInfo.

Величина, полученная функцией **CurrentFont( )**, может быть присвоена переменной типа Font или использована как параметр в операторах, в которых используется установка шрифта.

Для вывода отдельных характеристик стиля шрифта (например, цвета) используется функция **StyleAttr( )**.

Более подробно о стиле можно прочитать в описании предложения **Font**.

#### Пример:

```
Dim f_user_text As Font
f_user_text = CurrentFont( )
```

#### Смотрите также:

**Font, MakeFont( ), Set Style, StyleAttr( )**

## Функция CurrentLinePen( )

### Назначение

Возвращает текущий стиль линии Pen.

### Синтаксис

**CurrentLinePen( )**

### Возвращаемое значение

Pen

### Описание

Функция **CurrentLinePen( )** возвращает текущий стиль линии Pen. MapInfo Professional присваивает текущий стиль любой линии или полилинии, нарисованной пользователем. Если программа MapBasic создает объект с помощью оператора, такого как Create Line, но при этом оператор не использует предложение Pen, то объект будет использовать текущий стиль Pen. Возвращаемая величина может быть присвоена переменной Pen, или может быть использована в качестве параметра внутри оператора, который использует установки Pen в качестве параметра (например, Set Map).

Чтобы извлечь специфические атрибуты стиля Pen (такие как color), надо вызвать функцию **StyleAttr( )**.

Более подробная информация об установках Pen находится в описании предложения Pen.

### Пример

```
Dim p_user_pen As Pen    p_user_pen = CurrentPen( )
```

### Смотрите также

Pen, CurrentBorderPen(), Set Style, StyleAttr( )

### Функция CurrentPen( )

#### Назначение:

Возвращает значение установленного на данный момент стиля линии и устанавливает стиль границы региона, который будет таким же как и стиль линии.

#### Синтаксис:

**CurrentPen( )**

#### Величина, полученная в результате:

Величина типа Pen.

#### Описание:

Функция **CurrentPen( )** возвращает значение текущей установки стиля линии. MapInfo Professional присваивает текущий стиль линии любой линии или полилинии, нарисованной пользователем. Если программа MapBasic создает объект используя оператор типа **Create Line**, на при этом сам оператор не включает в себя предложение **Pen**, то объект будет использовать текущий стиль Pen. Если Вам надо использовать текущий стиль линии не перустанавливая стиль границы региона, то используйте функцию **CurrentLinePen()**. Возвращаемое значение может быть присвоено переменной Pen, или может быть использовано как параметр внутри оператора, который использует настройку Pen (например, **Set Map**). Для извлечения отдельных атрибутов стиля Pen (таких как color), вызовите функцию **StyleAttr( )**.

Более подробно настройки Pen описаны в предложении **Pen**.

#### Пример

```
Dim p_user_pen As Pen
p_user_pen = CurrentPen()
```

#### Смотрите также

**MakePen( ), Pen, Set Style, StyleAttr( )**

## Функция **CurrentSymbol( )**

### Назначение:

Возвращает значение установленного на данный момент стиля символа для точечного объекта.

### Синтаксис:

**CurrentSymbol( )**

### Величина, полученная в результате:

Величина типа **Symbol**.

### Описание:

Функция **CurrentSymbol( )** возвращает значение текущей установки стиля символа для точечного объекта. В MapInfo это значение может изменяться в диалоге команды НАСТРОЙКИ > СТИЛЬ СИМВОЛОВ. Когда Вы рисуете в окне точечный объект, MapInfo обозначает его установленным символом. Когда программа MapBasic создает такой объект при помощи оператора **Create Point** без слова **Symbol**, объект изображается в соответствии с текущей установкой стиля символа в MapInfo. Величина, полученная функцией **CurrentSymbol( )**, может быть присвоена переменной типа **Symbol** или использована как параметр в операторах, в которых используется установка стиля символа (таких, как **Set Map** или **Shade**).

Для вывода отдельных характеристик стиля символа (например, цвета) используется функция **StyleAttr( )**.

Более подробно о стиле можно прочитать в описании предложения **Symbol**.

### Пример:

```
Dim sym_user_symbol As Symbol
sym_user_symbol = CurrentSymbol( )
```

Смотрите также: **MakeSymbol( )**, **Set Style**, **StyleAttr( )**, **Symbol**

### Функция DateWindow( )

#### Назначение:

Возвращает текущую дату, установленную как целое в диапазоне от 0 до 99, или (-1) если режим управления окном даты отключен.

#### Синтаксис:

**DateWindow(context)**

context - это короткое целое, которое может быть или DATE\_WIN\_CURPROG или DATE\_WIN\_SESSION.

#### Описание:

Это зависит от того, какой контекст (context) передается. Если context это DATE\_WIN\_SESSION, то возвращаются текущие настройки компьютера. Если context это DATE\_WIN\_CURPROG, то возвращаются локальные настройки MapBasic, если context пропущен, то возвращаются настройки текущего сеанса.

MBX скомпилированные ранее версии v5.5 будут до сих пор конвертировать двузначные года в текущее столетие (5.0 и более ранние версии). Для получения результатов, адаптированных к следующему тысячелетию, перекомпилируйте приложение с помощью MapBasic v5.5.

#### Пример:

В следующем примере переменные Date1 = 19890120, Date2 = 20101203 и MyYear = 1990.

```
DIM Date1, Date2 as Date
DIM MyYear As Integer
Set Format Date "US"
Set Date Window 75
    Date1 = StringToDate("1/20/89")
    Date2 = StringToDate("12/3/10")
MyYear = Year("12/30/90")
```

#### Смотрите также:

**Оператор SetDateWindow**



## Функция Day( )

### Назначение:

Возвращает номер дня (от 1 до 31) из даты.

### Синтаксис:

**Day**(*date\_expr*)

где *date\_expr* – выражение, в результате которого получается дата.

### Величина, полученная в результате:

Короткое целое число. Величина типа SmallInt.

### Описание:

Функция возвращает целочисленное значение от одного до тридцати одного, являющееся номером дня в месяце. Например, для даты 12/17/92 функция будет равна 17.

### Пример:

```
Dim day_var As SmallInt, date_var As Date
date_var = StringToDate("05/23/1985")
day_var = Day(date_var)
```

### Смотрите также:

**CurDate( ), Month( ), Timer( ), Year( )**

## Оператор DDEExecute

### Назначение:

Посылает на выполнение команду для другой программы по каналу DDE-связи (динамического обмена данными).

### Предупреждение:

Использование этого оператора возможно только в среде Microsoft Windows.

### Синтаксис:

**DDEExecute** *channel*, *command*

где

*channel* – номер открытого канала DDE-связи, целое число;

*command* – команда, посылаемая для выполнения в другую программу (DDE-сервер), строковая величина.

### Описание:

Посылает на выполнение команду в другую программу, присоединенную к каналу DDE-связи.

Номер канала должен быть задан функцией **DDEInitiate**( ), которая открывает канал динамического обмена данными. Оператор **DDEExecute** возможен только после выполнения функции **DDEInitiate**( ). Параметр *command* должен зависеть от той программы, которой будет послана команда. Обе программы должны поддерживать DDE-связь. Правильный формат команды зависит от правил, принятых в подсоединенной программе. Вы должны изучить документацию другой программы, прежде чем налаживать DDE-связь.

### Ошибки:

Оператор может генерировать следующие коды ошибок:

ERR-CMD-NOT-SUPPORTED, если программа выполняется не в среде Windows;

ERR-NO-RESPONSE-FROM-APP, если программа-сервер не отвечает.

### Пример:

Программными средствами MapBasic Вы можете открыть канал DDE-связи с Microsoft Excel и обращаться к нему как к программе-серверу. Если канал был открыт для объекта (topic) "System", Вы можете выполнить как команду, так и макрофункцию Excel. В следующем примере открывается рабочая таблица "TRIAL.XLS" в программе Excel:

```
Dim i_chan As Integer
i_chan = DDEInitiate("Excel", "System")
DDEExecute i_chan, "[OPEN(""C:\DATA\TRIAL.XLS"")] "
```

### Смотрите также:

**DDEInitiate**, **DDEPoke**, **DDERequest**( )

## Функция DDEInitiate( )

### Назначение:

Открывает новый канал DDE-связи и возвращает его номер.

### Предупреждение:

Использование этой функции возможно только в среде Microsoft Windows.

### Синтаксис:

**DDEInitiate**(*appl\_name*, *topic\_name*)

где

*appl\_name* – имя подключаемой программы (например, "MapInfo"), строковая величина;

*topic\_name* – имя документа или некоторого объекта (topic) программы (например, "System"), строковая величина.

### Величина, полученная в результате:

Целое число. Величина типа Integer.

### Описание:

Функция иницирует один канал DDE-связи (динамического обмена данными), назначая ему номер. Этот канал связи позволяет соединить две программы, работающие в среде Microsoft Windows, для пересылки информации. Как только канал открыт, MapBasic может читать информацию из документа другой программы (функция **DDERequest\$( )**) или записывать в этот документ (оператор **DDEPoke**). После информационного обмена каналы связи рекомендуется закрыть, используя операторы **DDETerminate** или **DDETerminateAll**.

DDE-связь возможна только в среде Microsoft Windows. И, если Ваше приложение будет обращаться к услугам DDE в среде другой вычислительной платформы, то MapBasic выдаст ошибку. Для обхода такого рода конфликта, Вы можете определить, в какой среде выполняется Ваша программа с помощью функции **SystemInfo( )**.

Параметр *appl\_name* задает имя программы, которое понятно для DDE-связи (например, для Microsoft Excel имя "Excel"). Программа, с которой связывается MapBasic, должна быть уже загружена перед выполнением функции **DDEInitiate( )**. Из программы на MapBasic это можно сделать при помощи оператора **Run Program**.

**Замечание:** Не все программы Windows поддерживают DDE-связь. Для получения информации о поддержке DDE смотрите документацию этих программ.

Параметр *topic\_name* задает некоторый объект для программы и зависит от нее. Каждая программа обладает некоторым набором объектов. О списке объектов, сопровождающих определенную программу, Вы можете прочитать в соответствующей документации. Для многих программ именем объекта является имя файла документа. Например, пусть "ORDERS.XLS" – имя файла рабочей таблицы программы Microsoft Excel и канал связи с ним открывается так:

```
Dim i_chan As Integer  
i_chan = DDEInitiate("Excel", "C:\ORDERS.XLS")
```

Многие программы поддерживают специальный объект "System". Начав DDE-обмен, используя объект "System", Вы можете с помощью функции **DDERequest\$( )** получить список строк, представляющих корректные имена объектов, поддерживаемые данной программой, например, список открытых файлов. Получив список открытых файлов, можно начать новый сеанс DDE-связи с любым документом.

## Функция DDEInitiate( )

---

В следующей таблице приводится список некоторых программ и их объектов для использования в функции **DDEInitiate( )**:

### Вызов функции

DDEInitiate("Excel", "System")

DDEInitiate("Excel", *wks*)

DDEInitiate("MapInfo", "System")

DDEInitiate("MapInfo", *mbx*)

### Что происходит при DDE-обмене

Функция **DDERequest\$( )** может извлечь системную информацию Excel, такую, как список рабочих таблиц, загруженных сейчас в среде Excel, и оператор **DDEExecute** может послать команду на выполнение в Excel.

Если *wks* – это имя документа Excel (например, "Sheet1" или "May.xls"), то следующий оператор **DDEPoke** может поместить какую-нибудь величину в таблицу Excel и функция **DDERequest\$( )** может прочитать информацию из нее.

Следующая функция **DDERequest\$( )** может извлечь такую информацию, как список выполняемых сейчас в MapInfo программ MapBasic.

Если *mbx* – это имя программы на MapBasic, которая сейчас выполняется (например, "C:\GRIDS.MBX"), то следующий оператор **DDEPoke** может назначить какую-нибудь величину глобальной переменной определенной программы MapBasic и функция **DDERequest\$( )** может прочитать текущее значение такой переменной.

Когда программа MapBasic выполняет функцию **DDEInitiate( )**, то MapBasic выступает как "клиент" в DDE-связи. Другие программы Windows выступают как программы-"серверы". В пределах одной связи клиент всегда активен, а сервер только отвечает на запросы клиента. MapBasic может поддерживать одновременно сразу столько каналов, сколько могут позволить ресурсы памяти и системы Вашего компьютера.

Прикладная программа одновременно может быть как клиентом в сеансе одной связи (выполняя такие операторы как **DDEInitiate( )** и др.), так и сервером для другого сеанса связи (выполняя процедуру со стандартным именем **RemoteMsgHandler**).

### Ошибки:

Функция может вернуть следующие коды ошибок:

ERR-CMD-NOT-SUPPORTED, если программа выполняется не в среде Windows;

ERR-INVALID-CHANNEL, если неправильно задан номер канала.

### Пример:

Следующий фрагмент текста программы иллюстрирует DDE-связь Microsoft Excel. Целью является передача простого сообщения ("Привет от MapInfo!") в первую ячейку первой рабочей таблицы, но только если она пуста. Если ячейка не пуста, ничего записываться не будет, содержимое ячейки будет показано пользователю MapInfo.

```
Dim chan_num, tab_marker As Integer
Dim topiclist, topicname, cell As String
```

```

chan_num = DDEInitiate("EXCEL", "System")
If chan_num = 0 Then
    Note "Простите, но Excel не отзывается."
    End Program
End If

' Вызов списка документов - рабочих таблиц сейчас
' загруженных в среде Excel
topiclist = DDERequest$(chan_num, "topics")
' Здесь должен быть получен список topiclist
' с элементами в виде:
'      ":   Sheet1      System"
' (если раб. таблица еще не имеет имени)
' или в таком виде:
'      ":   C:Orders.XLS  Sheet1  System"
' Если мы имеем дело с Excel версии 5,
' то topiclist может выглядеть так:
'      "[Book1]Sheet1  [Book2]Sheet2 ..."
' Теперь выделим из списка первое имя (может быть "Sheet1")
' для этого извлечем из строки текст между первым и
' вторым символом табуляции;
' или для Excel 5 версии текст до первой табуляции.
If Left$(topiclist, 1) = ":" Then
    ' ...then it's Excel 4.
    tab_marker = InStr(3, topiclist, Chr$(9) )
    If tab_marker = 0 Then
        Note "В программе Excel нет открытых документов! Остановка."
        End Program
    End If
    topicname = Mid$(topiclist, 3, tab_marker - 3)
Else
    ' ... для Excel 5.x
    tab_marker = Instr(1, topiclist, Chr$(9) )
    topicname = Left$( topiclist, tab_marker - 1)
End If

' Открываем канал для связи с открытым документом
' (например, "Sheet1")
DDETerminate chan_num
chan_num = DDEInitiate("Excel", topicname)
If chan_num = 0 Then
    Note "Problem communicating with " + topicname
    End Program
End If

' Теперь проверим первую ячейку таблицы Excel.
' Если она пуста, напомним туда сообщение.
' Если нет, то прочитаем содержимое и, используя
' оператор NOTE, покажем его в MapBasic.
' Заметим, что чистая ячейка возвращает знаки
' "возврат каретки" и "перевод строки": Chr$(13) + Chr$(10).
cell = DDERequest$( chan_num, "R1C1" )
If cell <> Chr$(13) + Chr$(10) Then
    Note

```

## Функция DDEInitiate( )

---

```
    "Сообщение не послано; В ячейке уже содержится:" + cell  
Else  
    DDEPoke chan_num, "R1C1", "Привет от MapInfo!"  
    Note "Сообщение послано в Excel,"+topicname+ ",R1C1."  
End If  
DDETerminateAll
```

Приведенный пример не гарантирует от ошибок. Например, в Excel может в момент вызова быть активно окно графика ("Chart1" или подобные ему); тогда обращение к ячейке R1C1 будет ошибочным. Кроме того, если Вы работаете с русской версией Excel, то обращаться нужно не к R1C1, а к C1K1 и т.д.

### Смотрите также:

**DDEExecute, DDEPoke, DDERequest\$, DDETerminate, DDETerminateAll**

## Оператор DDEPoke

### Назначение:

Посылает данные в программу-сервер по каналу DDE-связи.

### Предупреждение:

Использование этого оператора возможно только в среде Microsoft Windows.

### Синтаксис:

**DDEPoke** *channel, itemname, data*

где

*channel* – номер открытого канала DDE-связи, целое число;

*itemname* – имя элемента объекта, документа программы-сервера, строковая величина;

*data* – символьная строка, посылаемая в *itemname*.

### Описание:

Оператор **DDEPoke** посылает данные по каналу DDE-связи в объект программы-сервера.

Параметр *channel* – это целое число, полученное в результате выполнения функции **DDEInitiate( )** перед оператором **DDEPoke**.

Параметр *itemname* задает элемент, который соответствует определенному каналу *channel*. Каждая программа, поддерживающая DDE-связь, обладает некоторым набором объектов. О списке объектов, сопровождающих определенную программу, Вы можете прочитать в соответствующей документации.

Для DDE-связи с документом программы Excel именем элемента является строка, такая как "R1C1" (или "C1K1" в русской версии Excel) для ячейки в первой строке и первой колонке. Именем элемента может быть также имя глобальной переменной из другой программы, если с этой программой открыт канал связи.

### Ошибки:

Оператор может генерировать следующие коды ошибок:

ERR-CMD-NOT-SUPPORTED, если программа выполняется не в среде Windows;

ERR-INVALID-CHANNEL, если неправильно задан номер канала.

### Пример 1:

В этом примере посылается сообщение ("Привет от MapInfo!") в первую ячейку рабочей таблицы. Этот пример выполняется при условии, что Excel уже загружен и рабочая таблица (Sheet1) открыта и находится в режиме редактирования ячеек.

```
Dim chan_num As Integer
chan_num = DDEInitiate("EXCEL", "Sheet1")
DDEPoke chan_num, "R1C1", "Привет от MapInfo!"
```

### Пример 2:

В этом фрагменте глобальной переменной "Address" из программы DISPATCH.MBX посылается новое значение:

```
i_chan_num = DDEInitiate("MapInfo", "C:\DISPATCH.MBX")
DDEPoke i_chan_num, "Address", "23 Main St."
```

### Смотрите также:

**DDEExecute, DDEInitiate, DDERequest\$( )**

### Функция DDERequest\$( )

#### Назначение:

Возвращает данные, запрошенные через канал DDE-связи.

#### Предупреждение:

Использование этой функции возможно только в среде Microsoft Windows.

#### Синтаксис:

**DDERequest\$(channel, itemname)**

где

*channel* – номер открытого канала DDE-связи, целое число;

*itemname* – имя объекта для возвращаемой информации, строковая величина.

#### Величина, полученная в результате:

Строка. Величина типа String.

#### Описание:

Функция **DDERequest\$( )** возвращает информацию через канал DDE-связи. Если информация недоступна, функция возвращает пустое значение ("").

Параметр *channel* задает номер канала, открытого при помощи функции **DDEInitiate( )**.

Параметр *itemname* задает элемент, который соответствует каналу *channel*. Каждая программа, поддерживающая DDE-связь, обладает некоторым набором объектов. О списке объектов, сопровождающих определенную программу, Вы можете прочитать в соответствующей документации.

В следующей таблице две первые колонки содержат значения, которые были использованы функцией **DDEInitiate( )** при открытии канала связи с Excel, как сервером, а во второй колонке соответствующие значения для параметра *itemname* в функции **DDERequest\$( )**.

#### Имя объекта обмена

#### Значения *itemname* и результат функции

"System"

"Sysitems" для получения списка имен элементов, которые доступны по каналу обмена для "System";

"Topics" для получения списка объектов для DDE-связи с Excel, включая имена всех открытых рабочих таблиц;

"Formats" для получения списка форматов для системного буфера (Clipboard), поддерживаемых Excel (например, "TEXT", "BITMAP ...").

*wks* (имя таблицы Excel)

"RnCn" для получения строки с содержимым ячейки (здесь первый символ n должен быть номером строки, а второй – номером колонки).

**Замечание:** С помощью функции **DDERequest\$( )** можно считывать значения глобальных переменных одной программы в другую, выполняющуюся в то же время.

Следующая таблица приводит комбинации значений *itemname* и объектов обмена, с которыми открыта связь, для MapInfo как сервера.



Имя объекта обмена	Значения <i>itemname</i> и результат функции
"System"	<p>"Sysitems" для получения списка имен элементов, которые доступны по каналу обмена для "System";</p> <p>"Topics" для получения списка имен объектов для DDE-связи с MapInfo, включая имена выполняемых в данный момент прикладных программ;</p> <p>"Formats" для получения списка форматов для системного буфера (Clipboard ), поддерживаемых MapInfo (например, "TEXT");</p> <p>"Version" для получения числа, по которому можно определить версию MapInfo (например, "300" означает MapInfo 3.0)</p>
<i>mbx</i> (MBX-имя программы)	<p>"{items}" для получения списка имен глобальных переменных выполняющейся программы;</p> <p>имя глобальной переменной для получения ее текущего значения.</p>

### Ошибки:

В результате выполнения оператора могут генерироваться следующие коды ошибок:  
 ERR-CMD-NOT-SUPPORTED, если программа запущена не в системе Windows;  
 ERR-INVALID-CHANNEL, если неправильно задан номер канала;  
 ERR-CANT-INITIATE-LINK, если MapBasic не связан с этим объектом обмена.

### Пример 1:

В этом примере функция **DDERequest\$( )** читает содержимое первой ячейки рабочей таблицы. Пример работает, если Excel уже загружен.

```
Dim chan_num As Integer
Dim cell As String
chan_num = DDEInitiate("EXCEL", "Sheet1")
cell = DDERequest$(chan_num, "R1C1")
```

### Пример 2:

Следующий пример подразумевает, что уже действует другая MapBasic-программа под названием "Dispatch", в которой определена глобальная переменная "Address". С помощью функции **DDERequest\$( )** мы можем получить значение этой переменной.

```
Dim chan_num As Integer
Dim addr_copy As String
chan_num = DDEInitiate("MapInfo", "C:\DISPATH.MBX")
addr_copy = DDERequest$(chan_num, "Address")
```

### Оператор DDETerminate

#### Назначение:

Закрывает канал DDE-связи.

#### Предупреждение:

Использование этого оператора возможно только в среде Microsoft Windows.

#### Синтаксис:

**DDETerminate** *channel*

где

*channel* – номер открытого канала DDE-связи, целое число.

#### Описание:

Оператор **DDETerminate** закрывает канал динамического обмена данными, который был открыт функцией **DDEInitiate**( ).

Параметр *channel* является целым числом, которое было возвращено функцией **DDEInitiate**( ).

Продлав необходимые Вам действия по обмену, необходимо завершить его закрытием канала одним из операторов **DDETerminate** или **DDETerminateAll**.

Каждая из программ может открывать и закрывать только свои каналы.

#### Ошибки:

В результате выполнения оператора могут генерироваться следующие коды ошибок:

ERR-CMD-NOT-SUPPORTED, если программа выполняется не в среде Windows;

ERR-INVALID-CHANNEL, если неправильно задан номер канала.

#### Пример:

```
DDETerminate i_chan_num
```

#### Смотрите также:

**DDEInitiate**, **DDETerminateAll**

### Оператор DDETerminateAll

**Назначение:**

Закрывает все каналы DDE-связи, которые были открыты программой MapBasic.

**Предупреждение:**

Использование этого оператора возможно только в среде Microsoft Windows.

**Синтаксис:**

**DDETerminateAll**

**Описание:**

Оператор **DDETerminateAll** закрывает все каналы DDE-связи, которые были открыты "клиентом", программой MapBasic. Программы MapBasic могут быть запущены одновременно, и в каждой может быть открыто множество каналов. Оператор **DDETerminateAll** закрывает только те каналы, которые были открыты программой, выполнившей этот оператор. То есть, каждая из прикладных программ может открывать и закрывать только свои каналы.

Проделав необходимые Вам действия по обмену, необходимо завершить его закрытием канала одним из операторов **DDETerminate** или **DDETerminateAll**.

**Ошибки:**

В результате выполнения оператора может генерироваться код ошибки:

ERR-CMD-NOT-SUPPORTED, если программа выполняется не в среде Windows.

**Смотрите также:**

**DDEInitiate, DDETerminate**

## Оператор Declare Function

### Назначение:

Объявляет имя функции и список ее параметров.

### Предупреждения:

Этот оператор не может быть использован в окне MapBasic.

Вызов внешних функций (во втором варианте синтаксиса) зависит от вычислительной платформы. К DLL-библиотекам могут обращаться только Windows-программы; вызов функций XFCN возможен только на компьютерах Macintosh.

### Синтаксис (вариант 1):

```
Declare Function fname  
    ( [ [ ByVal ] parameter As var_type ]  
      [ , [ ByVal ] parameter As var_type ... ] ) As return_type
```

где

*fname* – имя функции;

*parameter* – имя параметра функции;

*var\_type* – стандартный для MapBasic тип данных для параметра или определенный с помощью оператора **Type**;

*return\_type* – стандартный для MapInfo скалярный тип для величины, полученной в результате.

### Синтаксис (вариант 2 – для Windows DLL или для Macintosh XFCN)):

```
Declare Function fname Lib "file_name" [ Alias "function_alias" ]  
    ( [ [ ByVal ] parameter As var_type ]  
      [ , [ ByVal ] parameter As var_type ... ] ) As return_type
```

где

*fname* – имя вызываемой функции;

*file\_name* – имя DLL-файла (для Windows) или имя файла, содержащего XFCN (для Macintosh);

*function\_alias* – настоящее имя внешней функции;

*parameter* – имя параметра функции;

*var\_type* – стандартный для MapBasic тип данных для параметра или определенный с помощью оператора **Type** (в Macintosh – только стандартный скалярный тип);

*return\_type* – стандартный для MapInfo скалярный тип для величины, полученной в результате.

### Описание:

Оператор **Declare Function** объявляет об использовании функции, написанной на языке MapBasic, или внешней функции, а также объявляет типы параметров и результата функции.

Для написания функции на MapBasic используется оператор **Function... End Function**. Каждая такая функция должна быть объявлена оператором **Declare Function**.

Параметры могут пересылать значения функции двумя способами: ссылкой ("by-reference") или значением ("by-value"). Для того, чтобы объявить параметр значением, надо перед его именем использовать ключевое слово **ByVal**. По умолчанию параметр объявляется ссылкой. О различии в природе этих двух типов параметров можно прочитать в 4 главе *Руководства пользователя MapBasic*.

### Обращение к внешней функции

Второй вариант синтаксиса используется для объявления внешней функции, которая может быть написана на другом языке (например, C или Pascal), или может храниться в отдельном файле. Если Ваша программа использует внешнюю функцию, то она должна быть объявлена так же, как и внутренняя функция программы MapBasic.

В Windows внешняя функция может быть вызвана из динамически загружаемой библиотеки (Microsoft Windows Dynamic Link Library или DLL). В Macintosh – из XFCN. Следует отметить, что синтаксис для обеих платформ ничем не различается. Дополнительную информацию о Windows DLL Вы можете получить в 11 главе *Руководства пользователя MapBasic*. Если оператор **Declare Function** объявляет внешнюю функцию, то параметр *file\_name* должен задавать имя файла, содержащего его. Файл с внешними процедурами должен быть доступен во время выполнения программ.

Каждая внешняя функция имеет имя, которое ей дает автор при создании. Если это имя совпадет с именем стандартной функции MapBasic или не устраивает нас по другой причине, то нужно придумать другое имя для обращения к DLL-процедуре и присвоить его переменной *fname*, а настоящее имя, которое ей задал автор, указать в предложении **Alias**. Если нужды в псевдонимах не возникает, то переменной *fname* присваивается настоящее имя функции, а предложение **Alias** не нужно.

Если оператор **Declare Function** включает предложение **Alias**, то параметр *function\_alias* должен совпадать с настоящим именем функции, а параметр *fname* должен задавать имя, по которому к ней обращается MapBasic.

### Ограничения для параметров внешней функции Windows DLL

Вы можете использовать разные типы значений. DLL-библиотека должна быть откомпилирована в режиме упаковки структур ("structure packing"). Смотрите 11 главу *Руководства пользователя MapBasic*.

### Пример:

В этом примере определяется функция пользователя под названием "CubeRoot", которая вычисляет кубический корень из числа. Так как функция "CubeRoot" вызывается в тексте программы до определения самой функции **Function... End Function**, то в этом примере используется оператор **Declare Function**, определяющий аргумент функции и тип возвращаемого значения.

```
Declare Sub Main
Declare Function CubeRoot(ByVal x As Float) As Float
Note Str$( CubeRoot(23) )
Function CubeRoot(ByVal x As Float) As Float
    CubeRoot = x ^ (1 / 3)
End Function
```

Обратите внимание на то, что синтаксис оператора **Declare Function** такой же, как оператора **Function**, только нет слова "Declare".

### Смотрите также:

**Declare Sub, Function... End Function**

### Оператор Declare Sub

#### Назначение:

Объявляет имя и типы параметров подпрограммы.

#### Предупреждение:

Этот оператор не может быть использован в окне MapBasic.

Вызов внешних функций (во втором варианте синтаксиса) зависит от вычислительной платформы. К DLL-библиотекам могут обращаться только Windows-программы; вызов функций XFCN возможен только на компьютерах Macintosh.

#### Синтаксис: 1

```
Declare Sub sub_proc  
[ ( [ ByVal ] parameter As var_type [, ... ] ) ]
```

где

*sub\_proc* – имя подпрограммы;

*parameter* – имя параметра из списка подпрограммы;

*var\_type* – тип значения параметра, стандартный или результат оператора **Type**.

#### Синтаксис (вариант 2 – для Windows DLL или Macintosh XCMD):

```
Declare Sub sub_proc Lib "file_name" [ Alias "sub_alias" ]  
[ ( [ ByVal ] parameter As var_type [, ... ] ) ]
```

где

*sub\_proc* – имя внешней процедуры;

*file\_name* – строка:

в Windows – имя DLL-файла, включая DOS-маршрут,

в Macintosh – имя файла, содержащего XCMD;

*sub\_alias* – оригинальное имя процедуры;

*parameter* – имя параметра процедуры;

*var\_type* – тип значения параметра:

в Windows – стандартные типы или тип, заданный оператором **Type**,

в Macintosh – только скалярные стандартные типы.

#### Описание:

Оператор **Declare Sub** объявляет имена процедур и списки их параметров. Обычно, каждому оператору **Declare Sub** соответствует одна sub-процедура, которая находится ниже соответствующего оператора. Такая процедура создается в программе оператором **Sub... End Sub** и называется внутренней.

Параметры могут пересылать значения процедуре двумя способами: ссылкой ("by-reference") или значением ("by-value"). Для того, чтобы объявить параметр значением, надо перед его именем использовать ключевое слово **ByVal**. По умолчанию параметр объявляется ссылкой. О различии в природе этих двух типов параметров можно прочитать в 4 главе *Руководства пользователя MapBasic*.

#### Обращение к внешним процедурам

Второй вариант синтаксиса используется для объявления внешней процедуры, которая может быть написана на другом языке (например, C или Pascal), или храниться в отдельном файле. Если Ваша программа использует внешнюю процедуру, то она должна быть объявлена так же, как и внутренняя процедура программы MapBasic.

В Windows внешняя процедура может быть вызвана из динамически загружаемой библиотеки (Microsoft Windows Dynamic Link Library или DLL). В Macintosh – из XFCN. В UNIX – из RPC. Следует отметить, что синтаксис для трех платформ ничем не различается. Дополнительную информацию о Windows DLL читайте в 11 главе *Руководства пользователя MapBasic*.

Если оператор **Declare Sub** объявляет внешнюю процедуру, то параметр *file\_name* должен задавать имя файла, содержащего его. Файл с внешними процедурами должен быть доступен во время выполнения программ.

Каждая DLL-процедура имеет имя, которое ей дает автор при создании. Если это имя совпадет с именем стандартной процедуры MapBasic или не устраивает нас по другой причине, то нужно придумать другое имя для обращения к внешней процедуре и присвоить его переменной *fname*, а настоящее имя, которое ей задал автор, указать в предложении **Alias**. Если нужды в псевдонимах не возникает, то переменной *fname* присваивается настоящее имя процедуры, а предложение **Alias** не нужно.

Если оператор **Declare Function** включает предложение **Alias**, то параметр *function\_alias* должен совпадать с настоящим именем функции, а параметр *fname* должен задавать имя, по которому к ней обращается MapBasic.

### Ограничения для параметров внешней процедуры Windows DLL

Вы можете использовать разные типы значений. DLL-библиотека должна быть откомпилирована в режиме упаковки структур ("structure packing"). Смотрите описание оператора **Type**.

### Ограничения для параметров внешней функции Macintosh XCMD

Вы не можете использовать массивы и сложные, специально определенные типы для пересылки значений между программой и внешней функцией.

### Пример:

Процедура **Cube** вычисляет кубический корень от своего первого параметра.

```
Declare Sub Cube(ByVal original As Float,
                cubed As Float)
Dim x, result As Float
Call Cube(2, result)
' переменная result сейчас равна 8 (2 x 2 x 2)
x = 1
Call Cube(x + 2, result)
' переменная result сейчас равна 27 (3 x 3 x 3)
End Sub

Sub Cube (ByVal original As Float, cubed As Float)
    cubed = original ^ 3
End Sub
```

### Смотрите также:

**Call, Sub... End Sub**

## Оператор Define

### Назначение:

Назначает имя для постоянной величины.

### Предупреждение:

Этот оператор не может быть использован в окне MapBasic.

### Синтаксис:

**Define** *identifier* *definition*

где

*identifier* – имя, слово не более 31 символа длиной, начинающееся с буквы или символа подчеркивания (-);

*definition* – значение, которое MapBasic подставит вместо каждого *identifier* в тексте программы.

### Описание:

Оператор **Define** определяет идентификатор *identifier* для постоянной величины. Перед компиляцией MapBasic сначала заменяет каждый идентификатор *identifier* на текст, который определен параметром *definition*. Примеры применения оператора **Define** Вы можете увидеть в файле **MAPBASIC.DEF**.

Строчные и прописные символы в именах идентификаторов не различаются. То есть, если Вы с помощью оператора **Define** задали некоторой величине идентификатор **FOO**, то в тексте программы можно использовать и **Foo**, и **foo**.

В операторе **Define** нельзя применять для имен слова, используемые как ключевые, например, **Set** или **Create**. Список "запретных" слов приведен в описании оператора **Dim**.

### Примеры:

Оператор **Define** делает Вашу программу более понятной, т. к. Вы можете задать осмысленные имена константам. Например, в программе используется число  $\pi$  (пи), которое примерно равно 3.141593. Вы можете присвоить этой константе имя **Pi** и использовать его в тексте программы. Для этого в начале программы напишите:

```
Define PI 3.141593
```

В параметре *definition* оператора **Define** можно использовать кавычки.

```
Define FILE_NAME "World.tab"
```

Следующий оператор входит в состав файла стандартных определений MAPBASIC.DEF. Выполнение этого определения приводит к открытию пустого окна "Сообщение":

```
Define CLS Print Chr$(12)
```



## Функция DeformatNumber\$( )

### Назначение:

Очищает строку, представляющую число, от форматирующих символов.

### Синтаксис:

**DeformatNumber\$(numeric\_string)**

где

*numeric\_string* – строковая величина, представляющая число, такое как “12,345,678”

### Величина, полученная в результате:

Строка. Величина типа String.

### Описание:

Функция возвращает строку, представляющую число. При этом из исходного значения удаляются разделители тысяч, если они были в значении параметра *numeric\_string*. Также заменяется знак отделения десятичной части на точку, если в строке *numeric\_string* он не точка.

### Примеры:

Функция **Val( )**, преобразующая строку в число, воспринимает разделители тысяч как просто текст. Потому для корректного перевода строкового представления числа в число требуется сначала применить функцию **DeformatNumber\$( )**:

```
Dim s_number As String
Dim f_value As Float

s_number = "1,222,333.4"
s_number = DeformatNumber$(s_number)

' теперь переменная s_number имеет
' значение: "1222333.4"

f_value = Val(s_number)

Print f_value
```

### Смотрите также:

**FormatNumber\$( )**, **Val( )**

### Оператор Delete

#### Назначение:

Удаляет один или более графических объектов. Или же удаляет одну или более строк из таблицы.

#### Синтаксис:

**Delete [Object] From table [ Where Rowid = id\_number ]**

где

*table* – имя открытой таблицы;

*id\_number* – номер строки, целое число от 1 и более.

#### Описание:

Оператор **Delete** удаляет графический объект или всю запись, соответствующую этому объекту в таблице.

По умолчанию удаляются все строки таблицы. Если в операторе указано ключевое слово **Object**, то, удалив графические объекты, MapBasic не удалит записи, к которым эти объекты были присоединены.

Параметр *id\_number* из предложения **Where Rowid = ...**, задает номер строки, которая будет удалена из колонки. Если оператор **Delete** не содержит ключевого слова **Object** и в нем не задан номер удаляемой строки, то будут удалены все записи из таблицы.

Оператор **Delete Object From** отличается от оператора **Drop Map**. Первый действует только с объектами или записями и не влияет на структуру таблицы. Оператор **Drop Map**, напротив, изменяет структуру таблицы так, что из нее исключаются графические объекты.

#### Пример 1:

В этом примере оператор **Delete** удаляет все записи из таблицы CLIENTS.TAB. Сама таблица не удаляется, а становится пустой, похожей на ту таблицу, которая создается в MapInfo командой ФАЙЛ > НОВАЯ ТАБЛИЦА.

```
Open Table "clients.tab"
Delete From clients
Commit Table clients
```

#### Пример 2:

Оператор **Delete** удаляет только графический объект, присоединенный к десятой записи таблицы.

```
Open Table "clients.tab"
Delete Object From clients Where Rowid = 10
Commit Table clients
```

#### Смотрите также:

**Drop Map, Insert**

## Оператор Dialog

### Назначение:

Создает новое диалоговое окно.

### Предупреждение:

Этот оператор не может быть использован в окне MapBasic.

### Синтаксис:

```
Dialog
[ Title title ]
[ Width w ] [ Height h ] [ Position x, y ]
[ Calling handler ]
Control control_clause
[ Control control_clause ... ]
```

где

*title* – строковая величина, которая помещается в строку заголовка диалогового окна;

*h* – задает высоту диалогового окна в специальных единицах измерения высоты диалога (высота одного символа в диалоге равна 8 единицам);

*w* – задает ширину диалогового окна в специальных единицах измерения ширины диалога (ширина одного символа в диалоге равна 4 единицам);

*x*, *y* – координаты верхнего левого угла диалога в пикселах относительно верхнего левого угла рабочей области MapInfo's (по умолчанию диалог будет расположен в середине);

*handler* – имя процедуры, которая выполняется перед выводом диалога на экран; обычно в эти процедуры помещаются операторы **Alter Control**.

Каждый параметр *control\_clause* может быть одной из конструкций, которая начинается одним из следующих ключевых слов:

**ButtonOKButtonCancelButton,**  
**CheckBoxGroupBoxRadioGroup**  
**EditTextStaticTextPopupMenu**  
**PenPickerBrushPickerSymbolPicker**  
**FontPicker ListBoxMultiListBox**

Каждому типу предложения **Control** *control\_clause* посвящен отдельный раздел этого настоящего *Справочника*.

### Описание:

Оператор **Dialog** создает диалоговое окно произвольного вида для организации диалога программы с пользователем.

Это так называемый модалый диалог; другими словами, пользователь должен закрыть диалог (например, нажав на кнопку "ОК" или "Отмена") для того, чтобы продолжить работу с MapInfo.

Все, что находится внутри диалогового окна, называется элементами диалога. Каждый элемент создается с помощью предложения, начинающегося со слова **Control**. Обычно диалог содержит несколько элементов: кнопки подтверждения и отмены, окошко для ввода текста, флажки, списки. Сколько элементов должно содержать диалоговое окно, столько предложений, начинающихся со слова **Control**, должно быть в операторе **Dialog**. Тип элемента диалога задает следующее за **Control** ключевое слово (смотрите выше). Например, для кнопки подтверждения используется предложение **Control OKButton**.

Оператор **Dialog** позволяет создавать Вам диалоговые окна произвольного вида. Если Вы хотите использовать стандартные диалоги в своей программе, используйте следующие операторы и функции: **Ask( )**, **Note**, **ProgressBar**, **FileOpenDlg( )**, **FileSaveAsDlg( )**.

Информацию об основных концепциях построения диалогового окна в MapBasic Вы можете найти в 6 главе *Руководства пользователя MapBasic*.

### Размер и расположение элементов диалога

В операторе **Dialog** размер и расположение элемента диалога измеряются в долях шрифта диалога. Каждая единица измерения ширины в диалоге равна одной четвертой ширины символа (первая координата) и каждая единица измерения высоты в диалоге – одной восьмой высоты символа (вторая координата). Так, если элемент имеет ширину 40 и высоту 40, то это значит, что в элемент можно уместить слово в десять букв и список в пять строк.

Центром координат для определения места элементов диалога взят верхний левый угол окна диалога, он имеет координаты (0,0). Следующий пример определяет расположение в диалоге на пять букв правее и на две буквы ниже левого верхнего угла окна диалога:

Предложения **Position**, **Height** и **Width** не являются обязательными. Если Вы их опустили, то MapBasic разместит элементы по умолчанию в порядке следования соответствующих предложений **Control** в операторе.

Следует учитывать, что один и тот же диалог может в Macintosh показаться больше чем Windows, так как системный шрифт в системе Macintosh крупнее.

### Закрывание диалога

Диалоговое окно, открытое MapBasic оператором **Dialog**, можно закрыть одним из четырех способов:

- пользователь нажимает кнопку элемента **OkButton** (если этот элемент присутствует в диалоге);
- пользователь нажимает кнопку элемента **CancelButton** (если этот элемент присутствует в диалоге);
- пользователь отменяет диалог, используя системное меню диалогового окна или клавишу ESC;
- пользователь указывает на элемент диалога, обработчик которого выполняет оператор **Dialog Remove**.

Обычно для закрытия используются кнопки, создаваемые элементами **OKButton** или **CancelButton**. Но, используя обработчики для этих элементов, Вы можете перехватывать закрытие диалога. Например, после нажатия кнопки "ОК" появляется окно с вопросом: "Прежние значения записи будут утеряны. Вы хотите сохранить таблицу?". Если Вы нажали на кнопку с отрицательным ответом, Вы возвращаетесь в первый диалог.

Для достижения этого эффекта используется оператор **Dialog Preserve**. Оператор работает только в подпрограммах-обработчиках элементов **OkButton** и **CancelButton**.

### Чтение введенных значений

После оператора **Dialog** с помощью функции **CommandInfo( )** можно определить, как был закрыт диалог пользователем, кнопкой "ОК" или "Отмена". Если пользователь нажал на кнопку "ОК", то функция

```
CommandInfo(CMD_INFO_DLG_OK)
```

вернет значение TRUE.

Для определения, какие значения ввел пользователь в окошки диалога, какие режимы выбрал, есть два способа: включить предложение **Into** в состав оператора **Dialog** или вызвать функцию **ReadControlValue( )** из процедуры-обработчика.

Если используется предложение **Into** и диалог был закрыт кнопкой типа **OKButton**, MapInfo присвоит финальные значения из элементов диалога соответствующим переменным. Заметим, что MapInfo обновляет значения зарезервированных переменных, только когда пользователь закрыл диалог.

Прочитать значения из диалога, пока он не закрыт, можно только из процедуры-обработчика, вызвав функцию **ReadControlValue( )**.

### Задание клавишных сокращений для элементов диалога

Если прикладная программа выполняется в среде MapInfo для Windows, то элементам диалога могут быть назначены клавишные сокращения, которые позволяют пользователю обращаться к элементам диалога с клавиатуры.

Символ для клавишного сокращения выбирается из заголовков и подписей элементов, которые задаются элементам в предложении **Title**. В тексте заголовка перед нужным символом надо поставить знак амперсанда (&). На экране этот символ подчеркивается. Например, следующая конструкция создает элемент диалога **Button** с клавишным сокращением ALT+B

```
Control Button
Title "&Восстановить"
```

Если Вы хотите вывести сам знак амперсанда в заголовок элемента, то используйте его два раза. Например, предложение:

```
Title "&Вернуть && обработать"
```

создаст следующую подпись элемента "Вернуть & обработать".

Если Вы задаете элемент **StaticText** сразу до или после элемента **EditText**, то определенное клавишное сокращение в тексте **StaticText** можно использовать для перехода в окошко элемента **EditText**.

Если программу, создающую диалог с использованием клавишных сокращений, запустить в среде компьютера Macintosh, то сокращения будут проигнорированы.

### Порядок элементов диалога для клавиши TAB

Для изменения фокуса в диалоге при помощи клавиатуры пользователь может использовать клавишу TAB. Фокус перемещается от элемента к элементу в том порядке, в котором соответствующие предложения **Control** следуют в операторе **Dialog**.

### Пример 1:

Следующий оператор **Dialog** строит диалог с окошком для ввода текста пользователем.

```
Dialog
Title "Поиск строки"
Control StaticText
Title "Введите строку для поиска:"
Control EditText
Value gs_searchfor ' это строковая глобальная переменная
```

```
    Into gs_searchfor
Control OKButton
Control CancelButton
    Title "Отмена"
If CommandInfo(CMD_INFO_DLG_OK) Then
    ' ... если пользователь нажал на кнопку "ОК",
    ' то переменной "gs_searchfor" присваивается текст,
    ' который ввел пользователь.
End If
```

### Пример 2:

В следующем фрагменте Вы сможете найти применение элементов диалога всех типов.

```
Declare Sub resetsub      ' установка стандартных значений
Declare Sub oksub        ' подпрограмма, подтверждающая
                          ' нажатие кнопки ОК.

Declare Sub Main
Sub Main
    Dim maptitle As String      ' заголовок карты
    Dim showlegend As Logical  ' TRUE означает показ легенды
    Dim details As SmallInt    ' 1 = полная детализация; 2 = частичная
    Dim quarter As SmallInt    ' 1 = за 1-ый квартал;... 4 = за 4-ый,
                                ' 5 = за весь год
    Dim mapscope As SmallInt   ' 1 = карта города;2 = области;
                                ' 3 = территории;
                                ' 4 = региона;5 = карта всей страны
    Dim storesymbol As Symbol  ' стиль символа для показа торговых точек
    Dialog
        Title "Карта торговых точек"
        Width 210
        Height 210
        Control StaticText
            Title "Заголовок карты:"      Position 5, 10
        Control EditText
            Value "Торговые точки"      Position 65, 8   Width 120
            ID 1
            Into maptitle
        Control GroupBox
            Title "Уровень детализации"    Position 5, 30
            Height 40   Width 90
        Control RadioGroup
            Title "&Всё;В&ыборчно"
            Value 2
            ID 2
            Into details
            Position 12, 40   Width 72
        Control StaticText
            Title "Показывать пункты как:"
            Position 110, 30
        Control SymbolPicker
            Position 140, 42
            Into storesymbol
```

```

Control StaticText
  Title "Показать результаты для:" Position 5, 80
Control ListBox
  Title "1-ого квартала;2-ого квартала;3-его квартала;
        4-ого квартала;текущего месяца;года"
  ID 3
  Value 5
  Into quarter
  Position 10, 92 Height 40
Control StaticText
  Title "Показать слои:" Position 110, 80
Control MultiListBox
  Title "Улицы;Шоссе;Города;Области;Территории"
  ID 4
  Value 3
  Position 115, 92 Height 40
Control StaticText
  Title "Охват карты:" Position 5, 145
Control PopupMenu
  Title "Город;Область;Территория;Регион;Вся страна"
  Value 2 ID 5
  Into mapscope
  Position 10, 157
Control CheckBox
  Title "Показывать &Легенду"
  Into showlegend
  ID 6
  Position 115, 155
Control Button
  Title "&Вернуть"
  Calling resetsub
  Position 10, 190
Control CancelButton
  Title "Отмена"
  Position 110, 190
Control OKButton
  Title "ОК"
  Calling oksub
  Position 160, 190

If CommandInfo(CMD_INFO_DLG_OK) Then
' ... пользователь выбирает кнопку "ОК"
Else
' ... пользователь выбирает кнопку "Отмена".
End If
End Sub
Sub resetsub
'
' Восстановление Начальных значений в диалоге
'
Alter Control 1 Value "Торговые точки"

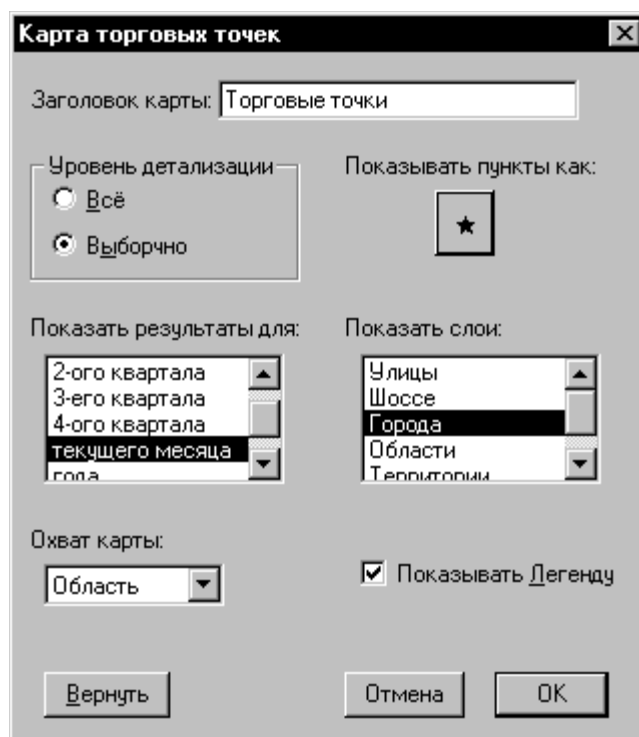
```

## Оператор Dialog

---

```
Alter Control 2 Value 2
Alter Control 3 Value 5
Alter Control 4 Value 3
Alter Control 5 Value 2
Alter Control 6 Value 1
End Sub
```

Результатом этой процедуры будет следующий диалог:



### Смотрите также:

[Alter Control](#), [Ask\( \)](#), [Dialog Preserve](#), [Dialog Remove](#), [FileOpenDlg\( \)](#), [FileSaveAsDlg\( \)](#), [Note](#), [ReadControlValue\( \)](#)



## Оператор Dialog Preserve

### Назначение:

Открывает диалог снова после нажатия клавиш типа "ОК" или "Отмена".

### Синтаксис:

**Dialog Preserve**

### Предупреждение:

Этот оператор может быть использован только в процедуре обработчика кнопок типа **OkButton** и **CancelButton**.

Этот оператор не может быть использован в окне MapBasic.

### Описание:

Оператор **Dialog Preserve** позволяет приостановить закрытие диалога, созданного в программе оператором **Dialog**, нажатием на кнопки типа **OkButton** или **CancelButton**.

Оператор **Dialog Preserve** позволяет Вам подтверждать закрытие диалога. Например, пользователь нажимает на кнопку "ОК" (элемент диалога **OkButton**). Обработчик кнопки вызывает диалог, спрашивающий: "Вы хотите заменить старые значения колонки на новые?" (функция **Ask( )**). Если пользователь выберет кнопку "Нет", обработчик выполнит оператор **Dialog Preserve** и вернет выполнение в первоначальный диалог.

### Пример:

Следующий отрывок может быть процедурой-обработчиком кнопки **CancelButton**.

```
Sub confirm_cancel

    If Ask("Изменения будут утеряны. Продолжить?",
        "Да", "Нет") = FALSE Then

        Dialog Preserve

    End If

End Sub
```

### Смотрите также:

**Alter Control, Dialog, Dialog Remove, ReadControlValue( )**

## Оператор Dialog Remove

### Назначение:

Закрывает диалог, созданный оператором **Dialog**.

### Синтаксис:

**Dialog Remove**

### Предупреждение:

Оператор работает только в процедуре обработчика элемента диалога.

Вы не можете использовать оператор в окне MapBasic.

### Описание:

Оператор **Dialog Remove** закрывает диалог, построенный с помощью оператора **Dialog**.

Автоматически диалоговое окно закрывается по нажатию на кнопки **OkButton** или **CancelButton**.

Оператор **Dialog Remove** позволяет выйти из диалога до выбора этих кнопок. Это полезно, например, если диалоговое окно имеет списковый элемент диалога **ListBox**. Указав дважды на элемент списка, Вы можете закрыть диалог.

### Пример:

Эта процедура – часть программы NIEWS.MB. Она выполняет роль обработчика спискового элемента (**ListBox**) диалога "Именованные Виды". Если пользователь указывает в список, используя одно нажатие на клавишу мышки, обработчик оставляет возможность закрыть диалог при помощи кнопок. Если пользователь указывает в список, используя двойное нажатие на клавишу мышки, обработчик использует оператор **Dialog Remove** для закрытия диалогового окна. Заметим, что Map-Info вызывает этот обработчик и тогда, когда клавиша мышки нажимается один раз, и тогда, когда пользователь использует двойное нажатие.

```
Sub listbox_handler
  Dim i As SmallInt
  Alter Control 2 Enable
  Alter Control 3 Enable
  If CommandInfo(CMD_INFO_DLG_DBL) = TRUE Then
    ,
    ' ... когда в списке было использовано
    '     двойное указание.
    ,
    i = ReadControlValue(1)
    Dialog Remove
    Call go_to_view(i)
  End If
End Sub
```

### Смотрите также:

**Alter Control, Dialog, Dialog Preserve, ReadControlValue( )**

## Оператор Dim

### Назначение:

Объявляет тип одной или более переменных.

### Предупреждение:

Если оператор **Dim** используется в окне MapBasic, то один оператор **Dim** может объявить тип только одной переменной. В окне MapBasic нельзя объявить массив переменных.

### Синтаксис:

```
Dim var_name [, var_name ... ] As var_type
[, var_name [, var_name ... ] As var_type ... ]
```

где  
*var\_name* – имя объявляемой переменной;  
*var\_type* – тип для переменной.

### Описание:

Оператор **Dim** объявляет имя и тип локальной переменной перед употреблением их в процедуре.

В следующей таблице перечислены все стандартные типы переменных, которые Вы можете объявить в операторе **Dim**:

Значение <i>var_type</i>	Описание
SmallInt	Короткое целое число от -32767 до 32767 включительно; используется два байта.
Integer	Целое число от -2147483647 до +2147483647 включительно; используется четыре байта.
Float	Число с плавающей запятой; занимает восемь байт в формате IEEE.
String	Символьная строка не более 32767 байт.
String * <i>length</i>	Символьная строка фиксированной длины (здесь <i>length</i> задает длину строки до 32767 байт). Заранее предполагается, что это строки пробелов.
Logical	TRUE или FALSE, 1 или 0; используется два байта.
Date	Дата в формате MM/DD/YYYY, используется четыре байта: два байта – год, один байт – месяц, один байт – день.
Object	Графический объект (Точечный, Регион, Линия, Полилиния, Дуга, Прямоугольник, Сглаженный Прямоугольник, Эллипс, Текстовый или Рамка).
Alias	Имя колонки.
Pen	Установка стиля линии.
Brush	Установка стиля штриха.
Font	Установка стиля шрифта.
Symbol	Установка стиля символа для точечного объекта.

### Место для оператора Dim и область использования переменных в программе

Каждая локальная переменная должна быть объявлена в процедуре или функции перед тем, как на неё ссылаться (то есть внутри конструкций **Sub...End Sub** и **Function...End Function**). Другими словами, оператор **Dim** должен предшествовать тем операторам, в которых эти переменные используются. Обычно операторы **Dim** для всех переменных, используемых в процедуре, располагаются в первых строках процедуры или функции.

Если оператор **Dim** используется вне тела процедуры или функции, то объявляются переменные для многомодульного уровня. Значения таких переменных могут использоваться во всех процедурах и функциях одного модуля, который участвует в сборке проекта.

Для объявления глобальных переменных используется оператор **Global**. Значение глобальной переменной доступно всем процедурам и функциям всех модулей проекта.

### Объявление нескольких переменных в одном операторе

Один оператор **Dim** позволяет объявлять сразу несколько переменных одного типа. Напишите их имена через запятую:

```
Dim i_counter, i_min, i_max As Integer
Dim s_name As String
```

Можно также в одном операторе объединить объявление нескольких переменных разного типа. Например:

```
Dim i_counter, i_min, i_max As Integer, s_name As String
```

### Массивы переменных

MapBasic поддерживает одномерные массивы переменных. Если в операторе **Dim** после имени *var\_name* стоит пара скобок, то это имя понимается как имя массива. При объявлении типа массива переменных Вы можете также задать размерность массива. Например, следующему массиву вещественных переменных задана размерность десять:

```
Dim f_stats(10) As Float
f_stats(1) = 17.23
```

Число в скобках во второй строке примера является индексом массива, и задает порядковый номер элемента массива.

Изменить размерность массива в ходе программы можно оператором **ReDim**. Для определения текущей размерности массива используется функция **UBound( )**. Если оператор **Dim** объявляет массив с пустыми скобками, то массиву присваивается нулевая размерность. При этом массив не занимает в памяти места. Для его использования в нужный момент Вы всегда можете задать ему новую размерность оператором **ReDim**.

В 16-битной версии Windows массив MapBasic может содержать не более 7000 элементов. В Macintosh и 32-битной версии Windows – не более 32767 элементов.

### Строковые переменные

Длина строковой переменной не должна превышать 32 К. Однако символьная строка, участвующая в операции присваивания (как в следующем примере), не должна превышать 256 символов.

```
Dim status As String
status = "Это строковая константа... "
```

При выполнении оператора **Dim** MapBasic, подобно другим BASIC-языкам, автоматически заполняет строковые переменные фиксированной длины пробелами. Другими словами, когда Вы объявляете строковую переменную в 10 байт, а в нее помещаете пятисимвольную строку, то оставшиеся пять позиций будут заполнены пробелами. (Это может быть полезно при формировании таблицы из строк текста.) Строковые переменные свободной длины при объявлении ничем не заполняются, вернее, значение переменной типа String сразу после выполнения оператора объявления равно пустому значению.

В следующем фрагменте оператор **If... Then** определяет равенство на самом деле неравных строк:

```
Dim s_var_len As String
Dim s_fixed_len As String * 10
s_var_len = "тест"
s_fixed_len = "тест"
If s_var_len = s_fixed_len Then
    Note "строки равны" ' этого никогда не случится
Else
    Note "строки не равны" ' это случится обязательно
End If
```

## Ограничения при выборе имени для переменной

В именах переменных большие и маленькие буквы не различаются. Так, если оператор **Dim** объявил переменную *abc*, то в дальнейшем к ней можно обращаться и *abc*, и *ABC*, и *Abc*.

Имя переменной может состоять не более чем из 31 символа, которыми могут быть буквы латинского алфавита, цифры и знак подчеркивания. Имена переменных могут также использовать следующие знаки пунктуации: \$, %, &, !, # и @, но только в конце имени. Например, именем переменной может быть *LastName\$*, но не *Last\$Name*. Имя переменной также не может начинаться с цифры.

Для имен переменных Вы также не можете использовать слова, которые использует язык MapBasic как ключевые, например, такие как **Open**, **Close**, **Set** и **Do**. Если Вы объявили переменную под именем “*Set*”, то при компиляции MapBasic сгенерирует ошибку. В следующей таблице приведены слова, которые *не должны* использоваться в качестве имен переменных.

Add	Drop	Insert	Rename
Alter	Else	Layout	Resume
Browse	Elseif	Map	Rollback
Call	End	Menu	Run
Close	Error	Note	Save
Commit	Event	Objects	Seek
Create	Exit	OnError	Select
DDE	Export	Open	Set
DDEExecute	Fetch	Pack	Shade
DDEPoke	For	Print	StatusBar

## Оператор Dim

---

DDETerminate	Function	PrintWin	Stop
DDETerminateAll	Get	ProgressBar	Sub
Declare	Global	Put	Type
Delete	Goto	ReDim	Update
Dialog	Graph	Register	While
Dim	If	Reload	
Do	Import	Remove	

В некоторых BASIC-языках тип переменной диктует написание ее имени. Например, если переменная имеет имя со знаком доллара в конце (*LastName\$*), то она понимается как строковая. В языке MapBasic это не работает, тип переменной надо объявлять явно.

### Начальные значения переменных

При объявлении численных переменных MapBasic присваивает им начальное значение 0 (ноль). Строковые переменные неопределенной длины имеют начальное значение пустая строка (""). Строковые переменные фиксированной длины заполняются пробелами. Объектным переменным и переменным типа Pen, Brush, Font и Symbol при объявлении не присваивается начальных значений. Перед использованием этих переменных Вы должны их инициализировать.

### Пример:

```
' Создадим сложный тип данных Person,
' используя оператор Type
Type Person
    Name As String
    Age As Integer
    Phone As String
End Type
' Следующий оператор Dim объявляет переменную типа Person
Dim customer As Person
' Этот оператор Dim объявляет массив переменных типа Person :
Dim users(10) As Person
' Этот оператор Dim объявляет целочисленную переменную "counter"
' массив целочисленных переменных "counters" :
Dim counter, counters(10) As Integer
' Этот оператор присваивает элементу "Name" первого значения
' массива "users" следующее значение :
users(1).Name = "Поликарп"
```

### Смотрите также:

[Global](#), [ReDim](#), [Type](#), [UBound\( \)](#)

## Функция Distance( )

### Назначение:

Возвращает расстояние между двумя точками.

### Синтаксис:

**Distance**(*x1, y1, x2, y2, unit\_name*)

где

*x1* и *x2* – X-координаты (или широты);

*y1* и *y2* – Y-координаты (или долготы);

*unit\_name* – строка, задающая имя единицы измерения (например, "km")

### Величина, полученная в результате:

Вещественное число. Величина типа Float

### Описание:

Функция **Distance( )** вычисляет расстояние между двумя определенными точками и возвращает значение в указанных единицах. Список всех возможных имен единиц измерения приведен в описании оператора **Set Distance Units**.

Координаты X и Y понимаются MapBasic относительно заданной системы координат. Если система координат не объявлялась, то используется система широта/долгота. Объявить систему координат можно оператором **Set CoordSys**.

Если текущей системой координат является система координат Земли, то функция **Distance( )** возвращает расстояние между двумя точками по дуге большого земного сечения (большой окружности, полученной в результате сечения земного шара плоскостью, заданной этими двумя точками и центром Земного шара). Если текущая система координат – система координат для непроекцированных карт (планов), функция **Distance( )** возвращает расстояние в Декартовых координатах.

### Пример:

```
Dim dist, start_x, start_y, end_x, end_y As Float
Open Table "cities"
Fetch First From cities
start_x = CentroidX(cities.obj)
start_y = CentroidY(cities.obj)
Fetch Next From cities
end_x = CentroidX(cities.obj)
end_y = CentroidY(cities.obj)
dist = Distance(start_x, start_y, end_x, end_y, "mi")
```

### Смотрите также:

**Area( ), ObjectLen( ), Set CoordSys, Set Distance Units**

### Оператор Do Case...End Case

#### Назначение:

Выполняет ту группу операторов, условия для которой истинны.

#### Предупреждение:

Оператор **Do Case** не может быть выполнен в окне MapBasic.

#### Синтаксис:

```
Do Case do_expr  
    Case case_expr [, case_expr ]  
        statement_list  
[ Case... ]  
[ Case Else  
    statement_list ]  
End Case
```

где

*do\_expr* – выражение;

*case\_expr* – выражение, результат которого будет сравниваться со значением *do\_expr*;

*statement\_list* – группа операторов.

#### Описание:

Оператор **Do Case**, подобно оператору **If... Then... Else**, проверяет истинность определенных выражений и, в зависимости от результата, выполняет одну из групп операторов. Оператор **Do Case** в языке MapBasic аналогичен оператору **Select Case** в языке BASIC. (Имя оператора изменено для того, чтобы не возникало путаницы с оператором **Select**.)

Выполняя оператор **Do Case**, MapBasic вычисляет выражение *case\_expr* за первым операционным словом **Case**. Если результат *case\_expr* равен результату вычисления выражения *do\_expr*, то MapBasic выполнит группу операторов *statement\_list*, которая расположена за первым словом **Case**; и после этого управление программой передается оператору, следующему после ключевых слов **End Case**.

Если результаты вычислений выражений *do\_expr* и *case\_expr* из первого предложения **Case** не равны, MapBasic переходит к проверке выражения *case\_expr* из следующего предложения **Case**. И так далее. Проверяется каждое условие *case\_expr* по порядку, пока не будет найдено равное *do\_expr*.

Если равного выражения не будет найдено вообще и в операторе нет предложения **Case Else**, то не будет выполнено ни одного оператора из групп *statement\_list*. Если есть предложение **Case Else** и не найдено ни одного равенства, то MapBasic выполнит операторы *statement\_list*, заключенные между ключевыми словами **Case Else** и **End Case**.

Заметим, что оператор **Do Case** такой формы:

```
Do Case expr1  
Case expr2  
    statement_list1  
Case expr3, expr4  
    statement_list2  
Case Else  
    statement_list3  
End Case
```

работает так же, как оператор **If... Then ... Else** следующей конструкции:



```
      If expr1 = expr2 Then
        statement_list1
      ElseIf expr1 = expr3 Or expr1 = expr4 Then
        statement_list2
      Else
        statement_list3
      End If
```

### Пример:

В примере строятся такие текстовые строки: "Первый квартал", "Второй квартал" и т.п., в зависимости от текущего месяца.

```
Dim cur_month As Integer, msg As String
cur_month = Month( CurDate( ) )
Do Case cur_month
  Case 1, 2, 3
    msg = "Первый квартал"
  Case 4, 5, 6
    msg = "Второй квартал"
  Case 7, 8, 9
    msg = "Третий квартал"
  Case Else
    msg = "Четвертый квартал"
End Case
```

### Смотрите также:

**If... Then... Else**

### Оператор Do...Loop

#### Назначение:

Выполняет последовательность действий по циклу до тех пор, пока выполнится (или, наоборот, не выполнится) некое условие.

#### Предупреждение:

Оператор **Do Loop** не может быть выполнен в окне MapBasic.

#### Синтаксис (вариант 1):

```
Do
    statement_list
Loop [ { Until | While } condition ]
```

#### Синтаксис (вариант 2):

```
Do [ { Until | While } condition ]
    statement_list
Loop
```

где

*statement\_list* – группа операторов для одного шага цикла;

*condition* – условие, управляющее выполнением цикла.

#### Описание:

Оператор повторяет группу операторов *statement\_list*, расположенных между словами **Do** и **Loop** до тех пор, пока выражение *condition* за предложением **While** не станет истинным (примет значение TRUE), либо так долго, пока выражение *condition* за предложением **Until** истинно (все время цикла будет равным TRUE).

Если цикл **Do... Loop** не содержит слов **Until/While**, то управление выполнением цикла ведется из операторов *statement\_list*. Это можно осуществить с помощью следующих операторов: **Goto**, **Exit Do**. Оператор **Exit Do** немедленно прекращает любой цикл **Do... Loop** независимо от присутствия слов **Until/While** и передает управление программой оператору, следующему за словом **Loop**.

Предложения **Until** или **While** могут находиться как после слова **Do**, так и после **Loop**. В зависимости от этого проверка выполнения условия происходит до или после выполнения операторов *statement\_list*. Это важно для первой итерации. Следующая конструкция:

```
Do
    statement_list
Loop While condition
```

выполняет операторы *statement\_list* и потом вычисляет *condition*. Если *condition* равно TRUE, MapBasic вновь выполняет *statement\_list*. И так далее, пока *condition* не станет равным FALSE. Цикл прекращается, и программа выполняет следующие операторы.

В следующей конструкции:

```
Do While condition
    statement_list
Loop
```

сначала вычисляется *condition*, и, если условие равно TRUE, MapBasic выполнит *statement\_list*.

#### Пример:

В цикле **Do... Loop** читаются первые пять записей таблицы:

```
Dim sum As Float, counter As Integer
```

```
Open Table "world.tab"  
Fetch First From world  
counter = 1  
Do  
    sum = sum + world.population  
    Fetch Next From world  
    counter = counter + 1  
Loop While counter <= 10
```

**Смотрите также:**

Exit Do, For... Next

### Оператор Drop Index

#### Назначение:

Удаляет индекс из таблицы.

#### Синтаксис:

**Drop Index** *table(column)*

где

*table* – имя открытой таблицы;

*column* – имя колонки в таблице.

#### Описание:

Оператор **Drop Index** отменяет индексирование колонки в открытой таблице. Удаление индекса происходит непосредственно на диске, где расположена таблица. (Для создания индекса вновь Вы можете воспользоваться оператором **Create Index**).

**Замечание:** MapInfo не может отменить индексирование в таблице с несохраненными изменениями. Используйте сначала оператор **Commit** для сохранения таблицы.

Оператор **Drop Index** не имеет обратного действия и не требует последующего сохранения изменений на диске, т.е. не работают команды **ФАЙЛ > ВОССТАНОВИТЬ** и **ПРАВКА > ОТМЕНИТЬ** в MapInfo, а также оператор MapBasic **Rollback**.

#### Пример:

Следующий код отменяет индексирование колонки "Страна" в таблице WORLD:

```
Open Table "world.tab"  
Drop Index world(Страна)
```

#### Смотрите также:

**Create Index**

## Оператор Drop Map

### Назначение:

Удаляет все графические объекты из таблицы. Не может быть использована со связанными таблицами.

### Синтаксис:

**Drop Map** *table*

где *table* – имя открытой таблицы.

### Описание:

Используйте оператор **Drop Map** для удаления сразу всех графических объектов (точка, линия, регион, окружность и т. п.) из открытой таблицы, изменяя структуру таблицы так, что к ее записям уже не могут присоединяться графические объекты.

Оператор **Drop Map** не имеет обратного действия и не требует последующего сохранения изменений на диске. Т. е. не работают команды ФАЙЛ > ВОССТАНОВИТЬ и ПРАВКА > ОТМЕНИТЬ в MapInfo.

Оператор MapBasic **Rollback** также не может отменить действие оператора **Drop Map**. Поэтому **Вы должны быть очень внимательны, используя оператор Drop Map.**

После выполнения оператора **Drop Map**, соответствующая таблица более не может быть показана в окне Карты. Для изменения структуры таблицы так, чтобы к ней можно было вновь присоединять графические объекты, используйте оператор **Create Map**. Оператор **Drop Map** не влияет на данные в записях таблицы, которые можно выводить в окно Списка.

Если Вам надо удалить все графические объекты без изменения структуры таблицы, воспользуйтесь оператором **Delete Object** вместо **Drop Map**.

Оператор **Drop Map** не работает со связанными таблицами.

### Пример:

```
Open Table "clients"  
Drop Map clients
```

### Смотрите также:

**Create Map, Create Table, Delete**

### Оператор Drop Table

#### Назначение:

Удаляет таблицу полностью.

#### Синтаксис:

**Drop Table** *table*

где *table* – имя открытой таблицы.

#### Описание:

Оператор **Drop Table** используется для удаления открытой таблицы с диска.

При этом удаляется как сам файл таблицы, так и сопутствующие файлы, а также файлы других форматов (баз данных и электронных таблиц), которые были использованы для создания файла таблицы MapInfo.

Оператор **Drop Table** имеет непосредственный эффект и не имеет обратного действия, т.е. не работают команды ФАЙЛ > ВОССТАНОВИТЬ и ПРАВКА > ОТМЕНИТЬ в MapInfo, а также оператор MapBasic **Rollback**.

С оператором Drop Table нужно быть осторожным!

Заметим, что многие операции с таблицами в MapInfo помещают результат во временные таблицы (например, Запрос1). Временная таблица удаляется автоматически с закрытием сеанса работы MapInfo и нет необходимости применять оператор **Drop Table** для удаления временной таблицы.

Временные таблицы, образующиеся в результате запросов, автоматически удаляются при закрытии MapInfo.

Таблицы, которые имеют более сложную структуру, нельзя удалить оператором **Drop Table**.

Например, таблицы формата StreetInfo, использующие в своем определении предложение "View", и фактически объединяющие две таблицы в одну, нельзя удалить оператором **Drop Table**.

#### Пример:

```
Open Table "clients"  
Drop Table clients
```

#### Смотрите также:

**Create Table, Delete, Kill**

### Оператор End MapInfo

#### Назначение:

Прекращает выполнение программы на MapBasic и закрывает MapInfo.

#### Синтаксис:

**End MapInfo [ Interactive ]**

#### Описание:

Оператор **End MapInfo** используется для остановки выполнения прикладной программы и завершения сеанса работы MapInfo.

Программа может содержать процедуру-обработчик завершения работы, для которой в MapBasic зарезервировано имя **EndHandler**. При выполнении оператора **End MapInfo** эта процедура будет автоматически загружена.

Если оператор **End MapInfo** выполнен тогда, когда были открыты таблицы и в них были произведены изменения, не сохраненные в файлах, MapInfo выдаст запрос о сохранении этих изменений.

Если оператор использует ключевое слово **Interactive**, и если были открыты окна Карт с объектами на Косметическом слое или с не сохраненными тематическими объектами, то MapInfo выдаст пользователю сообщения, предлагающие сохранить эти объекты. Однако, если в MapInfo включен режим автоматического сохранения Рабочего Набора MAPINFOW.WOR перед закрытием, то сообщения выдаваться не будут. Если слово **Interactive** опущено, то предложений сохранить косметические и тематические объекты выдаваться не будут.

Для остановки программы на MapBasic без завершения сеанса работы MapInfo используйте оператор **End Program**.

#### Смотрите также:

**End Program, EndHandler**

### Оператор End Program

**Назначение:**

Прекращает выполнение прикладной программы.

**Предупреждение:**

Оператор **End Program** не может быть выполнен в окне MapBasic.

**Синтаксис:**

**End Program**

**Описание:**

Оператор **End Program** используется для остановки выполнения прикладной программы.

Прикладная программа – это программа, написанная на MapBasic и откомпилированная. Если программа дополняет меню новыми системами команд и снабжает их процедурами-обработчиками, то после закрытия программы оператором **End Program** удаляются как изменения в системе меню, так и модули-обработчики событий.

Если в программе есть процедура с именем **EndHandler**, то MapBasic автоматически выполнит ее при остановке программы, каким бы образом эта остановка не совершалась.

**Смотрите также:**

**End MapInfo**



## Процедура EndHandler

### Назначение:

Специальная подпрограмма, sub-процедура, автоматически выполняющаяся при завершении выполнения программы.

### Синтаксис:

**Declare Sub EndHandler**

**Sub EndHandler**  
    *statement\_list*  
**End Sub**

где

*statement\_list* – список операторов.

### Описание:

**EndHandler** – зарезервированное имя для процедуры MapBasic. Процедура **EndHandler** называется обработчиком завершения программы. Когда пользователь запускает программу, в которой есть обработчик завершения программы, то эта процедура будет автоматически вызвана при завершении программы. Обработчик сработает как при закрытии MapInfo, так и при остановке выполнения программы оператором **End Program** или любым другим способом.

Если несколько MapBasic-программ снабжены обработчиками завершения программы, то при закрытии MapInfo каждая из них сработает по очереди.

### Смотрите также:

**RemoteMsgHandler, SelChangedHandler, ToolHandler, WinChangedHandler, WinClosedHandler**

### Функция EOF( )

#### Назначение:

Возвращает FALSE, если MapBasic читает запись из файла, и TRUE, если пробует прочитать что-то после конца файла.

#### Синтаксис:

**EOF(*filenum*)**

где *filenum* – номер открытого файла оператора **Open File**.

#### Величина, полученная в результате:

Логическая. Величина типа Logical.

#### Описание:

Функция **EOF( )** позволяет определить, достигнут ли конец файла, открытого в программе под номером *filenum*. Возвращается логическое "да" (TRUE), если оператор **Get** попытался прочитать запись, следующую после последней, и логическое "нет" (FALSE), если была прочитана запись файла.

Функция **EOF( )** работает с открытыми файлами (текстовыми или произвольной природы); если же Вы хотите отследить конец чтения из открытой таблицы MapInfo, то используйте функцию **EOT( )**.

Пример использования функции **EOF( )** можно посмотреть в тексте программы NIEWS.MB.

#### Ошибки:

В результате выполнения оператора может генерироваться код ошибки  
ERR-FILEMGR-NOTOPEN, если файл не был открыт.

#### Смотрите также:

**EOT( )**, **Open File**

## Функция EOT( )

### Назначение:

Возвращает FALSE, если MapBasic читает запись из таблицы, и TRUE, если пробует прочитать что-то после конца таблицы.

### Синтаксис:

**EOT(*table*)**

где *table* – имя открытой таблицы.

### Величина, полученная в результате:

Логическая. Величина типа Logical.

### Описание:

Функция **EOT( )** позволяет определить, достигнут ли конец таблицы, открытой в программе под именем *table*. Возвращается логическое "да" (TRUE), если MapBasic попытался прочитать запись, следующую после последней, и логическое "нет" (FALSE), если прочитана строка таблицы.

### Ошибки:

В результате выполнения оператора может генерироваться код ошибки ERR-TABLE-NOT-FOUND, если таблица не найдена.

### Пример:

Здесь результат функции **EOT( )** является критерием прекращения выполнения цикла.

```
Dim total As Float
Open Table "customer.tab"
Fetch First From customer
Do While Not EOT(customer)
    total = total + customer.order
    Fetch Next From customer
Loop
```

### Смотрите также:

**Fetch, EOF( ), Open File, Open Table**

### Функция Erase( )

#### Назначение:

Возвращает объект, полученный из другого удалением части, которая перекрывается другим объектом.

#### Синтаксис:

**Erase** (*source\_object*, *eraser\_object*)

где

*source\_object* – объект, кроме точечного или текстового;

*eraser\_object* – замкнутый объект, выполняющий роль "ластика".

#### Величина, полученная в результате:

Величина типа Object.

#### Описание:

Функция **Erase( )** возвращает объект, полученный в результате удаления части объекта *source\_object*, перекрываемой объектом *eraser\_object*, выполняющим роль ластика.

Объект *source\_object* может быть линейным (прямой линией, полилинией или дугой) или замкнутым объектом (область, прямоугольник, скругленный прямоугольник или эллипс), но не может быть точечным или текстовым объектом. Объект *eraser\_object* должен быть замкнутым.

Объект, полученный в результате, унаследует стиль оформления (тип линии, штриховки и цвет) от объекта *source\_object*.

#### Пример:

```
' В этом примере o1 и o2 - объектные переменные,
' которые уже имеют значения.
If o1 Intersects o2 Then
    If o1 Entirely Within o2 Then
        Note "Удаление не состоялось: первый объект
            полностью перекрывается "ластиком". "
    Else
        o3 = Erase( o1, o2 )
    End If
Else
    Note "Удаление не состоялось: объекты не пересекаются."
End If
```

#### Смотрите также:

**Objects Erase, Objects Intersect**

## Функция Err( )

### Назначение:

Возвращает числовой код, соответствующий текущей ошибке.

### Синтаксис:

**Err( )**

### Величина, полученная в результате:

Целое число. Величина типа Integer.

### Описание:

Функция **Err( )** возвращает целочисленный код, соответствующий последней из происшедших ошибок.

Обычно, когда в прикладной программе генерируется ошибка, выводится сообщение о ней и выполнение программы прекращается. Если в программе есть процедура обработчика ошибки, вход в которую обеспечивает оператор **OnError**, выполнение программы передается обработчику, как только какой-нибудь оператор, находящийся после **OnError**, возвращает ошибку. В обработчике ошибок Вы можете применить функцию **Err( )** для определения происшедшей ошибки.

После того, как программа выполнит оператор **Resume** для возвращения из обработчика в процедуру, вызвавшую ее, код ошибки в программе обнуляется. Поэтому не имеет смысла использовать функцию **Err( )** вне обработчика ошибок, т. к. функция, вызванная после обработчика, вернет ноль.

В этой книге после описаний некоторых операторов и функций следует раздел "Ошибки:", в котором перечисляются коды некоторых возможных ошибок, генерируемых описываемым оператором или функцией. Но это не все возможные ошибки, а только связанные с наиболее часто встречающимися ситуациями.

Некоторые коды ошибок MapBasic генерируются только в специфических, редко встречающихся ситуациях. Например, код ERR-INVALID-CHANNEL генерируется только функциями и операторами, обслуживающими DDE-связь. Если оператор может генерировать такой "специальный" код, то он также приводится в разделе "Ошибки:" для этого оператора.

Другие ошибки MapBasic, напротив, порождаются часто и многими функциями. Например, функции **CentroidX( )**, **Area( )** и **ObjectInfo( )** используют выражение типа Object как параметр. Этими функциями может генерироваться код ошибки ERR-FCN-OBJ-FETCH-FAILED, если Вы задали параметр в форме *tablename.obj*, тогда как текущая строка этой таблицы не имеет присоединенного объекта.

Аналогично, можно еще упомянуть коды двух математических ошибок – ERR-FP-MATH-LIB-DOMAIN и ERR-FP-MATH-LIB-RANGE, которые возникают в результате ошибки в численном параметре. Эти ошибки могут возникать в случае выполнения любой из следующих функций: **Asin( )**, **Acos( )**, **Atn( )**, **Cos( )**, **Exp( )**, **Log( )**, **Sin( )**, **Sqr( )** или **Tan( )**.

Полный список кодов ошибок MapBasic находится в файле ERRORS.DOC. (в Macintosh этот файл называется Error List).

### Смотрите также:

**Error**, **Error\$( )**, **OnError**

### Оператор Error

#### Назначение:

Генерирует ошибку определенного кода.

#### Синтаксис:

**Error** *error\_num*

где

*error\_num* – целочисленный код, соответствующий стандартной ошибке в MapBasic.

#### Описание:

Оператор **Error** генерирует ошибку по заданному коду.

Если в программе до этого была определена процедура обработчика ошибок (смотрите оператор **OnError**), то MapBasic передает управление процедуре-обработчику ошибок. Если обработчик отсутствует в программе, то после выполнения оператора **Error** MapBasic выводит сообщение о ней и останавливает выполнение программы.

Оператор **Error** обычно используется при отладке.

#### Смотрите также:

**Err( ), Error\$( ), OnError**

## Функция **Error\$( )**

### Назначение:

Возвращает сообщение о текущей ошибке.

### Синтаксис:

**Error\$( )**

### Величина, полученная в результате:

Строка. Величина типа String.

### Описание:

Функция **Error\$( )** возвращает текст сообщения о последней ошибке, если она есть. Если ошибки нет, то функция возвращает пустую строку.

Функция **Error\$( )** может быть вызвана только из процедуры-обработчика ошибок. Смотрите также описание функции **Err( )**.

### Смотрите также:

**Err( )**, **Error**, **OnError**

### Оператор Exit Do

#### Назначение:

Прекращает действие **Do**-цикла.

#### Предупреждение:

Оператор **Exit Do** не может быть выполнен в окне MapBasic.

#### Синтаксис:

**Exit Do**

#### Описание:

Оператор прекращает выполнение цикла, организованного оператором **Do... Loop**, и передает управление первому оператору после цикла.

Цикл **Do... Loop** может быть вложенным, так что один цикл может быть внешним по отношению к другому. Оператор **Exit Do** прекращает выполнение только того цикла, внутри которого употребляется.

Так в конструкции:

```
      Do While условие1
...
      Do While условие2
...
      If условие_ошибки
        Exit Do
      End If
...
    Loop
...
  Loop
```

оператор **Exit Do** прерывает вложенный цикл (организованный вторым оператором **Do While условие2**), не останавливает первый цикл (**Do While условие1**).

#### Смотрите также:

**Do... Loop, Exit For, Exit Sub**



## Оператор Exit For

### Назначение:

Оператор **Exit For** не может быть выполнен в окне MapBasic.

### Предупреждение:

Оператор **Exit For** не может быть выполнен в окне MapBasic.

### Синтаксис:

**Exit For**

### Описание:

Оператор прекращает выполнение цикла, организованного оператором **For...Next**, и передает управление первому оператору после конструкции цикла. Надо заметить, что оператор **Exit For** отменяет только цикл, организованный оператором **For...Next**.

Цикл **For... Next** может быть вложенным, так что один цикл может быть внешним по отношению к другому. Оператор **Exit For** прекращает выполнение только того цикла, внутри которого употребляется.

Так в конструкции:

```
      For x = 1 to 5
      ...
      For y = 2 to 10 step 2
      ...
      If условие_ошибки
        Exit For
      End If
      ...
    Next
  ...
Next
```

оператор **Exit For** прерывает вложенный цикл (организованный вторым оператором **For y = 2 to 10 step 2,**) не останавливает первый цикл (**For x = 1 to 5**).

### Смотрите также:

**For...Next, Exit Do**

### Оператор Exit Function

**Назначение:**

Прекращает действие функции, заданной операторами **Function... End Function**.

**Предупреждение:**

Оператор **Exit Function** не может быть выполнен в окне MapBasic.

**Синтаксис:**

**Exit Function**

**Описание:**

Оператор прекращает выполнение функции на MapBasic. Оператор **Exit Function** действителен только для функции, организованной при помощи оператора **Function... End Function**.

Функция **Function... End Function** может быть вложенная, то есть одна функция может вызывать другую. Оператор **Exit Function** прекращает выполнение только той функции, внутри которой употребляется.

**Смотрите также:**

**Function... End Function**

### Оператор Exit Sub

**Назначение:**

Прекращает выполнение sub-процедуры.

**Предупреждение:**

Оператор **Exit Sub** не может быть выполнен в окне MapBasic.

**Синтаксис:**

**Exit Sub**

**Описание:**

Оператор **Exit Sub** осуществляет выход из текущей процедуры, организованной при помощи оператора **Sub... End Sub**. Оператор может выполняться в теле только таких процедур.

Одна sub-процедура может вызывать другую sub-процедуру, которая, в свою очередь, вызывает третью и т. д. Один оператор **Exit Sub** прекращает выполнение только той подпрограммы, внутри которой употребляется.

**Смотрите также:**

**Call, Sub... End Sub**

### Функция Exp( )

#### Назначение:

Вычисляет значение экспоненты.

#### Синтаксис:

**Exp(*num\_expr*)**

где *num\_expr* – численное выражение.

#### Величина, полученная в результате:

Вещественное число. Величина типа Float.

#### Описание:

Математическая функция **Exp( )** вычисляет значение числа  $e$ , возведенного в степень *num\_expr*.

Число  $e$  иррационально и приблизительно равно 2.7182818.

Операция возведения в степень производится с помощью математического оператора ^.

#### Пример:

```
Dim e As Float
e = Exp(1)
' переменная e примет значение
' равное примерно 2.7182818
```

## Оператор Export

### Назначение:

Экспортирует таблицу в файл другого формата.

### Синтаксис (вариант 1 - для экспорта в формат MIF/MID, DBF или ASCII)

```
Export table
Into filespec
[ Type
    { "MIF" |
      "DBF" [ CharSet char_set ] |
      "ASCII" [ CharSet char_set ] [ Delimiter "d" ] [ Titles ] } ]
[ Overwrite ]
```

### Синтаксис (вариант 2 - для экспорта в формат DXF)

```
Export table
Into file_name
[ Type "DXF" ]
[ Overwrite ]
[ Preserve [ AttributeData ] [ Preserve ] [ MultiPolygonRgns [ As Blocks ] ] ]
[ { Binary | ASCII [ DecimalPlaces decimal_places ] } ]
[ Version { 12 | 13 } ]
[ Transform
    (MI_x1, MI_y1) (MI_x2, MI_y2)
    (DXF_x1, DXF_y1) (DXF_x2, DXF_y2) ]
```

где

*table* – имя открытой таблицы (в этом операторе кавычки при задании имени таблицы не используются);

*filespec* – строка с именем файла, в которую будут экспортироваться данные (если строка не включает DOS-маршрут, то файл будет создан в рабочем каталоге);

*char\_set* – величина типа String, слово, определяющее кодировку символов в экспортированном файле, такое как “MacRoman” или “WindowsLatin1” (смотрите описание предложения **CharSet**);

*d* – символ, который будет использован как разделитель для ASCII-файла;

*decimal\_places* – целое число типа small integer (короткое целое) от 0 до 16 (по умолчанию 6), задающее количество позиций после десятичной точки для экспорта числа с плавающей точкой в текстовый формат ASCII;

*MI\_x1*, *MI\_y1* и т. п. – пары чисел, представляющие собой координаты в таблице MapInfo;

*DXF\_x1*, *DXF\_y1* и т. п. – пары чисел, представляющие собой координаты в файле DXF.

### Описание:

Оператор **Export** копирует содержимое таблицы MapInfo в отдельный файл другого формата для работы с этим файлом в других программах и операционных средах. Например, Вы можете экспортировать содержимое таблицы в DXF-файл, чтобы импортировать его в AutoCAD или другую программу, которая работает с форматом DXF. Исходный файл с таблицей при экспорте не меняется.

### Задание формата файла

Формат будущего файла объявляется при помощи предложения **Type**.

Предложение Type	Формат будущего файла
Type "MIF"	Формат обмена MapInfo (MapInfo Interchange File format). Описание формата MIF смотрите в документации MapInfo.
Type "DXF"	Формат DXF (графический формат программ CAD, таких как AutoCAD).
Type "DBF"	Формат dBASE и других совместимых баз данных. <b>Замечание:</b> графические объекты не экспортируются при использовании этого формата.
Type "ASCII"	Текстовый формат с разделителями. <b>Замечание:</b> графические объекты не экспортируются при использовании этого формата.

Если предложение **Type** отсутствует, MapInfo пытается определить формат экспорта по расширению в имени файла. Например, если в операторе задается имя "PARCELS.DXF", то MapInfo создаст файл формата DXF.

Если файл с заданным именем уже существует, MapInfo создаст экспортный файл, записав его поверх старого только тогда, когда в **Export** используется слово **Overwrite**. Если слово **Overwrite** опущено и файл с заданным именем уже есть на диске, MapInfo не будет переписывать файл.

### Экспорт в текстовые ASCII-файлы

Если таблица экспортируется в текстовый ASCII-файл, то последний будет содержать разделители. Разделителем называется специальный символ, разделяющий данные полей в строке записи. Для назначения символа-разделителя используется параметр *d* за словом **Delimiter**. Если предложение **Delimiter** отсутствует, то разделителем будет символ табуляции (Chr\$(9)). В следующем примере символом разделителя назначается двоеточие (:):

```
Export sites Into "sitedata.txt"  
Type "ASCII" Delimiter ":" Titles
```

При импорте в коды ASCII, если указано ключевое слово **Titles**, MapBasic первой строкой в экспортном файле поместит запись из заголовков колонок таблицы. При обратной операции (импорт экспортного файла) эта строка может быть прочитана как строка заголовков колонок.

### Экспорт в DXF-файл

Для получения файла в формате DXF используется второй вариант синтаксиса оператора **Export**, который может включать следующие специфические для формата DXF предложения и ключевые слова:

#### **Preserve AttributeData**

Это предложение используется для экспорта табличных данных в атрибуты DXF-файла.

#### **Preserve MultiPolygonRgns As Blocks**

Используется для того, чтобы MapInfo экспортировала область, состоящую из нескольких полигонов, в качестве блока DXF. Если это предложение будет опущено, то каждый полигон сложносоставной области будет представлен отдельным объектом.

### **Binary** или **ASCII** [ **DecimalPlaces** *decimal\_places* ]

Ключевое слово **Binary** используется для экспорта таблицы в бинарный файл DXF. Предложение **ASCII** задает текстовый вид файла DXF. Если не используется ни один из этих вариантов, то MapInfo создаст файл ASCII DXF. Бинарный файл DXF в основном меньше, чем текстовый, и поэтому изготавливается быстрее. При создании тестового файла можно задать число знаков после десятичной точки для экспорта численных данных, имеющих тип Float в таблице MapInfo. Параметр *decimal\_places* может принимать значения от 0 до 16, по умолчанию 6.

### **Version 12** или **Version 13**

Это предложение указывает, для какой версии программы будет создан файл DXF. Если предложение не используется, MapInfo будет экспортировать в DXF-файл 12-ой версии.

### **Transform**

Предложение используется для задания координатного искажения. После слова **Transform** задаются координатные пары минимума и максимума в таблице MapInfo и какие пары координат будут соответствовать им в файле DXF.

### Пример:

Здесь открывается таблица "FACILITY " и экспортируется в файл FACIL.DXF.

```
Open Table "facility"
Export facility
  Into "FACIL.DXF"
  Type "DXF"
  Overwrite
  Preserve AttributeData
  Preserve MultiPolygonRgns As Blocks
  ASCII DecimalPlaces 3
  Transform (0, 0) (1, 1) (0, 0) (1, 1)
```

### Смотрите также:

**Import**

### Функция ExtractNodes( )

#### Назначение:

Возвращает полилинию или область из подмножества узлов существующего объекта.

#### Синтаксис:

**ExtractNodes(object, polygon\_index, begin\_node, end\_node, b\_region)**

где

*object* – объект типа "полилиния" или "область";

*polygon\_index* – короткое целое число, 1 или больше; в случае области задает номер полигона, в случае полилинии – номер ломаной;

*begin\_node* – короткое целое число, 1 или больше, которое задает первый узел для выбираемого подмножества узлов;

*end\_node* – короткое целое число, 1 или больше, которое задает последний узел для выбираемого подмножества узлов;

*b\_region* – логическая величина, управляющая типом объекта, полученным в результате:  
для области – логическое "да" (TRUE), для полилинии – логическое "нет" (FALSE).

#### Величина, полученная в результате:

Объект типа "полилиния" или "область". Величина типа Object.

#### Описание:

Если значение параметра *begin\_node* равно или больше значения *end\_node*, то MapBasic будет отсчитывать узлы в следующем порядке:

1. от узла *begin\_node* до последнего в полигоне;
2. от первого узла следующего полигона до *end\_node*.

Если параметр *object* задает область и оба параметра *begin\_node* и *end\_node* равны 1 (единице), то MapBasic вернет полное множество узлов одного полигона (многоугольника). Это простой механизм для выделения одного полигона из многокомпонентной области. Для определения числа узлов полигона из области используется функция **ObjectInfo( )**.

MapBasic присваивает все стили (цвет и т. п.) оригинального объекта *object* полученному в результате объекту. Если каких-то стилей недостает, то берутся текущие в MapBasic значения.

#### Ошибки:

Функция может вернуть код ошибки

ERR-FCN-ARG-RANGE, если значение параметра *b\_region* равно FALSE и подмножество узлов состоит менее, чем из двух узлов или если значение параметра *b\_region* равно TRUE и подмножество узлов состоит менее, чем из трех узлов.



## Оператор Fetch

### Назначение:

Задание текущей позиции курсора в таблице (т. е. какая строка таблицы должна быть текущей)

### Синтаксис:

**Fetch { First | Last | Next | Prev | Rec *n* } From *table***

где

*n* – номер записи для чтения;

*table* – имя открытой таблицы.

### Описание:

Оператор **Fetch** используется для позиционирования записи в открытой таблице. Выполняя оператор, Ваша программа помещает курсор на определенную строку таблицы. Это состояние записи в таблице называется “текущим”.

**Замечание:** Термин “курсор” используется здесь для отражения расположения строки в таблице, и никак не связан с визуальным курсором экрана.

После оператора **Fetch** приложение MapBasic может извлекать данные из текущей записи, используя выражения:

Синтаксис	Пример
<i>table.column</i>	World.Country
<i>table.col#</i>	World.col1
<i>table.col ( number )</i>	World.col( <i>variable_name</i> )

Оператор **Fetch First** переводит курсор на первую неудаленную запись таблицы.

Оператор **Fetch Last** переводит курсор на последнюю неудаленную запись таблицы.

Оператор **Fetch Next** переводит курсор на следующую неудаленную запись таблицы.

Оператор **Fetch Prev** переводит курсор на предыдущую неудаленную запись таблицы.

Оператор **Fetch Rec *n*** переводит текущую позицию на определенную строку, даже если она удалена.

**Замечание:** если запись удалена, то оператор генерирует ошибку выполнения 404.

Надо учитывать, что многие операторы MapBasic и команды MapInfo сбрасывают текущее положение курсора в таблице (например, оператор **Select**, оператор **Update**, перерисовка экрана). Используйте обращение к текущей строке сразу после выполнения оператора **Fetch**.

### Определение конца таблицы

После выполнения оператора **Fetch** функция **EOT( )** возвращает логическую величину (TRUE или FALSE) в зависимости от того, достигнут ли предел таблицы после последней записи.

Если оператор **Fetch** помещает курсор на действительную запись таблицы, то функция **EOT( )** возвращает значение FALSE.

Если оператор **Fetch** попытается установить текущей запись после последней записи таблицы, то функция **EOT( )** вернет значение TRUE.

## Оператор Fetch

---

В следующем примере оператор **Fetch Next** проходит по циклу по всем строкам таблицы. После каждого применения оператора вызывается функция **EOT( )** и, если она вернет истинное значение, то цикл будет прерван.

```
Dim i As Integer

i = 0
Fetch First From world
Do While Not EOT(world)
    i = i + 1
    Fetch Next From world
Loop

Print "Число неудаленных записей: " + i
```

### Примеры:

В этом примере показано, как подвести курсор к 3-ей записи в таблице STATES:

```
Open Table "states"
Fetch Rec 3 From states      'позиционирование на 3-ю запись
Note states.abbr             'вывод значения колонки Abbr в записи '
```

Таким же образом можно доставать значения из временных таблиц (например, из таблицы Selection).

```
Select * From states Where pop90 > pop80
Fetch First From Selection
Note Selection.col1 + " содержит отрицательный прирост населения"
```

### Смотрите также:

**EOT( ), Open Table**

## Функция FileAttr( )

### Назначение:

Возвращает информацию об открытом файле.

### Синтаксис:

**FileAttr**(*filenum*, *attribute*)

где

*filenum* – номер файла, открытого оператором **Open File**;

*attribute* – код возвращаемого атрибута.

### Величина, полученная в результате:

Целое число. Величина типа Integer или Small Integer.

### Описание:

Функция **FileAttr( )** используется для получения информации об открытом в MapInfo файле.

Параметр *attribute* должен принимать одно из следующих значений:

Значение <i>attribute</i>	Результат
FILE_ATTR_MODE	Короткое целое, величина типа Small Integer, соответствующая коду режима, в котором был открыт файл: MODE_INPUT MODE_OUTPUT MODE_APPEND MODE_RANDOM MODE_BINARY
FILE_ATTR_FILESIZE	Целое число, величина типа Integer, размер файла в байтах.

### Ошибки:

В результате выполнения функции Вы можете получить код ошибки

ERR-FILEMGR-NOTOPEN, если файл не открыт.

### Смотрите также:

**EOF( ), Get, Open File, Put, TableInfo**

### Функция FileExists( )

#### Назначение:

Возвращает логическую величину (TRUE или FALSE), в зависимости от того, существует файл или нет.

#### Синтаксис:

**FileExists(*filespec*)**

где *filespec* – строка с полным именем файла, включая DOS-маршрут.

#### Величина, полученная в результате:

Логическая. Величина типа Logical.

#### Пример:

```
If FileExists("C:\MapInfo\TODO.TXT") Then  
    Open File "C:\MapInfo\TODO.TXT" For INPUT As #1  
  
End If
```

#### Смотрите также:

**TempFileName\$( )**

## Функция FileOpenDlg( )

### Назначение:

Вызывает диалоговое окно команды ФАЙЛ > ОТКРЫТЬ ТАБЛИЦУ из меню в MapInfo и возвращает имя файла, выбранное пользователем.

### Синтаксис:

**FileOpenDlg(path, filename, filetype, prompt)**

где

*path* – строковая величина, определяющая начальное значение выбранного каталога;

*filename* – строковая величина, определяющая начальное значение выбранного файла в каталоге (в Macintosh этот параметр должен быть пустым (""));

*filetype* – строковая величина в три символа или меньше, определяющая тип файла;

*prompt* – строковая величина, задающая заголовок диалога.

### Величина, полученная в результате:

Строка с именем файла или пустая, если была отмена. Величина типа String.

### Описание:

Функция **FileOpenDlg( )** открывает диалоговое окно подобно команде MapInfo ФАЙЛ > ОТКРЫТЬ ТАБЛИЦУ.

Чтобы выбрать файл из списка, представленного в диалоге, Вы можете указать на имя файла и нажать на кнопку ОК, или просто дважды указать мышкой на имя файла. В этом случае функция возвращает полное имя выбранного файла, включая DOS-маршрут. Если Вы нажали на кнопку "Отмена", то возвращаемая строка будет пуста ("").

Заметим, что функция **FileOpenDlg( )** не открывает выбранный файл. Для того, чтобы открыть этот файл, необходимо выполнить оператор **Open Table**. Если же Вам нужно, чтобы в Вашей программе выбранный файл открывался автоматически, используйте оператор **Run Menu Command** с использованием аргумента **M\_FILE\_OPEN** или **M\_FILE\_ADD\_WORKSPACE**.

Параметр *path* назначает каталог, в котором помещены файлы для выбора. Значение каталога является начальным, и ничто не мешает пользователю выбрать другой каталог в диалоговом окне. Если параметр не задан (пустое значение), то каталогом выбора будет текущий рабочий каталог.

Параметр *filename* определяет начальное значение имени файла, выбранного в диалоге. Если программа запускается в среде Macintosh, то в этом параметре Вы должны задать пустую строку ("").

Параметр *filetype* принимает строковые значения обычно длиной в три символа и определяет тип файлов для выбора в диалоговом окне. И если, например, параметр посылает значение "TAB", то диалог покажет список файлов таблиц MapInfo, если посылается значение "WOR", то диалог покажет список файлов с Рабочими Наборами MapInfo.

Существуют и другие трехсимвольные значения. Эти специальные значения параметра *filetype* приведены в следующей таблице (только из трех символов). Если надо вывести список всех файлов, то в качестве параметра *filetype* используется маска "\*. \*".

Значения <i>type</i>	Типы файлов
"TAB"	Таблицы MapInfo.

## Функция FileOpenDlg( )

---

"WOR"	Рабочие Наборы MapInfo.
"MIF"	Формат обмена данными MapInfo (MapInfo Interchange Format), используется для импорта/экспорта карт в/из ASCII-текстовых файлов.
"DBF"	Формат dBASE или других совместимых баз данных.
"WKS", "WK1"	Табличные файлы Lotus.
"XLS"	Табличные файлы Excel.
"XLS3"	Табличные файлы Excel 3.0 (только для Macintosh).
"DXF"	Формат обмена данными AutoCAD.
"MMI", "MBI"	Формат обмена данными с MapInfo для DOS.
"MB"	Файлы с текстами программ MapBasic.
"MBX"	Файлы с откомпилированными программами MapBasic.
"TXT",	Текстовые файлы.
"TEXT"	Текстовые файлы (только для Macintosh).
"BMP"	Формат растрового образа Windows.
"WMF"	Метафайлы Windows.
"PICT"	Формат растрового образа Macintosh.

В Windows каждому типу соответствует определенное DOS-расширение имени файла. Другими словами, тип файла "WOR" говорит MapBasic, чтобы был показан список файлов с расширением ".WOR", т. к. имена с таким расширением имеют файлы Рабочих Наборов в MapInfo для Windows.

Если Вы хотите, чтобы Вашу программу можно было использовать в любой системе, используйте только трехсимвольную установку параметра *filetype*. Например, если Вы пишете прикладную программу для работы в MapInfo для Macintosh, которая показывает диалог со списком файлов Рабочих Наборов, созданных в Macintosh, то задайте тип "WOR" (параметр *filetype*). Файлы Рабочих Наборов в Macintosh могут и не иметь расширения имени ".WOR", но зато каждый такой файл имеет встроенный тип. При вызове функции **FileOpenDlg( )** MapBasic автоматически конвертирует трехсимвольный тип *filetype* в принятый в системе Macintosh код типа файла.

В Windows некоторые специальные значения *filetype* соответствуют строчке меню в нижнем левом углу диалога. Если в функции **FileOpenDlg( )** параметр *filetype* задает значение не из приведенного списка, то в диалоге в окошке списка типов файлов будет пустое место.

### Пример:

```
Dim s_filename As String
s_filename = FileOpenDlg("", "", "TAB", "Open Table")
```

### Смотрите также:

**FileSaveAsDlg( ), Open File, Open Table**

## Функция FileSaveAsDlg( )

### Назначение:

Вызывает диалоговое окно команды ФАЙЛ > СОЗДАТЬ КОПИЮ в MapInfo и возвращает имя файла, введенное пользователем.

### Синтаксис:

**FileSaveAsDlg(*path*, *filename*, *filetype*, *prompt*)**

где

*path* – строковая величина, определяющая начальное значение выбранного каталога;

*filename* – строковая величина, определяющая начальное значение выбранного файла в каталоге;

*filetype* – строковая величина в три символа или меньше, определяющая тип файла (в Macintosh этот параметр должен быть пустым (""));

*prompt* – строковая величина, задающая заголовок диалога.

### Величина, полученная в результате:

Строка с именем файла или пустая строка, если диалог был закрыт кнопкой отмены. Величина типа String.

### Описание:

Эта функция, подобно команде ФАЙЛ > СОЗДАТЬ КОПИЮ в MapInfo, открывает диалоговое окно.

Пользователь может ввести с клавиатуры имя файла или выбрать файл из списка, представленного в диалоге; для этого Вы можете указать на имя файла и нажать на кнопку ОК, или просто дважды указать мышкой на имя файла. Если файл уже существует на диске, MapBasic спросит, действительно ли необходимо заменить старое содержимое файла на новое.

Функция возвращает полное имя введенного или выбранного файла, включая DOS-маршрут после того, как пользователь нажмет на кнопку "ОК". Если Вы нажали на кнопку "Отмена", то возвращаемая строка будет пуста ("").

Параметр *path* назначает каталог, в котором должен быть сохранен файл. Значение каталога является начальным, и ничто не мешает пользователю изменить каталог в диалоговом окне. Если параметр не задан (пустое значение), то каталогом будет текущий рабочий каталог.

Параметр *filename* определяет начальное значение имени файла.

Параметр *filetype* принимает строковые значения обычно длиной в три символа и определяет тип файлов для выбора в диалоговом окне. Значение *filetype* может быть одним из специальных значений. Например, если параметр посылает значение "ТАВ", то диалог покажет список файлов таблиц MapInfo, а если – "WOR", то диалог покажет список файлов с Рабочими Наборами MapInfo. Все специальные значения параметра *filetype* приведены в таблице в описании функции **FileOpenDlg( )**. Если программа запускается в среде Macintosh, то в этом параметре Вы должны задать пустую строку ("").

Функция **FileSaveAsDlg( )** не сохраняет выбранный файл, но использует диалог соответствующей команды для получения полного имени, под которым надо сохранить файл. После этого для сохранения выбранного файла можно выполнить оператор **Commit Table**.

### Смотрите также:

**Commit Table, FileOpenDlg( )**

## Оператор Find

### Назначение:

Поиск объекта или пересечения объектов таблицы на Карте.

### Синтаксис:

**Find** *address* [, *region* ] [ **Interactive** ]

где

*address* – строковая величина, определяющая адрес или имя искомого объекта Карты; для поиска пересечения двух улиц параметр использует синтаксис:

*улица && улица*;

*region* – имя области, в которой будет осуществляться поиск.

### Описание:

Оператор **Find** осуществляет поиск на Карте именованных элементов, заданных параметром *address*. Результаты поиска сохраняются в системных переменных, значения которых доступны функции **CommandInfo**( ).

Если оператор включает ключевое слово **Interactive** и если MapBasic не может сразу обнаружить адрес, то поиск сопровождается диалоговым окном, в котором можно выбирать из нескольких "похожих" адресов.

Оператор **Find** может просматривать только таблицы, имеющие присоединенные графические объекты. Таблица должна быть открыта. Колонка таблицы, используемая для поиска, назначается оператором **Find Using**. Если оператор **Find Using** не был выполнен перед осуществлением поиска оператором **Find**, MapBasic будет вести поиск с установками, которые были заданы ранее в первом диалоге команды ЗАПРОС > НАЙТИ в MapInfo.

Оператор **Find** может уточнять поиск, если Вы зададите необязательный параметр *region* – обычно это имя области, в которой должен содержаться искомый объект. Другими словами, Вы можете просто попробовать найти город по названию (например, "Albany") в таблице городов или сузить область поиска, задав имя города и область, где он может находиться (например, "Albany", "CA"). Подробную информацию об уточнении поиска с использованием таблицы областей Вы можете найти в описании команды ЗАПРОС > НАЙТИ в *Руководстве пользователя MapInfo*.

**Замечание:** Оператор **Find** не отмечает найденный объект символом, так как это делает команда поиска. Если необходимо показать пользователю результаты поиска, то можно использовать функцию **CreatePoint**( ) или оператор **Create Point** (смотрите пример ниже).

### Анализ результата поиска

Если применить функцию **CommandInfo**(CMD\_INFO\_FIND\_RC) после выполнения оператора **Find**, то Вы сможете узнать результат поиска. В случае успеха координаты X и Y найденного объекта можно узнать при помощи функции **CommandInfo**( ) с аргументами **CMD\_INFO\_Y** и **CMD\_INFO\_X** соответственно. Если после оператора **Find** вызвать функцию **CommandInfo**(CMD\_INFO\_FIND\_ROWID), то она вернет номер строки, которой соответствует найденный объект.



Прикладная программа на MapBasic может определить, насколько успешно сработал оператор **Find**, с помощью вызова функции **CommandInfo(CMD\_INFO\_FIND\_RC)**. Оператор **Find** может завершиться либо нахождением объекта, либо выбором ближайшего объекта, либо объект не будет найден. В случае точного попадания функция **CommandInfo(CMD\_INFO\_FIND\_RC)** возвратит единицу. Если Вы выбрали ближайший вариант, то эта функция возвратит число большее единицы. В случае неудачи поиска функция возвращает отрицательное значение.

В следующей таблице собраны различные варианты кодов, возвращаемых функцией **Command-Info(CMD\_INFO\_FIND\_RC)**. Каждое значение задается не более чем тремя значащими цифрами, каждая из которых представляет результат одного из трех возможных этапов поиска.

Значения единиц	Смысл
xx1	Точное совпадение.
xx2	Был задействован файл сокращений.
xx3 ( - )	Точного совпадения не найдено.
xx4 ( - )	Не было задано имя объекта; точного совпадения не найдено.
xx5 ( + )	Пользователь выбрал имя в диалоговом окне <b>Interactive</b> .
Значения десятков	Смысл
x1x	Не была определена сторона улицы.
x2x ( + / - )	Номер дома попал в заданные пределы для данной улицы.
x3x ( + / - )	Номер дома оказался вне заданных пределов для данной улицы.
x4x ( + / - )	Номер дома не задан.
x5x ( - )	Улицы не пересекаются.
x6x ( - )	Улицы пересекаются более одного раза.
x7x ( + )	Пользователь выбрал номер дома в диалоговом окне <b>Interactive</b> .
Значения сотен	Смысл
1xx ( + / - )	Имя найдено только один раз вне уточняющей области.
2xx ( - )	Имя найдено в нескольких регионах, но не в уточняющей области.
3xx ( + / - )	Имя найдено только один раз, но уточняющая область не была задана.
4xx ( - )	Имя найдено несколько раз, но уточняющая область не была задана.
5xx ( + )	Имя найдено несколько раз в уточняющей области.
6xx ( + )	Пользователь выбрал имя области в диалоговом окне <b>Interactive</b> .

## Оператор Find

---

Определить, какая цифра находится в определенном разряде числа Вам поможет арифметический оператор **Mod**. Например, чтобы определить последнюю цифру в числе, используйте деление по модулю 10. Чтобы определить две последние цифры в числе, используйте деление по модулю 100 и так далее.

Разницу между точным совпадением и приблизительным можно пояснить на следующем примере. Если таблица, содержащая имена городов, содержит одну запись для города "Albany", и оператор **Find Using** задает поиск без уточняющей области, и оператор **Find** использует значение "Albany" как параметр *address*, то результатом будет полное совпадение. Следующая за подобной группой операторов функция **CommandInfo(CMD\_INFO\_FIND\_RC)** возвратит 1 (единицу), что означает точное совпадение.

Теперь представим, что оператор **Find** действует в режиме использования уточняющей области; другими словами, оператор **Find** ожидает, что название города будет дополнено названием штата (например, "Albany", "NY"). Если программа выполнит оператор **Find** со значениями "Albany" как адреса и нулевой строки в качестве уточняющего названия штата, то, с технической точки зрения, точного совпадения не будет, потому что MapBasic ожидал явного задания уточняющей области. Однако, так как город "Albany" упоминается только в одной записи таблицы, то MapBasic найдет эту запись. Следующая за подобной группой операторов функция **Command-Info(CMD\_INFO\_FIND\_RC)** возвратит 301. Цифра 1 обозначает совпадение адреса с названием города из таблицы, а цифра 3 означает частичный успех при поиске уточняющей области.

Если таблица улиц содержит запись для улицы "Main St" и оператор **Find** пытается найти улицу "Main Street", MapBasic оценит результат как частичное совпадение (принимая во внимание использование файла сокращений, смотрите также описание **Find Using**). Строго говоря, строки "Main Street" и "Main St" не совпадают. Однако MapBasic определяет равенство этих строк, применяя подстановки из файла сокращений (MAPINFOW.ABB). Следующая за подобной группой операторов функция **CommandInfo(CMD\_INFO\_FIND\_RC)** возвратит 2.

Если операция поиска сопровождается диалогом, в котором пользователь вводит свой текст, то код возврата будет иметь 1 (единицу) в разряде миллионов. Более подробная информация о геокодировании приведена в *Справочнике MapInfo*.

### Пример:

```
Include "mapbasic.def"

Dim x, y As Float, win_id As Integer

Open Table "states" Interactive
Map From States
win_id = FrontWindow()

Find Using states(state)
Find "NY"
If CommandInfo(CMD_INFO_FIND_RC) >= 1 Then
    x = CommandInfo(CMD_INFO_X)
    y = CommandInfo(CMD_INFO_Y)
    Set Map
        Window win_id
        Center (x, y)
```

```
' Теперь создадим объект для маркировки найденного.  
' Точечный объект помещается на Косметическом слое.  
Insert Into  
    WindowInfo( win_id, WIN_INFO_TABLE) (Object)  
    Values ( CreatePoint(x, y) )  
Else  
    Note "Объект не найден."  
End If
```

**Смотрите также:**

**Find Using**

### Оператор Find Using

#### Назначение:

Устанавливает, какие таблицы и какие колонки будут рассматриваться при последующем поиске оператором **Find**.

#### Синтаксис:

```
Find Using table (column)  
[ Refine Using table (column) ]  
[ Options [ Abbrs { On | Off } ]  
          [ ClosestAddr { On | Off } ]  
          [ OtherBdy { On | Off } ]  
          [ Symbol symbol_style ] ]  
[ Inset inset_value { Percent | Distance Units dist_unit } ]  
[ Offset value ] [ Distance Units dist_unit ] ]
```

где

*table* – имя открытой таблицы;

*column* – имя индексированной колонки в таблице;

*symbol\_style* – величина типа Symbol для задания стиля символа, которым будет отмечен найденный объект, если пользователь выполнит команду ЗАПРОС > НАЙТИ.

*inset\_value* - положительное целое значение, определяющее величину смещения от концов улиц для размещения адреса. Если указан Percent, то смещение определяется как процент от длины улицы. Для Percent возможны следующие значения *inset\_value*: от 0 до 50. Если указан Distance Units, *inset\_value* представляет расстояние от концов улицы для размещения адреса. Для distance возможны следующие значения *inset\_value* are - от 0 до 32,767.

*value* - определяет значение отступа Offset (расстояние от улицы - в глубину от улицы). *value* положительное целое значение, определяющее как далеко "отступать" от улицы для размещения адреса. Возможные значения - от 0 до 32,767.

*dist\_unit* - строка, определяющая название единиц измерения расстояний.

#### Описание:

Оператор **Find Using** определяет таблицу (таблицы) и колонку (колонки), в которых будет осуществляться поиск оператором **Find**. Колонки должны быть индексированы.

Предложение **Refine** используется для назначения второй таблицы, уточняющей поиск. Вторая таблица должна содержать объекты типа "область", и заданная колонка должна быть индексирована. Если оператор **Find Using** не содержит предложения **Refine**, то следующий оператор **Find** будет действовать по упрощенной схеме, без уточнений (например, "Быково"). В противном случае оператор **Find** обязательно проведет попытку уточнения, и адрес должен содержать уточняющую компоненту (например, "Быково, Московская область").

Предложение **Abbrs** определяет, будет ли применяться файл сокращений в процедурах поиска. По умолчанию режим применения файла сокращений включен (**On**); чтобы отключить его, нужно явно задать предложение **Abbrs Off**.

Предложение **ClosestAddr** задает режим автоматической подстановки ближайшего адреса в случае не достижения точного совпадения. По умолчанию режим подстановки ближайшего адреса включен (**On**); чтобы отключить его, нужно явно задать предложение **ClosestAddr Off**.

Предложение **OtherBdy** задает режим автоматической подстановки адреса из другого региона, если в заданном уточняющем регионе ничего подходящего не было найдено. По умолчанию этот режим выключен (**Off**); чтобы включить его, нужно явно задать предложение **OtherBdy On**.

Mapinfo сохраняет последние установки для inset и offset сделанные пользователем в настройках поиска в меню ЗАПРОС > НАЙТИ и вариантах геокодирования в меню таблица > геокодирование, или с использованием оператора **Find using**, таким образом, последние настройки становятся параметрами по умолчанию для следующего сеанса работы.

### Пример:

```
Find Using city_1k(city)
  Refine Using states(state)

Find "Albany", "NY"
```

### Смотрите также:

**Create Index, Find**

### Функция Fix( )

#### Назначение:

Возвращает целое число, полученное из целой части действительного числа.

#### Синтаксис:

**Fix(*num\_expr*)**

где *num\_expr* – численное выражение.

#### Величина, полученная в результате:

Целое число. Величина типа Integer.

#### Описание:

Функция **Fix( )** отсекает дробную часть от действительного числа, полученного в результате вычисления выражения *num\_expr*, и возвращает целую часть.

Функция **Fix( )** похожа на функцию **Int( )**, но не идентична. Функции различаются способом удаления дробной части отрицательного числа. Когда *num\_expr* представляет отрицательное число, функция **Fix( )** возвращает ближайшее целое, больше или равное оригиналу. Например:

**Fix(-2.3)**

возвращает значение -2. В случае функции **Int( )** результатом будет ближайшее целое, меньше или равное оригиналу. Например:

**Int(-2.3)**

возвращает значение -3.

#### Пример:

```
Dim whole As Integer
whole = Fix(5.999)
'
' whole сейчас имеет значение 5.
whole = Fix(-7.2)
'
' whole сейчас имеет значение -7.
```

#### Смотрите также:

**Int( ), Round( )**

## Предложение Font

### Назначение:

Определяет шрифт для текстов.

### Синтаксис:

**Font** *font\_expr*

где

*font\_expr* – описание шрифта, величина или имя переменной типа Font, вызов функции, такой как **MakeFont(fontname, style, size, fgcolor, bgcolor)**.

### Описание:

Предложение **Font** не является отдельным оператором, а используется в других операторах, где требуется задать стиль оформления текста (шрифта). Например, оператор **Create Text** позволяет установить атрибуты шрифта будущего текстового объекта: имя шрифта, написание, размер. Если предложение **Font** не употребляется в операторе, то шрифт новому объекту назначается в соответствии с текущим стилем в MapInfo.

За словом **Font** может следовать выражение, имеющее значение типа Font, например, переменная, тип которой объявлен как Font:

**Font font\_var**

или вызов функций, возвращающих такой тип данных (например, **CurrentFont( )** или **MakeFont( )**):

**Font MakeFont("Helvetica", 1, 12, BLACK, WHITE)**

В некоторых операторах MapBasic (например, **Set Legend**) за словом **Font** могут следовать заключенные в скобки пять параметров, описывающих стиль: *fontname, style, point\_size, foreground\_color, background\_color*:

**Font("Helvetica", 1, 12, BLACK, WHITE)**

В следующей таблице приводится описание этих параметров шрифта:

Компонента	Описание
<i>fontname</i>	Строковое значение, имя шрифта. Зависит от набора шрифтов, используемых операционной системой, в которой будет работать приложение MapBasic.
<i>style</i>	Целое число, величина типа Integer, управляющее написанием: <b>жирным шрифтом</b> , <i>курсивом</i> или <u>подчеркнутым</u> . Смотрите следующую таблицу.
<i>point_size</i>	Целое число, величина типа Integer, размер шрифта в пунктах. Размер 12 пунктов соответствует высоте буквы, равной одной шестой дюйма.
<i>foreground_color</i>	Целое число, величина типа Integer, задающее цвет букв по шкале RGB. Смотрите описание функции <b>RGB( )</b> .

## Предложение Font

---

background_color	Целое число, величина типа Integer, задающее цвет фона или каймы текста по шкале RGB. Для задания прозрачного фона текста просто опустите этот параметр. Например, <b>Font( "Helvetica", 1, 12, BLACK)</b> . В функции <b>MakeFont( )</b> пятый параметр должен быть равен 1 для задания прозрачного фона текста.
------------------	--

Следующая таблица содержит значения для параметра *style*, задающего написание шрифта:

Значения <i>style</i>	Стиль написания
0	Нормальный
1	Жирный
2	Курсив
4	Подчеркнутый
16	Контурный. (Поддерживается только в Macintosh.)
32	Оттененный
256	Кайма
512	Полная капителизация
1024	Разреженный

Для задания двух или более стилей написания, значения из левой колонки складываются. Например, для написания жирным шрифтом и только заглавными буквами параметр *style* должен иметь значение 513. MapInfo также имеет три специальных имени шрифта: "Helvetica", "Times" и "Courier", вместо которых подставляются подобные имеющиеся сейчас в системе. Например, если Вы использовали в операторе имя шрифта "Times" и запустили Вашу прикладную программу в системе Microsoft Windows 3.1, то MapInfo автоматически преобразует имя шрифта в "Times New Roman" (имя шрифта стандартной поставки Windows 3.1). MapInfo выполняет автоматическую подстановку только для этих трех имен: "Helvetica", "Times" and "Courier".

### Пример:

```
Include "MAPBASIC.DEF"
Dim titlevar As Object
Create Text
  Into Variable titlevar
  "Здесь будет Ваш текст"
  (73.5, 42.6) (73.67, 42.9)
  Font MakeFont("Helvetica",1,12,BLACK,WHITE)
```

### Смотрите также:

**Alter Object, Chr\$( ), Create Text, RGB( )**



## Оператор For...Next

### Назначение:

Выполняет последовательность действий в цикле с заданным числом итераций.

### Предупреждение:

Оператор **For... Next** не может быть выполнен в окне MapBasic.

### Синтаксис:

```
For var_name = start_expr To end_expr [ Step inc_expr ]  
    statement_list  
Next
```

где

*var\_name* – имя переменной, выполняющей роль счетчика цикла;

*start\_expr* – численное выражение;

*end\_expr* – численное выражение;

*inc\_expr* – численное выражение;

*statement\_list* – группа операторов, выполняющихся за одну итерацию цикла.

### Описание:

Оператор **For... Next** последовательно выполняет группу операторов заданное число раз. Управление циклом происходит при помощи целочисленной переменной *var\_name*, выступающей в роли счетчика. Выражения *start\_expr*, *end\_expr* и *inc\_expr* определяют, сколько раз будет выполнена группа операторов *statement\_list*.

Оператор **For** присваивает счетчику значение, заданное выражением *start\_expr* и первый раз выполняет группу операторов *statement\_list*. Затем значение счетчика увеличивается на *inc\_expr*, и операторы группы *statement\_list* выполняются второй раз и так далее.

Если предложение **Step** *inc\_expr* отсутствует в операторе, то счетчик увеличивается на одну единицу. Оператор **For** повторяет выполнение группы операторов до тех пор, пока значение счетчика меньше или равно *end\_expr*. Как только *var\_name* превысит значение выражения *end\_expr*, цикл будет прекращен. Далее выполнение программы передается оператору, следующему за словом **Next**.

Если задать шаг отрицательным числом в предложении **Step**, то счетчик будет уменьшаться при выполнении цикла. В этом случае оператор **For** будет выполнять каждый шаг цикла до тех пор, пока *var\_name* больше или равно *end\_expr*.

Все операторы, которые находятся между операторами **For** и **Next**, являются операторами тела цикла *statement\_list* и выполняются за одну итерацию (проход) цикла. Эти операторы образуют тело цикла. Для немедленного прекращения цикла, независимо от значения счетчика, используется оператор **Exit For** в теле цикла. Оператор **Exit For** просто передает управление программы следующему оператору после слова **Next**.

MapBasic позволяет Вам изменять величину переменной *var\_name* в теле цикла и тем самым влиять на выполнение цикла. Но для соблюдения хорошего стиля программирования эту возможность использовать не рекомендуется.

## Оператор For...Next

---

### Пример:

```
Dim i As Integer
' Сообщение Note будет выдаваться пять раз
For i = 1 to 5
    Note "Добро пожаловать в MapInfo!"
Next
' Этот цикл выполнится за три шага
For i = 1 to 5 Step 2
    Note "Добро пожаловать в MapInfo!"
Next
' Этот цикл выполнится за три шага
' (на первом шаге i будет равно 5, на втором
' i будет 3, на третьем шаге i равно 1)
For i = 5 to 1 Step -2
    Note "Добро пожаловать в MapInfo!"
Next
' MapBasic не сможет выполнить следующий оператор For
For i = 100 to 50 Step 5
    Note "Это сообщение никогда не появится "
Next
```

### Смотрите также:

Exit For, Do While... Loop, Do... LoopWhile

## Процедура **ForegroundTaskSwitchHandler**

### Назначение:

Процедура, автоматически выполняющаяся при изменении фокуса окна программы MapInfo относительно других программ, выполняющихся в среде Windows.

### Синтаксис:

```
Declare Sub ForegroundTaskSwitchHandler  
Sub ForegroundTaskSwitchHandler  
    statement_list  
End Sub
```

где

*statement\_list* – список операторов процедуры.

### Описание:

**ForegroundTaskSwitchHandler** – зарезервированное имя для процедуры MapBasic. Когда пользователь запускает программу, в которой есть такая процедура, программа не завершается после того, как выполнятся все операторы процедуры Main и другие процедуры, вызванные из нее. Программа будет находиться в режиме ожидания до тех пор, пока рабочее окно MapInfo не получит или потеряет фокус (т.е. сменится активность окна). Как только это произойдет, программа активизируется, выполняя процедуру с именем **ForegroundTaskSwitchHandler**. После выполнения процедуры программа вновь переходит в режим ожидания.

В процедуре **WinChangedHandler** может быть использована функция **CommandInfo()** для определения, приобретает ли MapInfo фокус или теряет его.

### Пример:

```
Sub ForegroundTaskSwitchHandler  
  
    If CommandInfo(CMD_INFO_TASK_SWITCH)  
        = SWITCHING_INTO_MAPINFO Then  
  
        ' ... когда активно окно MapInfo  
    Else  
        ' ... когда активно окно другой программы  
    End If  
  
End Sub
```

### Смотрите также:

**CommandInfo( )**

### Функция **Format\$( )**

#### Назначение:

Возвращает строковое представление числа в заданном формате.

#### Синтаксис:

**Format\$(value, pattern)**

где

*value* – численное выражение;

*pattern* – шаблонная строка, задающая формат.

#### Величина, полученная в результате:

Строка. Величина типа String.

#### Описание:

Функция **Format\$( )** преобразует число *value* в строковую величину, по шаблону, заданному параметром *pattern*. Например, число **12345.67** функция может преобразовать в строку **“\$12,345.67”**. Параметр *pattern* представляет собой строковый шаблон, по которому преобразуется число *value*. Он должен состоять из специальных символов, управляющих отображением числа в строке, таких как **#**, **0**, **%**, запятая, точка, точка с запятой. Шаблон может также иметь косметические символы, такие как **\$**, **-**, **(**, **)**. Роль каждого из этих символов в формировании результирующей строки приведена в следующей таблице:

Символ в <i>pattern</i>	Какую роль играет для результата функции
<b>#</b>	Соответствует всем цифрам из <i>value</i> до десятичной точки и одной цифре из <i>value</i> после десятичной точки. Если шаблон содержит один или более символов <b>#</b> левее десятичного знака, а форматируемое число меньше единицы, но больше нуля, то в результате ноль в целых частях не будет отображен, строка будет начинаться с точки.
<b>0</b>	Применяется левее десятичной точки. Если управляющая строка содержит один или более символов <b>"0"</b> левее десятичной точки и значение числа меньше единицы и больше нуля, то результирующая строка будет иметь ноль перед десятичной точкой.
<b>.</b> (точка)	Соответствует десятичной точке. Число символов <b>#</b> правее точки определяет число десятичных разрядов, то есть сколько будет выведено символов после десятичной точки. Результирующая строка будет включать в себя тот десятичный разделитель, который установлен в Вашей системе. Используя оператор <b>Set Format</b> можно добиться назначить в качестве десятичного разделителя точку, даже если в системе принят другой знак.

, (запятая)	Разделитель тысяч. Если управляющая строка содержит запятую перед знаком решетки, определяющей целую часть числа, то в результате каждые три цифры будут отделены запятой. Например, десять миллионов будут показаны так: "10,000,000". Результирующая строка будет включать в себя тот десятичный разделитель, который установлен в Вашей системе. Используя оператор <b>Set Format</b> можно назначить в качестве десятичного разделителя точку, даже если в системе принят другой знак.
%	Процентное представление числа. Если управляющая строка содержит знак процента, то результатом будет строка с числом, умноженным на сто и знаком процента справа. Например, число 0.75 будет преобразовано в строку "75%". Если Вы не намерены преобразовывать число в проценты, но хотите иметь в строке знак %, используйте его в комбинации со знаком \ (обратный слэш).
E+, e+	Представление числа в научном (экспоненциальном) формате. Например, число 1234 будет преобразовано в строку "1.234e+03". Если экспонента положительна, то после символа "e" будет помещен плюс. Если экспонента отрицательна, после символа "e" будет помещен знак минус.
E-, e-	Представление числа в научном (экспоненциальном) формате. От предыдущего отличается тем, что в случае положительной экспоненты знак плюс не отображается.
; (точка с запятой)	Знаком "точка с запятой" Вы разделяете в одной управляющей строке два шаблона для положительного и отрицательного значения. Так, в управляющей строке сначала определяется формат для представления положительных значений числа, потом через точку с запятой задается другой формат для отрицательных значений. Если Вы задаете управляющую строку таким образом, то знак минус уже автоматически не отображается для отрицательных значений. Если необходимо иметь знак минус, включите "-" в формат для отрицательных значений.
\	Разрешает включение текстового символа. Если обратный слэш стоит перед специальным символом (например, перед %), то MapBasic будет рассматривать этот символ в шаблоне как текстовый символ.

#### Ошибки:

В результате выполнения функции Вы можете получить код ошибки ERR-FCN-INVALID-FMT, если неправильно задан параметр *pattern*.

#### Примеры:

Следующие примеры показывают, какие результаты получаются при применении различных строковых шаблонов. Результаты показаны в комментариях. Внимание: в локализованных системах Вы можете получить те же результаты, но немного в другом виде.

```
Format$( 12345, ",#")      ' возвращает "12,345"
Format$(-12345, ",#")      ' возвращает "-12,345"
Format$( 12345, "$#")      ' возвращает "$12345"
Format$(-12345, "$#")      ' возвращает "$-12345"
```

## Функция Format\$( )

---

```
Format$( 12345.678, "$,###")           ' возвращает "$12,345.68"
Format$(-12345.678, "$,###")           ' возвращает "-$12,345.68"
Format$( 12345.678, "$,###;($,###)")    ' возвращает "$12,345.68"
Format$(-12345.678, "$,###;($,###)")    ' возвращает "($12,345.68)"

Format$( 12345.6789, ",###")           ' возвращает "12,345.679"
Format$( 12345.6789, ",#.#")           ' возвращает "12,345.7"
Format$(-12345.6789, "###E+##")        ' возвращает "-1.235e+04"

Format$( 0.054321, "###E+##")          ' возвращает "5.432e-02"

Format$(-12345.6789, "###E-##")        ' возвращает "-1.235e04"

Format$( 0.054321, "###E-##")          ' возвращает "5.432e-02"
Format$( 0.054321, "###%")             ' возвращает "5.43%"
Format$( 0.054321, "##\%")             ' возвращает ".05%"
Format$( 0.054321, "0.##\%")           ' возвращает "0.05%"
```

Смотрите также:

[Str\\$\( \)](#)

## Функция **FormatDate\$**

### Назначение:

Возвращает дату в укороченном формате, в том стиле, который определен Панелью управления.

### Синтаксис:

**FormatDate\$( value )**

*value* - это число или строка, представляющие дату в формате YYYYMMDD.

### Возвращаемое значение:

Строковая величина.

### Описание:

Функция **FormatDate\$( )** возвращает строковую величину, представляющую дату с локальным системном формате, как определено в Панели управления.

### Примеры:

Предположим, что настройки Панели управления это d/m/y для даты, '-' это разделитель для даты и "dd-МММ-уууу" - это укороченный формат даты:

```
Dim d_Today As Date
d_Today = CurDate()
Print d_Today          'returns "19970910"
Print FormatDate$(d_Today) 'returns "10-Sep-1997"
Dim s_EnteredDate As String
s_EnteredDate = "03-02-61"
Print FormatDate$(s_EnteredDate)      'returns "03-Feb-1961"
s_EnteredDate = "12-31-61"
Print FormatDate$( s_EnteredDate ) ' returns ERROR: not d/m/y ordering
s_EnteredDate = "31-12-61"
Print FormatDate$( s_EnteredDate ) ' returns "31-Dec-1961"
```

### Функция `FormatNumber$( )`

#### Назначение:

Форматирование числа с использованием символов десятичной точки и разделителя тысяч, тех, которых в данное время использует система, в которой выполняется приложение.

#### Синтаксис:

**`FormatNumber$(num )`**

где *num* – численная или строковая величина, представляющее число, например, “1234.56”

#### Величина, полученная в результате:

Строка. Величина типа String.

#### Описание:

Функция возвращает отформатированную строку с числом. Если число достаточно велико, то функция вставит разделители тысяч. MapInfo прочтет конфигурацию пользовательской системы и определит какие символы используются для десятичной точки и разделителя тысяч.

#### Примеры:

Следующая таблица демонстрирует, как работает функция **`FormatNumber$( )`** в компьютере, в котором точка обозначает десятичную точку, а запятая – разделитель тысяч (стандартная установка в США):

Вызов функции	Результат
<code>FormatNumber\$("12345.67")</code>	“12,345.67” (добавлен разделитель тысяч)
<code>FormatNumber\$("12,345.67")</code>	“12,345.67” (без изменений)

Если в компьютере знак запятой зарезервирован за десятичной точкой, а точка разделяет тысячи, то функция будет работать так:

Вызов функции	Результат
<code>FormatNumber\$("12345.67")</code>	“12.345,67” (добавлен разделитель тысяч и изменен знак десятичной точки)
<code>FormatNumber\$("12,345.67")</code>	“12.345,67” (изменены оба знака)

#### Смотрите также:

**`DeformatNumber$( )`**



### Функция FrontWindow( )

**Назначение:**

Возвращает идентификатор активного окна.

**Синтаксис:**

**FrontWindow( )**

**Величина, полученная в результате:**

Целое число. Величина типа Integer.

**Описание:**

Функция **FrontWindow( )** возвращает целочисленный идентификатор самого верхнего открытого окна в MapInfo. Этот оператор полезно применять сразу после создания нового окна (Карты, Графика, Списка или Отчета), чтобы запомнить значение идентификатора самого верхнего окна.

**Пример:**

```
Dim map_win_id As Integer
Open Table "states.tab"
Map From states
map_win_id = FrontWindow( )
```

**Смотрите также:**

**NumWindows( ), WindowID( ), WindowInfo( )**

### Оператор Function...End Function

#### Назначение:

Объявляет внутреннюю функцию.

#### Предупреждение:

Оператор **Function... End Function** не может быть выполнен в окне MapBasic.

#### Синтаксис:

```
Function name ( [ [ ByVal ] parameter As datatype ]  
                [ , [ ByVal ] parameter As datatype ... ] ) As return_type  
  
    statement_list  
End Function
```

где

*name* – имя функции;

*parameter* – имя параметра функции;

*datatype* – тип данных, стандартный в MapInfo, или тип, созданный при помощи оператора **Type**;

*return\_type* – скалярный тип данных, стандартный в MapInfo; массив или тип, созданный при помощи оператора **Type** не используется;

*statement\_list* – группа операторов, составляющая тело функции.

#### Описание:

Оператор **Function** позволяет Вам создавать свои функции. Сделанные таким образом функции являются внутренними и могут использоваться в программе, в которой находятся наравне со стандартными функциями MapInfo.

Каждое имя функции, созданной при помощи оператора **Function**, а также типы параметров и результата должны быть заранее объявлены в операторе **Declare Function**.

Внутренняя функция подобна процедуре, сделанной при помощи оператора **Sub**, но в отличие от процедуры функция возвращает значение, тип которого определяется параметром *datatype*, расположенным за последним ключевым словом **As**. Функция по сравнению с процедурой в применении более гибка, т.к. Вы можете составлять выражения, содержащие вызов к одной и более функциям. Например, следующее выражение содержит два вызова функции **Proper\$()**:

```
fullname = Proper$(firstname) + " " + Proper$(lastname)
```

В конструкции оператора **Function... End Function** параметр *name*, определяющий имя функции, трактуется как имя переменной. Этой переменной и присваивается значение, возвращаемое функцией. Если переменной *name* не было ничего присвоено, функция возвращает нулевое значение для числовых функций, FALSE для логических функций или нулевую строку ("") для строчных функций.

#### Ограничения при передаче параметров

Результат функции может быть только скалярным. Другими словами, функция не может возвращать массивы и данные сложных типов, созданных оператором **Type**.

По умолчанию, каждый параметр во внутреннюю функцию передается "ссылкой" ("by-reference"). Это означает, что оператор, вызывающий функцию, должен в качестве параметров использовать имена переменных. Если функция изменяет значение параметра, передаваемого ссылкой, то изменяется и переменная в вызывающей процедуре.

Если перед именем параметра используется слово **ByVal**, то параметр передается значением ("by-value"). В этом случае оператор, вызывающий функцию, может использовать в качестве параметра выражение, а не имя переменной. Однако, если изменить значение такого параметра, новое значение нельзя будет вернуть в вызывающий модуль.

Вы не можете передавать в виде значения во внутреннюю функцию массивы, а также переменные, заданные оператором **Type** или объявленные предложением **Alias**. Такие переменные можно передавать только ссылкой.

Если функция не должна иметь параметров, то оператор **Function... End Function** может включать пустые скобки или не использовать их. То есть конструкция:

```
Function Foo( )  
'  
' ... здесь список операторов функции  
'  
End Function
```

идентична конструкции:

```
Function Foo  
'  
' ... здесь список операторов функции  
'  
End Function
```

Но при вызове этой функции использование скобок обязательно:

```
var_name = Foo( )
```

### Доступность функций пользователя

Пользователь не может помещать вызовы к внутренним функциям выполняющейся прикладной программы при заполнении стандартных диалогов в MapInfo. Однако сама программа может это делать. Так, пользователь не может обратиться к определенной программой функции из диалога команды SQL-ЗАПРОС, однако, в скомпилированной MapBasic-программе может содержаться оператор **Select**, использующий эту функцию.

Внутреннюю функцию можно вызывать только в той программе, в которой она была определена. Поэтому, если Вы хотите использовать такую функцию в другой программе, то Вам придется полностью скопировать определение **Function... End Function** в текст другой программы.

### Имена функций

Имя функции, определенное оператором **Function**, может совпадать с именем стандартной функции MapBasic, таким как **Abs** или **Chr\$**. Подобная функция *полностью замещает* одноименную стандартную в пределах прикладной программы, в которой она определена. Например, если Вы создадите свою функцию **Abs**, то все вызовы функции **Abs** будут обращаться к Вашей функции, а не к стандартной функции MapBasic **Abs( )**.

Если произошло описанное переопределение, то на другие программы оно не действует. Поэтому, если Вы хотите переопределить какую-либо функцию для нескольких программ, то Вам придется переопределить ее в каждой программе.

## Оператор Function...End Function

---

### Пример:

В этом примере определяется функция пользователя под названием CubeRoot, которая вычисляет кубический корень из числа. Так как функция CubeRoot вызывается в тексте программы до определения самой функции, то в этом примере используется оператор **Declare Function**, определяющий аргумент функции и тип возвращаемого значения.

```
Declare Function CubeRoot(ByVal x As Float) As Float
Declare Sub Main

Sub Main
    Dim result As Float
    result = CubeRoot(23)
    Note Str$(result)
End Sub

Function CubeRoot(ByVal x As Float) As Float
    CubeRoot = x ^ 0.33333333333
End Function
```

### Смотрите также:

**Declare Function, Declare Sub, Sub... End Sub**

## Оператор Get

### Назначение:

Читает данные из файла, открытого в бинарном режиме (**BINARY**) или в режиме произвольного доступа (**RANDOM**).

### Синтаксис:

**Get** [#] *filenum*, [ *position* ], *var\_name*

где

*filenum* – номер файла, открытого оператором **Open File**;

*position* – позиция, с которой начинается чтение из файла;

*var\_name* – имя переменной.

### Описание:

Оператор **Get** читает данные из файла, открытого оператором **Open File**, и помещает их в переменную *var\_name*. Способ чтения оператором **Get** определяется режимом, в котором файл был открыт оператором **Open File**.

Если файл открыт в режиме произвольного доступа, то параметр *position* определяет строку данных, с которой начинается чтение. Сразу после открытия файла чтение можно производить из первой строки. По мере чтения оператором **Get** позиция считывания автоматически наращивается, и нет нужды каждый раз менять значение **Position**. Однако, если Вы хотите перейти не к следующей по порядку строке, то Вам придется явно задать значение **Position**.

Если файл открыт в режиме бинарного доступа, то одним оператором **Get** читается одно значение *var\_name*. То, как читаются данные, зависит от порядка задания байтов в исходном файле и типа переменной *var\_name*. Если переменная имеет тип Integer, то прочитываются 4 байта из двоичного файла и превращаются в переменную MapBasic. Переменные заполняются при чтении по следующим правилам:

Тип	Данные
Logical	Однобайтовое значение, или 0, или другое ненулевое число.
SmallInt	Двубайтовое значение, целое число.
Integer	Четырехбайтовое значение, целое число.
Float	Восьмибайтовое число в формате IEEE.
String	Длина строки плюс один байт для нулевого значения, обозначающего конец строки.
Date	4 байта: SmallInt для года, один байт для месяца и один байт для дня.
Другие типы	Чтение не производится.

## Оператор Get

---

В режиме чтения двоичных кодов параметр **Position** используется для задания позиции считывания на определенное значение смещения в файле. Сразу после открытия файла значение позиции устанавливается на единицу (начальный байт файла). По мере выполнения оператора **Get** значение смещения автоматически увеличивается на общую длину считанной информации. Если Вы не задаете значение **Position**, каждый новый оператор **Get** продолжает чтение с того места, на котором оно остановилось в прошлый раз.

В операторе **Get** нужно поставить две запятые, если параметр *position* опущен.

Если файл был открыт в режиме **BINARY**, оператор **Get** не может заполнять строку (переменную типа String) неопределенной длины; любая переменная типа String в операторе **Get** должна иметь определенную длину.

### Смотрите также:

**Open File, Put**

## Функция GetFolderPath\$( )

### Назначение

Возвращает путь к специальным папкам MapInfo или Windows.

### Синтаксис

**GetFolderPath\$( *folder\_id* )**

*folder\_id* индекс папки, может принимать одно из следующих значений:

FOLDER\_MI\_APPDATA  
FOLDER\_MI\_LOCAL\_APPDATA  
FOLDER\_MI\_PREFERENCE  
FOLDER\_MI\_COMMON\_APPDATA  
FOLDER\_APPDATA  
FOLDER\_LOCAL\_APPDATA  
FOLDER\_COMMON\_APPDATA  
FOLDER\_COMMON\_DOCS  
FOLDER\_MYDOCS  
FOLDER\_MYPICS

### Возвращаемое значение

Строка

### Описание

Получив индекс одной из определенных папок MapInfo или Windows, функция GetFolderPath\$( ) возвращает путь к этой папке. Примером специальной папки Windows является папка My Documents. Примером специальной папки MapInfo является папка с настройками.

Расположение многих из этих папок варьирует в различных версиях Windows. Оно также может отличаться для отдельных пользователей. Обратите внимание, что FOLDER\_MI\_APPDATA, FOLDER\_MI\_LOCAL\_APPDATA и FOLDER\_MI\_COMMON\_APPDATA могут не существовать. Перед попыткой войти в одну из этих папок, убедитесь в их существовании, используя функцию FileExists(). Папка FOLDER\_MI\_PREFERENCE всегда существует.

Индексы папок в FOLDER\_MI возвращают путь к папкам, указанным в MI Pro. Остальные индексы возвращают путь для папок Windows и соответствуют индексам, определенным для функции WIN32 API, а именно SHGetFolderPath. Самые общие из этих индексов определены для удобного использования в приложении MapBasic. Любые индексы id возможные для SHGetFolderPath будут работать и с функцией GetFolderPath\$().

### Пример

```
include "mapbasic.def"
declare sub main
sub main
dim sMiPrfFile as string
sMiPrfFile = GetFolderPath$(FOLDER_MI_PREFERENCE)
Print sMiPrfFile
end sub
```

## Функция GetFolderPath\$ ( )

---

### Смотрите также

LocateFiles( )



## Функция GetMetadata\$( )

### Назначение:

Извлекает из таблицы метаданные.

### Синтаксис:

**GetMetadata\$( table\_name, key\_name )**

где

*table\_name* – имя открытой таблицы, заданное как недвусмысленное имя (например, World) или как строка (например, "World");

*key\_name* – строковая величина, представляющая собой имя ключа метаданных.

### Величина, полученная в результате:

Величина типа String, строка длиной до 239 байт. Если ключ не существует или соответствующее значение пусто, то MapInfo вернет пустую строку.

### Описание:

Эта функция возвращает метаданные из таблицы. Более подробное описание процесса извлечения метаданных из таблицы см. в описании оператора **Metadata**, а также в главе 7 *Руководства пользователя MapBasic*.

### Пример:

Если в таблице PARCELS есть ключ метаданных "\Copyright", то следующий оператор распечатывает соответствующее значение метаданных:

```
Print   GetMetadata$(Parcels, "\Copyright")
```

### Смотрите также:

**Metadata**

### Функция GetSeamlessSheet( )

#### Назначение:

Выдает диалоговый запрос, в котором пользователь должен выбрать одну из таблиц-компонент сшитой таблицы и возвращает имя выбранной таблицы.

#### Синтаксис:

**GetSeamlessSheet( *table\_name* )**

где *table\_name* – имя открытой сшитой таблицы.

#### Величина, полученная в результате:

Строка. Величина типа String, представляющая собой имя таблицы (или пустую строку при отмене выбора пользователем).

#### Описание:

Эта функция показывает диалоговое окно со списком таблиц, составляющих сшитую таблицу. Если пользователь выберет таблицу и нажмет на кнопку “ОК”, то функция вернет имя выбранной таблицы. Если пользователь отменит диалог, то результатом будет пустая строка.

#### Пример:

```
Sub Browse_A_Table(ByVal s_tab_name As String)
    Dim s_sheet As String

    If TableInfo(s_tab_name, TAB_INFO_SEAMLESS) Then
        s_sheet = GetSeamlessSheet(s_tab_name)
        If s_sheet <> "" Then
            Browse * From s_sheet
        End If
    Else
        Browse * from s_tab_name
    End If

End Sub
```

#### Смотрите также:

**Set Table, TableInfo( )**

## Оператор Global

### Назначение:

Объявляет имена и типы глобальных переменных.

### Синтаксис:

```
Global var_name [, var_name ... ] As var_type
                        [, var_name ... As var_type ... ]
```

где

*var\_name* – имя глобальной переменной;

*var\_type* – тип данных: Integer, Float, Date, Logical, String или тип, созданный при помощи оператора **Type**.

### Описание:

Оператор **Global** объявляет одну или более глобальных переменных. Оператор употребляется вне текста процедур и функций.

Синтаксис оператора **Global** такой же, как в операторе **Dim**. Отличает операторы смысл переменных: в операторе **Dim** объявляются локальные переменные, а в **Global** – глобальные переменные.

Локальные переменные могут использоваться только в процедурах, в которых они объявлены.

Значения глобальных переменных могут использоваться и изменяться во всех процедурах программы.

В процедурах могут быть объявлены локальные переменные с именами, совпадающими с именами глобальных переменных. В этом случае, в процедуре под этим именем используется только локальная переменная, а значения глобальной переменной с таким же именем из этой процедуры не доступно.

Размерность массива глобальных переменных может быть изменена при помощи оператора **ReDim**.

В Windows значения глобальных переменных выполняющейся программы доступны другим программам, поддерживающим DDE-связь.

### Пример:

```
Declare Sub testing( )
Declare Sub Main( )
Global global_var As Integer
Sub Main( )
    Call testing
    Note Str$(global_var) ' в окне будет число "23"
End Sub
Sub testing( )
    global_var = 23
End Sub
```

### Смотрите также:

**Dim**, **ReDim**, **Type**, **UBound( )**

### Оператор Goto

#### Назначение:

Передаёт управление программой оператору с меткой.

#### Предупреждение:

Оператор **Goto** не может выполняться в окне MapInfo.

#### Синтаксис:

**Goto** *label*

где

*label* – имя метки.

#### Описание:

Оператор **Goto** определяет безусловный переход выполнения программы. Выполнение продолжается с оператора, отмеченного меткой *label*. Метка в тексте программы представляет собой произвольное слово перед оператором, отделенное от оператора двоеточием и пробелом. В операторе **Goto** метка *label* пишется без двоеточия.

Оператор **Goto** не должен использоваться для выхода из цикла. Для этого используйте операторы **Exit Do** и **Exit For**.

Оператор **Goto** осуществляет переход только в пределах одной процедуры.

#### Пример:

```
Goto endproc
```

```
...
```

```
endproc: End Program
```

#### Смотрите также:

**Do, For, OnError, Resume**

## Оператор Graph

### Назначение:

Открывает новое окно Графика.

### Синтаксис (версии 5.5):

```
Graph
  label_column , expr [ , ... ]
From table
  [ Position (x , y) [ Units paperunits ] ]
  [ Width width [ Units paperunits ] ]
  [ Height height [ Units paperunits ] ]
  [ Min | Max ]
  [ Using template_file [ Restore ] [ Series In Columns ] ]
```

где

*label\_column* – имя колонки, используемой для подписывания оси Y;

*expr* – выражение;

*table* – имя открытой таблицы;

*paperunits* – строковая величина, задающая единицу измерения листа (например, "mm");

*x* , *y* – координаты позиции верхнего левого угла окна Графика в "бумажных" единицах измерения;

*width* и *height* – определяют размер окна Графика в "бумажных" единицах измерения.

*template file* - это файл шаблона графика

### Синтаксис (версии до 5.5)

```
Graph
  label_column , expr [ , ... ]
From table
  [ Position (x , y) [ Units paperunits ] ]
  [ Width width [ Units paperunits ] ]
  [ Height height [ Units paperunits ] ]
  [ Min | Max ]
```

*label\_column* – имя колонки, используемой для подписывания оси Y;

*expr* – выражение;

*table* – имя открытой таблицы;

*paperunits* – строковая величина, задающая единицу измерения листа (например, "mm");

*x* , *y* – координаты позиции верхнего левого угла окна Графика в "бумажных" единицах измерения;

*window\_width* и *window\_height* – определяют размер окна Графика в "бумажных" единицах измерения;

### Описание:

Если предложение **Using** присутствует и *template\_file* указывает файл шаблона, то график версии 5.5 будет создан на основе указанного шаблона. В противном случае будет создан график версии 5.0.

Если включено предложение **Restore**, то текст заголовка из файла шаблона будет использован для окна графика.

В противном случае для заголовка графика будет использован стандартный текст. Ключевое слово **Restore** будет использовано когда команда **Graph** записывается в рабочий набор, при открытии такого рабочего набора текст заголовка будет восстановлен в точности таким, каким он был при

сохранении рабочего набора. Ключевое слово **Restore** не используется в **Graph** команде, создаваемой с помощью Мастера графиков, и для каждого заголовка будет использован стандартный текст. Если включается **Series In Columns**, то серии графиков будут организовываться из колонок таблицы. В противном случае, серии будут образовываться из строк таблицы.

Команды Graph в рабочих наборах или программах, которые созданы в версиях ранее 5.5 будут создавать окно графика версии 5.0. Когда окно графика 5.0 активно в версии MapInfo 5.5, то меню графика версии 5.0 так же будет активным, и пользователь может настраивать график используя диалоги версии 5.0. Мастер графиков будет всегда создавать окно графиков версии 5.5.

Оператор **Graph** создает новое окно Графика и показывает в нем данные, определенные в таблице *table*. Вид графика в окне будет столбцовый повернутый, если Вы заранее не определили другой вид оператором **Set Graph**. Выполнив оператор **Set Graph** после оператора **Graph** Вы можете также изменить заголовок графика, вид осей, подписей и т. п.).

Команда ОКНО > ГРАФИК в MapInfo вызывает диалог, в котором пользователь выбирает имена полей, значения которых будут отображены в графике. Оператор **Graph** в MapBasic, кроме этого, позволяет задавать выражения с именами полей для построения графика. В диалоге команды ГРАФИК пользователь может задать только четыре колонки, тогда как оператор может построить график из 255 колонок (если этот график имеет смысл).

Предложение **Position** задает расположение окна на экране. Координаты *x* и *y* определяют верхний левый угол окна Графика относительно верхнего левого угла окна MapInfo.

Предложения **Width** и **Height** определяют ширину и высоту окна Графика. Параметры *x*, *y*, *window\_width*, *window\_height* задаются в единицах, определенных в предложении **Units**. Если оно в каком-либо из трех вышеупомянутых предложений опущено, соответствующие параметры будут задаваться в "бумажных" единицах, определенных в Вашей программе (смотрите оператор **Set Paper Units**).

Ключевые слова **Max** и **Min** используются для открытия окна Графика на все рабочее окно MapInfo или для создания окна свернутым в иконку. Последнее действие не поддерживается в системе Macintosh.

### Пример (графики версии 5.5):

```
Graph State_Name, Pop_1980, Pop_1990, Num_Hh_80 From States Using  
»C:\Program Files\MapInfo\GRAPHSSUPPORT\Templates\Column\Percent.3tf»  
Graph City, Tot_hu, Tot_pop From City_125 Using »C:\Program Files\Map-  
Info\GRAPHSSUPPORT\Templates\Bar\Clustered.3tf» Series In Columns
```

### Пример (графики версии до 5.5):

```
Graph Country, Population From Selection
```

### Смотрите также:

**Set Graph**

## Функция HomeDirectory\$( )

### Назначение:

Возвращает строкой личный каталог пользователя.

### Синтаксис:

**HomeDirectory\$( )**

### Величина, полученная в результате:

Строка. Величина типа String.

### Описание:

Функция **HomeDirectory\$( )** возвращает строку, представляющую личный каталог пользователя. Значение, которое будет возвращено функцией, определяется системной платформой.

Система	Результат
Windows	Каталог, содержащий стартовые файлы Windows (например, WIN.INI). При работе в сети каждый пользователь может иметь свой личный Windows-каталог, что позволяет задавать свою личную конфигурацию.
Macintosh	Расположение папки System. Замечание: Вы можете избежать хранения файлов, относящихся к MapBasic-программе файлов непосредственно в папке System. Если Вам необходимо расположить файлы конфигурации в легкодоступном месте, то можно использовать папку Preferences, которая находится внутри папки System.

### Пример:

```
Dim s-home-dir As String
s-home-dir = HomeDirectory$( )
```

### Смотрите также:

**ApplicationDirectory\$( )**, **ProgramDirectory\$( )**, **SystemInfo( )**

### Оператор If...Then

#### Назначение:

Условное выполнение той или иной группы операторов.

#### Синтаксис:

```
If if_condition Then  
    if_statement_list  
[ ElseIf elseif_condition Then  
    elseif_statement_list ]  
[ ElseIf... ]  
[ Else  
    else_statement_list ]  
End If
```

где

*condition* – выражение, результат которого есть логическая величина (TRUE или FALSE);

*statement\_list* – группа операторов, количество которых может быть нулевым.

#### Предупреждение:

Вы не можете использовать оператор **If... Then** в окне MapBasic.

#### Описание:

Оператор **If... Then** позволяет выбрать, какую группу операторов выполнить при удовлетворении определенных условий, задаваемых выражениями *if\_condition* и *elseif\_condition*.

Возможна простая форма оператора **If**, оператор без предложений **ElseIf** и **Else**...

```
If if_condition Then  
    if_statement_list  
End If
```

В этом случае вычисляется значение выражения *if\_condition*. Если выражение равно логическому значению TRUE, то MapBasic выполнит операторы *if\_statement\_list*. Иначе MapBasic пропустит группу операторов *statement\_list* и передаст управление программой следующему оператору после **End If**.

Второй вариант формы **If** включает конструкцию **Else**:

```
If if_condition Then  
    if_statement_list  
Else  
    else_statement_list  
End If
```

Здесь вычисляется значение выражения *if\_condition*. Если выражение равно логическому значению TRUE, то MapBasic выполнит операторы *if\_statement\_list* и передаст управление программой следующему оператору после **End If**. Иначе MapBasic пропустит группу операторов *statement\_list* и выполнит *else\_statement\_list*.

Третий вариант формы **If** включает предложение **ElseIf**, и потом предложение **Else** (хотя это не обязательно):

```
If if_condition Then  
    if_statement_list  
ElseIf elseif_condition Then  
    elseif_statement_list
```



**Else**

*else\_statement\_list*

**End If**

В этом случае также сначала вычисляется значение выражения *if\_condition*. Если выражение равно логическому значению TRUE, то MapBasic выполнит операторы *if\_statement\_list* и передаст управление программой следующему оператору после **End If**. Иначе MapBasic будет проверять выражения *elseif\_condition* и, если оно истинно, выполняются операторы *elseif\_statement\_list* и управление программой передается следующему оператору после **End If**. Если выражение *elseif\_condition* ложно, будут выполнены операторы, определенные конструкцией **Else**.

Оператор **If** может содержать несколько конструкций **ElseIf**, позволяющих Вам рассматривать любое количество возможных состояний. Но если Вы хотите, чтобы программа была написана в хорошем стиле, используйте оператор **Do Case**, а не оператор **If** с несколькими предложениями **ElseIf**....

### Пример:

Проверим, не является ли сегодняшний день праздничным (Новый год, Рождество или Татьянин день), и выведем соответствующее поздравление. Если день не является праздником, будет выведено сообщение "Добрый день."

```
Dim today As Date
Dim today-mon, today-day, yearcount As Integer
today = CurDate( )      ' чтение текущей даты
today-mon = Month(today) ' чтение текущего месяца
today-day = Day(today)   ' чтение текущего числа (1-31)
If today-mon = 1 And today-day = 1 Then
    Note "С Новым Годом!"
    yearcount = yearcount + 1
ElseIf today-mon = 1 And today-day = 7 Then
    Note "С Рождеством!"
ElseIf today-mon = 1 And today-day = 25 Then
    Note "С днем ангела, Танечка!"
Else
    Note "Добрый день."
End If
```

### Смотрите также:

**Do Case**

### Оператор Import

#### Назначение:

Создает новую таблицу MapInfo Professional при импорте файла, такого как GML или DXF.

#### Синтаксис 1 (для файлов MIF/MID, PICT или MapInfo for DOS)

```
Import file_name  
    [ Type file_type ]  
    [ Into table_name ]  
    [ Overwrite ]
```

#### Синтаксис 2 (для файлов DXF)

```
Import file_name  
    [ Type "DXF" ]  
    [ Into table_name ]  
    [ Overwrite ]  
    [ Warnings { On | Off } ]  
    [ Preserve  
        [ AttributeData ] [ Preserve ] [ Blocks As MultiPolygonRgns ] ]  
    [ CoordSys . . . ]  
    [ Autoflip ]  
    [ Transform  
        ( DXF_x1 , DXF_y1 ) ( DXF_x2 , DXF_y2 )  
        ( MI_x1 , MI_y1 ) ( MI_x2 , MI_y2 ) ]  
    [ Read  
        [ Integer As Decimal ] [ Read ] [ Float As Decimal ] ]  
    [ Store [ Handles ] [ Elevation ] [ VisibleOnly ] ]  
    [ Layer DXF_layer_name  
        [ Into table_name ]  
        [ Preserve  
            [ AttributeData ] [ Preserve ] [ Blocks As MultiPolygonRgns ] ]  
    ]  
    [ Layer . . . ]
```

#### Синтаксис 3 (для файлов GML)

```
Import file_name  
    Type file_type  
    Layer layer_name  
    [ Into table_name ]  
    [ Overwrite ]
```

*file\_name* строка определяющая имя импортируемого файла

*file\_type* строка определяющая формат импортируемого файла (MIF, MBI, MMI, IMG, GML или PICT)

*table\_name* определяет имя новой создаваемой таблицы

DXF\_x1, DXF\_y1, числа, представляющие координаты в DXF файле

MI\_x1, MI\_y1, числа, представляющие координаты в таблице MapInfo

DXF\_layer\_name строка, имя слоя в DXF файле

Layer layername строка, имя слоя в GML файле.

### Описание

Оператор Import создает новую таблицу MapInfo при импорте содержимого существующего файла. Обратите внимание: Для создания таблицы MapInfo, основанной на списке или файле базы данных, используйте оператор Register Table, а не оператор Import.

Предложение Into позволяет переписать имя и местоположение создаваемой таблицы MapInfo. Если предложение Into не определено, то новая таблица создается в той же директории, что и исходный файл, с соответствующим именем. Например, если импортируется текстовый файл "WORLD.MIF", то новая таблица по умолчанию будет иметь имя "WORLD.TAB".

Дополнительное предложение Type определяет формат файла при импорте. Предложение Type может иметь один из следующих вариантов:

#### Предложение Type    Формат файлов

Type "DXF"	Файл DXF (a format supported by CAD packages, such as AutoCAD).
Type "MIF"	Файлы MIF / MID, создаваемые при экспорте таблиц MapInfo.
Type "PICT"	Файл PICT; поддерживается только в Macintosh.
Type "MBI"	Файл MapInfo Boundary Interchange, созданный в MapInfo для DOS.
Type "MMI"	Файл MapInfo Map Interchange, созданный в MapInfo для DOS.
Type "IMG"	Файл MapInfo Image, созданный в MapInfo для DOS.

Если пропущено предложение Type, MI Pro предполагает, что формат файла определяется расширением. Например, файл с именем "PARCELS.DXF" будет считаться файлом DXF.

Если Вы включаете дополнительное ключевое слово Overwrite, MI Pro создаст новую таблицу, независимо от того, существует или нет таблица с таким именем; новая таблица заменит существующую. Если ключевое слово Overwrite пропущено, а таблица с таким именем уже существует, то MI Pro не будет переписывать таблицу.

### Настройки импорта для файлов DXF

Если Вы импортируете файл DXF, то оператор Import может включать следующие предложения, определяющие настройки DXF.

Внимание: Порядок предложений крайне важен; помещение предложений в неправильном порядке приведет к ошибкам компиляции.

#### Warnings On или Warnings Off

Контролирует, появляются ли предупреждающие сообщения во время импорта. По умолчанию установлено warnings off.

#### Preserve AttributeData

Включите это предложение, если надо, чтобы MI Pro сохраняла атрибутивные данные из DXF.

### **Preserve Blocks As MultiPolygonRgns**

Включите это предложение, если надо, чтобы MI Pro сохраняла все полигоны полигоны из блока записи DXF в один объект-регион со множеством полигонов. Если это предложение пропущено, каждый полигон DXF становится отдельным объектом-регионом MI Pro.

### **CoordSys**

Контролирует проекцию и систему координат таблицы. Подробнее смотрите описание предложения CoordSys.

### **Autoflip**

Включите эту команду, если надо повернуть x-координаты карты относительно центральной линии карты. Команда применима только к несферическим координатам.

### **Transform**

Определяет трансформацию координат. В предложении Transform задается минимум и максимум x- и y-координат импортируемого файла, и задается минимум и максимум тех координат, которые Вам понадобятся в таблице MapInfo.

### **Read Integer As Decimal**

Используйте это предложение, если надо хранить целочисленные значения из файла DXF в десятичной колонке новой таблицы. Это предложение можно использовать только при применении предложения Preserve AttributeData.

### **Read Float As Decimal**

Используйте это предложение, если надо хранить вещественные значения из файла DXF в десятичной колонке новой таблицы. Это предложение можно использовать только при применении предложения Preserve AttributeData.

### **Store [ Handles ] [ Elevation ] [ VisibleOnly ]**

Если определено предложение Handles, то таблица MapInfo хранит уникальные ID номера при рисовке в колонке, с именем \_DXFHandle. Если определено предложение Elevation, то MI Pro хранит высоту центра каждого объекта в колонке с именем \_DXFElevation. (Для линий, MI Pro хранит высоты центров линий; для регионов, MI Pro хранит среднее значение высоты объекта.) Если включено предложение VisibleOnly, MI Pro игнорирует невидимые объекты.

### **Layer . . .**

Если нет предложений Layer, все объекты из файла DXF импортируются в одну таблицу MapInfo. Если используется одно или более предложений Layer, каждый слой DXF, который Вы перечислите, станет отдельной таблицей MapInfo.

Если файл DXF содержит много слоев, и Ваш оператор Import содержит одно или несколько предложений Layer, то MI Pro будет импортировать слои в соответствии с Вашими именами таблиц. Например, пусть файл DXF содержит четыре слоя (под номерами 0, 1, 2 и 3).

Следующий оператор Import будет импортировать все четыре слоя в одну таблицу MapInfo:

```
Import "FLOORS.DXF"  
Into "FLOORS.TAB"  
Preserve AttributeData
```

Следующий оператор будет импортировать слои 1 и 3, но не будет импортировать слои 0 или 2:

```
Import "FLOORS.DXF"  
Layer "1"  
Into "FLOOR_1.TAB"  
Preserve AttributeData
```

```
Layer "3"  
  Into "FLOOR_3.TAB"  
  Preserve AttributeData
```

### **Импорт GML файлов**

MapInfo Professional 7.0 поддерживает импорт OSGB (Ordnance Survey of Great Britain) GML файлов. Поддерживаются Cartographic Symbol, Topographic Point, Topographic Line, Topographic Area и Boundary Line; Cartographic Text не поддерживается. Topographic Area может иметься в двух формах; MI Pro поддерживает нетопологическую форму. Если файл содержит XLINKS, MI Pro импортирует только атрибутивные данные и не импортирует пространственные объекты. Эти XLINKы хранятся в файле GML как "xlink:href=". Если топологические объекты включены в файл, то появится предупреждающее сообщение о том, что пространственные объекты не могут быть импортированы. Откройте окно Списка, чтобы посмотреть атрибутивные данные.

### **Пример**

Следующий пример импортирует MIF (MapInfo Interchange Format) файл:

```
Import "WORLD.MIF"  
  Type "MIF"  
  Into "world_2.tab"
```

```
Map From world_2
```

Следующий пример импортирует GML файл:

```
Import sGMLDataPath + sGMLFileName + ".gml"  
  Type "GML"  
  Layer sGMLLayerName  
  Into sTabDataPath + sTabFileName + ".TAB"
```

### **Смотрите также**

**Export**

## Оператор Include

### Назначение:

Объединяет содержимое отдельного текстового файла с текстом программы MapBasic.

### Синтаксис:

**Include** "*filename*"

где

*filename* – имя текстового файла.

### Предупреждение:

Вы не можете использовать оператор **Include** в окне MapBasic.

### Описание:

Когда MapBasic компилирует программный файл с оператором **Include**, текст определенного в этом операторе файла рассматривается как часть текста программы. Файл, заданный параметром *filename*, должен состоять из допустимых операторов MapBasic.

Если имя файла в параметре *filename* задано без каталога и этот файл не находится в активном каталоге, то компилятор языка MapBasic делает попытку найти его в том каталоге, в котором расположен программный файл MapBasic. Это позволяет Вам не копировать каждый раз файлы стандартных определений (такие, как MAPBASIC.DEF) в каталог разрабатываемой программы.

Оператор используется для подсоединения к файлам стандартных определений, таких, как MAPBASIC.DEF и MENU.DEF. В них с помощью оператора **Define** целочисленным кодам заданы имена (например, TRUE и FALSE), использование которых в Вашей программе делает ее текст более понятным.

Однако, если Вы изменили содержимое текстового файла, подсоединенного оператором **Include**, то для внесения этих изменений в программу ее необходимо перекомпилировать.

### Пример:

```
Include "MAPBASIC.DEF"
```

## Оператор Input #

### Назначение:

Читает данные из файла и помещает их в переменные.

### Синтаксис:

**Input #***filenum* , *var\_name* [ , *var\_name* ... ]

где

*filenum* – номер файла, под которым он был открыт оператором **Open File**;

*var\_name* – имя переменной.

### Описание:

Оператор **Input #** читает данные из файла, открытого в режиме последовательного доступа (**INPUT**, **OUTPUT** или **APPEND**), и присваивает их как значения одной или более переменным MapBasic.

Оператор **Input #** читает данные последовательно от верхней строки к последней, присваивая каждую порцию одной переменной *var\_name*. Оператор **Input #** считывает порции данных из файла, используя в качестве разделителей запятую и символ конца строки. Поэтому, чтобы прочитать одну строку с запятыми в одну строковую переменную, надо использовать оператор **Line Input #**.

Когда оператор **Input #** читает пустую строку, то значением строковой переменной будет пустая строка (""). Если переменная была объявлена численным типом, то значением будет ноль.

После выполнения оператора **Input #** применение функции **EOF( )** помогает определить корректность результата чтения. Если чтение произошло успешно, результатом функции **EOF( )** будет логическое "нет" (FALSE), и, если был прочитан признак конца файла, **EOF( )** вернет логическое "да" (TRUE).

Пример использования оператора **Input #** Вы можете увидеть в программе NIEWS ("Виды") из стандартной поставки MapBasic.

### Смотрите также:

**EOF( )**, **Line Input #**, **Open File**, **Write #**

## Оператор Insert

### Назначение:

Добавляет новую строку в открытую таблицу

### Синтаксис:

```
Insert Into table [ ( columnlist ) ]  
{ Values ( exprlist ) | Select columnlist From table }
```

где

*table* – имя открытой таблицы;

*columnlist* – список выражений;

*exprlist* – одно выражение или список, разделенный запятыми.

### Описание:

Оператор **Insert** вставляет новую строку в открытую таблицу. Этот оператор применяется в двух формах, позволяющих либо добавлять таблице по одной строке, либо добавлять группы строк из другой таблицы. В каждом случае порядок и количество значений в списке *exprlist*, которые будут вставлены в поля новой строки, должны соответствовать колонкам, перечисленным в списке *columnlist*. Если колонки не перечислены, то будут рассматриваться все поля в новой строке. Если Вы хотите сохранить таблицу с новыми строками на диске, то после ввода используйте оператор **Commit Table**.

Если Вы точно знаете, сколько колонок в таблице, и если Вам уже известно, какие именно величины в какие поля надо поместить, то можно опустить предложение (*columnlist*). Достаточно составить список величин в порядке расположения полей в таблице. В следующем примере результатом оператора будет новая строка в таблице, состоящей из четырех колонок ("Имя", "Адрес", "Город", "Область"), и каждому полю будет определено значение:

```
Insert Into customers  
Values ("Мария Павловна Носова", "ул. Солнечная, 2, 23",  
        "Троицк", "Московская")
```

Результатом этого оператора может быть ошибка, если таблица будет иметь больше или меньше колонок, чем четыре. Если Вы не знаете точно, сколько колонок составляют таблицу, или не знаете, в каком порядке расположены колонки в таблице, необходимо в операторе использовать список имен колонок (*columnlist*).

Следующий оператор создает новую строку со значением в поле "Имя". Для успешного выполнения оператора необязательно знать, сколько всего полей в структуре таблицы.

```
Insert Into customers ( Имя )  
Values ("Степан Иванович Гарин")
```

Следующий оператор создает точечный объект и вставляет его в поле "Obj" новой записи в таблице SITES. Таким образом, мы создаем новую чистую запись с присоединенным точечным объектом.

```
Insert Into sites (Obj)  
Values ( CreatePoint(-73.5, 42.8) )
```

Следующий пример иллюстрирует, как оператором **Insert** можно добавить запись в таблицу, используя запись из другой. Допустим, что в таблице NY-ZIPS содержатся ZIP-коды штата Нью-Йорк и в таблице NJ-ZIPS содержатся ZIP-коды штата Нью-Джерси. Добавим все ZIP-коды из одной таблицы в другую. Такая операция иногда необходима для поиска, так как оператор **Find** может работать только с одной таблицей.



```
Insert Into NY-ZIPS
```

```
  Select * From NJ-ZIPS
```

В следующем примере берутся все объекты из таблицы WORLD и каждый подсоединяется к новой строке таблицы OUTLINE.

```
Open Table "world"
```

```
Open Table "outline"
```

```
Insert Into outline (Obj)
```

```
  Select Obj From World
```

**Смотрите также:**

**Commit, Delete, Rollback**

### Функция InStr( )

#### Назначение:

Возвращает позицию первого символа подстроки в строке.

#### Синтаксис:

**InStr**(*position*, *string*, *substring*)

где

*position* – стартовая позиция для поиска, положительное целое число;

*string* – строковое выражение;

*substring* – строковое выражение.

#### Величина, полученная в результате:

Целое число. Величина типа Integer.

#### Описание:

Функция **InStr( )** проверяет, входит ли в состав строки, представленной выражением *string*, другая строка (или символ), представленная выражением *substring*, и возвращает позицию вхождения. Поиск в первой строке начинается с символа, номер которого задается параметром *position*. Если *position* равен единице, MapBasic начинает поиск с начала строки *string*.

Если строка *substring* не входит в *string*, то результатом функции **InStr( )** будет 0 (ноль).

Если строка *substring* найдена в *string*, то функция **InStr( )** вернет номер символа в строке *string*, который является первым в *substring*.

Если параметр *substring* – пустая строка, то результатом будет 0 (ноль).

Функция **InStr( )** различает строчные и прописные буквы. Другими словами, функция **InStr( )** не найдет подстроку "ИЯ" в строке "Россия" и вернет ноль.

#### Ошибки:

В результате выполнения функции может генерироваться код ошибки

ERR-FCN-ARG-RANGE, если *position* больше, чем количество символов в строке *string*.

#### Пример:

```
Dim fullname As String, pos As Integer
fullname = "Галина Петровна Соколова"
pos = InStr(1, fullname, "Галина")
'
' переменная pos равна 1 (единице)
'
pos = InStr(1, fullname, "Сокол")
'
' теперь pos равна 17 (семнадцати)
'
pos = InStr(1, fullname, "СОКОЛ")
'
' теперь pos равна 0 (нулю);
'
```

#### Смотрите также:

**Mid\$( )**

## Функция Int( )

### Назначение:

Возвращает целую часть действительного числа.

### Синтаксис:

**Int**(*num\_expr*)

где

*num\_expr* – численное выражение.

### Величина, полученная в результате:

Целое число. Величина типа Integer.

### Описание:

Функция отсекает дробную часть от действительного числа, полученного в результате вычисления выражения *num\_expr*, и возвращает целую часть.

Функция **Int( )** похожа на функцию **Fix( )**, но не идентична. Функции различаются способом удаления дробной части отрицательного числа. Когда *num\_expr* представляет отрицательное число, функция **Fix( )** возвращает ближайшее целое, большее или равное оригиналу. Например:

**Fix**(-2.3)

возвращает значение -2. В случае функции **Int( )** результатом будет ближайшее целое, меньше или равное оригиналу. Например:

**Int**(-2.3)

возвращает значение -3.

### Пример:

```
Dim whole As Integer
whole = Int(5.999)
' whole сейчас имеет значение 5.
whole = Int(-7.2)
' whole сейчас имеет значение -8.
```

### Смотрите также:

**Fix( ), Round( )**

### Функция `IntersectNodes( )`

#### Назначение:

Вычисляет точки пересекающихся объектов и возвращает полилинию, имеющую узлы в точках пересечения.

#### Синтаксис:

**`IntersectNodes(object1, object2, points_to_include)`**

где

*object1* и *object2* – объектные выражения, которые могут представлять объекты любого типа, кроме точечного и текстового;

*points\_to\_include* – один из следующих кодов:

INCL\_CROSSINGS – функция возвращает узлы, в которых сегменты обоих объектов пересекаются;

INCL\_COMMON – функция возвращает узлы отрезков, на которые накладываются сегменты обоих объектов;

INCL\_ALL – функция возвращает узлы, в которых сегменты пересекаются и накладываются.

#### Величина, полученная в результате:

Полилиния. Величина типа Object.

#### Описание:

Функция **`IntersectNodes( )`** возвращает объект типа "полилиния", узлы которого лежат в точках пересечения объектов *object1* и *object2*.

## Функция IsPenWidthPixels()

### Назначение:

Функция **IsPenWidthPixels** определяет в каких величинах измеряется ширина линии - в пикселах или в пунктах.

### Синтаксис:

**IsPenWidthPixels ( *penwidth* )**

*penwidth* - это малое целое, определяющее ширину линии.

### Возвращаемое значение:

Истинно, если ширина линии задана в пикселах, ложно, если ширина задана в пунктах.

### Описание:

Функция **IsPenWidthPixels ( )** возвратит истинное значение, если ширина линии задана в пикселах. Ширина линии может быть определена использованием функции **StyleAttr ( )**.

### Пример:

```
Include "MAPBASIC.DEF"
Dim CurPen As Pen
Dim Width As Integer
Dim PointSize As Float
CurPen = CurrentPen()
Width = StyleAttr(CurPen, PEN_WIDTH)
If Not IsPenWidthPixels(Width) Then
    PointSize = PenWidthToPoints(Width)
End If
```

### Смотрите также:

**CurrentPen(), MakePen(), Pen, PenWidthToPoints()**

### Оператор Kill

#### Назначение:

Удаляет файл с диска.

#### Синтаксис:

**Kill** *filespec*

где

*filespec* – строка, представляющая имя файла и, если необходимо, DOS-маршрут.

#### Описание:

Оператор **Kill** удаляет файл с диска. Действие оператора нельзя отменить потом, поэтому рекомендуется применять его с осторожностью.

#### Пример:

```
kill "C:\TEMP\JUNK.TXT"
```

#### Смотрите также:

[Open File](#)

## Функция LabelFindByID( )

### Назначение:

Инициализирует внутренний указатель подписи, так что Вы можете запросить подпись из определенной строки в слое Карты.

### Синтаксис:

**LabelFindByID**(*map\_window\_id* , *layer\_number* , *row\_id* , *table* , *b\_mapper*)

*map\_window\_id* – целочисленный индекс окна id, определяющий окно Карты;

*layer\_number* – номер слоя в текущем окне Карты (например, 1 для верхнего слоя);

*row\_id* – положительное целочисленное значение, указывающее номер строки, в которой запрашивается подпись;

*table* – имя таблицы или пустой строки (""): когда Вы делаете запрос к таблице, входящей в сшитую таблицу, укажите имя такой таблицы, входящей в сшитую; в противном случае, укажите пустую строковую переменную;

*b\_mapper* – логическая величина. "Да" (TRUE) при запросе подписей, которые появляются при активном окне Карты; "Нет" (FALSE) при запросе подписей, которые появляются, когда Карта помещена в Отчет.

### Величина, полученная в результате:

Логическая величина: "Да" (TRUE) означает, что подпись существует в определенной строке.

### Описание:

Вызывайте **LabelFindByID( )** когда Вы хотите запросить подпись из определенной строки в слое Карты. Если величина, полученная в результате – "Да" (TRUE), то подпись существует в строке, и Вы можете запросить подпись вызовом функции **Labelinfo( )**.

### Пример:

Следующий пример изображает карту мира, автоматически ее подписывает и затем определяет, была ли изображена подпись из определенной строки таблицы.

```
Include "mapbasic.def"
Dim b_morelabels As Logical
Dim i_mapid As Integer
Dim obj_mytext As Object

Open Table "World" Interactive As World
Map From World
i_mapid = FrontWindow()
Set Map Window i_mapid Layer 1 Label Auto On

' Убедитесь, что все подписи изображены, перед тем как мы продолжим...
Update Window i_mapid

' Теперь посмотрим, подписана ли автоматически строка # 1
```

## Функция LabelFindByID( )

---

```
b_morelabels = LabelFindByID(i_mapid, 1, 1, "", TRUE)

If b_morelabels Then
    ' Объект подписан, теперь запросим его подпись.

    obj_mytext = LabelInfo(i_mapid, 1, LABEL_INFO_OBJECT)

    ' В этом месте Вы можете сохранить объект obj_mytext
    ' в постоянной таблице; или можете запросить его
    ' с помощью функций ObjectInfo() или ObjectGeography().

End If
```

**Смотрите также:**

**LabelFindFirst( ), LabelFindNext( ), Labelinfo( )**



## Функция **LabelFindFirst( )**

### Назначение:

Инициализирует внутренний указатель подписи, так что Вы можете запросить первую подпись на слое Карты.

### Синтаксис:

**LabelFindFirst**(*map\_window\_id*, *layer\_number*, *b\_mapper*)

*map\_window\_id* – целочисленный индекс окна id, определяющий окно Карты;

*layer\_number* – номер слоя в текущем окне Карты (например, 1 для верхнего слоя);

*b\_mapper* – логическая величина. “Да” (TRUE) при запросе подписей, которые появляются при активном окне Карты; “Нет” (FALSE) при запросе подписей, которые появляются, когда Карта помещена в Отчет.

### Величина, полученная в результате:

Логическая величина: “Да” (TRUE) означает, что подпись существует в определенном слое (либо подписи видимы в данный момент, либо пользователь их редактировал, либо эти отредактированные подписи в данный момент невидимы).

### Описание:

Вызовите функцию **LabelFindFirst( )** когда Вам понадобится запросить подписи на слое Карты в цикле. Запрос подписей является процессом из двух шагов:

1. Установите внутренний указатель подписи MapBasic’a вызовом одной из следующих функций: **LabelFindFirst( )**, **LabelFindNext( )** или **LabelFindByID( )**.
2. Если функция, которую Вы вызвали на шаге 1, не возвращает “Нет” (FALSE), Вы можете запросить текущую подпись вызовом функции **Labelinfo( )**.

Для продолжения запроса дополнительных подписей, вернитесь к шагу 1.

### Пример:

Для примера, смотрите **Labelinfo( )**.

### Смотрите также:

**LabelFindByID( )**, **LabelFindNext( )**, **Labelinfo( )**

### Функция `LabelFindNext( )`

#### Назначение:

Перемещает внутренний указатель подписи, так что Вы можете запрашивать следующую подпись на слое Карты.

#### Синтаксис:

**`LabelFindNext(map_window_id , layer_number)`**

*map\_window\_id* – целочисленный индекс окна id, определяющий окно Карты;

*layer\_number* – номер слоя в текущем окне Карты (например, 1 для внешнего слоя).

#### Величина, полученная в результате:

Логическая величина: “Да” (TRUE) означает, что указатель подписи передвинут к следующей подписи; “Нет” (FALSE) означает, что на этом слое более нет подписей.

#### Описание:

После того, как Вы вызовете **`LabelFindFirst( )`**, чтобы начать запрос подписей, можно вызвать функцию **`LabelFindNext( )`**, чтобы переместиться к следующей подписи на этом же слое.

#### Пример:

Для примера смотрите **`Labelinfo( )`**.

#### Смотрите также:

**`LabelFindByID( )`**, **`LabelFindFirst( )`**, **`Labelinfo( )`**

## Функция Labelinfo( )

### Назначение:

Возвращает информацию о подписи на Карте.

### Синтаксис:

**Labelinfo( map\_window\_id , layer\_number , attribute )**

*map\_window\_id* – целочисленный индекс окна id, определяющий окно Карты;

*layer\_number* – номер слоя в текущем окне Карты (например, 1 для верхнего слоя);

*attribute* – код, указывающий тип возвращаемой информации, см. таблицу ниже.

### Величина, полученная в результате:

Тип результата зависит от значения параметра *attribute*.

### Описание:

Функция **Labelinfo( )** возвращает информацию о подписи в окне Карты. Обратите внимание на то, что подписи отличаются от текстовых объектов. Для запроса текстовых объектов используйте функции **ObjectInfo( )** или **ObjectGeography( )**.

Перед вызовом **Labelinfo( )** Вы должны инициализировать внутренний указатель подписей MapBasic, используя функции **LabelFindFirst( )**, **LabelFindNext( )** или **LabelFindByID( )**. Смотрите пример ниже.

Параметр *attribute* должен быть одним из кодов, которые приведены в следующей таблице; коды определены в MAPBASIC.DEF.

коды <i>attribute</i>	Величина, полученная в результате действия Labelinfo( )
LABEL_INFO_ANCHORX	Величина типа Float, указывает X-координату места привязки подписи.
LABEL_INFO_ANCHORY	Величина типа Float, указывает Y-координату места привязки подписи.
LABEL_INFO_DRAWN	Логическая величина; “Да” (TRUE) если подпись в текущий момент видима.
LABEL_INFO_EDIT	Логическая величина; “Да” (TRUE) если подпись редактировалась.
LABEL_INFO_EDIT_ANCHOR	Логическая величина; “Да” (TRUE) если подпись переместилась.
LABEL_INFO_EDIT_ANGLE	Логическая величина; “Да” (TRUE) если угол наклона подписи изменился.
LABEL_INFO_EDIT_FONT	Логическая величина; “Да” (TRUE) если шрифт подписи изменился.
LABEL_INFO_EDIT_OFFSET	Логическая величина; “Да” (TRUE) если расстояние в точках от подписи до места прикрепления изменилось.
LABEL_INFO_EDIT_PEN	Логическая величина; “Да” (TRUE) если стиль линии указки изменился.

## Функция Labelinfo( )

---

LABEL_INFO_EDIT_POSITION	Логическая величина; “Да” (TRUE) если позиция подписи (относительно места прикрепления) изменилась.
LABEL_INFO_EDIT_TEXT	Логическая величина; “Да” (TRUE) если текст подписи изменился.
LABEL_INFO_EDIT_TEXTARROW	Логическая величина; “Да” (TRUE) если стрелка указки подписи изменилась.
LABEL_INFO_EDIT_TEXTLINE	Логическая величина; “Да” TRUE если указка переместилась.
LABEL_INFO_EDIT_VISIBILITY	Логическая величина; “Да” (TRUE) если видимость подписи установлена на “Скрыть” (OFF).
LABEL_INFO_OBJECT	Возвращает текстовый объект, который практически является подписью. Эта функция позволяет конвертировать подпись в текстовый объект, который Вы можете сохранить в постоянной таблице.
LABEL_INFO_OFFSET	Целочисленное значение от 0 до 50, показывающее расстояние (в точках) от подписи до места прикрепления.
LABEL_INFO_POSITION	Целочисленное значение от 0 до 8, показывающее положение подписи относительно места прикрепления. Величина, полученная в результате соответствует одному из следующих кодов: LAYER_INFO_LBL_POS_CC (0), LAYER_INFO_LBL_POS_TL (1), LAYER_INFO_LBL_POS_TC (2), LAYER_INFO_LBL_POS_TR (3), LAYER_INFO_LBL_POS_CL (4), LAYER_INFO_LBL_POS_CR (5), LAYER_INFO_LBL_POS_BL (6), LAYER_INFO_LBL_POS_BC (7), LAYER_INFO_LBL_POS_BR (8). Например, если подпись ниже и правее места прикрепления, ее позиция обозначается кодом 8; если подпись располагается выше и левее места прикрепления, ее код будет 1.
LABEL_INFO_ROWID	Целочисленное значение, представляющее индекс ID строки, в которой расположена подпись; возвращает 0, если подписи не существует.
LABEL_INFO_SELECT	Логическая величина; “Да”(TRUE) если подпись выбрана.
LABEL_INFO_TABLE	Строковая величина, представляющая имя таблицы, в которой находится данная подпись. Используется при работе с сшитой таблицей, в том случае, если Вам надо узнать в какой из составляющих ее таблиц находится данная подпись.

### Пример:

Следующий пример показывает, как в цикле идет поиск подписи среди других подписей в определенной строке, используя функцию **Labelinfo( )**, запрашивающую каждую подпись.

```
Dim b_morelabels As Logical
Dim i_mapid, i_layernum As Integer
Dim obj_mytext As Object

' В этом месте Вы присваиваете окну Карты ID индекс i_mapid,
' и присваиваете слою номер i_layernum.

b_morelabels = LabelFindFirst(i_mapid, i_layernum, TRUE)
Do While b_morelabels
    obj_mytext = LabelInfo(i_mapid, i_layernum, LABEL_INFO_OBJECT)
    ' В этом месте Вы можете сохранить объект obj_mytext
    ' в постоянной таблице; или можете запросить его,
    ' вызывая функции ObjectInfo() или ObjectGeography().
    b_morelabels = LabelFindNext(i_mapid, i_layernum)
Loop
```

### Смотрите также:

**LabelFindByID( ), LabelFindFirst( ), LabelFindNext( )**

### Функция LayerInfo( )

#### Назначение:

Возвращает информацию о слое в окне Карты.

#### Синтаксис:

**LayerInfo(*mapper\_window\_id*, *layer\_number*, *attribute* )**

где

*mapper\_window\_id* – идентификатор окна Карты;

*layer\_number* – номер слоя (1 – самый верхний слой); чтобы определить номер слоя в окне Карты, обратитесь к функции MapperInfo( ).

*attribute* – целочисленный код.

#### Величина, полученная в результате:

Тип величины зависит от значения *attribute*.

#### Предупреждение:

Многие коды в функции **LayerInfo( )** используются только для "нормальных" слоев Карты, то есть функция не всегда применима к Косметическому слою, тематическому слою (слою, содержащему результат условного выделения) и слою с растровыми изображениями.

#### Описание:

Функция **LayerInfo( )** возвращает информацию об определенном слое в окне Карты *mapper\_window\_id*. Параметр *layer\_number* задает слой на карте (0 – Косметический слой, 1 – самый верхний слой, и т. д.). Параметр *attribute* должен принимать значения целочисленного кода, управляющего типом возвращаемой функцией информации. В следующей таблице в первой колонке приводятся для этих кодов имена, которые установлены в файле стандартных определений MapBasic MAPBASIC.DEF. Отсюда также можно запросить настройки Hotlink, используя атрибуты Layer\_Hotlink.

<i>attribute</i> code	LayerInfo( ) Возвращаемое значение
LAYER_INFO_NAME	Строка (тип String), содержащая имя таблицы, данные которой представлены на этом слое. Если слой Косметический, то строка будет "Cosmetic1". Имя может быть использовано в следующих операторах (например, <b>Select</b> ).
LAYER_INFO_EDITABLE	Логическая величина (тип Logical), показывающая, изменяем ли слой или нет; TRUE если слой изменяемый.
LAYER_INFO_LBL_PARTIALSEGS	Логическая величина; TRUE если флажок Label Partial Objects установлен для этого слоя.
LAYER_INFO_SELECTABLE	Логическая величина (тип Logical), показывающая, доступен ли слой или нет; TRUE если слой изменяемый.
LAYER_INFO_PATH	Строка (тип String), содержащая DOS-маршрут, по которому находится файл таблицы, данные которой представлены на этом слое.

LAYER_INFO_ZOOM_LAYERED	Логическая величина (тип Logical), показывающая, применяется ли масштабный эффект для данного слоя; TRUE если масштабный эффект применяется.
LAYER_INFO_ZOOM_MIN	Число (тип Float), минимальный порог для масштабного эффекта (в текущих в MapBasic единицах измерения расстояний). Для установки единиц измерения расстояний используется оператор <b>Set Distance Units</b> .
LAYER_INFO_ZOOM_MAX	Число (тип Float), максимальный порог для масштабного эффекта.
LAYER_INFO_COSMETIC	Логическая величина (тип Logical). "Да" (TRUE), если данный слой косметический.
LAYER_INFO_DISPLAY	Целочисленный код (тип SmallInt), говорящий, как показывается слой:  LAYER_INFO_DISPLAY_OFF (слой не отображается);  LAYER_INFO_DISPLAY_GRAPHIC (объекты слоя появляются в "стандартном" стиле—стили хранятся в таблице);  LAYER_INFO_DISPLAY_GLOBAL (объекты слоя появляются в "измененном" стиле, определенном в диалоге Управление слоями);  LAYER_INFO_DISPLAY_VALUE (объекты в этом слое появляются в тематической раскраске)
LAYER_INFO_OVR_LINE	Стиль линии, используемый для отображения линейных объектов.
LAYER_INFO_OVR_PEN	Стиль линии, используемый для отображения контуров площадных объектов.
LAYER_INFO_OVR_BRUSH	Стиль штриха, используемый для отображения объектов, имеющих площадь.
LAYER_INFO_OVR_SYMBOL	Стиль символа, используемый для отображения точечных объектов.
LAYER_INFO_OVR_FONT	Стиль шрифта, используемый для отображения текстовых объектов.
LAYER_INFO_LBL_CURFONT	Логическая величина. В приложениях, откомпилированных с MapBasic версии 4.0 или позднее результат всегда будет "Нет".  В приложениях, откомпилированных с MapBasic версии 3.x, результатом будет:

## Функция LayerInfo( )

---

	"Да" (TRUE), если слой использует текущую установку для стиля шрифта, или "Нет" (FALSE), если слой использует специальную установку для стиля шрифта (смотрите LAYER_INFO_LBL_FONT).
LAYER_INFO_LBL_FONT	Стиль шрифта для подписей.
LAYER_INFO_LBL_EXPR	Строка с текстом подписи.
LAYER_INFO_LBL_LT	Целочисленный код (тип SmallInt), показывающий, какой тип указки используется: LAYER_INFO_LBL_LT_NONE (если нет указки) LAYER_INFO_LBL_LT_SIMPLE (если указка является простой линией) LAYER_INFO_LBL_LT_ARROW (если указка является линией со стрелкой)
LAYER_INFO_LBL_PARALLEL	Логическая величина: "Да" (TRUE), если на слое разрешено поворачивать подписи.
LAYER_INFO_LBL_POS	Целочисленный код (тип SmallInt), показывающий ориентацию подписи (литера T обозначает верх, литера B – низ, C – центр, R – право, L – лево): LAYER_INFO_LBL_POS_TL LAYER_INFO_LBL_POS_TC LAYER_INFO_LBL_POS_TR LAYER_INFO_LBL_POS_CL LAYER_INFO_LBL_POS_CC LAYER_INFO_LBL_POS_CR LAYER_INFO_LBL_POS_BL LAYER_INFO_LBL_POS_BC LAYER_INFO_LBL_POS_BR
LAYER_INFO_LBL_VISIBILITY	Целочисленная величина, типа Smallint, соответствующая режиму показа подписей на слое (смотрите предложение <b>Visibility</b> оператора <b>Set Map</b> ). Результатом будет один из следующих кодов: LAYER_INFO_LBL_VIS_ON (если подписи на слое видимы) LAYER_INFO_LBL_VIS_OFF (если подписи на слое невидимы)



	LAYER_INFO_LBL_VIS_ZOOM (если подписи видимы в результате масштабного эффекта)
LAYER_INFO_LBL_ZOOM_MIN	Действительная величина (тип Float), минимальное значение показа подписей при масштабном эффекте.
LAYER_INFO_LBL_ZOOM_MAX	Действительная величина (тип Float), максимальное значение показа подписей при масштабном эффекте.
LAYER_INFO_LBL_AUTODISPLAY	Логическая величина: TRUE, если слой подписывается автоматически. Смотрите предложение <b>Auto</b> оператора <b>Set Map</b> .
LAYER_INFO_LBL_OVERLAP	Логическая величина: "Да" (TRUE), если допускается пересечение подписей.
LAYER_INFO_LBL_DUPLICATES	Логическая величина: "Да" (TRUE), если допускается дублирование подписей.
LAYER_INFO_LBL_OFFSET	Целочисленная величина типа Smallint от 0 до 50, смещение от подписи до центроида подписываемого объекта в точках.
LAYER_INFO_LBL_MAX	Целочисленная величина типа Integer, максимальное число подписей, разрешенное для этого слоя. Если такое число не назначено, то возвращается число 2 147 483 647.
LAYER_INFO_LBL_PARTIALSEG	Логическая величина: "Да" (TRUE), если флажок Label Partial Segments для этого слоя установлен.
LAYER_INFO_ARROWS	Логическая величина: "Да" (TRUE), если в линейных объектах используется стрелка.
LAYER_INFO_NODES	Логическая величина: "Да" (TRUE), если показываются узлы объектов слоя.
LAYER_INFO_CENTROIDS	Логическая величина: "Да" (TRUE), если показываются центроиды объектов слоя.
LAYER_INFO_SELECTABLE	Логическая величина: "Да" (TRUE), если слой изменяемый.
LAYER_INFO_PATH	Строка, полный путь к таблице, ассоциированной со слоем карты.
LAYER_INFO_TYPE	Короткое целое, показывающее тип слоя: LAYER_INFO_TYPE_NORMAL для обычного слоя; LAYER_INFO_TYPE_COSMETIC для Косметического слоя; LAYER_INFO_TYPE_IMAGE для растрового слоя;

## Функция LayerInfo( )

---

	LAYER_INFO_TYPE_THEMATIC для тематического слоя.
	LAYER_INFO_TYPE_GRID для грида.
LAYER_HOTLINK_EXPR	Возвращает выражение имени файла для Hotlink.
LAYER_HOTLINK_MODE	Возвращает режим Hotlink для слоя, одно из следующих предопределенных значений: HOTLINK_MODE_LABEL HOTLINK_MODE_OBJ HOTLINK_MODE_BOTH
LAYER_HOTLINK_RELATIVE	Возвращает "ДА" (True), если указан относительный адрес.

### Пример

Многие коды из приведенных выше не применимы к Тематическому слою, тематическому слою и слою с растровым изображением. Функцию **LayerInfo( )** с аргументом LAYER-INFO-TYPE используют для определения, каким является слой. Например:

```
i_layer_type = LayerInfo( map_id, layer_num, LAYER_INFO_TYPE)

If i_layer_type = LAYER_INFO_TYPE_NORMAL Then
    / ... then this is a "normal" layer
    /
End If
```

### Смотрите также

MapperInfo( )

## Оператор Layout

### Назначение:

Открывает новое окно Отчета.

### Синтаксис:

```
Layout
[ Position (x, y) [ Units paperunits ] ]
[ Width window_width [ Units paperunits ] ]
[ Height window_height [ Units paperunits ] ]
[ { Min | Max } ]
```

где

*paperunits* – имя единицы измерения (например, "in" – дюйм);

*x, y* – координаты левого верхнего угла окна Отчета в определенных единицах измерения;

*window\_width* – ширина окна;

*window\_height* – высота окна.

### Описание:

Оператор **Layout** создает новое пустое окно Отчета.

Ключевые слова **Max** и **Min** используются для открытия окна Отчета на все рабочее окно MapInfo или свертывания в иконку. В системе Macintosh окно не может быть свернуто, поэтому ключ **Min** игнорируется.

Предложения **Width** и **Height** определяют ширину и высоту окна Отчета. Но эти характеристики не влияют на размеры страницы макета. Для назначения размеров страниц и их количества используйте оператор **Set Layout**.

MapInfo организывает для каждого окна Отчета специальную, скрытую таблицу, которой дается имя *Layoutn*. Первому открытому окну Отчета дается имя *Layout1* и так далее. Программа, написанная на MapBasic, может создавать, выбирать или изменять объекты в окне Отчета, обращаясь к нему по этому имени. Например, следующий оператор выбирает все объекты в окне Отчета:

```
Select * From Layout1
```

### Пример:

В следующем примере создается окно Отчета в два дюйма шириной и четыре высотой. Верхний левый угол окна располагается в верхнем левом углу рабочей области MapInfo.

```
Layout Position (0,0) Width 2 Height 4
```

### Функция LCase\$( )

#### Назначение:

Возвращает строку, преобразуя все прописные буквы в строчные.

#### Синтаксис:

**LCase\$(string\_expr)**

где

*string\_expr* – строковое выражение.

#### Величина, полученная в результате:

Строка. Величина типа String.

#### Описание:

Функция **LCase\$( )** возвращает строку, полученную из строки, представленной выражением *string\_expr*, преобразованием всех заглавных букв в строчные.

Преобразованию подвергаются только буквы: латинские – от А до Z, и русские – от А до Я. Цифры и другие текстовые символы не преобразуются. Например, функция **LCase\$("A#12a")** равна "a#12a".

#### Пример:

```
Dim regular, lower-case As String
regular = "Вышний Волочек"
lower-case = Lcase$(regular)
'
' Первый элемент массива равен строке "вышний волочек",
'
```

#### Смотрите также:

**Proper\$( )**, **UCase\$( )**

## Функция Left\$( )

### Назначение:

Возвращает левую часть строки, выделяя определенное количество символов из исходной.

### Синтаксис:

**Left\$(string\_expr, num\_expr)**

где

*string\_expr* – строковое выражение;

*num\_expr* – численное выражение, результат которого ноль или более.

### Величина, полученная в результате:

Строка. Величина типа String.

### Описание:

Функция **Left\$( )** возвращает строковую величину, полученную из исходной строки, представленной выражением *string\_expr*, путем выделения первых символов. Количество символов задается вторым параметром *num\_expr*.

В результате вычисления выражения *num\_expr* должно получаться целое положительное число, не равное нулю. Если численный параметр меньше единицы, то функция **Left\$( )** вернет пустую строку.

Если численный параметр *num\_expr* больше, чем число символов в строке *string\_expr*, результат функции **Left\$( )** будет равен строке, представленной выражением *string\_expr*.

### Пример:

```
Dim whole, partial As String
whole = "Казахстан"
partial = Left$(whole, 5)
'
' переменная partial теперь равна строке "Казах"
'
```

### Смотрите также:

**Mid\$( ), Right\$( )**

### Функция LegendFrameInfo( )

#### Назначение

Возвращает информацию о разделе в легенде.

#### Синтаксис

**LegendFrameInfo( *window\_id*, *frame\_id*, *attribute* )**

*window\_id* - число, указывающее какое окно легенды Вы хотите опросить.

*frame\_id* - число, указывающее какой раздел в окне легенды Вы хотите опросить. Разделы пронумерованы от 1 до n где n это номер в легенде.

*attribute* - это целочисленный код, указывающий какой тип информации возвращается.

#### Возвращаемое значение

Оно зависит от параметра атрибутов.

#### Коды атрибута

FRAME_INFO_TYPE	Возвращает одно из следующих предопределенных констант, определяющих тип раздела: FRAME_TYPE_STYLE FRAME_TYPE_THEME
FRAME_INFO_MAP_LAYER_ID	Возвращает индекс id слоя, с которым соотносится раздел.
FRAME_INFO_REFRESHABLE	Возвращает true если раздел был создан без ключевого слова No-refresh. Всегда возвращает true для тематических разделов.
FRAME_INFO_POS_X	Возвращает расстояние от верхнего левого угла раздела до левого края канвы легенды (в бумажных единицах).
FRAME_INFO_POS_Y	Возвращает расстояние от верхнего левого угла раздела до верхнего края канвы легенды (в бумажных единицах).
FRAME_INFO_WIDTH	Возвращает ширину раздела (в бумажных единицах).
FRAME_INFO_HEIGHT	Возвращает высоту раздела (в бумажных единицах).
FRAME_INFO_TITLE	Возвращает заголовок раздела или тематического раздела.
FRAME_INFO_TITLE_FONT	Возвращает шрифт заголовка раздела. Возвращает стандартный шрифт заголовка, если раздел не имеет заголовка или это тематический раздел.
FRAME_INFO_SUBTITLE	Возвращает подзаголовок раздела.
FRAME_INFO_SUBTITLE_FONT	Шрифт подзаголовка раздела.
FRAME_INFO_BORDER_PEN	Возвращает параметры линии, использованной для рамки.
FRAME_INFO_NUM_STYLES	Возвращает число типа раздела. Ноль для тематического раздела.
FRAME_INFO_VISIBLE	Возвращает true если раздел видимый (тематические разделы могут быть невидимыми).

FRAME_INFO_COLUMN	Возвращает атрибуты имени колонки для легенды в виде строки. Возвращает пустую строку, если раздел тематический.
FRAME_INFO_LABEL	Возвращает выражение подписи в виде строки. Возвращает пустую строку для тематического раздела.

## Функция LegendInfo( )

---

### Функция LegendInfo( )

#### Назначение

Возвращает информацию о легенде.

#### Синтаксис

**LegendInfo**( *window\_id*, *attribute* )

*window\_id* - это число, указывающее, какое окно легенды Вы опрашиваете.

*attribute* - это целочисленный код, указывающий, какой тип информации возвращается.

#### Возвращаемое значение

Зависит от атрибута параметра.

<u>Код атрибута</u>	<u>Описание</u>
LEGEND_INFO_MAP_ID	Возвращает id материнского окна карты (можно так же получить это значение из WindowInfo() с кодом WIN_INFO_TABLE).
LEGEND_INFO_ORIENTATION	Возвращает предопределенное значение, характеризующее ориентацию легенды:
ORIENTATION_PORTRAIT	Возвращает ориентацию бумаги
ORIENTATION_LANDSCAPE	Возвращает ориентацию бумаги
ORIENTATION_CUSTOM	Возвращает ориентацию бумаги
LEGEND_INFO_NUM_FRAMES	Возвращает число разделов в легенде.

#### Смотрите также:

**Функция LegendStyleInfo( )**



## Функция LegendStyleInfo( )

### Назначение

Возвращает информацию о стиле, используемом в разделе легенды.

### Синтаксис

**LegendInfo( *window\_id*,, *frame\_id*, *style\_id*, *attribute* )**

*window\_id* - это число, указывающее, какое окно легенды Вы хотите опросить.

*frame\_id* - это число, определяющее, какой раздел в легенде Вы хотите опросить. Разделы пронумерованы от 1 до n где n это число всех разделов в легенде.

*style\_id* - это число, определяющее, какой стиль внутри раздела Вы хотите опросить. Стили пронумерованы от 1 до n где n это число стилей в разделе.

*attribute* - это целочисленный код, показывающий, какой тип информации возвращается.

### Возвращение

Код атрибута	Описание
LEGEND_STYLE_INFO_TEXT	Возвращает текст стиля.
LEGEND_STYLE_INFO_FONT	Возвращает шрифт стиля.
LEGEND_STYLE_INFO_OBJ	Возвращает объект стиля.

### Сообщение об ошибке

Генерируется сообщение об ошибке, когда раздел не имеет стилей (тематический раздел).

### Смотрите также

**LegendInfo( )**

### Функция Len( )

#### Назначение:

Возвращает количество символов в строке или число байтов в переменной.

#### Синтаксис:

**Len(*expr*)**

где *expr* – выражение (не используются типы Pen, Brush, Symbol, Font и Alias).

#### Величина, полученная в результате:

Короткое целое число. Величина типа SmallInt.

#### Описание:

Информация, которую несет величина, возвращаемая функцией **Len( )**, зависит от параметра *expr*.

Если выражение *expr* является строкой, то функция **Len( )** вернет количество символов в строке.

Если *expr* – переменная MapBasic, то результат функции **Len( )** будет размером переменной в байтах.

Так, если тип переменной был объявлен как Integer (целое число), то функция **Len( )** возвращает 4 (четыре), т. к. переменной типа Integer отводится 4 байта. Для переменной типа SmallInt (короткое целое число), результатом будет двойка, т. к. переменной типа SmallInt отводится 2 байта.

#### Пример:

```
Dim name-length As SmallInt
name-length = Len("Москва")
' переменная name-length равна 6
```

#### Смотрите также:

**ObjectLen( )**

## Функция Like( )

### Назначение:

Возвращает TRUE или FALSE, сравнивая строку с шаблоном.

### Синтаксис:

**Like**(*string*, *pattern\_string*, *escape\_char*)

где

*string* – строка;

*pattern\_string* – шаблон для сравнения, который является строкой, состоящей из регулярных и специальных символов;

*escape\_char* – строковое выражение, задающее символ (например, "/"), отменяющий специальный символ, если он явно должен использоваться в шаблоне. Если отменяющий символ не назначается, то используется пустая строка ("").

### Величина, полученная в результате:

Логическая. Величина типа Logical.

### Описание:

Функция **Like( )** определяет, подходит ли строка *string* под шаблон *pattern\_string*. Шаблон представляет собой строку из регулярных символов и специальных. При сравнении регулярные символы должны совпадать с соответствующими символами в строке *string*, причем строчные и прописные буквы различаются. Специальные символы задают неопределенное совпадение:

– (знак подчеркивания)      соответствует одному символу;

% (percent)      соответствует нескольким символам или не одному.

Для явного задания знаков подчеркивания и процента они используются вместе с символом *escape\_char* перед специальным. Примеры приведены в следующей таблице:

#### Критерий совпадения

Начало с "South"

Окончание "America"

Используется "ing" в любом месте

Начало с подчеркивания

#### Вызов функции

Like( string-var, "South%", "" )

Like( string-var, "%America", "" )

Like( string-var, "%ing%", "" )

Like( string-var, "\-%", "" )

### Оператор Line Input

#### Назначение:

Читает строку из текстового файла в переменную.

#### Синтаксис:

**Line Input** [#]*filenum*, *var\_name*

где

*filenum* – номер открытого файла, целое число;

*var\_name* – имя переменной строкового типа.

#### Описание:

Оператор **Line Input** читает текущую строку из текстового файла в переменную, объявленную как String. Текстовый файл должен быть открыт для последовательного доступа (**INPUT**).

Оператор **Line Input** читает каждую строку полностью. Если строка содержит список выражений, разделенный запятыми и Вы хотите каждое выражение присвоить отдельной переменной, то используйте оператор **Input** вместо **Line Input**.

#### Пример:

В нижеприведенном коде читается строка за строкой из первого файла и копируется во второй файл.

```
Dim str As String
Open File "original.txt" For Input As #1
Open File "copy.txt"    For Output As #2
Do While Not EOF(1)
    Line Input #1, str
    If Not EOF(1) Then
        Print #2, str
    End If
Loop
Close File #1
Close File #2
```

#### Смотрите также:

**Input #, Open File, Print #**

## Функция LocateFile\$( )

### Назначение

Возвращает путь к одному из служебных файлов MapInfo.

### Синтаксис

**LocateFile\$(*file\_id*)**

*file\_id* - это одно из следующих значений:

LOCATE_PREF_FILE	файл настроек (mapinfow.prf)
LOCATE_DEF_WOR	файл стандартного рабочего набора (mapinfow.wor)
LOCATE_CLR_FILE	файл цветов (mapinfow.clr)
LOCATE_PEN_FILE	файл стилей линий (mapinfow.pen)
LOCATE_FNT_FILE	файл символов (mapinfow.fnt)
LOCATE_ABB_FILE	файл аббревиатур (mapinfow.abb)
LOCATE_PRJ_FILE	файл проекций (mapinfow.prj)
LOCATE_MNU_FILE	файл меню (mapinfow.mnu)
LOCATE_CUSTSYMB_DIR	пользовательская директория символов (custsymb)
LOCATE_THMTMPLT_DIR	папка тематических шаблонов (thmtmpl)
LOCATE_GRAPH_DIR	папка поддержки графики (GraphSupport)

### Возвращаемое значение

Строковое

### Описание

В зависимости от запрашиваемого индексного значения *id* эта функция возвращает путь к тому месту, где MapInfo найдет этот файл. В версиях до 6.5 эти файлы большей частью устанавливались в ту же директорию, где находился файл mapinfow.exe.

В версии 6.5 MapInfo устанавливает эти файлы в директорию Application Data, но есть несколько других допустимых мест, включая директорию, где находится mapinfow.exe. Приложения MapBasic не могут определить положение этих файлов, для такого определения надо использовать функцию LocateFile\$().

### Пример

```
include "mapbasic.def"
declare sub main
sub main
dim sGraphLocations as string
sGraphLocations = LocateFile$(LOCATE_GRAPH_DIR)
Print sGraphLocations
end sub
```

### Смотрите также

Функция GetFolderPath\$( )

### Функция LOF( )

#### Назначение:

Возвращает длину открытого файла.

#### Синтаксис:

**LOF**(*file*)

где *file* – номер открытого файла.

#### Величина, полученная в результате:

Целое число. Величина типа Integer.

#### Описание:

Функция **LOF( )** возвращает размер открытого файла в байтах. Параметр *file* должен быть целочисленным номером файла, открытого до вызова функции оператором **Open File**.

#### Ошибки:

В результате выполнения функции может генерироваться код ошибки

ERR-FILEMGR-NOTOPEN, если файл не открыт.

#### Пример:

```
Dim size As Integer
Open File "import.txt" For Binary As #1
size = LOF(1)
'
' переменная size теперь равна размеру файла,
' открытого под номером 1
'
```

#### Смотрите также:

**Open File**

## Функция Log( )

### Назначение:

Вычисляет натуральный логарифм.

### Синтаксис:

**Log(*num\_expr*)**

где *num\_expr* – численное выражение.

### Величина, полученная в результате:

Вещественное число. Величина типа Float.

### Описание:

Функция **Log( )** возвращает значение натурального логарифма от числа, полученного в результате вычисления выражения *num\_expr*.

Функция натурального логарифма обратна функции экспоненты (число *e* в степени *num\_expr*).

Число *e* иррационально и примерно равно 2.7182818.

Логарифм может вычисляться только от положительного числа. Если *num\_expr* есть отрицательная величина, то **Log( )** функция вернет ошибку.

Вы можете вычислить логарифм и по другому основанию (например, 10), используя натуральный логарифм. Для вычисления логарифма по основанию 10 от числа *n* надо разделить натуральный логарифм от числа *n* (**Log( *n* )**) на натуральный логарифм от 10 (**Log( 10 )**).

### Пример:

```
Dim original-val, log-val As Float
original-val = 2.7182818
log-val = Log(original-val)
'
' log-val будет равно 1 (приблизительно),
' т. к. число e в степени 1 (единица) равно
' 2.7182818 (приблизительно)
'
```

### Смотрите также:

**Exp( )**

### Функция LTrim\$ ( )

#### Назначение:

Удаляет пробелы в начале строки.

#### Синтаксис:

**LTrim\$(string\_expr)**

где *string\_expr* – строковое выражение.

#### Величина, полученная в результате:

Строка. Величина типа String.

#### Описание:

Функция **LTrim\$ ( )** возвращает строковую величину, полученную из выражения *string\_expr*, удалением пробелов в начале строки, если они есть.

#### Пример:

```
Dim name As String
name = "  Мария Анатольевна Смирнова"
name = Ltrim$(name)
'
' name теперь имеет значение "Мария Анатольевна Смирнова"
'
```

#### Смотрите также:

**RTrim\$ ( )**



## Процедура Main

### Назначение:

Главная процедура, которая выполняется первой при загрузке прикладной программы.

### Синтаксис:

```
Declare Sub Main
Sub Main
  statement_list
End Sub
```

где

*statement\_list* – список операторов.

### Описание:

**Main** – стандартное имя процедуры MapBasic. Если текст программы на MapBasic содержит процедуру под этим именем, то выполнение программы начнется с этой процедуры. **Main**-процедура может вызывать другие процедуры (смотрите описание оператора **Call**).

Вы можете не объявлять процедуру **Main**. В этом случае первый оператор программы понимается как оператор из процедуры **Main**. И MapBasic начинает выполнять программу, как если бы процедура Main была объявлена перед этим оператором. Назовем этот случай "неявным заданием процедуры **Main**".

### Пример:

Прикладная программа может состоять из одной строки. Например, эта программа выполняет только один оператор:

```
Note "Проверка: один, два, три. Как видно?"
```

В этой программе процедура Main задана неявно.

Мы можем эту программу написать и так:

```
Declare Sub Main
Sub Main
  Note "Проверка: один, два, три. Как видно?"
End Sub
```

Здесь главная процедура задана в явном виде. При этом результат выполнения этих двух программ будет один и тот же.

Следующая программа также содержит неявную Main-процедуру. Из нее вызывается подпрограмма, процедура которой объявлена под именем "Talk". Для вызова используется оператор **Call**.

```
Declare Sub Talk(ByVal msg As String)
Call Talk("Привет")
Call Talk("Всего хорошего")
Sub Talk(ByVal msg As String)
  Note msg
End Sub
```

Следующий пример содержит явную процедуру Main, из которой вызывается подпрограмма "Talk".

```
Declare Sub Main
Declare Sub Talk(ByVal msg As String)
Sub Main
```

## Процедура Main

---

```
Call Talk("Привет")
Call Talk("Всего хорошего")
End Sub

Sub Talk(ByVal msg As String)
    Note msg
End Sub
```

Результат выполнения программ будет таким же, как в предыдущем случае.

### Смотрите также:

[EndHandler](#), [RemoteMsgHandler](#), [SelChangedHandler](#), [Sub... End Sub](#), [ToolHandler](#), [Win-ClosedHandler](#)

## Функция MakeBrush( )

### Назначение:

Возвращает установку стиля штриха.

### Синтаксис:

**MakeBrush**(*pattern*, *forecolor*, *backcolor*)

где

*pattern* – тип штриха, целое число от 1 до 8 или от 12 до 71 (рисунки штриха смотрите в описании предложения **Brush**);

*forecolor* – цвет штриха в системе RGB;

*backcolor* – цвет фона в системе RGB.

Чтобы сделать фон прозрачным задайте значение *backcolor* -1 и значение *pattern* 3 или более.

### Величина, полученная в результате:

Величина типа Brush.

### Описание:

Функция **MakeBrush** возвращает величину типа Brush, определяющую стиль штриховки графического объекта. Возвращаемая величина может быть присвоена переменной типа Brush или использована как параметр оператора (таких как **Create Ellipse**, **Set Map**, **Set Style**, или **Shade**).

Смотрите описание предложения **Brush** об установках стиля Brush.

### Пример:

```
Include "MAPBASIC.DEF"  
Dim b-water As Brush  
b-water = MakeBrush(64, CYAN, BLUE)
```

### Смотрите также:

**Brush**, **CurrentBrush**( ), **StyleAttr**( )

### Функция MakeCustomSymbol( )

#### Назначение:

Возвращает символ, созданный из растрового файла.

#### Синтаксис:

**MakeCustomSymbol**(*filename, color, size, customstyle*)

где

*filename* – строка до 31 символа длиной, представляющая имя растрового файла (файл должен находиться в каталоге, специально зарезервированном для этого пользователем);

*color* – цвет в системе RGB;

*size* – целочисленный размер в пунктах от 1 до 48;

*customstyle* – целочисленный код, управляющий атрибутами фона и цвета.

#### Величина, полученная в результате:

Символ. Величина типа Symbol.

#### Описание:

Функция **MakeCustomSymbol( )** возвращает величину типа Symbol, основанную на растровом файле. Смотрите описание предложения **Symbol** о других типах символа.

В следующей таблице перечислены возможности настройки растрового символа:

Значение <i>customstyle</i>	Стиль символа
0	Не действуют режимы из группы “Эффекты” диалога “Стиль символа”, и символ появляется таким, какой он есть. Все белые пиксели растра прозрачны.
1	Действует режим “Добавить фон”; все белые пиксели растра непрозрачны.
2	Действует режим “Покрасить одним цветом”; все не белые точки растра закрашены одним цветом.
3	Установлены оба флажка (действуют оба режима).

#### Пример:

```
Include "mapbasic.def"  
Dim sym_marker As Symbol  
sym_marker = MakeCustomSymbol("CAR1-64.BMP", BLUE, 18, 0)
```

#### Смотрите также:

**CurrentSymbol( ), MakeFontSymbol( ), MakeSymbol( , StyleAttr( ), Symbol**

## Функция MakeFont( )

### Назначение:

Возвращает величину, являющуюся установкой стиля шрифта.

### Синтаксис:

**MakeFont**(*fontname, style, size, forecolor, backcolor*)

где

*fontname* – имя шрифта, строковая величина (например, "Helv");

*style* – численное выражение, в результате которого получается положительное целое число от 0 до 7 включительно;

*size* – размер шрифта, целое число;

*forecolor* – цвет символов шрифта в системе RGB;

*backcolor* – цвет фона или каймы в системе RGB или -1 для прозрачного фона.

### Величина, полученная в результате:

Установка стиля шрифта. Величина типа Font.

### Описание:

Функция **MakeFont( )** возвращает величину типа Font для определения шрифта, который может быть назначен текстовому объекту. Возвращаемая величина может быть присвоена переменной типа Font или использована как параметр в других операторах (таких как **Create Text** или **Set Style**).

Смотрите описание предложения **Font** для дополнительной информации о стиле шрифта.

### Пример:

```
Include "MAPBASIC.DEF"
Dim big-title As Font
big-title = MakeFont("Helvetica", 1, 20,BLACK,WHITE)
```

### Смотрите также:

**CurrentFont( ), Font, StyleAttr( )**

### Функция MakeFontSymbol( )

#### Назначение:

Возвращает символ, используя букву (символ) шрифта TrueType.

#### Синтаксис:

**MakeFontSymbol(*shape*, *color*, *size*, *fontname*, *fontstyle*, *rotation* )**

*shape* – целое число, величина типа SmallInt, от 31 или больше (31 – значение для невидимого символа), задающая код шрифта TrueType;

*color* – цвет символа шрифта в системе RGB (смотрите описание функции **RGB( )**);

*size* – целое число, величина типа SmallInt, от 1 до 48, назначающая размер символа в пунктах;

*fontname* – строка, имя шрифта TrueType (например, “WingDings”).

*fontstyle* – численный код, управляющий атрибутами шрифта, такими как жирное написание, курсив, контур;

*rotation* – действительное число, задающее угол поворота символа в градусах.

#### Величина, полученная в результате:

Символ. Величина типа Symbol.

#### Описание:

Функция **MakeFontSymbol( )** возвращает величину типа Symbol, используя заданный символ шрифта TrueType. Смотрите описание предложения **Symbol** для получения информации о других типах символов.

Следующая таблица приводит значения для параметра *fontstyle*, задающего стиль символа шрифта, в котором он будет скопирован в символ MapInfo:

Значение <i>fontstyle</i>	Стиль символа
0	Нормальное написание
1	Жирное написание
16	Черная кайма
32	Написание с тенью
256	Белая кайма

Для комбинирования двух или более стилей надо сложить код. Например, чтобы задать символ жирным и с тенью, надо использовать код 33. Белая и черная кайма взаимно исключают друг друга.

#### Пример:

```
Include "mapbasic.def"
Dim sym_marker As Symbol
sym_marker = MakeFontSymbol(65,RED,24,"WingDings",32,0)
```

#### Смотрите также:

**CurrentSymbol( ), MakeCustomSymbol( ), MakeSymbol( ), StyleAttr( ), Symbol**

## Функция MakePen( )

### Назначение:

Возвращает установку стиля линии.

### Синтаксис:

**MakePen**(*width, pattern, color*)

где

*width* – толщина линии в пунктах, целое число от 0 до 7;

*pattern* – тип линии (список в описании предложения **Pen**);

*color* – цвет линии в системе RGB.

### Величина, полученная в результате:

Величина типа Pen.

### Описание:

Функция **MakePen( )** возвращает величину типа Pen, определяющую стиль линии графического объекта. Возвращаемая величина может быть присвоена переменной типа Pen или использована как параметр в других операторах (таких как **Create Line**, **Create Pline**, **Set Style** или **Set Map**).

Смотрите описание предложения **Pen** для дополнительной информации о стиле линии.

### Пример:

```
Include "MAPBASIC.DEF"  
Dim p-bus-route As Pen  
p-bus-route = MakePen(3, 9, RED)
```

### Смотрите также:

**CurrentPen( ), Pen, StyleAttr( ), RGB( )**

### Функция **MakeSymbol( )**

#### Назначение:

Возвращает установку стиля символа. Действует для символов формата MapInfo версии 3.

#### Синтаксис:

**MakeSymbol**(*shape, color, size*)

где

*shape* – форма символа, целое число от 31 и более (31 для невидимого знака); стандартный набор символов использует коды от 31 до 67 (список смотрите в описании предложения **Symbol**);

*color* – цвет символа в системе RGB;

*size* – размер символа в пунктах, целое число от 1 до 48.

#### Величина, полученная в результате:

Величина типа Symbol.

#### Описание:

Функция **MakeSymbol( )** возвращает величину типа Symbol, определяющую стиль отображения точечного объекта. Возвращаемая величина может быть присвоена переменной типа Symbol или использована как параметр в других операторах (таких как **Set Map**, **Set Style** или **Shade**).

Чтобы создать символ из буквы или знака шрифта TrueType, используется функция

**MakeFontSymbol( )**.

Чтобы создать символ из растрового файла, используется функция **MakeCustomSymbol( )**.

Смотрите описание предложения **Symbol** для дополнительной информации о стиле символа.

#### Пример:

```
Include "MAPBASIC.DEF"  
Dim sym-marker As Symbol  
sym-marker = MakeSymbol(44, RED, 16)
```

#### Смотрите также:

**CurrentSymbol( )**, **MakeCustomSymbol( )**, **MakeFontSymbol( )**, **StyleAttr( )**, **Symbol**



## Оператор Map

### Назначение:

Открывает новое окно Карты.

### Синтаксис:

```
Map From table [, table ... ]
[ Position (x, y) [ Units paperunits ] ]
[ Width window_width [ Units paperunits ] ]
[ Height window_height [ Units paperunits ] ]
[ { Min | Max } ]
```

где

*table* – имя открытой таблицы;

*paperunits* – строка с именем единицы измерения (например, "in" – дюйм);

*x*, *y* – координаты верхнего левого угла окна Карты в заданных единицах;

*window\_width* – ширина окна в заданных единицах;

*window\_height* – высота окна в заданных единицах.

### Описание:

Оператор **Map** открывает новое окно Карты. После выполнения оператора приложение может изменять это окно при помощи оператора **Set Map**.

Таблица *table* должна быть заранее открыта. Не обязательно, чтобы в таблице были графические объекты, но она должна иметь структуру, позволяющую иметь объекты, связанные с информацией в записях.

Оператор **Map** должен содержать указание хотя бы на одну таблицу, так как Карта должна иметь хотя бы один слой, кроме Косметического. Если Вы хотите открыть окно Карты для данных из нескольких таблиц, то задайте их списком через запятую. В этом же порядке таблицы будут отображены в слоях карты: первая таблица для верхнего слоя. Обычно первой (на верхний слой), загружают таблицу с точечными объектами, а таблицу с областями (границами) – последней, на нижний слой.

Размер окна по умолчанию равен примерно четверти экрана, и положение его зависит от того, сколько окон уже открыто. Если Вы хотите при создании карты сами определить размер и место окна, то используйте оператор **Map**, в конструкцию которого входят предложения **Position**, **Height** и **Width**. Предложение **Position** задает расположение окна на экране. Координаты *x* и *y* определяют верхний левый угол окна Карты относительно верхнего левого угла окна MapInfo, а не относительно верхнего левого угла экрана (правда это не имеет смысла в системе Macintosh).

Ключевые слова **Max** и **Min** используются для открытия окна Карты на все рабочее окно MapInfo или для открытия окна свернутым в иконку. В системе Macintosh окно не может быть свернуто, поэтому ключ **Min** игнорируется.

Карта может иметь свои проекции. Открывая окно Карты, MapInfo использует проекции таблицы первого слоя. Пользователь может изменять проекции Карты при помощи команды КАРТА > ЕДИНИЦЫ ИЗМЕРЕНИЙ. Программа меняет проекции оператором **Set Map**.

### Пример:

Откроем окно Карты в 3 дюйма шириной и 2 дюйма высотой с двумя слоями (Косметический слой не считается). Верхний левый угол окна Карты будет ниже на 1 и правее на 1 дюйм от верхнего левого угла окна MapInfo.

## Оператор Map

---

```
Open Table "world.tab"  
Open Table "cust1994.tab" As customers  
Map from customers, world  
    Position (1,1) Width 3 Height 2
```

### Смотрите также:

Add Map, Remove Map, Set Map, Set Shade, Shade

## Функция Map3dInfo( )

### Назначение

Возвращает свойства окна 3DКарты.

### Синтаксис

**Map3DInfo**( *window\_id* , *attribute* )

*window\_id* - это целочисленный идентификатор окна

*attribute* - это целочисленный код, указывающий, какой тип информации должен быть возвращен.

### Возвращаемые величины

Вещественное, Логическое или Строковое значение, в зависимости от атрибута параметра.

### Описание

Функция **Map3DInfo()** возвращает информацию об окне 3DКарты.

Параметр *window\_id* определяет какое окно 3DКарты опрашивается. Для получения идентификатора окна, вызовите функцию **FrontWindow()** немедленно после открытия окна, или вызовите функцию **WindowID()** в любое время после создания окна.

Есть несколько числовых атрибутов, которые **Map3DInfo()** может вернуть для любого окна 3DКарты. Атрибуты параметра сообщают функции **Map3DInfo()** какая статистика окна Карты возвращается. Атрибут параметра должен принимать одно из следующих значений в таблице; коды определены в MAPBASIC.DEF.

Атрибуты	Возвращаемая величина
MAP3D_INFO_SCALE	Вещественное, масштабный фактор 3DКарты.
MAP3D_INFO_RESOLUTION_X	Целое, разрешение по X грида в окне 3DКарты.
MAP3D_INFO_RESOLUTION_Y	Целое, разрешение по Y грида в окне 3DКарты.
MAP3D_INFO_BACKGROUND	Целое, цвет фона, см. функцию RGB.
MAP3D_INFO_UNITS	Строка, представляющая сокращение единиц измерения площади, »mi» для миль.
MAP3D_INFO_LIGHT_X	Вещественное, координата X источника света.
MAP3D_INFO_LIGHT_Y	Вещественное, координата Y источника света.
MAP3D_INFO_LIGHT_Z	Вещественное, координата Z источника света.
MAP3D_INFO_LIGHT_COLOR	Целое, цвет источника, см. функцию RGB.
MAP3D_INFO_CAMERA_X	Вещественное, координата X камеры.
MAP3D_INFO_CAMERA_Y	Вещественное, координата Y камеры.
MAP3D_INFO_CAMERA_Z	Вещественное, координата Z камеры.
MAP3D_INFO_CAMERA_FOCAL_X	Вещественное, координата X фокальной точки камеры.

## Функция Map3dInfo( )

---

MAP3D_INFO_CAMERA_FOCAL_Y	Вещественное, координата Y фокальной точки камеры.
MAP3D_INFO_CAMERA_FOCAL_Z	Вещественное, координата Z фокальной точки камеры.
MAP3D_INFO_CAMERA_VU_1	Вещественное, первое значение нормального вектора точки просмотра.
MAP3D_INFO_CAMERA_VU_2	Вещественное, второе значение нормального вектора точки просмотра.
MAP3D_INFO_CAMERA_VU_3	Вещественное, третье значение нормального вектора точки просмотра.
MAP3D_INFO_CAMERA_VPN_1	Вещественное, первое значение нормального вектора плоскости просмотра.
MAP3D_INFO_CAMERA_VPN_2	Вещественное, второе значение нормального вектора плоскости просмотра.
MAP3D_INFO_CAMERA_VPN_3	Вещественное, третье значение нормального вектора плоскости просмотра.
MAP3D_INFO_CAMERA_CLIP_NEAR	Вещественное, приближение камеры.
MAP3D_INFO_CAMERA_CLIP_FAR	Вещественное, удаление камеры.

### Примеры

Распечатка штатных переменных определенных для окна 3DКарты:

```
include »Mapbasic.def»
Print »MAP3D_INFO_SCALE: » + Map3DInfo(FrontWindow(), MAP3D_INFO_SCALE)
Print »MAP3D_INFO_RESOLUTION_X: » + Map3DInfo(FrontWindow(),
MAP3D_INFO_RESOLUTION_X)
Print »MAP3D_INFO_RESOLUTION_Y: » + Map3DInfo(FrontWindow(),
MAP3D_INFO_RESOLUTION_Y)
Print »MAP3D_INFO_BACKGROUND: » + Map3DInfo(FrontWindow(),
MAP3D_INFO_BACKGROUND)
Print »MAP3D_INFO_UNITS: » + Map3DInfo(FrontWindow(), MAP3D_INFO_UNITS)
Print »MAP3D_INFO_LIGHT_X : » + Map3DInfo(FrontWindow(),
MAP3D_INFO_LIGHT_X )
Print »MAP3D_INFO_LIGHT_Y : » + Map3DInfo(FrontWindow(),
MAP3D_INFO_LIGHT_Y )
Print »MAP3D_INFO_LIGHT_Z: » + Map3DInfo(FrontWindow(),
MAP3D_INFO_LIGHT_Z)
Print »MAP3D_INFO_LIGHT_COLOR: » + Map3DInfo(FrontWindow(),
MAP3D_INFO_LIGHT_COLOR)
```

```
Print »MAP3D_INFO_CAMERA_X: » + Map3DInfo(FrontWindow(),
MAP3D_INFO_CAMERA_X)
Print »MAP3D_INFO_CAMERA_Y : » + Map3DInfo(FrontWindow(),
MAP3D_INFO_CAMERA_Y )
Print »MAP3D_INFO_CAMERA_Z : » + Map3DInfo(FrontWindow(),
MAP3D_INFO_CAMERA_Z )
Print »MAP3D_INFO_CAMERA_FOCAL_X: » + Map3DInfo(FrontWindow(),
MAP3D_INFO_CAMERA_FOCAL_X)
Print »MAP3D_INFO_CAMERA_FOCAL_Y: » + Map3DInfo(FrontWindow(),
MAP3D_INFO_CAMERA_FOCAL_Y)
Print »MAP3D_INFO_CAMERA_FOCAL_Z: » + Map3DInfo(FrontWindow(),
MAP3D_INFO_CAMERA_FOCAL_Z)
Print »MAP3D_INFO_CAMERA_VU_1: » + Map3DInfo(FrontWindow(),
MAP3D_INFO_CAMERA_VU_1)
Print »MAP3D_INFO_CAMERA_VU_2: » + Map3DInfo(FrontWindow(),
MAP3D_INFO_CAMERA_VU_2)
Print »MAP3D_INFO_CAMERA_VU_3: » + Map3DInfo(FrontWindow(),
MAP3D_INFO_CAMERA_VU_3)
Print »MAP3D_INFO_CAMERA_VPN_1: » + Map3DInfo(FrontWindow(),
MAP3D_INFO_CAMERA_VPN_1)
Print »MAP3D_INFO_CAMERA_VPN_2: » + Map3DInfo(FrontWindow(),
MAP3D_INFO_CAMERA_VPN_2)
Print »MAP3D_INFO_CAMERA_VPN_3: » + Map3DInfo(FrontWindow(),
MAP3D_INFO_CAMERA_VPN_3)
Print »MAP3D_INFO_CAMERA_CLIP_NEAR: » + Map3DInfo(FrontWindow(),
MAP3D_INFO_CAMERA_CLIP_NEAR)
Print »MAP3D_INFO_CAMERA_CLIP_FAR: » + Map3DInfo(FrontWindow(),
MAP3D_INFO_CAMERA_CLIP_FAR)
```

### Смотрите также

Оператор Create 3dMap, оператор Set 3dMap

### Функция MapperInfo( )

#### Назначение:

Возвращает информацию о координатах или расстояниях в окне Карты.

#### Синтаксис:

**MapperInfo**(*window\_id*, *attribute*)

где

*window\_id* – идентификатор окна Карты;

*attribute* – целочисленный код.

#### Величина, полученная в результате:

Величина типа Float, Logical или String, в зависимости от значения кода в параметре *attribute*.

#### Описание:

Функция **MapperInfo( )** возвращает информацию об окне Карты.

Параметр *window\_id* задает идентификатор окна Карты. Значения идентификатора можно получить, используя функцию **FrontWindow( )** сразу после открытия окна Карты, или в другом случае при помощи функции **WindowID( )**.

Параметр *attribute* задает тип информации, которая будет получена в результате. Значение параметра должно быть целочисленным кодом. В следующей таблице в первой колонке приводятся имена кодов, которые установлены в файле стандартных определений MapBasic MAPBASIC.DEF.

Значения <i>attribute</i>	Результат <b>MapperInfo( )</b>
MAPPER_INFO_AREAUNITS	Строковая величина с именем единицы измерения площади (например, "sq mi" – квадратные мили).
MAPPER_INFO_CENTERX	X-координата центральной точки окна.
MAPPER_INFO_CENTERY	Y-координата центральной точки окна.
MAPPER_INFO_COORDSYS_CLAUSE	Строка, соответствующая установке предложения CoordSys для этого окна.
MAPPER_INFO_COORDSYS_NAME	Строка с именем координатной системы карты такая, как она обозначена в файле MAPINFOW.PRJ (но без суффикса "\r...", который можно видеть в файле MAPINFOW.PRJ). Возвращает пустую строку, если значение CoordSys не найдено в файле MAPINFOW.PRJ.
MAPPER_INFO_DISPLAY	Целое число типа SmallInt, соответствующее типу информации, которая показывается в строке сообщений окна Карты. Соответствует установке оператора Set Map Display. Результатом может быть один из следующих кодов: MAPPER_INFO_DISPLAY_SCALE MAPPER_INFO_DISPLAY_ZOOM MAPPER_INFO_DISPLAY_POSITION
MAPPER_INFO_DISTUNITS	Имя единицы измерения расстояния (например, "mi").
MAPPER_INFO_EDIT_LAYER	Целое число типа SmallInt, являющееся номером изменяемого слоя. Ноль, если объекты изменяются в Косметическом слое, единица, если изменяемый слой первый некосметический, и т. д. Если результатом будет минус единица, то ни один слой не находится в изменяемом состоянии.
MAPPER_INFO_LAYERS	Число слоев на Карте, включая Косметический (число типа SmallInt).
MAPPER_INFO_MINX	Минимальная X-координата части Карты, показанной в окне.
MAPPER_INFO_MINY	Минимальная Y-координата части Карты, показанной в окне.
MAPPER_INFO_MAXX	Максимальная X-координата части Карты, показанной в окне.
MAPPER_INFO_MAXY	Максимальная Y-координата части Карты, показанной в окне.
MAPPER_INFO_NUM_THEMATIC	Короткое целое число, номер слоя, который является тематическим.

## Функция MapperInfo( )

---

MAPPER_INFO_SCALE	Текущий масштаб; количество единиц измерения расстояния Карты (например, километров), помещающееся в одной "бумажной" единице (например, в сантиметре). Значение возвращается в единицах согласно текущим установкам в MapBasic.
MAPPER_INFO_SCROLLBARS	Логическая величина, показывающая, есть ли в окне Карты полосы прокрутки.
MAPPER_INFO_XYUNITS	Строка, представляющая название единиц измерения координат карты, например, "градусы". Короткое целое, определяющее, отображаются ли координаты карты в десятичных градусах, градусах/минутах/секундах или в формате "Армейская система США". Возвращаемое значение может быть одним из: MAPPER_INFO_DISPLAY_DECIMAL, MAPPER_INFO_DISPLAY_DMS или MAPPER_INFO_DISPLAY_MGRS
MAPPER_INFO_COORDSYS_CLAUSE_WITH_BOUNDS	Строковая величина, указывающая предложение CoordSys включая ограничивающий данную систему координат прямоугольник.
MAPPER_INFO_MOVE_DUPLICATE_NODES	Короткое целое, указывающее надо ли удалять дублирующиеся узлы в режиме Форма окна Карты. Если значение 0, дублирующиеся узлы не удаляются. Если значение 1, все дублирующиеся узлы на этом слое будут удалены. Атрибут.
MAPPER_INFO_DIST_CALC_TYPE	Короткое целое, указывающее тип алгоритма вычисления расстояния, длины, периметра и площади. Соответствует Set Map Distance Type. Возвращаемые значения включают MAPPER_INFO_DIST_SPHERICAL MAPPER_INFO_DIST_CARTESIAN
MAPPER_INFO_CLIP_REGION	Возвращает строку, определяющую, используется ли регион для отсечения части карты. Возвращает »on« если регион отсечения применяется. В других случаях, возвращает »off«.
MAPPER_INFO_CLIP_TYPE	Тип отсечения, применяемый к карте. Варианты включают: MAPPER_INFO_CLIP_DISPLAY_ALL MAPPER_INFO_CLIP_DISPLAY_POLYOBJ MAPPER_INFO_CLIP_OVERLAY
MAPPER_INFO_ZOOM	Размер показанной части Карты (расстояние от Западного до Восточного края) в единицах измерения расстояния, установленных в MapBasic (смотрите описание оператора <b>Set Distance Units</b> ).

Когда вызывается **MapperInfo( )** для получения значений координат (с указанием



MAPPER\_INFO\_CENTERX в качестве *attribute*), возвращаемое значение будет координатами в текущей для MapBasic системе координат, которая может отличаться от системы координат в окне Карты. Используйте оператор **Set CoordSys** для задания другой системы координат.

Настройки для окна Карты и обеспечение поддержки MapBasic можно делать для каждого окна Карты.

Когда создано новое окно Карты, можно настроить режим *Совмещения при перемещении* (Настройки / Режимы / Окно Карты / Совмещать при перемещении).

Существующее окно Карты может быть опрошено на предмет параметров режима *Совмещения при перемещении* использованием новых атрибутов функции MapperInfo( ).

Текущие настройки могут изменяться оператором Set Map.

Информация о врезке региона

Начиная с MapInfo Professional 6.0, существуют 3 метода, применяющиеся для создания врезки.

Метод **MAPPER\_INFO\_CLIP\_OVERLAY** был единственным до версии MI Pro 6.0. Используя этот метод, функция **Overlap()** используется так, что она определяется системой. Функция Overlap() не обрезает текстовые объекты, а подписи и точечные объекты обрезаются только в том случае если они целиком выходят за область врезки.

Метод **MAPPER\_INFO\_DISPLAY\_ALL** определяется системой и обрезает все типы объектов, тематика, растры и сетки (гриды) тоже обрезаются. Стили (ширина линий, символы, текст) всегда обрезаются. Это стандартный метод обрезания.

Метод **MAPPER\_INFO\_CLIP\_DISPLAY\_POLYOBJ** используется для воспроизведения возможностей метода **MAPPER\_INFO\_CLIP\_OVERLAY**. Обрезаются все поли- объекты (регионы и полилинии) и те объекты, которые могут быть превращены в поли- объекты (прямоугольники, скругленные прямоугольники, эллипсы и дуги). Такие объекты всегда обрезаются. Точки, Подписи и текст не обрезаются.

Для использования имен кодов, а не их целочисленных значений, необходимо включить в текст Вашей программы оператор **Include "MAPBASIC.DEF"**.

Заметим, что если Вы используете функцию **MapperInfo( )** для получения координат центра окна Карты, то возвращенное значение будет представлено в текущей координатной системе MapBasic. По умолчанию MapBasic использует систему координат широта/долгота. Но Вы можете задать другую оператором **Set CoordSys**.

### Ошибки:

ERR-BAD-WINDOW, если нет такого окна;

ERR-FCN-ARG-RANGE, если аргумент выходит за допустимые пределы;

ERR-WANT-MAPPER-WIN, если окно не Карта.

### Смотрите также:

**LayerInfo( ), Set Distance Units, Set Map**

### Функция **Maximum( )**

#### Назначение:

Возвращает наибольшее из двух заданных чисел.

#### Синтаксис:

**Maximum**(*num\_expr*, *num\_expr*)

где

*num\_expr* – численное выражение.

#### Величина, полученная в результате:

Вещественное число. Величина типа Float.

#### Описание:

Функция **Maximum( )** возвращает наибольшее из двух чисел, заданных численными выражениями *num\_expr*.

#### Пример:

```
Dim x, y, z As Float
x = 42
y = 27
z = Maximum(x, y)

' z равно 42
```

#### Смотрите также:

**Minimum( )**

## Функция MBR( )

### Назначение:

Возвращает прямоугольный объект, представляющий минимальное прямоугольное покрытие заданного объекта.

### Синтаксис:

**MBR**(*obj\_expr*)

где

*obj\_expr* – объектное выражение.

### Величина, полученная в результате:

Величина типа Object. Графический объект типа "прямоугольник".

### Описание:

Функция **MBR( )** возвращает графический объект – наименьший прямоугольник, в который можно вписать объект, заданный выражением *obj\_expr*.

Такой прямоугольник представляет минимальное прямоугольное покрытие объекта. Например, минимальное прямоугольное покрытие США представляет собой прямоугольник, у которого правая сторона включает в себя самую западную точку границы штата Мен, нижняя сторона – самую южную точку границы Гавайи, и левая и верхняя стороны – самую восточную и самую северную точки границы штата Аляска.

Минимальное прямоугольное покрытие точечного объекта имеет нулевую ширину и нулевую высоту.

### Пример:

```
Dim o-mbr As Object
Open Table "world"
Fetch First From world
o-mbr = MBR(world.obj)
```

### Смотрите также:

**Centroid( ), CentroidX( ), CentroidY( )**

### Оператор Menu Bar

#### Назначение:

Показывает или скрывает строку меню.

#### Синтаксис:

**Menu Bar { Hide | Show }**

#### Описание:

Оператор **Menu Bar** управляет отображением строки меню в рабочем окне MapInfo. Программа, используя этот оператор, может освободить больше места на экране для окна Карты, Списка, Отчета или Графика.

Чтобы вновь показать строку меню, скрытую оператором **Menu Bar Hide**, используйте оператор **Menu Bar Show**. Вам следует аккуратно использовать этот оператор, так как пользователь может быть поставлен в тупик, оказавшись без строки меню. За каждым оператором **Menu Bar Hide**, по возможности, должен следовать оператор **Menu Bar Show**.

Пока строка меню отсутствует, MapInfo будет игнорировать клавишные сокращения для вызова команд. Т.е., например, для вызова диалога команды ФАЙЛ > ОТКРЫТЬ Вы можете использовать клавиши CTRL+O, но, если строка меню скрыта, то нажатие на эти клавиши ни к чему не приведет.

Заметим, что в MapInfo для Macintosh оператор **Menu Bar Hide** не может скрыть меню.

#### Смотрите также:

**Alter Menu Bar, Create Menu Bar**

### Функция MenuItemInfoByHandler( )

#### Назначение:

Возвращает информацию об элементе меню MapInfo.

#### Синтаксис:

**MenuItemInfoByHandler(handler , attribute )**

где

*handler* – либо строка с именем процедуры-обработчика, заданной для элемента меню предложением

**Calling**, или целое число (тип Integer), код, который был задан в предложении **Calling**;

*attribute* – целое число типа Integer, код, задающий, какая информация необходима в результате.

#### Величина, полученная в результате:

Тип величины зависит от значения параметра *attribute*.

#### Описание:

Параметр *handler* может быть как строковым, так и численным. Если Вы выбрали строковый вид (имя процедуры), и соответствующую процедуру вызывают два или более элемента меню, то MapInfo будет рассматривать первый элемент, вызвавший эту процедуру. Поэтому, если Вам необходима информация о другом элементе, то используйте для идентификации ID-номер, который был назначен элементу меню в операторе **Create Menu** и используйте функцию **MenuItemInfoByID( )** вместо **MenuItemInfoByHandler( )**.

Значение параметра *attribute* должно быть целочисленным кодом. В следующей таблице в первой колонке приводятся имена кодов, установленных в файле стандартных определений MapBasic MAP-BASIC.DEF.

## Функция MenuItemInfoByHandler( )

---

### Значения *attribute*

MENUIITEM\_INFO\_ACCELERATOR

MENUIITEM\_INFO\_CHECKABLE

MENUIITEM\_INFO\_CHECKED

MENUIITEM\_INFO\_ENABLED

MENUIITEM\_INFO\_HANDLER

MENUIITEM\_INFO\_HELPMSG

MENUIITEM\_INFO\_ID

### Результат

Строка, величина типа String:  
строковый код акселератора элемента меню (например, “/W^Z” или “/W#%119”) или пустая строка, акселератор не был назначен. Информацию о назначении элементу меню акселератора смотрите в описании оператора **Create Menu**.

Логическая величина:  
“Да” (TRUE), если элемент меню фиксируется (рядом с именем элемента в меню может появляться галочка)

Логическая величина:  
“Да” (TRUE), если элемент меню можно фиксировать и в данный момент он фиксирован (есть галочка);  
“Да” (TRUE) также, если элемент меню имеет несколько вариантов текста (например, “Показать что-то” и “Скрыть что-то”) и при этом элемент меню находится в состоянии “Показать”;  
“Нет” (FALSE) во всех остальных случаях.

Логическая величина:  
“Да” (TRUE), если элемент меню активен.

Целое число типа Integer:  
Номер обработчика элемента меню. Если при создании элемента меню в предложении **Calling** был задан код (например, **Calling M\_FILE\_SAVE**), то результатом будет значение этого кода. Если предложение **Calling** задавало “OLE”, “DDE” или имя процедуры, то результатом будет уникальное целое число, которое может быть использовано функцией **MenuItemInfoByHandler( )** и оператором **Run Menu Command**.

Строка, величина типа String:  
подсказка для элемента меню, которая была назначена в предложении **HelpMsg** оператора **Create Menu** или пустая строка, если подсказка не назначалась.

Целое число типа Integer:  
идентификатор элемента меню, который был назначен предложением **ID** в операторе **Create Menu** или 0, если элемент меню не имеет идентификатора.

MENUIITEM_INFO_SHOWHIDEABLE	Логическая величина: “Да” (TRUE), если элемент меню имеет несколько вариантов текста (например, “Показать что-то” и “Скрыть что-то”). Несколько вариантов текста задаются помещением символа “!” в начало строки описания элемента меню (в операторах <b>Create Menu</b> или <b>Alter Menu</b> ) и символа “^” перед началом альтернативного текста.
MENUIITEM_INFO_TEXT	Строка, величина типа String: полный текст, используемый при создании элемента меню (например, в операторе <b>Create Menu</b> ).

### Смотрите также:

**MenuItemInfoByID( ), Create Menu, Alter Menu**

### Функция MenuItemInfoByID( )

#### Назначение:

Возвращает информацию об элементе меню MapInfo.

#### Синтаксис:

**MenuItemInfoByID**(*menuItem\_ID*, *attribute* )

*menuItem\_ID* – целое число типа Integer, идентификатор элемента меню, который он получил при создании в предложении **ID** оператора **Create Menu**;

*attribute* – целое число типа Integer, код, задающий, какую информацию необходимо вернуть.

#### Величина, полученная в результате:

Тип величины зависит от значения параметра *attribute*.

#### Описание:

Функция работает аналогично функции **MenuItemInfoByHandler( )**. Различие состоит в том, как задается элемент меню в первом параметре. В функции **MenuItemInfoByID( )** используется его идентификатор. В функции **MenuItemInfoByHandler( )** элемент меню задается обработчиком, который запускается, когда пользователь выберет элемент меню.

Параметр *attribute* должен быть целочисленным кодом, одним из тех, имена которым присвоены в файле MAPBASIC.DEF (например, MENUITEM\_INFO\_CHECKED). Список возможных значений параметра *attribute* и какой результат должен быть получен в результате тот же, что и для функции **MenuItemInfoByHandler( )** (смотрите описание функции выше).

#### Смотрите также:

**MenuItemInfoByHandler( )**



## Оператор Metadata

### Назначение:

Управление метаданными таблицы.

### Синтаксис (вариант 1):

```
Metadata Table table_name
{ SetKey key_name To key_value |
  DropKey key_name [ Hierarchical ] |
  SetTraverse starting_key_name [ Hierarchical ] Into ID traverse_ID_var }
```

где

*table\_name* – имя открытой таблицы;

*key\_name* – строка, представляющая имя ключа метаданных (должна начинаться с обратного слэша (“\”) и не должна заканчиваться обратным слэшем);

*key\_value* – строка до 239 символов длиной, значение, присваиваемое ключу;

*starting\_key\_name* – строка, представляющая первое имя ключа для извлечения соответствующего значения из таблицы. Чтобы начать структурное извлечение с самого начала списка ключей, добавьте “\” (обратный слэш);

*traverse\_ID\_var* – имя переменной типа Integer. С помощью этой переменной MapInfo управляет последовательными операторами Metadata Traverse.

### Синтаксис (вариант 2):

```
Metadata Traverse traverse_ID
{ Next Into Key key_name_var Into Value key_value_var |
  Destroy }
```

где

*traverse\_ID* – целое число типа Integer (такое как значение переменной *traverse\_ID\_var* из предыдущего варианта синтаксиса оператора);

*key\_name\_var* – имя строковой переменной (MapInfo помещает в эту переменную название ключа для извлечения);

*key\_value\_var* – имя строковой переменной (MapInfo помещает в эту переменную извлекаемое значение).

### Описание:

Оператор **Metadata** управляет метаданными, размещаемыми в таблице MapInfo 4.0. Metadata – это информация, размещаемая в файле таблицы (.TAB), а не в строках и столбцах файла данных.

Каждая таблица может иметь ключи метаданных. Каждый ключ определяет категорию хранимой информации, такой как имя автора, его права и т.д. Каждому ключу соответствует некое значение. Например, ключу “\Copyright” может соответствовать значение “Copyright 1995 MapInfo Corporation.” Более подробно метаданные описываются в главе 7 *Руководства пользователя MapBasic*.

### Изменение метаданных в таблице

Чтобы создать, поменять или удалить метаданные, используйте Синтаксис 1. Если Вы создаете метаданные, то они записываются немедленно; не нужно проводить операцию сохранения. При этом используются следующие предложения:

#### SetKey

Присваивает значение ключу. Если ключ уже существует, то MapInfo присваивает ему новое значение. Если ключ не существует, MapInfo создает его.

## Оператор Metadata

---

```
Metadata Table Parcels SetKey "\Info\Date" To Str$(CurDate())
```

**Внимание:** MapInfo автоматически создает ключ метаданных “\IsReadOnly” (со стандартным значением “FALSE”) в первый раз, когда Вы добавляете метаданные в таблицу. Ключ “\IsReadOnly” – это специальный ключ, который MapInfo использует для своих нужд.

### DropKey

Удаляет ключ из таблицы. Если добавить слово **Hierarchical**, MapInfo удаляет вместе с ключом всю структуру метаданных, подчиненных этому ключу. Например, если в таблице есть ключи “\Info\Author” и “\Info\Date”, то оба они удаляются следующим оператором:

```
Metadata Table Parcels DropKey "\Info" Hierarchical
```

### Чтение метаданных из таблицы

Чтобы прочитаты метаданные из таблицы, используется предложение **SetTraverse**, которое инициализирует операцию структурного извлечения, продолжаемую затем одноразовыми извлечениями значений предложением **Next**. Для завершения операции извлечения применяется предложение **Destroy**, которое освобождает память, используемую при этой операции. Более подробно о предложениях этого оператора:

#### SetTraverse

Подготавливает операцию структурного извлечения метаданных, начиная с определенного ключа. Чтобы начать извлечение с самого первого в иерархии ключа, задайте в качестве имени начального ключа “\”. Если будет добавлено слово **Hierarchical**, то операция извлечения пройдет по всей структуре. Если слово **Hierarchical** опущено, то операция извлечения не будет опускаться на уровни ниже начального (т.е. извлечение, стартовавшее с ключа “\Info”, не затронет ключ “\Info\Date”).

#### Next Into Key ... Into Value ...

Попытка чтения следующего ключа. Если еще остаются ключи для чтения, то MapInfo помещает ключ в переменную *key\_name\_var* и значение, соответствующее ключу, в переменную *key\_value\_var*. Если ключи исчерпаны, MapInfo помещает в обе переменные пустые значения.

#### Destroy

Завершает операцию структурного извлечения и освобождает занятую под нее память.

**Внимание:** Операция структурного извлечения может проникать до десятого уровня иерархии (т.е. “\Один\Два\Три\Четыре\Пять\Шесть\Семь\Восемь\Девять\Десять”), начиная с нулевого или корневого (“\”). Если Вы хотите проникнуть глубже, чем на 10 уровней, то начинайте операцию структурного извлечения с ненулевого уровня (например, с уровня “\Один\Два\Три\Четыре\Пять”).

### Пример:

Следующая процедура извлекает все метаданные из таблицы; имя таблицы определяется вызывающей программой. Все ключи и соответствующие им значения распечатываются в окне Сообщения.

```
Sub Print_Metadata(ByVal table_name As String)
    Dim i_traversal As Integer
    Dim s_keyname, s_keyvalue As String

    ' Инициализация операции извлечения:
    Metadata Table table_name
        SetTraverse "\ " Hierarchical Into ID i_traversal
```

```

' Попытка извлечь значение по первому ключу:
Metadata Traverse i_traversal
  Next Into Key s_keyname Into Value s_keyvalue

' Теперь в цикле извлекаются все значения по одному
' до тех пор, пока есть непустые ключи,
' и они распечатываются в окне Сообщений.
Do While s_keyname <> ""
  Print " "
  Print "Key name: " & s_keyname
  Print "Key value: " & s_keyvalue

  Metadata Traverse i_traversal
    Next Into Key s_keyname Into Value s_keyvalue
Loop

' Освобождение памяти, занятой под операцию извлечения:
MetaData Traverse i_traversal Destroy

End Sub

```

**Смотрите также:**

`GetMetadata$( ), TableInfo( )`

### Функция Mid\$( )

#### Назначение:

Возвращает строку, извлекая ее из середины другой.

#### Синтаксис:

**Mid\$(string\_expr, position, length)**

где

*string\_expr* – выражение, результат которого есть строка;

*position* – целочисленное выражение, результат которого есть номер первого символа, извлекаемого из строки;

*length* – целочисленное выражение, результат которого есть количество извлекаемых символов.

#### Величина, полученная в результате:

Строка. Величина типа String.

#### Описание:

Функция **Mid\$( )** возвращает подстроку из строки, заданной выражением *string\_expr*.

Функция копирует *length* символов из *string\_expr*, начиная с символа, номер которого определен параметром *position*. Если значение этого параметра меньше или равно единице, то копирование будет производиться с самого начала строки *string\_expr*.

Если длина строки *string\_expr* меньше, чем заданный параметр *length*, то функция вернет укороченную строку. Если параметр *position* задает позицию за пределами строки *string\_expr*, то функция возвратит пустую строку. Такой же результат Вы получите, если зададите параметр длины *length* меньше единицы.

#### Пример:

```
Dim str-var, substr-var As String
str-var = "New York City"
substr-var = Mid$(str-var, 10, 4)
'
' substr-var теперь равна "City"
'
```

#### Смотрите также:

**InStr( ), Left( ), Right( )**

## Функция MidByte\$( )

### Назначение:

Позволяет извлекать байты из строки, состоящей из двухбайтовых символов (например, Windows Japanese).

### Синтаксис:

**MidByte\$(string\_expr, position, length )**

где

*string\_expr* – строковое выражение;

*position* – целочисленное выражение, результат которого есть номер первого символа, извлекаемого из строки;

*length* – целочисленное выражение, задающее количество извлекаемых байт.

### Величина, полученная в результате:

Строка. Величина типа String.

### Описание:

Функция **MidByte\$( )** возвращает некоторое количество байт из строки, заданной выражением *string\_expr*.

Функция **MidByte\$( )** используется для извлечения нескольких байтов из строки, образованной символами двухбайтовой кодировки. Двухбайтовая кодировка символов (DBCS) используется, например, в японской версии Windows.

В системах с однобайтовыми наборами символов функция **MidByte\$( )** возвращает те же результаты, что и функция **Mid\$( )**.

### Смотрите также:

**InStr( ), Left\$( ), Right\$( )**

### Функция **Minimum( )**

#### Назначение:

Возвращает наименьшее из двух заданных чисел.

#### Синтаксис:

**Minimum**(*num\_expr*, *num\_expr* )

где

*num\_expr* – численное выражение.

#### Величина, полученная в результате:

Вещественное число. Величина типа Float.

#### Описание:

Функция **Minimum( )** возвращает наименьшее из двух чисел, заданных численными выражениями *num\_expr*.

#### Пример:

```
Dim x, y, z As Float
x = 42
y = -100
z = Minimum(x, y)

' z равно -100
```

#### Смотрите также:

**Maximum( )**

## Функция Month( )

### Назначение:

Возвращает из даты компоненту, соответствующую номеру месяца в году (1 – 12).

### Синтаксис:

**Month**(*date\_expr*)

где

*date\_expr* – выражение, результатом которого является величина типа Date.

### Величина, полученная в результате:

Короткое целое число от 1 до 12, включительно. Величина типа SmallInt.

### Описание:

Функция **Month( )** возвращает целое число, являющееся номером месяца в дате *date\_expr*.

### Примеры:

Определим, какой сейчас месяц:

```
If Month(CurDate( )) = 12 Then  
    Note "Это Декабрь"  
End If
```

Функцию **Month( )** можно использовать в SQL-запросе. Следующий оператор **Select** выбирает из таблицы ORDERS только строки со значениями в колонке "Order-Date" (тип Date), относящиеся к декабрю 1993.

```
Open Table "orders"  
Select *  
From orders  
Where Month(order-date) = 12 And Year(order-date) = 1993
```

### Смотрите также:

**CurDate( ), Day( ), Weekday( ), Year( )**

### Оператор Note

#### Назначение:

Показывает сообщение в простом диалоговом окне.

#### Синтаксис:

**Note** *message*

где

*message* – выражение, результат которого будет показан в окне.

#### Описание:

Оператор **Note** создает простое диалоговое окно сообщений, сопровождающееся одной кнопкой "ОК".

Параметр *message* может быть выражением, не обязательно строковым. Если в результате вычисления выражения *message* получается величина объектного типа (Object), MapBasic автоматически преобразует его в строку (так как это делает функция **Str\$( )**). Это строковое представление и будет выведено в диалоговом окне сообщений. Если *message* – строка, то максимальная длина должна быть не более 300 символов и может занимать только 6 строк.

#### Пример:

```
Note "Всего использовано записей: " + Str$( i-count )
```

#### Смотрите также:

**Ask( ), Dialog, Print**



## Функция NumAllWindows( )

### Назначение:

Возвращает количество окон, открытых MapInfo, включая специальные окна, такие как инструментальные панели и окно Информации.

### Синтаксис:

**NumAllWindows( )**

### Величина, полученная в результате:

Короткое целое число. Величина типа SmallInt.

### Описание:

Функция **NumAllWindows( )** определяет, сколько всего открыто окон программой MapInfo.

Чтобы определить количество открытых “документальных” окон MapInfo (Карт, Списков, Графиков и Отчетов), используйте функцию **NumWindows( )**.

### Смотрите также:

**NumWindows( )**, **WindowID( )**

### Функция **NumberToDate( )**

#### Назначение:

Возвращает величину типа Date, созданную из величины типа Integer.

#### Синтаксис:

**NumberToDate**(*numeric\_date* )

где *numeric\_date* – восьмизначное целое число типа Integer в форме ГГГГММДД (например, 19951231).

#### Величина, полученная в результате:

Величина типа Date.

#### Описание:

Функция **NumberToDate( )** возвращает дату, величину типа Date, используя восьмизначное целое число. Например, следующая функция будет иметь результат, равный 31 декабря 1995:

```
NumberToDate(19951231)
```

#### Пример:

```
Dim i_elapsed As Integer

i_elapsed = CurDate() - NumberToDate(19950101)

' i_elapsed теперь равен числу дней, прошедших
' с 1 января 1995
```

#### Смотрите также:

**StringToDate( )**

## Функция NumCols( )

### Назначение:

Возвращает число колонок таблицы.

### Синтаксис:

**NumCols**(*table*)

где

*table* – имя открытой таблицы.

### Величина, полученная в результате:

Короткое целое число. Величина типа SmallInt.

### Описание:

Функция **NumCols( )** возвращает число колонок, из которых состоит открытая таблица *table*.

В это число не входит специальная колонка "Object" (или "Obj" сокращенно), содержащая ссылки на графические объекты, присоединенные к таблице. Также в число колонок не включается другая специальная колонка **RowID**, содержащая номера строк таблицы.

Замечание: если таблица временная (например, полученная после оператора **Add Column**), то число колонок, полученное от функции **NumCols( )** будет включать временную или временные колонки.

### Ошибки:

В результате выполнения функции может генерироваться код ошибки:

ERR-TABLE-NOT-FOUND, если таблица недоступна.

### Пример:

```
Dim i-counter As Integer
Open Table "world"
i-counter = NumCols(world)
' i-counter содержит теперь число колонок WORLD
```

### Смотрите также:

**ColumnInfo( )**, **NumTables( )**, **TableInfo( )**

### Функция NumTables( )

#### Назначение:

Возвращает число открытых на данный момент таблиц.

#### Синтаксис:

**NumTables( )**

#### Величина, полученная в результате:

Короткое целое число. Величина типа SmallInt.

#### Описание:

Функция **NumTables( )** возвращает число открытых на данный момент таблиц.

Если в MapInfo открыта таблица, содержащая Карту улиц (StreetInfo), то на самом деле открыты две связанные таблицы. Например, когда Вы открываете таблицу DCWASHS (карта улиц Вашингтона), MapInfo открывает две составляющие таблицы DCWASHS1.TAB и DCWASHS2.TAB. Тем не менее MapInfo считает DCWASHS одной таблицей, поскольку составляющие таблицы являются частями одной Карты. Так же и функция **NumTables( )** таблицу, содержащую Карту улиц, будет считать одной открытой таблицей, несмотря на то, что практически она состоит из двух.

#### Пример:

```
If Numtables( ) < 1 Then
    Note "Нет открытых таблиц. Продолжение невозможно."
End Program
End If
```

#### Смотрите также:

**Open Table, TableInfo( ), ColumnInfo( )**

## Функция NumWindows( )

### Назначение:

Возвращает количество открытых на данный момент окон (Карт, Списков, Графиков и Отчетов).

### Синтаксис:

**NumWindows( )**

### Величина, полученная в результате:

Короткое целое число. Величина типа SmallInt.

### Описание:

Функция **NumWindows( )** возвращает число открытых на данный момент окон Карт, Списков, Графиков и Отчетов. Результат функции не зависит от того, в каком состоянии находится окно: свернуто в иконку или нет.

Чтобы определить общее количество выведенных на экран окон, включая вспомогательные (такие как окно Легенды, Информации), используется функция **NumAllWindows( )**.

### Пример:

```
Dim num-open-wins As SmallInt
num-open-wins = NumWindows( )
```

### Смотрите также:

**NumAllWindows( ), WindowID( )**

### Функция **ObjectGeography( )**

#### Назначение:

Возвращает информацию о графическом объекте, определяющую его расположение.

#### Синтаксис:

**ObjectGeography**(*object*, *attribute*)

где

*object* – объектное выражение;

*attribute* – целочисленный код, определяющий результат функции.

#### Величина, полученная в результате:

Вещественное число. Величина типа Float.

#### Описание:

Функция **ObjectGeography( )** возвращает информацию об определенных координатах и угловых величинах географических объектов. Координатами могут быть координаты точечного объекта или начальной точки прямой линии, минимальные или максимальные координаты объектов. А угловыми величинами – начальный и конечный углы дуги, угол поворота текста.

Параметр *attribute* должен принимать значения целочисленного кода, управляющего типом возвращаемой функцией информации. В следующей таблице в первой колонке приводятся имена кодов для функции **ObjectGeography( )**, которые установлены в файле стандартных определений MapBasic MAPBASIC.DEF.

Заметим, что некоторые значения для параметра *attribute* могут адресоваться объектам определенного типа. Например, начальный угол может быть считан только у объекта типа "дуга".

Значения <i>attribute</i>	Результат функции (Величина типа Float)
OBJ-GEO-MINX	Минимальная X-координата минимального прямоугольного покрытия объекта, если его тип не "линия". Иначе возвратится значение, равное OBJ_GEO_LINEBEGX.
OBJ-GEO-MINY	Минимальная Y-координата минимального прямоугольного покрытия объекта, если его тип не "линия". Иначе возвратится значение, равное OBJ_GEO_LINEBEGY.
OBJ-GEO-MAXX	Максимальная X-координата объекта или его минимального прямоугольного покрытия. Код не применим для объекта типа "точка". Для линий возвращается значение, равное OBJ_GEO_LINEENDX.
OBJ-GEO-MAXY	Максимальная Y-координата объекта или его минимального прямоугольного покрытия. Код не применим для объекта типа "точка". Для линий возвращается значение, равное OBJ_GEO_LINEENDY.
OBJ-GEO-ARCBEGANGLE	Начальный угол дуги. Только для объекта типа "дуга".
OBJ-GEO-ARCENDANGLE	Конечный угол дуги. Только для объекта типа "дуга".
OBJ-GEO-LINEBEGX	X-координата начальной точки прямой линии. Только для объекта типа "линия".
OBJ-GEO-LINEBEGY	Y-координата начальной точки прямой линии. Только для объекта типа "линия".
OBJ-GEO-LINEENDX	X-координата конечной точки прямой линии. Только для объекта типа "линия".
OBJ-GEO-LINEENDY	Y-координата конечной точки прямой линии. Только для объекта типа "линия".
OBJ-GEO-POINTX	X-координата точечного объекта.
OBJ-GEO-POINTY	Y-координата точечного объекта.
OBJ-GEO-ROUNDRADIUS	Диаметр окружности, которую можно вписать в закругление угла объекта типа "скругленный прямоугольник". Результат выдается в текущих координатных единицах (например, в градусах).
OBJ-GEO-TEXTLINEX	X-координата конца строки в текстовом объекте.
OBJ-GEO-TEXTLINEY	Y-координата конца строки в текстовом объекте.
OBJ-GEO-TEXTANGLE	Угол поворота текстового объекта.

## Функция **ObjectGeography( )**

---

Функция **ObjectGeography()** расширена поддержкой групп точек и коллекций. Оба типа поддерживают атрибуты 1 - 4 (координаты минимального описывающего объект прямоугольника (МОП)).

OBJ\_GEO\_MINX (1)    минимальная координата X МОП.

OBJ\_GEO\_MINY (2)    минимальная координата Y МОП.

OBJ\_GEO\_MAXX (3)    максимальная координата X МОП.

OBJ\_GEO\_MAXY (4)    максимальная координата Y МОП.

### Пример:

Здесь наряду с функцией **ObjectGeography( )** используется функция **ObjectInfo( )** для определения типа объекта и оператор **Set Map** для перемещения центра просмотра Карты в начальную точку прямой линии.

```
Include "MAPBASIC.DEF"
Dim i-obj-type As Integer, f-x, f-y As Float
Open Table "city"
Map From city
Fetch First From city
'
' В этом месте программы выражение city.obj
' представляет графический объект, присоединенный
' к первой строке таблицы CITY.
'

i-obj-type = ObjectInfo(city.obj, OBJ-INFO-TYPE)
If i-obj-type = OBJ-LINE Then
    f-x = ObjectGeography(city.obj, OBJ-GEO-LINEBEGX)
    f-y = ObjectGeography(city.obj, OBJ-GEO-LINEBEGY)
    Set Map Center (f-x, f-y)
End If
```

### Смотрите также:

**Centroid( ), CentroidX( ), CentroidY( ), ObjectInfo( )**



## Функция **ObjectInfo( )**

### Назначение:

Возвращает стиль линии, штриха и другие величины, описывающие графический объект, а также его тип.

### Синтаксис:

**ObjectInfo**(*object*, *attribute*)

где

*object* – выражение, результат которого есть величина типа Object;

*attribute* – целочисленный код, определяющий результат функции.

### Величина, полученная в результате:

Тип величины может быть SmallInt, Integer, String, Float, Pen, Brush, Symbol или Font в зависимости от значения параметра *attribute*.

### Описание:

Функция **ObjectInfo( )** возвращает основную информацию о графическом объекте, заданном параметром *object*. Объект может быть задан объектной переменной или выражением в формате *tablename.obj* (где *tablename* – имя таблицы, к которой присоединен объект).

Каждый объект обладает некоторыми атрибутами. Например, атрибутом является тип объекта – это область, линия, дуга и т. д. Объекты определенных типов могут обладать различными наборами других атрибутов – стилей. Ими являются величины типа Pen, Brush и др. Функция **ObjectInfo( )** может возвращать значения этих атрибутов. Возвращаемое значение зависит от того, каким задан параметр *attribute*.

Параметр *attribute* должен принимать значения целочисленного кода. В следующей таблице в первой колонке приводятся имена кодов для функции **ObjectGeography( )**, которые установлены оператором **Define** в файле стандартных определений MapBasic MAPBASIC.DEF. Для того, чтобы использовать имена кодов, Ваша программа в начале должна иметь оператор **Include "MAPBASIC.DEF"**.

Значения <i>attribute</i>	Результат функции
OBJ-INFO-TYPE	Целое число типа SmallInt, определяющее тип объекта (смотрите вторую таблицу).
OBJ-INFO-PEN	Величина типа Pen. Стилль линии для объектов типа "дуга", "эллипс", "линия", "полилиния", "рамка", "область", "прямоугольник" и "сглаженный прямоугольник".
OBJ-INFO-BRUSH	Величина типа Brush. Стилль штриховки объектов типа "эллипс", "рамка", "область", "прямоугольник" и "сглаженный прямоугольник".
OBJ-INFO-TEXTFONT	Величина типа Font. Стилль шрифта текстового объекта. <b>Замечание:</b> Если текстовый объект принадлежит таблице (а не Отчету), то размер шрифта равен нулю, а размер шрифта динамически определяется MapInfo в зависимости от размера окна Карты.
OBJ-INFO-SYMBOL	Величина типа Symbol. Стилль символа точечного объекта.
OBJ-INFO-NPNTS	Величина типа Integer. Число узлов в полилинии или в многоугольнике области.
OBJ-INFO-SMOOTH	Величина типа Logical. Признак сглаженности объекта типа "полилиния".
OBJ-INFO-FRAMEWIN	Величина типа Integer. Идентификатор окна, присоединенного к объекту типа "рамка".
OBJ-INFO-FRAMETITLE	Величина типа String с заголовком рамки.
OBJ-INFO-NPOLYGONS	Величина типа SmallInt. Число полигонов в объекте типа "область" или число ломаных компонент в объекте типа "полилиния".
OBJ-INFO-NPOLYGONS+N	Величина типа Integer. Число узлов в <i>n</i> -ом полигоне в объекте типа "область" или число узлов в <i>N</i> -ой ломаной линии, являющейся компонентой объекта типа "полилиния". <b>Замечание:</b> для объектов типа "область" число узлов многоугольника будет на единицу больше, чем число вершин у многоугольника, потому что MapInfo считает первый узел дважды (один раз как первый узел и второй раз как последний узел). Так, функция <b>ObjectInfo( )</b> возвращает 4 для треугольной монокомпонентной области.
OBJ-INFO-TEXTSTRING	Величина типа String. Текстовое содержимое объекта типа "текст". Если объект состоит из нескольких строк, то результат будет включать символ конца строки (Chr\$(10)).

OBJ-INFO-TEXTSPACING	Вещественное число 1, 1.5 или 2, определяющее интерлиньяж в текстовом объекте.
OBJ-INFO-TEXTJUSTIFY	Число типа SmallInt, определяющее выравнивание текста: 0 – по левому краю, 1 – по центру, 2 – по правому краю.
OBJ-INFO-TEXTARROW	Число типа SmallInt, определяющее стиль указки в текстовом объекте: 0 – нет указки, 1 – просто линия, 2 – стрелка.
OBJ_INFO_FILLFRAME	Величина типа Logical. “Да” (TRUE), если объект типа “рамка” показывает Карту и для него установлен режим “Заполнить Рамку Картой”.

Заметим, что некоторые значения для параметра *attribute* могут адресоваться объектам определенного типа. Например, значение стиля символа может быть определено только для объекта типа “точка”.

Если Вы используете код OBJ-INFO-TYPE как значение параметра *attribute*, то функция **ObjectInfo( )** вернет код, соответствующий типу графического объекта. В следующей таблице приведены имена этих кодов из файла стандартных определений.

Код объекта	Соответствующий тип объекта
OBJ-ARC	Дуга
OBJ-ELLIPSE	Эллипс или окружность
OBJ-LINE	Прямая линия
OBJ-PLINE	Полилиния
OBJ-POINT	Точечный объект
OBJ-FRAME	Рамка в окне Отчета
OBJ-REGION	Область
OBJ-RECT	Прямоугольник
OBJ-ROUNDRECT	Скругленный прямоугольник
OBJ-TEXT	Текстовый объект

Функция **ObjectInfo()** также поддерживает следующие типы объектов:

Группа точек

Атрибут	Возвращаемое значение
---------	-----------------------

## Функция ObjectInfo( )

---

1 OBJ_INFO_TYPE	Короткое целое, тип объекта. Возвращаемое значение для группы точек - это 11
2 OBJ_INFO_SYMBOL	Стиль символа.
11 OBJ_INFO_NONEMPTY	Логическое, возвращает TRUE если объект группа точек имеет узлы, FALSE - если объект пустой.

Настройки атрибута	Возвращаемое значение
--------------------	-----------------------

20 OBJ_INFO_NPNTS	Целое, определяющее число узлов в группе точек.
----------------------	---

### Коллекция

(1) OBJ_INFO_REGION	Короткое целое, определяющее тип объекта. Возвращаемое значение для коллекции - это 12.
(8) OBJ_INFO_PLINE	Возвращает значение соответствующее полигонам, входящим в коллекцию. Если коллекция не содержит полигонов, будет возвращен пустой полигон. Этот запрос действует только для объектов типа "коллекция".
(9) OBJ_INFO_MPOINT	Возвращает значение соответствующее полилиниям, входящим в коллекцию. Если коллекция не содержит полилиний, будет возвращен пустой полилиния. Этот запрос действует только для объектов типа "коллекция".
10 OBJ_INFO_NONEMPTY	Возвращает значение соответствующее группам точек, входящим в коллекцию. Если коллекция не содержит групп точек, будет возвращен пустая группа точек. Этот запрос действует только для объектов типа "коллекция".
11 OBJ_INFO_NONEMPTY	Логическое, возвращает TRUE, если коллекция имеет узлы, FALSE - если объект пустой.

### Пример:

Узнаем, какой объект присоединен к первой записи таблицы CITY.

```
Include "MAPBASIC.DEF"
Dim counter, obj-type As Integer
Open Table "city"
Fetch First From city
'
' В этом месте программы выражение city.obj
' представляет графический объект, присоединенный
' к первой строке таблицы CITY.
'
obj-type = ObjectInfo(city.obj, OBJ-INFO-TYPE)
```

```
Do Case obj-type
Case OBJ-LINE
  Note "Первый объект в таблице - прямая линия."
Case OBJ-PLINE
  Note "Первый объект в таблице - полилиния,..."
  counter = ObjectInfo(city.obj, OBJ-INFO-NPNTS)
  Note " ... которая имеет " + Str$(counter) + " узлов."
Case OBJ-REGION
  Note "Первый объект в таблице - область,..."
  counter = ObjectInfo(city.obj, OBJ-INFO-NPOLYGONS)
  Note ", которая состоит из " + Str$(counter) + " полигонов..."
  counter = ObjectInfo(city.obj, OBJ-INFO-NPOLYGONS+1)
  Note "и первый полигон имеет " + Str$(counter) + " узлов"
End Case
```

### Смотрите также:

[Alter Object](#), [ObjectGeography\( \)](#), [Pen](#), [Brush](#), [Symbol](#), [Font](#)

### Функция ObjectLen( )

#### Назначение:

Вычисляет географическую длину объекта типа "линия" или "полилиния".

#### Синтаксис:

**ObjectLen**(*expr*, *unit\_name*)

где

*expr* – выражение, результат которого есть величина типа Object,

*unit\_name* – строковая величина, задающая единицы измерения расстояний (например, "km" – километры)

#### Величина, полученная в результате:

Вещественное число. Величина типа Float.

#### Описание:

Функция **ObjectLen( )** возвращает длину объекта, представленного выражением *expr*. Только линия или полилиния могут иметь ненулевое значение длины. Для вычисления периметра областей, эллипсов, прямоугольников используйте функцию **Perimeter( )**.

Функция возвращает периметр в единицах, заданных вторым параметром. Полный список строковых значений, определяющих единицы расстояний, представлен в описании оператора **Set Distance Units**.

#### Пример:

```
Dim geogr-length As Float
Open Table "streets"
Fetch First From streets
geogr-length = ObjectLen(streets.obj, "km")
' geogr-length теперь содержит значение длины
' сегмента улицы в километрах
```

#### Смотрите также:

**Distance( )**, **Perimeter( )**, **Set Distance Units**

## Функция **ObjectNodeX( )**

### Назначение:

Возвращает X-координату определенного узла определенного полигона в определенной области или X-координату узла полилинии.

### Синтаксис:

**ObjectNodeX(object, polygon\_num, node\_num)**

где

*object* – выражение, результат которого есть величина типа Object;

*polygon\_num* – номер полигона или ломаной линии;

*node\_num* – номер узла.

### Величина, полученная в результате:

Вещественное число. Величина типа Float.

### Описание:

Функция **ObjectNodeX( )** возвращает координату по оси X узла полигона в области.

Соответствующую координату по оси Y Вы можете получить при помощи функции **ObjectNodeY( )**.

Параметр *polygon\_num* должен иметь значение 1 или более. Он указывает, какой полигон (если опрашивается регион) или какая секция (если опрашивается полилиния) будет опрошен. Вызовите функцию **ObjectInfo( )** для определения номера полигона в регионе или секции в полилинии.

Функция **ObjectNodeX( )** теперь поддерживает группы точек и возвращает координату X указанного узла в этом типе объекта.

Параметр *node\_num* должен принимать целочисленные значения от 1 и более. Для задания номера узла в объекте Вы можете использовать в этом параметре вызов функции **ObjectInfo( )**.

Координата, которую Вы получите в результате применения **ObjectNodeX( )**, будет выведена в системе координат, которая определена как текущая для MapBasic. Если система координат заранее не была выбрана (смотрите описание оператора **Set CoordSys**), то MapBasic использует систему широта/долгота.

### Пример:

Здесь открывается таблица ROUTES. Если первый объект есть полилиния, то считываются координаты первого узла. На этом месте создается точечный объект.

```
Dim i-obj-type As SmallInt,
    x, y As Float,
    new-pnt As Object
Open Table "routes"
Fetch First From routes
' В этом месте программы выражение city.obj
' представляет графический объект, присоединенный
' к первой строке таблицы ROUTES.
i-obj-type = ObjectInfo(routes.obj, OBJ-INFO-TYPE)
If i-obj-type = OBJ-PLINE Then
' ... тогда объект - полилиния...
x = ObjectNodeX(routes.obj, 1, 1) ' чтение долготы
y = ObjectNodeY(routes.obj, 1, 1) ' чтение широты
Create Point Into Variable new-pnt (x, y)
Insert Into routes (obj) Values (new-pnt)
End If
```

## Функция `ObjectNodeX( )`

---

### Смотрите также:

`Alter Object`, `ObjectGeography( )`, `ObjectInfo( )`, `ObjectNodeY( )`, `Set CoordSys`



## Функция **ObjectNodeY( )**

### Назначение:

Возвращает Y-координату определенного узла определенного полигона в определенной области или возвращает Y-координату узла полилинии.

### Синтаксис:

**ObjectNodeY**(*object*, *polygon\_num*, *node\_num*)

где

*object* – выражение, результат которого есть величина типа Object;

*polygon\_num* - это положительное целое, определяющее, какой полигон или секция опрашиваются. Параметр игнорируется для групп точек (используется для полигонов и полилиний).

*node\_num* - это положительное целое, определяющее, какой узел считывается.

### Величина, полученная в результате:

Вещественное число. Величина типа Float.

### Описание:

Функция **ObjectNodeY( )** возвращает координату по оси Y узла полигона в области.

Соответствующую координату по оси X Вы можете получить при помощи функции **ObjectNodeX( )**.

Правила использования этой функции такие же, как для **ObjectNodeX( )**.

### Пример:

Смотрите в описании функции **ObjectNodeX( )**.

### Смотрите также:

**Alter Object**, **ObjectGeography( )**, **ObjectInfo( )**, **ObjectNodeX( )**, **Set CoordSys**

# Оператор Objects Check

### Назначение

Проверяет таблицу на предмет обнаружения некорректных данных различных типов.

### Синтаксис

```
Objects Check From tablename  
[ Into Table tablename]  
[Overlap]  
[Symbol Clause]  
[Pen Clause]  
[Brush Clause]
```

### Описание

**Objects Check** будет проверять таблицу, указанную предложением **From** на предмет различных некорректных данных. Проверке подлежат только объекты типа регионов. Регионы будут проверяться на предмет самопересечения.

Самопересечение может вызвать проблемы при вычислении площадей и др.. Могут быть проблемы и при операциях с объектами, например, комбинации, удалении части и др..

При работе этого оператора, в обнаруженных местах самопересечения создаются точечные объекты, которые помещаются в выходящую таблицу. Выходящая таблица может быть задана предложением **Into Table**. Если предложения **Into Table** нет, то выходящие данные помещаются в ту же таблицу, где и исследуемые данные. Точечные объекты оформлены в стиле, указанном в **Symbol Clause**. По умолчанию, это красная булавка размером 28.

Многие карты регионов должны не иметь наложения граничащих друг с другом объектов, например штатов или областей. Предложение **Overlap** будет проверять таблицу на предмет обнаружения наложения регионов. Из областей наложения формируются полигоны, которые помещаются в выходящую таблицу. Эти полигоны будут создаваться с использованием предложения **Brush Clause** определяющего заливку полигона и предложения **Pen Clause**, представляющего границу полигона. Стандартные цвета - желтая заливка и тонкая черная граница.

### Пример

В этом примере запускается **Objects Check** для проверки таблицы **TestFile** и результат сохраняется в таблице **DumpFile**. Будет определен параметр наложения и изменены стили для **Point** и **Polygon**.

```
objects check from TestFile into table Dumpfile Overlap  
Point_Symbol_Clause = "Symbol (67, 16711680, 28)"  
Polygon_Symbol_Clause = "Brush (2, 16776960, 0)"
```

### Смотрите так же

**Objects Enclose**

## Оператор Objects Clean

### Назначение

Корректирует топологию объектов из данной таблицы, дополнительно может удалять перекрытия и закрывать бреши между полигонами. Таблица должна быть таблицей выборки (Selection). Все объекты, подлежащие коррекции, должны быть замкнутыми (полигоны, прямоугольники, скругленные прямоугольники или эллипсы).

### Синтаксис

**Objects Clean From *tablename***  
**[Overlap]**  
**[Gap Area [Unit Units] ]**

### Описание

Объекты из исходной *tablename* проверяются на предмет топологической корректности, например, самопересечений, наложения и пустот. Самопересекающиеся полигоны в форме "восьмерки" будут превращены в два полигона, соприкасающихся в общей вершине. Полигоны, содержащие острые выступы, подвергаются обработке, при которой часть таких выступов удаляется. Подвергнутый коррекции объект будет помещен вместо исходного объекта.

Если включено ключевое слово **Overlap**, то области наложения полигонов друг на друга удаляются из полигонов. Часть перекрытия будет удалена из всех перекрывающихся полигонов, кроме того, у которого наибольшая площадь.

Пустоты (пустоты) - это замкнутые области, в которых нет объектов. В таблице с границами, полигоны должны иметь общие границы. Здесь не должно быть перекрытия полигонов и не должно быть брешей между полигонами. В некоторых случаях пустоты между полигонами имеют смысл и право на существование. Например, Великие озера на карте мира являются лежат между Канадой и США (однако не включаются в какой-либо штат США или провинцию Канады). Тем не менее, большинство пустот является результатом плохого согласования общих границ между полигонами. Такие бреши обычно имеют малые размеры.

Чтобы успешно разделить допустимые пустоты, такие как Великие озера, от мелких, ненужных брешей, используется предложение **Area**. Любые пустоты больше некоторого заданного размера будут оставляться без изменения. Единицы измерения площади предельной бреши **Area** задаются предложением **Units**. Если подпредложение **Units** не задано, то площадь брешей **Area** будет измеряться в текущих единицах измерения MapBasic.

Пустоты будут удалены путем их объединения с соседним полигоном, причем именно тем, у которого площадь больше.

Чтобы иметь представление об использовании предложения **Area**, посмотрите использование команды **Objects Check**. Величина **Gaps** в команде **Objects Check** действует аналогично команде "Коррекция топологии" из MapInfo.

### Пример

```
Open Table "STATES.TAB" Interactive
Map From STATES
Set Map Layer 1 Editable On
select * from STATES
Objects Clean From Selection Overlap Gap 10 Units "sq m"
```

## Оператор Objects Clean

---

### Смотрите также

[Оператор Objects Create](#)

[Оператор Objects Disaggregate](#)

[Оператор Objects Check](#)

## Оператор **Objects Combine**

### Назначение:

Объединяет объекты в таблице так, как это делает команда ОБЪЕКТЫ > КОМБИНАЦИЯ.

### Синтаксис:

```
Objects Combine  
[ Into Target ]  
[ Data column = expression [ , column = expression ... ] ]
```

где

*column* — имя колонки в таблице.

### Описание:

Оператор **Objects Combine** создает объект, представляющий собой географическое объединение выбранных объектов. С помощью оператора **Objects Combine** можно также обобщать данные, вычислять суммы и средние значения величин из записей, к которым присоединены объекты.

Оператор **Objects Combine** соответствует команде MapInfo ОБЪЕКТЫ > КОМБИНАЦИЯ. Если Вы в MapInfo выполните команду и при этом будет открыто окно MapBasic, то в протоколе выполненных действий будет использован оператор **Objects Combine**. Описание выполняющейся операции смотрите в описании команды в документации MapInfo.

### Предложение **Into Target**

Это предложение используется, если выбран изменяемый объект (с помощью оператора **Set Target**) и если изменяемый объект один.

Если в операторе есть предложение **Into Target**, то MapInfo будет комбинировать изменяемый объект с выбранными объектами на Карте. Объект, полученный в результате комбинирования, заменит изменяемый объект в таблице.

Если выбранные объекты, участвующие в комбинировании, находятся в той же таблице, что и изменяемый объект, то MapInfo удалит записи, к которым присоединены выбранные объекты.

Если выбранные объекты находятся в другой таблице, то они и соответствующие записи не будут удалены.

Если Вы не используете предложение **Into Target**, то MapInfo комбинирует только выбранные объекты без использования изменяемого объекта, если он назначен. Выбранные объекты и соответствующие им строки в таблице удаляются, новая строка с объектом, полученным в результате комбинирования, добавляется в конец таблицы.

### Предложение **Data**

Предложение **Data** управляет обобщением данных. Информация об обобщении данных приводится в описании команды ОБЪЕКТЫ > КОМБИНАЦИЯ в Руководстве пользователя MapInfo. За ключевым словом **Data** должен следовать список определений через запятую. Каждое определение является выражением, по которому будет вычислено (или изменено) значение в определенной колонке для записи, которая будет получена в результате выполнения оператора **Objects Combine**. Вычисления или изменения должны производиться в соответствии с типом колонки (численным, строковым и т. п.)

Следующая таблица приводит некоторые варианты определений для колонки:

## Оператор Objects Combine

---

Выражение	Описание
$col\_name = col\_name$	Содержимое колонки не меняется.
$col\_name = value$	MapBasic помещает значение <i>value</i> в поле записи объекта, полученного в результате.
$col\_name = \text{Sum}(col\_name)$	Используется только для численных колонок. MapBasic помещает сумму значений колонки <i>col_name</i> из всех записей объектов, участвующих в комбинации, в поле записи объекта, полученного в результате.
$col\_name = \text{Avg}(col\_name)$	Используется только для численных колонок. MapBasic помещает среднее из значений колонки <i>col_name</i> из всех записей объектов, участвующих в комбинации, в поле записи объекта, полученного в результате.
$col\_name = \text{WtAvg}(col\_name, wt\_col\_name)$	Используется только для численных колонок. MapBasic помещает взвешенное среднее число из значений колонки <i>col_name</i> из всех записей объектов, участвующих в комбинации, в поле записи объекта, полученного в результате. В качестве коэффициентов веса используются значения из колонки <i>wt_col_name</i> .

Список **Data** может состоять из определений для всех колонок таблицы. Если в списке определены не все колонки, то MapBasic разместит пустые значения в неописанные поля новой записи.

Если предложения **Data** нет в операторе, но используется предложение **Into Target**, то MapInfo сохранит значения изменяемого объекта в записи.

Если в операторе не используется ни предложение **Data**, ни предложение **Into Target**, то результирующий объект будет помещен в новую строку и MapInfo разместит нулевые и пустые значения в поля этой записи.

### Смотрите также:

**Combine( ), Set Target**

## Оператор Objects Disaggregate

### Назначение

Разбивает объект на составные части.

### Синтаксис

```
Objects Disaggregate [IntoTable name]  
[ All | Collection ]  
[ Data      column_name = expression ]  
[ , column_name = expression ... ] ]
```

### Описание

Если объект включает в себя несколько других объектов, то для каждого объект, входящего в состав основного объекта, а в итоговой таблице создается новый объект.

По умолчанию, каждый составной объект будет разделен на элементарные одиночные объекты. Регион будет разделен на некоторое число полигонов, в зависимости от того, установлен ли флаг **All**. Если флажок **All** установлен, то будет создано множество отдельных одиночных объектов-полигонов. Для островов (внутренние границы) будут созданы отдельные объекты-полигоны. Если флажок **All** не выставлен, то в результирующих объектах острова сохраняются. Например, если исходный регион содержит 3 полигона и один из них является островом в другом полигоне, то в результате получится 2 объекта-полигона - один из которых содержит остров (пустой).

Сложные полилинейные объекты будут разбиты на отдельные полилинии, группы точек - на отдельные точечные объекты будут создавать новые точечные объекты, по одному объекту для каждого узла в исходном объекте "Группа точек".

Если коллекция содержит регион, то в зависимости от переключателя **All** будут созданы новые объекты-полигоны; если она включает сложные полилинии, то из каждой отдельной полилинии будет создан отдельный объект; если в коллекции содержится группа точек, то будут созданы новые объекты-точки, по одной точке для каждого узла из группы. Все другие типы объектов, включая точки, линии, дуги, прямоугольники, скругленные прямоугольники и эллипсы, которые уже являются простыми объектами, этой операцией не изменяются.

Если регион содержит единственный полигон, то на выходе он останется без изменений. Если сложная полилиния содержит одну полилинию, то на выходе она останется без изменений.

Если группа точек содержит единственный узел, то выходящий объект преобразуется в точечный объект, содержащий этот узел. Дуги, прямоугольники, скругленные прямоугольники, эллипсы на выходе остаются без изменений. Другие типы объектов (например, текстовые) не обрабатываются командой **Objects Disaggregate**, при попытке это сделать выдается сообщение об ошибке.

Переключатель **Collection** разделяет только объекты коллекции. Если коллекция содержит регион, то этот регион на выходе станет новым объектом. Если объект коллекция содержит полилинию, то эта полилиния будет на выходе новым объектом. Если объект коллекция содержит группу точек, то эта группа точек будет на выходе новым объектом. В этом различие от опции, описанной в начале, поскольку выходящий регион может содержать несколько полигонов, выходящая сложная полилиния может содержать несколько полилиний. При использовании опции, описанной выше, группа точек не будет создана.

## Оператор Objects Disaggregate

---

Переключатель **Collection** передает на выход без изменений все другие типы объектов, включая точки, группы точек, линии, дуги, регионы, прямоугольники, скругленные прямоугольники и эллипсы.

Если не задается **Into Table**, то в качестве таблицы для выходящих данных используется текущая редактируемая таблица. Входящие для обработки командой объекты берутся из текущей выборки.

Дополнительное предложение **Data** определяет, какие значения хранятся в колонках изменяемых объектов. Предложение **Data** может содержать список присваиваемых значений, разделенных точкой с запятой. Все значения, которые могут быть присвоены, описаны в таблице ниже:

Выражение	Действие
<code>col_name = col_name</code>	Не изменяет величины, хранящиеся в колонке.
<code>col_name = column_value</code>	Хранит указанные величины в колонке. Если колонка строковая, то значение тоже будет строковым; если колонка числовая, то значение будет числовым.
<code>col_name = Proportion( col_name )</code>	Используется только для числовых колонок; уменьшает число, содержащееся в колонке, пропорционально удаленной площади объекта.

Предложение **Data** может содержать назначения для каждой колонки в таблице. Если предложение **Data** содержит назначения только для части колонок, пустые значения будут присвоены тем колонкам, которые не поименованы в предложении **Data**.

Если Вы пропускаете предложение **Data** целиком, все колонки будут пустыми, содержащими нулевые значения для числовых колонок и пустые значения для строковых колонок.

### Пример

```
Open Table "STATES.TAB" Interactive
Map From STATES
Set Map Layer 1 Editable On
select * from STATES
Objects Disaggregate Into Table STATES
```

### Смотрите также

Оператор ObjectsCreate



## Оператор Objects Enclose

### Назначение

Создает регионы, которые формируются из коллекции полилиний; соответствует команде MapInfo "Замкнуть".

### Синтаксис

**Objects Enclose**

[ Into Table *tablename* ]

[ Region ]

*tablename* - это таблица, в которую надо поместить создаваемые объекты.

### Описание

**Objects Enclose** создает объекты, представляющие области, ограниченные замкнутыми линейными объектами (линии, полилинии и дуги). Новый регион создается для каждой замкнутой полигональной области. Исходные объекты получают из текущей выборки. В отличие от оператора **Objects Combine**, оператор **Objects Enclose** не перемещает исходные объекты. Объединения данных не производится.

Дополнительное предложение **Region** будет позволять замыкать объекты (регионы, прямоугольники, скругленные прямоугольники и эллипсы), которые будут использоваться как входящие в операции **Objects Enclose**. Входящие полигоны будут конвертироваться в полилинии для осуществления этой операции. Получаемый результат аналогичен первичной конвертации любых замкнутых объектов в полилинии, и затем осуществится операция **Objects Enclose**.

Все входящие объекты должны быть линейными или замкнутыми, а использование других типов объектов (например, точек, групп точек, коллекций и текста) приведет к ошибке с соответствующим сообщением. Если замкнутые объекты существуют в выборке, и переключатель **Region** не определен, то такие объекты будут проигнорированы.

Функция MapBasic **Combine()** обновлена и позволяет производить объединение точек, групп точек и коллекций. Ранее объединяемые объекты должны были являться линейными (линии, полилинии, дуги) - тогда в результате операции являлись полилинии, или замкнутыми (полигоны, прямоугольники, скругленные прямоугольники, эллипсы) - тогда в результате операции создавались регионы. Объединение разнородных объектов (точек, групп точек, коллекций) не допускалось. В новой версии нельзя объединять только текстовые объекты.

Как уже отмечалось, в MapInfo Pro 6.5 появились новые типы объектов: группы точек и коллекции, которые теперь можно использовать в операции объединения. Следующая таблица подробно отражает возможные комбинации и результат их действия:

Тип входящих объектов	Тип выходящих объектов	Тип объекта в результате
точечные или группы точек	точечные или группы точек	группы точек
линейные (линия, полилиния, дуга)	линейные	полилинии

## Оператор Objects Enclose

---

Тип входящих объектов	Тип выходящих объектов	Тип объекта в результате
замкнутые (полигон, прямоугольник, скругленный прямоугольник, эллипс)	замкнутые	регионы
точечные, группы точек, линейные, замкнутые, коллекции	точечные, группы точек, линейные, замкнутые, коллекции	коллекции

### Пример

Будут выбраны все объекты в таблице testfile, выполнено действие Objects Enclose и сохранен результат в таблице dump\_file.

```
select * from testfile  
Objects Enclose Into Table dump_file
```

### Смотрите также

[Objects Combine](#), [Objects Check](#)

## Оператор **Objects Erase**

### Назначение:

Удаляет часть изменяемого объекта, которая перекрывается другим объектом (или объектами).  
Оператор соответствует команде ОБЪЕКТЫ > УДАЛИТЬ ЧАСТЬ в MapInfo.

### Синтаксис:

**Objects Erase Into Target**  
[ **Data** *column\_name* = *expression* ]  
[ , *column\_name* = *expression* ... ] ]

где

*column\_name* – имя колонки в таблице.

### Описание:

Оператор **Objects Erase** удаляет часть объекта (или весь объект), который объявлен как изменяемый. Оператор **Objects Erase** соответствует команде MapInfo ОБЪЕКТЫ > УДАЛИТЬ ЧАСТЬ. Если Вы в MapInfo выполните команду и при этом будет открыто окно MapBasic, то в протоколе выполненных действий будет использован оператор **Objects Erase**. Описание выполняющейся операции смотрите в описании команды в документации MapInfo.

Удаляется та часть изменяемого объекта, которая перекрывается выбранным объектом (или объектами). Если изменяемый объект перекрывается полностью, то он удаляется полностью. Если Вам надо удалить часть, свободную от перекрытия объектами, из текущего выбора, то используйте оператор **Objects Intersect**.

Перед выполнением оператора **Objects Erase** должен быть выбран изменяемый объект и один или более замкнутых объектов (типа "область", "прямоугольник", "скругленный прямоугольник" или "эллипс"), играющих роль "ластика". Изменяемый объект может быть назначен командой в MapInfo ОБЪЕКТЫ > ВЫБРАТЬ ИЗМЕНЯЕМЫЙ ОБЪЕКТ или оператором **Set Target** из прикладной программы.

Предложение **Data** управляет размещением данных в колонках изменяемой таблицы. За ключевым словом **Data** должен следовать список определений через запятую. Каждое определение является выражением, по которому будет изменено значение в определенной колонке в записи изменяемого объекта. Вычисления или изменения должны производиться в соответствии с типом колонки (численным, строковым и т. п.) Следующая таблица приводит некоторые варианты определений для колонки:

## Оператор Objects Erase

---

### Выражение

*col\_name* = *col\_name*

*col\_name* = *value*

*col\_name* = **Proportion**(*col\_name*)

### Эффект

Содержимое колонки не меняется.

MapBasic помещает значение *value* в поле записи объекта. Если тип колонки символьный, то *value* должно быть строкой. Если тип колонки численный, то *value* должно быть числом.

Используется только для численных колонок. MapBasic пропорционально изменяет значение в колонке *col\_name*. Так, если была удалена половина объекта, то значение в колонке уменьшится наполовину.

Список **Data** может состоять из определений для всех колонок таблицы. Если в списке определены не все колонки, то MapBasic разместит пустые значения в неописанные поля новой записи.

Если в операторе не используется предложение **Data**, то MapBasic заполнит все поля записи изменяемого объекта нулевыми и пустыми значениями.

### Примеры:

В результате следующего оператора **Objects Erase**, не использующего предложение **Data**, все записи, к которым присоединены изменяемые объекты, теряют свои значения, независимо от того, перекрываются ли они выбранными объектами или нет.

#### **Objects Erase Into Target**

Следующий оператор **Objects Erase** имеет предложение **Data**, которое задает выражения для трех колонок ("State-Name", "Pop-1990" и "Med-Inc-80"). Этот оператор присваивает строку "остаток" колонке "State\_Name" и определяет, что значения в колонке "Pop\_1990" будут уменьшены пропорционально оставшейся после удаления площади. Значения в колонке "Med\_Inc\_80" сохраняются нетронутыми. Остальные колонки изменяемого объекта очищаются.

#### **Objects Erase Into Target**

##### **Data**

```
State_Name = "остаток",  
Pop_1990 = Proportion( Pop_1990 ),  
Med_Inc_80 = Med_Inc_80
```

### Смотрите также:

**Erase( ), Objects Intersect**

## Оператор Objects Intersect

### Назначение:

Удаляет часть изменяемого объекта, которая остается свободной от перекрытия другим объектом (или объектами). Оператор соответствует команде ОБЪЕКТЫ > УДАЛИТЬ ВНЕШНЮЮ ЧАСТЬ в MapInfo.

### Синтаксис:

```
Objects Intersect Into Target  
[ Data column_name = expression ]  
[ , column_name = expression ... ] ]
```

где

*column\_name* – имя колонки в таблице;

*expression* – выражение.

### Описание:

Оператор **Objects Intersect** удаляет часть объекта (или объект полностью), который назначен как изменяемый. Оператор **Objects Intersect** соответствует команде MapInfo ОБЪЕКТЫ > УДАЛИТЬ ВНЕШНЮЮ ЧАСТЬ. Если Вы в MapInfo выполните команду и при этом будет открыто окно MapBasic, то в протоколе выполненных действий будет использован оператор **Objects Intersect**. Описание выполняющейся операции смотрите в описании команды в *Справочнике MapInfo*.

Удаляется та часть изменяемого объекта, которая остается свободной от перекрытия выбранным объектом (или объектами). Если изменяемый объект не пересекается с выбранными объектами, ничего не происходит. Если Вам надо удалить часть, перекрываемую объектами, из текущего выбора, то используйте оператор **Objects Erase**.

Операторы **Objects Intersect** и **Objects Erase** похожи способом удаления части объекта и различаются только тем, какую часть объекта удаляют.

Для более подробной информации смотрите оператор **Objects Erase**.

### Смотрите также:

**Create Object, Overlap( )**

### Оператор Objects Overlay

#### Назначение:

Добавляет узлы изменяемому объекту в точках пересечения линий или контуров выбранных объектов. Оператор соответствует команде ОБЪЕКТЫ > ДОБАВИТЬ УЗЛЫ.

#### Синтаксис:

**Objects Overlay Into Target**

#### Описание:

Перед выполнением оператора **Objects Overlay** должен быть назначен изменяемый объект и один или более объектов любого типа, кроме текстового или точечного. Изменяемый объект может быть назначен командой в MapInfo ОБЪЕКТЫ > ВЫБРАТЬ ИЗМЕНЯЕМЫЙ ОБЪЕКТ или оператором **Set Target** из прикладной программы.

Более подробная информация приводится в описании команды ОБЪЕКТЫ > ДОБАВИТЬ УЗЛЫ в *Справочнике MapInfo*.

#### Смотрите также:

**OverlayNodes( ), Set Target**

## Оператор Objects Snap

### Назначение

Выполняет коррекцию объектов из данной таблицы и осуществляет различные топологические операции над объектами, включая совмещение узлов разных объектов, прилегающих друг к другу и генерализацию/разреживание узлов. Таблица может быть и Selection. Все объекты, подвергающиеся действию оператора, должны быть или все линейными (то есть, полилинии или дуги) или все замкнутые (то есть, полигоны, прямоугольники, скругленные прямоугольники или эллипсы). Смешанные линейные и замкнутые объекты не могут обрабатываться этим оператором, будет выдано сообщение об ошибке.

### Синтаксис

**Objects Snap From** *tablename*

[Tolerance [Node *node\_distance*] [Vector *vector\_distance*] [Units *unit\_string*] ]  
 [Thin [Bend *bend\_distance*] [Distance *spacing\_distance*] [Units *unit\_string*] ]  
 [Cull Area *cull\_area* [Units *unit\_string*] ] ]

### Описание

Объекты из исходной таблицы *tablename* проверяются на предмет различных проблем и несоответствий, таких, как самопересечения. Самопересекающиеся полигоны в форме "восьмерки" будут превращаться в два полигона, соприкасающиеся в одной точке. Полигоны, содержащие острые выступы, будут обработаны так, что эти пики будут удалены. Результирующий, исправленный объект помещается вместо исходного объекта. Если существуют наложения полигонов друг на друга, то такие наложения будут удалены. Удаление наложений состоит в удалении наложенной части. Эта часть - "избыточный полигон" - образует с одним из двух исходных объектов регион. Такой регион составлен из "избыточного полигона" и непересекающейся части исходного полигона.

Значения **Node** и **Vector Tolerances** предназначены для совмещения узлов из разных близлежащих объектов и удаляют мелкие пустоты и мелкие пустоты между двумя объектами. Субпредложение **Units** предложения **Tolerances** позволяет задать единицы измерения расстояния (например, "km" для километров), применяемых к значениям **Node** и **Vector**. Если субпредложение **Units** отсутствует, то значения **Node** и **Vector** будут интерпретироваться в текущих значениях измерения расстояния MapBasic. По умолчанию MapBasic использует мили; для изменения единиц измерения смотрите оператор **Set Distance Units**.

**Node** - это радиус вокруг конечных узлов полилиний. Если узлы из других объектов попадают в этот радиус, то один или два узла перемещаются так, что попадают в одну точку (т.е. они совмещаются). **Vector** - это радиус, используемый для промежуточных точек полилиний. Его назначение тоже, что и у радиуса **Node**, кроме того, что он используется только для промежуточных точек полилиний. Обратите внимание, что для полигонов не определено понятие конечных точек (значение **Node** не используется) ввиду их замкнутости. Для них используется только величина **Vector**, которая применяется ко всем узлам объекта. Для полилиний значение **Node** должно быть больше или равно значению **Vector**.

Значения **Bend** и **Distance** могут использоваться для удобства операций разреживания узлов и обобщения контуров. Они уменьшают число узлов, используемых в объекте, сохраняя

## Оператор Objects Snap

---

основные черты формы объекта. Субпредложение **Units** предложения **Thin** позволяет указать имя единицы измерения расстояния (например, "km" для километров) в которых измеряется значение **Bend** и **Distance**. Если субпредложение **Units** отсутствует, то значения **Bend** и **Distance** будут интерпретироваться в текущих единицах измерения MapBasic.

Значение **Bend** используется для управления коллинеарным отклонением группы из 3 последовательных узлов. Эти 3 узла связываются в треугольник. Измеряется перпендикуляр, опущенный из средней точки на длинную сторону треугольника. Если это расстояние меньше значения **Bend**, то эти три узла рассматриваются как коллинеарные и второй (средний) узел удаляется из объекта.

Расстояние **Distance** используется для удаления узлов из одного объекта, если узлы расположены слишком близко друг к другу. Измеряется расстояние между двумя соседними точками объекта. Если это расстояние меньше, чем **Distance**, то один из двух узлов будет удален.

Значение **Cull Area** используется для удаления избыточных полигонов, которые меньше некоторого заданного этой величиной значения. Субпредложение **Units** из предложения **Cull** позволяет настроить единицы измерения площади (например, "кв км" для квадратных километров) применяемые в значении **Area**. Если субпредложение **Units** отсутствует, то значение **Area** будет интерпретироваться в текущих единицах измерения площади MapBasic. По умолчанию MapBasic использует квадратные мили в качестве единиц измерения площади; для изменения этих единиц см. оператор **Set Area Units**.

Внимание: Для всех расстояний и площадей, упомянутых выше, всегда используется тип измерений на плоскости. Систему координат и проекцию всегда надо учитывать. Вычисления расстояний и площадей в Долготе/Широте на плоскости осуществляются не с математической точностью. Убедитесь, что Вы работаете в подходящей системе координат (декартовых) перед работой с этим оператором.

### Пример

```
Open Table "STATES.TAB" Interactive
Map From STATES
Set Map Layer 1 Editable On
select * from STATES
Objects Snap From Selection Tolerance Node 3 Vector 3 Units "mi"
Thin Bend 0.5 Distance 1 Units "mi" Cull Area 10 Units "sq mi"
```

### Смотрите также

- Оператор Create Object
- Оператор Overlap
- Оператор Objects Clean



## Оператор Objects Split

### Назначение:

Разделяет изменяемые объекты на части, используя форму выбранных объектов как "ластик".  
Оператор соответствует команде ОБЪЕКТЫ > РАЗРЕЗАТЬ.

### Синтаксис:

```
Objects Split Into Target  
[ Data column_name = expression ]  
  [ , column_name = expression ... ] ]
```

где

*column\_name* – имя колонки в таблице.

### Описание:

Оператор **Objects Split** разрезает каждый изменяемый объект на несколько. Оператор **Objects Split** соответствует команде MapInfo ОБЪЕКТЫ > РАЗРЕЗАТЬ. Если Вы в MapInfo выполните команду и при этом будет открыто окно MapBasic, то в протоколе выполненных действий будет использован оператор **Objects Split**. Описание выполняющейся операции смотрите в описании команды в *Справочнике MapInfo*.

Перед выполнением оператора **Objects Split**, должен быть выбран изменяемый объект (или объекты) и один или более замкнутых объектов (типа "область", "прямоугольник", "скругленный прямоугольник" или "эллипс"), играющих роль "ластика". Изменяемый объект может быть назначен командой в MapInfo ОБЪЕКТЫ > ВЫБРАТЬ ИЗМЕНЯЕМЫЙ ОБЪЕКТ или оператором **Set Target** из прикладной программы.

Предложение **Data** управляет обобщением данных. За ключевым словом **Data** должен следовать список определений через запятую. Каждое определение является выражением, по которому будет изменено значение из определенной колонки в записи изменяемого объекта. Вычисления или изменения должны производиться в соответствии с типом колонки (численным, строковым и т. п.)  
Следующая таблица приводит некоторые варианты определений для колонки:

#### Выражение

*col\_name* = *col\_name*

*col\_name* = *value*

*col\_name* = **Proportion**(*col\_name*)

#### Эффект

Содержимое колонки не меняется. Каждый объект, полученный в результате, в своей записи имеет то же значение, что и объект, из которого он был получен.

MapBasic помещает значение *value* в поле записи объекта. Если тип колонки символьный, то *value* должно быть строкой. Если тип колонки численный, то *value* должно быть числом.

Используется только для численных колонок. MapBasic пропорционально размерам объектов разделяет значение в колонке *col\_name* между записями с объектами, полученными в результате.

Список **Data** может состоять из определений для всех колонок таблицы. Если в списке определены не все колонки, то MapBasic разместит пустые значения в неописанные поля новой записи.

## Оператор Objects Split

---

Если в операторе не используется предложение **Data**, то MapBasic заполнит все поля записи объектов нулевыми и пустыми значениями.

### Примеры:

В результате следующего оператора **Objects Split**, который не использует предложение **Data**, все записи, к которым присоединены объекты, получают пустые или нулевые значения.

#### **Objects Split Into Target**

Следующий оператор **Objects Split** имеет предложение **Data**, которое задает выражения для трех колонок ("State-Name", "Pop-1990" и "Med-Inc-80"). Первая часть предложения **Data** присваивает строку "подразделение" колонке "State\_Name"; то есть строка "подразделение" будет помещена в колонку "State\_Name" для каждого объекта, являющегося результатом разрезания. Далее в предложении **Data** определяется, что население из колонки "Pop\_1990" пропорционально разделяется между результатами разрезания, а значение "Med\_Inc\_80" сохраняется для всех объектов.

#### **Objects Split Into Target**

```
Data
  State_Name = "подразделение",
  Pop_1990 = Proportion( Pop_1990 ),
  Med_Inc_80 = Med_Inc_80
```

## Оператор **OnError**

### Назначение:

Объявляет процедуру обработчика ошибок.

### Синтаксис:

```
OnError Goto {label | 0 }
```

где

*label* – метка в тексте некоторой подпрограммы или функции.

### Предупреждение:

Вы не можете использовать оператор **OnError** в окне MapBasic.

### Описание:

Оператор **OnError** используется либо для запуска процедуры-обработчика ошибок, если ошибка имела место, либо для отмены обработки ошибок (форма **OnError Goto 0**). Процедура-обработчик ошибок представляет собой группу операторов, которые выполняются в случае ошибки.

В отличие от стандартных версий BASIC оператор **OnError** в MapBasic пишется в одно слово.

Оператор **OnError Goto label** объявляет, что операторы после метки *label* являются обработчиком ошибок, и, если один из операторов, следующих за **OnError Goto label**, вернет код ошибки, то MapBasic передаст выполнение программы метке *label*. Предполагается, что операторы должны обработать конфликтную ситуацию, возникшую в результате ошибки, так, чтобы она не повлияла на корректность выполнения программы, или предупредить пользователя о случившейся ошибке, или то и другое.

Заметим, что если Ваша программа имеет обработчик ошибок, то Вы должны перед оператором с меткой *label* расположить оператор управления выполнением программы (например, **Exit Sub** или **End Program**). Это не позволяет программе передать управление процедуре-обработчику без наличия ошибки. Смотрите пример ниже.

Оператор **OnError Goto 0** отменяет установленный до этого обработчик ошибок. Если ошибка происходит в программе, где нет обработчика ошибок или он отменен, то MapBasic выводит на экран окно сообщения об ошибке и прекращает выполнение программы.

Операторы обработчика ошибок могут располагаться в отдельной процедуре или функции. Так, sub-процедуру обработчика ошибок можно определить следующим оператором:

```
OnError Goto recover
```

(при этом подразумевается, что в этой процедуре есть метка "recover"). Если после выполнения такого оператора **OnError** процедура выполнит оператор **Call** и перейдет в другую sub-процедуру, то обработчик с именем "recover" не будет способен реагировать на ошибку, пока действует другая процедура. Это происходит потому, что каждая метка локальна по отношению к процедуре и функции, в которой она задана. Этот прием позволяет каждой функции и каждой процедуре сопоставить собственный обработчик ошибок.

За ошибки, возникшие в процессе обработки других ошибок, отвечает Ваш программист на MapBasic.

### Пример:

```
OnError GoTo no-states  
Open Table "states"
```

## Оператор OnError

---

```
OnError GoTo no-cities
Open Table "cities"

Map From cities, states
after-mapfrom:
  OnError GoTo 0
  '
  ' ...
  '
End Program

no-states:
  Note "Не может быть открыта таблица States...
        окно Карты не будет открыто."
  Resume after-mapfrom

no-cities:
  Note "Данные о расположении городов недоступны..."
  Map From states
  Resume after-mapfrom
```

### Смотрите также:

`Err( )`, `Error`, `Error$( )`, `Resume`

## Оператор Open File

### Назначение:

Открывает файл для операций ввода/вывода.

### Синтаксис:

```
Open File filespec
[ For { Input | Output | Append | Random | Binary } ]
[ Access { Read | Write | Read Write } ]
As [#] filenum
[ Len = recordlength ]
[ ByteOrder { LOWHIGH | HIGHLOW } ]
[ CharSet char_set ]
[ Filetype macfiletype ]
```

где

*filespec* – строка, содержащая имя файла;

*filenum* – целочисленный номер, который будет присвоен файлу вплоть до завершения работы с ним (используется в операторах **Get** и **Put**);

*recordlength* – число символов в одной записи (включая символ конца строки) для доступа к файлу в режиме **Random**;

*char\_set* – определяет кодировку символов в файле;

*macfiletype* – тип файла, только для использования в операционной системе Macintosh (например, "MIwo").

### Предупреждение:

Вы не можете использовать оператор **Open File** в окне MapBasic.

### Описание:

Оператор **Open File** открывает текстовый файл для операций ввода/вывода прикладной программой. MapBasic может считывать данные из файла или записывать в него только после открытия файла. Для операций ввода/вывода используются операторы **Get**, **Put**, **Input #**, **Print #** и **Write #**.

В MapBasic различаются понятия "файл" и "таблица". Под таблицей понимается база данных MapInfo, данные которой можно показать в окнах Списка и Карты. MapBasic применяет одни команды к таблицам (например, **Open Table**, **Fetch**, **Select**), а другие команды к файлам, которые таблицами не являются.

Предложение **For** задает режим доступа к данным файла: режим последовательного доступа, режим произвольного доступа или режим бинарного доступа. Если предложения **For** нет в операторе, то для открытия файла используется режим произвольного доступа.

### Файлы, открытые в режиме последовательного доступа

Если Вы собираетесь читать текст из файла, записи которого имеют разную длину (например, одна строка имеет 55 символов, а следующая – 72 и т. д.), то Вам надо использовать режим последовательного доступа. Для задания этого режима в предложении **For** используется ключевое слово **Input**, **Output** или **Append**.

Если использовано предложение **For Input**, то для чтения Вы можете использовать операторы **Input #** и **Line Input #**.

Если использовано предложение **For Output** или предложение **For Append**, то для записи в файл Вы можете использовать операторы **Print #** и **Write #**.

Если Вы используете предложение **For Input**, то в предложении **Access** Вы можете использовать только ключевое слово **Read**. Аналогично, с предложением **For Output** может использоваться в предложении **Access** только ключевое слово **Write**.

Предложение **Len** не должно использоваться в операторе, если задается режим последовательного доступа.

### Файлы, открытые в режиме произвольного доступа

Если Вы собираетесь читать текст из файла, записи которого имеют одинаковую длину (например, каждая строка по 80 символов длиной), то Вы должны открыть файл в режиме произвольного доступа. Для задания этого режима в предложении **For** используется ключевое слово **Random**.

Для режима произвольного доступа необходимо задать длину записи в предложении **Len = recordlength**. Величина в параметре *recordlength* должна задать количество символов одной записи, включая символы конца записи, например, пара символов "возврат каретки" и "новая строка".

Для режима произвольного доступа в предложении **Access** Вы можете использовать все комбинации ключевых слов: **Read**, **Write** или **Read Write**. Для чтения из файла и записи в файл, открытый в режиме произвольного доступа, используются операторы **Get** и **Put**.

### Файлы, открытые в режиме бинарного доступа

Если файл открыт в бинарном режиме, то MapBasic конвертирует величину переменной MapBasic в бинарную величину в случае записи и наоборот в случае чтения. Хранение численных данных в бинарном файле более компактно, чем хранение бинарных данных в текстовом файле. Но бинарный файл нельзя показывать и распечатывать как текстовый.

Для открытия файла в бинарном режиме используется предложение **For Binary**.

Для бинарного режима в предложении **Access** Вы можете использовать все комбинации ключевых слов: **Read**, **Write** или **Read Write**. Для чтения из файла и записи в файл, открытый в режиме бинарного доступа, используются операторы **Get** и **Put**.

Предложения **Len** и **CharSet** не должны использоваться в операторе, если задается режим бинарного доступа.

### Управление порядком чтения байта

Предложение **CharSet** задает кодировку символов в файле. Параметр *char\_set* должен быть строковой константой, такой как "MacRoman" или "WindowsLatin1". Если предложение **CharSet** опущено, то MapInfo будет использовать кодировку вычислительной платформы, в которой выполняется программа. Заметим, что предложение **CharSet** используется только в том случае, если файл открывается в режимах **Input**, **Output**, или **Random**. Читайте описание стандартного предложения **CharSet** для более подробной информации.

Если Вы открыли файл в режиме произвольного или бинарного доступа (**Random** или **Binary**), предложение **ByteOrder** задает, как число представлено в файле. В разных вычислительных платформах используется разный порядок байтов для представления данных: DOS-компьютеры используют порядок LOW HIGH (в порядке возрастания разрядов), в компьютерах Macintosh и в рабочих станциях UNIX бинарные данные хранятся в порядке HIGH LOW (в порядке убывания разрядов).

Если прикладная программа действует только в пределах одной вычислительной платформы, то Вам не надо беспокоиться о порядке разрядов байтов в файле. Но, если Вам необходимо читать из бинарного файла или писать в бинарный файл, который был создан или будет использоваться в

другой платформе, то Вам придется контролировать порядок расположения байтов с помощью предложения **ByteOrder**.

Допустим, что Ваш файл был создан в операционной системе DOS и имеет порядок байтов LOW HIGH (этот порядок используется по умолчанию в DOS). Если Вы намереваетесь использовать этот файл в Macintosh, то Ваш оператор **Open File** должен включать в себя предложение **ByteOrder LOWHIGH**, иначе в Macintosh бинарный файл будет прочитан в порядке HIGH LOW.

### Типы файлов системы Macintosh

В Macintosh, оператор **Open File** может включать предложение **Filetype** для задания типа файла в системе Macintosh. Каждый файл в Macintosh имеет тип, состоящий из четырех букв (например, "MIwo" для Рабочих Наборов), скрытый от пользователя и не зависящий от имени файла. Тип файла управляет представлением его в Macintosh Finder, поэтому судить о нем пользователь может только по иконке.

Предложение **Filetype** игнорируется всеми вычислительными платформами, за исключением Macintosh. Следующая таблица приводит некоторые четырехбуквенные последовательности, используемые для определения типа файла:

Значение. macfile-type	Тип файла
"MIwo"	Рабочий Набор MapInfo.
"MIta"	Таблица MapInfo.
"MIap"	Программа, написанная на MapBasic (откомпилированная).
"TEXT"	Текстовый файл (например, этот тип используется MapInfo для файла сокращений).
"MIpr"	Файл настройки MapInfo.
"MIhe"	Файл справочника MapInfo.
"MIsy"	Файл с символами и линиями MapInfo.
"MIdb"	База данных MapInfo (файлы, являющиеся компонентами таблицы: файлы с неграфическими данными, файлы с географическими данными, индексные файлы).

**Замечание:** большие и маленькие буквы при задании типа различаются.

### Примеры:

```
Open File "cxdata.txt" For INPUT As #1
Open File "cydata.txt" For RANDOM As #2 Len=42
Open File "czdata.bin" For BINARY As #3
```

### Смотрите также:

Close File, EOF( ), Get, Input #, Print #, Put, Open Table, Write #

### Оператор Open Report

#### Назначение:

Загружает отчет в модуль **Crystal Report Designer**

#### Синтаксис:

**Open Report** *reportfilespec*

*reportfilespec* - это полный путь для существующего файла отчета.

#### Смотри также:

**Create Report From Table**



## Оператор Open Table

### Назначение:

Открывает таблицу MapInfo.

Использует новые ключевые слова **Password** и **NoIndex** для таблиц Access.

### Синтаксис:

```
Open Table filespec [ As tablename ]  
[ Hide ] [ ReadOnly ] [ Interactive ] [ Password pwd ] [ NoIndex ] [ View Automatic ] [ Deny-  
Write ]
```

*filespec* – строковая величина, задает таблицу MapInfo;

*tablename* – имя, под которым открывается таблица;

*pwd* – пароль на уровне базы данных, определяемый при включении защиты базы данных.

Применяется только для таблиц Access.

### Описание:

Оператор **Open Table** открывает уже существующую таблицу. Эффект от этого оператора такой же, как и от команды MapInfo ФАЙЛ > ОТКРЫТЬ ТАБЛИЦУ. Таблица должна быть открыта командой или оператором прежде чем MapInfo сможет производить какие-либо действия с таблицей.

Имя файла, который будет открыт (определяемый параметром *filespec*) должен соответствовать таблице, которая уже существует; для создания новой таблицы см. оператор **Create Table**. Заметим так же, что оператор **Open Table** применяется только для таблиц MapInfo; для использования файлов других форматов, используйте операторы **Register Table** и **Open File**.

Если оператор включает предложение **As**, MapInfo открывает таблицу под именем, задаваемым параметром *tablename*, которое мы будем называть “синонимом” таблицы. Следующие операторы при обращении к таблице будут использовать этот синоним. В MapInfo синонимы открываемым таблицам задаются автоматически. Имя-синоним далее представляет эту таблицу во всех списках MapInfo. Более того, если определен синоним для таблицы, то MapBasic должен обращаться к синониму, а не к фактическому имени таблицы вплоть до закрытия этой таблицы командой ФАЙЛ > ЗАКРЫТЬ ТАБЛИЦУ или оператором **Close Table**. Назначение имени-синонима не имеет ничего общего с операцией переименования таблицы.

Если оператор включает предложение **Hide**, то имя таблицы не появится ни в каком диалоге, показывающем список открытых таблиц (например, в диалоге ФАЙЛ > ЗАКРЫТЬ ТАБЛИЦУ). Используйте предложение **Hide** если Вам надо открыть таблицу, которая останется скрытой от пользователя.

Если оператор включает предложение **ReadOnly**, пользователь не сможет редактировать таблицу.

Дополнительное ключевое слово **Interactive** дает команду MapBasic подсказать пользователю место нахождения таблицы, если она не найдена по указанному пути. Ключевое слово **Interactive** полезно в ситуации, когда Вы не знаете местонахождения нужных файлов.

Если оператор включает ключевое слово **NoIndex**, индекс MapInfo не будет встраиваться в таблицу MS Access при ее открытии.

**View Automatic** это дополнительное предложение оператора **Open Table**, позволяющее таблице MapInfo, рабочему набору или файлу приложения, ассоциированному с объектом, запускаться автоматически. Если View Automatic присутствует, то после открытия таблицы, MapInfo или добавит

ее к существующей карте или откроет новое окно карты или откроет окно списка. Особенно полезно использовать совместно с HotLinks.

**DenyWrite** это дополнительное предложение для таблиц MS Access, если оно используется, то другие пользователи не смогут редактировать таблицу. Если другой пользователь уже имеет доступ в режиме чтение-запись к таблице, команда **Open Table** выдаст ошибку.

### Открытие двух таблиц с одинаковым именем

MapInfo может открыть две отдельные таблицы, которые имеют одно и тоже имя. В этом случае, MapInfo должно открыть вторую таблицу под специальным именем, что бы избежать конфликтов. В зависимости от того, включает ли оператор **Open Table** ключевое слово **Interactive**, MapBasic или присваивает специальное имя таблице автоматически, или показывает диалог, позволяющий пользователю интерактивно выбрать специальное имя таблицы.

Например, пользователь может хранить две копии таблицы "Sites", одну копию в директории 1993 ("C:\1993\SITES.TAB") и другую, возможно более новую, в другой директории ("C:\1994\SITES.TAB"). Когда пользователь (или приложение) открывает первую таблицу Sites, MapInfo открывает таблицу под ее именем ("Sites"). Если приложение использует оператор **Open Table** для открытия второй таблицы Sites, MapInfo автоматически открывает вторую таблицу под измененным именем (например, "Sites\_2") что бы отличать ее от первой таблицы. С другой стороны, если оператор **Open Table** включает предложение **Interactive**, MapInfo откроет диалог, позволяющий пользователю выбрать для таблицы альтернативное имя.

Не смотря на то, использует ли оператор **Open Table** ключевое слово **Interactive**, в результате таблица может быть открыта с нестандартным именем. Вслед за оператором **Open Table**, вызывается функция:

```
TableInfo(0, TAB_INFO_NAME)
```

для определения имени, под которым MapInfo открыло таблицу.

### Открытие таблицы, которая уже открыта

Если таблица уже открыта, и оператор **Open Table...As** пытается заново открыть ту же таблицу под новым именем, MapBasic сгенерирует код ошибки. Одна таблица не может быть открыта под двумя различными именами одновременно.

Таким образом, если таблица уже открыта, и затем оператор **Open Table** пытается заново открыть таблицу без указания нового имени, MapBasic не будет генерировать код ошибки. Таблица просто останется открытой под ее текущим именем.

### Пример:

Следующий пример открывает таблицу STATES.TAB, затем отображает таблицу в окне Карты. Поскольку оператор **Open Table** использует предложение **As** для открытия таблицы под псевдонимом (USA), в окне Карты появится заголовок "USA Map" а не "States Map."

```
Open Table "States" As USA  
Map From USA
```

Следующий пример показывает вызов функции **TableInfo( )** после оператора **Open Table**. Поскольку таблица с таким же именем (States) уже открыта, в момент запуска программы, MapBasic откроет "C:STATES.TAB" под псевдонимом (например, "STATES\_2"). Функция **TableInfo( )** вызовет псевдоним, под которым открыта таблица "C:STATES.TAB".

```
Include "MAPBASIC.DEF"  
Dim s_tab As String
```

```
Open Table "C:states"  
s_tab = TableInfo(0, TAB_INFO_NAME)  
Browse * From s_tab  
Map From tab
```

### Смотрите также:

Close Table, Create Table, Delete, Fetch, Insert, TableInfo( ), Update

### Оператор Open Window

#### Назначение:

Открывает вспомогательные окна.

#### Синтаксис:

**Open Window** *window\_name*

где

*window\_name* – имя окна (например, **Ruler**) или код окна (например, WIN-RULER)

#### Описание:

Оператор **Open Window** используется для открытия вспомогательных окон в MapInfo. Например, следующий оператор открывает окно "Статистика", как если бы пользователь открыл его командой НАСТРОЙКИ > ПОКАЗАТЬ ОКНО СТАТИСТИКИ.

**Open Window Statistics**

Параметр *window\_name* должен быть именем окна или целочисленным кодом. В следующей таблице в первой колонке приводятся имена окон, а во второй – описание и имена кодов, которые установлены в файле стандартных определений MapBasic MAPBASIC.DEF.

Значения <i>window_name</i>	Описание окна и его код из файла MAPBASIC.DEF
MapBasic	Окно MapBasic (WIN-MAPBASIC)
Statistics	Окно "Статистика" (WIN-STATISTICS)
Legend	Окно "Легенда" (WIN-LEGEND)
Info	Окно "Информация" (WIN-INFO)
Ruler	Окно "Линейка" (WIN-RULER)
Help	Окно Справочной системы (WIN-HELP)
Message	Окно "Сообщение", используемое оператором <b>Print</b> (WIN-MESSAGE)

Вы не можете открыть документальное окно (Карты, Графика, Списка, Геогрупп или Отчета) при помощи оператора **Open Window**. Эти окна открываются другими способами (смотрите операторы **Map**, **Graph**, **Browse**, **Layout** и **Create Redistricter**).

#### Смотрите также:

**Close Window**, **Print**, **Set Window**

## Функция **Overlap( )**

### Назначение:

Возвращает объект, полученный в результате географического пересечения двух объектов.

### Синтаксис:

**Overlap**(*object1*, *object2*)

где

*object1* и *object2* – параметры, задающие пересекающиеся объекты, но не точечные.

### Величина, полученная в результате:

Величина типа Object.

### Описание:

Функция **Overlap( )** вычисляет географическое пересечение двух объектов (площадь перекрытия одним объектом другого), и возвращает объект, представляющий пересечение.

MapBasic передает все стили оформления объекта *object1* результирующему объекту. Если необходимо, стиль оформления объекта можно изменить на текущий в прикладной программе.

Если один из объектов является линейным (например, полилинией), а второй замкнутым (например, область), функция **Overlap( )** вернет часть линейного объекта, которая находит на площадь замкнутого.

### Смотрите также:

**AreaOverlap( )**, **Erase( )**, **Objects Intersect**

### Функция `OverlayNodes( )`

#### Назначение:

Возвращает объект, созданный на основе существующего, добавлением узлов в точках пересечения со вторым объектом.

#### Синтаксис:

**`OverlayNodes(input_object, overlay_object)`**

где

*input\_object* – объект, на основе которого будет создан результирующий, и который не может быть точечным;

*overlay\_object* – объект, пересекающий объект *input\_object* (также не может быть точечным).

#### Величина, полученная в результате:

Величина типа `Object`.

#### Описание:

Функция **`OverlayNodes( )`** возвращает объект, созданный из узлов первого плюс узлы, полученные пересечением линий или контуров объектов *input\_object* и *overlay\_object*.

Если объект *input\_object* замкнут (область, прямоугольник, скругленный прямоугольник или эллипс), то функция **`OverlayNodes( )`** вернет область. Если объект *input\_object* линейный (прямая линия, полилиния или дуга), то функция **`OverlayNodes( )`** вернет полилинию.

MapBasic передает все стили оформления объекта *input\_object* результирующему.

Для определения, прибавила ли функция **`OverlayNodes( )`** несколько узлов к тем, которые были у объекта *input\_object*, используйте функцию **`ObjectInfo( )`**. Заметим, что если объект *input\_object*, пересекающийся с другим, уже имеет узлы в точках пересечения, то функция **`OverlayNodes( )`** не будет добавлять новых узлов к имеющимся в *input\_object*, то есть результирующий объект будет состоять только из тех узлов, которые были у первого объекта.

#### Смотрите также:

**`Objects Overlay`**

## Оператор Pack Table

### Назначение:

Соответствует команде MapInfo Таблица > Изменить > Упаковать.

### Синтаксис:

**Pack Table** *table* { **Graphic** | **Data** | **Graphic Data** } [ **Interactive** ]

где

*table* – имя открытой таблицы, которая не имеет несохраненные изменения.

### Описание:

Для упаковки неграфических данных таблицы в операторе используется ключевое слово **Data**. Когда Вы сжимаете данные таким образом, MapInfo физически удаляет все строки, которые были помечены как удаленные.

Для упаковки графических объектов таблицы в операторе используется ключевое слово **Graphic**. Упаковывая графику, удаляются пустые места из .MAP-файла таблицы. Однако упаковка графических объектов несколько замедляет графические операции.

Оператор **Pack Table** может использовать одновременно и слово **Data**, и слово **Graphic**, или должен включать хотя бы одно.

Выполнение оператора **Pack Table** влечет за собой удаление таблицы из слоя Карты и, возможно, тематических и косметических объектов. Если Вы используете ключевое слово **Interactive**, то MapInfo перед упаковкой выведет диалог, предлагающий пользователю сохранить тематические или косметические объекты.

MapInfo не может сжимать таблицу, если она была изменена и эти изменения не были сохранены на диске. Для сохранения на диск таблицы используйте оператор **Commit**.

**Внимание:** Упаковка таблицы может повлиять на подписи, созданные или измененные пользователем и сохраненные в Рабочем Наборе. Это происходит потому, что в Рабочем Наборе подписи пользователя соотносятся с номером строки таблицы; операция упаковки меняет порядок записей, потому что из таблицы исключаются удаленные записи; потому после упаковки подписи могут появляться не там, где ожидалось и иметь неправильный вид. Если же Вы удаляли записи из нижней части таблицы, а подписи соотнесены с верхними записями, то упаковка не испортит подписей.

### Пример:

```
Pack Table parcels Data
```

### Функция PathToDirectory\$( )

#### Назначение:

Извлекает из полной спецификации файла имя каталога.

#### Синтаксис:

**PathToDirectory\$(filespec)**

где

*filespec* – строка, содержащая полное имя файла (маршрут + имя).

#### Величина, полученная в результате:

Строка. Величина типа String.

#### Описание:

Функция **PathToDirectory\$( )** возвращает часть полного имени, которая содержит имена диска и каталогов.

Полное имя файла может содержать имя диска, имена каталогов или папок и само имя файла. В DOS имя каталога начинается с литеры с двоеточием, обозначающими диск (например, "C:"). В Macintosh имя папки всегда предваряет имя диска (например, "Applications").

Например, в DOS для имени

**"C:\MAPINFO\DATA\WORLD.TAB"**

функция вернет строку с DOS-маршрутом "C:\MAPINFO\DATA\". В Macintosh для имени

**"HD:MAPINFO:DATA:WORLD"**

функция вернет строку "HD:MAPINFO:DATA:".

#### Пример:

```
Dim s-filespec, s-filedir As String
s-filespec = "C:\MAPINFO\DATA\STATES.TAB"
s-filedir = PathToDirectory$(s-filespec)

' переменная s-filedir теперь равна строке "C:\MAPINFO\DATA\"
```

#### Смотрите также:

**PathToFileName\$( ), PathToTableName\$( )**



## Функция PathToFileName\$( )

### Назначение:

Извлекает из полного имени файла имя файла.

### Синтаксис:

**PathToFileName\$(filespec)**

где

*filespec* – строка, содержащая полное имя файла (маршрут + имя).

### Величина, полученная в результате:

Строка. Величина типа String.

### Описание:

Функция **PathToFileName\$( )** возвращает часть полного имени, которая содержит имя файла.

Полное имя файла может содержать имя диска, имена каталогов или папок и само имя файла. В DOS имя файла может состоять из первого имени длиной не более восьми символов и трех букв расширения. Первое имя файла и расширение разделено точкой.

Например, в DOS для спецификации

**"C:\MAPINFO\DATA\WORLD.TAB"**

функция вернет строку "WORLD.TAB". В среде Macintosh в спецификации

**"HD:MapInfo:Data:World"**

имя папки – это "HD:MapInfo:Data:", а имя файла – "World".

### Пример:

```
Dim s-filespec, s-filename As String
s-filespec = "C:\MAPINFO\DATA\STATES.TAB"
s-filename = PathToFileName$(s-filespec)
'
' переменная s-filename теперь равна строке "STATES.TAB"
'
```

### Смотрите также:

**PathToDirectory\$( ), PathToTableName\$( )**

### Функция PathToTableName\$ ( )

#### Назначение:

Возвращает имя таблицы, синоним, используя полное имя файла таблицы.

#### Синтаксис:

**PathToTableName\$(filespec)**

где

*filespec* – строка, содержащая полное имя файла (маршрут + имя).

#### Величина, полученная в результате:

Строка длиной до 31 символа. Величина типа String.

#### Описание:

Получая полное имя файла с расширением .TAB, функция возвращает строку, которая может быть для этой таблицы в данный момент псевдонимом (alias). Именно такой синоним видит пользователь в строке заголовка документального окна MapInfo (например, в строке заголовка Списка).

Этот абзац отличается от текста в вета-книге, но он мне больше нравится. Для получения синонима таблицы при ее открытии MapInfo удаляет из полного имени файла имя носителя, каталога и расширение “.TAB” (для системы DOS). Любые специальные символы, такие как тире, пробелы и др. (имена файлов в Macintosh могут включать пробелы и знаки пунктуации) заменяются знаками подчеркивания (\_). Если имя файла начинается с цифры, то MapInfo вставляет знак подчеркивания в начало имени таблицы. Если результирующая строка получается длинее 31 символа, то MapInfo отсекает лишние с конца.

Заметим, что таблица может быть открыта с именем-синонимом, отличающимся от имени файла.

Программа MapBasic может открыть таблицу с именем-синонимом оператором **Open Table** с предложением **As**. Например, откроем таблицу WORLD под синонимом “Earth”:

```
Open Table "C:\MapInfo\Data\World.tab" As Earth
```

Могут быть также открыты две одноименные таблицы, расположенные в разных каталогах, и MapInfo автоматически изменит имя одной из таблиц. В таких случаях имя таблицы, возвращаемое функцией **PathToTableName\$ ( )**, может не совпадать с именем, под которым она открыта в MapInfo. Для того, чтобы определить синонимическое имя открытой таблицы, используйте функцию **TableInfo(TAB\_INFO\_NAME)**.

#### Пример:

```
Dim filespec, tablename As String
filespec = "C:\MAPINFO\DATA\RUSSIA.TAB"
tablename = PathToTableName$(filespec)
' Переменная tablename теперь равна строке "RUSSIA"
```

#### Смотрите также:

**PathToDirectory\$ ( )**, **PathToFileName\$ ( )**, **TableInfo ( )**

## Предложение Pen

### Назначение:

Задаёт стиль линии для графических объектов.

### Синтаксис:

**Pen** *pen\_expr*

где

*pen\_expr* – выражение, результат которого есть величина типа Pen (или переменная типа Pen, или вызов функции, возвращающей такую величину, например, **MakePen** (*width, pattern, color*)).

### Описание:

Предложение **Pen** не является отдельным оператором, а входит в состав некоторых операторов, в которых необходимо задавать стиль линии для некоторых графических объектов. Стиль линии представляет собой набор из атрибутов толщины линии, типа линии и цвета.

Это предложение используется, например, в операторе **Create Line**, который создаёт новый объект типа "линия". Предложение **Pen** задаёт стиль для нового объекта. Если оператор не использует это предложение, то будет использована текущая настройка этого стиля в MapInfo.

Параметр *pen\_expr* должен быть величиной типа Pen и может задаваться переменной или выражением соответствующего типа:

**Pen** *pen-var*

или может задаваться вызовом функций **CurrentPen**( ) или **MakePen**( ), которые возвращают величины типа Pen:

**Pen** **MakePen**(1, 2, BLUE)

В некоторых операторах (например, **Set Map**) после слова **Pen** стиль задаётся непосредственно набором из трёх целочисленных параметров (*width, pattern* и *color*), например:

**Pen**(1, 2, BLUE)

Заметим, что некоторые операторы MapBasic используют выражения тип Pen как параметры без ключевого слова. Одним из примеров является оператор **Alter Object**.

В следующей таблице приводится описание параметров стиля линии:

Компонента стиля	Описание
width	Толщина линии в точках, величина типа Integer, от 1 до 7 включительно. Если Вы хотите иметь невидимую линию в объекте, то задайте нулевую толщину. При этом, если параметр будет равен 0, то параметр типа линии <i>pattern</i> должен быть равен 1 (единице).
pattern	Тип линии, величина типа Integer, от 1 до 77 (смотрите таблицу ниже). Значение 1 обозначает невидимую линию.
color	Цвет линии в системе RGB, величина типа Integer. Смотрите описание функции <b>RGB</b> ( ).

## Предложение Pen

В следующей таблице показаны образцы линий и их номера:

01		31		61		91	
02		32		62		92	
03		33		63		93	
04		34		64		94	
05		35		65		95	
06		36		66		96	
07		37		67		97	
08		38		68		98	
09		39		69		99	
10		40		70		100	
11		41		71		101	
12		42		72		102	
13		43		73		103	
14		44		74		104	
15		45		75		105	
16		46		76		106	
17		47		77		107	
18		48		78		108	
19		49		79		109	
20		50		80		110	
21		51		81		111	
22		52		82		112	
23		53		83		113	
24		54		84		114	
25		55		85		115	
26		56		86		116	
27		57		87		117	
28		58		88		118	
29		59		89			
30		60		90			

### Пример:

```

Include "MAPBASIC.DEF"
Dim cable As Object
Create Line
    Into Variable cable
    (73.5, 42.6) (73.67, 42.9)
    Pen MakePen(1, 2, BLACK)

```

### Смотрите также:

Alter Object, Create Line, Create Pline, CurrentPen( ), MakePen( ), RGB( ), Set Style

## Функция PenPattern()

### Назначение:

Возвращает номер стиля линии.

### Синтаксис:

**PenPattern** ( *pattern*, *isinterleaved* )

*pattern* - это целое значение, соответствующее номеру стиля линии.

*isinterleaved* - это булевская величина, истинная, если линии перекрещивающиеся, и ложная, если линии накладываются.

### Описание:

Функция **PenPattern** возвращает стиль линии для заданного номера стиля с и ли без установок для пересекающихся линий. Эта функция возвращает переменную типа *целое*, соответствующую определенному стандартному типу перекрещивающихся или наложенных линий. Используйте эту функцию для получения информации о стиле линии из перечня имеющихся стилей перекрещивающихся или наложенных стилей линий.

### Пример:

```
Include "MAPBASIC.DEF"  
  
Dim Cable As Object  
  
Create Line  
    Into Variable Cable  
    (73.5, 42.6) (73.67, 42.9)  
    Pen MakePen (3, PenPattern (65, True), Red)
```

### Смотри также:

**CurrentPen()**, **MakePen()**, **Pen**, **StyleAttr()**

### Функция PenWidthToPoints()

#### Назначение:

Функция **PenWidthToPoints** возвращает размер в пунктах для линии данной ширины.

#### Синтаксис:

**PenWidthToPoints** ( *penwidth* )

*penwidth* - это целое значение, большее чем 10, представляющее ширину линии.

#### Возвращаемое значение:

Вещественное.

#### Описание:

Функция **PenWidthToPoints** берет ширину линии и возвращает ее значение в пунктах. Ширина линии для стиля линии может быть возвращена с помощью функции **StyleAttr**. Ширина линии, возвращаемая функция **StyleAttr** может быть в пунктах или пикселах. Ширина шинии, меньшая чем 10 задается в пикселах. Любая ширина от 10 и более задается в пунктах. **PenWidthToPoints** будет возвращать значения только для линий с шириной, заданной в пунктах. Чтобы определить, в каких величинах задана ширина линии, в пунктах или пикселах, используйте функцию **IsPenWidthPixels**.

#### Пример:

```
Include "MAPBASIC.DEF"

Dim CurPen As Pen
Dim Width As Integer
Dim PointSize As Float

CurPen = CurrentPen()

Width = StyleAttr(CurPen, PEN_WIDTH)

If Not IsPenWidthPixels(Width) Then
    PointSize = PenWidthToPoints(Width)
End If
```

#### Смотри также:

**CurrentPen()**, **IsPenWidthPixels()**, **MakePen()**, **Pen**, **PointsToPenWidth()**, **StyleAttr()**

### Функция PointsToPenWidth()

#### Назначение:

Функция **PointsToPenWidth** возвращает ширину линии для заданного в размерах - пунктах.

#### Синтаксис:

**PointsToPenWidth ( *pointsize* )**

*pointsize* - это вещественное значение, показывающее значение в десятых долях пунктов.

#### Возвращаемое значение:

Малое целое.

#### Описание:

Функция **PointsToPenWidth** берет значения в десятых долях пунктов и конвертирует их в ширину линии.

#### Пример:

```
Include "MAPBASIC.DEF"  
Dim Width As Integer  
Dim p_bus_route As Pen  
Width = PointsToPenWidth(1.7)  
p_bus_route = MakePen(Width, 9, RED)Смотри также:
```

#### Смотрите также:

**CurrentPen(), IsPenWidthPixels(), MakePen(), Pen, PenWidthToPoints(), StyleAttr()**

### Функция **Perimeter( )**

#### Назначение:

Возвращает периметр графического объекта.

#### Синтаксис:

**Perimeter**(*obj\_expr*, *unit\_name*)

где

*obj\_expr* – объектное выражение;

*unit\_name* – строковая величина, задающая единицы измерения расстояний (например, "km" – километры)

#### Величина, полученная в результате:

Вещественное число. Величина типа Float.

#### Описание:

Функция **Perimeter( )** вычисляет периметр объекта, определенного выражением *obj\_expr*. Периметр объекта может вычисляться только для следующих типов: "эллипс", "прямоугольник", "скругленный прямоугольник", "полигон". Для других типов объектов результатом функции будет ноль.

Функция возвращает периметр в единицах, заданных вторым параметром. Полный список строковых значений, определяющих единицы измерения расстояний, представлен в описании оператора **Set Distance Units**.

Функция **Perimeter( )** для объекта типа "скругленный прямоугольник" вычисляет периметр приблизительно, как если бы объект был преобразован в простой прямоугольник.

#### Примеры:

Следующий пример показывает применение функции **Perimeter( )** для определения периметра географического объекта.

```
Dim perim As Float
Open Table "world"
Fetch First From world
perim = Perimeter(world.obj, "km")
' Переменная perim теперь равна
' периметру полигона, присоединенного
' к первой записи таблицы World.
```

Функцию **Perimeter( )** можно использовать в операторе **Select** для формирования SQL-запроса. Например, извлечем из таблицы RUSSIA некоторую информацию и поместим ее во временную таблицу, которую назовем "Results". В одной из колонок временной таблицы будут содержаться протяженности границ областей, краев и республик, входящих в состав России.

```
Open Table "russia"
Select state, name, Perimeter(obj, "km")
From russia
Into results
```

#### Смотрите также:

**Area( )**, **ObjectLen( )**, **Set Distance Units**



## Оператор Print

### Назначение:

Печатает пояснительный текст или текст сообщения из программы в окне "Сообщение".

### Синтаксис:

**Print** *message*

где

*message* – строковая величина.

### Описание:

Оператор **Print** используется для вывода текста в окне "Сообщение".

Это окно является одним из вспомогательных окон в MapInfo и предназначено для вывода информации из программ, написанных на MapBasic. Эти сообщения можно использовать для комментирования действий программы без ее остановки. Например, "Запись удалена".

Программа MapBasic может перед выполнением оператора **Print** открыть окно сообщений, используя оператор **Open Window**, и назначить шрифт для текста, размер окна и положение его на экране при помощи оператора **Set Window**. Если оператор **Print** выполняется, когда окно сообщений закрыто, то MapBasic откроет его автоматически.

Оператор **Print** выполняет действия, похожие на действия оператора **Note**, с той разницей, что последний выводит сообщение в диалоговом окне, и пока пользователь не закроет это окно, выполнение программы будет приостановлено.

Выполнение программы после оператора **Print** продолжается, а каждый следующий оператор **Print** будет печатать текст с новой строки в уже открытом окне "Сообщение". Если все окно сообщений будет заполнено или строка будет длиннее ширины окна, то пользователь может прокручивать окно в горизонтальном и вертикальном направлениях.

Чтобы очистить окно "Сообщение" перед выводом сообщения, используйте символ прогона листа (ASCII-код 12):

```
Print Chr$(12) 'Это сообщение очищает поле окна
```

Для начала новой строки в тексте сообщения используйте символ возврата каретки (ASCII-код 10).

Следующий оператор **Print** выводит сообщение в две строки:

```
Print "Слой карты:" + Chr$(10) + " World, Capitals"
```

Оператор **Print** преобразует табуляцию (ASCII-код 09) в символ пробела (ASCII-код 32).

### Пример:

Откроем окно "Сообщение", назначим шрифт Helvetica, жирный, размером в 10 пунктов, синего цвета. Назначим также размер окна, 3 на 1 дюйм, расположение ниже и правее на четверть дюйма от правого верхнего угла основного окна MapInfo. Теперь можно печатать:

```
Include "MAPBASIC.DEF"           ' понадобится для цвета 'BLUE'
Open Window Message              ' открываем окно...
Set Window Message
    Font ("Helv", 1, 10, BLUE)    ' назначаем шрифт...
    Position (0.25, 0.25)         ' позицию на экране...
    Width 3.0                     ' ширину окна...
    Height 1.0                    ' высоту...
Print "MapBasic-диспетчер на линии"
```

### Смотрите также:

**Ask( ), Close Window, Note, Open Window, Set Window**

### Оператор Print #

#### Назначение:

Записывает данные в файл, открытый в режиме последовательного доступа.

#### Синтаксис:

**Print #***file\_num* [, *expr* ]

где

*file\_num* – номер файла, который был задан при выполнении оператора **Open File**;

*expr* – выражение для записи в файл.

#### Описание:

Оператор **Print #** выводит данные в файл, который должен быть открыт оператором **Open File** в последовательном режиме доступа, разрешающем запись (**OUTPUT** или **APPEND**).

Параметр *file\_num* должен соответствовать номеру, с которым файл был открыт оператором **Open File**.

MapInfo записывает выражение *expr* в одну строку файла. Для записи выражений списком, каждое в отдельную строку файла, используется оператор **Write #** вместо **Print #**.

#### Смотрите также:

**Line Input #, Open File, Write #**

## Оператор PrintWin

### Назначение:

Печатает содержимое окна.

### Синтаксис:

**PrintWin** [ **Window** *window\_id* ] [ **Interactive** ]

где

*window\_id* – идентификатор окна.

### Описание:

Оператор **PrintWin** используется для вывода содержимого окна на печать.

Если используется предложение **Window**, то MapBasic будет печатать заданное окно. Если окно не задано, то напечатается содержимое активного окна.

Параметр *window\_id* должен быть идентификатором окна, который Вы можете получить при помощи функций **FrontWindow( )** и **WindowInfo( )**.

Если оператор включает ключевое слово **Interactive**, MapBasic показывает стандартный диалог "Печать". Без этого ключевого слова печать будет производиться автоматически, без диалога с пользователем, используя текущие установки печати.

В Windows при печати изображений из окна Карты на принтере, использующем PostScript, если Карта содержит растровые изображения, иногда получаются плохие результаты. Точка растра печатается маленьким черным прямоугольником. Вы можете использовать настройки принтеров в Windows Control Panel, установив режим "Conform to Adobe Document Structuring Convention".

В системе Macintosh оператор **PrintWin** использует последние установки для печати, которая производилась перед выполнением оператора. Если перед оператором ничего не печаталось, то используются начальные установки печати, задающиеся в начале сеанса работы в Macintosh.

### Пример:

```
Dim win-id As Integer
Open Table "world"
Map From world
win-id = FrontWindow( )
'
' Зная идентификатор окна,
' теперь можно печатать карту WORLD
'
PrintWin Window win-id Interactive
```

### Смотрите также:

**FrontWindow( )**, **Run Menu Command**, **WindowInfo( )**

## Функция PrismMapInfo( )

### Назначение

Возвращает настройки окна Карты-призмы.

### Синтаксис

**PrismMapInfo( *window\_id* , *attribute* )**

*window\_id* - целое, идентификатор окна.

*attribute* - целочисленный код, определяющий тип возвращаемых данных.

### Возвращаемые значения

Вещественные, логические или строковые в зависимости от параметра атрибута.

### Описание

Функция **PrismMapInfo()** возвращает информацию об окне Карты-призмы.

Параметр *window\_id* определяет, какое окно Карты-призмы обрабатывается функцией. Чтобы получить идентификатор окна, вызовите функцию **FrontWindow()** сразу же после открытия окна или вызовите функцию **WindowID()** в любой момент после создания окна.

Существует несколько числовых атрибутов, которые **PrismMapInfo()** может вернуть для каждого окна Карты-призмы. Параметр атрибута сообщает функции **PrismMapInfo()**, какие данные об окне Карты возвращаются. Параметр атрибута должен быть одним из кодов, представленных в следующей таблице; коды определены в файле **MAPBASIC.DEF**.<sup>1</sup>

#### Атрибут

#### Возвращаемое значение

PRISMMAP_INFO_SCALE	Вещественное, масштабный фактор Карты-призмы.
PRISMMAP_INFO_BACKGROUND	Целое, цвет фона (см. функцию RGB).
PRISMMAP_INFO_LIGHT_X	Вещественное, координата X источника освещения.
PRISMMAP_INFO_LIGHT_Y	Вещественное, координата Y источника освещения.
PRISMMAP_INFO_LIGHT_Z	Вещественное, координата Z источника освещения.
PRISMMAP_INFO_LIGHT_COLOR	Целое, цвет источника освещения (см. функцию RGB).
PRISMMAP_INFO_CAMERA_X	Вещественное, координата X камеры.
PRISMMAP_INFO_CAMERA_Y	Вещественное, координата Y камеры.
PRISMMAP_INFO_CAMERA_Z	Вещественное, координата Z камеры.
PRISMMAP_INFO_CAMERA_FOCAL_X	Вещественное, координата X фокальной точки камеры.

PRISMMAP_INFO_CAMERA_FOCAL_Y	Вещественное, координата Y фокальной точки камеры.
PRISMMAP_INFO_CAMERA_FOCAL_Z	Вещественное, координата Z фокальной точки камеры.
PRISMMAP_INFO_CAMERA_VU_1	Вещественное, первое значение параметра ViewUp.
PRISMMAP_INFO_CAMERA_VU_2	Вещественное, второе значение параметра ViewUp.
PRISMMAP_INFO_CAMERA_VU_3	Вещественное, третье значение параметра ViewUp.
PRISMMAP_INFO_CAMERA_VPN_1	Вещественное, первое значение параметра ViewPlane.
PRISMMAP_INFO_CAMERA_VPN_2	Вещественное, второе значение ViewPlane.
PRISMMAP_INFO_CAMERA_VPN_3	Вещественное, третье значение параметра ViewPlane.
PRISMMAP_INFO_CAMERA_CLIP_NEAR	Вещественное, ближняя режущая плоскость камеры.
PRISMMAP_INFO_CAMERA_CLIP_FAR	Вещественное, дальняя режущая плоскость камеры.

### Пример

Распечатываем все стандартные переменные, определенные для окна Карты-призмы:

```
include "Mapbasic.def"
Print "PRISMMAP_INFO_SCALE: " + PrismMapInfo(FrontWindow(),
    PRISMMAP_INFO_SCALE)
Print "PRISMMAP_INFO_BACKGROUND: " + PrismMapInfo(FrontWindow(),
    PRISMMAP_INFO_BACKGROUND)
Print "PRISMMAP_INFO_UNITS: " + PrismMapInfo(FrontWindow(),
    PRISMMAP_INFO_UNITS)
Print "PRISMMAP_INFO_LIGHT_X : " + PrismMapInfo(FrontWindow(),
    PRISMMAP_INFO_LIGHT_X )

Print "PRISMMAP_INFO_LIGHT_Y : " + PrismMapInfo(FrontWindow(),
    PRISMMAP_INFO_LIGHT_Y )
Print "PRISMMAP_INFO_LIGHT_Z: " + PrismMapInfo(FrontWindow(),
    PRISMMAP_INFO_LIGHT_Z)
Print "PRISMMAP_INFO_LIGHT_COLOR: " + PrismMapInfo(FrontWindow(),
    PRISMMAP_INFO_LIGHT_COLOR)
Print "PRISMMAP_INFO_CAMERA_X: " + PrismMapInfo(FrontWindow(),
    PRISMMAP_INFO_CAMERA_X)
Print "PRISMMAP_INFO_CAMERA_Y : " + PrismMapInfo(FrontWindow(),
    PRISMMAP_INFO_CAMERA_Y )
Print "PRISMMAP_INFO_CAMERA_Z : " + PrismMapInfo(FrontWindow(),
    PRISMMAP_INFO_CAMERA_Z )
```

---

1.

## Функция PrismMapInfo( )

---

```
Print "PRISMMAP_INFO_CAMERA_FOCAL_X: " + PrismMapInfo(FrontWindow(),
    PRISMMAP_INFO_CAMERA_FOCAL_X)
Print "PRISMMAP_INFO_CAMERA_FOCAL_Y: " + PrismMapInfo(FrontWindow(),
    PRISMMAP_INFO_CAMERA_FOCAL_Y)
Print "PRISMMAP_INFO_CAMERA_FOCAL_Z: " + PrismMapInfo(FrontWindow(),
    PRISMMAP_INFO_CAMERA_FOCAL_Z)
Print "PRISMMAP_INFO_CAMERA_VU_1: " + PrismMapInfo(FrontWindow(),
    PRISMMAP_INFO_CAMERA_VU_1)
Print "PRISMMAP_INFO_CAMERA_VU_2: " + PrismMapInfo(FrontWindow(),
    PRISMMAP_INFO_CAMERA_VU_2)
Print "PRISMMAP_INFO_CAMERA_VU_3: " + PrismMapInfo(FrontWindow(),
    PRISMMAP_INFO_CAMERA_VU_3)

Print "PRISMMAP_INFO_CAMERA_VPN_1: " + PrismMapInfo(FrontWindow(),
    PRISMMAP_INFO_CAMERA_VPN_1)
Print "PRISMMAP_INFO_CAMERA_VPN_2: " + PrismMapInfo(FrontWindow(),
    PRISMMAP_INFO_CAMERA_VPN_2)
Print "PRISMMAP_INFO_CAMERA_VPN_3: " + PrismMapInfo(FrontWindow(),
    PRISMMAP_INFO_CAMERA_VPN_3)
Print "PRISMMAP_INFO_CAMERA_CLIP_NEAR: " + PrismMapInfo(FrontWindow(),
    PRISMMAP_INFO_CAMERA_CLIP_NEAR)
Print "PRISMMAP_INFO_CAMERA_CLIP_FAR: " + PrismMapInfo(FrontWindow(),
    PRISMMAP_INFO_CAMERA_CLIP_FAR)
```

### Смотрите также

Оператор Create PrismMap

Оператор Set PrismMap

## Функция ProgramDirectory\$( )

### Назначение:

Возвращает название диска и маршрут, в котором была установлена рабочая версия MapInfo.

### Синтаксис:

**ProgramDirectory\$( )**

### Величина, полученная в результате:

Строка. Величина типа String.

### Описание:

Функция **ProgramDirectory\$( )** возвращает в виде строки маршрут, по которому установлена рабочая версия MapInfo.

### Пример:

```
Dim s-prog-dir As String  
s-prog-dir = ProgramDirectory$( )
```

### Смотрите также:

**HomeDirectory\$( ), SystemInfo( )**

# Оператор **ProgressBar**

### Назначение:

Показывает диалог, иллюстрирующий выполнение процесса с кнопкой остановки и шкалой выполнения.

### Синтаксис:

```
ProgressBar status_message  
    Calling handler  
    [ Range n ]
```

где

*status\_message* – строка для сообщения в диалоговом окне;

*handler* – имя процедуры-обработчика диалога;

*n* – число для правого предела шкалы.

### Предупреждение:

Вы не можете использовать оператор **ProgressBar** в окне MapBasic.

### Описание:

Оператор **ProgressBar** используется для создания диалогового окна-индикатора выполнения процесса. Такой диалог снабжен процентной шкалой и кнопкой "Отменить". Диалог сопровождает определенные действия и показывает, насколько они выполнены. Выполнение можно прервать, нажав кнопку "Отменить". Информацию о том, как завершился процесс – самостоятельно или был прерван – можно узнать при помощи функции **CommandInfo(CMD\_INFO\_DLG\_OK)**, вызвав ее после оператора **ProgressBar**.

Строковый параметр *status\_message* может задавать поясняющий текст, который будет отображен внутри диалога процесса над шкалой. Например, это может быть: "Идет процесс копирования...".

Параметр *handler* является именем процедуры-обработчика, выполняющей те действия, которые сопровождаются диалогом процесса. Процедура-обработчик является sub-процедурой на MapBasic.

Параметр *n* задает число, которое будет стоять справа у шкалы в окне диалога. Например, процедура *handler* обрабатывает 7000 строк в таблице. Вы задаете параметр *n* равным семи тысячам. Теперь шкала отображает информацию о построчном выполнении процедуры. Если предложение **Range** отсутствует, то по умолчанию устанавливается значение 100.

Когда программа выполняет оператор **ProgressBar**, MapBasic вызывает подпрограмму *handler*.

Работа, выполняемая этой процедурой, должна быть разбита на небольшие по длительности отрезки, не более нескольких секунд каждый. Это делается для того, чтобы пользователь мог нажать на кнопку "Отмена", после чего MapBasic убирает с экрана диалог, и выполнение программы передается следующему после **ProgressBar** оператору. Если же пользователь не нажимал на эту кнопку, MapBasic продолжает выполнение процедуры-обработчика *handler*. Если пользователь ни разу не нажал кнопку отмены, то процедура-обработчик благополучно завершает свое дело.

Поэтому в тексте процедуры-обработчика *handler* должны быть предусмотрены средства для разбиения процесса на небольшие отрезки, а также средства слежения за возобновлением процесса. Пока оператор **ProgressBar** работает, MapBasic периодически обращается к процедуре-обработчику до тех пор, пока пользователь не нажмет кнопку отмены, либо процедура не закончит работу. Для контроля за этими событиями поддерживается специальная переменная, имеющая имя **ProgressBar**. Если обработчик присвоит этой переменной значение "минус единица"



```
ProgressBar = -1,
```

то MapBasic прерывает процесс и убирает диалог с экрана. И, наоборот, любое положительное значение переменной ProgressBar, например,

```
ProgressBar = 50
```

используется MapBasic для отображения "процента выполнения" в диалоге. MapBasic вычисляет "процент выполнения" делением текущего значения переменной ProgressBar на значение переменной **Range**. Например, если переменной **Range** присвоено значение

```
Range 400,
```

то, если значение ProgressBar равно 100, "процент выполнения" будет равен 25% и это будет отображено в диалоге.

В выражениях, следующих за оператором **ProgressBar**, Вы можете определить причину окончания процесса, описанного в **ProgressBar**: либо нормальное завершение, либо нажатие на кнопку "Отмена". Для этого используется функция

```
CommandInfo(CMD-INFO-DLG-OK) ,
```

возвращающая значение TRUE, если процесс завершился нормально, и FALSE, если процесс был прерван пользователем.

### Пример:

Этот пример демонстрирует, как должна быть написана процедура, вызываемая оператором **ProgressBar**. Процесс в этом примере состоит из 600 итераций, это может быть однотипная обработка 600 строк таблицы. Оператор **ProgressBar** вызывает из основной процедуры sub-процедуру "write-out". Sub-процедура "write-out" обрабатывает записи в течение 2 секунд. Затем она возвращает управление, и MapBasic проверяет, не была ли нажата кнопка отмены. Если пользователь не нажимал на кнопку отмены, то MapBasic возвращает управление в процедуру-обработчик. Такая последовательность действий повторяется до тех пор, пока не будет выполнена вся задача.

```
Include "mapbasic.def"
Declare Sub Main
Declare Sub write_out

Global next_row As Integer

Sub Main
    next_row = 1
    ProgressBar "Мы записываем..." Calling write_out Range 600
    If CommandInfo(CMD_INFO_STATUS) Then
        Note "Запись завершена! Спасибо за терпение."
    Else
        Note "Запись прервана!"
    End If
End Sub

Sub write_out
    Dim start_time As Float
    start_time = Timer()
```

## Оператор ProgressBar

---

```
' Записи обрабатываются до тех пор, пока (a) задача
' не будет выполнена, или (b) пока не пройдут 2 секунды

Do While next_row <= 600 And Timer() - start_time < 2
    .....
    ''' Здесь может быть любой другой продолжительный '''
    ''' процесс, а сейчас мы просто тянем время          '''
    .....
    next_row = next_row + 1
Loop

' Далее определяется причина остановки процесса:
' нормальное окончание работы или истечение
' двухсекундного срока на одну итерацию
If next_row > 600 Then
    ProgressBar = -1      ' tell caller "Все сделано!"
Else
    ProgressBar = next_row ' tell caller "Сделано отчасти"
End If
End Sub
```

**Смотрите также:**

**CommandInfo( ), Note, Print**

## Функция Proper\$( )

### Назначение:

Возвращает строку, преобразуя все первые буквы слов в прописные, а остальные в строчные.

### Синтаксис:

**Proper\$(string\_expr)**

где

*string\_expr* – строковое выражение.

### Величина, полученная в результате:

Строка. Величина типа String.

### Описание:

Функция **Proper\$( )** возвращает строку, полученную из строки, представленной выражением *string\_expr*, преобразованием всех первых букв слов в прописные и остальных в строчные. Такое преобразование имеет смысл для собственных имен и названий.

### Пример:

```
Dim name, propername As String
name = "нижний новгород"
propername = Proper$(name)
'
' переменная propername равна "Нижний Новгород"
name = "АВВ 123"
propername = Proper$(name)
'
' переменная propername теперь равна "Абв 123"
name = "а б в г"
propername = Proper$(name)
'
' переменная propername теперь равна "А Б В Г"
```

### Смотрите также:

**LCase\$( ), UCase\$( )**

### Функция **ProportionOverlap( )**

#### Назначение:

Вычисляет процент перекрытия одного объекта другим.

#### Синтаксис:

**ProportionOverlap(*object1*, *object2*)**

где

*object1* – объект снизу (не может быть точечным или текстовым);

*object2* – объект сверху (не может быть точечным или текстовым).

#### Величина, полученная в результате:

Величина типа Float.

#### Описание:

Функция **ProportionOverlap( )** возвращает число, представляющее процент пересечения двух объектов по отношению к площади первого. Функцию можно заменить формулой **Area(Overlap(*object1*, *object2*) ) / Area(*object1*)**.

#### Смотрите также:

**AreaOverlap( )**

## Оператор Put

### Назначение:

Записывает в открытый файл содержимое переменной.

### Синтаксис:

**Put** [#]*filenum*, [*position* ], *var\_name*

где

*filenum* – номер открытого файла, присвоенный оператором **Open File**;

*position* – позиция файла для записи (не для последовательного доступа);

*var\_name* – имя переменной, значение которой будет использовано как данные для записи.

### Описание:

Оператор **Put** позволяет записывать значение переменной в файл, открытый в режиме произвольного или бинарного доступа.

Если файл был открыт оператором **Open File** в режиме последовательного доступа (**OUTPUT** или **APPEND**), то для записи в файл используйте операторы **Print #** или **Write #**.

Если файл был открыт оператором **Open File** в режиме произвольного доступа (**RANDOM**), то параметр *position* задает номер записи, в которую будет записано новое значение. Сразу после открытия файла позиция для записи – это начало (первая запись) файла. Последовательное выполнение оператора **Put** автоматически наращивает это смещение, если при этом не переопределяется значение *position*.

Если файл был открыт оператором **Open File** в режиме бинарного доступа (**BINARY**), то одновременно можно записывать только одну переменную. Порядок, по которому байты записываются в файл, зависит от Вашей вычислительной платформы (см. также предложение **Byte-Order** в описании оператора **Open File**). Количество записанных байтов зависит от типа переменной *var\_name*. Например, записывая целое число, оператор **Put** внесет четыре байта. Переменные MapBasic записываются следующим образом:

- Logical – однобайтовое значение, или 0, или другое ненулевое число.
- SmallInt – двубайтовое значение, целое число.
- Integer – четырехбайтовое значение, целое число.
- Float – восьмибайтовое число в формате IEEE.
- String – длина строки плюс один байт для нулевого значения, обозначающего конец строки.
- Date – 4 байта: SmallInt для года, байт для месяца и байт для дня.
- Другие типы – не записываются оператором Put.

параметр *position* задает смещение в файле в байтах. Сразу после открытия файла позиция для записи – это первый байт файла. Так же, как в предыдущем случае, последовательное выполнение оператора **Put** автоматически наращивает это смещение, с той разницей, что смещение происходит по байтам. При выполнении записи будет замещено столько байтов, сколько их в переменной, начиная с текущей позиции или заданной параметром *position*.

В файл, открытый в режиме **BINARY**, оператор **Put** не может записывать строки неопределенной длины; все переменные типа String должны иметь фиксированную длину. В файл, открытый в режиме **RANDOM**, оператор **Put** не может записывать строку, которая длиннее длины записи в файле.

### Смотрите также:

EOF( ), Get, Open File, Print#, Write #

## Оператор Randomize

### Назначение:

Инициализирует функцию случайных чисел в MapBasic.

### Синтаксис:

**Randomize** [ **With** *seed* ]

где

*seed* – целочисленное выражение.

### Описание:

Оператор **Randomize** включает генератор случайных чисел, результат которого используется при последующих вызовах функции **Rnd( )**. Если оператор **Randomize** не был выполнен, то серия вызовов функций **Rnd( )** будет образовывать хотя и случайную, но всегда одинаковую последовательность. Это происходит потому, что каждое новое значение функции **Rnd( )** создается из предыдущего значения случайной последовательности. Оператор **Randomize** изменяет начальное значение для функции **Rnd( )**, и потому все последующие вызовы этой функции будут образовывать другую случайную последовательность.

Таким образом, оператор **Randomize** должен сработать один раз перед первым вызовом функции **Rnd( )**, чтобы направить процесс построения случайной последовательности в случайном направлении.

Если оператор **Randomize** используется с предложением **With**, то для создания начального случайного значения используется псевдослучайный генератор с входным значением *seed*. Если предложение **With** отсутствует, для построения начального случайного числа используется таймер Вашего компьютера.

### Пример:

```
Randomize
```

### Смотрите также:

**Rnd( )**

## Функция ReadControlValue( )

### Назначение:

Читает текущую величину или состояние элемента активного диалога.

### Синтаксис:

**ReadControlValue(id\_num)**

где

*id\_num* – целочисленный идентификатор элемента диалога.

### Величина, полученная в результате:

Величина типа Integer, Logical, String, Pen, Brush, Symbol или Font. Тип величины зависит от вида элемента диалога.

### Описание:

Функция **ReadControlValue( )** возвращает текущее значение одного из элементов активного диалога. Так как функция может считывать значения только из открытого диалога, ее можно употреблять только в процедурах обработчика элементов диалога.

Параметр *id\_num* должен задавать идентификатор элемента, значение которого прочтает функция. Значение идентификатора должно быть присвоено элементу диалога в предложении **ID** в операторе **Dialog**. Если значение параметра будет равно -1 (минус единица), функция **ReadControlValue( )** вернет значение элемента, который на текущий момент был изменен последним. Чтобы точно указать, какой элемент диалога Вы хотите считать, установите в **ReadControlValue( )** индекс Integer ID с соответствующим значением. Внимание: управление диалогом не имеет уникального идентификатора ID пока Вы не включите предложение **ID** в предложение **Control** оператора **Dialog**.

Некоторые типы элементов диалога имеют несчитываемые значения (например, статические текстовые подписи). Таблица ниже показывает возможные типы значений, которые будут возвращены различными элементами диалога. Обратите внимание, что для элемента **MultiListBox** существует специальная процедура чтения текущего значения. Так как пользователь может одновременно выбрать сразу несколько строк в списке элемента **MultiListBox**, то чтение значений этого элемента требует многократного вызова функции **ReadControlValue( )**.

Элемент диалога:	Результат функции ReadControlValue( )
EditText	Строка (тип String) длиной до 32767 байт, содержащая текст из окошка; если окошко элемента многострочное, то текст может содержать символы конца строки (код 10)
CheckBox	Логическое "Да" (TRUE), если флажок установлен, и логическое "Нет" (FALSE), если сброшен
DocumentWindow	Целое, представляющее индекс HWND для управления окном. Этот HWND должен быть передан как обработчик "родительского" окна в оператор Set Next Document Parent
RadioGroup	Целое число (тип SmallInt), номер выбранной кнопки.

## Функция ReadControlValue( )

---

PopupMenu	Целое число (тип SmallInt), номер выбранной строки в списке меню.
ListBox	Целое число (тип SmallInt), номер выбранной строки в списке (1 – первая строка, 0 – ни одной)
BrushPicker	Величина типа Brush
FontPicker	Величина типа Font
PenPicker	Величина типа Pen
SymbolPicker	Величина типа Symbol
MultiListBox	<p>Целое число (тип Integer), номер выбранной строки в списке.</p> <p>Пользователь может выбрать одну или сразу несколько строк в списке <b>MultiListBox</b>. Но функция <b>ReadControlValue( )</b> может возвращать только одну величину за один вызов. Поэтому для чтения всех значений необходимо функцию вызывать несколько раз.</p> <p>Первый вызов функции дает номер первой строки, выбранной в списке. Следующий – второй, и т. д. Когда все значения будут прочитаны, функция вернет ноль. Если ноль будет получен при первом вызове, то, следовательно, в списке ничего не выбрано.</p>

### Ошибки

В результате выполнения функции могут генерироваться следующие коды ошибок:

ERR\_FCN\_ARG\_RANGE, если неправильно значение аргумента;

ERR\_INVALID\_READ\_CONTROL, если функция **ReadControlValue( )** вызвана не при активном диалоге.

### Пример

Создадим диалог, предлагающий пользователю ввести свое имя. Если после этого пользователь нажал кнопку "ОК", будет выведено приветствие (Например, "Добро пожаловать, Света!").

```
Declare Sub Main
Declare Sub okhandler
Sub Main
  Dialog
    Title "Представьтесь, пожалуйста"
    Control OKButton
      Position 135, 120      Width 50
      Title "ОК"
      Calling okhandler
    Control CancelButton
      Position 135, 100      Width 50
      Title "Отмена"
    Control StaticText
      Position 5, 10
      Title "Введите ваше имя:"
    Control EditText
      Position 55, 10        Width 160
      Value "(имярек)"
```



```
        Id 23                'arbitrary ID number
End Sub

Sub okhandler
    ' эта подпрограмма выполнится, если
    ' была нажата кнопка "ОК"
    Note "Добро пожаловать, " + ReadControlValue(23) + "!"
End Sub
```

### Смотрите также

Alter Control, Dialog, Dialog Preserve, Dialog Remove

## Оператор ReDim

### Назначение:

Изменяет размерность массива.

### Синтаксис:

**ReDim** *var\_name* (*newsize*) [, ... ]

где

*var\_name* – имя массива локальных или глобальных переменных;

*newsize* – целое число, задающее новую размерность:

в 16-битной версии Windows от 0 до 7000 включительно;

в Macintosh и 32-битных Windows от 0 до 32767 включительно.

### Описание:

Оператор **ReDim** изменяет размерность (то есть количество элементов) одного или более уже объявленных массивов. Имя массива переменных, заданное в параметре *var\_name*, должно быть до этого объявлено в операторе **Dim** или **Global**. Оператор **ReDim** не применим к переменным, которые не являются массивами.

Если параметр *newsize*, задающий новое значение размерности, не задан, то функция **ReDim** уменьшает размерность массива до нуля. Массив с нулевой размерностью занимает минимально возможное место в памяти.

Надо учитывать, что при изменении размерности массива все значения, присвоенные элементам массива, теряются.

В отличие от других BASIC-языков, MapBasic не позволяет задавать произвольный номер первого элемента массива. Другими словами, первый элемент массива в MapBasic всегда имеет номер 1 (единица).

### Пример 1:

```
Dim names-list(10) As String, cur-size As Integer
'
' Следующий оператор считывает значение
' размерности массива, оператор ReDim увеличивает
' это значение на 10
'
cur-size = Ubound(names-list)
ReDim names-list(cur-size + 10)
'
' Следующий оператор ReDim обнуляет размерность
' нашего массива. Предположительно, этот массив больше
' не понадобится и обнуление его размерности
' сэкономит нам ресурсы памяти.
'
ReDim names-list(0)
```

### Пример 2:

Оператор **ReDim** может применяться к массивам переменных сложного типа, составленным при помощи оператора **Type**, и к массивам, которые являются элементами переменных сложного типа.

```
Type customer
  name As String
  serial-nums(0) As Integer
End Type

Dim new-customers(1) As customer
'
' Сначала увеличим размерность массива new-customers
' до пяти элементов
'

ReDim new-customers(5)
'
' Теперь изменим размерность массива serial-nums,
' который является элементом массива new-customers
'

ReDim new-customers(1).serial-nums(10)
```

**Смотрите также:**

**Dim, Global, UBound( )**

# Оператор Register Table

## Назначение

Создает таблицу MapInfo Professional из списка, базы данных, текстового файла, растра или грид-файла.

## Синтаксис

```
Register Table source_file
{ Type "NATIVE" |
  Type "DBF" [ Charset char_set ] |
  Type "ASCII" [ Delimiter delim_char ] [ Titles ] [ CharSet char_set ] |
  Type "WKS" [ Titles ] [ Range range_name ] |
  Type "XLS" [ Titles ] [ Range range_name ] |
  Type "Access" Table table_name [ Password pwd ] [ CharSet char_set ] }
Type ODBC
  Connection { Handle ConnectionNumber | ConnectionString }
  Toolkit toolkitname
  Cache { On | OFF }
  Type "GRID" |
Type "RASTER"
  [ ControlPoints
    ( MapX1 , MapY1 ) ( RasterX1 , RasterY1 ),
    ( MapX2 , MapY2 ) ( RasterX2 , RasterY2 ),
    ( MapX3 , MapY3 ) ( RasterX3 , RasterY3 )
    [ , ... ]
  ]
  [ CoordSys ... ]
  Type "SHAPEFILE" [ Charset char_set ] CoordSys... [ PersistentCache { On | Off } ]
[Symbol...] [ Linestyle Pen(...)] [ Regionstyle Pen(...)] Brush(...)]
[ Interactive ]
[ Into destination_file ]
```

**ControlPoints** дополнительное предложение, но может быть определено если тип данных Grid или Raster. Если ControlPoints указываются, то надо задать как минимум 3 пары точек координат Map или Raster, которые используются для регистрации изображения. Если ControlPoints указываются, то они будут использоваться вместо любых контрольных точек, которые уже ассоциировались с изображением или ассоциировались с регистрационным файлом World.

Предложение **CoordSys** дополнительное, но может быть определено если тип данных Grid или Raster. Если CoordSys определено, то им будут переписаны любые системы координат, ассоциированные с изображением. Это полезно когда регистрируется растр, имеющий ассоциированный регистрационный файл World.

В случае работы с шейп файлами, предложение **CoordSys** является обязательным. Компилятор выдаст ошибку если это предложение будет пропущено.

Ключевое слово **Interactive** является дополнительным, но может быть определено, если тип данных это Grid или Raster. Если ключевое слово Interactive определено, пользователь получит подсказку при пропуске контрольной точки или информации о проекции. Если ключевое слово Interactive не определено, то .ТАВ файл будет генерироваться без интерактивного ввода пользователем, а сам файл будет создаваться таким образом, как и при выборе в диалоге "Открыть таблицу" команды ФАЙЛ>ОТКРЫТЬ при открытии растра.

*source\_file* - строка определяющая имя существующей базы данных, электронной таблицы или текстового файла. Если Вы регистрируете таблицу Access, этот аргумент должен идентифицировать доступную базу данных Access.

*char\_set* is - название установленного шрифта; смотрите обсуждение оператора **CharSet**.

*delim\_char* - знак разделителя между значениями полей таблицы (только для текстовых ASCII-файлов). Если в файле используется Tab в качестве разделителя, укажите 9. Если используется запятая, укажите 44.

*range\_name* - строка с именем области электронной таблицы (например, "MyTable") или с ссылкой на ячейки (например, в Excel ячейки могут быть заданы как "Sheet1!R1C1:R9C6" или как "Sheet1!A1:F9").

*table\_name* - строка, определяющая имя таблицы Access.

*pwd* - пароль на уровне базы данных, определяемый при включении защиты базы данных.

*destination\_file* - имя будущей таблицы MapInfo table (.TAB file). Строка может включать в себя путь, если же он не указан, то файл будет строиться в той же директории, где и исходный файл.

*ConnectionNumber* целое, идентифицирующее существующее соединение с базой данных.

*ConnectionString* строка используемая для связи с сервером базы данных. Смотрите описание функции **Server Connect**.

*toolkitname* это или "ODBC" или "ORAINET."

**SQLQuery** это SQL Запрос, используемый для определения таблицы MapInfo.

### Описание

Перед тем, как использовать в MapInfo файлы "неродных" форматов (например, dBASE файл), Вы должны их зарегистрировать. Оператор **Register Table** приводит к тому, что MapInfo проверяет, может ли использоваться этот формат. Далее MapInfo заготавливает для него файлы-компоненты таблицы (*filename.TAB*, и др.). Как только оператор **Register Table** создаст файлы-компоненты, Вы можете открывать таблицу и редактировать ее данные в Списке или на Карте.

Оператор **Register Table** не копирует и не изменяет оригинал файла данных. Вместо этого, он проверяет данные, определяет типы данных в колонках и создает отдельную таблицу.

Замечание: Таблица не открывается автоматически. Для открытия таблицы надо использовать оператор **Open Table**.

Каждый файл данных должен быть зарегистрирован только раз. Так как при регистрации создаются файлы-компоненты, то при следующем сеансе работы MapInfo может просто открывать зарегистрированную ранее таблицу (оператор **Open**), а не регистрировать файл заново оператор **Register Table**.

Предложение **Type** задает формат, в котором был создан файл данных. За словом **Type**, может следовать одна из следующих строк: NATIVE, DBF, ASCII, WKS, XLS или Access.

Предложение **CharSet** является стандартным предложением и задает кодировку, в которой был создан файл-источник, например, "MacRoman" или "WindowsLatin1". Если этого предложения нет в операторе, MapBasic будет использовать текущую кодировку системы, в которой Вы работаете.

Предложение **Delimiter** задает знак разделителя между значениями полей для текстовых файлов. По умолчанию принимается символ табуляции. Предложение **Titles** определяет первую строку данных как названия полей в таблице MapInfo. Предложение **Range** определяет именованную область из

исходного файла как базу данных (для электронных таблиц). Предложение **Into** задает в виде строки имя файла .TAB и расположение на диске. Если предложения в операторе нет, имя файла таблицы будет таким же, как имя файла-источника и располагаться он должен в том же каталоге. В случае, если для диска, на котором расположен файл-источник, разрешено только чтение (например, CD-ROM), то файл таблицы будет сохранен где-то в другом месте, где запись разрешена.

Оператор **Register Table** не может быть использован для построения таблиц из файлов с растровым изображением (например, Photo.GIF). Чтобы из программы создать таблицу, основанную на растровом изображении (например, Photo.GIF), нужно использовать операторы ввода/вывода, такие, как **Open File** и **Print #**. Пример построения таблицы на основе растрового файла приводится в главе 7 *Руководства пользователя MapBasic*.

**PersistentCache On** определяет сохраняются ли файлы .MAP и .ID, которые генерируются во время открытия шейп файлов, на жестком диске после закрытия таблицы или нет. Если PersistentCache установлено на Off, то эти файлы .MAP и .ID будут удалены после закрытия таблицы, и будут генерироваться каждый раз при открытии таблицы.

**Symbol (...)** предложение определяющее стиль символа, применяемого для типа точечных объектов, создаваемых из шейп файла

**Linestyle Pen (...)** предложение определяющее стиль линии, применяемый для линейных типов объектов, создаваемых из шейп файлов

**Regionstyle Pen (...)** **Brush(...)** предложение определяющее стиль линии и стиль заливки применяемый для регионов, создаваемых из шейп файлов

### Регистрация таблиц Access

При регистрации таблицы Access, MI Pro проверяет колонку счетчик с уникальным индексом. Если колонка-счетчик существует, то MI Pro регистрирует эту колонку в .TAB файле. Эта колонка только для чтения.

Если таблица Access не имеет колонки-счетчика, то MI Pro изменит таблицу Access путем добавления колонки с именем MAPINFO\_ID имеющей тип данных counter. В этом случае колонка-счетчик не будет отображаться в окне списка MapInfo.

Внимание: Не пытайтесь изменить каким-либо образом колонку-счетчик. Это может происходить только автоматически средствами самого MapInfo.

Типы данных Access транслируются в аналогичные или близкие по сути типы данных MapInfo. Специальные типы данных Access, такие как OLE объекты и бинарные поля, не будут редактируемыми в MapInfo Professional.

### Регистрация таблиц ODBC

Перед сеансом прямого доступа к таблице из удаленных баз данных, настоятельно рекомендуется открыть таблицу карты (например, canada.tab) для таблицы базы данных. Если Вы не откроете таблицу карты, то сразу будет загружена полностью таблица базы данных, а это может занять много времени.

Откройте таблицу с картой и выберите необходимый масштаб окна, показав тот кайон, для которого Вам надо подгрузить данные из удаленной базы данных. Например, если Вы хотите загрузить записи, касающиеся провинции Онтарио, то сначала откройте в окне карту самой провинции Онтарио, подобрав необходимый масштаб. В результате получится, что из базы

данных к Вам поступят только те данные, которые относятся к минимальному описывающему прямоугольнику - окну Карты, а не вся таблица. Так экономится время на трафик.

Ниже следует список известных проблем/вопросов, связанных с прямым доступом:

- Каждая таблица должна иметь одну уникальную ключевую таблицу.
- FastEdit не поддерживается.
- Для MS ACCESS, если ключ символьный, не будут показаны строки, где значение ключа меньше чем полная ширина колонки, например, если ключ это char(5) то значение 'aaaa' будет выглядеть как удаленная запись.
- Для прямого доступа, окошко ReadOnly в диалоге сохранения таблицы будет серого цвета (неактивно).
- Изменения, сделанные другим пользователем не видны пока браузер прокручивается или обновляется. Вставки, сделанные другим пользователем не видны у Вас пока: 1). Поиск в МОП не вернет новую запись или 2). Команда PACK запущена.
- Возникнут проблемы, если клиент объединил (через меню SQL Select или MapBasic) 2 или более таблиц SPATIALWARE, которые хранятся в различных системах координат. Это не эффективный путь работы (лучше сделать объединение в операторе SQL, который определяет таблицу) но все равно это остается проблемой текущей версии.
- Таблицы Oracle 7, которые индексируются десятичным полем размером более 8 байт приведут к к "вылету" MI Pro при редактировании.
- Если оператор Cache имеет значение OFF перед соединением, то будет сгенерировано сообщение об ошибке при компиляции.

### Регистрация шейп файлов

Когда регистрируются шейп файлы, то они могут открываться в MapInfo Professional исключительно в режиме "только для чтения". Поскольку шейп файл не содержит в себе информации о системе координат, надо определить предложение CoorSys. Также возможно определить стили оформления объектов для отображения в MapInfo Professional. Информация о проекции и стилях хранится как метаданные в TAB файле.

### Пример1

```
Register Table "c:\mapinfo\data\rpt23.dbf"
Type "DBF"
Into "Report23"
```

```
Open Table "c:\mapinfo\data\Report23"
```

### Пример2

```
Open Table "C:\Data\CANADA\Canada.tab" Interactive
Map From Canada
set map redraw off
Set Map Zoom 1000 Units "mi"
set map redraw on
Register Table "odbc_cancaps"
TYPE ODBC
TABLE "Select * From informix.can_caps"
CONNECTION
    DSN=ius_adak;UID=informix;PWD=informix;DATABASE=sw;HOST=adak;
    SERVER=adak_tli;SERVICE=sqlexec;PROTOCOL=onsoctcp;"
Into
    "D:\MI\odbc_cancaps.TAB"
Open Table "D:\MI\odbc_cancaps.TAB" Interactive
Map From odbc_cancaps
```

## Оператор Register Table

---

### Пример3

Регистрация полностью геопривязанного изображения (обработчик растра может вернуть как минимум 3 контрольные точки и проекцию)

```
Register Table "GeoRef.tif" type "raster" into "GeoRef.TAB"
```

### Пример4

Регистрация растра с ассоциированным World файлом, содержащим информацию о контрольных точках, но без проекции.

```
Register Table "RasterWithWorld.tif" type "raster" coordsys earth  
projection 9, 62, "m", -96, 23, 29.5, 45.5, 0, 0 into "RasterWithWorld.TAB"
```

### Пример5

Регистрация растра, который не имеет контрольной точки или информации о проекции.

```
Register Table "NoRegistration.BMP" type "raster" controlpoints (1000,2000)  
(1,2), (2000,3000) (2, 3), (5000,6000) (5,6) coordsys earth projection 9,  
62, "m", -96, 23, 29.5, 45.5, 0, 0 into "NoRegistration.tab"
```

### Пример6

Пример регистрации шейп файла.

```
Register Table "C:\Shapefiles\CNTYLN.SHP" TYPE SHAPEFILE Charset  
"WindowsLatin1" CoordSys Earth Projection 1, 33 PersistentCache Off  
linestyle Pen (2,26,16711935) Into "C:\Temp\CNTYLN.TAB"  
Open Table "C:\Temp\CNTYLN.TAB" Interactive  
Map From CNTYLN
```

### Смотрите также

Open Table, Create Table



### Оператор Reload Symbols

#### Назначение:

Открывает и перезагружает файл символов MapInfo и тем самым меняет набор символов, используемый MapInfo и показываемый в диалоге НАСТРОЙКИ > СТИЛЬ СИМВОЛА.

#### Синтаксис (версия для символов MapInfo 3.0):

**Reload Symbols**

#### Синтаксис (только для растровых символов):

**Reload Custom Symbols From** *directory*

где

*directory* – строка с маршрутом.

#### Описание:

Оператор используется в утилите SYMBOL.MBX, которая дает пользователю возможность создавать свои символы.

## Процедура RemoteMapGenHandler

### Назначение:

Имя специальной процедуры, вызываемой тогда, когда клиент OLE Automation вызывает метод MapGenHandler Automation.

### Синтаксис:

```
Declare Sub RemoteMapGenHandler  
Sub RemoteMapGenHandler  
    statement_list  
End Sub
```

*statement\_list* – список операторов MapBasic, выполняемых когда клиент OLE Automation вызывает метод MapGenHandler.

### Описание:

**RemoteMapGenHandler** это имя процедуры MapBasic специального назначения, которая вызывается через OLE Automation. Если Вы используете OLE Automation для управления MapInfo, и вызываете метод MapGenHandler, MapInfo вызывает процедуру **RemoteMapGenHandlers** любого работающего приложения MapBasic. Метод MapGenHandler – это компонента механизма MapGen Automation, представленного в MapInfo 4.1.

Метод MapGenHandler Automation берет один аргумент: строку. Внутри процедуры **RemoteMapGenHandler** Вы можете восстановить строковый аргумент с помощью вызова следующей функции ...

**CommandInfo(CMD\_INFO\_MSG)**

... и присвоить результат строковой переменной.

### Приложения MapInfo ProServer

MapInfo ProServer – это специальная версия MapInfo, которая поддерживает приложения MapInfo в среде Internet/Intranet. Процедура **RemoteMapGenHandler** обычно используется, если Вы разрабатываете приложения с использованием MapInfo ProServer, и Вам необходимо контролировать элементы MapInfo, которые Вы не можете контролировать используя метод OLE Automation.

1. В Вашей программе .MB, создайте процедуру, вызывающую RemoteMapGenHandler.
2. В процедуре RemoteMapGenHandler перечисляются операторы MapBasic, которые должны будут выполняться в Вашем ProServer-приложении.
3. Откомпилируйте приложение MapBasic (MBX-файл).
4. Сконфигурируйте ProServer-приложение так, чтобы ваш MBX-файл предназначался для сеанса.
5. В ProServer-приложении, вызовите метод OLE Automation *MapGenHandler*. Когда Вы вызываете этот метод, MapInfo вызовет процедуру RemoteMapGenHandler в MBX-файле.

Более подробная информация о MapInfo ProServer содержится в документации о ProServer.

### Пример:

В качестве примера использования RemoteMapGenHandler, смотрите программу MAPSRVR.MB, которая поставляется как компонента MapInfo ProServer.

## Процедура RemoteMsgHandler

### Назначение:

Процедура, автоматически выполняющаяся при получении командного сообщения из другой программы.

### Синтаксис:

```
Declare Sub RemoteMsgHandler  
Sub RemoteMsgHandler  
    statement_list  
End Sub
```

где

*statement\_list* – список операторов процедуры-обработчика.

### Описание:

**RemoteMsgHandler** – зарезервированное имя процедуры MapBasic, предназначенной для обслуживания связи между прикладными программами и другими программами. В среде Windows для связи между программами используется DDE (Dynamic Data Exchange), в Macintosh используется Apple Events. Однако основные принципы связи для всех платформ, где может работать программа MapInfo, одинаковы.

Если Вы запустили программу, которая имеет процедуру **RemoteMsgHandler**, то MapInfo автоматически вызовет процедуру **RemoteMsgHandler**, как только другая программа (например, работающая с базами данных) пошлет "выполняемую" команду ("DoScript" в системе Macintosh). Для получения "послания" в виде строки в процедуре **RemoteMsgHandler** используется функция **CommandInfo()**.

Для завершения программы в теле процедуры **RemoteMsgHandler** используется оператор **End Program**. При этом полностью освобождается оперативное пространство, занимаемое прикладной программой, и связь с ним будет уже невозможно установить.

### Связь между программами в среде Windows

Для налаживания связи между программами сначала создается один канал DDE-связи (динамического обмена данных) и ему назначается номер. Этот канал связи позволяет соединить две программы, работающие в среде Microsoft Windows, для пересылки информации.

Прикладная программа может быть как "клиентом" DDE-связи в пределах одного канала, так и "сервером" для другого канала, используя в последнем случае процедуру **RemoteMsgHandler**. Когда пользователь запускает программу, в которой есть такая процедура, программа не завершается после того, как выполнятся все операторы процедуры Main и других процедур, вызванных из нее.

Прикладная программа будет пребывать в режиме ожидания до тех пор, пока "клиент" (какая-нибудь другая программа в Windows) не обратится к "серверу".

В процедуре обработчика DDE-связи для прочтения сообщения в строковом представлении используется функция **CommandInfo(CMD\_INFO\_MSG)**.

В динамическом обмене данных имя программы, выступающей в роли сервера (например, "C:\MAP-BASIC\DISPATCH.MBX") должно использоваться как объект (topic) обмена.

### Связь между программами в среде Macintosh

В Macintosh связь между программами поддерживается через Apple Events. При этом внешняя программа является "клиентом" (активная роль), а прикладная программа MapBasic – "сервером"

## Процедура RemoteMsgHandler

---

(пассивная роль).

"Клиент" связи Apple Event должен посылать имя прикладной программы, находящейся в состоянии ожидания (например, "AE-Handler.MBX"), заключенной в кавычки, как первую часть послания DoScript event или же использовать ключевое слово "kobj" как тип объекта.

MapInfo автоматически вызывает **RemoteMsgHandler** процедуру всякий раз, когда клиент связи Apple Event выполняет оператор **Do Script**. Для прочтения послания в процедуре используется вызов функции:

**CommandInfo (CMD-INFO-MSG)**

### Смотрите также:

**DDEExecute, DDEInitiate( ), SelChangedHandler, ToolHandler, WinChangedHandler, Win-ClosedHandler**

## Функция RemoteQueryHandler( )

### Назначение:

Функция, которая вызывается, когда MapBasic-программа работает DDE-сервером и получает от DDE-клиента внешний “считывающий” запрос.

### Синтаксис:

```
Declare Function RemoteQueryHandler( ) As String
```

```
Function RemoteQueryHandler( ) As String
    statement_list
End Function
```

где

*statement\_list* – группа операторов, которую нужно выполнить при получении “считывающего” запроса.

### Описание:

Функция **RemoteQueryHandler( )** работает с механизмом DDE (Dynamic Data Exchange). Подробнее механизм DDE описан в главе 11 *Руководства пользователя MapBasic*.

Внешняя программа может начать сеанс DDE-связи с Вашей MapBasic-программой. При инициации связи внешняя программа использует значение “MapInfo” как имя DDE-программы и имя Вашей MapBasic-программы как DDE-тему (topic). Как только связь установится, клиент (внешняя программа) может посылать считывающие запросы к MapBasic-программе (серверу).

Для обработки таких запросов в Вашу MapBasic-программу нужно включать обработчик **RemoteQueryHandler( )**. Как только от клиента поступает считывающий запрос, MapInfo автоматически вызывает функцию **RemoteQueryHandler( )**. Следующие считывающие запросы ожидают своей очереди, т.е. выполняются после того, как **RemoteQueryHandler( )** вернет значение.

**Внимание:** DDE-клиент может считывать значения глобальных переменных MapBasic-программы без помощи **RemoteQueryHandler( )**. Если клиент выдаст считывающий запрос к глобальной переменной MapBasic, MapInfo автоматически возвратит требуемое значение вместо того, чтобы обратиться к **RemoteQueryHandler( )**.

### Пример:

В следующем примере вызывается функция **CommandInfo( )**, чтобы определить, какой объект требует DDE-клиент. Если объект называется “code1”, то действия продолжаются по одному сценарию, если нет, то по другому.

```
Function RemoteQueryHandler() As String
    Dim s_item_name As String
    s_item_name = CommandInfo(CMD_INFO_MSG)
    If s_item_name = "code1" Then
        RemoteQueryHandler = custom_function_1()
    Else
        RemoteQueryHandler = custom_function_2()
    End If
End Function
```

### Смотрите также:

**DDEInitiate( ), RemoteMsgHandler**

### Оператор Remove Cartographic Frame

#### Назначение:

Оператор **Remove Cartographic Frame** позволяет Вам удалять разделы из существующей картографической легенды, созданной оператором **Create Cartographic Legend**.

#### Синтаксис:

```
Cartographic Frame  
[ Window legend_window_id ]  
Id frame_id, frame_id, frame_id, ...
```

*legend\_window\_id*- это целое, соответствующее идентификатору окна, которое Вы можете получить при вызове функций **FrontWindow()** и **WindowId()**.

*frame\_id* - это индекс ID раздела легенды. Вы не можете использовать имя слоя. Например, три раздела легенды вполне могут иметь индексы ID 1,2 и 3.

#### Смотрите также:

**DDEdxdref Create Cartographic Legend, Set Cartographic Legend, Add Cartographic Frame, Alter Cartographic Frame**

## Оператор Remove Map

### Назначение:

Удаляет один или более слоев из окна Карты.

### Синтаксис:

**Remove Map** [ **Window** *window\_id* ]  
**Layer** *map\_layer* [, *map\_layer* ... ] [ **Interactive** ]

где

*window\_id* – идентификатор окна Карты, целое число; его можно получить функциями **FrontWindow()** или **WindowID()**.

*map\_layer* – задает слой Карты.

### Описание:

Оператор **Remove Map** удаляет один или более слоев из окна Карты. Если оператор не содержит предложения **Window** *window\_id*, оператор будет работать с тем окном, которое является окном Карты и лежит выше остальных.

Параметр *map\_layer* может быть либо целым числом, большим нуля, либо строкой с именем таблицы, либо ключевым словом **Animate**, как показано в следующих примерах.

#### Примеры

Remove Map Layer 1

Remove Map Layer "Zones"

Remove Map Layer "Zones(1)"

Remove Map Layer Animate

#### Описание примера

Единица обозначает самый верхний некосметический слой. Данный оператор удаляет из Карты слой, идущий сразу за Косметическим слоем. Если параметр *map\_layer* был бы "1, 2", то были бы удалены два верхних некосметических слоя.

Оператор удаляет слой Zones (то есть тот, который в списке слоев обозначен именем "Zones").

Оператор удаляет первый тематический слой, построенный на основе слоя Zones.

Оператор удаляет анимационный слой. Информация об анимационных слоях содержится в описании оператора **Add Map**.

Если используется ключевое слово **Interactive** и если удаление слоя ведет к потере подписей или тематических объектов, то MapInfo выведет на экран диалог, который позволит сохранить пользователю Рабочий Набор перед удалением слоя. Если ключевое слово в операторе **Interactive** опущено, то пользователь не предупреждается о потере.

Удаляя слой, который отображает данные некоторой открытой таблицы, оператор **Remove Map** не закрывает ее.

Один оператор **Remove Map** может убрать несколько слоев. Для этого задайте номера слоев, которые хотите удалить из окна, списком через запятую.

Если оператор удаляет последний некосметический слой, то MapInfo автоматически закрывает окно Карты.

### Смотрите также:

**Add Map, Map, Set Map**

### Оператор Rename File

#### Назначение:

Изменяет имя открытого файла.

#### Синтаксис:

**Rename File** *old\_filespec* **As** *new\_filespec*

где

*old\_filespec* – строка с именем файла (и, если необходимо, DOS-маршрут); файл не должен быть открыт;

*new\_filespec* – строка с новым именем файла (и, если необходимо, маршрутом).

#### Описание:

Оператор **Rename File** переименовывает файл.

Параметр *new\_filespec* задает спецификацию с новым именем файла. Если *new\_filespec* маршрут отличается от оригинального маршрута, то MapInfo перенесет файл в заданный каталог.

#### Пример:

```
Rename File "startup.wor" As "startup.bak"
```

#### Смотрите также:

**Rename Table, Save File**



## Оператор Rename Table

### Назначение:

Изменяет имя (и расположение на диске) файла таблицы MapInfo.

### Синтаксис:

**Rename Table** *table* **As** *newtablespec*

где

*table* – имя открытой таблицы;

*newtablespec* – строка с новым именем файла таблицы (и, если необходимо, маршрутом).

### Описание:

Оператор **Rename Table** позволяет изменить либо имя таблицы, либо ее расположение на диске, либо то и другое для таблицы, открытой под рабочим именем *table*.

Параметр *newtablespec* задает либо спецификацию с новым именем файла таблицы, либо только расположение на диске (например, в Windows DOS-маршрут), по которому будет перемещена таблица под старым именем.

Оператор **Rename Table** переименовывает физически все файлы, которые являются компонентами таблицы. Так как оператор работает с файлами таблицы непосредственно на диске, то результат его работы отменить нельзя. Также, если до переименования были сохранены файлы Рабочих Наборов, переименования его составляющих могут повлиять на их загрузку, так как Рабочие Наборы могут искать файлы таблиц там, где их нет.

Не следует с помощью оператора **Rename Table** назначать имена для строки заголовка таблицы. Этого можно добиться, применяя оператор **Open Table** с предложением **As..**

Оператор **Rename Table** нельзя применить к таблице, которая имеет в своем определении слово "View". Например, нельзя переименовывать таблицы улиц (стандарта StreetInfo), такие, как SF-STRTS, состоящие фактически из двух связанных таблиц, (в нашем случае SF-STRT1 и SF-STRT2). Оператор также нельзя применять к временным таблицам (Например, Запрос1).

Вы не можете переименовывать временные таблицы запросов (такие как ЗАПРОС1) или связанные таблицы. Нельзя также переименовывать таблицы с несохраненными изменениями. Перед выполнением оператора **Rename Table** в этом случае надо сохранить или отменить удаление (операторы **Commit** и **Rollback**).

### Пример:

Переименуем таблицу CASANFRA в SF-HIWAY, используя оператор **Rename Table**.

```
Open Table "C:\DATA\CASANFRA.TAB"
```

```
Rename Table CASANFRA As "SF-HIWAY.TAB"
```

В этом примере таблица переименовывается из CASANFRA в SF-HIWAY и перемещается на другой каталог.

```
Open Table "C:\DATA\CASANFRA.TAB"
```

```
Rename Table CASANFRA As "c:\MAPINFO\SF-HIWAY"
```

В этом примере переименование таблицы происходит в среде Macintosh.

```
Open Table "CASANFRA.TAB"
```

```
Rename Table CASANFRA As "LDisk:Maps:Hiways"
```

### Оператор Reproject

#### Назначение:

Позволяет определить, какая колонка таблицы появится следующей при открытии окна Списка.

#### Синтаксис:

**Reproject** *column* [ , *column* ... ] **From** *table*

где

*column* – имя колонки;

*table* – имя открытой таблицы.

#### Описание:

Оператор **Reproject** позволяет Вам определить список колонок таблицы, которые появятся при следующем открытии окна Списка. Если Вы выполните оператор **Reproject**, а затем оператор **Browse**, то новое окно Списка будет показывать только те колонки, которые перечислены в операторе **Reproject**.

#### Пример:

Следующие операторы открывают таблицу World и показывают ее в окне Списка. Поскольку использован оператор **Reproject**, окно Списка покажет только две колонки.

```
Open Table "world" Interactive As World
```

```
Reproject Country, Population From World
```

```
Browse * From World
```

#### Смотрите также:

**Browse**

## Оператор Resume

### Назначение:

Выход из процедуры-обработчика ошибок, назначенного оператором **OnError**.

### Синтаксис:

**Resume** { **0** | **Next** | *label* }

где

*label* – метка в некоторой процедуре или функции.

### Предупреждение:

Вы не можете использовать оператор **Resume** в окне MapBasic.

### Описание:

Оператор **Resume** осуществляет выход из процедуры-обработчика ошибок, назначенного оператором **OnError**.

Группа операторов после метки, объявленной оператором **OnError**, выполняется, если в каком-нибудь операторе после **OnError** произошла ошибка. Эта группа является обработчиком ошибок и может содержать один или более операторов **Resume** для выхода из обработчика.

Оператор **Resume 0** возвращает управление программой туда, откуда был вызван обработчик ошибок с повторением оператора, в котором произошла ошибка.

Оператор **Resume Next** осуществляет перевод управления программой к оператору, следующему за оператором, где произошла ошибка.

Оператор **Resume label** передает управление программой строке с меткой, заданной параметром *label*. Метка *label* должна находиться в теле какой-нибудь процедуры. Прописные и строчные символы в имени метки не различаются.

### Пример:

```
OnError GoTo no-states
Open Table "states"
Map From states
after-mapfrom:
' ...
End Program
no-states:
Note "Нельзя открыть таблицу; окно Карты не будет открыто"
Resume after-mapfrom
```

### Смотрите также:

**Err( ), Error, Error\$( ), OnError**

### Функция RGB( )

#### Назначение:

Возвращает значение цвета в системе RGB, вычисляя из установок концентрацию красного, зеленого и синего цветов.

#### Синтаксис:

**RGB**(*red*, *green*, *blue*)

где

*red* – численное выражение в диапазоне от 0 до 255, определяющее концентрацию красного;

*green* – численное выражение в диапазоне от 0 до 255, определяющее концентрацию зеленого;

*blue* – численное выражение в диапазоне от 0 до 255, определяющее концентрацию синего.

#### Величина, полученная в результате:

Целое число. Величина типа Integer.

#### Описание:

MapBasic использует цвет в операторах как часть установки стиля линии, штриха или символа (например, оператор **Create Point**). Каждый цвет в стилях линии и штриха задается целым числом, понимаемым как RGB-величина. Функция **RGB( )** позволяет получить это число.

В таком представлении цвета используется сочетание трех компонент – красного, зеленого и синего цветов. Соответственно, в функции присутствуют три параметра – *red*, *green* и *blue*. Каждая цветовая компонента должна иметь целочисленное значение в диапазоне от 0 до 255, включительно.

RGB-величина вычисляется по формуле:

$(red * 65536) + (green * 256) + blue$

Замечание: Файл стандартных определений MAPBASIC.DEF с помощью оператора **Define** назначает имена для основных 8 цветов:

- BLACK – черный;
- WHITE – белый;
- RED – красный;
- GREEN – зеленый;
- BLUE – синий;
- CYAN – голубой;
- MAGENTA – розовый;
- YELLOW – желтый.

Если Вам необходимо задать красный цвет, то можно использовать имя RED вместо вызова функции **RGB(255,0,0)**. Имена Вы можете использовать, если Ваша программа имеет в начале оператор **Include "MAPBASIC.DEF"**.

#### Пример:

```
Dim red, green, blue, color As Integer
red = 255
green = 0
blue = 0
color = RGB(red, green, blue)
```

#### Смотрите также:

**Brush, Font, Pen, Symbol**

## Функция Right\$( )

### Назначение:

Извлекает из правой части строки определенное количество символов.

### Синтаксис:

**Right\$(string\_expr, num\_expr)**

где

*string\_expr* – строковое выражение;

*num\_expr* – целочисленное выражение.

### Величина, полученная в результате:

Строка. Величина типа String.

### Описание:

Функция **Right\$( )** возвращает строку, составленную из *num\_expr* правых символов строки *string\_expr*.

Параметр *num\_expr* должен принимать положительное целочисленное значение, ноль или больше.

Если значение параметра больше, чем длина строки *string\_expr*, то результатом будет полная строка *string\_expr*. Если значение параметра *num\_expr* меньше единицы, то функция вернет пустое значение.

### Пример:

```
Dim whole, partial As String
whole = "Казахстан"
partial = Right$(whole, 4)

' переменная partial теперь равна строке "стан"
```

### Смотрите также:

**Mid\$( ), Left\$( )**

### Функция Rnd( )

#### Назначение:

Генератор случайных чисел.

#### Синтаксис:

**Rnd(list\_type)**

где *list\_type* – целое число, задающее режим случайной последовательности.

#### Величина, полученная в результате:

Вещественное число в диапазоне от 0 до 1 (не включая). Величина типа Float.

#### Описание:

Функция **Rnd( )** возвращает случайное десятичное число с плавающей запятой, большее нуля и меньшее единицы.

Любой положительный аргумент *list\_type* приводит к тем же результатам.

Обычно используется в форме **Rnd(1)**, возвращающее случайное число. Последовательные вызовы функций **Rnd( )** образуют последовательность случайных значений. Оператор **Randomize** позволяет задать для такой последовательности случайное начальное значение. Любой положительный аргумент *list\_type* приводит к тем же результатам.

Для отладочных целей можно использовать форму **Rnd(0)**, возвращающую предыдущее значение, порожденное функцией **Rnd( )**.

Если в качестве аргумента *list\_type* задается отрицательное значение, например, **Rnd(-1)**, то функция возвращает то же значение, независимо от действия оператора **Randomize**. Этот эффект также используется в отладочных целях.

#### Пример:

```
Chknum = 10 * Rnd(1)
```

#### Смотрите также:

**Randomize**

## Оператор Rollback

### Назначение:

Отменяет все изменения в таблице, которые были сделаны до последнего сохранения таблицы на диск.

### Синтаксис:

**Rollback Table** *tablename*

где

*tablename* – имя открытой таблицы.

### Описание:

Если в таблице были сделаны изменения, но эти изменения не были сохранены, оператор **Rollback** их отменяет. Оператор выполняет такие же действия, как команда **ФАЙЛ > ВОССТАНОВИТЬ** в MapInfo.

**Замечание:** Таблица запросов (например, "ЗАПРОС1") обычно отражает текущее содержание какой-то постоянной таблицы. Поэтому, когда оператор **Rollback** применяется к таблице запроса, MapBasic автоматически отменяет все изменения в постоянной таблице, на основе которой была сделана временная таблица (кроме тех случаев, когда в процессе запроса использовалось объединение или когда запрос возвратил обобщенные значения, т.е. использовалось предложение **Group By** оператора **Select**).

Например, Вы изменили данные в таблице WORLD, сделали выбор в таблице. MapInfo динамически создает временную таблицу выбора, которая будет названа, например, ЗАПРОС1. Когда Вы применяете оператор **Rollback** к таблице ЗАПРОС1, MapBasic восстановит данные таблицы WORLD.

Применение оператора **Rollback** к связанной таблице удаляет из нее все несохраненные изменения и возвращает ее к виду, который она имела после последнего сохранения.

### Пример:

```
If keep-changes Then
  Commit Table ГОРОДА
Else
  Rollback Table ГОРОДА
End If
```

### Смотрите также:

**Commit**

### Функция Rotate( )

#### Назначение

Позволяет вращать объект (не текстовый) вокруг точки центроида.

#### Синтаксис

**Rotate(*object*, *angle*)**

*object* представляет объект, который может вращаться. Это не может быть текстовый объект.

*angle* это вещественная величина, определяющая угол (в градусах), на который поворачивается объект.

#### Возвращаемое значение

Повернутый объект.

#### Описание

Поворачивает все типы объектов кроме текстовых, при этом исходный объект не изменяется.

Для поворота текстовых объектов используйте оператор **Alter Object OBJ\_GEO\_TEXTANGLE**.

Если поворачивается дуга, эллипс, прямоугольник и скругленный прямоугольник, то результирующий объект будет преобразован в полилинию/полигон для того, чтобы узлы могли быть повернуты.

#### Пример

```
dim RotateObject as object
Open Table "C:\MapInfo_data\TUT_USA\USA\STATES.TAB"
map from states
select * from States where state = "IN"
RotateObject = rotate(selection.obj, 45)
insert into states (obj) values (RotateObject)
```

#### Смотрите также

RotateAt Point



## Функция RotateAtPoint( )

### Назначение

Позволяет вращать объект (не текстовый) вокруг определенной точки.

### Синтаксис

**RotateAtPoint(object, angle, anchor\_point\_object)**

*object* представляет объект, который может вращаться. Это не может быть текстовый объект.

*angle* это вещественная величина, определяющая угол (в градусах), на который поворачивается объект.

*anchor\_point\_object* это объект, представляющий точку, вокруг которой поворачиваются узлы вращаемого объекта.

### Возвращаемое значение

Повернутый объект.

### Описание

Поворачивает все типы объектов кроме текстовых, при этом исходный объект не изменяется.

Для поворота текстовых объектов используйте оператор **Alter Object OBJ\_GEO\_TEXTANGLE**.

Если поворачивается дуга, эллипс, прямоугольник и скругленный прямоугольник, то результирующий объект будет преобразован в полилинию/полигон для того, чтобы узлы могли быть повернуты.

### Пример

```
dim RotateAtPointObject as object
dim obj1 as object
dim obj2 as object
Open Table "C:\MapInfo_data\TUT_USA\USA\STATES.TAB" ]
map from states
select * from States where state = "CA"
obj1 = selection.obj
select * from States where state = "NV"
obj2 = selection.obj
oRotateAtPointObject = RotateAtPoint(obj1 , 65, centroid(obj2))
insert into states (obj) values (RotateAtPointObject )
```

### Смотрите также

Rotate()

### Функция Round( )

#### Назначение:

Округляет число с заданной точностью.

#### Синтаксис:

**Round**(*num\_expr*, *round\_to*)

где

*num\_expr* – численное выражение;

*round\_to* – число, определяющее точность округления.

#### Величина, полученная в результате:

Вещественное число. Величина типа Float.

#### Описание:

Функция **Round( )** возвращает округленное значение от числа, полученного в результате вычисления выражения *num\_expr*. Число *round\_to* задает точность округления. Под округлением здесь понимается замена числа *num\_expr* на ближайшее, кратное *round\_to*.

Например, если параметр *round\_to* равен 0.01, то число *num\_expr* будет округлено до сотых. Если *round\_to* равен 5, то MapBasic вернет значение, ближайшее к *num\_expr* и кратное пяти.

#### Пример:

```
Dim x, y As Float
x = 12345.6789
y = Round(x, 100)
'
' y равен 12300
'
y = Round(x, 1)
'
' y равен 12346
'
y = Round(x, 0.01)
'
' y равен 12345.68
'
```

#### Смотрите также:

**Fix( ), Format\$( ), Int( )**

## Функция RTrim\$( )

### Назначение:

Удаляет пробелы в конце строки.

### Синтаксис:

**RTrim\$(string\_expr)**

где

*string\_expr* – строковое выражение.

### Величина, полученная в результате:

Строка. Величина типа String.

### Описание:

Функция **RTrim\$( )** возвращает строку, заданную параметром *string\_expr*, но без пробелов в конце строки.

### Пример:

```
Dim name As String
name = "Мария Смирнова   "
name = Rtrim$(name)
'
'  name сейчас равно строке "Мария Смирнова"
'  (пробелов в конце строке нет)
'
```

### Смотрите также:

**LTrim\$( )**

### Оператор Run Application

#### Назначение:

Загружает прикладную программу или Рабочий Набор в MapInfo, окна которого будут добавлены к уже открытым.

#### Синтаксис:

**Run Application** *file*

где

*file* – имя файла прикладной программы или Рабочего Набора.

#### Описание:

Оператор **Run Application** запускает программу MapBasic или загружает Рабочий Набор.

Выполняя оператор **Run Application**, одна прикладная программа, написанная на MapBasic, может запустить на выполнение другую программу, определенную параметром *file*. Значение параметра должно быть именем файла откомпилированной программы (файл с расширением .MBX). Оператор не может запускать файлы с расширением .MB, то есть файлы, содержащие тексты программ.

Для остановки одной программы из другой используется оператор **Terminate Application**.

#### Пример:

Следующий оператор загружает программу REPORT.MBX:

```
Run Application "C:\MAPBASIC\APP\REPORT.MBX"
```

Следующий оператор загружает Рабочий Набор, PARCELS.WOR:

```
Run Application "Parcels.wor"
```

#### Смотрите также:

**Run Command, Run Menu Command, Run Program, Terminate Application**

## Оператор Run Command

### Назначение:

Выполняет оператор MapBasic, заданный строкой.

### Синтаксис:

**Run Command** *command*

где

*command* – символьная строка, представляющая оператор MapBasic.

### Описание:

Оператор **Run Command** интерпретирует строку *command* в оператор MapBasic и выполняет его.

Оператор имеет некоторые ограничения, обусловленные тем, что интерпретация происходит уже из откомпилированной программы. Имена переменных не могут быть заданы в строке *command*, т. к. может не произойти подстановки их значений. Например, следующая группа операторов не будет работать, так как указание на переменные **x** и **y** попадает в строку, ограниченную кавычками:

```
Dim cmd-string As String    ' Этот пример НЕ РАБОТАЕТ
Dim x, y As Float
...
cmd-string = " x = Abs(y) "
Run Command cmd-string
```

Оператор позволяет конструировать командную строку из строковых величин, используя строковые переменные, если конфликта с кавычками можно избежать:

```
Dim cmd-string As String    ' Этот пример будет работать
Dim map-it, browse-it As Logical
Open Table "world"
If map-it Then
    cmd-string = "Map From "
    Run Command cmd-string + "world"
End If
If browse-it Then
    cmd-string = "Browse * From "
    Run Command cmd-string + "world"
End If
```

### Пример 1:

Оператор **Run Command** представляет гибкий путь выполнения операторов, аргументы которых представляются списками переменной длины. Так, оператор **Map From** может включать в себя имя одной таблицы или список имен таблиц через запятую. Программа может интерпретировать оператор во время выполнения, чтобы определить (используя данные, введенные пользователем), сколько таблиц включить в карту. Оператор **Run Command** позволяет на ходу сконструировать и выполнить оператор **Map From**.

```
Dim cmd-text As String
Dim cities-wanted, counties-wanted As Logical
```

## Оператор Run Command

---

```
Open Table "states.tab"
Open Table "cities.tab"
Open Table "counties.tab"

cmd-text = "states" ' всегда будет включен слой STATES
If counties-wanted Then
    cmd-text = "counties, " + cmd-text
End If
If cities-wanted Then
    cmd-text = "cities, " + cmd-text
End If
cmd-text = "Map from " + cmd-text
Run Command cmd-text
```

### Пример 2:

Следующий пример показывает как можно дублировать окно Карты с помощью функции **WindowInfo( )** и оператора **Run Command**.

```
Dim i_map_id As Integer

' Сначала, определим идентификатор окна Карты
' (предполагая, что оно активно):
i_map_id = FrontWindow()

' Теперь дублируем это окно:
Run Command WindowInfo(i_map_id, WIN_INFO_CLONEWINDOW)
```

### Смотрите также:

**Run Application, Run Menu Command, Run Program**

## Оператор Run Menu Command

### Назначение:

Позволяет вызвать стандартную команду MapInfo так, как если бы Вы указали на ее имя в списке меню.

### Синтаксис:

**Run Menu Command** { *command\_code* | **ID** *command\_ID* }

где

*command\_code* – целочисленный код из файла MENU.DEF, соответствующий коду команды из действующей системы меню MapInfo или кнопке;

*command\_ID* – число, представляющее созданный элемент меню или кнопку.

### Описание:

Если Ваша программа включает в себя файл стандартных определений MENU.DEF, то можно использовать в операторе **Run Menu Command** имена кодов, которые определены в вышеупомянутом файле при помощи оператора **Define**. Например, для вызова команды ФАЙЛ > НОВАЯТАБЛИЦА применяется следующий оператор:

```
Run Menu Command M-FILE-NEW
```

Файл MENU.DEF содержит также определения кодов для инструментов Пенала. Например, код M-TOOLS-RULER соответствует инструменту Линейка. Если подставить значение кода в оператор, то он будет производить выбор инструмента так, как если бы пользователь указал на его картинку в Пенале. Например, следующий оператор выбирает инструмент Циркуль:

```
Run Menu Command M-TOOLS-SEARCH-RADIUS
```

Для выбора созданных кнопки или команды в меню (то есть кнопка или элемент меню, созданные и обрабатываемые программой MapBasic), используется предложение **ID**. Например, если Ваша программа создала кнопку инструмента и выполнила следующий оператор:

```
Alter ButtonPad ID 1 Add  
ToolButton  
Calling sub_procedure_name  
ID 23  
Icon MI_ICON_CROSSHAIR
```

то следовательно кнопка инструмента имеет идентификатор 23. Следующий оператор выбирает эту кнопку:

```
Run Menu Command ID 23
```

В MapInfo диалог “Режимы” – это специальный случай. Этот диалог имеет несколько кнопок, которые вызывают другие поддиалоги. Оператор **Run Menu Command** позволяет вызвать отдельно один из этих диалогов. Например, следующий оператор показывает поддиалог “Режимы окна Карты”:

```
Run Menu Command M_EDIT-PREFERENCES_MAP
```

Вы можете получить доступ к обращенной выборке, используя следующую команду MapBasic:

```
Run Menu Command M_QUERY-INVERTSELECT.
```

### Смотрите также:

**Run Application, Run Program**

### Оператор Run Program

#### Назначение:

Загружает другие программы.

#### Синтаксис:

**Run Program** *program\_spec*

где

*program\_spec* – командная строка, задающая имя программы и, если необходимо, список аргументов.

#### Описание:

Параметр *program\_spec* задает некоторую командную строку. В Windows командная строка задается так же, как в окошке "Command Line" диалога команды FILE > PROPERTIES в программе Windows Program Manager.

#### Использование оператора в среде Windows

Если программа, заданная строкой *program\_spec*, не является программой для Windows, то Map-Basic сначала создаст копию командного процессора (DOS shell), а потом будет загружена DOS-программа в его среде. Если параметр *program\_spec* имеет значение "COMMAND.COM", Map-Basic только откроет окно командного процессора. Если из прикладной программы была запущена программа для DOS, то для возвращения в MapInfo надо ввести команду "Exit".

Даже если Вы запустили DOS-программу оператором **Run Program**, Windows продолжает управлять компьютером: то есть может параллельно поддерживать другие Windows-программы и даже саму MapBasic-программу в многозадачном режиме. Возникающие в этой ситуации конфликты Ваша прикладная программа обычно не может разрешить.

Поэтому в программе должны быть предприняты меры предосторожности, чтобы избежать конфликтов многозадачности при выполнении оператора **Run Program**. Например, оператор **Run Program** можно поместить в конце программы или группы операторов, которые могли бы конфликтовать с вызываемой программой. Например, Вы создаете элемент меню, который вызывает обработчик, в котором в самом конце используется оператор **Run Program**.

#### Пример:

Оператор **Run Program** загружает текстовый редактор Windows, который называется "Notepad", и открывает в нем текстовый файл "THINGS.2DO".

```
Run Program "notepad.exe things.2do"
```

#### Смотрите также:

**Run Application, Run Command, Run Menu Command**



## Оператор Save File

### Назначение:

Копирует файл.

### Синтаксис:

**Save File** *old\_filespec* **As** *new\_filespec* [ **Append** ]

где

*old\_filespec* – строка с именем (и, если необходимо, с маршрутом) существующего на диске файла, который не должен быть открыт;

*new\_filespec* – строка с именем (и, если необходимо, с маршрутом), с которым будет произведена копия; при этом файл с этим именем не должен быть открыт.

### Описание:

Оператор **Save File** копирует файл. Файл при этом должен быть закрыт для операций ввода/вывода.

Если в операторе используется ключевое слово **Append** и параметр *new\_filespec* задает имя и маршрут уже существующего файла, то содержимое файла *old\_filespec* будет дописано в конец файла *new\_filespec*.

Не надо использовать оператор **Save File** для копирования файлов, являющихся компонентами таблицы (таких как *filename.TAB*, *filename.MAP* и т. п.). Для копирования таблиц правильно будет использовать оператор **Commit Table... As**.

Оператор **Save File** не может копировать файл сам в себя.

### Пример:

```
Save File "settings.txt" As "settings.bak"
```

### Смотрите также:

**Kill, Rename File**

# Оператор Save Window

### Назначение:

Создает файл с изображением из окна так, как это делает команда в MapInfo ФАЙЛ > ЭКСПОРТ ОКНА.

### Синтаксис:

```
Save Window window_id  
  As filespec  
  Type filetype  
  [ Width image_width [ Units paper_units ] ]  
  [ Height image_height [ Units paper_units ] ]  
  [ Resolution output_dpi ]  
  [ Copyright notice [ Font ... ] ]
```

*window\_id* задает идентификатор окна Карты, Отчета, Графика, Легенды, Статистики, Информации или Линейки (в качестве параметра можно использовать вызов функции **FrontWindow( )** или **WindowID( )**)

*filespec* строка, имя создаваемого файла

*filetype* строка, задающая один из форматов файла:

- “BMP” задает растровый формат Bitmap;
- “WMF” задает формат метафайла в Windows;
- “JPEG” задает растровый формат JPEG;
- “JP2” задает растровый формат JPEG 2000
- “PNG” задает растровый формат Portable Network Graphics;
- “TIFF” задает растровый формат TIFF;
- “TIFFCMYK” задает растровый формат TIFF CMYK
- “PSD” задает растровый формат Photoshop 3.0;
- “EMF” задает формат Windows Enhanced Metafile.

*image\_width* число, задающее ширину изображения в заданных единицах

*image\_height* число, задающее высоту изображения в заданных единицах

*paper\_units* строка, задающая единицу измерения (например, “cm” – сантиметры)

*output\_dpi* число, задающее разрешение изображения в DPI (dots per inch).

*notice* строка, задающая copyright; появляется внизу изображения

Предложение **Font** указывает стиль текста

### Описание

Оператор **Save Window** сохраняет изображение окна в файле. Действие оператора аналогично действию команды ФАЙЛ > ЭКСПОРТ ОКНА, с тем исключением, что оператор не выводит диалог на экран.

Размер изображения, полученного из окна Карты, Отчета или Графика, по умолчанию будет равен размеру самого окна. Размер изображения, полученного из окна Легенды, Информации, Статистики или Линейки, по умолчанию будет устанавливаться таким, чтобы показать в окне все данные. Вы можете определить свои размеры для экспортируемых изображений в предложениях **Width** и **Height**.

**Resolution** позволяет задать разрешение изображения в dpi при экспорте изображения в растровый формат.

Предложение **Font** определяет стиль текста в указании авторских прав.

### Настройка авторских прав (Copyright)

Чтобы включить текст авторских прав в нижнюю часть изображения, используйте дополнительное предложение **Copyright**. Смотрите пример ниже.

Чтобы стереть текст авторских прав, задайте предложение **Copyright** с пустой строкой ("").

### Ошибки

В случае нехватки места на диске при экспорте окна может быть зафиксирована ошибка под номером 408. Имейте это в виду, если Вы пытаетесь создать слишком большое изображение.

### Примеры

В этом примере создается метафайл Windows:

```
Save Window i_mapper_ID As "riskmap.wmf" Type "WMF"
```

Этот пример показывает как задать строку с авторскими правами. Функция **Chr\$( )** используется для вставки символа авторских прав.

```
Save Window i_mapper_ID As "riskmap.bmp"  
Type "BMP"  
Copyright "Copyright " + Chr$(169) + " 1996, MapInfo Corp."
```

### Смотрите также

Export

### Оператор Save Workspace

#### Назначение:

Создает файл Рабочего Набора с текущим состоянием рабочего окна MapInfo.

#### Синтаксис:

**Save Workspace As** *filespec*

где

*filespec* – строка с именем Рабочего Набора.

#### Описание:

Оператор **Save Workspace** создает Рабочий Набор по текущему состоянию программы MapInfo.

Действие оператора аналогично действию команды **ФАЙЛ > СОХРАНИТЬ РАБОЧИЙ НАБОР**, с тем исключением, что оператор не выводит диалог на экран.

Для загрузки существующего Рабочего Набора используйте оператор **Run Application**.

#### Пример:

**Save Workspace As** "market.wor"

#### Смотрите также:

**Run Application**

## Функция **SearchInfo( )**

### Назначение:

Возвращает информацию о результатах поиска, сделанного функцией **SearchPoint( )** или **SearchRect( )**.

### Синтаксис:

**SearchInfo(sequence\_number, attribute)**

где

*sequence\_number* – целое число от 1 до количества найденных объектов;

*attribute* – короткое целое число, код результата функции.

### Величина, полученная в результате:

Строка или целое число. Тип зависит от значения параметра *attribute*.

### Описание:

После вызова функции **SearchPoint( )** или **SearchRect( )**, осуществляющих поиск объектов на Карте, функция **SearchInfo( )** обрабатывает результаты поиска.

Параметр *sequence\_number* должен принимать значения от 1 и более. Максимальное значение для этого параметра равняется результату функции **SearchPoint( )** или **SearchRect( )**.

Параметр *attribute* должен принимать значение одного из кодов, имена для которых заданы в файле стандартных определений MAPBASIC.DEF:

#### Значение *attribute*

SEARCH\_INFO\_TABLE

SEARCH\_INFO\_ROW

#### Результат:

Строка, величина типа String: имя таблицы, содержащей этот объект. Если объект принадлежит Косметическому слою, то строка будет равна "CosmeticN" (где N – число от 1 и более).

Целое число, величина типа Integer: номер записи. Этот номер Вы можете использовать в операторе **Fetch** или в предложении **Where** оператора **Select**.

Использовать имена кодов в программе Вы можете, включив в текст Вашей программы оператор **Include "MAPBASIC.DEF"**.

Результаты поиска хранятся в памяти до тех пор, пока выполняется программа или до следующего поиска. Результаты поиска не будут удалены из памяти, если пользователь закроет окно или таблицу, в которых осуществлялся поиск; поэтому не надо затягивать с обработкой результатов поиска. Чтобы форсировать удаление результатов поиска из памяти, проведите поиск, который обязательно ничего не найдет (например, поиск по координатам 0, 0).

MapInfo поддерживает отдельные наборы значений для результатов поиска для каждой действующей MapBasic-программы, а также набор для результатов поиска самой программы MapInfo (для команд, введенных из окна MapBasic).

## Функция SearchInfo( )

---

### Ошибки:

В результате выполнения функции может генерироваться код ошибки ERR\_FCN\_ARG\_RANGE, если значение параметра *sequence\_number* больше числа найденных объектов.

### Пример:

Следующая программа создает кнопки для двух инструментов. Если пользователь пользуется точечным инструментом, то программа вызывает функцию **SearchPoint( )**; если пользователь рисует рамки, то программа вызывает функцию **SearchRect( )**. В каждом случае программа использует функцию **SearchInfo( )** для определения того, какие объекты попались пользователю.

```
Include "mapbasic.def"
Include "icons.def"
Declare Sub Main
Declare Sub tool_sub

Sub Main
  Create ButtonPad "Поиск" As
    ToolButton Calling tool_sub ID 1
    Icon      MI_ICON_ARROW
    Cursor    MI_CURSOR_ARROW
    DrawMode  DM_CUSTOM_POINT
    HelpMsg   "Укажите мышкой на Карту\nПоиск в точке"
  Separator
  ToolButton Calling tool_sub ID 2
  Icon      MI_ICON_SEARCH_RECT
  Cursor    MI_CURSOR_FINGER_LEFT
  DrawMode  DM_CUSTOM_RECT
  HelpMsg   "Нарисуйте прямоугольник на Карте\nПоиск в рамке"
  Width 3

  Print "Работает программа поиска."
  Print "Выберите инструмент из панели Поиск"
  Print "и укажите на Карту."
End Sub

Sub tool_sub
  ' Эта процедура вызывается, если пользователь действует
  ' одним из инструментов из панели Поиск.
  Dim x, y, x2, y2 As Float,
    i, i_found, i_row_id, i_win_id As Integer,
    s_table As Alias
  i_win_id = FrontWindow()
  If WindowInfo(i_win_id, WIN_INFO_TYPE) <> WIN_MAPPER Then
    Note "Этот инструмент работает только в окне Карты."
    Exit Sub
  End If
```

```

' Определяем начальную точку действия инструмента.
x = CommandInfo(CMD_INFO_X)
y = CommandInfo(CMD_INFO_Y)

If CommandInfo(CMD_INFO_TOOLBTN) = 1 Then
    ' случай, когда действует точечный инструмент.
    ' определяем, сколько объектов содержат данную точку.
    i_found = SearchPoint(i_win_id, x, y)
Else
    ' случай, когда действует инструмент, рисующий рамку.
    ' определяем, сколько объектов содержатся в рамке.
    x2 = CommandInfo(CMD_INFO_X2)
    y2 = CommandInfo(CMD_INFO_Y2)
    i_found = SearchRect(i_win_id, x, y, x2, y2)
End If

If i_found = 0 Then
    Beep ' Объектов в этом месте нет.
Else
    Print Chr$(12)
    If CommandInfo(CMD_INFO_TOOLBTN) = 2 Then
        Print "Прямоугольник: x1= " + x + ", y1= " + y
        Print "x2= " + x2 + ", y2= " + y2
    Else
        Print "Точка: x=" + x + ", y= " + y
    End If
End If

' Обработка результатов поиска.
For i = 1 to i_found
    ' Определяем имя таблицы, содержащей найденный объект.
    s_table = SearchInfo(i, SEARCH_INFO_TABLE)

    ' Определяем номер записи, содержащей найденный объект.
    i_row_id = SearchInfo(i, SEARCH_INFO_ROW)

    If Left$(s_table, 8) = "Cosmetic" Then
        Print "Объект на Косметическом слое"
    Else
        ' извлекаем строку таблицы, содержащую объект.
        Fetch rec i_row_id From s_table
        s_table = s_table + ".col1"
        Print s_table
    End If
Next
End If
End Sub

```

**Смотрите также:**

[SearchPoint\( \), SearchRect\( \)](#)

### Функция **SearchPoint( )**

#### Назначение:

Ищет объекты в заданной точке Карты.

#### Синтаксис:

**SearchPoint(*map\_window\_id*, *x*, *y* )**

где

*map\_window\_id* – идентификатор окна Карты;

*x* – координата по оси X (например, долгота);

*y* – координата по оси Y (например, широта).

#### Величина, полученная в результате:

Целое число. Величина типа Integer.

#### Описание:

Функция **SearchPoint( )** осуществляет поиск объектов в заданной точке Карты и возвращает количество найденных объектов. Поиск проводится по всем доступным слоям окна Карты, включая Косметический слой, если для него установлен режим доступности.

Функция не выбирает найденные объекты и не отменяет текущего выбора. Функция помещает список выбранных объектов в оперативную память. Для чтения этого списка используется функция **SearchInfo( )**, вызываемая после выполнения функции **SearchPoint( )**.

Поиск имеет небольшой допуск, аналогичный допуску при действии инструмента Информация. Точки и линейные объекты, расположенные слишком близко к месту указания мышкой, считаются “найденными”, даже если координаты нажатия кнопки мышки не совпадают с точкой или ложатся непосредственно на линию.

Для того, чтобы пользователь мог осуществить выбор точки на карте при помощи мышки, можно использовать оператор **Create ButtonPad** или **Alter ButtonPad** для создания нового инструмента. Используйте код DM\_CUSTOM\_POINT как код рисования “точечным” инструментом. В обработчике инструмента используйте вызов функции **CommandInfo( )** для определения координат точки, которую пользователь задал новым инструментом.

#### Пример:

Смотрите пример в описании функции **SearchInfo( )**.

#### Смотрите также:

**SearchInfo( )**, **SearchRect( )**



## Функция **SearchRect( )**

### Назначение:

Ищет объекты на Карте в заданном прямоугольнике (рамке).

### Синтаксис:

**SearchRect**(*map\_window\_id*, *x1*, *y1*, *x2*, *y2* )

где

*map\_window\_id* – идентификатор окна Карты;

*x1* и *y1* – координаты, задающие один из углов прямоугольника;

*x2* и *y2* – координаты, задающие противоположный по диагонали угол прямоугольника.

### Величина, полученная в результате:

Целое число. Величина типа Integer.

### Описание:

Функция **SearchRect( )** осуществляет поиск объектов в заданном прямоугольнике в окне Карты и возвращает количество найденных объектов. Поиск осуществляется по всем доступным слоям окна Карты, включая Косметический слой, если для него установлен режим доступности. Функция не выбирает найденные объекты и не отменяет текущего выбора. Функция помещает список выбранных объектов в оперативную память. Для чтения этого списка используется функция **SearchInfo( )**, вызываемая после выполнения функции **SearchRect( )**.

Механизм поиска работает аналогично механизму инструмента MapInfo Выбор-в-Рамке: если центроид объекта попадает в прямоугольник, то объект включается в список найденных.

Для того, чтобы пользователь мог задавать прямоугольник на Карте при помощи мышки, можно использовать оператор **Create ButtonPad** или **Alter ButtonPad** для создания нового инструмента. Используйте код DM\_CUSTOM\_RECT как код рисования “рамочным” инструментом. В обработчике инструмента используйте вызов функции **CommandInfo( )** для определения координат прямоугольника, который пользователь задал новым инструментом.

### Пример:

Смотрите пример в описании функции **SearchInfo( )**.

### Смотрите также:

**SearchInfo( )**, **SearchPoint( )**

### Функция Seek( )

#### Назначение:

Возвращает текущую позицию в файле для операций ввода/вывода.

#### Синтаксис:

**Seek(*filenum*)**

где

*filenum* – номер открытого файла, целое число.

#### Величина, полученная в результате:

Целое число. Величина типа Integer.

#### Описание:

Функция **Seek( )** возвращает текущую позицию в открытом файле для операций ввода/вывода. Значение параметра *file* должно быть номером файла, под которым он был открыт оператором **Open File**.

Результатом функции **Seek( )** будет целое число. Если файл был открыт в режиме прямого доступа, **Seek( )** вернет номер записи (запись, которая будет прочитана или записана). Если файл открыт в двоичном режиме, функция вернет номер байта, который будет прочитан или записан следующей операцией ввода/вывода.

#### Ошибки:

В результате выполнения функции может генерироваться код ошибки  
ERR-FILEMGR-NOTOPEN, если файл не был открыт.

#### Смотрите также:

**Get, Open File, Put, Seek**

## Оператор Seek

### Назначение:

Назначает в определенном файле текущую позицию для дальнейших операций ввода/вывода.

### Синтаксис:

**Seek** [#]*filenum*, *position*

где

*filenum* – номер открытого файла, целое число;

*position* – целое число, определяющее новую текущую позицию.

### Описание:

Оператор **Seek** устанавливает доступ к содержимому открытого файла, начиная с позиции *position*.

Файл должен быть открыт оператором **Open File** под номером *filenum*. Если файл был открыт в режиме прямого доступа (**RANDOM**), то значение *position* задает номер записи.

Если файл был открыт в режиме последовательного доступа, то оператор **Seek** устанавливает доступ к содержимому файла, начиная с байта, номер которого задается параметром *position*.

Операции чтения из файла или записи в файл после выполнения этого оператора будут начинаться с этой позиции в файле.

### Смотрите также:

**Get, Input #, Open File, Print #, Put, Seek( ), Write #**

## Процедура SelChangedHandler

### Назначение:

Процедура, которая автоматически вызывается при изменении выбора строк в таблицах.

### Синтаксис:

```
Declare Sub SelChangedHandler  
Sub SelChangedHandler  
    statement_list  
End Sub
```

где

*statement\_list* – список операторов процедуры-обработчика.

### Описание:

**SelChangedHandler** – зарезервированное имя для процедуры MapBasic. Когда пользователь запускает программу, в которой есть такая процедура, программа не завершается после того, как выполняются все операторы процедуры Main и других процедур, вызванных из нее. Программа будет пребывать в режиме ожидания до тех пор, пока не произойдет изменение в выборе строк таблиц. После этого автоматически начнется выполнение процедуры с именем **SelChangedHandler**. После выполнения процедуры прикладная программа вновь переходит в режим ожидания. И так всякий раз при новых изменениях.

Для получения информации об изменениях используйте в теле процедуры **SelChangedHandler** функцию **CommandInfo( )** со следующими кодами:

Код	Результат функции
CMD_INFO_SELTYPE	1 – если строка была добавлена в выборку; 2 – если строка была исключена из предыдущей выборки; 3 – если все строки в таблице выбраны; 4 – если все строки таблицы были исключены из предыдущей выборки.
CMD_INFO_ROWID	Целое число (тип Integer): номер строки, которая была добавлена в выбор или исключена из выбора.
CMD_INFO_INTERRUPT	Логическая величина (тип Logical): “Да” (TRUE), если обновление экрана прервано клавишей ESC; “Нет” (FALSE) – иначе. Для завершения программы в теле процедуры <b>SelChangedHandler</b> используется оператор <b>End Program</b> . При этом полностью освобождается память, занимаемая "ожидающей" программой, и после следующего изменения в выборе процедура <b>SelChangedHandler</b> уже не будет вызываться. Одновременно в состоянии ожидания могут находиться несколько прикладных программ. Поэтому при изменении выбора автоматически выполняются все процедуры <b>SelChangedHandler</b> из этих программ, одна за другой. <b>Замечание:</b> Процедура <b>SelChangedHandler</b> не может активизировать другие окна. Другими словами, в процедуре <b>SelChangedHandler</b> не могут быть выполнены такие операторы, как <b>Note</b> , <b>Print</b> или <b>Dialog</b> .

### Смотрите также:

**CommandInfo( )**, **SelectionInfo( )**

## Оператор Select

### Назначение:

Выбирает отдельные строки и колонки из одной или более таблиц и составляет из них временные таблицы запросов. Этот оператор часто используется для сортировки и вычисления промежуточных сумм.

### Синтаксис:

```
Select expression_list
      From table_name [ , ... ]
[ Where expression_group ]
[ Into results_table [ Noselect ] ]
[ Group By column_list ]
[ Order By column_list ]
```

где

*expression\_list* – список выражений через запятую, задающих содержимое колонок временной таблицы "Selection";

*expression\_group* – одно выражение или список выражений, разделенных словами **AND** или **OR**;

*table\_name* – имя открытой таблицы;

*results\_table* – имя таблицы, в которую будут помещены результаты выбора;

*column\_list* – имя колонки или список имен, разделенных запятыми.

### Описание:

Оператор **Select** предоставляет программисту MapBasic возможности диалога команды ЗАПРОС > SQL-ЗАПРОС.

Оператор **Select** построен по образцу одноименного оператора в языке запросов SQL (Structured Query Language). Вариант оператора **Select**, применяемый в MapBasic, позволяет использовать уникальные географические возможности MapInfo, которые не имеют многие базы данных, использующие язык SQL.

Имена колонок, задаваемые выражениями типа *имятаблицы.имя колонки* в операторе **Select** могут использовать имена только тех таблиц, которые перечислены в предложении **From**.

Например, оператор **Select** выбирает значения в колонке STATES.OBJ, если таблица STATES включена в список предложения **From** в операторе **Select**.

Оператор **Select** может выполнять множество различных задач. Один оператор **Select** может перестроить таблицу и отфильтровать (т.е. выбрать по критерию) записи. Также оператор **Select** может вычислить промежуточные и итоговые суммы для всей таблицы. Оператор **Select** может отсортировать записи в таблице, создать и заполнить новую колонку из других и скомбинировать колонки из разных таблиц.

Общий смысл оператора **Select** состоит в том, что из одной или нескольких таблиц выбираются строки, которые помещаются во временную таблицу под именем "Selection" или под именем, заданным параметром *results\_table*. Эта временная таблица в дальнейшем интерпретируется MapInfo, как и любая другая.

После того, как оператор **Select** образует выборку, обращение к функции **SelectionInfo( )** дает информацию о составе выборки.

Синтаксис оператора **Select** допускает несколько предложений, большинство которых не обязательны. Природа и функция оператора **Select** зависит от того, какие предложения входят в состав оператора. Например, если Вы хотите отфильтровать записи по критерию, то нужно включить предложение **Where**; если нужно просуммировать значения, то включите в состав оператора предложение **Group By**; для сортировки включите предложение **Order By**. Употребление каждого из этих предложений не исключает другого. Оператор **Select** может состоять из всех возможных предложений сразу.

### Предложение Select

Параметр *expression\_list* задает колонки, которые будут включены в результирующую таблицу. Список колонок располагается сразу за первым словом **Select** в операторе и является обязательным. Самое простое значение для параметра *expression\_list* – звездочка (\*). Это значит, что в результирующую таблицу будут включены все колонки. Например, оператор

```
Select * From world
```

говорит MapBasic включить все колонки во временную таблицу "Selection".

Другим возможным значением параметра *expression\_list* является список выражений через запятую. Каждое из выражений представляет колонку для результирующей таблицы. Обычно такое выражение включает в себя имя одной колонки или даже имена нескольких колонок таблицы, из которой производится выбор. При составлении списка колонок можно использовать функции и/или операторы MapBasic. Например, в следующем операторе список выражений состоит из двух выражений:

```
Select Страна, Round(Население, 1000000)
From world
```

Первое – имя колонки "Страна", второе – вызов функции округления (**Round( )**) значений из колонки "Население" до целых миллионов.

После выполнения этого оператора первая колонка в результирующей таблице будет содержать величины из колонки "Страна" с названиями стран в таблице WORLD, а вторая – величины из колонки "Население" с численностью населения, округленной до миллионов.

Выражение из списка *expression\_list* может также назначить синоним, которым будет именоваться соответствующая колонка, если, например, показывать эту результирующую таблицу в окне Списка. Следующий оператор назначает для второй колонки синоним "Миллионы":

```
Select Страна, Round(Население, 1000000) "Миллионы"
From world
```

Таблицы, которые могут иметь присоединенные графические объекты, имеют специальную колонку, которая имеет имя "object" (или "obj"). Если в список колонок включить слово "obj", то в результирующую таблицу будет добавлена строка, содержащая тип графического объекта, присоединенного к данной записи или ничего, если такого объекта нет.

Значение параметра *expression\_list* может быть или списком выражений, или звездочкой, но звездочка не может употребляться в списке выражений. Пример следующего оператора НЕ РАБОТАЕТ:

```
Select *, object From world ' неправильно!!!
```

### Предложение From

Обязательное предложение, в котором должно указываться имя открытой таблицы, из которой производится выбор. Значения колонок из таблицы *table\_name* будут скопированы в новую, полученную в результате действий оператора **Select**.

### Предложение Where

Одна из функций предложения **Where**, основная, заключается в задании критерия выбора строк в таблице *table\_name*. Здесь могут использоваться любые выражения (смотрите раздел "Выражения" ниже). Два или более выражения разделяются словами **And** или **Or**, а не запятыми. Например:

```
Where Доход > 15000 And Доход < 25000
```

Оператор **And** играет роль логического "И", то есть оба условия должны выполняться, а оператор **Or** играет роль логического "ИЛИ", то есть MapBasic выберет запись, если она удовлетворяет любому из условий. Вы можете также применять оператор **Not**, отрицание. Например, следующий оператор выбирает строки, лишённые присоединенного графического объекта.

```
Where Not Object
```

В этом примере используется имя специальной колонки "object".

Если в операторе **Select** используются две или более таблицы, то предложение **Where** должно присутствовать обязательно, и, более того, в этом предложении должны быть заданы условия объединения. Обычно такой объединяющий оператор выглядит примерно так: **Where таблица1.поле = таблица2.поле**, где две колонки в разных таблицах сопоставляются друг другу. В следующем примере показано, как можно объединить таблицы RUSSIA и CITY200, если в колонке "Аббр" таблицы CITY200 и в колонке "Аббр" таблицы STATES содержатся аббревиатуры областей России:

```
Where RUSSIA.A66p = CITY200.A66p
```

Объединение Вы можете произвести, используя также географический оператор:

```
Where RUSSIA.obj Contains CITY200.obj
```

Предложение **Where** можно также применять для более изощренного выбора, для чего используются специальные операторы **Any** и **All**. Оператор **Any**, применяемый к некоему множеству значений, позволяет проверить, выполняется ли выражение в предложении **Where** для какого-либо из этих значений. Оператор **All**, наоборот, требует, чтобы выражение в предложении **Where** выполнялось для всех значений, объединяемых этим оператором.

Следующий запрос выбирает любую запись о клиентах, в колонке "Аббр" которой содержатся значения "МОС", "СПБ" или "ЯРС". Оператор **Any( )** работает так же, как и оператор "IN" в обычном языке SQL.

```
Select * From customers  
Where A66p = Any ( "МОС" , "СПБ" , "ЯРС" )
```

В предложение **Where** может быть включен самостоятельный оператор **Select**, называемый "подзапросом". В следующем примере используются две таблицы: PRODUCTS, содержащая записи о продаваемых товарах, и ORDERS, с данными о заказах на товары. Оплаченные товары могут в данный момент не находиться на складе. Задача состоит в том, чтобы выяснить номенклатуру заказанных товаров, находящихся на складе. Другими словами, нужно "выбрать все заказы, которые не находятся в списке распроданных товаров".

```
Select * From orders
Where partnum <>
All(Select partnum from products
where not instock)
```

Во второй строке запроса слово **Select** появляется второй раз, уже как подзапрос. Этот подзапрос выбирает все товары, которых в данный момент **НЕТ** на складе. Главный запрос **Where** выполняет поставленную задачу с помощью оператора **All( )** и результатов подзапроса. (Слово "instock" означает "на складе"; это логическая переменная).

В примере, приведенном выше, подзапрос создает набор значений, а предложение **Where** главного запроса проверяет условие запроса для них. В других случаях подзапрос может использовать функции обобщения, вычисляющие одно значение. В следующем примере используется функция **Avg( )** для вычисления среднего значения в поле "Население" таблицы RUSSIA. В результате выполнения оператора **Select** выбираются записи о более чем среднем населении.

```
Select * From RUSSIA
Where население >
(Select Avg(население) From RUSSIA)
```

MapInfo также поддерживает ключевое слово SQL **In**. В операторе **Select** слово **In** может заменять "**= Any**". Другими словами, предложение:

```
Where state = Any ("МОС", "СПБ", "ЯРС")
```

эквивалентно предложению:

```
Where state In ("МОС", "СПБ", "ЯРС"),
```

а фраза **Not In** в запросе эквивалентна фразе **<> All**.

Оператор **Select** не поддерживает в MapBasic "синхронные" или "коррелирующие" подзапросы. "Синхронные" подзапросы ссылаются на внешние таблицы. Например:

```
'
' Следующий запрос содержит в своем тексте ссылку
' на внешнюю таблицу. Он НЕ работает в MapBasic
'

Select * from ДРУЗЬЯ
Where друг.ммя =
(Select люди.ммя From люди
Where друг.ммя = клиент.ммя)
```

Последнее замечание адресовано профессионалам, привыкшим составлять сложные SQL-запросы к другим базам данных.



## Предложение Into

Задаёт имя для результирующей таблицы. Если предложения **Into** нет в операторе, то таблица с результатами выбора будет названа именем "Selection". Если последующие запросы обращаются к таблице "Selection", то результаты запросов будут называться именами ЗАПРОСН (например, ЗАПРОС1).

Если в предложении используется ключевое слово **Noselect**, то оператор формирует запрос не меняя предыдущую таблицу "Selection". То есть те записи, которые уже были выбраны, после оператора с ключом **Noselect** останутся выбранными.

**Замечание:** Если в операторе использовалось ключевое слово **Noselect**, то операция не повлечёт за собой запуска процедуры-обработчика **SelChangedHandler**.

## Предложение Group By

Предложение **Group By** определяет порядок группировки строк при выполнении обобщающих действий (вычислений промежуточных сумм и других значений). В предложении **Group By** обычно задаётся имя колонки (или несколько имен колонок), на основании значений из которых группируются промежуточные результаты. Например, если нужно обобщить какие-либо данные для областей России, то в предложении **Group By** должна быть задана колонка, содержащая имена областей России.

Функции обобщения **Sum( )**, **Min( )**, **Max( )**, **Count(\*)**, **Avg( )** и **WtAvg( )** обычно не используются в предложении **Group By**. Они помещаются в список *expression\_list* оператора **Select**, а предложение **Group By** только задаёт группирующую колонку.

В следующем примере таблица Q4SALES содержит информацию о продажах за четвёртый квартал. Каждая запись содержит информацию о сумме каждой сделки в колонке "amount". В колонке "territory" задана территория, на которой произошла сделка. В запросе вычисляется, сколько сделок произошло на каждой территории и суммарный объём этих сделок.

```
Select territory, Count(*), Sum(amount)
From q4sales
Group By territory
```

Предложение **Group By** заставляет MapBasic сначала сгруппировать записи по "территориям", а затем сосчитать сумму сделок на каждой "территории". Запрос создаёт три колонки: имя территории, количество записей в таблице Q4SALES, относящихся к данной территории, и сумму сделок на этой территории.

Функция **Sum( )** принимает в качестве параметра имя колонки. Функция **Count( )** с параметром-звездочкой просто пересчитывает количество записей в каждой из групп. Функция **Count( )**, в отличие от других функций обобщения, не нуждается в аргументах – именах колонок.

В следующей таблице приведено описание функций обобщения данных:

### Функция

### Описание

**Avg(column)**

Возвращает среднее значение из значений в колонке *column*.

**Count(\*)**

Возвращает количество записей в группе.

## Оператор Select

---

<b>Max(<i>column</i> )</b>	Возвращает наибольшее значение из значений в колонке <i>column</i> для всех записей группы.
<b>Min(<i>column</i> )</b>	Возвращает наименьшее значение из значений в колонке <i>column</i> для всех записей группы.
<b>Sum(<i>column</i> )</b>	Возвращает сумму значений в колонке <i>column</i> для всех записей группы.
<b>WtAvg(<i>column</i>, <i>weight_column</i> )</b>	Возвращает среднее взвешенное значение из значений в колонке <i>column</i> для всех записей группы.

### Вычисление взвешенной средней величины

Функция вычисления взвешенной средней величины **WtAvg( )** использует значения из дополнительной колонки в качестве коэффициентов значений из основной колонки. Например, следующий оператор использует функцию **WtAvg( )** для вычисления уровня грамотности на каждом континенте:

```
Select continent, Sum(Нас_1994), WtAvg(Грамотность, Нас_1994)
From World
Group By Континент
Into Lit_query
```

Предложение **Group By** задает группировку строк таблицы. MapInfo группирует записи по значениям из колонки “Континент”. Все строки, имеющие значение “Северная Америка” в колонке “Континент” будут рассматриваться как одна группа; Все записи о континенте “Азия” попадут в другую группу и т.д. Для каждой группы записей, т.е. для каждого континента, MapInfo вычисляет взвешенное среднее показателей грамотности.

Простое среднее (функция **Avg( )**) вычисляется как сумма, деленная на количество слагаемых. Взвешенное среднее (функция **WtAvg( )**) более сложна; при вычислении взвешенного среднего одни значения могут иметь больший вес и больше влиять на результат. В нашем примере среднее значение грамотности вычисляется с учетом населения (колонка “Нас\_1994”); другими словами, страны с большим населением вносят больший вклад в результат.

### Выражения, задающие колонки, в предложении Group By

В предложении **Group By** колонки могут задаваться именем, например, как в приведенном выше примере группировки по территориям. Альтернативным способом задания колонок является использование ее порядкового номера в таблице или слова типа **col#**, где вместо знака # должен стоять номер колонки. Группирующее предложение в операторе **Select** из предыдущего примера может выглядеть как **Group By col1**, так и **Group By 1**, вместо **Group By territory** и не может содержать вызов функции, возвращающей переменное значение. Например, так как функция **ObjectInfo( )** возвращает переменное значение, она не может быть использована для группировки в операторе **Select**.

Альтернативный синтаксис в обозначениях колонок бывает необходим в тех случаях, когда нужно обозначить колонку, имя которой является результатом вычислений. В следующем примере функция **Month( )** используется для вычисления группирующих значений, но название колонки из выражения получить нельзя. Поэтому нужно применить альтернативный синтаксис:

```
Select Month(день-болезни), Count(*)
From Болезни
Group By 1
```

В этом примере каждая запись таблицы БОЛЕЗНИ должна содержать дату болезни. В результате запроса образуется таблица из 12 строк (по количеству месяцев); во второй колонке будет содержаться количество дней, пропущенных в этом месяце по болезни.

### Группировка по значениям из нескольких колонок

В некоторых случаях Вам надо будет задать более одного имени колонки в предложении **Group By**. Это нужно, когда информации в одной колонке недостаточно для точного задания условия группировки. Пусть, например, Вы располагаете картой районного деления России; названия некоторых районов, принадлежащих разным областям, совпадают. Например, Первомайский район есть в нескольких областях. Поэтому, определив в предложении **Group By** только название района, Вы получите в результате запроса недостоверные данные – обобщенные по всем районам с именем "Первомайский", независимо от того, в какой области они находятся. Чтобы решить эту проблему (или аналогичную), задайте колонку имен областей, как дополнительную в предложении **Group By**:

```
Group By район, область
```

В этом случае для каждой уникальной пары "район-область" будет произведена отдельная группировка; то есть суммарные значения для пары "Первомайский-Нижегородская" будут различаться со значениями для пары "Первомайский-Дагестан".

### Предложение Order By

Предложение **Order By** задает колонку или колонки, по значениям которых должна происходить сортировка в результирующей таблице. Также как и в предложении **Group By**, колонки задаются списком имен полей или списком номеров этих полей. Элементы списка разделяются запятыми.

Стандартный порядок сортировки – по возрастанию, то есть от "А" до "Z", от "А" до "Я" и от 0 до 9. Сортировка по убыванию задается ключевым словом **Desc**, так как показано в следующем примере:

```
Select * From Города
Order By Область, население Desc
```

Этот запрос выполняет двухуровневую сортировку таблицы ГОРОДА. Сначала MapBasic сортирует города в областях по возрастанию имен областей, после чего MapBasic внутри каждой группы, представляющей одну область, сортирует по убыванию населения городов. Слово **Desc** отделяется пробелом, а не запятой.

Предложение **Order By** не может содержать вызов функции, возвращающей переменное значение. Например, так как функция **ObjectInfo( )** возвращает переменное значение, она не может быть использована для сортировки в операторе **Select**.

### Географические операторы

MapBasic поддерживает несколько географических операторов. Они используются в любых выражениях и особенно полезны в предложении **Where** для задания критерия выбора на основании взаимного расположения объектов на Карте. Все географические операторы работают только со значениями объектного типа и в результате дают логическую величину.

## Оператор Select

---

Выражение	Выражение истинно, если:
objectA <b>Contains</b> objectB	центроид объекта B лежит в границах объекта A;
objectA <b>Contains Part</b> objectB	границы объекта B частично лежат внутри границ объекта A;
objectA <b>Contains Entire</b> objectB	граница объекта B полностью лежит внутри границ объекта A;
objectA <b>Within</b> objectB	центроид объекта A лежит в границах объекта B;
objectA <b>Partly Within</b> objectB	границы объекта A частично лежат внутри границ объекта B;
objectA <b>Entirely Within</b> objectB	граница объекта A полностью лежит внутри границ B;
objectA <b>Intersects</b> objectB	если объекты имеют хотя бы одну общую точку.

### Ускоренный выбор

Некоторые варианты оператора **Select** выполняют операцию выбора быстрее, чем другие. Скорость выбора зависит от содержания предложения **Where**, задающего критерий выбора. Если после слова **Where** стоит одно выражение в форме:

*columnname = constant\_expression*

(где *columnname* – имя колонки, а *constant\_expression* – выражение из постоянных строковых величин) или выражений такой формы несколько и они разделены операционными словами **And**, то такой оператор **Select** будет выполняться быстрее, так как в этом случае максимально используются преимущества индексации. Если подобные выражения объединены оператором **Or**, то преимущества индексации не используются.

Также MapInfo сравнительно быстрее выполняет выбор с предложениями **Where** вида:

*[ tablename . ] obj geographic\_operator object\_expression*

(где *tablename* – имя таблицы, *geographic\_operator* – географический оператор, *object\_expression* – выражение из постоянных строковых величин)

и вида:

*RowID = constant\_expression*

**RowID** – это имя специальной колонки, содержащей номер записи в таблице и обычно скрытой от пользователя.

### Примеры:

В этом примере выбираются все клиенты в Московской, Ленинградской и Ярославской областях. Каждая запись о клиенте не обязательно содержит имя области; запрос вместо этого опирается на географическое расположение клиента.

```
Select * From Клиенты
Where obj Within Any(Select obj From RUSSIA
    Where A66p = "МОС" or A66p = "СПБ" or A66p = "ЯРС")
```

В этом примере демонстрируется действие подзапроса. Мы желаем выбрать все территории сбыта, в которых проживают клиенты, имеющие признак "Кадастр". Подзапрос сначала выбирает подмножество записей о клиентах с признаком "Кадастр", после чего основной запрос выбирает территории по значениям из подмножества.

```
Select * From Территории  
Where obj Contains Any (Select obj From клиенты  
Where клиенты.тип = "Кадастр")
```

Следующий запрос выбирает все земельные участки, пересекающиеся с участком номер 120059.

```
Select * From Участки  
Where obj Intersects (Select obj From Участки  
Where Ном_участка = 120059)
```

**Смотрите также:**

[Open Table](#)

### Функция SelectionInfo( )

#### Назначение:

Возвращает информацию о текущем выборе в таблицах (информацию о временной таблице "Selection").

**Замечание:** Выбранные подписи не присоединяются к выбору, так как сами являются атрибутами других объектов и не являются отдельными объектами.

#### Синтаксис:

**SelectionInfo**(*attribute*)

где

*attribute* – целочисленный код.

#### Величина, полученная в результате:

Целое число или строка в зависимости от значения *attribute*. Величина типа Integer или String.

#### Описание:

В файле стандартных определений MapBasic MAPBASIC.DEF определены имена для кодов, которые можно использовать в функции **SelectionInfo**( ).

#### Значения *attribute*

#### Результат SelectionInfo( )

SEL-INFO-TABLENAME

Строковая величина. Имя таблицы, на базе которой создана таблица "Selection". MapBasic вернет ошибку, если не было выбора и Вы вызвали функцию с аргументом SEL-INFO-TABLENAME.

SEL-INFO-SELNAME

Строковая величина. Имя временной таблицы (например, "Запрос1"). Если ничего не выбрано, будет возвращена ошибка.

SEL-INFO-NROWS

Целое число. Количество выбранных строк в таблице. Если ничего не выбрано, будет возвращен 0 (ноль).

Для использования имен кодов необходимо включить в текст Вашей программы оператор **Include "MAPBASIC.DEF"**.

**Замечание:** Если записи выбраны в таблице, которая является объединением двух или более таблиц, то функция **SelectionInfo(SEL\_INFO\_NROWS)** возвращает количество строк в базовой таблице, которое может быть меньшим, чем количество строк в таблице "Selection". Смотрите пример ниже.

#### Ошибки:

В результате выполнения функции может генерироваться код ошибки ERR\_FCN\_ARG\_RANGE, если неправильно значение аргумента.

#### Пример:

Оператор **Select** используется для объединения. После всего переменная **i** будет равна 40 (числу строк, выбранных в базовой таблице RUSSIA), а переменная **j** будет равна 125 (числу строк таблицы запроса).

```
Dim i, j As Integer
Select * From RUSSIA, City200
    Where RUSSIA.obj Contains City200.obj Into РЕЗУЛЬТАТ
i = SelectionInfo(SEL_INFO_NROWS)
j = TableInfo(РЕЗУЛЬТАТ, TAB_INFO_NROWS)
```

**Смотрите также:**

[Select, TableInfo\( \)](#)

### Оператор Server Begin Transaction

#### Назначение:

Посылает уведомление на удаленный сервер о начале нового сеанса работы.

#### Синтаксис:

**Server *ConnectionNumber* Begin Transaction**

*ConnectionNumber* – целое число, номер соединения.

#### Описание:

Оператор **Server Begin Transaction** используется для обозначения начала сеанса обработки транзакций. Результаты последующих операторов языка SQL Insert, Delete и Update (внести, удалить и обновить), выполняемых функцией **Server\_Execute()**, не сохраняются в базе данных до тех пор, пока не будет выполнена команда **Server Commit**. Команда **Server Rollback** используется для отмены изменений.

#### Пример:

```
Dim hdbc As Integer
hdbc = Server_Connect("QELIB", "DSN=ORACLE7;DLG=1")
Server hdbc Begin Transaction
' ... Другие операторы ...
Server hdbc Commit
```

#### Смотрите также:

**Server Commit, Server Rollback**



## Оператор Server Bind Column

### Назначение:

Назначает локальную область хранения для удаленного сервера базы данных.

### Синтаксис:

**Server StatementNumber Bind Column *n* To Variable, StatusVariable**

*StatementNumber* – целое значение, номер SQL-оператора.

*n* – номер столбца в результирующем наборе, связываемого с переменной.

*Variable* – MapBasic-переменная для хранения значения столбца после выборки.

*StatusVariable* – переменная состояния, в которую записывается код, указывающий статус значения: пустое, усеченное или целое положительное значение.

### Описание:

Команда **Server Bind Column** назначает переменную приложения для сохранения значения столбца в результирующем наборе, специфицированном удаленным SQL-оператором **Select**. Когда последующий оператор **Server Fetch** выбирает строку данных из базы, значение столбца *n* присваивается этой переменной. Статус результата сохраняется в переменной состояния, указанной параметром *StatusVariable*.

#### Значение StatusVariable

#### Условие

SRV\_NULL\_DATA

Возвращается, если столбец не имеет данных в полученной строке (пустое значение).

SRV\_TRUNCATED\_DATA

Возвращается, если столбец содержит больше данных, чем может быть сохранено в указанной MapBasic-переменной.

Целое положительное значение

Число байт, возвращенное сервером данных.

### Пример:

```
' Приложение для "печати" адресных этикеток
' Предполагается, что существует реляционная таблица ADDR с 6 столбцами
Dim hdbc, hstmt As Integer
Dim first_name, last_name, street, city, state, zip As String
Dim fn_stat, ln_stat, str_stat, ct_stat, st_stat, zip_stat As Integer
hdbc = Server_Connect("QELIB", "DSN=ORACLE7;DLG=1")
hstmt = Server_Execute(hdbc, "select * from ADDR")
Server hstmt Bind Column 1 To first_name,fn_stat
Server hstmt Bind Column 2 To last_name, ln_stat
Server hstmt Bind Column 3 To street, str_stat
Server hstmt Bind Column 4 To city, ct_stat
Server hstmt Bind Column 5 To state, st_stat
Server hstmt Bind Column 6 To zip, zip_stat
Server hstmt Fetch NEXT
While Not Server_Eot(hstmt)
    Print first_name + " " + last_name
    Print street
    Print city + ", " + state + " " + zip
```

## Оператор Server Bind Column

---

```
        Server hstmt Fetch NEXT  
    Wend  
    Server hstmt Close  
    Server hdbc Disconnect
```

### Смотрите также:

[Server\\_ColumnInfo\(\)](#)

## Оператор Server Close

### Назначение:

Освобождает ресурсы, занятые удаленным SQL-оператором доступа к данным.

### Синтаксис:

*Server StatementNumber Close*

*StatementNumber* – целое значение, номер SQL-оператора.

### Описание:

Оператор **Server Close** используется для оповещения сервера о завершении обработки текущего удаленного SQL-оператора. Все ресурсы, ассоциированные с исполнением этого оператора, возвращаются в распоряжение системы.

Не забывайте вызывать оператор **Server Close** сразу же после исполнения функции **Server\_Execute()** для любого не выполняющего запрос данных SQL-оператора, обработка которого закончена в Вашем приложении.

### Пример:

```
' Выбирает пятую запись и закрывает SQL-оператор Select
hstmt = Server_Execute(hdbc, "Select * from Massive_Database")
Server hstmt Fetch Rec 5
Server hstmt Close
```

### Смотрите также:

**Server\_Execute()**

### Функция `Server_ColumnInfo( )`

#### Назначение:

Возвращает информацию о столбцах результирующего набора.

#### Синтаксис:

**`Server_ColumnInfo(StatementNumber, ColumnNo, Attr)`**

*StatementNumber* – целое значение, номер SQL-оператора.

*ColumnNo* – номер столбца в наборе; нумерация слева направо, начиная с 1.

*Attr* – код, указывающий характер возвращаемой информации.

#### Возвращаемое значение:

Возвращаемое значение зависит от значения атрибута (параметр *Attr*).

#### Описание:

Функция **`Server_ColumnInfo()`** возвращает информацию о текущем выбранном столбце результирующего набора (определенного ранее исполненным SQL-оператором **`Select`**) в удаленной базе данных. Параметр *StatementNumber* задает номер-указатель (handle) SQL-оператора, ассоциированный с данным соединением с сервером данных. Параметр *ColumnNo* указывает столбец, информацию о котором Вы хотите получить. Параметр *Attr* выбирает тип возвращаемой информации.

В следующей таблице перечислены возможные атрибуты (значения параметра *Attr*; определены в файле MAPBASIC.DEF).

#### Атрибут

SRV\_COL\_INFO\_NAME

SRV\_COL\_INFO\_TYPE

#### **`Server_ColumnInfo()` возвращает:**

Имя столбца.

Целый результат; код типа столбца:

SRV\_COL\_TYPE\_NONE

SRV\_COL\_TYPE\_CHAR

SRV\_COL\_TYPE\_DECIMAL

SRV\_COL\_TYPE\_INTEGER

SRV\_COL\_TYPE\_SMALLINT

SRV\_COL\_TYPE\_DATE

SRV\_COL\_TYPE\_LOGICAL

SRV\_COL\_TYPE\_FLOAT

SRV\_COL\_TYPE\_FIXED\_LEN\_STRING

SRV\_COL\_TYPE\_BIN\_STRING

Информацию об интерпретации типов данных приложением MapInfo Вы можете найти в *Руководстве пользователя MapBasic*.

SRV_COL_INFO_SCALE	Целый результат, указывающий число разрядов справа от десятичной точки для столбца типа SRV_COL_TYPE_DECIMAL, или -1 для столбца любого другого типа.
SRV_COL_INFO_PRECISION	Целый результат, указывающий общее число разрядов для столбца типа SRV_COL_TYPE_DECIMAL, или -1 для столбца любого другого типа.
SRV_COL_INFO_WIDTH	Целый результат, указывающий максимальное число символов в столбце типа SRV_COL_TYPE_CHAR или SRV_COL_TYPE_FIXED_LEN_CHAR. <b>Замечание:</b> При использовании модуля QELIB пустой терминатор не учитывается. <b>Замечание:</b> Возвращаемое значение совпадает с шириной столбца таблицы базы данных.
SRV_COL_INFO_VALUE	Тип результата варьируется. Возвращается актуальное значение данных в столбце для текущей выбранной записи. Длинные строковые значения столбца, превышающие 32766 символов, усекаются. Двоичные (неструктурированные) значения столбца возвращаются в виде шестнадцатиричных символьных строк двойной длины.
SRV_COL_INFO_STATUS	Целый результат; статус значения столбца: SRV_NULL_DATA – возвращается, если столбец не имеет данных для выбранной строки. SRV_TRUNCATED_DATA – возвращается, если в столбце содержится больше данных, чем может быть сохранено в указанной переменной. Положительное целое значение – число байт, возвращенных сервером данных.
SRV_COL_INFO_ALIAS	Строковый результат; псевдоним столбца, если в запросе данных использовался псевдоним.

### Пример:

```
Dim hdbc, Stmt As Integer
Dim Col As Integer
hdbc = Server_Connect("QELIB", "DSN=ORACLE7;DLG=1")
Stmt = Server_Execute(hdbc, "Select * from emp")
Server Stmt Fetch NEXT
For Col = 1 To Server_NumCols(Stmt)
    Print Server_ColumnInfo(Stmt, Col, SRV_COL_INFO_NAME) + " = " +
        Server_ColumnInfo(Stmt, Col, SRV_COL_INFO_VALUE)
Next
```

### Смотрите также:

**Server Bind Column, Server Fetch, Server\_NumCols()**

### Оператор Server Commit

#### Назначение:

Вызывает фиксацию транзакции в удаленной базе данных.

#### Синтаксис:

**Server** *ConnectionNumber* **Commit**

*ConnectionNumber* – целое значение, номер соединения с сервером данных.

#### Описание:

Оператор **Server Commit** фиксирует транзакцию, т.е. сохраняет в базе данных изменения, произведенные в данном сеансе соединения всеми удаленными SQL-операторами, выполненными с момента исполнения оператора **Server Begin Transaction**. Оператор **Server Commit** выполняется только при наличии открытой транзакции, инициированной оператором **Server Begin Transaction**. Для запуска новой транзакции Вы должны выдать серверу новый оператор **Server Begin Transaction**, за которым в дальнейшем должен быть исполнен оператор **Server Commit**.

#### Пример:

```
hdbc = Server_Connect("QELIB", "DSN=ORACLE7;DLG=1")
Server hdbc Begin Transaction
hstmt = Server_Execute(hdbc, "Update Emp Set salary = salary * 1.5")
Server hdbc Commit
```

#### Смотрите также:

**Server Begin Transaction, Server Rollback**

## Функция **Server\_Connect( )**

### Назначение:

Устанавливает соединение с удаленным сервером данных.

### Синтаксис:

**Server\_Connect**(*toolkit*, *connect\_string*)

*toolkit* указывает модуль интерфейса удаленного доступа MapInfo, "ODBC", "ORAINET", через который будет осуществляться соединение с сервером данных. Значения могут быть получены из функции **Server\_DriverInfo( )**.

*connect\_string* параметр, который предоставляет интерфейскому модулю дополнительную информацию, необходимую для подключения к серверу данных.

### Возвращаемое значение:

Целое.

### Описание:

Функция **Server\_Connect()** выполняет соединение с базой данных и возвращает номер-указатель соединения, который должен быть передан всем операторам удаленного доступа (как параметр *ConnectionNumber*), которые Вы хотите выполнить в данном сеансе соединения с сервером данных.

Параметр *toolkit* определяет модуль интерфейса удаленного доступа MapInfo (динамически загружаемую библиотеку), через который будет осуществляться соединение с сервером данных.

Информация о возможных значениях параметра может быть получена вызовом функций **Server\_NumDrivers** и **Server\_DriverInfo()**.

Параметр *connect\_string* передает модулю *toolkit* дополнительную информацию, необходимую для подключения к серверу данных. Значение строки подключения определяется требованиями удаленного сервера данных, к которому осуществляется доступ.

Строка подключения, задаваемая в функции **Server\_Connect()**, имеет формат:

*attribute=value[;attribute=value...]*

(В строке подключения должны отсутствовать пробелы.)

Прохождение соединения с DLG=1 обеспечивает удобный диалог соединения с активными кнопками справочной системы.

### Атрибуты **Microsoft ACCESS**

В следующей таблице перечислены атрибуты, используемые СУБД ACCESS:

<u>Атрибут</u>	<u>Описание</u>
DSN	Имя ODBC-источника для Microsoft ACCESS.
UID	Регистрационный идентификатор пользователя ID.
PWD	Пароль пользователя.

## Функция **Server\_Connect( )**

---

SCROLL	По умолчанию присваивается значение YES. Если SCROLL=NO, то библиотека ODBC не используется для этого соединения, дающего возможность вызывать первую, последнюю, предыдущую или произвольную запись в базе данных.
--------	---

Пример строки подключения для СУБД ACCESS:

"DSN=MI ACCESS;UID=ADMIN;PWD=SECRET"

### Подключение к **ORACLE** через **ODBC**

Если Ваше приложение требует передачи строки параметров для обеспечения подключения к источнику данных, необходимо указать имя источника данных. Это имя, источника данных уже настроенного для использования в Вашей системе, который будет вызван драйвером, для определения стандартных параметров, обеспечивающих подключение. Кроме того, можно явно указать пары "параметр-значение параметра", которые могут быть использованы для подключения к базе данных без применения уже зарегистрированного в системе источника данных. В этом случае значения параметров не будут записаны в качестве системных.

Можно использовать как полные имена параметров, так и их сокращенную форму:

**DSN=data\_source\_name[;attribute=value[;attribute=value]...]**

Пример строки подключения к базе данных Oracle:

**DSN=Accounting;HOST=server1;PORT=1522;SID=ORCL;UID=JOHN;PWD=XYZZY**

Ниже перечислены полные имена параметров, их сокращенные формы, возможные используемые значения и краткие описания. Отдельно отмечены стандартные значения для каждого параметра, используемые при отсутствии этого параметра в строке подключения, как при явном указании имени параметра, так и при использовании системного имени подключения. При настройке параметра в подключении, имеющем имя в Вашей системе, такие настройки будут использованы как стандартные.

**ApplicationUsingThreads (AUT):** ApplicationUsingThreads={0 | 1}. Обеспечивает работу драйвера с многопоточковыми приложениями.

Если установлен значение равное 1 (используется по умолчанию), драйвер гарантирует работу в каждом потоке.

При использовании приложений, работающих с единственным потоком, можно установить значение параметра, равное 0. В этом случае не требуется дополнительная обработка запроса, требующаяся для обеспечения стандартов безопасности работы драйвера ODBC с несколькими потоками данных.

**ArraySize (AS):** Количество байт, которое использует драйвер при обращении к нескольким записям. Может принимать значение от 1 до 4GB. Большое значение увеличивает производительность за счет меньшего числа обращений по сети. Меньшее значение параметра уменьшает время отклика сервера, так как приходится передавать меньше данных.

В качестве стандартного используется значение 60 000.

**CatalogOptions (CO):** CatalogOptions={0 | 1}. Определяет выполнение команд REMARKS для функций обращения к спискам SQLTables и SQLColumns и COLUMN\_DEF для функции обращения к списку SQLColumns при использовании Oracle. Если необходимо получить актуальные для сервера результаты выполнения таких команд, то необходимо установит значение CO равным 1.



Стандартно используется значение 0.

**DataSourceName (DSN):** Строка, определяющая имя источника данных в Вашей системе, обеспечивающего подключение к Oracle. В примерах используются имена "Accounting" или "Oracle-Serv1".

**DescribeAtPrepare (DAP):** DescribeAtPrepare={0 | 1}. Передает драйверу указание создавать описание команды SQL при ее подготовке.

Стандартно используется значение 0, которое не приводит к созданию описания команды SQL при ее подготовке драйвером.

**EnableDescribeParam (EDP):** EnableDescribeParam={0 | 1}. Выключает или включает использование ODBC API функции SQLDescribeParam. Применение этой функции обеспечивает использование описателей для всех параметров, имеющих тип данных SQL\_VARCHAR.

При использовании для доступа к данным Microsoft Remote Data Objects (RDO) этот атрибут необходимо установить, равным 1. Стандартно, по умолчанию, используется значение, равное 0.

**EnableStaticCursorsForLongData (ESCLD):** EnableStaticCursorsForLongData={0 | 1}.

Определяет поддержку драйвером использования статического курсора при обращении к столбцам, имеющим тип данных Long. Поддержка статического курсора приводит к потерям производительности при обращении к данным, имеющим тип данных Long.

Стандартно используется значение, равное 0.

**HostName (HOST):** HostName={servername | IP\_address}. Определяет имя сервера Oracle, к которому осуществляется подключение. Если Ваша сеть допускает обращение к серверу по имени, то можно указать нужное имя сервера, например, "Oracleserver", во всех остальных случаях следует указать адрес IP, например, 199.226.224.34.

**LockTimeOut (LTO):** LockTimeOut={0 | -1}. Управляет включением блокировки ожидания отклика для генерации сообщения об ошибке при обработке запроса вида Select ...For Update.

Если установлено значение 0, то Oracle обрабатывает ошибку без задержки.

Если установлено значение 1 (по умолчанию), задержка ожидания включена, но её значение не определено.

**LogonID (UID):** Задаёт идентификатор (имя) пользователя при входе в систему. Это имя используется приложением при установлении подключения (соединения) к базе данных Oracle. Этот идентификатор требуется только в том случае, если включены средства обеспечения безопасности и защиты базы данных. Для получения необходимого идентификатора следует обратиться к администратору базы данных.

**Password (PWD):** Пароль, который используется приложением для подключения к базе данных Oracle.

**PortNumber (PORT):** Устанавливает номер порта службы контроля соединений (listener) Oracle. По умолчанию принято стандартное для Oracle значение 1521. Правильное значение можно узнать у администратора базы данных.

**ProcedureRetResults (PRR):** ProcedureRetResults={0 | 1}. Устанавливает возможность получения результатов от хранимых процедур функций.

## Функция `Server_Connect( )`

---

Если установлено значение 0 (используется по умолчанию) драйвер не возвращает результаты выполнения хранимых процедур.

Если установлено значение 1, драйвер возвращает результаты выполнения хранимых процедур. В случае, если установлено 1, но после выполнения хранимой процедуры результат не может сформирован, следует ожидать небольшой потери производительности.

**SID (SID): Системный идентификатор Oracle, по которому обращаются к экземпляру базы данных Oracle.**

**UseCurrentSchema (UCS):** UseCurrentSchema={0 | 1}. Определяет, что при выполнении запросов вида SQLProcedures драйвером используются настройки только текущего пользователя.

Если установлено значение, равное 0, то драйвер не определяет пользователя, от имени которого выполняется запрос.

Если установлено значение, равное 1 (используется по умолчанию), то обращение к SQLProcedures оптимизируется, но возвращаются только процедуры, право применения которых, разрешено пользователю, от чьего имени осуществляется запрос.

### Атрибуты Oracle8i Spatial

Oracle8i Spatial это новое издание пространственной базы данных от Oracle Corporation. Здесь есть сходство с ранним Oracle SDO. MapInfo не поддерживает реляционную схему Oracle SDO через OCI. MapInfo не поддерживает одновременные соединения с Oracle8i через OCI и с другими базами данных через ODBC. MapInfo не поддерживает загрузку геометрических таблиц Oracle8i Spatial через ODBC используя текущий драйвер ODBC фирмы Intersolv. Здесь нет компонента DSN.

Атрибут	Описание
LogonID (UID)	Имя пользователя (logon ID) которое приложение использует для связи с Вашей базой данных Oracle. Имя пользователя требуется, если на Вашей базе есть возможность установить защиту данных. В этом случае обратитесь к системному администратору для получения имени.
Password (PWD)	Ваш пароль. Его тоже выдает системный администратор.
ServerName (SRVR)	Имя сервера Oracle.

Пример строки соединения для доступа к серверу Oracle8i Spatial с использованием TCP/IP:

**"SRVR=FATBOY;USR=SCOTT;PWD=TIGER"**

### Атрибуты SQL SERVER

Если Ваше приложение требует передачи строки параметров для обеспечения подключения к источнику данных, необходимо указать имя источника данных. Это имя, уже настроенного в Вашей системе источника данных, который будет вызван драйвером, для определения стандартных параметров, обеспечивающих подключение. Кроме того, можно явно указать пары "параметр-значение параметра", которые могут быть использованы для подключения к базе данных без применения уже зарегистрированного в системе источника данных ODBC. В этом случае значения параметров не будут записаны в качестве системных.

**DSN=data\_source\_name[;attribute=value[;attribute=value] ...]**

Пример строки подключения к базе данных SQL Server:

**DSN=Accounting;UID=JOHN;PWD=XYZZY**

Ниже перечислены полные имена параметров, их сокращенные формы, возможные используемые значения и краткие описания. Отдельно отмечены стандартные значения для каждого параметра, используемые при отсутствии этого параметра в строке подключения, как при явном указании имени параметра, так и при использовании системного имени подключения. При настройке параметра в подключении, имеющем имя в Вашей системе, такие настройки будут использованы как стандартные.

**Address:** сетевой адрес сервера, на котором выполняется SQL Server. Этот параметр следует использовать только в том случае, если в параметре Server явно не указано имя сервера на котором работает сервер базы данных SQL Server. Адресом может служить имя сервера в сети, а также другие способы адресации к этой вычислительной машине, например, имя или номер канала, TCP/IP адрес и номер порта, адрес программного интерфейса (socket). Например, для TCP/IP: 199.199.199.5, 1433 или MYSVR, 1433.

**AnsiNPW:** AnsiNPW={yes | no}. Определяет использование правил ANSI.

В случае, если установлено значение, равное логической единице, драйвером используются правила ANSI (американского национального института стандартов), в которых определяются способы обработки сравнений значений с пустыми элементами (NULL), последовательности символов и конкатенации значений с пустыми (NULL) элементами.

В случае, если установлено значение, равное логическому нулю правила ANSI не применяются..

**APP:** Определяет имя приложения, вызывающего команду SQLDriverConnect (может использоваться для дополнительной проверки условий выполнения запроса). Если этот параметр используется, то значение указанное при его вызове будет сохранено в столбце master.dbo.sysprocesses . При вызове функций sp\_who и Transact-SQL APP\_NAME возвращается значение program\_name.

**AttachDBFileName:** Имя основного (primary) файла базы данных, к которой выполняется подключение. Должно содержать полный путь. Необходимо использовать выделение всех символов "косая черта" ( \ ) в случае, если используется объявление символьных переменных в C нотации: AttachDBFileName=c:\\MyFolder\\MyDB.mdf

Устанавливает соединение с указанной базой данных и эта база данных становится используемой по умолчанию для этого подключения. Для того, что бы применять адресацию к базе данных по имени файла с помощью параметра AttachDBFileName, необходимо объявлять соответствующее значение в параметре SQLDriverConnect, DATABASE или в атрибуте подключения через SQL\_COPT\_CURRENT\_CATALOG.

**AutoTranslate:** AutoTranslate={yes | no}. Определяет способ перекодирования символов ANSI.

В случае если установлено значение, равное логической единице, последовательности символов ANSI передаваемые между сервером и клиентом перекодируются в соответствии с кодовыми страницами Unicode. Таким образом удастся решить проблему соответствия дополнительных наборов символов на сервере и у клиента.

Перекодировка выполняется на вычислительной машине клиента драйвером SQL Server Wire Protocol. При этом необходимо, чтобы на сервере и у клиента были установлены одни и те же наборы

## Функция **Server\_Connect( )**

---

кодových страниц ANSI(ACP).

Настройка этого параметра не влияет на перекодировку символов при передаче данных следующих типов:

Данные от клиента типа Unicode SQL\_C\_WCHAR передаются на сервер с типом данных char, varchar или text.

Данные с сервера, имеющие тип char, varchar или text передаются клиенту для переменной типа Unicode SQL\_C\_WCHAR.

Данные от клиента типа ANSI SQL\_C\_CHAR передаются на сервер в переменную с типом данных Unicode nchar, nvarchar или ntext

Данные с сервера, имеющие тип Unicode char, varchar или text передаются клиенту для переменной типа ANSI SQL\_C\_CHAR.

Если значение параметра установлено равным логическому нулю (No) перекодировка не проводится.

Драйвер SQL Server Wire Protocol не перекодирует символы ANSI SQL\_C\_CHAR при передаче от клиента на сервер данных в переменные типов char, varchar или text, параметров и имен столбцов. Не перекодируются символы из переменных типов char, varchar или text при передаче данных от сервера клиенту в переменные типа SQL\_C\_CHAR.

Если на клиенте и на сервере SQL Server используются разные активные кодовые страницы (ACP), то в этом случае дополнительные символы могут обрабатываться неправильно.

**DATABASE:** Имя базы данных SQL Server, используемой в устанавливаемом подключении по умолчанию. Если этот параметр не задан, используется база данных определенная, как используемая по умолчанию для пользователя при входе в систему. Имя базы данных, определенное, как используемая по умолчанию, в источнике данных ODBC, будет использовано вместо имени базы данных, определенной по умолчанию для пользователя при входе в систему, в случае если подключение устанавливается по имени источника данных ODBC. База данных должна существовать, кроме случая, когда применяется параметр AttachDBFileName. При совместном применении с параметром AttachDBFileName, используется основной (primary) файл, на который указывает значение параметра AttachDBFileName, а базе данных присваивается имя, определенное значением параметра DATABASE.

**LANGUAGE:** может использоваться как дополнительный, не обязательный параметр. SQL Server может хранить системные сообщения на разных языках. При подключении к серверу SQL Server, для которого предусмотрена возможность выдавать системные сообщения на разных языках, можно указать язык, на котором следует создавать такие сообщения.

**Network:** Имя сетевой библиотеки. Для этого имени не нужно указывать полного пути и расширения имени файла .dll, например, Network=dbnmpntw.

**PWD:** Пароль, указываемый пользователем при входе в SQL Server, который определен в переменной UID. PWD не нужно определять, если для входа в систему установлен пустой пароль или в случае использования авторизации пользователей средствами Windows NT (Trusted\_Connection=yes).

**QueryLogFile:** Полное имя файла, который будет использоваться для создания журнала обработки данных при продолжительных запросах.

**QueryLog\_On:** QueryLog\_On={yes | no}. Включает процесс создания журнала обработки

продолжительных запросов.

Если значение параметра установлено равным логической единице (Yes), то журнал обработки долгоиграющих запросов для этого подключения ведется.

Если значение параметра установлено равным логическому нулю (No), то журнал обработки продолжительных запросов не будет вестись.

**QueryLogTime:** Последовательность цифр, определяющих интервал (в миллисекундах) записи журнала продолжительных запросов. Если отклик на запрос не пришел в течении заданного интервала, то этот запрос будет добавлен в журнал.

**QuotedID:** QuotedID={yes | no}. Включает или выключает обработку идентификаторов в кавычках. Если значение параметра установлено равным логической единице (Yes), то параметр QUOTED\_IDENTIFIERS — включен. SQL Server, в этом случае, использует правила SQL-92, определяющие использование кавычек в предложениях SQL.

Если значение параметра установлено равным логическому нулю (No), то параметр QUOTED\_IDENTIFIERS — выключен. В этом случае, SQL Server, использует правила Transact-SQL, определяющие использование кавычек в предложениях SQL.

**Regional:** Regional={yes | no}. Включает автоматическое преобразование формы представления символов валюты, даты и времени.

Если значение параметра установлено равным логической единице (Yes), то драйвер SQL Server Wire Protocol использует настройки машины клиента при преобразовании символов валюты, даты и времени. Это преобразование — одностороннее: драйвер не распознает форматы символов валют или даты, не определенные стандартом ODBC.

Если значение параметра установлено равным логическому нулю (No), то драйвер использует стандартные в ODBC форматы представления валюты, даты и времени и в соответствии с ними преобразует такие данные в последовательности символов.

**SAVEFILE:** Имя файла источника данных ODBC, в который записываются атрибуты соединения при успешном подключении.

**SERVER:** Имя сервера в сети, на котором работает SQL Server. В качестве такого имени можно использовать либо имя существующего в сети компьютера, либо имя, описанное как существующее в SQL Server Client Network Utility. В случае использования копии SQL Server на том же самом компьютере, с которого выполняется подключение, можно ввести стандартное имя "(local)", как имя сервера Windows NT.

**StatsLogFile:** Полное имя, включая путь, файла, который будет использоваться для ведения журнала операций драйвера SQL Server Wire Protocol.

**StatsLog\_On:** StatsLog\_On={yes | no}. Включает и выключает сбор статистики операций, выполняемых драйвером SQL Server Wire Protocol.

Если значение параметра установлено равным логической единице (Yes), то сведения об операциях, выполненных драйвером SQL Server Wire Protocol заносятся в журнал.

Если значение параметра установлено равным логическому нулю (No), то сведения об операциях драйвера SQL Server Wire Protocol, выполненных с использованием этого подключения, не сохраняются.

**Trusted\_Connection:** Trusted\_Connection={yes | no}. Определяет сведения, которые будут

## Функция **Server\_Connect( )**

---

использоваться драйвером SQL Server Wire Protocol, для подтверждения полномочий пользователя при входе в систему.

Если значение параметра установлено равным логической единице (Yes), то драйвер SQL Server Wire Protocol будет работать в режиме проверки прав пользователей при входе в систему Windows NT Authentication Mode. Дополнительно можно задать параметры UID и PWD.

Если значение параметра установлено равным логическому нулю (No), то, для подтверждения прав доступа к данным пользователем, драйвер SQL Server Wire Protocol будет использовать имя и пароль пользователя базы данных SQL Server. Обязательно нужно задать параметры UID и PWD.

**UID:** Полноценная учетная запись входа в систему SQL Server. Пользуясь штатными средствами подтверждения входа в систему Windows NT, параметр UID можно не задавать.

**WSID:** Идентификатор рабочей станции. Обычно, сетевое имя компьютера, на котором выполняется приложение. Использовать этот параметр — не обязательно. Если этот параметр задается, то его значение будет добавлено в столбец hostname таблицы master.dbo.sysprocesses. Это значение будет возвращаться по вызову функций sp\_who и Transact-SQL HOST\_NAME.

### Атрибуты Informix

Если Ваше приложение требует передачи строки параметров для обеспечения подключения к источнику данных, необходимо указать имя источника данных. Это имя, уже настроенного в Вашей системе источника данных, который будет вызван драйвером, для определения стандартных параметров, обеспечивающих подключение. Кроме того, можно явно указать пары “параметр-значение параметра”, которые могут быть использованы для подключения к базе данных без применения уже зарегистрированного в системе источника данных ODBC. В этом случае значения параметров не будут записаны в качестве системных.

Можно использовать, как полные имена параметров, так и их сокращенную форму:  
Строка подключения имеет следующий вид:

**DSN=data\_source\_name[;attribute=value[;attribute=value]...]**

Пример строки подключения к базе данных Oracle:

**DSN=Informix TABLES;DB=PAYROLL**

Ниже перечислены полные имена параметров, их сокращенные формы, возможные используемые значения и краткие описания. Отдельно отмечены стандартные значения для каждого параметра, используемые при отсутствии этого параметра в строке подключения, как при явном указании имени параметра, так и при использовании системного имени подключения. После настройки параметра в подключении, имеющем имя в Вашей системе, такие настройки будут использованы как стандартные.

**ApplicationUsingThreads (AUT):** ApplicationUsingThreads={0 | 1}. Обеспечивает работу драйвера с многопоточковыми приложениями. Если установлен значение равное 1 (используется по умолчанию), драйвер гарантирует работу в каждом потоке. При использовании приложений, работающих с единственным потоком, можно установить значение параметра, равное 0. В этом случае не требуется дополнительная обработка запроса, требующаяся для обеспечения стандартов безопасности работы драйвера ODBC с несколькими потоками данных.

**CancelDetectInterval (CDI):** Определяет значение в секундах, используемое как интервал проверки драйвером выполнения запроса, который может быть прерван командой SQLCancel. При обнаружении драйвером того факта, что была применена команда SQLCancel, запрос отменяется. Если установлено значение, равное 0 (используется по умолчанию), выполнение запроса не

прерывается даже в случае применения команды SQLCancel.

Например, если установить значение параметра CancelDetectInterval равным 5, то драйвер будет проверять каждые пять секунд все ожидающие своей очереди на выполнение запросы, не пришло ли от приложения, в котором был создан этот запрос, прерывание его выполнения по команде SQLCancel.

**Database (DB):** Имя базы данных, к которой Вы хотите подключиться.

**DataSourceName (DSN):** Имя ODBC-источника данных для INFORMIX. В примерах используются имена "Accounting" или "Informix-Serv1."

**HostName (HOST):** Имя резидентной машины сервера INFORMIX.

**LogonID (UID):** Ваше имя пользователя для сервера INFORMIX.

**PortNumber (PORT):** Устанавливает номер порта службы контроля соединений (listener). Стандартного значения, используемого по умолчанию, нет.

**ServerName (SRVR):** Имя сервера на котором работает база данных Informix .

**TrimBlankFromIndexName (TBFIN):** TrimBlankFromIndexName={0 | 1}. Управляет способом обработки пустых символов или "пробелов" в начале имен индексов, автоматически созданных системой. Этот параметр предназначен для использования в приложениях, которые не умеют обрабатывать пробелы в начале имен индексов. Если установлено значение, равное 1 (используется по умолчанию), драйвер отрезает пробелы в начале имен индексов. Если установлено значение, равное 0, драйвер оставляет пробелы в начале имен индексов.

### Пример

```
Dim hdbc As Integer
hdbc = Server_Connect("ODBC",
"DSN=Informix;SRV=IUSSrvr;USR=atsmipro;PWD=miproats")
```

### Смотрите также

Server Disconnect



# Оператор Server Create Map

## Назначение

Этот оператор присваивает геоинформацию таблице MapInfo, связанной с таблицей в удаленной базе данных. Модификация таблицы (например, добавлением в таблицу столбцов с пространственной информацией) при этом не осуществляет.

## Синтаксис

**Server ConnectionNumber Create Map**

**For linked\_table**

**Type { MICODE columnname | XYINDEX columnname | SPATIALWARE }**

**CoordSys ...**

**[ ObjectType { Point | Line | Region | ALL } ]**

**[ Symbol (...) ]**

**[ Linestyle Pen(...) ]**

**[ Regionstyle Pen(...) Brush(...) ]**

**[ Style Type style\_number (0 or 1) [ Column column\_name ] ]**

*connectionNumber* номер, идентифицирующий соединение с сервером данных.

*linked\_table* имя открытой связанной ODBC-таблицы

*columnname* имя столбца, содержащего координаты специфицированного типа

**CoordSys ...** предложение, задающее координатную систему и проекцию

**ObjectType** предложение, задающее тип объекта в таблице: точки, линии, регионы или все объекты. Если это предложение не задано, по умолчанию используется тип точки.

**Symbol (...)** предложение, задающее стиль символа, используемого для точечного объекта

**Linestyle Pen (...)** предложение, определяющее стиль линии, используемый для объекта типа линия

**Regionstyle Pen (...) Brush(...)** предложение, задающее стиль линии и заливки фона, используемый для объекта типа область

**StyleType** устанавливает символы для отдельных строк. Когда **Type** установлено на 1 (единицу), то подпредложение **Column** и его аргумент должны быть представлены. Когда **Type** установлено на 1 (единицу). Когда *style\_number* установлен на ноль, то **Column** игнорируется и создаются колонки соответствия в MAPCATALOG.

## Описание

Оператор **Server Create Map** присваивает геоинформацию таблице MapInfo, связанной с таблицей в удаленной базе данных. Для таблицы SpatialWare, Oracle Spatial or Oracle SDO можно отразить на карте точки, линии и регионы. Для всех других таблиц можно отображать на карте только точки. Любая таблица MapInfo может быть отображена в окне Списка, но только таблица с геоинформацией может иметь присоединенные графические объекты, и только такие таблицы могут быть отображены в окнах Карты MapInfo.

**Внимание:** Если сервер это Oracle9i и система координат определена как Долгота/Широта без определения датума, то по умолчанию будет использоваться стандартный датум World Geodetic System 1984(WGS 84). Такое поведение согласуется с оператором Server Create Table и программой Easyloader.

### Типы атрибутов

### Описание

ORA\_SP

OracleSpatial



IUS_SW	SpatialWare IUS Blade
IUS_MM_SW >	MapInfo MapMarker Geocoding DataBlade for SpatialWare
IUS_MM_XY	MapInfo MapMarker Geocoding DataBlade for XY
<columnname>	
SPATIALWARE	SpatialWare for SQL Server
MICODE	XYINDEX

### Примеры

```

Sub Main
Dim ConnNum As Integer
ConnNum = Server_Connect("ODBC",
"DSN=SQLServer;DB=QADB;UID=mipro;PWD=mipro")
Server ConnNum Create Map For "Cities"
Type SPATIALWARE
CoordSys Earth Projection 1, 0
ObjectType All
ObjectType Point
    Symbol (35,0,12)
Server ConnNum Disconnect
End Sub

```

### Смотрите также

Server Link, Unlink

# Оператор Server Create Table

### Назначение

Создает новую таблицу в указанной удаленной базе данных.

### Синтаксис

**Server ConnectionNumber Create Table TableName (ColumnName ColumnType [...])**

<b>[KeyColumn</b>	<b>ColumnName]</b>
<b>[ObjectColumn</b>	<b>ColumnName]</b>
<b>[StyleColumn</b>	<b>ColumnName]</b>
<b>[CoordSys...</b>	<b>]</b>

*ConnectionNumber* целое, идентификатор соединения с базой данных.

*TableName* имя таблицы, которую Вы создаете в удаленной базе данных.

*ColumnName* имя создаваемой колонки. Имя колонки может быть длиной до 31 символа, может содержать буквы, числа и символ подчеркивания(\_). Имя колонки не может начинаться с цифры.

*ColumnType* тип данных, ассоциированных с колонкой.

**KeyColumn** предложение, определяющее ключевую колонку таблицы.

**ObjectColumn** предложение, определяющее колонку пространственной геометрии/объектов таблицы.

**StyleColumn** предложение, определяющее колонку Per Row Style, которая позволяет использовать различные стили объектов для каждой записи таблицы.

**CoordSys...** предложение, определяющее систему координат и проекцию.

### Описание

Оператор **Server Create Table** создает новую пустую таблицу в базе данных с числом колонок до 250.

Длина имени таблицы *TableName* изменяется в зависимости от типа баз данных. Мы рекомендуем использовать 14 или менее символов для имени таблицы, чтобы быть уверенными при работе с любой базой данных. Таким образом, пусть максимальная длина имени таблицы будет 14 символов.

*ColumnType* тот же тип данных, что и определенный в **Create Table Statement**. Некоторые типы данных могут быть конвертированы в те типы, которые поддерживаются используемой базой данных.

Если задано дополнительное предложение **KeyColumn**, то будет создан уникальный индекс для данной колонки. Мы рекомендуем использовать это предложение, так как оно позволяет MapInfo Professional открывать таблицу при прямом доступе к базе данных.

Дополнительное предложение **ObjectColumn** позволит Вам создать таблицу с колонкой пространственной геометрии/объектами. Если предложение определено, то пространственный индекс также будет создан для этой колонки. Таким образом, если сервер не имеет возможности обработать пространственную геометрию/объекты, то таблицы создана не будет. Если сервер это SQL Server со SpatialWare, то таблица будет настроена на пространственную геометрию/объекты с момента создания. Если сервер это Oracle Spatial, то пространственные метаданные обновятся в момент создания таблицы.

Если используется **Server Create Table** и предложение **ObjectColumn** пропущено в операторе, Вам также надо будет использовать Server Create Map для того, чтобы открыть таблицу в MapInfo Professional.

Дополнительное предложение **CoordSys...** становится обязательным только если таблица с пространственной геометрией/объектами создается на Oracle Spatial (Oracle8i или более поздние версии с пространственной поддержкой). Если Oracle9i является сервером и система координат определена как Долгота/Широта без указания датума, то будет использован стандартный датум 1984(WGS 84). Система координат должна быть такой же как и система определенная в операторе **Server Create Map Statement**. Для других баз данных это предложение не влияет на создание таблицы.

Поддерживаемые базы данных это Oracle, SQL Server, IUS и Microsoft Access. Таким образом, для создания таблицы с колонкой пространственной геометрии/объектами, SpatialWare/Blade требуется для SQL Server и IUS, а для Oracle требуются пространственные настройки.

### Примеры

Следующие примеры показывают как создать таблицу с именем ALLTYPES, которая содержит семь колонок, охватывающих каждый из типов данных, поддерживаемых MI Pro, плюс три колонки Key, SpatialObject и Style. Всего колонок должно быть десять.

Для SQL Server со SpatialWare или IUS со SpatialWare Blade:

```
dim hodb as integer
hodb = server_connect("ODBC", "dlg=1")
Server hodb Create Table ALLTYPES( Field1 char(10),Field2
integer,Field3 smallint,Field4 float,Field5 decimal(10,4),Field6
date,Field7 logical)
    KeyColumn      SW_MEMBER
    ObjectColumn    SW_GEOMETRY
    StyleColumn     MI_STYLE
```

Для Oracle Spatial:

```
dim hodb as integer
hodb = server_connect("ORAINET", "SRVR=cygnus;UID=mipro;PWD=mipro")
Server hodb Create Table ALLTYPES( Field1 char(10),Field2
integer,Field3 smallint,Field4 float,Field5 decimal(10,4),Field6
date,Field7 logical)
    KeyColumn      MI_PRINX
    ObjectColumn    GEOLOC
    StyleColumn     MI_STYLE
    Coordsys       Earth Projection 1, 0
```

### Смотрите также

Server Link, Unlink, Create Map, Server Create Map

# Оператор Server Create Style

### Назначение

Изменяет настройки стиля объекта для геокодированной таблицы. Этот оператор действует аналогично оператору **Server Set Map** и возвращает положительный или отрицательный результат.

### Синтаксис

**Server ConnectionNumber** Set Map *linked table...*

[ Style Type *style\_number* (0 or 1) [ Column *<column\_name>*] ]

*connectionNumber* - целое число, номер соединения.

*linked\_table* - имя открытой связанной ODBC-таблицы

*columnname* - имя столбца, содержащего координаты специфицированного типа.

**StyleType** устанавливает стиль для каждой записи. Символ и аргумент **Column** определяется, когда **Type** установлен на 1. Когда *style\_number* установлен на 0, то символ **Column** игнорируется и в Каталоге карт (MAPCATALOG) очищаются колонки исполнения (rendition columns).

### Описание

Значение **Column** используется и задается, когда параметр **Type** установлен на 1. Когда *style\_number* установлен на 0, то **Column** игнорируется, а в Каталоге карт (MAPCATALOG) очищаются колонки исполнения (rendition columns).

Чтобы оператор сработал правильно, Каталог карт должен иметь структуру, поддерживающую стили и должен содержать колонки **RENDITIONTYPE**, **RENDITIONCOLUMN** и **RENDITIONTABLE**. Команда не будет успешно выполнена, если колонки стилей не являются текстовыми. Оператор SQL сам выдаст ошибку, если попытается установить строковую величину в колонку с различными типами данных.

### Пример

```
Server 2 Create Map For "qadb:informix.arc"  
Type MICODE "mi_sql_micode" ("mi_sql_x","mi_sql_y")  
CoordSys Earth Projection 1, 0 ObjectType Point Symbol (35,0,12)  
Style Type 1 Column "mi_symbology"
```

### Смотрите также

Функция `Server_Connect()`

### Оператор Server Disconnect

#### Назначение:

Прекращает связь, установленную с удаленным сервером данных вызовом функции **Server\_Connect()**.

#### Синтаксис:

**Server** *ConnectionNumber* **Disconnect**

*ConnectionNumber* – номер, номер соединения с сервером данных.

#### Описание:

Оператор **Server Disconnect** отключает приложение от базы данных. Все ресурсы, выделенные для указанного соединения, возвращаются в распоряжение системы.

#### Пример:

```
Dim hdbc As Integer
hdbc = Server_Connect("QELIB", "DSN=ORACLE7;DLG=1")
Server hdbc Disconnect
```

#### Смотрите также:

**Server\_Connect()**

### Функция **Server\_DriverInfo( )**

#### Назначение:

Выдает информацию об установленных интерфейсных модулях и источниках данных.

#### Синтаксис:

**Server\_DriverInfo(DriverNo, Attr)**

*DriverNo* – целое значение, назначенное приложением MapInfo модулю интерфейса удаленного доступа при запуске MapInfo.

*Attr* – код, определяющий характер возвращаемой информации.

#### Возвращаемое значение:

Строка.

#### Описание:

Функция **Server\_DriverInfo()** возвращает информацию об источниках данных. Первый параметр выбирает модуль интерфейса удаленного доступа (начиная с 1). Общее число установленных интерфейсных модулей может быть получено вызовом функции **Server\_NumDrivers()**. Второй параметр (атрибут) *Attr* выбирает тип возвращаемой информации, как показано в следующей таблице:

#### Атрибут (Attr)

SRV\_DRV\_INFO\_NAME

SRV\_DRV\_INFO\_NAME\_LIST

SRV\_DRV\_DATA\_SOURCE

#### Server\_DriverInfo() возвращает:

Строку, показывающую имя модуля.

Строку – список имен всех установленных интерфейсных модулей, разделенных точками с запятой. Параметр *DriverNo* игнорируется.

Строку – имена источников данных, поддерживаемых указанным модулем интерфейса. Последовательные вызовы функции последовательно выбирают имена источников. После выборки последнего имени для данного модуля функция возвратит пустую строку. Следующий вызов функции для того же модуля установит список на начало и возвратит первое имя в списке.

#### Пример:

```
Dim dlg_string, source As String
dlg_string = Server_DriverInfo(0, SRV_DRV_INFO_NAME_LIST)
source = Server_DriverInfo(1, SRV_DRV_DATA_SOURCE)
While source <> ""
    Print " Доступные источники данных" +
        Server_DriverInfo(1, SRV_DRV_INFO_NAME) + ": " + source
    source = Server_DriverInfo(1, SRV_DRV_DATA_SOURCE)
Wend
```

#### Смотрите также:

**Server\_NumDrivers()**

## Функция **Server\_EOT( )**

### Назначение:

Определяет, был ли достигнут конец результирующего набора в процессе последовательной выборки записей, выполнявшейся оператором **Server Fetch**.

### Синтаксис:

**Server\_EOT** (*StatementNumber*)

*StatementNumber* – целое значение, номер SQL-оператора.

### Возвращаемое значение:

Логическое.

### Описание:

Функция **Server\_EOT()** возвращает TRUE, если предыдущий оператор выборки не нашел в результирующем наборе данных для возврата; в противном случае возвращает FALSE. Значение TRUE возвращается как при попытке выбрать предыдущую запись сразу же после выборки первой записи набора, так и в случае выборки следующей записи после последней записи набора.

### Пример:

```
Dim hdbc, hstmt As Integer
hdbc = Server_Connect("QELIB", "DSN=ORACLE7;DLG=1")
hstmt = Server_Execute(hdbc, "Select * from ADDR")
Server hstmt Fetch FIRST
While Not Server_EOT(hstmt)
    ' Обработка каждой строки данных ...
    Server hstmt Fetch Next
Wend
```

### Смотрите также:

**Server Fetch**

### Функция `Server_Execute()`

#### Назначение:

Посылает SQL-строку для исполнения на удаленный сервер данных.

#### Синтаксис:

**Server\_Execute**(*ConnectionNumber*, *server\_string*)

*ConnectionNumber* – номер соединения с сервером данных.

*server\_string* – строка, представляющая любой корректный SQL-оператор, поддерживаемый сервером, с которым установлено соединение.

#### Возвращаемое значение:

Целое.

#### Описание:

Функция **Server\_Execute** пересылает SQL-строку, заданную параметром *server\_string* и представляющую SQL-оператор, через соединение с сервером, указанное параметром *ConnectionNumber*. Любой корректный SQL-оператор, поддерживаемый активным сервером, является допустимым значением параметра *server\_string*. Информацию о корректных SQL-операторах смотрите в руководстве по языку SQL для СУБД на Вашем сервере.

Эта функция возвращает номер-указатель (handle) оператора, используемый для ассоциации (через параметр *StatementNumber*) последующих SQL-обращений (таких как **Fetch** и **Close**) с конкретным SQL-оператором.

Вы должны обеспечить выполнение оператора **Server Close** для каждого вызова функции **Server\_Execute()** как можно быстрее по завершении использования указателя SQL-оператора. Для операторов типа **Select** – после выборки требуемых данных. При этом на удаленном сервере данных будет закрыт курсор и освобожден результирующий набор. В противном случае Вы можете превысить лимит на число открытых курсоров, и дальнейшие обращения к базе данных исполняться не будут. Не все серверы баз данных поддерживают курсоры с прямой и обратной прокруткой. Для других SQL-операторов выдайте оператор **Server Close** сразу же после вызова функции **Server\_Execute**.

```
Dim hdbc, hstmt As Integer
hdbc = Server_Connect("QELIB", "DSN=ORACLE7;DLG=1")
hstmt = Server_Execute(hdbc, "Select * from ADDR")
Server hstmt Close
```

#### Пример:

```
Dim hdbc, hstmt As Integer
hdbc = Server_Connect("QELIB", "DSN=ORACLE7;DLG=1")
hstmt = Server_Execute(hdbc, "CREATE TABLE NAME_TABLE (NAME CHAR
(20))")
Server hstmt Close
hstmt = Server_Execute(hdbc, "INSERT INTO NAME_TABLE VALUES ('Steve')")
Server Close hstmt
hstmt = Server_Execute(hdbc, "UPDATE NAME_TABLE SET name = 'Tim'")
Server Close hstmt
Server hdbc Disconnect
```

#### Смотрите также:

**Server Close**, **Server Fetch**



## Оператор Server Fetch

### Назначение:

Осуществляет выборку записей результирующего набора с удаленного сервера данных.

### Синтаксис:

**Server StatementNumber Fetch** [NEXT|PREV|FIRST|LAST|([REC] *recno*)

или

**Server StatementNumber Fetch INTO Table** [FILE *path*]

*StatementNumber* – целое значение, номер SQL-оператора.

### Описание:

Оператор **Server Fetch** извлекает записи результирующего набора (заданного значением *StatementNumber* SQL-оператора, создавшего набор) из сервера данных. Для построчной выборки данных они помещаются в локальную область хранения, и могут быть связаны с переменными посредством команд **Server Bind Column**. Для выборки данных по столбцам применяется функция **Server\_ColumnInfo**(SRV\_COL\_INFO\_VALUE). Можно также в одной операции выбрать полный результирующий набор в таблицу MapInfo, используя предложение **INTO Table**.

Выполнение операторов **Server Fetch** и **Server Fetch Into** прерывается с установкой кода ошибки ERR() = ERR\_SRV\_ESC при нажатии пользователем клавиши ESC, что позволяет Вашему MapBasic-приложению использовать команды **Server Fetch** для обработки этого события.

По исполнении оператора **Server Fetch Into** таблица MapInfo фиксируется, и для нее нет незавершенных транзакций. Все символьные поля, превышающие 254 байта, усекаются; все двоичные (неструктурированные) поля загружаются в таблицу как шестнадцатиричные символьные строки двойной длины. Имена столбцов в загруженной таблице будут использовать псевдонимы столбцов, если в запросе задавались псевдонимы.

### Обработка Null-значений

Если был исполнен SQL-оператор **Select** с последующей выборкой записи, включающей столбец таблицы, который содержит пустое (null) значение, то происходит следующее. Поскольку в MapInfo не поддерживается концепция пустых значений в таблице или переменной, используется значение по умолчанию в рамках домена для соответствующего типа данных – значение MapBasic-переменной, декларированной в инструкции **Dim**, но не инициализированной.

Однако при этом обеспечивается индикация возврата пустого значения.

Для связанных переменных (см. оператор **Server Bind Column**) могут быть заданы переменные состояния, значения которых будут указывать на возврат пустого значения при выборке. Для столбцов, не связанных с переменными, функция **Server\_ColumnInfo()** с атрибутом SRV\_COL\_INFO\_STATUS будет возвращать статус столбца, информирующий о возможно пустом значении.

### Как MapInfo интерпретирует типы данных?

Смотрите в Приложении 4 *Руководства пользователя MapBasic* информацию о том, как MapInfo интерпретирует типы данных.

## Оператор Server Fetch

---

### Ошибки:

Оператор **Server *n Fetch Into Table*** будет генерировать ошибку для любых неудачных попыток вставки записей в локальную таблицу MapInfo. Операторы типа **Server *n Fetch*** [**Next**|**Prev**|**recno**] генерируют ошибки, если запрашиваемой записи нет в наличии.

### Примеры:

```
' Пример выборки в таблицу MapInfo
Dim hdbc, hstmt As Integer
hdbc = Server_Connect("QELIB", "DSN=ORACLE7;DLG=1")
hstmt = Server_Execute(hdbc, "Select * from emp")
Server hstmt Fetch Into "MyEmp"
Server hstmt Close

' Пример выборки с использованием связанных переменных
Dim hdbc, hstmt As Integer
dim NameVar, AddrVar as String
dim NameStatus, AddrStatus as Integer
hdbc = Server_Connect("QELIB", "DSN=ORACLE7;DLG=1")
hstmt = Server_Execute(hdbc, "Select Name, Addr from emp")
Server hstmt Bind Column 1 to NameVar, NameStatus
Server hstmt Bind Column 2 to AddrVar, AddrStatus
Server hstmt Fetch Next
While Not Server_Eot(hstmt)
    Print "Name = " + NameVar + "; Address = " + AddrVar
    Server hstmt Fetch Next
Wend
```

### Смотрите также:

**Server\_ColumnInfo()**

## Функция **Server\_GetodbcHConn( )**

### Назначение:

Возвращает целое значение, содержащее указатель связи ODBC, ассоциированной со связью с сервером данных.

### Синтаксис:

**Server\_GetodbcHConn** (*ConnectionNumber*)

*ConnectionNumber* – целое значение возвращаемое функцией **Server\_Connect**, которая определяет номер соединения с сервером данных.

### Описание:

Эта функция возвращает целое значение, содержащее указатель связи ODBC, ассоциированной со связью с сервером данных. Это позволяет Вам любую функцию в ODBC DLL, чтобы расширить функциональные возможности, посредством использования MapBasic операторов типа **Server**.

### Пример:

```
'* Нахождение связи с сервером данных
DECLARE FUNCTION SQLGetInfo LIB "ODBC32.DLL" (BYVAL odbchdbc AS INTE-
GER, BYVAL infoflag AS INTEGER, val AS STRING, BYVAL len AS INTEGER,
outlen AS INTEGER) AS INTEGER
Dim rc, outlen, hdbc, odbchdbc AS INTEGER
Dim DBName AS STRING
' Связь с сервером данных
hdbc      = Server_Connect("QELIB", "DLG=1")
odbchdbc  = Server_GetodbcHConn(hdbc) ' получение указателя связи ODBC
' Получение имени базы данных из ODBC
DBName    = STRING$(33, "0") ' Инициализация выходного буфера
rc = SQLGetInfo(odbchdbc, 17 , DBName, 40, outlen) ' получение имени
базы данных ODBC
' Отображение результатов (имя базы данных)
if rc <> 0 THEN
    Note "SQLGetInfo Error rc=" + rc + ", outlen=" + outlen
else
    Note "Connected to Database: " + DBName
end if
```

### Смотрите также:

**Server\_GetodbcHStmt()**

### Функция **Server\_GetodbcHStmt( )**

#### Назначение:

Возвращает указатель оператора ODBC, ассоциированный с MapBasic-оператором типа **Server**.

#### Синтаксис:

**Server\_GetodbcHStmt**(*StatementNumber*)

*StatementNumber* - целое значение, возвращаемое функцией **Server\_Execute()**, которое указывает на результирующий набор исполненного SQL-оператора.

#### Описание:

Эта функция возвращает указатель оператора ODBC, ассоциированный с MapBasic-оператором типа **Server**. Это позволяет Вам вызывать любую функцию ODBC, для расширения функциональных возможностей посредством использования MapBasic-операторов типа **Server**.

#### Пример:

```
' Нахождение номеров строк, которые будут обновляться
Dim rc, outlen, hdbc, hstmt, odbchstmt AS INTEGER
Dim RowsUpdated AS INTEGER
' Нахождение номеров строк, подлежащих обновлению
DECLARE FUNCTION SQLRowCount LIB "ODBC32.DLL" (BYVAL odbchstmt AS INTEGER, rowcnt AS INTEGER) AS INTEGER
hdbc = Server_Connect("QELIB", "DLG=1")
hstmt = Server_Execute(hdbc, "UPDATE TIML.CUSTOMER SET STATE='NY' WHERE STATE='NY'")
odbchstmt = Server_GetodbcHStmt(hstmt)
rc = SQLRowCount(odbchstmt, RowsUpdated)
Note "Updated " + RowsUpdated + " New customers to Tier 1"
```

#### Смотрите также:

**Server\_GetodbcHConn( )**

## Оператор Server Link Table

### Назначение:

Создает связанную таблицу. Здесь приводится синтаксис для версии 4.1, который позволяет создавать связанную таблицу, используя существующую связь с базой данных.

### Синтаксис1:

```
Server Link Table
  SQLQuery
Using ConnectionString
Into TableName
[ File FileSpec ]
[ ReadOnly ]
```

### Синтаксис2:

```
Server ConnectionNumber Link Table
  SQLQuery
Into TableName
[ File FileSpec ]
[ ReadOnly ]
```

*ConnectionNumber* – номер соединения с сервером данных.

*SQLQuery* – SQL-оператор запроса (на активном диалекте с добавлением объектных ключевых слов), который генерирует результирующий набор. Таблица MapInfo связывается именно с этим результирующим набором.

*ConnectionString* – строка, используемая для подключения к серверу базы данных (см. описание функции **Server Connect**).

*TableName* – псевдоним создаваемой таблицы MapInfo.

*FileSpec* – имя табличного файла. Если этот параметр отсутствует, имя файла генерируется в текущем каталоге диска на базе псевдонима таблицы. Если параметр *FileSpec* задан, а табличный файл с указанным именем уже существует, то генерируется ошибка.

**ReadOnly** – задает использование таблицы только для чтения.

### Описание:

Этот оператор создает связанную таблицу MapInfo на диске. Эта таблица открывается и к ней обращается запрос. Связанная таблица обрабатывается как обычная таблица MapInfo в большинстве случаев, кроме следующих:

- оператор **Alter Table** не выполняется для связанных таблиц;
- связанные таблицы не могут быть упакованы;
- в список диалога “Упаковка” эти таблицы не включаются.

Синтаксис **Server Link Table** используется для установки связи сервера базы данных и связанной таблицы. **Server ConnectionNumber Link Table** используются для связи таблицы с подключенным сервером базы данных.

Связанные таблицы содержат информацию для переустановки связи с удаленным сервером и идентификации удаленных данных, которые обновляются. Эта информация хранится в виде метаданных в TAB-файле.

## Оператор Server Link Table

---

Отсутствие ключевого слова **ReadOnly** не означает возможности редактирования таблицы. Связанная таблица может быть запрещена для записи при следующих обстоятельствах:

1) результирующий набор разрешен только для чтения; 2) результирующий набор не содержит первичного ключа; 3) в результирующем наборе отсутствуют редактируемые столбцы; 4) указан режим ReadOnly.

### Примеры:

```
Declare Sub Main
Sub Main
Open table "C:\mapinfo\data\states.tab"
Server Link Table "Select * from Statecap" Using "DSN=MS
Access;DBQ=C:\MSOFFICE\ACCESS\DB1.mdb" Into test File "C:\tmp\test"
Map From Test,States
End Sub 'Main

Declare Sub Main
Sub Main
Dim ConnNum As Integer
ConnNum = Server_Connect("qelib","DSN=SQS;PWD=sysmal;SRVR=seneca")
Server ConnNum Link Table
    "Select * from CITY_1"
    Into temp
Map From temp

Server ConnNum Disconnect
End Sub
```

### Смотрите также:

**Unlink, Drop Table, Save File, Commit Table, Rollback Table, Close Table, Server Refresh**

## Функция **Server\_NumCols( )**

### Назначение:

Возвращает число столбцов в результирующем наборе.

### Синтаксис:

**Server\_NumCols**(*StatementNumber*)

*StatementNumber* – целое значение, номер SQL-оператора.

### Возвращаемое значение:

Целое.

### Описание:

Функция **Server\_NumCols()** возвращает число столбцов в результирующем наборе, ссылка на который осуществляется по указателю *StatementNumber*.

### Пример:

```
Dim hdbc, hstmt As Integer
hdbc = Server_Connect("QELIB", "DSN=ORACLE7;DLG=1")
hstmt = Server_Execute(hdbc, "Select Name, Addr from emp")
Print "Number of columns = " + Server_NumCols(hstmt)
```

### Смотрите также:

**Server\_ColumnInfo()**

### Функция `Server_NumDrivers( )`

#### Назначение:

Возвращает число интерфейсных модулей удаленного доступа к базам данных, установленных в данный момент для доступа из MapInfo.

#### Синтаксис:

`Server_NumDrivers()`

#### Возвращаемое значение:

Целое.

#### Описание:

Функция `Server_NumDrivers()` возвращает число модулей интерфейса удаленного доступа, через которые может осуществляться соединение с удаленным сервером данных, установленных для использования приложением MapInfo.

#### Пример:

```
Print "Всего драйверов = " + Server_NumDrivers()
```

#### Смотрите также:

`Server_DriverInfo()`



## Оператор Server Refresh

### Назначение:

Осуществляет синхронизацию связанной таблицы MapInfo с данными в удаленной базе данных. Эта команда может выполняться только в отсутствие ожидающих запросов на редактирование связанной таблицы.

### Синтаксис:

**Server Refresh** *TableName*

*TableName* – имя открытой связанной таблицы MapInfo.

### Описание:

Синхронизация связанной таблицы включает следующие этапы:

1. Удаление всех записей и объектов из связанной таблицы (если она содержит записи) методом удаления и воссоздания табличного файла (не используя MapBasic-оператор **Delete**).
2. Если указатель (handle) соединения сохранен вместе со структурой TABLE, используется этот указатель. В противном случае соединение с сервером базы данных производится с использованием строки подключения, сохраненной в метаданных связанной таблицы.
3. Выполняется преобразование SQL-запроса, сохраненного в метаданных, в запрос, специфичный для удаленной СУБД.
4. Этот SQL-запрос исполняется на удаленном сервере базы данных.
5. Таблица заполняется строками, выбранными из курсора удаленной СУБД. Эта операция может сопровождаться индикатором выполнения MapInfo.
6. Курсор в удаленной СУБД закрывается.

### Пример:

```
Server Refresh "City_1k"
```

### Смотрите также:

**Commit Table, Server Link, Unlink**

### Оператор Server Rollback

#### Назначение:

Выполняет откат транзакции на удаленном сервере данных.

#### Синтаксис:

**Server** *ConnectionNumber* **Rollback**

*ConnectionNumber* – номер соединения с сервером данных.

#### Описание:

Оператор **Server Rollback** ликвидирует все изменения, внесенные в базу данных всеми SQL-операторами, выполненными в данном соединении с сервером с момента исполнения оператора **Server Begin Transaction**, и восстанавливает исходное состояние базы данных. Для выдачи этого оператора необходимо иметь открытую транзакцию, инициированную оператором **Server Begin Transaction**.

#### Пример:

```
hdbc = Server_Connect("QELIB", "DSN=ORACLE7")
Server hdbc Begin Transaction
...
' Все изменения, внесенные с момента исполнения Begin_Transaction,
' будут отменены (откат транзакции)
Server hdbc Rollback
```

#### Смотрите также:

**Server Begin Transaction**, **Server Commit**

## Оператор Server Set Map

### Назначение:

Этот оператор позволяет менять стили объектов для изображаемой в виде Карты таблицы ODBC.

### Синтаксис:

```
Server ConnectionNumber Set Map linked_table
[ ObjectType { Point | Line | Region } ]
[ Symbol (...) ]
[ Linestyle Pen(...) ]
[ Regionstyle Pen(...) Brush(...) ]
```

*ConnectionNumber* целое число, номер соединения;

*linked\_table* – имя открытой связанной таблицы ODBC;

*ObjectType* – указывает тип объектов в таблице;

*Symbol* (...) – указывает стиль используемый для точечного объекта;

*Linestyle Pen* (...) – указывает стиль линии, используемый для объекта типа линия;

*Regionstyle Pen* (...) *Brush*(...) – указывает стиль линии и заливки, используемых для объектов типа полигон.

### Описание:

Оператор **Server Set Map** изменяет стили объектов на Карте открытой таблицы ODBC. Таблица ODBC становится отображаемой в виде Карты с помощью оператора **Server Create Map**.

### Пример:

```
Declare Sub Main
Sub Main
  Dim ConnNum As Integer
  ConnNum = Server_Connect("qelib", "DSN=SQS;PWD=sys;SRVR=seneca")
  Server ConnNum Create Map "Cities"
    Type SQS "MI_SQS_POINT"
    CoordSys Earth Projection 1, 0
    ObjectType Point
    Symbol (35,0,12)
  Server ConnNum Create Map "States"
    Type SQS "MI_SQS_POLYGON"
    CoordSys Earth Projection 1, 0
    ObjectType Region
    RegionStyle Pen (1,2,255) Brush (2,16777215,16777215)
  Server ConnNum Disconnect
End Sub
```

### Смотрите также:

**Server Create Map**

## Функция SessionInfo ()

---

### Функция SessionInfo ()

#### Назначение

Возвращает различные блоки информации о сеансе работы MapInfo Professional.

#### Синтаксис

`SessionInfo( attribute )`

*attribute* целочисленный код, определяющий, атрибуты какого сеанса запрашиваются

#### Возвращаемое значение

Строка

#### Описание

Функция SessionInfo( ) возвращает информацию о состоянии сеанса работы MI Pro. Атрибут может принимать одно из значений, показанных в таблице. Коды определены в файле MAPBASIC.DEF.

attribute code	Возвращаемое значение
SESSION_INFO_COORDSYS_CLAUSE	Результирующая строка, определяющая предложение CoordSys текущего сеанса.
SESSION_INFO_DISTANCE_UNITS	Результирующая строка, определяющая единицы измерения расстояния текущего сеанса.
SESSION_INFO_AREA_UNITS	Результирующая строка, определяющая а единицы измерения площади текущего сеанса.
SESSION_INFO_PAPER_UNITS	Результирующая строка, определяющая "бумажные" единицы измерения текущего сеанса.

#### Возникновение ошибки

ERR\_FCN\_ARG\_RANGE ошибка генерируется если значение аргумента выходит за допустимые пределы

#### Пример

```
include "mapbasic.def"
print SessionInfo(SESSION_INFO_COORDSYS_CLAUSE)
```

## Оператор Set Application Window

### Назначение:

Устанавливает, какое окно будет порождающим для всех новых диалогов и окон.

### Предупреждение:

Этот оператор используется только в Microsoft Windows.

### Синтаксис:

**Set Application Window *HWND***

где

*HWND* – целое число типа Integer, уникальный системный номер окна.

### Описание:

Этот оператор объявляет, какое окно будет окном приложения. Для всех последующих окон диалогов MapInfo будет считаться, что они порождены этим другим окном. Этот прием используется в “Интегрированной Картографии”, когда окна MapInfo показываются из других приложений, написанных, например, на Visual Basic.

Обычно Ваша программа, написанная на Visual Basic, сначала создает объект MapInfo Object и затем посылает MapInfo оператор **Set Application Window**, после чего приложение на Visual Basic становится порождающим окном для диалогов MapInfo. Если оператор **Set Application Window** не был послан, то становится очень трудно координировать передачу фокуса между MapInfo и Visual Basic.

Этот оператор переподчиняет окно диалога. Для переподчинения документального окна, такого как окна Карты, используйте оператор **Set Next Document**.

**Замечание:** Если Вы задаете параметр *HWND* как шестнадцатеричное значение, то Вы должны использовать приставку **&H** с шестнадцатеричным числом. Иначе MapInfo попытается интерпретировать параметр как десятичное значение. (Это бывает, когда программа на Visual Basic создает командную строку, содержащую оператор **Set Application Window**.)

Для получения другой информации об интегрированной картографии смотрите 12 главу *Руководства пользователя MapBasic*.

### Смотрите также:

**Set Next Document**

## Оператор Set Area Units

### Назначение:

Устанавливает единицы измерения площади для использования в операторах и функциях MapBasic по умолчанию.

### Синтаксис:

**Set Area Units** *area\_name*

где

*area\_name* – строковое представление имени единицы измерения площади (например, "acre" – акр)

### Описание:

Оператор **Set Area Units** устанавливает единицы измерения площади. Установки единиц измерения площади используются в диалоге "SQL-запрос" в MapInfo. По умолчанию, MapBasic использует квадратные мили ("sq mi"), т. е. если в Вашей программе нет оператора **Set Area Units**, то единицами измерения площади будут квадратные мили.

Параметр *area\_name* должен иметь строковое значение, список которых приведен в таблице:

<u>Значение <i>area_name</i></u>	<u>Единицы измерения площади</u>
"acre"	акр
"hectare"	гектар
"sq cm"	квадратный сантиметр
"sq ft"	квадратный фут
"sq in"	квадратный дюйм
"sq km"	квадратный километр
"sq m"	квадратный метр
"sq mi"	квадратная миля
"sq mm"	квадратный миллиметр
"sq survey ft"	квадратный топографический фут в США
"sq yd"	квадратный ярд

В MapInfo также используются единицы измерения "perch", "rood", "rod", "chain" и "link", не применяемые в России.

### Пример:

```
Set Area Units "acre"
```

### Смотрите также:

**Area( ), Set Distance Units**

## Оператор Set Browse

### Назначение:

Изменяет представление открытого окна Списка.

### Синтаксис:

```
Set Browse
  [ Window window_id ]
  [ Grid { On | Off } ]
  [ Row row_num ]
  [ Column column_num ]
```

где

*window\_id* – идентификатор окна Списка, целое число типа Integer;

*row\_num* – целое число типа SmallInt от одного и более, где 1 представляет первую строку таблицы;

*column\_num* – целое число типа SmallInt от нуля и более, где 0 представляет первую колонку таблицы.

### Описание:

Оператор **Set Browse** управляет представлением окна Списка. Если параметр *window\_id* не задан, то действия оператора распространяются на самое верхнее из открытых окон Списка.

Предложения **Row** и **Column** позволяют назначить строку, которая будет видна первой, и колонку, которая будет самой левой в окне Списка.

**Grid** включает или выключает сетку (соответственно **On** и **Off**).

Для того, чтобы изменить высоту, ширину и местоположение окна, используйте оператор **Set Window**.

### Пример:

```
Dim i-browser-id As Integer
Open Table "world"
Browse * From world
i-browser-id = FrontWindow( )
Set Browse Window i-browser-id Row 47
```

### Смотрите также:

**Browse, Set Window**

### Оператор Set Cartographic Legend

#### Назначение:

Оператор **Set Cartographic Legend** позволяет Вам включать или выключать режим перерисовки, обновления, устанавливать книжную или альбомную ориентацию легенды, или изменять порядок следования разделов в картографической легенде, созданной оператором **Create Cartographic Legend**. (Для изменения размера, позиции или заголовка окна легенды используйте оператор Set Window.)

#### Синтаксис:

```
Set Cartographic Legend  
[ Window legend_window_id ]  
Redraw { On | Off }
```

или

```
Set Cartographic Legend  
[ Window legend_window_id ]  
[ Refresh ]  
[ Portrait | Landscape ]  
[ Frame Order { frame_id, frame_id, frame_id, ... } ]
```

*legend\_window\_id* - это целое, идентификатор окна, который можно получить вызовом функций **FrontWindow()** и **WindowId()**.

*frame\_id* - это индекс *ID* раздела легенды. Вы не можете использовать имя слоя. Например, три раздела легенды вполне могут иметь индексы *ID*: 1, 2 и 3.

#### Описание:

Если предложение **Window** не описано, то MapInfo будет использовать самое верхнее окно легенды.

Нельзя использовать другие предложения, если используется **Redraw**.

Ключевое слово **Refresh** приводит к обновлению окна легенды. Таблицы, для которых будут обновляться разделы легенды, просканируются на предмет используемых в них стилей.

Ключевые слова **Portrait** или **Landscape** определяют книжная или альбомная ориентация окна легенды будет использована.

Предложение **Frame Order** изменяет порядок следования разделов в легенде.

#### Смотрите также:

**Create Cartographic Legend, Alter Cartographic Frame, Add Cartographic Frame, Remove Cartographic Frame**



## Оператор Set Command Info

### Назначение:

Помещает в память значения, которые могут быть прочитаны функцией **CommandInfo( )** из другой процедуры.

### Синтаксис:

**Set Command Info** *attribute To new\_value*

где

*attribute* – один из кодов, используемых функцией **CommandInfo( )** (такой как CMD\_INFO\_ROWID);

*new\_value* – новая величина, тип которой должен соответствовать значению кода *attribute* (например, если используется код CMD\_INFO\_ROWID, то параметр *new\_value* должен быть положительным целым числом типа Integer).

### Описание:

Обычно функция **CommandInfo( )** возвращает значения, описывающие системные события. Оператор **Set Command Info** помещает свои значения в память, так что следующий за этим вызов функции **CommandInfo( )** возвращает заданные значения, вместо тех, которые несут информацию о системных событиях.

### Пример:

Допустим, что Ваша программа имеет процедуру-обработчик **SelChangedHandler**. Из ее тела следующая функция определяет идентификатор строки, которая была выбрана или исключена из выбора:

```
CommandInfo(CMD_INFO_ROWID)
```

Когда MapInfo вызывает **SelChangedHandler** автоматически, то устанавливаются данные для чтения функцией **CommandInfo( )**. Теперь допустим, что для отладки Вы хотите вызвать процедуру **SelChangedHandler** оператором **Call**. Перед оператором **Call** должен быть оператор, посылающий значение для функции:

```
Set Command Info CMD_INFO_ROWID To 1
```

### Смотрите также:

**CommandInfo( ), Set Handler**

## Оператор Set CoordSys

### Назначение:

Назначает координатную систему, в дальнейшем используемую прикладной программой MapBasic.

### Синтаксис:

**Set CoordSys...**

где

**CoordSys** – слово, с которого начинается стандартное предложение оператора для определения координатной системы.

### Описание:

Оператор **Set CoordSys** устанавливает координатную систему, которая в дальнейшем будет использоваться прикладной программой MapBasic. По умолчанию, MapBasic использует долготу и широту. Данная установка влияет на возвращаемые значения географическими функциями, такими как **CentroidX( )** и **ObjectNodeX( )**.

MapBasic-программа может выполнить собственный оператор **Set CoordSys**, вследствие чего значения, возвращаемые географическими функциями, будут автоматически отражать новую координатную систему. При этом координатная система в MapInfo может оставаться прежней.

Оператор **Set CoordSys** может использовать подпредложения **Table** или **Window** для установки координатной системы, такой же, как в таблице, или такой же, как в окне.

Смотрите описание предложения **CoordSys** для более полной информации.

### Пример:

Установим непроецированную координатную систему Земли:

```
Set CoordSys Earth
```

Следующий оператор **Set CoordSys** устанавливает в качестве координатной системы равноплощадную проекцию Алберса.

```
Set CoordSys Earth  
Projection 9,7,"m",-96.0,23.0,20.0, 60.0, 0.0, 0.0
```

В окне Отчета координатная система определяется оператором **Set CoordSys**. Перед работой с объектами Отчета Вы должны определить координатную систему на странице.

Следующий оператор готовит MapBasic к работе с объектами в окнах Отчета. Вы должны использовать координатную систему Отчета перед тем, как создавать и обрабатывать объекты Отчета.

```
Set CoordSys Layout Units "in"
```

Выполнив оператор **Set CoordSys Layout**, MapBasic продолжает использовать координатную систему Отчета до тех пор, пока Вы не переопределите ее. Аналогично, если Вы хотите работать с объектами на проецированной Карте, выполните оператор **Set CoordSys Earth**.

### Смотрите также:

**CoordSys, Set Area Units, Set Distance Units, Set Paper Units**

## Оператор Set Date Window

### Назначение

Отображает окно даты, которое переводит двузначный год в четырехзначный. В нем можно поменять стандартную настройку.

### Синтаксис

**Set Date Window** { *nYear* | Off }

*nYear* - это короткое целое, от 0 до 99, которое определяет число, значения превышающие это число будут отнесены к 20-му столетию, а числа, меньшие заданного, будут отнесены к 21-му столетию.

*Off* - отключает окно даты. Двузначные годы будут конвертироваться в текущее столетие (по системному календарю и времени).

### Описание

Запуск команды **Set Date Window** из окна MapBasic изменит поведение дат, но не изменит системные настройки даты в MIPro.

При запуске команды **Set Date Window** из программы MapBasic, поведение дат изменится только локально, настройки дат в системе не изменятся.

**Внимание:** Все атрибуты переменных, связанных с датами в версиях, ранее MapBasic 5.5 являются действующими. Здесь берутся двузначные значения года и используется окно даты для определения того, какое столетие использовать. Эти переменные: Year(date\_expr), StringToDate(datestring), FormatDate\$(date\_expr)

MBX, скомпилированные до версии 5.5 будут конвертировать двузначный год в текущее столетие (поведение версии 5.0 и более ранних). Что бы получить новое поведение отнесения двузначных данных к разным столетиям, перекомпилируйте программу в MapBasic v5.5.

### Пример

В следующем примере переменные Date1 = 19890120, Date2 = 20101203 и MyYear = 1990.

```
DIM Date1, Date2 as Date
DIM MyYear As Integer
Set Format Date "US"
Set Date Window 75
Date1 = StringToDate("1/20/89")
Date2 = StringToDate("12/3/10")
MyYear = Year("12/30/90")
```

### Смотрите так же

Функция Datewindow( )

## Оператор Set Digitizer

### Назначение:

Назначает координаты для оцифровки изображения с бумажной карты. А также включает и выключает режим дигитайзера.

### Синтаксис (вариант 1):

```
Set Digitizer
(mapx1, mapy1) (tabletx1, tablety1) [ Label name ],
(mapx2, mapy2) (tabletx2, tablety2) [ Label name ]
[, ... ]
CoordSys...
[   Units... ]
[   Width tabletwidth ]
[   Height tabletheight ]
[   Resolution xresolution, yresolution ]
[   Button click button_num, double_click button_num ]
[ Mode { On | Off } ]
```

### Синтаксис (вариант 2):

```
Set Digitizer Mode { On | Off }
```

где

*mapx*# – расположение относительно Запада и Востока на бумажной карте;

*mapy*# – расположение относительно Севера и Юга на бумажной карте;

*tablet<sub>x</sub>*# – X-координата на планшете, соответствующая *mapx*#;

*tablety*# – Y-координата на планшете, соответствующая *mapy*#;

*names* – имя контрольной точки;

*click\_button\_num* – номер кнопки, которая симулирует один щелчок мыши;

*double\_click\_button\_num* – номер кнопки, которая симулирует двойной щелчок мыши.

Слово **CoordSys** начинает стандартное предложение оператора для определения координатной системы на бумажной карте.

### Описание:

Оператор **Set Digitizer** используется для настройки планшета дигитайзера.

Параметры оператора **Set Digitizer** соответствуют режимам и данным, которые пользователь MapInfo может задать при помощи диалогового окна команды КАРТА > НАСТРОЙКА ДИГИТАЙЗЕРА. Все измерения проводятся в заданных пользователем единицах измерения бумажной карты. Оператор **Set Digitizer** не настраивает другие (системные) режимы работы с дигитайзером, такие как порты и скорость связи. Их пользователь должен настроить вне MapBasic и MapInfo.

Программа MapInfo понимает дигитайзер как устройство ввода, которое позволяет пользователю переводить данные с бумажной карты в таблицу MapInfo. Использование дигитайзера предполагает, что перед оцифровкой данных изображения пользователь должен задать, какие точки на карте соответствуют контрольным точкам на плоскости планшета.

Оператор **Set Digitizer** позволяет задать MapInfo координатную систему, используемую на бумажной карте, а также две или более контрольных точек. Каждая контрольная точка определяет соответствие координатной пары карты (долгота и широта) координатной паре на планшете. Координаты на планшете представляются в единицах измерения дигитайзера, измеряемых от левого верхнего угла планшета.

Предложение **CoordSys** задает координатную систему для бумажной карты. Оператор **Set Digitizer** игнорирует установку **Bounds** в предложении **CoordSys**.

MapInfo может сохранять настройку дигитайзера для следующих сеансов работы в файле Рабочего Набора в виде оператора **Set Digitizer**. Пользователь может поменять планшет, и это может привести к тому, что установка дигитайзера может оказаться неподходящей для нового планшета. Предложения **Width**, **Height** и **Resolution** в операторе **Set Digitizer** помогают MapInfo определить смену дигитайзера.

### Режим оцифровки

Если дигитайзер настроен, пользователь может включать и выключать режим оцифровки, нажимая на клавишу “D”. Этот режим может включать и программа MapBasic оператором

```
Set Digitizer Mode On
```

или выключать:

```
Set Digitizer Mode Off
```

Для определения включения режима оцифровки используется функция **SystemInfo(SYS\_INFO\_DIG\_MODE)**, которая возвращает “Да” (TRUE), если режим установлен.

При включенном режиме оцифровки в активном окне Карты помимо указателя мышки появляется также курсор дигитайзера в виде большого креста.

Если отключен режим оцифровки или окно Карты неактивно, курсор дигитайзера не показывается и панель дигитайзера начинает работать как мышь (если только Ваш дигитайзер поддерживает режим эмуляции мыши).

### Смотрите также:

**CoordSys, SystemInfo( )**

## Оператор Set Distance Units

### Назначение:

Устанавливает единицы измерения расстояний, используемые в (гео)графических операциях.

### Синтаксис:

**Set Distance Units** *unit\_name*

где

*unit\_name* – имя единицы линейных измерений (например, "m" для метров).

### Описание:

Оператор **Set Distance Units** устанавливает единицы линейных измерений. По умолчанию MapBasic использует мили ("mi"), то есть, если в Вашей программе нет оператора **Set Distance Units**, единицами измерения расстояния будут мили.

В следующих операторах и функциях будет использоваться установленная единица измерения расстояний, если единицы измерения специально не определяются в самих операторах или функциях. Например, в предложении **Width** оператора **Create Object** Вы можете задавать или не задавать единицы измерения ширины объекта. Если не зададите, оператор **Create Object** использует текущую установку единиц измерения расстояний.

Параметр *unit\_name* должен иметь строковые значения, список которых приведен в таблице:

Значение <i>unit_name</i>	Единицы измерения расстояний
"cm"	сантиметр;
"ft"	фут (также называется международным футом; один международный фут примерно равен 30.48 сантиметрам);
"in"	дюйм;
"km"	километр;
"m"	метр;
"mi"	миля;
"mm"	миллиметр;
"nmi"	морские мили (1 морская миля равна 1852 метрам);
"survey ft"	топографический фут в США (использовался при обмере территории США в 1927; один топографический фут примерно равен 30.48006 сантиметрам);
"yd"	ярд

В MapInfo также используются единицы измерения "perch", "rood", "rod", "chain" и "link", не применяемые в России.

### Пример:

```
Set Distance Units "km"
```

### Смотрите также:

**Distance( )**, **ObjectLen( )**, **Set Area Units**, **Set Paper Units**

### Оператор Set Drag Threshold

#### Назначение:

Назначает временную задержку, нужную для фиксации мыши на перемещаемом объекте.

#### Синтаксис:

**Set Drag Threshold** *pause*

где

*pause* – вещественное число, задающее задержку в секундах (по умолчанию – 1.0).

#### Описание:

Когда пользователь указывает на объект, оставляя клавишу мышки нажатой, MapInfo требует выдержать небольшую паузу. Эта задержка предотвращает нечаянное перемещение объекта.

Оператор **Set Drag Threshold** назначает продолжительность этой задержки.

#### Пример:

```
Set Drag Threshold 0.25
```

### Оператор Set Event Processing

#### Назначение:

Позволяет временно отключить реакцию на системные события и избежать лишних перерисовок экрана.

#### Синтаксис:

**Set Event Processing { On | Off }**

#### Описание:

Оператор **Set Event Processing** позволяет временно отключать реакцию на системные события и тем самым избежать ненужной перерисовки содержимого экрана. Тот же оператор позволяет затем снова включить реакцию на системные события.

Несколько последовательных команд могут изменять окно, что сопровождается перерисовкой его содержимого. Чтобы сэкономить время, пользователь может отключить перерисовку окон оператором

```
Set Event Processing Off,
```

а после того, как все операторы, изменяющие окно (например, **Set Map**), отработают, включить перерисовку снова:

```
Set Event Processing On.
```

Каждый оператор **Set Event Processing Off** должен иметь парный **Set Event Processing On**.

Если, работая в многозадачных системах (например, Windows или System 7), Вы забудете включить обработку событий обратно, это может повлиять на работу других программ.

Вы также можете управлять перерисовкой изображения в окне оператором **Set Map... Redraw Off**, действие которого похоже на действие оператора **Set Event Processing Off**. Однако оператор **Set Map... Redraw** управляет перерисовкой одного окна Карты, а действие оператора **Set Event Processing** распространяется на все окна MapInfo.



## Оператор Set File Timeout

### Назначение:

Предписывает MapInfo повторять попытку доступа к файлу после сетевого конфликта.

### Синтаксис:

**Set File Timeout  $n$**

где

$n$  — число от нуля и больше, задающее ожидание в секундах.

### Описание:

Обычно, если операция не может быть продолжена из-за конфликта в сети, MapInfo показывает диалог типа “Повторить/Отменить”. Если программа MapBasic выполнит оператор **Set File Timeout**, то MapInfo вместо вывода диалога будет автоматически повторять попытки открыть файл, доступ к которому в сети запрещен. Это может понадобиться тогда, когда несколько пользователей работают в сети с одной таблицей.

Если число  $n$  больше нуля, то через каждые  $n$  секунд MapInfo делает очередную попытку открыть файл. Если таблица все еще не доступна, MapInfo показывает диалог с соответствующим сообщением. Этот диалог нельзя перехватить и обработать средствами обработки ошибок MapBasic.

Если  $n=0$  MapInfo демонстрирует диалог немедленно, как только обнаруживает, что таблица недоступна.

**Внимание:** Не используйте одновременно оператор **Set File Timeout** и обработчик ошибок **OnError**. Там, где действует обработчик ошибок, значение ожидания должно быть равно нулю. Там, где ожидание не равно нулю, нужно отключать обработчик ошибок.

Более подробно возможные конфликты в сети описаны в 7 главе *Руководства пользователя MapBasic*.

### Пример:

```
Set File Timeout 100
```

## Оператор Set Format

### Назначение:

Задаёт, как MapBasic составляет строки из численных значений и значений даты и времени.

### Синтаксис (вариант 1):

**Set Format Date** { "US" | "Local" }

### Синтаксис (вариант 2):

**Set Format Number** { "9,999.9" | "Local" }

### Описание:

Пользователь может установить разные форматы для даты и чисел в своём компьютере. Например, Windows-пользователь может изменить формат, используя "Международные" ("International") установки в среде Windows версий 3.x или "Язык и стандарты" ("Regional") в среде Windows 95 (в программе Control Panel).

Некоторые функции MapBasic, такие как **Str\$( )**, используют эти системные установки. Другими словами, некоторые функции могут иметь разные результаты, так как выполнялись в компьютерах с разными системными конфигурациями.

Оператор **Set Format** заставляет программу MapBasic игнорировать внешние установки так, чтобы функции (такие как **Str\$( )**) вела себя предсказуемо.

### Оператор

Set Format Date "US"

Set Format Date "Local"

Set Format Number "9,999.9"

Set Format Number "Local"

### Эффект в программе MapBasic

MapBasic использует форму Месяц/День/Год для представления даты вместо внешней установки компьютера пользователя.

MapBasic использует форму для представления даты согласно установке компьютера пользователя.

Функция **Format\$( )** использует установку для форматирования числа, принятую в США (десятичной точкой является точка, а разделителем тысяч – запятая) вместо внешней установки компьютера пользователя.

Функция **Format\$( )** использует форму для представления чисел согласно установке компьютера пользователя.

Первый вариант синтаксиса (**Set Format Date**) имеет эффект для следующих случаев:

вызов функции **StringToDate( )**;

использование даты в функции **Str\$( )**;

выполнение операций в MapBasic, ведущих к автоматическому конвертированию строковых значений и значений дат (например, применение оператора **Print** для печати даты или при присваивании переменной строкового типа величины типа Date).

Второй вариант синтаксиса (**Set Format Number**) влияет на функции **Format\$( )** и **FormatNumber\$( )**.

Программы, откомпилированные MapBasic версии 3.0 и более ранней, по умолчанию используют стандарт США. Программы, откомпилированные в MapBasic версии 4.0 и в более поздних версиях по умолчанию используют установку “Local”.

Для определения, какой режим форматирования установлен, используется функция **SystemInfo( )**. Установка оператора **Set Format**, сделанная в одной программе MapBasic, не влияют на другие выполняющие программы.

### Пример:

Пусть переменная типа даты date\_var содержит “June 11, 1995”. Функция:

```
Str$( date_var )
```

будет иметь результат “06/11/95” или “95/11/06”, в зависимости от того, какой формат для дат установлен на Вашем компьютере.

Если Вы вставите в текст Вашей программы оператор **Set Format Date “US”** перед местом, где используется функция **Str\$( )**, то результатом будет “11/06/95”.

### Смотрите также:

**Format\$( ), FormatNumber\$( ), Str\$( ), StringToDate( ), SystemInfo( )**

## Оператор Set Graph

### Назначение:

Изменяет настройки отображения данных в окне Графика.

### Синтаксис 1 (Графики 5.5)

```
Set Graph
[ Window window_id ]
[ Title title_text ]
[ SubTitle subtitle_text ]
[ Footnote footnote_text ]
[ TitleSeries titleseries_text ]
[ TitleGroup titlegroup_text ]
[ TitleAxisY1 titleaxisy1_text ]
[ TitleAxisY2 titleaxisy2_text ]
```

где

*window\_id* – идентификатор окна Графика;

*title\_text* - заголовок, появляющийся в верху окна Графика

*subtitle\_text* - текст подзаголовка графика.

*footnote* - текст сноски графика.

*titleseries\_text* - текст заголовка серий графика.

*titlegroup\_text* - текст заголовка групп графика.

*titleaxisY1\_text* - текст заголовка оси Y.

*titleaxisY2* - текст заголовка для оси Y2.

### Синтаксис (Графики версий до 5.5)

```
Set Graph
[ Window window_id ]
[ Type { Area | Bar | Line | Pie | XY } ]
[ Stacked { On | Off } ]
[ Overlapped { On | Off } ]
[ Droplines { On | Off } ]
[ Rotated { On | Off } ]
[ Show3d { On | Off } ]
[ Overlap overlap_percent ]
[ Gutter gutter_percent ]
[ Angle angle ]
[ Title graph_title [ Font . . . ] ]
[ Series series_num
  [ Pen . . . ]
  [ Brush . . . ]
  [ Line . . . ]
  [ Symbol . . . ]
  [ Title series_title ] ]
[ Wedge wedge_num
  [ Pen . . . ]
  [ Brush . . . ] ] ]
[ { Label | Value } Axis
```

```

[ { Major | Minor } Tick { Cross | Inside | None | Outside } ]
[ { Major | Minor } Grid { On | Off } Pen ... ]
[ Labels { None | At Axis } [ Font ... ] ]
[ Min { min_value | Auto } ]
[ Max { max_value | Auto } ]
[ Cross { cross_value | Auto } ]
[ { Major | Minor } Unit { unit_value | Auto } ]
[ Pen ... ]
[ Title axis_title [ Font ... ] ]
[ Legend
  [ Title legend_title [ Font ... ] ]
  [ Subtitle legend_subtitle [ Font ... ] ]
  [ Range [ Font ... ] ]
]

```

*window\_id* – идентификатор окна Графика;

*overlap\_percent* – целое число от 0 до 100, задающее процент перекрывания двух соседних столбцов;

*gutter\_percent* – целое число от 0 до 100, задающее расстояние между столбцами в процентах;

*angle* – целое число от 0 до 360, задающее стартовый угол для круговой диаграммы;

*graph\_title* – строка с текстом заголовка графика;

*axis\_title* – строка с текстом заголовка одной из осей графика;

*min\_value* – минимальная величина, показанная на оси графика;

*max\_value* – максимальная величина, показанная на оси графика;

*cross\_value* – точка пересечения осей;

*unit\_value* – единица измерения для делений на одной из осей;

*series\_num* – номер серии данных в графике, которая подвергается изменениям (например, 2, 3, ...);

*series\_title* – имя серии для отображения его в легенде с образцами линии и штриха;

*legend\_title* и *legend\_subtitle* – строки с заголовком и подзаголовком легенды графика.

Предложение **Line** определяет стиль линии для линейного графика.

Предложение **Brush** определяет стиль штриховки.

Предложение **Pen** определяет стиль линии границы заштрихованной области.

Предложение **Symbol** определяет стиль символа.

Предложение **Font** определяет стиль шрифта для отображения текстов.

## Описание:

Оператор **Set Graph** изменяет вид графика в уже открытом окне График. Если идентификатор окна не указан в операторе (параметр *window\_id*), то оператор будет работать с окном Графика, которое располагается выше остальных открытых окон Графиков. Этот оператор позволяет программе управлять графиком и легендой так же, как это может делать пользователь при помощи команд из меню ГРАФИК в окне MapInfo.

Оператор **Set Graph** может использоваться в файле Рабочего Набора. Для примера Вы можете открыть окно Графика и сохранить Рабочий Набор (например, под именем GRAPHER.WOR). Теперь откройте файл Рабочего Набора в любом текстовом редакторе и увидите оператор **Set Graph**, задающий те настройки, которые были у открытого ранее окна Графика.

## Оператор Set Graph

---

Для изменения размеров окна Графика и расположения его на экране используйте оператор **Set Window**.

Команды Graph в рабочих наборах или программах, созданных в версиях ранее 5.5, будут генерировать окна графиков версии 5.0. Когда окно графика версии 5.0 активно в сеансе Map-Info 5.5, то появится меню графика 5.0, так что можно его редактировать в диалогах версии 5.0. Мастер графиков всегда генерирует окно графика версии 5.5.

### Пример:

график версии 5.5 и выше

```
include 'mapbasic.def'
graph_id = WindowId(4) ' window code for a graph is 4
Set Graph
Window graph_id
Title "United States"
SubTitle "1990 Population"
Footnote "Values from 1990 Census"
TitleGroup "States"
TitleAxisY1 "Population"
```

(график версии ранее 5.5)

Этот пример иллюстрирует использование оператора **Set Graph**, а также настройки элементов окна Легенды. Следующая за приведенными ниже операторами настроек команда может открыть окно Графика для данных из двух колонок (*orders\_rcvd* и *orders\_shipped*) из таблицы SELECTION (окно может быть открыто оператором **Graph**). Оператор **Graph** фактически определяет три колонки, первая из которых ("sales\_rep") используется для образования надписей у оси.

```
Open Window Legend
Set Window Legend
    Position (3.0, 1.6) Width 3.3 Height 0.750000

Graph sales_rep,orders_rcvd,orders_shipped
    From selection
    Position (0.2, 0.1) Width 4.5 Height 3.9
    '
    ' Первый оператор Set Graph задает тип
    ' графика и главный заголовок графика
    '
Set Graph
    Type Bar Stacked Off Overlapped Off
    Droplines Off Rotated Off Show3d Off
    Overlap 30 Gutter 10 Angle 0
    Title "График выполнения заказов"
    Font ("Helv",1,18,0)
    '
    ' Второй Set Graph задает все атрибуты
    ' оси X .
    '
Set Graph Label Axis
```

```

Major Tick Outside
Major Grid Off Pen (1,2,117440512)
Minor Tick None
Minor Grid Off Pen (1,2,117440512)
Min 1.0 Max 5.0
Cross 1.0 Major unit 1.0 Minor unit 0.5
Labels At Axis Font ("Helv",0,8,0)
Pen (1,2,117440512)
Title "Торговый представитель" Font ("Helv",0,8,0)
'
' надпись "Торговый представитель"
' появляется у оси X
'
'
' Следующий оператор Set Graph задает все атрибуты
' оси Y (оси значений).
'
Set Graph Value Axis
Major Tick Outside
Major Grid Off Pen (1,2,117440512)
Minor Tick None
Minor Grid Off Pen (1,2,117440512)
Min 0.0 Max 300000.0
Cross 0.0 Major unit 50000.0 minor unit 25000.0
Labels At Axis Font ("Helv",0,8,0)
Pen (1,2,117440512)
Title "Сумма заказов" Font ("Helv",0,8,0)
'
' надпись "Сумма заказов"
' появляется у оси Y
'
'
' Далее настраивается стиль оформления
' для второй серии данных. Данные из колонки
' orders-rcvd будут представлены цветными полосками.
' Кроме того, настраивается легенда.
'
' Так как график задан в виде полосок, нужно настроить
' стиль штриховки (Brush). Если бы график был линейным,
' то надо было бы настраивать стили линии и символа
' (Line и Symbol).
'
Set Graph Series 2
Brush (8,255,16777215)
Line (1,2,0,255) Symbol (32,255,12)

```

## Оператор Set Graph

---

```
Title "Принятые заказы"
'
'это заголовок легенды.
'
'
' Далее настраивается стиль оформления
' для третьей серии данных (orders-shipped).
'

Set Graph Series 3
Brush (2,12632256,201326591)
Line (1,2,0,0) Symbol (34,12632256,12)
Title "Выполненные заказы"
'
' и это заголовок легенды
'
'
' Последняя настройка определяет заголовок, подзаголовок
' и шрифты легенды.
'

Set Graph Legend
Title "Графику выполнения заказов"
Font ("Helv",0,10,0)      'шрифт для заголовка
Subtitle "торговыми представителями"
Font ("Helv",0,8,0)      'шрифт для подзаголовка
Range font ("Helv",2,8,0) 'шрифт для элементов легенды
```

**Смотрите также:**

**Graph, Set Window**



## Оператор Set Handler

### Назначение:

Включает и выключает вызов обработчика системных событий, вызов такой процедуры как **SelChangedHandler**.

### Предупреждение:

Вы не можете использовать оператор в окне MapBasic.

### Синтаксис:

**Set Handler** *handler\_name* { On | Off }

где

*handler\_name* – имя процедуры обработчика, такой как **SelChangedHandler**.

### Описание:

Обычно, если Ваша программа имеет в своем теле процедуры-обработчики, MapInfo вызывает их автоматически, как только происходит соответствующее событие. Например, если Ваша программа имеет процедуру **SelChangedHandler**, MapInfo вызовет ее автоматически, как только произойдет изменение в выборе.

Оператор **Set Handler** может отключить автоматический вызов определенного обработчика в Вашей программе MapBasic и включить его снова.

Оператор **Set Handler ... Off** имеет эффект только для автоматических вызовов, на вызовы процедуры оператором **Call** он не влияет.

### Пример:

Следующий пример показывает, как обойти лишние запуски обработчика изменения выбора при помощи оператора **Set Handler**.

```
Sub SelChangedHandler
  Set Handler SelChangedHandler Off

  ' На этом месте можно применить в цикле оператор Select
  ' не обрабатывая при этом изменения выбора.

  Set Handler SelChangedHandler On
End Sub
```

### Оператор Set Layout

#### Назначение:

Меняет уже открытое окно Отчета.

#### Синтаксис:

##### Set Layout

```
[ Window window_id ]  
[ Center (center_x, center_y) ]  
[ Extents { To Fit | (pages_across, pages_down) } ]  
[ Pagebreaks { On | Off } ]  
[ Frame Contents { Active | On | Off } ]  
[ Ruler { On | Off } ]  
[ Zoom { To Fit | zoom_percent } ]
```

где

*window\_id* – идентификатор окна Отчета;

*center\_x* – горизонтальная координата центра окна Отчета;

*center\_y* – вертикальная координата центра окна Отчета;

*pages\_across* – число страниц (одна или более), расположенных подряд по горизонтали;

*pages\_down* – число страниц (одна или более), расположенных подряд по вертикали;

*zoom\_percent* – процентное соотношение размера отображения Отчета к реальному размеру.

#### Описание:

Оператор **Set Layout** управляет настройкой отображения содержимого окна Отчета. Если идентификатор окна не указан в операторе (параметр *window\_id*), то оператор будет работать с окном Отчета, которое располагается выше остальных открытых окон Отчетов. Этот оператор позволяет программе управлять отображением данных в окне Отчета так же, как это делает пользователь при помощи команд из меню ОТЧЕТ в окне MapInfo.

Предложение **Center** задает точку, которая будет центральной в окне Отчета.

Предложение **Extents** управляет количеством страниц Отчета. Следующий вариант оператора:

```
Set Layout Extents To Fit
```

определяет, на скольких страницах установленного формата для принтера, который на данный момент подключен к системе, смогут поместиться все существующие объекты Отчета. После слова **Extents** можно также назначать и определенное количество печатных страниц. Например, оператор:

```
Set Layout Extents (3, 2)
```

задает шесть страниц, по три в два ряда.

Если Отчет состоит более чем из одной страницы, то переключатель **Pagebreaks** позволяет Вам управлять отображением линий, разделяющих изображение на страницы. По умолчанию используется режим **On**.

Предложение **Frame Contents** управляет режимом обновления содержимого рамок. Отчет может содержать от одного или более таких объектов, отображающих изображения из других открытых окон в MapInfo (Карт, Графиков, Легенд и т. п.). Предложение **Frame Contents** задает режим перерисовки изображения в Отчете при изменении данных в окне, на базе которого был создан объект типа "рамка". Следующий оператор задает программе MapInfo синхронную перерисовку в окне Отчета:

```
Set Layout Frame Contents On
```

Перерисовка может происходить только тогда, когда окно Отчета становится активным. Для этого используется следующий режим:

```
Set Layout Frame Contents Active
```

Для того, чтобы программа MapInfo вообще не обновляла изображение в окне Отчета, используйте следующий вариант:

```
Set Layout Frame Contents Off
```

В последнем случае все объекты типа "рамка" показываются в окне Отчета в виде закрашенных прямоугольников с именами соответствующих окон (например, "РОССИЯ Карта").

Предложение **Ruler** включает режим отображения линеек сверху и слева в окне Отчета. По умолчанию используется режим **On**.

Предложение **Zoom** определяет отношение реальных размеров отчета и представленного на экране макета. Например, следующее предложение задает пятидесятипроцентный масштаб макета:

```
Set Layout Zoom 50.0
```

Другими словами, размеры изображения на экране будут ровно в два раза меньше реальных размеров Отчета. Параметр *zoom\_percent* может принимать значения от 6.25% до 800% включительно.

В предложении **Zoom** можно не определять масштаб, а потребовать, чтобы все страницы поместились на экране:

```
Set Layout Zoom To Fit
```

Если в окне выбран объект "рамка", то, используя операторы **Run Menu Command** (для имитации команд ДОСТАТЬ НАВЕРХ или ПОДЛОЖИТЬ ВНИЗ) и **Alter Object**, Вы можете изменить стили линий и штриховок, порядок наложения объектов друг на друга.

Для изменения таких параметров окна Отчета, как ширина окна, высота и расположение его на экране, используйте оператор **Set Window**.

### Пример:

```
Set Layout
Zoom To Fit Extents To Fit
Ruler Off
Frame Contents On
```

### Смотрите также:

**Alter Object, Create Frame, Layout, Run Menu Command, Set Window**

### Оператор Set Legend

#### Назначение:

Изменяет настройки отображения информации в окне "Легенда".

#### Синтаксис:

```
Set Legend
[ Window window_id ]
[ Layer layer_id | layer_name | Prev } ]
  [ Display { On | Off } ]
  [ Shades { On | Off } ]
  [ Symbols { On | Off } ]
  [ Lines { On | Off } ]
  [ Count { On | Off } ]
  [ Title layer_title [ Font... ] ]
  [ SubTitle layer_subtitle [ Font... ] ]
  [ Ascending { On | Off } ]
  [ Ranges [Font... ]
    range_title [ Display { On | Off } ]
    [, ... ]
  ]
[ , ... ]
```

где

*window\_id* – идентификатор окна Карты;

*layer\_id* – целое число или строка, обозначающие слой Карты;

*layer\_name* – строка, задающая слой Карты;

*layer\_title* и *layer\_subtitle* – заголовок и подзаголовок легенды;

*range\_title* – текстовая строка, описывающая один диапазон при условном выделении.

Слово **Font** начинает стандартное предложение оператора для определения стиля шрифта.

#### Описание:

Оператор **Set Legend** задает настройки для окна "Легенда". Этот оператор позволяет программе управлять отображением условных обозначений. Для изменения таких параметров окна "Легенда", как ширина окна, высота и расположение его на экране, используйте оператор **Set Window**.

Оператор **Set Legend** может использоваться в файле Рабочего Набора. Для примера Вы можете открыть окно Карты, создать условное выделение, открыть окно Легенды и сохранить Рабочий Набор (например, под именем LEGEND.WOR). Теперь откройте файл Рабочего Набора в любом текстовом редакторе и Вы увидите оператор **Set Legend**, задающий те настройки, которые были использованы в открытом ранее окне Легенды.

Независимо от количества открытых окон Карт, на экране может присутствовать только одно окно "Легенды", относящееся к одному окну Карты. В операторе **Set Legend** параметр *window\_id* задает окно Карты, а не легенды. Если параметр *window\_id* не задан, то оператор использует самое верхнее окно Карты.

Предложение **Layer** задает изменения в описании слоя в Легенде. Слой идентифицируется либо своим порядковым номером в окне Карты, либо именем, либо задается словами **Layer Prev**.

Предложение **Layer Prev** идентифицирует слой, который был создан или изменен последним с помощью операторов **Shade** или **Set Shade**.

Если Карта содержит два или более тематических слоев, то оператор **Set Legend** может содержать столько же предложений **Layer**, по каждому на один тематический слой.

Оставшиеся ключевые слова в операторе **Set Legend** составляют предложение **Layer**, то есть описываемые ниже предложения являются ключевыми словами в предложении **Layer**.

Предложение **Count** определяет показ в скобках количества записей, принадлежащих к данному диапазону. Предложения **Shades**, **Symbols** и **Lines** задают показ в строке легенды элементов оформления, соответствующих диапазонам. Если оператор включает предложение **Shades On**, то в окне легенды будут показаны образцы штриховок, используемых картой. Если оператор включает предложение **Symbols On**, то в окне легенды будут показаны образцы символов точечных объектов. Если оператор включает предложение **Lines On**, то в окне легенды будут показаны образцы линий и контуров.

Предложения **Title** и **Subtitle** управляют показом заголовка и подзаголовка соответственно.

Каждая из этих строк не должна быть длиннее 32 символов. В окне легенды строка заголовка всегда располагается вверху, строка подзаголовка ниже, а затем идут описания диапазонов.

Если не задать предложение **Ascending On**, то строки легенды, описывающие диапазоны, будут располагаться в стандартном, возрастающем порядке. Иначе они расположатся в убывающем порядке.

Предложение **Ranges** задает текстовое описание диапазона (*range\_title*), скомбинированное с режимом, задаваемым предложением **Display**. Вы должны задать параметр *range\_title* для каждого диапазона, а управлять его представлением можно предложениями **Display On** или **Display Off**. Предложение **Ranges** должно включать предложение *range\_title Display* для каждого диапазона тематической Карты, даже для диапазонов, которые не будут показаны.

Если на некий слой условное выделение производилось методом размерных символов, то Вам понадобится две комбинации *range\_title Display*. Если условное выделение производилось методом задания плотности точек, то Вам понадобится задать одну комбинацию *range\_title Display*. В остальных случаях Вам надо будет дополнительно к описаниям диапазонов задать еще одну строку для диапазона "остальные", в который попадают все объекты, не охваченные диапазонами.

### Смотрите также:

**Map, Open Window, Set Map, Set Window, Shade**

### Оператор Set Map

#### Назначение:

Изменяет настройки отображения объектов в окне Карты.

#### Синтаксис:

Основная часть оператора **Set Map** имеет следующий синтаксис:

```
Set Map
[ Window window_id ]
[ Center ( longitude, latitude ) [ Smart Redraw ] ]
[ Clipping { Object clipper | Off | On } | Using [ Display { All | PolyObj } | Overlay ] } ]
[ Zoom { zoom_distance [ Units dist_unit ] | Entire [ Layer layer_id ] } ]
[ Preserve { Scale | Zoom } ]
[ Display { Scale | Position | Zoom } ]
[ Order layer_id, layer_id [ , layer_id ... ] ]
[ Pan pan_distance [ Units dist_unit ]
    { North | South | East | West } [ Smart Redraw ] ]
[ CoordSys... ]
[ Area Units area_unit ]
[ Distance Units dist_unit ]
[ Distance Type { Spherical | Cartesian } ]
[ XY Units xy_unit ]
[ Display Decimal { On | Off } [ Display Grid ]
[ Scale screen_dist [ Units dist_unit ] For map_dist [ Units dist_unit ] ]
[ Redraw { On | Off } ]
[ Inflect num_inflections [ by percent ]
    Contrast contrast_value ]
    [ Brightness brightness_value ]
    [ { ALPHA <alpha_value> } { TRANSLUCENCY <translucency_percent> } ]
    [ TRANSPARENCY { OFF | ON } ]
    [ COLOR <transparent_color_value> ]
    [ GrayScale { On | Off } ]
[ Round rounding_factor ]
[ Relief { On | Off } ]
[ Move Nodes { value | Default } ]
[
    LAYERCLAUSE
    LAYERCLAUSE ...
```

где

*window\_id* – целочисленный идентификатор окна Карты;

*longitude*, *latitude* – координаты нового центра Карты;

*clipper* – объектное выражение, определяющее фрагмент-врезку;

*zoom\_distance* – размер фрагмента, показанного в окне Карты;

*layer\_id* – идентификатор слоя на Карте, число типа Smallint или строка с именем таблицы, соответствующей слою Карты;

*pan\_distance* – сдвиг Карты;

Предложение **CoordSys** определяет систему координат.

*area\_unit* – строка, задающая единицу измерения площади (список единиц смотрите в описании оператора **Set Area Units**);

*paper\_unit* – строка, задающая "бумажную" единицу измерения (список единиц смотрите в описании оператора **Set Paper Units**);

*distance* - может быть рассчитана на сфере (**Spherical**) или на плоскости (**Cartesian**). Все расстояния, длины, периметры и площади для объекта в окне Карты могут рассчитываться по одному из двух методов. Обратите внимание, что если **Coordsys** окна Карты является план-схемой, то вычисления будут по декартовому методу **Cartesian**, независимо от настроек, и если **Coordsys** окна Карты это Широта/Долгота, то вычисления будут осуществляться по сферическому методу (**Spherical**), независимо от настроек.

*xy\_unit* – строка, задающая единицу измерения X/Y-координат (например, "m" – метр или "degree" – градус);

*Relief* - включает и отключает отмывку рельефа в гридах. Грид должен иметь рассчитанные значения необходимые для создания отмывки для того, чтобы это предложение имело эффект. Информация об отмывке рельефа может быть рассчитана для грида командой **Relief Shade**.

*Move Node* - может принимать значения 0 или 1. Если значение равно 0, дублирующиеся узлы не удаляются. Если значение равно 1, все дублирующиеся узлы с одного слоя удаляются. Если величина *Move Node* задана, то будет использоваться заданное значение. Если эта величина не задана, то будет использоваться стандартное значение из Настроек.

*screen\_dist* и *map\_dist* – задают масштаб Карты (например, *screen\_dist* = 1 mm, *map\_dist* = 1 km).

*num\_inflections* - это числовое выражение, определяющее число точек перелома цветов:значений.

*color:expr* - это выражение для цвета, часть значения точки перелома цвет:значение.

*alpha\_value* - это целое, представляющее значение альфа-канала для полупрозрачности.

Значение изменяется в пределах 0-255. 0 - это полная прозрачность. 255 - это полная непрозрачность. Значения между 0-255 делают изображение слоя полупрозрачным.

*translucency\_percent* - это целое, представляющее процент полупрозрачности для растров и слоев поверхности. Значение изменяется в диапазоне между 0-100. 0 - это полностью непрозрачное, 100 - это полностью прозрачное.

Или **ALPHA** или **TRANSLUCENCY** должны быть определены, но каждый из них определяет то же самое но разными путями. Если указано несколько таких параметров, будет использоваться последний.

Параметры **ALPHA** и **TRANSLUCENCY** являются новыми в операторе **Set Map**. Они применяются для растровых слоев и слоев поверхности.

Параметры **CONTRAST**, **BRIGHTNESS** и **GRAYSCALE** теперь поддерживаются и для растровых слоев. Их можно применять и для растровых слоев, и для слоев поверхности.

Параметры **TRANSPARENCY** и **COLOR** являются новыми для **Set Map** и применяются только для растровых слоев.

Параметр **TRANSPARENCY** определяет, является ли индивидуальный цвет прозрачным для растрового слоя.

*xy\_unit* - это строка, представляющая название единиц измерения координат X и Y (например, "m" для метров, "degree" для градусов). Если в качестве "XY Units" выставлены

## Оператор Set Map

---

градусы, то предложение **Display Decimal** определяет десятичные градусы. Установите **On** для отображения карты в десятичных градусах или **Off** для установки градусов, минут, секунд. Установите **Display Grid** для отображения в формате Military grid reference.

```
Set Map XY Units "m" Display Grid
Set Map XY Units "degree" Display Grid
Set Map XY Units "degree" Display Decimal On
Set Map XY Units "degree" Display Decimal Off
```

Следующий оператор определяет метры в качестве единиц измерения координат:

```
Set Map XY Units "m"
```

В синтаксисе выше *LAYERCLAUSE* - это не ключевое слово MapBasic, а название раздела, который будет описан ниже.

*LAYERCLAUSE* соответствует одному слою Карты и имеет следующий синтаксис:

```
[ Layer layer_id
[ Activate { [ Using launch_expr ] [ On { [ Labels ] [ Objects ] } ] [ Relative Path { On | Off } ] }
[ Editable { On | Off } ]
[ Selectable { On | Off } ]
[ Zoom ( min_zoom, max_zoom ) [ Units dist_unit ] [ { On | Off } ] ]
[ Arrows { On | Off } ]
[ Centroids { On | Off } ]
[ Default Zoom ]
[ Nodes { On | Off } ]
LABELCLAUSE
[ Display { Off | Graphic | Global } ]
[ Global Line ... ]
[ Global Pen ... ]
[ Global Brush ... ]
[ Global Symbol ... ]
[ Global Font ... ]
]
```

где

*layer\_id* – идентификатор слоя Карты, число типа Smallint или строка с именем таблицы, соответствующей слою Карты;

*min\_zoom* – минимальное значение для масштабного эффекта слоя;

*max\_zoom* – максимальное значение для масштабного эффекта слоя.

*launch\_expr* - это выражение, которое обеспечивает запуск файла, когда активизируется ассоциированный с ним объект.

Предложение **Using** устанавливает выражение для имени файла и предложение **On** устанавливает режим активизации. Требуется как минимум одно из этих предложений. Если включено предложение **Using**, то требуется задать параметр *filename\_expr*.

Если включено предложение **On clause**, то требуется или одно или оба предложения Labels и Objects. Если включено только предложение Labels, то произойдет активизация только подписей.



Если включено предложение **Objects**, то произойдет активизация только объектов. Если присутствуют оба этих ключевых слова, то произойдет активизация и объектов и подписей. В стандартном варианте активизируются только подписи то произойдет активизация только объектов.

Используйте **Relative Path On** когда запускаемые файлы хранятся в местах, которые определяются относительно таблицы, с которой они связаны.

Используйте **Relative Path Off** когда геолинки (HotLinks) имеют адреса URL или полный путь к файлам; это стандартный вариант.

Предложение **Line** определяет стиль объектов "линия" и "полилиния".

Предложение **Brush** определяет стиль штриховки.

Предложение **Pen** определяет стиль линии контуров заштрихованных объектов.

Предложение **Symbol** определяет стиль символа.

Предложение **Font** определяет стиль шрифта текстовых объектов.

**LABELCLAUSE** задает настройку подписей слоя и имеет следующий синтаксис:

```
[ Label [ Line { Simple | Arrow | None } ]
  [ Position [ Center ] [ Above | Below ] [ Left | Right ] ]
  [ Font... ]
  [ Pen... ]
  [ With label_expr ]
  [ Parallel { On | Off } ] ]
[ Visibility { On | Off | Zoom(min_vis, max_vis) [Units dist_unit] } ]
[ Auto [ { On | Off } ] ]
[ Overlap [ { On | Off } ] ]
[ Duplicates [ { On | Off } ] ]
[ Max [ number_of_labels ] ]
[ Offset offset_amount ]
[ Default ]
[ Object ID
  [ Table alias ]
  [ Visibility { On | Off } ]
  [ Anchor (anchor_x, anchor_y) ]
  Text text_string
  [ Position [ Center ] [ Above | Below ] [ Left | Right ] ]
  [ Font ... ]
  [ Pen ... ]
  [ Line { Simple | Arrow | None } ]
  [ Angle text_angle ]
  [ Offset offset_amount ]
  [ Callout (callout_x, callout_y) ] ]
[ Object ... ]
]
```

где

*label\_expr* – выражение, используемое для подписывания объекта;

*min\_vis*, *max\_vis* – минимальное и максимальное значения для масштабного эффекта подписи;

*dist\_unit* – строка с именем единицы измерения (например, "mi" для миль, "m" для метров; см.

описание оператора **Set Distance Units**);

## Оператор Set Map

---

*number\_of\_labels* – целое число типа Integer, представляющее максимальное количество подписей которое может MapInfo показать на слое (по умолчанию лимита нет);  
*offset\_amount* – целое число от 0 до 50, расстояние в точках от подписи до точки привязки на подписываемом объекте;  
*ID* – целочисленный идентификатор изменяемой подписи (идентификатор подписи равен идентификатору строки, к которой присоединен подписываемый объект (это предложение генерируется автоматически при сохранении Рабочего Набора);  
*alias* – псевдоним таблицы-компонента шитой Карты (предложение **Table alias** порождает ошибку, если слой не является компонентом шитой Карты).;  
*anchor\_x* и *anchor\_y* – координаты, задающие закрепленное положение подписи;  
*text\_string* – строка с текстом подписи;  
*text\_angle* – угол поворота подписи в градусах;  
*callout\_x* и *callout\_y* – координаты, определяют конец указки при подписи.

### Описание:

Оператор **Set Map** задает настройки для окна Карты. Этот оператор позволяет программе управлять отображением слоев Карты так же, как это может делать пользователь при помощи команд MapInfo КАРТА > УПРАВЛЕНИЕ СЛОЯМИ, КАРТА > ПОКАЗАТЬ ПО-ДРУГОМУ, КАРТА > РЕЖИМЫ. Если параметр *window\_id* не задан, то действие команды распространяется на самое верхнее окно Карты.

Заметим, что оператор управляет **Set Map** настройками в окне Карты. Для изменения таких атрибутов, как размер окна и его расположение на экране, используется оператор **Set Window**.

Оператор **Set Map** может использоваться в файле Рабочего Набора, если в нем есть хотя бы одно окно Карты. Для примера Вы можете открыть окно Карты и сохранить Рабочий Набор (например, под именем MAPPER.WOR). Теперь откройте файл Рабочего Набора в любом текстовом редакторе и Вы увидите оператор **Set Map**, задающий те настройки, которые были ранее установлены в окне Карты.

Все предложения оператора **Set Map** не являются обязательными, но хотя бы одно должно присутствовать. При этом важен порядок расположения предложений, несоблюдение порядка может привести к синтаксической ошибке в программе.

### Изменение изображения в окне Карты

Следующие предложения изменяют показ Карты в окне, т.е. центральную точку и размеры показываемой в окне области.

#### Center

Предложение задает центр карты в окне. Например, город Нью-Йорк расположен приблизительно на 74 долготы и 41 широты. Следующий оператор **Set Map** помещает Нью-Йорк в центр окна Карты:

```
Set Map Center (-74.0, 41.0)
```

При этом широта и долгота должны задаваться в десятичных единицах, а не в градусах, минутах и секундах.

Выполнение оператора **Set Map...Center** ведет к полной перерисовке окна Карты, если в оператор не включено предложение **Smart Redraw**. Детали о предложении **Smart Redraw** смотрите ниже.

### Pan

Предложение перемещает окно карты в заданном направлении. Например, следующий оператор перемещает Карту на 100 километров к северу:

```
Set Map Pan 100 Units "km" North
```

Обычно, после выполнения оператора **Set Map ... Pan** изображение в окне Карты перерисовывается полностью. Если в оператор включено предложение **Smart Redraw**, то Map-Info будет обновлять только ту часть изображения, которая этого требует (обновление изображения в окне Карты будет происходить так, как, если бы пользователь использовал в окне инструмент Ладонка).

```
Set Map Pan 100 Units "km" North Smart Redraw
```

**Внимание:** если используется предложение **Smart Redraw**, окно Карты передвигается шагами, кратными восьми пикселям. Из-за этого Карта может показываться не совсем так, как ожидалось. Например, передвигая Карту на Север на 100 км, Вы добьетесь только передвижения на 80 километров, из-за того, что 20 километров на экране занимают более восьми пикселей.

### Scale

Изменяет масштаб показа Карты в окне. Например, следующий оператор изменяет увеличение Карты так, чтобы в 1 дюйме изображения было показано 10 миль Карты:

```
Set Map Scale 1 Units "in" For 10 Units "mi"
```

### Zoom

Определяет размер в ширину фрагмента Карты, показанного в окне. Например, следующий оператор **Set Map** показывает участок шириной в 100 километров (если текущими единицами расстояний в MapBasic сейчас являются "km"):

```
Set Map Zoom 100 Units "km"
```

Предложение **Zoom Entire Layer *layer\_id*** является эквивалентом команды КАРТА > ПОКАЗАТЬ СЛОЙ ПОЛНОСТЬЮ в MapInfo. Если опустить из этого предложения слово "Layer", то будут полностью показаны все слои Карты.

```
Set Map Zoom Entire Layer 2 ' Полностью показать 2 слой
Set Map Zoom Entire ' Полностью показать всю карту
```

## Управление поведением Карты в окне

Следующие предложения определяют поведение Карты в окне.

### Area Units

Предложение задает единицы измерения площади. Список возможных единиц приведен в описании оператора **Set Area Units**.

```
Set Map Area Units "sq km"
```

### Clipping

Предложение задает фрагмент-врезку в окне Карты. Операция соответствует действию в Map-Info команды КАРТА > ВЫБРАТЬ ОБЛАСТЬ ВРЕЗКИ. После назначения области врезки Вы можете управлять показом фрагмента-врезки операторами **Clipping On** или **Clipping Off**.

```
Set Map Clipping Object obj_variable_name
```

### Оператор Set Map Statement для врезки (Clip Region)

Существует три режима, которые могут использоваться для создания врезки. Использование режима Overlay приводит к применению функциональности MapInfo Overlay (Erase Outside) для создания врезки. Полилинии и Регионы будут обрезаться по границе обрезающего полигона. Точки и Подписи будут показаны на врезке полностью только в том случае, если они лежат внутри обрезающего полигона. Текст всегда отображается и никогда не обрезается. Стили для всех объектов никогда не обрезаются. (Этот метод используется во всех версиях, ранее MapInfo 6.0.)

Использование режима Display All, приводит к созданию врезки системными средствами Windows. Все объекты (включая точки, подписи и текст) будут обрезаться по границе полигона, ограничивающего врезку. Все стили также обрезаются по границе полигона. Этот режим является стандартным.

Использование режима Display PolyObj приводит к созданию врезки системными средствами Windows, при чем обрезаются только полилинии и регионы. Стили для полилиний и регионов обрезаются по границе полигона врезки. Точки и Подписи будут изображены полностью только, если они лежат внутри границ полигона. Текст всегда отображается и никогда не обрезается. Этот режим по функциональности близок к режиму Overlay в ранних версиях MI Pro, до 6.0.

В главном, системные средства Windows из методов Display All и Display PolyObj обеспечивают лучший результат чем функциональность Overlay. Например:

Set Map Clipping Object *obj\_variable\_name* Using Display All

#### CoordSys...

Стандартное предложение задает в окне Карты систему координат и проекцию. Синтаксис смотрите в описании стандартного предложения **CoordSys**.

**Замечание:** Если оператор **Set Map** снабжен предложением **CoordSys**, то координатная система, установленная для прикладной программы, автоматически переназначается.

#### Display

Предложение **Display** выбирает, что показывать в левом нижнем углу активного окна Карты: размер (**Display Zoom**), масштаб (**Display Scale**) или положение курсора в десятичных единицах координат широта/долгота (**Display Position**).

**Set Map Display Position**

#### Distance Units

Предложение задает единицы измерения расстояний в окне "Линейка". Список единиц приведен в описании оператора **Set Distance Units**.

**Set Map Distance Units "km"**

#### Preserve

Предложение определяет поведение Карты при изменении пользователем размеров окна. Предложение **Preserve Scale** задает показ Карты всегда в одном масштабе, независимо от изменений размеров окна. И, наоборот, **Preserve Zoom** задает увеличение или уменьшение масштаба карты в зависимости от увеличения или уменьшения окна, сохраняя в окне Карты фрагмент постоянного размера. Установки этого предложения такие же, как соответствующие режимы в диалоге, вызываемом командой КАРТА > ПОКАЗАТЬ.

### Redraw

Предложение управляет автоматической перерисовкой Карты в окне. Если программа выполнила оператор **Set Map Redraw Off**, то следующие операторы, изменяющие карту (такие как **Set Map**, **Add Map Layer**, **Remove Map Layer**), будут выполняться без автоматического обновления изображения в окне Карты. Теперь, после некоторых изменений, выполним оператор **Set Map Redraw On**, восстанавливающий режим автоматической перерисовки окна. В окне отобразятся все изменения, которые были выполнены "вслепую". Смотрите также описание оператора **Set Event Processing**.

### XY Units

Предложение задает координатные единицы положения курсора, показываемые в левом нижнем углу окна Карты. Единицами могут быть градусы ("degree") для измерения широты и долготы или единицы измерения расстояния, например, метры ("m"):

```
Set Map XY Units "m"
```

## Изменение порядка слоев

Предложение **Order** задает порядок прорисовки слоев Карты на экране. Каждый параметр *layer\_num* – номер слоя Карты. Единица соответствует самому верхнему слою Карты (который рисуется последним, поверх остальных). Косметический слой является специальным слоем и имеет номер слоя, равный 0 (нулю). Он рисуется всегда последним и его не нужно задавать в предложении **Order**. В следующем примере первый (верхний) слой Карты и второй (находящийся под ним) меняются местами. Косметический слой остается самым верхним.

```
Set Map Order 2, 1, 3, 4
```

## Изменение поведения отдельного слоя

Предложение **Layer** управляет настройкой одного слоя. Так как окно карты обычно содержит несколько слоев, то и предложений **Layer** может быть несколько. Стоящие в операторе **Set Map** за предложением **Layer layer\_id** все другие предложения (до следующего **Layer**) являются подпредложениями, задающими настройки слоя *layer\_id*.

### Editable

В предложении **Layer** предложение **Editable** может установить режим изменяемости только для одного слоя. Если слой становится изменяемым, то он автоматически становится и доступным. Следующий оператор **Set Map** делает изменяемым только верхний некосметический слой:

```
Set Map
  Layer 1 Editable On
```

### Selectable

Предложение устанавливает для данного слоя режим доступности, т.е. разрешает выбирать на нем объекты такими инструментами, как Стрелка. Доступными могут быть несколько слоев Карты. Следующий оператор **Set Map** делает доступным верхний некосметический слой, а следующие два – недоступными:

```
Set Map
  Layer 1 Selectable On
  Layer 2 Selectable Off
  Layer 3 Selectable Off
```

### Zoom

Предложение задает пределы для масштабного эффекта, то есть режима показа слоя только в определенных пределах увеличения. Например, улицы города можно показывать только тогда, когда размер карты уменьшится до 10 км.

#### Set Map

```
Layer 1 Zoom (0, 10) Units "km" On
```

Слово **Zoom** может содержать слово **Off**, отключающее масштабный эффект для слоя.

## Предложение Set Map Clause для геолинка (HotLink)

Активный объект это такой объект в окне карты, который имеет ассоциированный адрес URL или ассоциированный файл. Щелкнув на активном объекте новым инструментом Геолинк, запускается ассоциированный адрес URL или файл. Например, если строка *http://www.esti-map.ru* ассоциирована с точкой на карте, то щелкнув на таком точечном объекте, или подписи, получим в результате открывшейся интернет-браузер, показывающий страницу *http://www.esti-map.ru*. Можно связывать с объектами карты другие типы файлов; Рабочие наборы MapInfo (.wor), таблицы (.tab) или приложения (.mbx), документы Word (.doc), исполняемые файлы (.exe), и др.. Любые типы файлов, которые система может запускать, могут быть связаны с объектами карты.

Настройка относительного адреса (Relative Path)

Настройка относительного адреса (Relative Path) позволяет определить местоположение файла относительно таблицы. Например: пусть таблица *c:\data\states.tab* содержит геолинки к файлам рабочих наборов, хранящихся в директории *c:\data*. Файл рабочего набора для Нью-Йорка, *newyork.wor*, хранится в *c:\data\ny* и Геолинк связан с "ny\newyork.wor". Установка относительного пути сообщает MapInfo, что надо добавить вторую часть к пути, по которому находится .tab файл, чтобы получился полный путь, и в результате получится строка "c:\data\ny\newyork.wor".

**Внимание:** Геолинки, определенные как адреса URL, не изменяются перед запуском, независимо от установки относительного пути. Чтобы определить, является ли Геолинк адресом URL, используется функция ShellAPI.

## Изменение представления отдельного слоя

### Arrows

Предложение управляет показом стрелок.

### Centroids

Предложение управляет показом центроидов объектов.

### Nodes

Предложение управляет показом узлов на объектах.

Следующий пример включает режим показа стрелок, центроидов и узлов объектов первого некосметического слоя Карты:

#### Set Map

```
Layer 1 Arrows On Centroids On Nodes On
```

## Display

Предложение **Display** управляет показом слоя в окне Карты. Предложение **Display Off** отменяет показ слоя; **Display Graphic** показывает объекты слоя в собственном (сохраненным в таблице) оформлении; **Display Global** позволяет настраивать отдельные компоненты оформления объектов:

**Global Line** определяет стиль линейных объектов: линий и полилиний. Предложение **Line** имеет конструкцию, подобную **Pen**.

**Global Pen** задает стиль линий, окружающих замкнутые объекты.

**Global Brush** задает стиль штриховки замкнутых объектов.

**Global Symbol** задает стиль символов для точечных объектов.

**Global Font** задает шрифт для текстовых объектов.

Следующий оператор разрешает показ объектов первого слоя в собственном стиле:

```
Set Map
  Layer 1 Display Graphic
```

Следующий оператор устанавливает режим показа объектов первого слоя тонкими зелеными линиями, контурами и зеленой заливкой:

```
Set Map
  Layer 1 Display Global
  Global Line(1, 2, GREEN)
  Global Pen (1, 2, GREEN)
  Global Brush (2, GREEN, WHITE)
```

## Изменение режима подписывания отдельного слоя

Предложение **Label** задает режим подстановки и поведения подписей на слое. Предложение имеет следующие подпредложения:

### Line

Определяет тип указки, которая сопровождает подпись при ее перемещении, или ее отсутствие.

Вы можете задать **Line Simple**, **Line Arrow** или **Line None**. Например:

```
Set Map Layer 1
  Label Line Arrow
```

### Position

Управляет расположением подписи относительно центра объекта, к которому она прикреплена. Например, следующий оператор располагает подписи сверху и справа от центра:

```
Set Map Layer 1
  Label Position Above Right
```

### Font

Определяет шрифт подписи.

### Pen

Задаст стиль линии указки. Стиль линии указки имеет смысл для режимов **Line Simple** и **Line Arrow**, и когда пользователь сдвигает подпись с заданного положения:

```
Set Map Layer 1
  Label Line Arrow
  Pen( 2, 1, 255)
```

### With

Задаёт выражение, образующее текст надписи. Например, следующий оператор использует функцию **Proper\$( )** к значениям из колонки с именами больших городов:

```
Set Map Layer 1
Label With Proper$(Cityname)
```

### Parallel

Определяет, будет ли строка подписи линейного объекта параллельна подписываемой линии.

```
Set Map Layer 1
Label Parallel On
```

### Visibility

Управляет показом подписей для одного слоя. Предложение **Visibility Off** выключает показ как автоматических подписей, так и созданных вручную. Предложение **Visibility Zoom ...** устанавливает показ подписей только, когда Карта находится в определенном масштабном диапазоне. Следующий пример разрешает подписывание тогда, когда размер Карты будет равен 2 километрам и менее.

```
Set Map Layer 1
Label Visibility Zoom (0, 2) Units "km"
```

### Auto

Управляет автоматическим подписыванием. Предложение **Auto Off** отключает автоматические подписи, но остаются созданные вручную.

### Overlap

Управляет режимом, разрешающим или запрещающим MapInfo рисовать пересекающиеся подписи. Чтобы избежать наложение подписей установите **Overlap Off**.

### PartialSegments

Управляет способностью MapInfo подписывать линейные объекты, когда центроид вне зоны видимости окна карты. Если Вы определили **PartialSegments On** (это соответствует установке флажка Подписывание сегментов линий в MI, то MapInfo подпишет видимые части объектов. Если Вы определили **PartialSegments Off**, линейные объекты будут подписаны только если их центроид находится в окне Карты.

### Duplicates

Управляет режимом, разрешающим или запрещающим MapInfo дублировать подписи.

### Max number\_of\_labels

Устанавливает максимальное число подписей, которое MapInfo может показать на этом слое. По умолчанию лимита нет.

### Offset offset\_amount

Задаёт отступ подписи от центроида. Параметр *offset\_amount* может принимать значения от 0 до 50 шрифтовых точек. Если задать **Offset 0**, то подпись будет примыкать к центроиду. Если задать **Offset 10**, то подпись будет располагаться на 10 точек в сторону. Отступ будет игнорирован, если расположение подписи будет задано в центре объекта (**Position Center**).

Следующий оператор задаёт расположение подписей справа и на 10 точек в сторону от центроида:



```
Set Map Layer 1
Label Overlap On Position Right Offset 10
```

## Default

Восстанавливает все подписи на слое и удаляет все внесенные вручную подписи и изменения в автоматических. Пример:

```
Set Map Layer 1 Label Default
```

## Object

Предложение позволяет задать отдельную подпись для индивидуального объекта. Например, если Вы изменили одну подпись в MapInfo и сохранили Рабочий Набор, то он будет содержать оператор **Set Map** со столькоими предложениями **Object**, сколько подписей было индивидуально изменено.

Пример использования предложения **Object** Вы можете увидеть, если откроете такой Рабочий Набор в любом текстовом редакторе.

## Настройки отдельного слоя, имеющие постоянный эффект

Предложение **Default Zoom** воздействует на таблицу, а не на Карту. Оно используется для установки значений стандартного увеличения и центральной точки для таблицы такими, какие они сейчас в окне Карты.

Каждая таблица, к которой присоединена геоинформация, имеет стандартное увеличение и центральную точку. Эти значения применяются для представления открываемой Карты.

Если в операторе **Set Map...Layer** содержится предложение **Default Zoom**, MapInfo помещает в таблицу текущие значения увеличения и координат центральной точки. Например, следующий оператор изменяет размер и центр окна для таблицы, изображенной на первом слое:

```
Set Map Layer 1 Default Zoom
```

**Default Zoom** срабатывает сразу, не дожидаясь операции сохранения таблицы.

## Настройка режима совмещения

Если окно Карты имеет свои собственные настройки, в частности режима совмещения, то общие настройки не будут использоваться. Общие настройки (стандартные), будут также применяться ко всем новым окнам. Установки для существующих окон могут быть сделаны с помощью оператора Set Map Move Nodes языка MapBasic.

## Пример:

Программа открывает две таблицы и показывает их в окне Карты. Затем оператор **Set Map** изменяет окно.

```
Open Table "world"
Open Table "cust1993" As customers
Map From customers, world

Set Map
Center (100, 40)      ' центрирование карты США
Zoom 4000 Units "mi" ' показ континентальной части США
Preserve Zoom         ' сохранять размер Карты
Display Position      ' показывать в углу широту и долготу
Layer 1
```

## Оператор Set Map

---

```
Editable On  
Layer 2  
Selectable Off  
Display Global  
Global Brush (2, 255, 65535)
```

### Смотрите также:

[Add Map](#), [LayerInfo\( \)](#), [Map](#), [MapperInfo\( \)](#), [Remove Map](#), [Set Window](#)

## Оператор Set Map3D

### Назначение

Изменяет настройки существующего окна 3DКарты.

### Синтаксис

#### Set Map3D

```
[ Window window_id ]
[ Camera [ Zoom factor | Pitch angle | Roll angle | Yaw angle | Elevation angle
          Position (x,y,z) | FocalPoint (x,y,z) ] ]
[ Light [ Position (x,y,z) | Color lightcolor ] ]
[ Resolution (res_x, res_y) ]
[ Scale grid_scale ]
[ Background backgroundcolor ]
[ Refresh ]
```

*window\_id* - это идентификатор окна карты, которое содержит слой Grid. Если слой Grid не найден, то будет выдано сообщение об ошибке.

*mapper\_creation\_string* - указывает командную строку, которая создает изображение по гриду.

**Camera** - определяет позицию и ориентацию камеры.

*angle* - это угол, измеряемый в градусах. Горизонтальный угол в диалоге может изменяться от 0 до 360 и вращает карту вокруг центральной точки грида. Вертикальный угол в диалоге изменяется от 0 до 90 и измеряет вращение в вертикальной плоскости прямо от стартовой точки прямо над картой.

**Pitch** регулирует поворот камеры вокруг оси X

**Roll** регулирует поворот камеры вокруг оси Z

**Yaw** регулирует поворот камеры вокруг оси Y

**Elevation** регулирует поворот камеры вокруг оси X относительно фокальной точки камеры

**Position** - регулирует позицию камеры/источника света

**FocalPoint** регулирует позицию фокуса камеры/источника света

**Orientation** регулирует позицию камеры ViewUp, ViewPlane Normal и Clipping Range (используется для повышения устойчивости точки наблюдения).

**Resolution** - это разрешение в направлении X и Y. Эти значения могут увеличиваться вплоть до максимального разрешения грида. Если грид имеет разрешение 200x200 то и разрешение в окне карты не будет больше чем это значение 200x200. Вы не можете увеличивать разрешение грида, можно менять только разрешение его изображения регулирует.

*grid\_scale* - значение масштаба грида в направлении Z. Значение больше 1, будет преувеличивать топологию в направлении Z, а значение <1 будет преуменьшать топологию в направлении Z.

*backgroundcolor* - это цвет, используемый для подложки, в RGB.

**Units** - определяет единицы измерения грида (третьей компоненты). Не указывайте единиц, например, для температурного поля или плотности. Эту настройку надо делать к моменту создания грида. Нельзя изменить единицы измерения позднее, используя оператор Set Map3D или диалог Настройки.

**Refresh** создает новую текстуру из исходной таблицы.

## Оператор Set Map3D

---

### Описание

Изменяет настройки уже существующей 3D Карты.

### Пример

```
Dim win3D as Integer
Create Map3D Resolution(75,75) Resolution(100,100) Scale 2 Background
RGB(255,0,0)
win3D = FrontWindow()
Set Map3D Window win3D Resolution(150,100) Scale 0.75 Background
RGB(255,255,0)
Changes the original 3DMap window's resolution in the X and Y, the
scale to de-emphasize the grid in the Z direction (< 1) and change the
background color to yellow.
```

### Смотрите также

Оператор Create Map3D, Функция Map3Dinfo

## Оператор Set Next Document

### Назначение:

Переподчиняет документальное окно в MapInfo (например, окно Карты становится подчиненным или порожденным окном приложения на Visual Basic).

### Предупреждение:

Этот оператор выполняется только в среде Microsoft Windows.

### Синтаксис:

#### Set Next Document

{ **Parent** *HWND* | **Style** *style\_flag* | **Parent** *HWND* **Style** *style\_flag* }

где

*HWND* – целое число типа, уникальный номер порождающего окна;

*style\_flag* – целочисленный код (смотрите таблицу ниже), задающий стиль окна.

### Описание:

Этот оператор используется в приложениях интегрированной картографии. Концепция интегрированной картографии описана в 12 главе *Руководства пользователя MapBasic*.

Чтобы переподчинить окно MapInfo выполните оператор **Set Next Document**, а за ним любой из создающих окно операторов: **Map**, **Browse**, **Graph**, **Layout** или **Create Legend**.

Предложение **Parent** используется для задания существующего окна, которое будет считаться порождающим по отношению к окну MapInfo, которое Вы собираетесь создать. Предложение **Style** определяет стиль окна. Если Вы создаете документальное окно (такое как Карту), то включите оба предложения.

Значение параметра *style\_flag* должно быть равно коду из следующей таблицы.

#### Значение style\_flag

WIN\_STYLE\_CHILD

WIN\_STYLE\_POPUP

WIN\_STYLE\_POPUP\_FULLCAPTION

WIN\_STYLE\_STANDARD

#### Воздействие на следующее окно документа:

Следующее окно создается подчиненным. (Значение кода 1.)

Следующее создаваемое окно имеет стиль рорип и строка заголовка показывается в половину обычной ее высоты. (Значение кода 3.)

Следующее создаваемое окно имеет стиль рорип и создается с обычной строкой заголовка. (Значение кода 2.)

Этот код восстанавливает стиль окна к стандартному виду. (Значение кода 0.) Если Вы выполнили оператор **Set Next Document Style 1**, а затем раздумали назначать окну подчиненный стиль, то восстановить стиль окна можно оператором **Set Next Document Style 0**.

## Оператор Set Next Document

---

Установка стиля и подчиненности окна срабатывает только для следующего создаваемого окна. После того, как оно создается и к нему будут применены стили и подчиненность, MapInfo восстанавливает стандартные режимы подчиненности и стиля. То есть каждое новое переподчиняемое окно требует предварительного срабатывания оператора **Set Next Document**.

**Внимание:** Оператор **Create ButtonPad** переустанавливает режимы подчиненности и стиля, однако новые инструментальные панели не переподчиняются.

Этот оператор переподчиняет окна документов. Чтобы переподчинить окна диалогов, используйте оператор **Set Application Window**. Чтобы переподчинить специальные окна типа “Информация”, используйте оператор **Set Window**.

### Пример:

Программа LEGENDS.MB использует следующие операторы для создания окна Легенды для Карты:

```
Dim win As Integer

win = FrontWindow()

...

Set Next Document
  Parent WindowInfo(win, WIN_INFO_WND)
  Style 1

Create Legend From Window win
```

### Смотрите также:

**Set Application Window, Set Window**

## Оператор Set Paper Units

### Назначение:

Устанавливает единицы измерения, описывающие размеры и положения окон на экране.

### Синтаксис:

**Set Paper Units** *unit*

где

*unit* – строка с именем единицы линейных измерений (например, "cm" – сантиметры).

### Описание:

Оператор **Set Paper Units** назначает так называемые "бумажные" единицы, т.е. единицы линейных измерений на экране, которые используются по умолчанию операторами MapBasic при определении размеров и положений окон MapInfo на экране или объектов на печатном листе. Если оператор **Set Paper Units** не участвовал в программе, то по умолчанию используются дюймы ("in").

Некоторые операторы MapBasic (такие, как **Set Window**) включают предложения **Position**, **Width** и **Height**, с помощью которых устанавливают положение, ширину и высоту окон. Если эти предложения не содержат предложение **Units**, то численные параметры задают размеры в единицах, объявленных ранее оператором **Set Paper Units**, или в дюймах. Например, следующий оператор **Set Window**:

```
Set Window Width 5
```

устанавливает ширину окна на экране. Если до этого оператора были установлены единицами измерений сантиметры, то ширина Карты будет пять сантиметров или же пять дюймов, если единицами измерений на экране по умолчанию приняты дюймы.

Заметим, что "бумажные" единицы, устанавливаемые MapBasic, являются внутренним атрибутом и не доступны пользователю MapInfo.

В следующей таблице в первой колонке представлены значения параметра *unit*, которые могут использоваться в операторе:

Значение <i>unit</i>	Единицы измерения
"cm"	сантиметр;
"in"	дюйм;
"mm"	миллиметр;
"pt"	пункт (точка);
"pica"	пика.

### Смотрите также:

**Set Area Units**, **Set Distance Units**

# Оператор Set PrismMap

## Назначение

Изменяет настройки существующего окна Карты-призмы.

## Синтаксис

**Set PrismMap**

```
[Window window_id ]  
[ Camera [ Zoom factor | Pitch angle | Roll angle | Yaw angle | Elevation angle  
          Position (x,y,z) | FocalPoint (x,y,z) ] ]  
[ Light [ Position (x,y,z) | Color lightcolor ] ]  
[ Scale grid_scale ]  
[ Background backgroundcolor ]  
[ Label With infotips_expr ]  
[ Refresh ]
```

*window\_id* - это идентификатор окна карты, которое содержит слой поверхности. Если такой слой не найден, появится сообщение об ошибке.

*mapper\_creation\_string* определяет командную строку, создающую текстуру для поверхности.

*Camera* определяет позицию и ориентацию камеры.

*angle* - это угол в градусах. Горизонтальный угол (измеряется в диапазоне 0-360 градусов) определяет вращение карты вокруг центральной точки поверхности (grid). Вертикальный угол изменяется в диапазоне 0-90 и измеряет наклон карты от начальной точки.

*Pitch* настраивает вращение камеры вокруг оси X, при этом точкой вращения считается центральная (начальная) точка камеры.

*Roll* настраивает вращение камеры вокруг оси Z, при этом точкой вращения считается центральная (начальная) точка камеры.

*Yaw* настраивает вращение камеры вокруг оси Y, при этом точкой вращения считается центральная (начальная) точка камеры.

*Elevation* настраивает вращение карты вокруг оси X, при этом точкой вращения считается фокальная точка камеры.

*Position* определяет позицию камеры или источника света

*FocalPoint* определяет фокальную точку камеры или источника света

*Orientation* определяет для камеры значение параметров ViewUp, ViewPlane и Clipping Range (используется для инерции зрительного восприятия).

*backgroundcolor* - это цвет, используемый для фона и он определяется функцией RGB.

*infotips\_expr* - выражение, используемое для всплывающей подсказки InfoTips.

*Refresh* восстанавливает текстуру из исходных таблиц.

## Описание

Изменяет настройки уже созданной карты-призмы.

## Пример

Здесь мы изменим разрешение окна карты-призмы по осям X и Y, масштаб по оси Z (< 1) и изменим цвет фона на желтый.

```
Dim win3D as Integer
```



```
Create PrismMap Resolution(75,75) Resolution(100,100) Scale 2  
    Background RGB(255,0,0)  
win3D = FrontWindow()  
Set PrismMap Window win3D Resolution(150,100) Scale 0.75 Background  
    RGB(255,255,0)
```

### Смотрите также

Оператор Create PrismMap

Функция PrismMapInfo()

### Оператор Set ProgressBars

#### Назначение:

Показывает или скрывает диалог-индикатор выполнения процесса.

#### Синтаксис:

**Set ProgressBars { On|Off }**

#### Описание:

Некоторые операторы MapBasic, такие как **Create Object As Buffer**, автоматически выводят диалог, показывающий процент выполнения. Оператор **Set ProgressBars Off** используется для подавления диалога-индикатора выполнения процесса. Если диалог не выводится на экран, то пользователь лишается возможности отменить выполнение процесса кнопкой "Отмена".

Оператор **Set ProgressBars On** возобновляет вывод диалогов-индикаторов на экран.

Если оператор **Set ProgressBars Off** выполняется из MapBasic-программы (MBX-файла), то отключается только порожденный MBX-файлом диалог-индикатор. Те диалоги-индикаторы, которые иллюстрируют действия пользователя, не отключаются. Кроме этого, оператор **Run Menu Command** может игнорировать запрещение показа диалогов-индикаторов, потому что он симулирует выполнение команд меню пользователем.

Чтобы отключить показ диалога-индикатора, появляющегося при выполнении оператора **Run Menu Command**, выполните оператор **Set ProgressBars Off** в окне MapBasic (или пошлите эту команду в MapInfo через механизмы OLE Automation или DDE).

**Замечание:** Если приложение свертывает MapInfo в иконку (например, оператором **Set Window MapInfo Min**), то диалог-индикатор выполнения нужно отключать, так как в этом случае диалог-индикатор "зависает" до тех пор, пока окно MapInfo не раскроется снова. Если отключить показ процента выполнения, то соответствующая операция будет продолжаться, даже если MapInfo свернуто в иконку.

#### Смотрите также:

**ProgressBar**

## Оператор Set Redistricter

### Назначение:

Изменяет состав и характеристики районов.

### Синтаксис (вариант 1):

```
Set Redistricter districts_table
[ Change district_name
  [ To new_district_name ] [ Pen... ] [ Brush... ] [ Symbol... ] ]
[ Add new_district_name [ Pen... ] [ Brush... ] [ Symbol... ] ]
[ Remove district_name ]
```

### Синтаксис (вариант 2):

```
Set Redistricter districts_table
Order { "Alpha" | "MRU" | "Unordered" }
```

где

*districts\_table* – имя таблицы районов (например, Districts);

*district\_name* – строка с именем открытого окна Районирование;

*new\_district\_name* – строка с новым именем для открытого окна Районирование(используется при добавлении или переименовании района).

Слово **Pen** начинает стандартное предложение оператора для определения стиля линии (например, **Pen MakePen(*width, pattern, color*)**).

Слово **Brush** начинает стандартное предложение оператора для определения стиля штриха (например, **Pen MakeBrush(*pattern, forecolor, backcolor*)**).

Слово **Symbol** начинает стандартное предложение оператора для определения стиля символа (например, **Pen MakeSymbol(*shape, color, size*)**).

### Описание:

Оператор **Set Redistricter** изменяет состав районов; процедура районирования начинается оператором **Create Redistricter**. Правила работы с районами подробно описаны в документации MapInfo.

Первый вариант синтаксиса используется для добавления, роспуска и изменения района.

Предложение **Change** изменяет название и/или стиль оформления районов. Предложение **Add** добавляет новый район, а предложение **Remove** распускает существующий, при этом освободившиеся объекты переходят в район "остальные".

Параметры *district\_name* и *new\_district\_name* должны быть строковыми константами или выражениями, даже если колонка Районы численная. Например, если район представлен числом 33, то параметр должен задаваться строкой "33".

Для сортировки строк в окне Районирование используется второй вариант синтаксиса оператора. Ключевое слово **Alpha** задает сортировку в алфавитном порядке. Ключевое слово **MRU** используется, если Вы хотите, чтобы последняя группа, с которой Вы работали, автоматически становилась первой в Списке Районов. В режиме **Unordered** все новые районы добавляются в конец Списка.

### Примеры:

В ходе процедуры районирования следующий оператор создает новый район:

## Оператор Set Redistricter

---

```
Set Redistricter Districts  
Add "NorthWest" Brush MakeBrush(2, 255, 0)
```

Следующий оператор переименовывает район "NE" в "NorthEast" в ходе процедуры районирования:

```
Set Redistricter Districts  
Change "NE" To "NorthEast"
```

Следующий оператор удаляет район "NorthWest" из таблицы DISTRICTS:

```
Set Redistricter Districts  
Remove "NorthWest"
```

Следующий оператор задает упорядочивание строк в Списке Районов по их использованию:

```
Set Redistricter Districts  
Order "MRU"
```

### Смотрите также:

[Create Redistricter](#)

## Оператор Set Resolution

### Назначение:

Устанавливает параметр графического разрешения для операций изменения типа объекта. Эта характеристика влияет на количество узлов в объекте, полученном преобразованием типа объекта.

### Синтаксис:

**Set Resolution** *node\_limit*

где

*node\_limit* – целое число типа SmallInt от 2 до 32 762 включительно; по умолчанию 100.

### Описание:

Оператор **Set Resolution** устанавливает число узлов для преобразования окружности в область. По умолчанию MapInfo создает 100 узлов на окружности или дуге при преобразовании их в область и полилинию. Прирост значения разрешения приводит к более гладким результатам.

Оператор **Set Resolution** влияет на результаты таких команд, как ОБЪЕКТЫ > ПРЕВРАТИТЬ В ОБЛАСТИ и ОБЪЕКТЫ > ПРЕВРАТИТЬ В ПОЛИЛИНИИ. Значение разрешения влияет также на результаты некоторых операторов и функций MapBasic, таких как **ConvertToRegion( )** и **ConvertToPline( )**. Кроме этого, от значения разрешения зависят результаты операций, в которых конвертирование производится автоматически (например, **Objects Split**, **Combine**).

Установка оператора **Set Resolution** не влияет на создание буферной области. Оператор **Create Object As Buffer** и функция **Buffer( )** имеют обязательный параметр, явно задающий разрешение для создания области.

### Смотрите также:

**ConvertToPline( ), ConvertToRegion( )**

## Оператор Set Shade

### Назначение:

Изменяет тематический слой Карты.

### Синтаксис:

```
Set Shade  
[ Window window_id ]  
{ map_layer_id | table (theme_layer_id) }  
...
```

где

*window\_id* – идентификатор окна Карты;

*map\_layer\_id* – число типа SmallInt, задающий номер слоя, который является тематическим;

*table* – имя таблицы, на которой основывается тематический слой;

*theme\_layer\_id* – число типа SmallInt от 1 и больше, задающий номер по хронологии создания тематического слоя.

### Описание:

После того, как оператор **Shade** создаст тематический слой карты, Вы с помощью оператора **Set Shade** можете изменять этот слой. Оператор **Set Shade** выполняет те же действия, что и команда в MapInfo КАРТА > НАСТРОЙКА УСЛОВНОГО ВЫДЕЛЕНИЯ.

Синтаксис оператора **Set Shade** такой же, как у оператора **Shade**, за исключением первых ключевых слов и параметров, задающих слой тематического выделения. Оператор **Set Shade** может идентифицировать слой по его номеру, как в следующем примере...

```
Set Shade  
Window i-map-winid  
2  
With Num-Hh-90  
Graduated 0.0:0 11000000:24 Vary Size By "SQRT"
```

...или по имени таблицы, на данных которой основано тематическое выделение (в скобках указывается номер созданного тематического слоя):

```
Set Shade  
Window i-map-winid  
RUSSIA(1)  
With Num-Hh-90  
Graduated 0.0:0 11000000:24 Vary Size By "SQRT"
```

"RUSSIA(1)" означает первый слой из тематических, основанный на данных таблицы RUSSIA.

### Смотрите также:

**Shade**

## Оператор Set Style

### Назначение

Изменяет текущие стили Pen, Brush, Symbol или Font.

### Синтаксис

Set Style

```
{ Brush ... |
  Font ... |
  Pen ... |
  BorderPen |
  LinePen |
  Symbol ... }
```

**Brush** предложение, определяющее стиль заливки

**Font** предложение, определяющее стиль текста

**Pen** предложение, определяющее стиль линии

**Symbol** предложение, определяющее стиль символа

**BorderPen** использует предложение Pen, определяющее стиль границы региона

**LinePen** использует предложение Pen определяющее стиль линии

### Описание

Оператор **Set Style** изменяет текущие стили Pen, Brush, Symbol или Font.

Предложение **Pen** устанавливает оба стиля - для линий и для границ региона. Чтобы установить их отдельно, используйте предложение **LinePen** для установки стиля линий и предложение **BorderPen** для установки стиля границы региона.

Когда пользователь рисует новый графический объект в окне Карты или Отчета, MapInfo Professional создает объект используя текущие стили Font, Pen, Brush и/или Symbol.

Более подробная информация о параметрах Pen, Brush, Symbol и Font обсуждается в описаниях предложений Pen, Brush, Font и Symbol.

### Пример

Пример изменения стилей Brush, Symbol и Font:

```
Include "mapbasic.def"
Set Style Brush MakeBrush(64, CYAN, BLUE)
Set Style Symbol MakeSymbol( 9, BLUE, 14)
Set Style Font MakeFont("Helv", 1, 14, BLACK,WHITE)
```

Пример изменения стиля Pen, линия и граница региона красного цвета.

```
Include "mapbasic.def"
Set Style Pen MakePen(3, 9, RED)
```

Пример изменения стиля LinePen и BorderPen где линия красная а граница региона зеленая.

```
Include "mapbasic.def"
Set Style LinePen MakePen(6, 77, RED)
Set Style BorderPen MakePen(6, 77, GREEN)
```

## Оператор Set Style

---

### Смотрите также

`CurrentBrush()`, `CurrentFont()`, `CurrentPen()`, `CurrentSymbol()`, `MakeBrush()`, `MakeFont()`,  
`MakePen()`, `MakeSymbol()`, `RGB()`, `LinePen()`, `BorderPen()`



## Оператор Set Table

### Назначение:

Изменяет некоторые режимы работы с открытой таблицей.

### Синтаксис:

```
Set Table tablename
[ FastEdit { On | Off } ]
[ Undo { On | Off } ]
[ ReadOnly ]
[ Seamless { On | Off } [ Preserve ] ]
[ UserMap { On | Off } ]
[ UserBrowse { On | Off } ]
[ UserClose { On | Off } ]
[ UserEdit { On | Off } ]
[ UserRemoveMap { On | Off } ]
[ UserDisplayMap { On | Off } ]
```

### Описание:

Оператор **Set Table** устанавливает, можно ли изменять данные таблицы и как это сделать. Оператор может задать открытой таблице режим "только чтение", и пользователь не сможет внести в эту таблицу никакие изменения. Оператор может активизировать и выключать специальный режим редактирования, который выключает механизм защиты исходных данных таблицы ради ускорения выполнения действий правки.

### Режим FastEdit

Обычно, всякий раз, когда таблица редактируется (независимо, кем – пользователем или прикладной программой), то MapInfo не записывает изменения непосредственно в таблицу. Вместо этого MapInfo помещает информацию об изменениях во временный файл, который называется файлом транзакций. Записывая изменения в файл транзакций вместо того, чтобы вносить их непосредственно в таблицу, MapInfo дает возможность пользователю удалить эти изменения (например, командой **ФАЙЛ > ВОССТАНОВИТЬ** в MapInfo).

Если Вы выполнили оператор **Set Table** с предложением **FastEdit On**, то MapInfo будет записывать изменения непосредственно в таблицу, минуя файл транзакций. Операции правки таблицы в таком режиме будут производиться быстрее.

Пока включен режим **FastEdit**, таблица изменяется немедленно и не требуется выполнение оператора **Commit** для фиксации изменений на диске. Восстановить исходное состояние таблицы командами **ФАЙЛ > ЗАКРЫТЬ ТАБЛИЦУ** и **ФАЙЛ > ВОССТАНОВИТЬ** нельзя.

Режим **FastEdit** может устанавливаться для нормальных, базовых таблиц. Вы не можете установить этот режим для временных таблиц, таких как ЗАПРОС1. Вы не можете установить режим редактирования **FastEdit** для таблиц, которые уже имеют несохраненные на диск изменения.

**Внимание:** Пока открытая таблица редактируется в режиме **FastEdit**, другие пользователи сети не могут ее открыть. После того, как Вы выполнили все изменения в режиме **FastEdit**, выполните оператор **Commit** или **Rollback** для установки состояния таблицы, чтобы она была доступна другим пользователям сети.

### Режим Только–чтение

Ключевое слово **ReadOnly** в операторе включает режим "только чтение" для таблицы *tablename*, так что пользователь не сможет ее изменять вплоть до конца сеанса в MapInfo. Оператор **Set Table** не может выключить режим "только чтение". Этот режим Вы также можете установить для таблицы при ее открытии оператором **Open Table**.

### Режим отмены

Предложение **Undo On** устанавливает режим работы с таблицей, при котором MapInfo запоминает всю информацию об изменениях, позволяя пользователю применять команду ПРАВКА > ОТМЕНИТЬ. Если Вы использовали в операторе **Set Table** предложение **Undo Off**, то MapInfo не запоминает информацию о последних изменениях в таблице. В последнем режиме экономятся ресурсы Вашего компьютера, а операции редактирования таблицы выполняются существенно быстрее.

### Управление сшитыми таблицам

MapInfo 4.0 поддерживает новый тип таблиц – сшитые таблицы. В сшитой таблице группируются несколько таблиц в единое целое. Концепция сшитых таблиц подробно описана в документации MapInfo.

Предложение **Seamless** устанавливает или отменяет атрибут сшитости для таблицы. Режим **Seamless Off** открывает таблицу, входящую в группу сшитых, для редактирования. Режим **Seamless On** восстанавливает атрибут сшитости. Ключевое слово **Preserve** сохраняет режим; то есть MapInfo записывает режим в таблицу. Без слова **Preserve** смена режима действует только до конца сеанса работы.

### Защита таблиц от доступа пользователя

Предложения **User...** позволяют выборочно запрещать пользователю применять определенные операции к таблице. Это полезно, если Вы хотите запретить пользователю открывать, закрывать или изменять определенные таблицы или окна. Эти предложения ограждают таблицы только от действий пользователя, но не от операторов MapBasic-программ. **Внимание:** Эти предложения не действуют на Косметический слой.

#### Пример

#### Эффект

UserMap Off	Таблица не появится в окне диалоге "Новое окно Карты" и "Добавить слой".
UserBrowse Off	Таблица не появится в окне диалоге "Новое окно Списка".
UserClose Off	Таблица не появится в окне диалоге "Заккрыть таблицу".
UserEdit Off	Пользователь не может редактировать таблицу: окна Списка и Информации не редактируются и соответствующий слой Карты невозможно сделать изменяемым.
UserRemoveMap Off	Когда эта таблица появляется в диалоге "Управление слоями", кнопка "Удалить" для нее неактивна.

UserDisplayMap Off

Когда эта таблица появляется в диалоге “Управление слоями”, флажок видимости для нее сброшен и отключен.

### Пример:

Следующий пример не допускает таблицу World в диалог закрытия таблиц.

```
Set Table World UserClose Off
```

### Смотрите также:

[TableInfo\( \)](#)

### Оператор Set Target

#### Назначение:

Назначает изменяемый объект или объекты или отменяет назначение.

#### Синтаксис:

**Set Target { On | Off }**

#### Описание:

Оператор **Set Target** управляет выбором изменяемого объекта на Карте. Действие оператора аналогично действию команд в MapInfo ОБЪЕКТЫ > ВЫБРАТЬ ИЗМЕНЯЕМЫЙ ОБЪЕКТ и ОБЪЕКТЫ > ОСВОБОДИТЬ ИЗМЕНЯЕМЫЙ ОБЪЕКТ. Некоторые операции в MapInfo предваряются выбором объектов, которые надо изменить. Например, Вы должны сначала назначить изменяемым объект перед тем, как выполнить оператор **Objects Split**. Правила назначения изменяемого объекта описаны в документации MapInfo.

Оператор **Set Target On** соответствует команде ОБЪЕКТЫ > ВЫБРАТЬ ИЗМЕНЯЕМЫЙ ОБЪЕКТ. После выполнения оператора или команды объекты, выбранные на данный момент, становятся изменяемыми. Если не выбрано ни одного объекта, оператор генерирует ошибку.

Оператор **Set Target Off** соответствует команде ОБЪЕКТЫ > ОСВОБОДИТЬ ИЗМЕНЯЕМЫЙ ОБЪЕКТ.

#### Смотрите также:

**Object Combine, Objects Erase, Objects Intersect, Objects Overlay, Objects Split**

## Оператор Set Window

### Назначение:

Изменяет состояние, размер и положение окна на экране, управляет принтером, размером бумаги и полями.

### Синтаксис:

```
Set Window window_id
  [ Position ( x , y ) [ Units paper_units ] ]
  [ Width win_width [ Units paper_units ] ]
  [ Height win_height [ Units paper_units ] ]
  [ Font ... ]
  [ Min | Max | Restore ]
  [ Front ]
  [ Title { new_title | Default } ]
  [ Help [ { File help_file | File Default | Off } [ Permanent ] ]
    [ Contents ] [ ID context_ID ] [ { Show | Hide } ] ]
  [ Printer { Default | Name printer_name }
    [ Orientation { Portrait | Landscape } ]
    [ Copies number ]
    [ Papersize number ]
    [ Border { On | Off } ]
    [ TrueColor { On | Off } ]
    [ Dither { Halftone | ErrorDiffusion } ]
    [ Method { Device | Emf } ]
    [ Transparency
      [ Raster { Device | ROP } ]
      [ Vector { Device | ROP } ] ]
    [ Margins
      [ Left d1 ]
      [ Right d2 ]
      [ Top d3 ]
      [ Bottom d4 ]
      Units <units> ] ]
    [ Export { Default |
      [ Border { On | Off } ]
      [ TrueColor { On | Off } ]
      [ Dither { Halftone | ErrorDiffusion } ]
      [ Transparency
        [ Raster { Device | ROP } ]
        [ Vector { Device | ROP } ] ]
    ] ]
  [ ScrollBars { On | Off } ]
  [ Autoscroll { On | Off } ]
  [ Parent HWND ]
  [ ReadOnly | Default Access ]
  [ Table table_name Rec record_number ]
  [ Show | Hide ]
  [ Smart Pan { On | Off } ]
  [ SysMenuClose { On | Off } ]
  [ Snap [ Mode { On | Off } ] [ Threshold { pixel_tolerance | Default } ] ]
```

*window\_id* целочисленный идентификатор окна или имя специального (например, **Statistics**)  
*x* расстояние от верхнего края рабочего поля окна MapInfo до верхнего края перемещаемого окна

*y* расстояние от левого края рабочего поля в окне MapInfo до левого края перемещаемого окна

*paper\_units* строка с именем единицы измерений на экране (например, "cm" для сантиметров)

Предложение **Font** определяет стиль текста.

*win\_width* ширина окна

*win\_height* высота окна

*new\_title* строка, задающая новый заголовок окна

*help\_file* имя файла Справочника (например, в Windows "FILENAME.HLP")

*context\_ID* целочисленный идентификатор контекста Справочника для задания раздела

*printer\_name* имя принтера. Принтер может быть локальным или сетевым.

*number* is the number of copies of a print job that should be sent to the printer.

*HWND* целочисленный номер окна. Окно с номером *HWND* станет порождающим окном по отношению к окну Легенды, Статистики, Информации, Линейки или Сообщений.

*table\_name* имя открытой таблицы для показа в окне сообщений

*record\_number* целое число типа Integer: значение от 1 и больше для показа определенной записи в окне Информации или 0 для показа сообщения "Нет записей"

**Printer** будет указывать окно, предназначенное для печати.

**Export** будет указывать окно, предназначенное для экспорта.

**Default** будут использоваться стандартные настройки печати/экспорта.

**Name printer\_name** определяет имя используемого принтера.

**Orientation Portrait** (книжная) ориентирует бумагу для печати в книжной ориентации.

**Orientation Landscape** (альбомная) ориентирует бумагу для печати в альбомной ориентации.

**Copies number** указывает число копий для печати.

**Papersize number** информация о размере бумаги для данного окна. Эти числа(номера) универсальные для всех принтеров под Windows. Например, 1 соответствует размеру Letter, 5 соответствует размеру Legal. Этот номер может быть найден в файле MapBasic под названием PaperSize.def. Некоторые драйверы принтера (например, крупные плоттеры) могут использовать собственную нумерацию для идентификации размера бумаги, отличающуюся от той, которая в файле "PaperSize.def". Учтите это при работе с плоттерами.

**Border** определяет будет ли отображаться черная рамка вокруг окна при печати и экспорте.

**Truecolor** определяет, будет ли создан 24-bit true color вывод, если это вообще возможно. Если truecolor отключен, вывод будет с 256 цветами.

**Dither** определяет какой метод растеризации будет использован, если надо отконвертировать 24-битное изображение в 256 цветов. Этот раздел используется при выводе растров и гридов. Растеризация произойдет, если truecolor отключен или если выводное устройство не может поддерживать 24-битные цвета.

**Method** это новое ключевое слово и определяет, будет ли печать проведена непосредственной передачей файла на принтер, или MapInfo создаст промежуточный Windows Enhanced

Metafile и только потом пошлет его на принтер. Ранние версии MapInfo Professional всегда посылали файл прямо на устройство. Новый метод позволяет печатать карты с растрами, чего не было раньше.

Transparency RasterInternal удалено в версии 7.0; таким образом, можно работать с данными предыдущих версий баз сообщения об ошибке.

**Transparency Raster** определяет, как полупрозрачные пиксели будут отображаться. Выберите Device или ROP в зависимости от драйвера принтера или формата экспортного файла. Сначала попробуйте оба способа, сравните и сделайте потом выбор.

**Transparency Raster ROP** обозначает "Использовать метод ROP для отображения полупрозрачного растра". Для этой настройки можно воспользоваться командой (РЕЖИМ > ПАРАМЕТРЫ ВЫВОДА, ФАЙЛ > ПЕЧАТЬ > ДОПОЛНИТЕЛЬНО и ФАЙЛ > ЭКСПОРТ ОКНА > ДОПОЛНИТЕЛЬНО). Если выбран ROP, то полупрозрачное изображение прорисовывается с использованием растровой операции (ROP), обрабатывающей полупрозрачные пиксели. Этот метод используется для обработки полупрозрачного изображения на экране; однако он не всегда хорошо работает при печати. Проведите эксперимент и посмотрите, как драйвер принтера обрабатывает методом ROP. Если Вы экспортируете изображение, используя команду ЭКСПОРТ ОКНА, то лучше выбрать формат метафайла (EMF или WMF). Использование метода ROP допускает отображение любых данных под полупрозрачным слоем в их исходной форме. Например, векторные данные, находящиеся под полупрозрачными пикселями не будут растеризованы. В метафайлах метод ROP не будет прорисовывать любые данные в области растровых пикселей, также возможно отображение базового изображения.

**Transparency Raster Device** MapInfo не будет осуществлять специальные преобразования при выводе растров и гридов, содержащих прозрачность. Изображение генерируется тем же методом, что применяется для отображения на экране. Могут возникнуть некоторые проблемы при генерации вывода.

**Transparency Vector Internal** MapInfo будет делать специальную обработку когда выводится прозрачная заливки или прозрачные растровые символы.

**Transparency Vector Device** MapInfo будет делать специальную обработку когда выводится прозрачная заливки или прозрачные растровые символы. Могут возникнуть некоторые проблемы при генерации вывода.

**Margins** Пользователь может установить поля для принтера в виде вещественных значений в требуемых единицах. Эти значения могут быть увеличены драйвером принтера если они меньше, чем физически допустимые поля для данного принтера.

## Описание

Оператор **Set Window** используется для изменения размеров и положения окна, шрифта в окне, заданном параметром *window\_id*.

Значение кода для этого параметра Вы можете получить, используя функции **FrontWindow()** и **WindowID()**. Для использования оператора **Set Window** по отношению к специальным окнам, таким как "Статистика", можно использовать имена окон (например, Statistics) или имена кодов (например, WIN-STATISTICS), определенных в файле стандартных определений MAPBASIC.DEF. Не забудьте включить в свою программу оператор **Include** "MAPBASIC.DEF".

Следующая таблица содержит имена окон и кодов окон, которые можно использовать в качестве параметра *window\_id*.

Имя окна	Описание окна и его код
MapInfo	Окно программы MapInfo. Код: WIN-MAPINFO.
MapBasic	Окно MapBasic. Код: WIN-MAPBASIC.
Help	Окно программы WinHelp (Справка). Код: WIN-HELP.
Statistics	Окно "Статистика". Код: WIN-STATISTICS.
Legend	Окно "Легенда". Код: WIN-LEGEND.
Info	Окно "Информация" (которое открывается при использовании инструмента Информация). Код: WIN-INFO.
Ruler	Окно "Линейка" (которое открывается при использовании инструмента Линейка). Код: WIN-RULER.
Message	Окно "Сообщение" (которое открывается оператором <b>Print</b> ). Код: WIN-MESSAGE.

Дополнительное предложение **Position** задает расположение окна на экране в рабочем наборе MI Pro. Верхний левый угол рабочего набора имеет позицию 0, 0. Дополнительные предложения **Width** и **Height** задают размер окна. Позиция окна и его размеры задаются в бумажных единицах "in" (дюймы) или "cm" (сантиметры). MapBasic стандартно использует дюймы, изменить эти единицы можно оператором **Set Paper Units**. Оператор **Set Window** может изменить текущие единицы измерения, для этого надо включить дополнительное подпредложение **Units** в предложения **Position**, **Width** и/или **Height**.

Если оператор включает в себя дополнительное слово **Max**, то окно будет maximized (это действует на весь рабочий набор). Если оператор включает в себя дополнительное ключевое слово **Min**, то окно будет minimized (окно исчезнет, останется только иконка в нижней части экрана). Если окно уже minimized или maximized, и если оператор включает добавочное ключевое слово **Restore**, то окно будет восстановлено в прежнем виде.

Если оператор включает в себя дополнительное ключевое слово **Front**, то MapBasic сделает окно активным; также будет установлен фокус окна. Окно станет активным сразу после кликанья на заголовке окна.

Оператор может всегда указывать предложение **Position** или предложение **Front**, независимо от типа окна. Таким образом, некоторые предложения в операторе **Set Window** применяются только к определенным типам окон. Например, окно Линейка не может изменять свой размер, maximized или minimized.

Для смены заголовка окна включите дополнительное предложение **Title**. Заголовок окна приложения (главный заголовок "MapInfo") не может быть изменен иначе, чем при запуске runtime MI Pro.



Предложение SysMenuClose позволяет отменить команду Заккрыть (Close) в системном меню (меню, появляющемся при щелчке в верхнем левом углу окна). Отключение команды Заккрыть действует только на пользовательский интерфейс; программы MapBasic могут отключать команду Заккрыть оператором Close Window. Следующий пример отключения команды Заккрыть для активного окна:

```
Set Window FrontWindow() SysMenuClose Off
```

### Синтаксис предложения Help

Для управления окном Справки, задайте ключевое слово **Help** вместо целочисленного аргумента *window\_id*. Например, следующий оператор покажет раздел 23 из пользовательского файла Справки:

```
Set Window Help File "custom.hlp" ID 23
```

Предложение **File** *help\_file* показывает, какой файл справки активен. В Windows, это действие автоматически показывает окно справки (если Вы не включили слово **Hide**). Если задать **File Default**, то MI Pro будет использовать стандартную справку MapInfo Professional, но не прорисует на экране файл справки. MI Pro имеет только один установленный файл справки, который применяется ко всем запущенным приложениям MapBasic. Если одно предложение устанавливает текущий файл справки, то и другие приложения могут им пользоваться.

Предложение **Off** отключает справку MI Pro, так что при нажатии F1 в MI Pro эффекта не будет. Используйте предложение **Off** если **Вы интегрируете функциональность MI Pro** в другое приложение (например, в программу Visual Basic), если Вы хотите, чтобы пользователь не видел справки MI Pro. (Справка MI Pro содержит ссылки к позициям меню MI Pro, которые могут не иметь действия в программе Visual Basic.)

Предложение **Permanent** заставляет MapInfo всегда пользоваться Справочником *help\_file*, даже если пользователь нажал F1 в диалоге MapInfo. (В среде Windows если предложения **Permanent** нет, то MapInfo обращается к стандартному файлу Справки MAPINFOW.HLP как только пользователь нажмет F1 в диалоге MapInfo.) Эта установка действует до конца сеанса MapInfo или до первого оператора **Set Window Help File**.

Чтобы сразу открыть Справочник на нужном месте, задавайте слова **Contents** (для показа оглавления Справки) или **ID** (для показа нужной информации).

В состав пакета MapBasic не входят средства изготовления Справочников. Более подробно Справочная система описана в *Руководстве пользователя MapBasic*.

### Синтаксис для окон Карт и Отчетов

Предложение **ScrollBars** применимо только к окнам Карт и управляет показом строки (полосы) прокрутки.

Предложение **Autoscroll** применимо окнам Карт и Отчетов. По умолчанию, режим автоматической прокрутки действует в Картах и Отчетах, т.е. при выполнении операции с нажатой кнопкой мыши в окне Карты и Отчета, содержимое окна автоматически сдвигается вслед за мышью при приближении ее указателя к краю окна. Чтобы отключить автоматическую прокрутку, задайте **Autoscroll Off**. Функция **WindowInfo( )** поможет определить, в каких окнах действует режим автоматической прокрутки.

**Smart Pan** изменяет статус операции сдвига в окне. Когда **Smart Pan** включен для окна Карты или Отчета, то при сдвиге и прокрутке используется отключение растровой прорисовки для уменьшения бликов. Стандартное состояние для **Smart Pan** это отключенное.

Когда **Smart Pan** активизирован для окна Отчета, то перерисовка действует только при использовании инструмента Сдвиг.

Когда **Smart Pan** активизирован для окна Карты, то эффект будет зависеть от метода перемещения Карты. Инструмент Сдвиг автоматически прорисовывает область окна при использовании самого инструмента. Карта будет перемещаться медленнее, чем при отключенном режиме Smart Pan. Чем сложнее карта, тем медленнее она будет перемещаться. Прокрутка и автопрокрутка действует похоже на инструмент Сдвиг, но скорость прокрутки не влияет на точность сдвига. Когда команда Set Map используется для центрирования или сдвига изображения со включенной командой Smart Redraw, то изменения окна Карты при прорисовки произойдут без бликования.

**Внимание:** Если растровый режим прорисовки изображения на экране отключен, то предложение **Smart Pan** в окне Карты ведет себя также как и в окне Отчета.

### Синтаксис для вспомогательных окон (Легенда, Линейка и т.д.):

Предложение **Parent**, позволяющее задать новое порождающее окно для окон легенды, Статистики, Информации, Линейки или Сообщений, действует только в Windows. Окно с номером *window\_id* становится родителем, подчиненным окну с номером-указателем *HWND*. **Внимание:** переподчинение окна таким способом изменяет значение **ID** для этого окна. Чтобы снова подчинить окно первоначальному “родителю”, MapInfo, задайте ноль в качестве *HWND*.

Предложения **ReadOnly** / **Default Access** применяются только к окну Информации и управляют возможностью изменения данных в нем. **ReadOnly** запрещает редактирование данных. Предложение **Default Access** снимает контроль со стороны MapBasic, и тогда уже действуют запреты или разрешения для самой таблицы. Это работает для главной легенды и картографических легенд, созданных операторами Create Legend или Create Cartographic Legend.

Предложение **Table** позволяет выбирать данные для показа в окне Информации (и только для него). Это предложение форсирует показ окна Информации.

Предложения **Show** и **Hide** управляют показом или скрыванием окон, для которых эта операция существенна (например, для Линейки), но может применяться и для окна MapInfo.

### Управление принтером

По умолчанию, окна распечатываются используя настройки системного принтера. Это может быть принтер Windows или настроенный в MI Pro принтер, в зависимости от установок пользователя. Использование предложения Name в приложении, рабочем наборе или окне MapBasic может изменить принтер для отдельного документа. Некоторые настройки для принтера также могут задаваться некоторыми другими командными предложениями. Таким образом, когда настройки принтера изменяются через пользовательский интерфейс, соответственно генерируются команды MapBasic. Подобные изменения настроек хранятся в рабочих наборах, а отменить их можно командой Set Window Printer Default.

Коды атрибутов, WIN\_INFO\_PRINTER\_NAME, WIN\_INFO\_PRINTER\_ORIENT или WIN\_INFO\_PRINTER\_COPIES, также возвращаются функцией WindowInfo().

### Пример

```
Set Window frontwindow()  
Printer Name "\\Discovery\HP 2500CP"  
Orientation Portrait  
Copies 10
```

Внимание: Чтобы узнать имя принтера, запустите MapInfo Professional, выполните ФАЙЛ>НАСТРОЙКА ПЕЧАТИ. Нажмите кнопку Принтер. Используйте имя принтера из этого диалога.

### Управление радиусом совмещения

Вы можете настроить радиус совмещения, измеряемый в пикселях для данного окна, вернуть стандартный размер радиуса совмещения для этого окна или получить информацию о текущем радиусе совмещения. Можно также включать/отключать режим совмещения для данного окна или запрашивать включен или нет режим совмещения для данного окна.

Настройки радиуса совмещения для обычного окна могут запрашиваться с использованием новых атрибутов параметров в функции WindowInfo(). Теперь режим совмещения может настраиваться для каждого окна Карты или Отчета. Эти настройки сохраняются в рабочем наборе для каждого окна.

### Пример

```
Dim win_id As Integer  
Open Table "world"  
Map From world  
win_id = FrontWindow()  
Set Window win_id Width 5 Height 3
```

### Смотрите также

Map, Browse, Graph, Layout, Set Paper Units

### Функция Sgn( )

#### Назначение:

Распознает знак числа.

#### Синтаксис:

**Sgn**(*num\_expr*)

где

*num\_expr* – численное выражение.

#### Величина, полученная в результате:

-1, 0, или 1, тип Float.

#### Описание:

Функция **Sgn( )** возвращает -1 (минус единица), если число, заданное выражением *num\_expr*, отрицательно, 0, если число равно нулю, и 1 (единица), если число больше нуля.

#### Пример:

```
Dim x As Integer
x = Sgn(-0.5)

' x равно -1
'
```

#### Смотрите также:

**Abs( )**

## Оператор Shade

### Назначение

Создает тематический слой и добавляет его в окно Карты.

### Синтаксис (вариант 1 - способ диапазонов):

```
Shade [ Window window_id ]
{ layer_id | layer_name }
With expr
[ Ignore value_to_ignore ]
Ranges
[ Apply { Color | Size | All } ]
[ Use { Color | Size | All } [ Pen... ] [ Line... ] [ Brush... ] [ Symbol... ] ]
{ [ From Variable float_array
  Style Variable style_array ] |
  minimum : maximum [ Pen... ] [ Line... ] [ Brush... ] [ Symbol... ]
  [ , minimum : maximum [ Pen... ] [ Line... ] [ Brush... ] [ Symbol... ] ... ] }
[ Default [ Pen... ] [ Line... ] [ Brush... ] [ Symbol... ] ]
```

### Синтаксис (вариант 2 - способ отдельных значений):

```
Shade [ Window window_id ]
{ layer_id | layer_name }
With expr
[ Ignore value_to_ignore ]
Values const [ Pen... ] [ Line... ] [ Brush... ] [ Symbol... ]
[ , const [ Pen... ] [ Line... ] [ Brush... ] [ Symbol... ] ... ]
[ Default [ Pen... ] [ Brush... ] [ Symbol... ] ]
```

### Синтаксис (вариант 3 - метод плотности точек):

```
Shade [ Window window_id ]
{ layer_id | layer_name }
With expr
Density dot_value : dot_size
[ Color color ]
```

### Синтаксис (вариант 4 - метод размерных символов):

```
Shade [ Window window_id ]
{ layer_id | layer_name }
With expr
Graduated min_value : symbol_size max_value : symbol_size
Symbol ...
[ Inflect Symbol ... ]
[ Vary Size By { "LOG" | "SQRT" | "CONST" } ]
```

### Синтаксис (вариант 5 - круговые диаграммы):

```
Shade [ Window window_id ]
{ layer_id | layer_name | Selection }
With expr [ , expr ... ]
[ Half ] Pie [ Angle angle ] [ Counter ]
```

```
[ Fixed ] [ Max Size chart_size [ Units unitname ]  
[ At Value max_value [ Vary Size By { "LOG" | "SQRT" | "CONST" } ] ] ]  
[ Border Pen ... ]  
[ Position [ { Left | Right | Center } ] [ { Above | Below | Center } ] ]  
[ Style Brush ... [ , Brush ... ] ]
```

### Синтаксис (вариант 6 - столбчатые диаграммы):

```
Shade [ Window window_id ]  
{ layer_id | layer_name | Selection }  
With expr [ , expr ... ]  
{ Bar [ Normalized ] | Stacked Bar [ Fixed ] }  
[ Max Size chart_size [ Units unitname ]  
[ At Value max_value [ Vary Size By { "LOG" | "SQRT" | "CONST" } ] ] ]  
[ Border Pen ... ]  
[ Frame Brush ... ]  
[ Width value [ Units unitname ] ]  
[ Position [ { Left | Right | Center } ] [ { Above | Below | Center } ] ]  
[ Style Brush ... [ , Brush ... ] ]
```

где

*window\_id* – целочисленный идентификатор окна Карты;

*layer\_id* – идентификатор слоя в окне Карты, от единицы и больше;

*layer\_name* – имя слоя в окне Карты;

*expr* – выражение, задающее условие выделения и раскраски, такое как имя колонки;

*value\_to\_ignore* – величина, которая должна быть проигнорирована, т.е. не будет создано тематического объекта для строки, значение которой равно параметру *value\_to\_ignore*; обычно это 0, если выражение численное, или пустая строка, если выражение строковое;

*float\_array* – численный массив типа Float, полученный оператором **Create Ranges**;

*style\_array* – массив величин типа Pen, Brush или Symbol, полученный оператором **Create Styles**;

*const* – численная или строковая константа или выражение, в котором не используется переменных;

*minimum* – минимальная величина для диапазона;

*maximum* – максимальная величина для диапазона;

*dot\_value* – число, соответствующее одной точке при использовании метода плотности точек;

*dot\_size* – размер одной точки в пикселах при использовании метода плотности точек;

*angle* – начальный угол для круговой диаграммы;

*chart\_size* – число типа Float, представляющее максимальную высоту для круговых диаграмм или столбчатых диаграмм;

*min\_value* и *max\_value* – числа, соответствующие точкам в методе размерных символов;

*symbol\_size* – размер символа в пунктах, используемого в методе размерных символов;

*unitname* – "бумажная" единица измерения (например, "in" – дюйм, "cm" – сантиметр).

Предложение **Pen** начинает стандартное предложение оператора для определения стиля контура для замкнутых объектов. Например, **Pen**(width, pattern, color ).

Предложение **Line** начинает стандартное предложение оператора для определения линии для объектов типа "линия", "полилиния" и "дуга". Предложение **Line** по строению идентично предложению **Pen**, за исключением начального ключевого слова **Line**.

Предложение **Brush** начинает стандартное предложение оператора для определения стиля штриха. Например, **Brush**(pattern, forecolor, backcolor ).

Предложение **Symbol** начинает стандартное предложение оператора для определения стиля символа точечного объекта. Например, **Symbol**(shape, color, size ).

### Описание:

Оператор **Shade** используется для создания тематического слоя в окне Карты. Этот оператор позволяет программе создать тематическую Карту так же, как это может делать пользователь при помощи диалога команды КАРТА > СОЗДАТЬ ТЕМАТИЧЕСКУЮ КАРТУ в MapInfo.

Правила создания тематической Карты читайте в документации MapInfo.

MapInfo может запоминать тип тематической Карты, помещая оператор **Shade** в файл Рабочего Набора, если в нем есть хотя бы одно окно Карты с тематическим слоем. Для примера Вы можете открыть окно Карты, выполнить команду КАРТА > СОЗДАТЬ ТЕМАТИЧЕСКУЮ КАРТУ и сохранить Рабочий Набор (например, под именем THEME.WOR). Теперь откройте файл Рабочего Набора в любом текстовом редакторе и Вы увидите оператор **Shade**, задающий те настройки слоя, которые были заданы в открытом ранее окне Карты. Вы можете копировать этот оператор из Рабочего Набора в свою программу. Если при тематическом картографировании было открыто окно MapBasic, то команда также будет запротоколирована оператором **Shade**.

Параметр *window\_id*, задающий окно Карты, является необязательным. Если он опущен, то MapBasic создаст тематический слой в самом верхнем окне Карты.

Оператор **Shade** должен явно задать слой, на котором будет производиться условное выделение, даже если это единственный слой на Карте. Слой может задаваться номером (*layer\_id*), причем первый номер соответствует самому верхнему неkosметическому слою, слой под первым будет вторым, и т. д. Кроме того, слой может задаваться именем (например, "world").

Способ тематического оформления оператором **Shade** зависит от того, какое предложение следует за выражением *expr*. Предложение **Ranges** включается в оператор для создания Карты диапазонов, предложение **Values** – для Карты отдельных значений, предложение **Density** – для Карты плотности точек, предложение **Graduated** – для Карты размерных символов, а предложения **Pie** и **Bar** – для Карты с круговыми диаграммами или столбчатыми диаграммами.

### Создание тематического слоя методом диапазонов

В первом варианте синтаксиса предложения **From Variable** и **Style Variable** используются для чтения заранее вычисленных значений, определяющих границы диапазонов и стили для их оформления. Значения для массивов переменных, имена которых задаются в этих предложениях, должны быть вычислены до оператора **Shade** с помощью операторов **Create Ranges** и **Create Styles**. Пример использования массивов в операторе **Shade** приводится в описании оператора **Create Ranges**.

Если оператор **Shade** строит тематический слой методом диапазонов (предложение **Ranges**) или отдельных значений (ключевое слово **Values**; смотрите описание ниже), то он может использовать предложение **Default**, позволяющее задать стиль оформления для объектов, которые не подходят ни для одного описания диапазонов или отдельных значений. То есть Вы можете задать единый стиль оформления для оставшихся объектов.

Всего может быть задано от двух до шестнадцати диапазонов. Каждое описание диапазона состоит из пары чисел (*minimum* и *maximum*), разделенных двоеточием и определяющих верхний и нижний предел диапазона, и следующими за ними описаниями стилей оформления объектов, принадлежащих этому диапазону. Если выражение `expr` для записи больше или равно *minimum* и меньше *maximum*, то эта запись принадлежит этому диапазону. Описания диапазонов в предложении **Ranges** разделяются друг от друга запятыми. В следующем примере используются значения из численной колонки "Население" из таблицы RUSSIA. На Карте выделяются три диапазона.

```
Open Table "RUSSIA"
Map From RUSSIA
Shade RUSSIA With Население
  Ranges
    4827000:29280000 Brush (2,0,201326591) ,
    1783000: 4827000 Brush (8,0,16777215) ,
    449000: 1783000 Brush (5,0,16777215)
```

При создании Карты диапазонов нужно задавать стиль раскраски и штриха предложением **Brush()**. Для Карты отдельных значений, задавайте предложения **Symbol()**. Линейные объекты раскрашиваются, используя предложение **Line()**, а не **Pen()**; синтаксис которых одинаков. (В операторе **Shade** предложение **Pen** управляет стилем контура замкнутых объектов.)

С помощью предложения **Apply**, Вы можете выбрать способ, которым будут выделены графические объекты на тематическом слое.

### Предложение Apply

### Эффект

Apply Color	Выделение производится только цветом, а размеры и другие атрибуты, такие как размер символа и тип линии, будут такими же, как на Карте.
Apply Size	Выделение производится только изменением размеров символов и толщиной линии. Точечные объекты при этом не меняют свой рисунок и раскраску, а линейные объекты – тип линий и цвет.
Apply All	Все атрибуты будут меняться при условном выделении – рисунок символа, его размер, тип линии, толщина линии, штриховка и раскраска объектов.

По умолчанию используется режим, который устанавливается предложением **Apply All**.

Предложение **Use** позволяет управлять тем, в каком объеме MapInfo применяет стилизацию объектов при раскраске диапазонов. В следующем примере показан смысл этого предложения. Пусть таблица WorldCap, содержащая точки, раскрашена без участия предложения **Use**.

```
Shade WorldCap With Cap_Pop Ranges
  Apply All
    0 : 300000 Symbol(35,YELLOW,9) ,
    300000 : 900000 Symbol(35,GREEN,18) ,
    900000 :20000000 Symbol(35,BLUE,27)
```



В результате тематическая Карта будет раскрашена звездочками (код 35) в соответствии с указаниями предложений **Symbol( )**: наименьшие объекты будут показаны желтыми звездочками в 9 точек, объекты среднего объема будут показаны зелеными звездочками в 18 точек и большие объекты будут показаны голубыми 27-точечными звездочками.

В следующем примере добавлено предложение **Use Size**.

```
Shade WorldCap With Cap_Pop Ranges
Apply All

Use Size Symbol(34, RED, 24)      ' <<<<< Внимание!

0      : 300000 Symbol(35,YELLOW,9)  ,
300000 : 900000 Symbol(35,GREEN,18)  ,
900000 :20000000 Symbol(35,BLUE,27)
```

Обратите внимание на то, что **Use Size** вносит свой стиль символа (код 34 представляет кружок).

Теперь все символы будут показаны красными кружками; от группы трех последних предложений **Symbol** применяются только размеры (9, 18, 27 точки). MapInfo игнорирует остальные атрибуты (например, YELLOW, GREEN, BLUE). Тематическая карта отобразит красные кружки, поскольку предложение **Use Size Symbol** определяет красные кружки.

Если Вы зададите **Use Color** вместо **Use Size**, MapInfo будет использовать только цвета, заданные в последнем предложении **Symbol**. На Карте будут показаны желтые, зеленые и голубые кружочки размером в 24 точки.

Предложение **Use All** равносильно отсутствию предложения **Use** вообще.

Предложение **Use** действует только в паре с **Apply All** (либо **Apply** отсутствует вообще).

## Создание тематического слоя методом отдельных значений

За ключевым словом **Values** (второй вариант синтаксиса) может следовать от одного до 255 описаний. Каждое состоит из уникального значения (строка или число) и описаний стилей оформления объектов, принадлежащих значению. Если выражение **expr** для записи равно одному из уникальных значений, определенных в операторе **Shade**, то объект, который присоединен к этой записи, будет оформлен соответственно описанию для этого значения. Список описаний разделяется запятыми.

Если оператор **Shade** имеет предложение **Ranges** или **Values**, то он может иметь и предложение **Default**, задающее единый стиль оформления объектов, которые попадают в группу "остальные", т. е. для них не выполняется условие тематического выделения.

В следующем примере таблица территорий Великобритании UK-SALES содержит колонку "Sales-Rep", заполненную фамилиями торговых представителей, действующих на территории Великобритании. Оператор **Shade** закрашивает каждое графство (область) в зависимости от того, какой торговый представитель его курирует. Так, территории, за которые отвечает Боб, будут закрашены одним цветом, а графства, в которых торгует Джон, другим и т.д.

```
Open Table "uk_sales"
Map From uk_sales
```

```
Shade 1 With Proper$(Sales_Rep)
  Ignore ""
  Values
    "Alan" ,
    "Amanda" ,
    "Bob" ,
    "Jan"
```

### Создание тематического слоя методом плотности точек

За ключевым словом **Density** (третий вариант синтаксиса) должно следовать только одно описание *dot\_value :dot\_size*. Раскрашиваемая карта должна содержать области, которые будут заполняться точками. Например, каждая точка соответствует одной тысяче домов.

Для такой карты численное выражение `ехрг` вычисляет значение для каждой области слоя. Карты и делит его на *dot\_value*. Например, если для области значение выражения равно 100, а *dot\_value* было задано равным 5, то MapBasic нарисует 20 точек в этой области.

Параметр *dot\_size* определяет размер иллюстративной точки в пикселах от 1 до 4.

В следующем примере в таблице штатов STATES выделяется количество домовладений из колонки "Num\_HH\_90" в каждом штате методом плотности точек. Каждая точка имеет размер в 4 пиксела и представляет 60 000 домовладений.

```
Open Table "states"
Map From states
Shade states With Num_HH_90
  Density 60000:4
```

### Создание тематического слоя методом размерных символов

За ключевым словом **Graduated** (четвертый вариант синтаксиса) следует пара описаний типа *value :symbol\_size*. Параметр *value* из первого описания соответствует минимальному значению, представляемому минимальным размером символа **symbol\_size**. Параметр *value* из второго описания соответствует максимальному значению, представляемому максимальным размером символа *symbol\_size*. Размер символов промежуточных значений MapInfo подбирает автоматически.

Предложение **Symbol** диктует, какой тип символа выбирается для выделения (кружок, звездочка и т. п.). Если оператор использует предложение **Inflect Symbol**, задающее второй вариант типа символа, то MapInfo обозначает этими символами объекты, имеющие отрицательное значение.

Следующий оператор создает тематический слой, отображающий доходы и убытки. Пункты, приносящие доход, обозначаются зелеными треугольниками, направленными вверх. Пункты, теряющие убытки, обозначаются красными треугольниками, направленными вниз.

```
Shade stores With Net_Profit
  Graduated
  0.0:0 15000:24
  Symbol(36, GREEN, 24)
  Inflect Symbol(37, RED, 24)
  Vary Size By "SQRT"
```

Предложение **Vary Size By** управляет методом, которым размерные символы отражают разницу между значениями, которые они представляют. Если предложение **Vary Size By** не задано, то MapInfo использует метод "SQRT" (квадратный корень), который задает размер символа пропорциональным квадратному корню из выделяемого значения. При действии этого метода, если в одной записи значение в два раза больше соответствующего значения в другой записи, то и площадь, занятая на экране символом для первого значения будет в два раза больше площади символа, представляющего второе значение. Увеличение площади в два раза – это не то, что увеличение размера символа в точках. Увеличение размера символа в точках увеличивает и длину, и ширину; следовательно, площадь увеличивается в квадрате.

## Создание тематического слоя методом круговых диаграмм

Ключевое слово **Pie** (пятый вариант синтаксиса) задает метод построения тематического выделения объектов путем создания для каждого объекта маленького графика типа круговая диаграмма. При этом в предложении **With** через запятую должны быть заданы два или более выражений, значения которых для каждого объекта будут показаны круговой диаграммой. Диаграммы могут иметь форму полного круга, и могут иметь форму половины окружности, если перед словом **Pie** поставить слово **Half**.

Стартовый угол для диаграммы Вы можете задать в предложении **Angle**. По умолчанию – 180 градусов.

Если используется ключевое слово **Counter**, то сектора в диаграмме располагаются от стартового угла против часовой стрелки.

Предложение **Max Size** определяет максимальный размер круговой диаграммы в "бумажных" единицах. Предложение **Fixed** задает рисование круговых диаграмм в кругах одинакового размера.

Следующий оператор задает выделение диаграммами одинакового размера, в четверть дюйма:

```
Shade sales_95 With phone_sales, retail_sales
Pie Fixed
Max Size 0.25 Units "in"
```

Чтобы круговая диаграмма могла устанавливать размер самостоятельно, применяйте вместо слова **Fixed** предложение **At Value**. Например, следующий оператор создает тематическую Карту с изменяющимся размером круговой диаграммы. Если сумма значений в записи равна 85 000, то Круговая диаграмма будет иметь радиус 2 см; записи с меньшими суммами породят меньшие диаграммы.

```
Shade sales_95 With phone_sales, retail_sales
Pie
Max Size 2 Units "cm" At Value 85000
```

Предложение **Vary Size By** можно включить для выбора метода градуировки размера круговых диаграмм. Это предложение описано выше в разделе для размерных символов.

Предложение **Position** задает место относительно объекта, где помещается диаграмма. По умолчанию диаграммы помещаются в центр объекта.

В предложении **Style** можно задать раскраску для секторов диаграммы. Каждое подпредложение **Brush** должно соответствовать выражению из списка в предложении **With**. Если стили не заданы, то MapInfo использует значение типа **Brush**, сохраненное пользователем.

## Оператор Shade

---

Следующий оператор создает тематический слой, в котором круговые диаграммы для двух значений помещаются выше центра объекта.

```
Shade sales_93
  With phone_sales, retail_sales
  Pie Angle 180
  Max Size 0.5 Units "in" At Value 1245000
  Vary Size By "SQRT"
  Border Pen (1, 2, 0)
  Position Center Above
  Style Brush(2, RED, 0), Brush(2, BLUE, 0)
```

### Создание тематического слоя методом столбчатых диаграмм

Ключевое слово **Bar** (шестой вариант синтаксиса) задает метод построения тематического выделения объектов путем создания для каждого объекта маленькой столбчатой диаграммы. При этом в предложении **With** через запятую должны быть заданы два или более выражений, значения которых для каждого объекта будут показаны диаграммы.

Если Вы использовали ключевое слово **Stacked** перед словом **Bar**, то MapInfo размещает столбики стопкой, иначе столбики располагаются в ряд. Если слово **Stacked** отсутствует, то можно задать слово **Normalized** и тогда столбики могут иметь независимые шкалы.

Если задано ключевое слово **Stacked**, то его можно дополнить словом **Fixed**, определяющим, что все стопки будут иметь одинаковый размер. Если слово **Fixed** не задано, то MapInfo задает размер каждой стопки пропорционально сумме отображаемых значений.

Предложение **Frame Brush** задает стиль раскраски фона графика.

Предложение **Position** управляет как ориентацией столбчатых диаграмм, так и ориентацией диаграммы по отношению к центроиду объекта. Если в предложении **Position** заданы слова **Left** или **Right**, то столбцы горизонтальны, в других случаях они вертикальны.

Предложение **Style** содержит перечисленные через запятую стили **Brush**. Каждый из стилей штриховки соответствует выражению из предложения **With**.

В следующем примере создается тематический слой на Карте, и выделение производится столбчатыми диаграммами, расположенными над центроидами объектов.

```
Shade sales_93
  With phone_sales, retail_sales
  Bar
  Max Size 0.4 Units "in" At Value 1245000
  Vary Size By "CONST"
  Border Pen (1, 2, 0)
  Position Center Above
  Style Brush(2, RED, 0), Brush(2, BLUE, 0)
```

### Смотрите также:

[Create Ranges](#), [Create Styles](#), [Map](#), [Set Legend](#), [Set Map](#), [Set Shade](#)

## Функция Sin( )

### Назначение:

Вычисляет синус.

### Синтаксис:

**Sin**(*num\_expr*)

где

*num\_expr* – численное выражение угла в радианах.

### Величина, полученная в результате:

Вещественное число. Величина типа Float.

### Описание:

Функция **Sin( )** возвращает синус числа, полученного в результате вычисления выражения *num\_expr*. Угол должен задаваться в радианах. Диапазон возвращаемого значения находится между единицей и минус единицей включительно.

Для перевода градусов в радианы число необходимо умножить на число DEG-2-RAD. Для обратного конвертирования используется коэффициент RAD-2-DEG. Чтобы Ваша программа могла использовать эти коэффициенты конвертирования, она должна содержать оператор

**Include "MAPBASIC.DEF"**:

### Пример:

```
Include "mapbasic.def"
Dim x, y As Float
x = 30 * DEG-2-RAD
y = Sin(x)
'
' y равен 0.5, поскольку синус от 30 градусов равен 0.5
'
```

### Смотрите также:

**Acos( ), Asin( ), Atn( ), Cos( ), Tan( )**

### Функция Space\$( )

#### Назначение:

Возвращает строку, состоящую из пробелов.

#### Синтаксис:

**Space\$(*num\_expr*)**

где

*num\_expr* – целочисленное выражение.

#### Величина, полученная в результате:

Строка. Величина типа String.

#### Описание:

Функция **Space\$( )** возвращает строку из пробелов. Количество пробелов определяется выражением *num\_expr*.

Если *num\_expr* принимает значение меньше или равное нулю, функция возвратит пустую строку.

#### Пример:

```
Dim filler As String
filler = Space$(7)
' filler будет равна строке "      "
' (7 пробелов)
Note "Здравствуйте," + filler + "господа!"
'Будет показано сообщение "Здравствуйте,      господа!"
```

#### Смотрите также:

**String\$( )**

## Функция Sqr( )

### Назначение:

Вычисляет квадратный корень.

### Синтаксис:

**Sqr**(*num\_expr*)

где

*num\_expr* – численное выражение, результатом которого должно быть положительное число.

### Величина, полученная в результате:

Вещественное число. Величина типа Float.

### Описание:

Функция **Sqr( )** возвращает квадратный корень от числа, полученного в результате вычисления выражения *num\_expr*. Это число должно быть большим или равным нулю.

Извлечение квадратного корня эквивалентно возведению числа в степень 0.5. Соответственно, **Sqr(n)=n ^ 0.5**. Причем надо отметить, что MapBasic быстрее вычисляет функцию, а не возведение в степень 0.5.

### Пример:

```
Dim n As Float
n = Sqr(25)
' n равно 5.
```

## Оператор StatusBar

### Назначение:

Показывает или прячет строку сообщений в рабочем окне MapInfo или выдает в ней сообщение.

### Синтаксис:

```
StatusBar { Show | Hide }  
[ Message message ]  
[ ViewDisplayPopup { On | Off } ]  
[ EditLayerPopup { On | Off } ]
```

где

*message* - сообщение, которое будет показано в строке сообщений.

### Описание:

Оператор **StatusBar** управляет отображением строки сообщений в рабочем окне MapInfo, а также позволяет печатать в ней сообщения из прикладной программы.

Чтобы напечатать сообщение в строке сообщений, используйте предложение **Message**.

**StatusBar Message "Вычисление координат..."**

MapInfo автоматически заменит Ваше сообщение другими, когда пользователь будет перемещать указатель мышки по командам меню или кнопкам инструментальных панелей. Поэтому текст сообщения должен быть лаконичен, что бы пользователь успел его заметить и прочесть за короткий промежуток времени. По этой причине не советуем Вам помещать важные сообщения в строку сообщений. Для этих целей можно воспользоваться окном "Сообщения", открываемым оператором **Print**.

Используйте параметр **ViewDisplayPopup**, позволяющий изменять вид строки сообщений. Если этот параметр установлен на значение "On", то пользователь сможет изменять масштаб и положение курсора, устанавливаемых из строки сообщений.

Использование параметра **EditLayerPopup** позволяет пользователю устанавливать редактируемый слой в окне Карты из строки сообщений. Если этот параметр установлен на значение "On", то пользователь сможет выбирать редактируемый слой из строки сообщений.



## Оператор Stop

### Назначение:

Приостанавливает выполнение прикладной программы для отладки.

### Синтаксис:

**Stop**

### Предупреждение:

Вы не можете использовать оператор **Stop** во внутренних функциях программы. Вы не можете использовать оператор **Stop** в обработчиках элемента диалога, так как отладочный процесс невозможен, пока открыто диалоговое окно.

### Описание:

Оператор **Stop** является отладочным средством. Оператор приостанавливает выполнение программы и передает управление пользователю. В данном случае пользователь – это программист, который отлаживает свою программу.

Если программа была приостановлена при активном окне MapBasic, то в нем выводится сообщение с номером строки программы, в которой был выполнен оператор **Stop**. Если окно MapBasic не было открыто, то оно будет открыто.

Далее пользователь может использовать окно MapBasic для исследования текущего состояния программы. Если набрать:

**? Dim**

в окне MapBasic, то MapInfo выведет список всех используемых локальных переменных в программе. Если набрать:

**? Global**

в окне MapBasic, то MapInfo выведет список всех используемых глобальных переменных в программе.

Задав после знака вопроса имя переменной, пользователь может получить ее текущее значение.

Пользователь сам может присвоить значение какой-нибудь переменной. Формат присвоения следующий:

**? *variable\_name* = *new\_value***

где *variable\_name* – имя локальной или глобальной переменной, и *new\_value* – выражение, представляющее новую величину для этой переменной.

Для продолжения выполнения приостановленной программы надо выполнить команду **ФАЙЛ > ПРОДОЛЖИТЬ ПРОГРАММУ** в MapInfo. Продолжить выполнение программы также можно, введя в окно MapBasic оператор **Continue**.

В режиме остановки (после оператора Stop) MapInfo продолжает держать открытым файл приложения. Поэтому его нельзя перекомпилировать. Поэтому выйдите из режима остановки командой **ФАЙЛ > ПРОДОЛЖИТЬ ПРОГРАММУ**, если хотите перекомпилировать файл.

Оператор **Stop** не может быть использован внутри процедуры-обработчика диалога и внутри процедуры функции.

### Функция Str\$( )

#### Назначение:

Возвращает строковое представление числа, объекта и стиля.

#### Синтаксис:

**Str\$(*expression*)**

где

*expression* – выражение, результатом которого является число или величина типа Date, Pen, Brush, Symbol, Font и Object

#### Величина, полученная в результате:

Строка. Величина типа String.

#### Описание:

Функция **Str\$( )** возвращает строку, которая представляет величину, полученную в результате вычисления выражения *expression*.

Если параметром *expression* является отрицательное число, то результатом будет строка, начинающаяся с символа знака минус (-). Если число положительно, то строка начинается с пробела.

Длина строки, возвращаемая функцией **Str\$( )**, зависит от точности вычисления выражения *expression* и количества цифр справа от запятой. Другими словами, функция **Str\$( )** вернет строку, используя округленную величину. Если Вы хотите управлять преобразованием, воспользуйтесь функций **Format\$( )**.

Если величина в *expression* имеет объектный тип (Object), функция **Str\$( )** вернет одно из значений, представленных в следующей таблице:

Графический объект	Строка в результате
Дуга	"Arc"
Окружность, эллипс	"Ellipse"
Рамка	"Frame"
Прямая линия	"Line"
Точечный объект	"Point"
Полилиния, ломаная	"Polyline"
Регион, область, многоугольник	"Region"
Прямоугольник	"Rectangle"
Скругленный прямоугольник	"Rounded Rectangle"
Текстовый объект	"Text"

Если параметр *expression* задает объект (тип Object) выражением в форме *tablename.obj* (где *tablename* – имя открытой таблицы) и если к текущей строке таблицы не присоединен графический объект, то функция **Str\$( )** вернет пустую строку. **Замечание:** передача неинициализированной объектной переменной приведет к тому, что функция **Str\$( )** вернет ошибку.

Если параметр *expression* задает дату (Date), функция **Str\$( )** вернет строку в формате, установленном в вычислительной системе, в которой выполняется эта прикладная программа. Например, следующее выражение:

```
Str$( NumberToDate(19951231) )
```

может быть равно и “12/31/1995”, и “1995/12/31”, в зависимости от того, какой формат использует пользователь в своем компьютере. Для управления форматом преобразования функцией **Str\$( )** используйте оператор **Set Format**.

Если параметр *expression* задает число, то функция **Str\$( )** будет использовать точку в качестве десятичной точки, даже если в компьютере пользователя используется другой знак. Функция **Str\$( )** не использует разделитель тысяч при конвертировании числа в строку. Если Вы хотите получить число с разделителями тысяч и десятичной точкой такой, какая задана в данной вычислительной платформе, то используйте функцию **FormatNumber\$( )**.

### Пример:

```
Dim s-spelled-out As String, f-profits As Float  
f-profits = 123456  
spelled-out = "Число улиц:" + Str$(f-profits)
```

### Смотрите также:

**Format\$( ), FormatNumber\$( ), Set Format, Val( )**

### Функция String\$( )

#### Назначение:

Строит строку, повторяя символ заданное количество раз.

#### Синтаксис:

**String\$(*num\_expr*, *string\_expr*)**

где

*num\_expr* – положительное целочисленное выражение;

*string\_expr* – строковое выражение.

#### Величина, полученная в результате:

Строка. Величина типа String.

#### Описание:

Функция **String\$( )** возвращает строку, состоящую из первых символов строки, заданной параметром *string\_expr*. Параметр *num\_expr* задает длину будущей строки (в символах).

#### Пример:

```
Dim filler As String
filler = String$(5, "ABCDEFGH")
'
' переменная filler равна строке "AAAAA"
' (5 раз повторяется первая буква)
'
```

#### Смотрите также:

**Space\$( )**

## Функция **StringCompare( )**

### Назначение:

Сравнивает две строки, учитывая различия строчных и прописных символов.

### Синтаксис:

**StringCompare**(*string1*, *string2*)

где

*string1* и *string2* – строковые выражения.

### Величина, полученная в результате:

Короткое целое число -1, 1 или 0. Величина типа SmallInt.

### Описание:

MapBasic, сравнивая строки с использованием операции "=", не учитывает разницу между прописными и строчными буквами. Например, в операторе:

```
if "ABC" = "abc" then
```

сравнение возвращает TRUE (истину).

Функция **StringCompare( )** позволяет делать сравнения строк, рассматривая строчные и прописные буквы как разные. Сравнение строк происходит посимвольно. Как только встречаются два разных символа, выполнение функции прекращается.

Результат	Что это значит
-1	Код символа из первой строки меньше, чем код соответствующего символа из второй строки.
0	Две строки равны.
1	Код символа из первой строки больше, чем код соответствующего символа из второй строки.

### Пример:

```
n = StringCompare("ABC", "abc")
'
' переменная n будет равна -1,
' т. к. ANSI-код "A" меньше "a".
'
```

### Функция `StringCompareIntl( )`

#### Назначение:

Сравнивает две строки, учитывая особенности сортировки для разных языков.

#### Синтаксис:

**`StringCompareIntl(string1, string2)`**

где

*string1* и *string2* – строковые выражения.

#### Величина, полученная в результате:

Короткое целое число -1, 1 или 0. Величина типа `SmallInt`.

#### Описание:

Функция **`StringCompareIntl( )`** позволяет сравнивать две строки, учитывая языковые особенности, заданные в программе Control Panel. В состав сравниваемых строк могут входить символы, используемые в других языках. Сравнение строк происходит посимвольно. Как только встречаются два разных символа, выполнение функции прекращается.

Результат	Что это значит
-1	Код символа из первой строки меньше, чем код соответствующего символа из второй строки.
0	Две строки равны.
1	Код символа из первой строки больше, чем код соответствующего символа из второй строки.

## Функция **StringToDate( )**

### Назначение:

Переводит строку в величину даты.

### Синтаксис:

**StringToDate**(*datestring*)

где

*datestring* – строка, представляющая дату.

### Величина, полученная в результате:

Дата. Величина типа Date.

### Описание:

Функция **StringToDate( )** создает из строки величину типа Date.

MapBasic интерпретирует данные в соответствии с форматами, которые установлены в компьютере, в котором работает программа. Если компьютер использует стандарт США, то дата представляется в формате Месяц/День/Год, но также существуют другие стандарты, принятые в других странах (например, День/Месяц/Год). Другим может быть также знак разделителя (например, точка вместо /).

Для использования стандарта США в функции **StringToDate( )** независимо от установок в компьютере, используйте оператор **Set Format**. **Замечание:** Программы, откомпилированные MapBasic версии 3.0, автоматически используют только формат США, даже если не был использован оператор **Set Format**.

Если используется формат США, то параметр *datestring* должен содержать компоненты числа, месяца и года, разделенные косой чертой (/). Первые два символа задают месяц, значение которого должно находиться в диапазоне от 1 до 12. Два символа второй компоненты задают число, значение которого должно находиться в диапазоне от 1 до 31. Год может задаваться двумя или четырьмя символами. Если год не указан, то принимается значение настоящего года.

**Замечание:** Если Вы не уверены, какой формат будет использован для представления данных в той системе, где будет запускаться Ваша программа, используйте функцию **NumberToDate( )** вместо **StringToDate( )**. На функцию **NumberToDate( )** не влияют установки вычислительной платформы.

### Пример:

```
Dim d_start, d_end As Date

Set Format Date "US"
d_start = StringToDate("12/17/92")
d_end   = StringToDate("01/02/1995")
Set Format Date "Local"
```

### Смотрите также:

**NumberToDate( )**, **Set Format**, **Str\$( )**

### Функция StyleAttr( )

#### Назначение:

Возвращает значение одной из компонент стиля оформления объекта: Pen, Brush, Font, или Symbol.

#### Синтаксис:

**StyleAttr( *style* , *attribute* )**

*style* - величина, выражающая стиль (величина типа Pen, Brush, Font или Symbol);

*attribute* - целночисленный код, управляющий результатом функции

#### Величина, полученная в результате:

Целое число или строка, в зависимости от значения параметра *attribute*.

#### Описание:

Функция **StyleAttr( )** извлекает из величины типа Pen, Brush, Symbol или Font определенную компоненту.

Все типы в MapBasic являются сложносоставными. Например, определение стиля Brush состоит из трех компонентов: pattern, foreground color и background color. При вызове функции **StyleAttr( )**, параметр *attribute* контролирует, какой из атрибутов возвращается.

Параметр *attribute* должен являться одним из кодов, показанных в нижеследующей таблице. Коды в левом столбце (например, PEN\_WIDTH) определены в файле стандартных определений MAPBASIC.DEF.

Если параметр *style* является значением Pen, используйте одно значение из таблицы:

#### Значение *attribute*

BRUSH\_PATTERN

BRUSH\_FORECOLOR

BRUSH\_BACKCOLOR

FONT\_NAME

FONT\_STYLE

FONT\_POINTSIZE

#### Результат StyleAttr( ):

Целое число (Integer), атрибут стиля Brush, задающий номер штриха.

Целое число (Integer), атрибут стиля Brush, задающий RGB-код цвета штриха.

Целое число (Integer), атрибут стиля Brush, задающий RGB-код цвета фона штриха или -1, если фон прозрачный.

Атрибут стиля Font, задающий строку с именем шрифта.

Целое число (Integer) от 0 до 7, атрибут стиля Font (0 = простой, 1 = жирный, и т.д.); см. Предложение Font для более подробной информации.

Целое число (Integer), атрибут стиля Font, задающий размер шрифта в точках.

Замечание: Если текстовый объект принадлежит таблице, а не Отчету, то размер будет равен 0 и высота букв будет определяться текущим



FONT\_FORECOLOR

масштабом Карты.

Целое число (Integer), атрибут стиля Font, задающий RGB-код цвета символов строки.

FONT\_BACKCOLOR

Целое число (Integer), атрибут стиля Font, задающий RGB-код цвета фона строки или -1, если фон прозрачный. Если стиль шрифта включает кайму, то задает RGB-цвет каймы.

**Значение attribute**

**Результат StyleAttr( ):**

PEN\_WIDTH

Целое число (Integer), атрибут стиля Pen, задающий ширину линии в точках.

PEN\_PATTERN

Целое число (Integer), атрибут стиля Pen, задающий номер вида линии.

PEN\_COLOR

Целое число (Integer), атрибут стиля Pen, задающий RGB-код цвета линии.

PEN\_INTERLEAVE

Логическое: TRUE если стиль пересекающихся линий.

PEN\_INDEX

Целое число (Integer), представляющее индекс линии из библиотеки линий.

SYMBOL\_KIND

Целое число (Integer), тип символа: 1 - символ Map Info версии 3.0; 2 - символ шрифта TrueType; 3 - символ из растрового файла.

SYMBOL\_CODE

Целое число (Integer), атрибут стиля Symbol, задающий номер символа. Используется для типа символа версии MapInfo 3.0 и TrueType.

SYMBOL\_COLOR

Целое число (Integer), атрибут стиля Symbol, задающий RGB-код цвета символа.

SYMBOL\_POINTSIZE

Целое число (Integer) от 1 до 48, атрибут стиля Symbol, задающий размер символа в пунктах.

SYMBOL\_FONT\_NAME

Строка (String), имя шрифта TrueType, который используется как библиотека символов.

SYMBOL\_FONT\_STYLE

Целое число (Integer), задающее написание символа TrueType (0 = нормальное написание, 1 = жирное, и т.д.). Смотрите описание предложения Symbol.

SYMBOL\_ANGLE

Действительное число (Float), угол поворота символа TrueType.

SYMBOL\_CUSTOM\_NAME

Строка (String), имя растрового файла.

SYMBOL\_CUSTOM\_STYLE

Целое число (Integer), задающее стиль для растрового символа (0 = нормальное, 1 = показать фон и т.д.). Смотрите предложение Symbol.

## Функция StyleAttr( )

---

### Ошибки:

ERR\_FCN\_ARG\_RANGE, если неправильно значение аргумента.

### Пример :

Воспользуемся функцией **CurrentPen( )** для того, что бы узнать, какой установлен сейчас стиль линии в MapInfo, затем с помощью функции **StyleAttr( )** узнаем толщину линии в пикселах.

```
Include "mapbasic.def"  
Dim cur_width As Integer  
cur_width = StyleAttr(CurrentPen(), PEN_WIDTH)
```

### Смотрите также :

**MakePen( ), MakeBrush( ), MakeFont( ), MakeSymbol( ) Pen( ), Brush( ), Font( ), Symbol( )**

## Оператор Sub...End Sub

### Назначение:

Определяет процедуру, которую можно вызвать из другой процедуры оператором **Call**.

### Синтаксис:

```
Sub subroutine [ ([ByVal] parameter As var_type [, ... ] ) ]  
    statement_list  
End Sub
```

где

*subroutine* – имя процедуры;

*parameter* – имя параметра процедуры;

*var\_type* – стандартный в MapBasic тип переменных (например, Integer) или сложно-определенный тип, заранее объявленный оператором **Type**;

*statement\_list* – список (от 0 и более) операторов процедуры.

### Предупреждение:

Вы не можете использовать оператор **Sub... End Sub** в окне MapBasic.

### Описание:

Оператор **Sub... End Sub** описывает процедуру в MapBasic. Как только sub-процедура определена, к ней можно обращаться из любой другой части программы оператором **Call**.

Каждая процедура, задаваемая оператором **Sub... End Sub**, должна быть заранее объявлена оператором **Declare Sub**.

Каждая подпрограмма может иметь или не иметь параметры, значения которых передаются из процедуры, вызвавшей эту sub-процедуру. В дальнейшем параметры используются процедурой наравне с локальными переменными, имена и типы которых объявляются оператором **Dim**. Параметры для подпрограммы задаются списком через запятую, где каждый параметр определяется следующим образом:

**[ByVal]** *parameter As var\_type*

где *parameter* – имя параметра. Каждое такое определение параметра процедуры должно быть уникально.

Если при определении параметра процедуры ключевое слово **ByVal** не участвует, то это значит, что параметр определен ссылкой ("by reference"). То есть программа, вызывая процедуру, параметры которой заданы ссылкой, должна использовать только имена переменных в качестве параметров вызова. Вы не можете передавать константы или выражения (такие как "Москва") ссылкой. Впоследствии, если sub-процедура изменила значения этих параметров, то также изменятся значения переменных, использовавшихся как параметры вызова в операторе **Call**. Таким образом, параметр, передаваемый ссылкой, позволяет не только доставлять значение из вызывающей процедуры, но и возвращать значение обратно в этом же параметре.

## Оператор Sub...End Sub

---

Если при объявлении использовалось ключевое слово **ByVal**, то параметр процедуры определен для передачи процедуре значением ("by value"). Программа, вызывая процедуру, параметры которой заданы значением, может использовать как имена переменных, так и константы или выражения в качестве параметров вызова. Однако, если изменить значение такого параметра в теле процедуры, новое значение нельзя будет вернуть в вызывающий модуль. Параметр, определенный как "значение", не может передавать массивы, значения переменных сложного типа, созданного оператором **Type** и объявленных в предложении **Alias**.

Sub-процедура может обрабатывать массивы. В списке параметров массив объявляется с помощью пустых скобок, следующих за именем переменной. В следующем примере в процедуре "ListProcessor" объявлен в качестве параметра массив целых величин "items".

```
Sub ListProcessor(items( ) As Integer)
```

При вызове такой процедуры оператором **Call** параметром вызова должно быть имя массива переменных целого типа без круглых скобок.

Следует помнить, что, если в процедуре объявляется локальная переменная с таким же именем, как у существующей уже глобальной переменной, то значение последней недоступно для использования в этой процедуре. Под данным именем в процедуре используется только локальная переменная.

Предложение **Exit Sub** заканчивает работу sub-процедуры.

### Пример:

Процедура **Cube** возводит число в куб (в третью степень) и возвращает результат. Процедура имеет два параметра: первый для возводимого в степень вещественного числа, второй для результата.

```
Declare Sub Main
Declare Sub Cube(ByVal original As Float, cubed As Float)

Sub Main
Dim x, result As Float
Call Cube(2, result)
' переменная result теперь имеет значение: 8 (2 x 2 x 2)
x = 1
Call Cube(x + 2, result)
' переменная result теперь имеет значение: 27 (3 x 3 x 3)
End Sub

Sub Cube (ByVal original As Float,
          cubed As Float)
    cubed = original ^ 3
End Sub
```

### Смотрите также:

**Call, Declare Sub, Dim, Exit Sub, Function... End Function, Global**

## Предложение **Symbol**

### Назначение:

Задание стиля символа для точечного объекта.

### Синтаксис (вариант 1 - версия для символов MapInfo 3.0):

**Symbol**(*shape, color, size* )

где

*shape* – целое число, величина типа Integer, от 31 или больше, задающий символ из стандартного набора MapInfo (значение 31 задает невидимый символ);

*color* – целочисленный код цвета в системе RGB (смотрите описание функции **RGB( )**);

*size* – целое число, величина типа Integer, размер символа в точках от 1 до 48.

### Синтаксис (вариант 2 - версия для символа из шрифта TrueType):

**Symbol**(*shape, color, size, fontname, fontstyle, rotation* )

где

*shape* – целое число, величина типа Integer, от 31 или больше, задающий символ из шрифта TrueType (значение 31 задает невидимый символ);

*color* – целочисленный код цвета в системе RGB (смотрите описание функции **RGB( )**);

*size* – целое число, величина типа Integer, размер символа в точках от 1 до 48;

*fontname* – строка с именем шрифта TrueType (например, “WingDings”);

*fontstyle* – целочисленный код, величина типа Integer, управляющий написанием шрифта;

*rotation* – вещественное число, угол поворота в градусах.

### Синтаксис (вариант 3 - версия для символа из растрового файла):

**Symbol**(*filename, color, size, customstyle* )

*filename* – строка до 31 символа длиной с именем растрового файла (файл должен находиться в каталоге, заданном пользователем);

*color* – целочисленный код цвета в системе RGB (смотрите описание функции **RGB( )**);

*size* – целое число, величина типа Integer, размер символа в точках от 1 до 48;

*customstyle* – целочисленный код типа Integer, управляющий цветом и фоном символа.

### Синтаксис (вариант 4):

**Symbol** *symbol\_expr*

где

*symbol\_expr* – выражение, результат которого есть величина типа Symbol, например, переменная типа Symbol или **MakeSymbol**(*shape, color, size*).

### Описание:

Предложение **Symbol** позволяет задавать стиль символа для точечного объекта. Предложение не является отдельным оператором, но входит в состав многих операторов, работающих с точечными объектами. Например, в операторе **Create Point** предложение определяет стиль символа для нового объекта. Если предложение **Symbol** в этом операторе опущено, то будет использован текущий стиль символа, установленный в среде MapInfo.

Некоторые операторы MapBasic (такие как **Alter Object...Info OBJ\_INFO\_SYMBOL**) используют выражение типа Symbol как параметр без ключевого слова **Symbol**.

### Стандартный набор символов MapInfo 3.0

В следующей таблице показаны символы и соответствующие им коды из стандартного набора, который используется в первом варианте синтаксиса предложения **Symbol**.

31		41	☆	51	✳	61	🛡
32	■	42	△	52	✈	62	🛕
33	◆	43	▽	53	🚩	63	🏠
34	●	44	■	54	🚩	64	✂
35	★	45	▲	55	📄	65	⚡
36	▲	46	●	56	✚	66	🚧
37	▼	47	➡	57	⚓	67	📌
38	□	48	↙	58	⚓		
39	◇	49	+	59	⊙		
40	○	50	×	60	🏠		

Стандартный набор символов состоит из символов, имеющих код от 31 до 67, однако пользователь может изменять и дополнять этот набор, используя приложение SYMBOL, программу, написанную на MapBasic и поставляемую в стандартном пакете MapInfo.

### Символы шрифта TrueType

Если Вы задаете символ из шрифта TrueType, то параметр *fontstyle* управляет написанием символа:

Значение <i>fontstyle</i>	Написание
0	Нормальное
1	Жирное
16	Черная кайма
32	Оттененное
256	Белая кайма

Для задания двух или более стилей написания коды складываются. Например, для того, чтобы получить символ жирного и оттененного написания, параметр *fontstyle* должен быть равен 33. Написание 16 и 256 взаимоисключают друг друга.

## Растровый символ

Если Вы задаете новый символ, то параметр *customstyle* управляет, какими будут цвет и фон символа:

<u>Значение <i>customstyle</i></u>	<u>Стиль символа</u>
0	Не действуют режимы из группы “Эффекты” диалога “Стиль символа”, и символ появляется таким, какой он есть. Все белые пиксеты раstra прозрачны.
1	Действует режим “Добавить фон”; все белые пиксеты раstra непрозрачны.
2	Действует режим “Покрасить одним цветом”; все не белые точки раstra закрашены одним цветом.
3	Установлены оба флажка (действуют оба режима).

## Пример:

Оператор **Set Map** может использовать предложение **Symbol**. В примере назначается стиль символов для показа точечных объектов первого слоя карты в виде кружочков (символ номер 34), закрашенных красным и размером в 18 пунктов.

```
Include "mapbasic.def"
```

```
Set Map  
  Layer 1 Display Global  
  Global Symbol MakeSymbol(34,RED,18)
```

## Смотрите также:

**MakeFontSymbol( ), MakeCustomSymbol( ), MakeSymbol( ), StyleAttr( )**

### Функция **SystemInfo( )**

#### Назначение:

Возвращает информацию об активной операционной среде и версии программы MapInfo.

#### Синтаксис:

**SystemInfo(attribute)**

где

*attribute* – целочисленный код, определяющий тему запроса.

#### Величина, полученная в результате:

Тип результата зависит от значения параметра *attribute*. Величина типа SmallInt, Logical или String

#### Описание:

Функция **SystemInfo( )** возвращает информацию об оперативной среде и версии программы MapInfo, в которых Вы сейчас работаете. Вид информации задает параметр *attribute*. В файле стандартных определений MapBasic MAPBASIC.DEF определены имена кодов, которые Вы можете использовать в качестве параметра функции **SystemInfo( )**. Для использования имен кодов Ваша программа должна иметь в начале оператор **Include "MAPBASIC.DEF"**.

#### Значения *attribute*

#### Результат, полученный **SystemInfo( )**

SYS\_INFO\_APPLICATIONWND

Целое число, уникальный номер Windows HWND, установленный оператором **Set Application Window** (или ноль, если HWND не устанавливался).

SYS-INFO-APPVERSION

Целое число, соответствующее версии языка MapBasic, на котором была написана Ваша программа, и компилятора, умноженное на 100. Например, если версия 3.0, то функция вернет 300.

SYS-INFO-CHARSET

Возвращает имя используемой системы кодов.

SYS\_INFO\_COPYPROTECTED

Логическая величина: “Да” (TRUE), если пользователь запустил защищенную от копирования версию MapInfo.

SYS\_INFO\_DATE\_FORMAT

Строка: “US” или “Local”, соответствующие способу форматирования даты. Смотрите описание оператора **Set Format**.

SYS\_INFO\_DDESTATUS

Целочисленная величина, количество элементов в DDE-очереди на выполнение. Если очередь пуста, то **SystemInfo( )** вернет ноль (если приходящие команды будут выстраиваться в очередь) или -1 (если приходящие команды будут немедленно выполняться).

SYS\_INFO\_DIG\_INSTALLED

Логическая величина: “Да” (TRUE), если дигитайзер и совместимый драйвер установлен.



SYS_INFO_DIG_MODE	Логическая величина: “Да” (TRUE), если включен режим оцифровки.
SYS_INFO_MAPINFOWND	Целое число, представляющее номер Windows HWND главного окна MapInfo. Функция возвращает 0, если программа выполняется не в Windows.
SYS_INFO_MDICLIENTWND	Целое число, представляющее номер Windows HWND для окна MapInfo MDICLIENT. Функция возвращает 0, если программа выполняется не в Windows.
SYS_INFO_MIPLATFORM	Целое число, тип программы MapInfo: MIPLATFORM_WIN16 (16-битная версия Windows) MIPLATFORM_WIN32 (32-битная версия Windows)
SYS_INFO_MIVERSION	Целое число, соответствующее версии программы MapInfo, в которой Вы сейчас работаете, умноженное на 100 (сто). Например, если версия 3.00, то функция вернет 300.
SYS_INFO_NUMBER_FORMAT	Строка: “9,999.9” или “Local”, соответствующие способу форматирования чисел. Смотрите оператор <b>Set Format</b> .
SYS-INFO-PLATFORM	Целочисленный код, определяющий операционную систему. Возвращенное число может быть одним из следующих кодов: PLATFORM-WIN, PLATFORM-MAC, PLATFORM-MOTIF.
SYS_INFO_PRODUCTLEVEL	Целое число, отражающее уровень (product level) MapInfo (200 для MapInfo Professional).
SYS-INFO-RUNTIME	Возвращает логическое “да” (TRUE) если работает runtime-версия MapInfo, и логическое “нет” (FALSE), если иначе.

### Ошибки:

В результате выполнения функции может генерироваться код ошибки:  
ERR-FCN-ARG-RANGE, если неправильно задано значение аргумента.

### Пример:

В зависимости от того, в какой операционной среде выполняется программа (в Windows или нет), будет запущена или нет процедура, использующая DDE-связь.

```
Include "MAPBASIC.DEF"
Declare Sub DDE-Setup
If SystemInfo(SYS-INFO-PLATFORM) = PLATFORM-WIN Then
    Call DDE-Setup
End If
```

### Функция TableInfo( )

#### Назначение:

Возвращает информацию об открытой таблице.

#### Синтаксис:

**TableInfo**(*table\_id*, *attribute*)

где

*table\_id* – либо целочисленный номер таблицы, либо имя таблицы в кавычках, либо 0;

*attribute* – целочисленный код, определяющий тему запроса.

#### Величина, полученная в результате:

Строка, логическое значение, целое или короткое целое число. Величина типа String, Integer, SmallInt или Logical, в зависимости от значения параметра *attribute*.

#### Описание:

Функция **TableInfo( )** используется для получения определенной информации об открытой таблице.

Первый параметр функции определяет, из какой таблицы была затребована информация. Параметр *table\_id* может быть строкой с именем открытой таблицы или числом, равным ее номеру. Если значение параметра *table\_id* равно 0, то функция будет опрашивать таблицу, которая была открыта или создана самой последней. Вы можете использовать функцию с нулевым первым параметром сразу после оператора **Open Table**, в котором предложение **As** не использовалось. Если в момент вызова функции **TableInfo( )** не было открыто ни одной таблицы или все таблицы уже закрыты, то результатом функции будет ошибка.

Второй параметр *attribute* определяет вид информации о данной таблице MapInfo, которая будет получена. Параметр *attribute* должен быть целочисленным кодом (например, TAB\_INFO\_NAME). В следующей таблице в первой колонке приводятся имена используемых кодов, которые установлены при помощи оператора **Define** в файле стандартных определений MapBasic MAPBASIC.DEF.

#### Значения *attribute*

TAB\_INFO\_COORDSYS Clause

TAB\_INFO\_COORDSYS\_MINX,  
TAB\_INFO\_COORDSYS\_MINY,  
TAB\_INFO\_COORDSYS\_MAXX,  
TAB\_INFO\_COORDSYS\_MAXY

#### Результат TableInfo( )

Строка с предложением **CoordSys**, соответствующим проекции таблицы, например “CoordSys Earth Projection 1, 0”. Возвращается пустая строка, если таблица не может иметь графические объекты.

Вещественные величины, минимальная и максимальная координаты по оси X и Y, которые могут быть сохранены в таблице. Если таблица не может иметь объектов, результатом будет ноль.

TAB_INFO_COORDSYS_NAME	Строка с именем проекции, каким она названа в файле MAPINFOW.PRJ (но без суффикса “\p...”). Результатом будет пустая строка, если таблица не может иметь графические объекты или нет соответствия в файле MAPINFOW.PRJ.
TAB_INFO_EDITED	Логическая величина: "Да" (TRUE), если таблица имеет несохраненные на диске изменения.
TAB_INFO_FASTEDIT	Логическая величина: "Да" (TRUE), если для редактирования таблицы включен режим <b>FastEdit</b> , и "Нет" (FALSE), если выключен. (Смотрите описание оператора <b>Set Table</b> , режим <b>FastEdit</b> .)
TAB-INFO-MAPPABLE	Логическая величина: "Да" (TRUE), если записям таблицы можно сопоставлять графические объекты.
TAB_INFO_MAPPABLE_TABLE	Строка, представляющая базовую таблицу, то есть такую, которая содержит графические объекты. Если таблица представляет собой результат объединения, то возвращает имя той, которая содержит графические объекты.
TAB_INFO_MINX, TAB_INFO_MINY, TAB_INFO_MAXX, TAB_INFO_MAXY	Вещественное число, минимальная и максимальная координаты по оси X и Y, координаты углов минимального прямоугольного покрытия всех объектов таблицы.
TAB-INFO-NAME	Строка, имя таблицы.
TAB-INFO-NCOLS	Целое число (тип SmallInt), количество колонок в таблице.
TAB-INFO-NROWS	Целое число (тип Integer), количество строк в таблице.
TAB-INFO-NUM	Целое число (тип SmallInt), номер открытой таблицы.
TAB-INFO-READONLY	Логическая величина: "Да" (TRUE), если таблица открыта в режиме "только чтение".
TAB_INFO_SEAMLESS	Логическая величина: "Да" (TRUE), если для таблицы включен атрибут сшитости.
TAB_INFO_TABFILE	Строка с полным именем файла таблицы, включая DOS-маршрут. Пустая строка возвращается для таблицы запроса.
TAB-INFO-TEMP	Логическая величина: "Да" (TRUE), если таблица является временной (например, ЗАПРОС1).

## Функция **TableInfo( )**

---

TAB-INFO-TYPE	Целое число (тип SmallInt), код, определяющий вид таблицы: TAB-TYPE-BASE, если таблица нормальная; TAB-TYPE-RESULT, если таблица запроса; TAB-TYPE-IMAGE, если таблица растровая; TAB-TYPE-VIEW, если таблица является представлением (view), например, таблица STREET-INFO является представлением; TAB_TYPE_LINKED, если таблица связанная.
TAB-INFO-UNDO	Логическая величина: "Да" (TRUE), если для редактирования таблицы включена система обратных действий ( <b>Undo</b> ), и "Нет" (FALSE), если система выключена оператором <b>Set Table</b> .
TAB_INFO_USERBROWSE	Логическая величина: "Нет" (FALSE), если оператором <b>Set Table</b> выключен режим <b>UserBrowse</b> .
TAB_INFO_USERCLOSE	Логическая величина: "Нет" (FALSE), если оператором <b>Set Table</b> выключен режим <b>UserDisplayMap</b> .
TAB_INFO_USERDISPLAYMAP	Логическая величина: "Нет" (FALSE), если оператором <b>Set Table</b> выключен режим <b>UserDisplayMap</b> .
TAB_INFO_USEREDITABLE	Логическая величина: "Нет" (FALSE), если оператором <b>Set Table</b> выключен режим <b>UserEdit</b> .
TAB_INFO_USERMAP	Логическая величина: "Нет" (FALSE), если оператором <b>Set Table</b> выключен режим <b>UserMap</b> .
TAB_INFO_USERREMOVEMAP	Логическая величина: "Нет" (FALSE), если оператором <b>Set Table</b> выключен режим <b>UserRemoveMap</b> .

### Ошибки:

В результате выполнения функции могут генерироваться коды ошибок:  
ERR-FCN-ARG-RANGE, если неправильно значение аргумента;  
ERR-TABLE-NOT-FOUND, если неправильно задана таблица или ее нет.

### Пример:

```
Include "mapbasic.def"  
Dim i-numcols As SmallInt, L-mappable As Logical  
Open Table "world"  
  
i-numcols = TableInfo("world", TAB-INFO-NCOLS)  
L-mappable = TableInfo("world", TAB-INFO-MAPPABLE)
```

### Смотрите также:

**Open Table, Set Table**

## Функция Tan( )

### Назначение:

Вычисляет тангенс.

### Синтаксис:

**Tan**(*num\_expr*)

где *num\_expr* – численное выражение.

### Величина, полученная в результате:

Вещественное число. Величина типа Float.

### Описание:

Функция **Tan( )** возвращает тангенс от числа, полученного в результате вычисления *num\_expr*. Число *num\_expr* является угловой величиной в радианах.

Для перевода градусов в радианы число необходимо умножить на DEG-2-RAD. Для обратного конвертирования используется коэффициент RAD-2-DEG. Чтобы Ваша программа могла использовать эти коэффициенты конвертирования, она должна содержать оператор **Include** "MAPBASIC.DEF".

### Пример:

```
Include "mapbasic.def"
Dim x, y As Float
x = 45 * DEG-2-RAD
y = Tan(x)
' y равно 1,
' так как тангенс от 45 градусов равен 1
```

### Смотрите также:

**Acos( ), Asin( ), Atn( ), Cos( ), Sin( )**

### Функция TempFileName\$( )

#### Назначение:

Возвращает имя, которое можно использовать при создании временного файла.

#### Синтаксис:

**TempFileName\$(*dir*)**

где

*dir* – строка с маршрутом (диск+каталог), где будет создан временный файл; пустая строка ("" ) задает системный каталог для создания временных файлов.

#### Величина, полученная в результате:

Строка. Величина типа String.

#### Описание:

Функция **TempFileName\$( )** используется, когда Вам необходимо создать временный файл, но Вы не знаете, какое имя ему дать.

Если Вы вызвали функцию **TempFileName\$( )**, MapBasic вернет строку с полным уникальным именем файла. Сама функция **TempFileName\$( )** не создает временный файл. Создать файл Вы можете оператором **Open File**.

Если параметр *dir* задан пустой строкой (""), то полное имя файла будет включать в себя системный каталог для временных файлов, например, "G:\TEMP\~MAP0023.TMP".

При работе в сети возможна ситуация, когда два пользователя одновременно могут создать временный файл на одном и том же месте, с одинаковым именем. Если Вы попытаетесь создать файл с именем и на каталоге, которые Вы получили от функции **TempFileName\$( )**, может возникнуть конфликтная ситуация, потому что файл уже существует, так как другой пользователь сети успел создать его после того, как Ваша программа вызвала функцию **TempFileName\$( )**, и до оператора создания файла. Для снижения вероятности такой конфликтной ситуации выполняйте оператор **Open File** непосредственно сразу после вызова функции **TempFileName\$( )**. Для снижения вероятности сетевых конфликтов создайте обработчик ошибок и поместите оператор **OnError** после оператора **Open File**.

#### Смотрите также:

**FileExists( )**

### Оператор Terminate Application

#### Назначение:

Закрывает другие выполняющиеся программы MapBasic или программы MapBasic, находящиеся в состоянии ожидания.

#### Синтаксис:

**Terminate Application** *app\_name*

где

*app\_name* – строковая величина с именем работающей программы (например, "scalebar.mbx")

#### Описание:

Если прикладная программа создает свои меню, инструментальные панели, то она сама не завершается, а остается загруженной в режиме ожидания событий, которые надо обработать (пользователь выполняет команду, нажимает на кнопки, созданные прикладной программой и т. д.). Чтобы закрыть программу, находящуюся в режиме ожидания, в другой программе используется оператор **Terminate Application**. Например, если во время отладочного процесса Вам необходимо закрыть программу, находящуюся в режиме ожидания, выполните оператор **Terminate Application** в окне MapBasic.

Ваша прикладная программа может запустить другую прикладную программу, используя оператор **Run Application**, она же может и завершить его оператором **Terminate Application**.

Заметим, что оператор **Terminate Application** используется для завершения только других прикладных программ, а для завершения самой программы используется оператор **End Program**.

#### Смотрите также:

**End Program, Run Application**

### Функция `TextSize( )`

#### Назначение

Возвращает в пунктах размер текста в окне.

#### Синтаксис

**`TextSize( window_id , text_obj )`**

*window\_id* целое, идентификатор, определяющий окно Карты или Отчета. Вызовите **`FrontWindow( )`** или **`WindowID( )`** чтобы получить идентификатор окна.

*text\_obj* текстовый объект.

Внимание: Если текстовый объект из окна Карты, идентификатор ID окна должен быть ID окна Карты. Если текстовый объект из окна Отчета, идентификатор ID окна должен быть ID окна Отчета.

#### Возвращаемое значение

Вещественное

#### Описание

Функция **`TextSize()`** возвращает значение текстового объекта в пунктах исходя из текущего масштаба окна. Эта функция соотносится с выборкой текстового объекта и информацию о тексте можно получить выполнив ПРАВКА>ГЕОИНФОРМАЦИЯ или нажав клавишу F7.

#### Пример

Если активное окно это окно Карты, а текстовый объект выбран:

```
print TextSize(FrontWindow(), selection.obj)
```

#### Смотрите также

Font



### Функция Time()

#### Назначение:

Возвращает текущее системное время в строковом формате. Время может быть в 12-и часовом или 24-часовом формате.

#### Синтаксис:

**StringVar**=Time (*Format*)

#### Описание:

**StringVar** - это строковая переменная, возвращающая системное время в формате ЧЧ:ММ:СС.

### Функция `Timer( )`

#### Назначение:

Возвращает число секунд.

#### Синтаксис:

`Timer( )`

#### Величина, полученная в результате:

Целое число. Величина типа `Integer`.

#### Описание:

Функция `Timer( )` возвращает число секунд, прошедших с полуночи первого января 1970 года. Вы можете вызывать функцию `Timer( )` до и после какой-либо операции, чтобы узнать, сколько времени в секундах она выполняется.

#### Пример:

```
Declare Sub Ubi
Dim start, elapsed As Integer
start = Timer( )
Call Ubi
elapsed = Timer( ) - start
'
' Переменная elapsed содержит время выполнения
' подпрограммы Ubi
'
```

## Процедура ToolHandler

### Назначение:

Специальная процедура для обеспечения работы со специальной инструментальной кнопкой (инструмент MapBasic ).

### Синтаксис:

```
Declare Sub ToolHandler  
Sub ToolHandler  
    statement_list  
End Sub
```

где

*statement\_list* – список операторов, выполняющихся при выборе пользователем инструмента MapBasic.

### Описание:

**ToolHandler** – зарезервированное имя для процедуры MapBasic. Процедура работает совместно с инструментом MapBasic.

Одним из простых способов создания своей кнопки на инструментальной панели "Операции", является специальная процедура **ToolHandler**. Однако, представление кнопки, связанной с процедурой **ToolHandler**, имеет некоторые ограничения. Вы не можете использовать произвольную картинку для кнопки или назначить свой режим рисования этой кнопке. Для создания кнопок, не имеющих этих ограничений, используются операторы **Alter ButtonPad** и **Create ButtonPad**.

Если пользователь запустил программу, которая имеет процедуру **ToolHandler**, то на инструментальной панели "Операции" появляется кнопка, на которой изображен знак плюс. Эта кнопка называется инструментом MapBasic. Инструмент может использоваться в активном окне Списка, Карты или Отчета. Как только пользователь указал инструментом MapBasic в область окна Списка, Карты или Отчета, MapBasic автоматически запускает на выполнение процедуру **ToolHandler**.

В процедуре **ToolHandler** может быть использована функция **CommandInfo( )** для распознавания идентификатора окна, на которое было указано. Если инструмент был использован в окне Списка, функция **CommandInfo( )** может также дать информацию о строке и колонке, на которые указал инструмент. Если инструмент был использован в окне Карты, функцию **CommandInfo( )** можно использовать для получения координат указанной точки в текущей системе координат (смотрите оператор **Set CoordSys**). Если инструмент был использован в окне Отчета, функция **CommandInfo( )** поможет Вам получить координаты точки на листе (т. е. расстояния от верхнего и левого края листа), в которой находился указатель мышки, когда Вы нажали на кнопку мыши. Координаты на листе Отчета возвращаются в текущих единицах измерения, которые устанавливаются оператором **Set Paper Units**.

Функция **CommandInfo( )** также может определить, была ли нажата клавиша SHIFT или CTRL или обе вместе, когда пользователь указывал на окно Списка, Карты или Отчета инструментом MapBasic. Это позволит Вам определить по крайней мере четыре варианта действий для разных способов использования инструмента.

## Процедура ToolHandler

---

Выбрать инструмент на инструментальной панели может сама прикладная программа, выполнив оператор:

**Run Menu Command M-TOOLS-MAPBASIC**

Когда пользователь запускает программу, в которой есть процедура **ToolHandler**, программа не завершается после того, как выполнятся все операторы процедуры **Main** и других процедур, вызванных из нее. Программа будет пребывать в режиме ожидания до тех пор, пока не будет выбран инструмент MapBasic и применен в окне Списка, Карты или Отчета. Программа автоматически начнет выполнение процедуры **ToolHandler**. Когда процедура закончит свои действия, прикладная программа вновь переходит в режим ожидания. И так всякий раз при новом применении инструмента.

Для завершения программы в процедуре **ToolHandler** используется оператор **End Program**. При этом полностью освобождается память, занимаемая всей прикладной программой, кнопка инструмента убирается с панели. Поэтому, если Вам снова понадобятся функции, выполняемые инструментом MapBasic, не используйте этот оператор в процедуре **ToolHandler**, и оставьте программу в режиме ожидания до нового выбора инструмента.

В процедуре **ToolHandler**, в зависимости от окна, в котором будет использован инструмент, должен использоваться оператор **Set CoordSys** перед распознаванием координат выбранной инструментом точки в окне. Если инструмент MapBasic используется в окне Списка, то оператор **Set CoordSys** не нужен. Перед распознаванием координат точки в окне Отчета в процедуре **ToolHandler** используется вариант **Set CoordSys Layout**.

Если пользователь указал на точку Карты, и координатная система Карты не совпадает с координатной системой, принятой в прикладной программе, то процедура **ToolHandler** должна выполнить либо оператор **Set CoordSys Earth**, либо **Set CoordSys NonEarth** перед тем, как определять координаты выбранной точки.

### Пример:

Следующий фрагмент программы тестирует инструмент MapBasic. При запуске программы сначала открывается окно сообщений, приглашающее пользователя выбрать инструмент MapBasic. Когда инструмент будет выбран, процедура **ToolHandler** будет выдавать сообщения о координатах каждой выбранной точки в окнах Карт, Списков, Отчетов.

```
Include "MAPBASIC.DEF"
Declare Sub ToolHandler
Note "Инструмент MapBasic готов для проверки."
Sub ToolHandler
    Note "x:" + Round(CommandInfo(CMD-INFO-X), 0.1) + Chr$(13) +
        " y:" + Round(CommandInfo(CMD-INFO-Y), 0.1)
End Sub
```

### Смотрите также:

**CommandInfo( )**

## Функция **TriggerControl( )**

### Назначение:

Возвращает идентификатор элемента диалога, к которому пользователь обращался последним.

### Синтаксис:

**TriggerControl( )**

### Величина, полученная в результате:

Целое число. Величина типа Integer.

### Описание:

Функция **TriggerControl( )** используется только внутри процедуры-обработчика диалога, построенного при помощи оператора **Dialog**. Функция возвращает номер элемента диалога, который был присвоен ему при создании диалога параметром из предложения **ID**.

Каждый элемент диалога может вызывать процедуру-обработчик. Это могут быть разные процедуры, а может быть так, что несколько элементов содержат вызов одного и того же обработчика. В последнем случае имеет смысл определить, какой именно элемент диалога был выбран пользователем.

### Ошибки:

В результате выполнения функции может генерироваться код ошибки:

ERR-INVALID-TRIG-CONTROL, если функция вызвана не при активном диалоге.

### Смотрите также:

**Alter Control**, **Dialog**, **Dialog Preserve**, **Dialog Remove**, **ReadControlValue( )**

### Функция TrueFileName\$( )

#### Назначение:

Возвращает полное имя файла на основе его неполной спецификации.

#### Синтаксис:

**TrueFileName\$(*file\_spec*)**

где

*file\_spec* – строковая величина с неполной спецификацией файла (например, в Windows "C:parcels.tab")

#### Описание:

Функция **TrueFileName\$( )** возвращает полную спецификацию файла (полное имя диска, все каталоги или папки и имя файла), используя неполную.

Например, в DOS следующий файл специфицирован частично (есть имя диска и имя файла, но нет каталогов, образующих маршрут):

**"C:parcels.tab"**

Если текущий каталог на диске C: является \MAPINFO\DATA, то функция **TrueFileName\$("C:parcels.tab")** вернет следующую строку:

**"C:\mapinfo\data\parcels.tab"**

Функцию **TrueFileName\$( )** удобно использовать в программе, которая подсказывает пользователю спецификацию для его файла на диске. Например, в диалоге перед сохранением файла.

Функция **TrueFileName\$( )** не подтверждает наличие файла с таким именем, а только составляет полную спецификацию для файла, такую, какую бы он мог иметь. Для определения существования файла на диске используйте функцию **FileExists( )**.

## Оператор Type

### Назначение:

Создает произвольную сложную структуру типа данных. Сложный тип позже можно будет использовать наравне со стандартными типами в операторах **Dim** и **Global** для объявления переменных.

### Синтаксис:

```
Type type_name  
    element_name As var_type  
    [ ... ]  
End Type
```

где

*type\_name* – имя сложного типа данных;

*element\_name* – имя элемента типа;

*var\_type* – тип данных для элемента.

### Предупреждение:

Все операторы **Type** должны находиться на "глобальном" уровне в тексте программы, то есть за рамками sub-процедур.

Вы не можете использовать оператор **Type** в окне MapBasic.

Вы не можете использовать переменные сложного типа, созданного оператором **Type** как параметр процедуры или функции для пересылки значением ("by-value"). Также нельзя использовать такие переменные для записи значений в файл оператором **Put**.

### Описание:

Оператор **Type** используется для создания нового типа данных, состоящего из элементов стандартных типов или других заранее определенных сложных типов. Использование сложных типов данных позволяет создавать многоуровневые структуры данных, такие как очереди, двоичные деревья, графы. Для обращения к элементу переменной сложного типа надо использовать следующий формат:

*variable\_name.element\_name...*

(имя переменной, затем, через точку, имя элемента первого уровня и т. д. )

В качестве элемента может использоваться другая сложная структура. Элементом также может быть массив, состоящий из элементов как стандартного типа данных, так и элементов сложного типа данных.

**Замечание:** Вы не можете присваивать значение одной переменной сложного типа, заданного оператором **Type**, другой сложной переменной в форме *var\_name = var\_name*.

### Пример:

```
Type Person  
    fullname    As String  
    age         As Integer  
    dateofbirth As Date  
End Type
```

## Оператор Type

---

```
Dim sales_mgr, sales_people(10) As Person

sales_mgr.fullname = "Варвара Петровна Воженова"
sales_people(1).fullname = "Игорь Михайлович Скобелев"
```

### Смотрите также:

[Dim](#), [Global](#), [ReDim](#)



## Функция **UBound( )**

### Назначение:

Возвращает размерность массива.

### Синтаксис:

**UBound(array)**

где

*array* – имя массива переменных.

### Величина, полученная в результате:

Целое число. Величина типа Integer.

### Описание:

Функция **UBound( )** возвращает текущую размерность массива переменных.

В параметре *array* должно быть задано имя массива локальных или глобальных переменных, объявленных ранее оператором **Dim** или **Global**. Размерность массивов может также изменяться по ходу программы (оператор **ReDim**). Функция **UBound( )** возвращает текущее значение размерности на момент вызова функции.

В 16-битной версии Windows массивы MapBasic могут иметь размерность от 0 до 7000 включительно, и, следовательно, результат функции будет находиться в этом диапазоне. В 32-битной версии Windows массивы могут иметь размерность до 32 767 элементов.

### Пример:

```
Dim matrix(10) As Float
Dim depth As Integer
depth = Ubound(matrix)
'
' переменная depth сейчас имеет значение 10
'

ReDim matrix(20)
depth = Ubound(matrix)
'
' переменная depth сейчас имеет значение 20
'
```

### Смотрите также:

**Dim, Global, ReDim**

### Функция UCase\$( )

#### Назначение:

Возвращает строку, в которой все буквы будут заглавными.

#### Синтаксис:

**UCase\$(*string\_expr*)**

где

*string\_expr* – выражение, результат которого есть строка.

#### Величина, полученная в результате:

Строка. Величина типа String.

#### Описание:

Функция **UCase\$( )** возвращает строку, преобразуя все строчные буквы в прописные, в строке, заданной параметром *string\_expr*.

Преобразованию подвергаются только буквы латинского и русского алфавитов. Цифры и другие символы остаются такими же, какими они были в строке *string\_expr*. Например, функция

**UCase\$( "A#12a" )**

возвращает строку "A#12A".

#### Пример:

```
Dim regular, lower-case As String
regular = "ВыШНИЙ Волочек"
lower-case = Ucase$(regular)
'
' Переменная lower-case теперь равна "ВЫШНИЙ ВОЛОЧЕК",
'
```

#### Смотрите также:

**Proper\$( ), LCase\$( )**

## Оператор UnDim

### Назначение:

Распускает переменную.

### Синтаксис:

**UnDim** *variable\_name*

где

*variable\_name* – имя переменной, которая была объявлена в Рабочем Наборе или окне MapBasic.

### Предупреждение:

Оператор **UnDim** не может использоваться в компилированных программах MapBasic. Его использование разрешено только в Рабочем Наборе или в окне MapBasic.

### Описание:

После выполнения оператора Dim, создавшего переменную, Вы можете использовать оператор **UnDim** для освобождения ресурсов, отведенных для этой переменной. Например, Вы ввели в окно MapBasic оператор Dim, объявляющий целочисленную переменную X:

```
Dim X As Integer
```

Но Вам стало необходимо изменить тип переменной. Следующие операторы переопределяют переменную X:

```
UnDim X
```

```
Dim X As Float
```

### Смотрите также:

**Dim, ReDim**

### Функция UnitAbbr\$( )

#### Назначение:

Возвращает сокращенное имя стандартной единицы измерения в MapInfo.

#### Синтаксис:

**UnitAbbr\$(unit\_name)**

где

*unit\_name* – строка, представляющая стандартное имя единицы измерения MapInfo (например, "km")

#### Величина, полученная в результате:

Строка. Величина типа String.

#### Описание:

Параметр *unit\_name* – это строка, представляющая единицу измерения в MapInfo в англоязычном стандарте. Например, "km" (километр) or "sq km" (квадратный километр).

Функция **UnitAbbr\$( )** возвращает сокращенный вариант имени единицы измерения.

Возвращаемая строка зависит от языковой версии MapInfo: там, где в английской и некоторых европейских версиях следующая функция возвратит "sq km", в русской будет возвращено "кв. км".

**UnitAbbr\$("sq km")**

Список имен единиц измерения MapInfo, которые могут быть использованы в качестве аргументов функции **UnitAbbr\$( )**, приведен в описании операторов **Set Distance Units**, **Set Area Units** и **Set Paper Units**.

В качестве параметра *unit\_name* можно также задать "degree" (функция **UnitAbbr\$( )** возвратит "deg" или "град").

#### Смотрите также:

**Set Area Units**, **Set Distance Units**, **Set Paper Units**, **UnitName\$( )**

## Функция **UnitName\$( )**

### Назначение:

Возвращает полное имя стандартной в MapInfo единицы измерения.

### Синтаксис:

**UnitName\$(unit\_name)**

где

*unit\_name* – строковая величина с именем стандартной в MapInfo единицы измерения (например, "km").

### Величина, полученная в результате:

Строка. Величина типа String.

### Описание:

Параметр *unit\_name* должен быть одним из стандартных в MapInfo англоязычных имен единиц измерения, такое как "km" (километр) или "sq km" (квадратный километр).

Функция **UnitName\$( )** возвращает строку с полной версией имени единицы измерения.

Возвращаемая строка зависит от языковой версии MapInfo: там, где в английской и некоторых европейских версиях следующая функция возвратит "square kilometers", в русской будет возвращено "квадратные километры".

**UnitName\$( "sq km" )**

Список имен единиц измерения MapInfo, которые могут быть использованы в качестве аргументов функции **UnitName\$( )**, приведен в описании операторов **Set Distance Units**, **Set Area Units** и **Set Paper Units**.

В качестве параметра *unit\_name* можно также задать "degree" (функция **UnitName\$( )** возвратит "degrees" или "градусы").

### Смотрите также:

**Set Area Units**, **Set Distance Units**, **Set Paper Units**, **UnitAbbr\$( )**

### Оператор Unlink

#### Назначение:

Разрыв связи с таблицей, которая была загружена с удаленной базой данных оператором **Server Link Table**.

#### Синтаксис:

**Unlink** *TableName*

где

*TableName* – имя открытой MapInfo связанной таблицы.

#### Описание:

Оператор устаряняет связи таблицы и удаленной базы данных. Этот оператор не работает, если в таблице есть изменения (другими словами, пользователь должен сначала сохранить или отменить изменения в таблице). Все метаданные, описывающие инструкции связи, удаляются. Поля, которые были помечены как неизменяемые, после разрыва связей становятся изменяемыми. В результате Вы получите обыкновенную базовую таблицу MapInfo.

#### Пример:

**Unlink** "City\_1k"

#### Смотрите также:

**Server Link Table, Commit Table**

## Оператор Update

### Назначение:

Изменяет одну или более строк в таблице.

### Синтаксис:

```
Update table
    Set column = expr [, column = expr, ... ]
    [ Where RowID = idnum ]
```

где

*table* – имя открытой таблицы;

*column* – имя колонки в таблице;

*expr* – выражение для колонки;

*idnum* – номер строки в таблице.

### Описание:

Оператор **Update** изменяет одну или более колонок в таблице. По умолчанию, оператор обновляет все строки таблицы *table*. Если в операторе используется предложение **Where RowID**, обновляются только указанные строки. В предложении **Set** определяются сами изменения в полях заданной строки или строк.

Используя имя **Obj** для специальной колонки графических объектов, присоединенных к строкам таблицы, Вы можете присоединять новые графические объекты к записям. Смотрите третий пример.

### Примеры:

Мы имеем данные о служащих. Каждая запись содержит отдел, в котором работает служащий, и его жалование. Теперь повысим жалование служащим отдела управления продажами, жалование которых было меньше \$20,000, на 7%. Для выбора записей о служащих, которым надо повысить жалование, используем оператор **Select**.

```
Select * From employees
Where department = "отдел-управления-продаж" And salary < 20000
Update Selection
Set salary = salary * 1.07
```

Теперь повысим жалование служащего, данные которого находятся в десятой записи.

```
Update employees
Set salary = salary * 1.07
Where Rowid = 10
```

Создадим точечный объект и присоединим его к первой записи таблицы:

```
Update sites
Set Obj = CreatePoint(x, y)
Where Rowid = 1
```

### Смотрите также:

**Insert**

### Оператор Update Window

#### Назначение:

Форсирует обновление изображения в окне.

#### Синтаксис:

**Update Window** *window\_id*

где

*window\_id* – идентификатор окна.

#### Описание:

Оператор **Update Window** обновляет изображение в одном из окон MapInfo.

В некоторых ситуациях операции в окне не отображаются сразу и увидеть изменения можно только после ближайшего обновления окна. Например, если программа использует оператор **Dialog** и из обработчика элементов диалога производятся изменения в окне Карты, то новое изображение пользователь увидит только после закрытия диалогового окна. Форсировать процесс изображения в этом случае можно с помощью оператора **Update Window**.

#### Смотрите также:

**Set Event Processing**



## Функция Val( )

### Назначение:

Возвращает численную величину, извлеченную из строки.

### Синтаксис:

**Val**(*string\_expr*)

где

*string\_expr* – строковое выражение.

### Величина, полученная в результате:

Вещественное число. Величина типа Float.

### Описание:

Функция **Val( )** возвращает число, выделяя его из строки, определенной выражением *string\_expr*. Считывание числа начинается с начала строки и заканчивается первым нечисленным символом. При этом функция игнорирует пробелы, символы табуляции и новой строки в начале строки *string\_expr*.

Если первый символ строки не является числом, одним из трех символов, описанных выше, точкой, знаком минус или плюс, амперсандом (&), то функция вернет 0. Амперсанд используется для шестнадцатиричных чисел.

**Замечание:** Если строка включает разделитель целой части числа и десятичной, то этот знак должен быть точкой, независимо от того, какой стандарт форматирования чисел используется в компьютере пользователя. Строка также не должна содержать разделители тысяч. Для удаления разделителей тысяч используйте функцию **DeformatNumber\$( )**.

### Пример:

```
Dim f_num As Float
f_num = Val("12 тысяч")
' f_num равно 12 (двенадцати)

f_num = Val("12,345")
' f_num is равно 12 (двенадцати)

f_num = Val("    52 - 62 дома ")
' f_num равно 52 (пятидесяти двум)

f_num = Val("Девятнадцать")
' f_num is равно 0 (нулю)

f_num = Val("&H1A")
' f_num равно 26 (равно шестнадцатиричному 1A)
```

### Смотрите также:

**DeformatNumber\$( )**, **Format\$( )**, **Set Format**, **Str\$( )**

### Функция Weekday( )

#### Назначение:

Возвращает целое число от 1 до 7, соответствующее дню недели.

#### Синтаксис:

**Weekday**(*date\_expr*)

где

*date\_expr* – выражение, результат которого есть дата (величина типа Data)

#### Величина, полученная в результате:

Короткое целое число от 1 до 7 включительно. Величина типа SmallInt.

#### Описание:

Функция **Weekday( )** возвращает номер дня в неделе. Число 1 соответствует Воскресенью. Если, например, день, заданный выражением *date\_expr*, является Вторником, то результатом функции будет 3.

Функция работает с датами, значения которых – не ранее первого января 100 года. Если выражение *date\_expr* принимает значение даты ранее указанного значения, функция

**Weekday( )** вернет ноль.

#### Пример:

```
If Weekday( CurDate( ) ) = 6 Then  
    Note "Сегодня ПЯТНИЦА!"  
End If
```

#### Смотрите также:

**CurDate( ), Day( ), Month( ), Year( )**

## Оператор While...Wend

### Назначение:

Циклически выполняет определенные действия, пока истинно определенное условие.

### Синтаксис:

```
While condition  
    statement_list  
Wend
```

где

*condition* – выражение, управляющее выполнением цикла;

*statement\_list* – группа операторов, выполняющаяся за один проход цикла.

### Предупреждение:

Вы не можете использовать оператор цикла **While... Wend** в окне MapBasic.

### Описание:

Оператор **While... Wend** является одной из конструкций цикла. MapBasic проверяет условие, заданное выражением *condition*. Если условие истинно, MapBasic выполняет операторы *statement\_list*. Далее снова проверяется условие *condition*, и, если оно истинно, все повторяется. Цикл выполняется до тех пор, пока значение *condition* не станет ложным. После этого MapBasic пропустит операторы *statement\_list* и передаст выполнение программы следующему после **Wend** оператору.

Заметим, что конструкция:

```
While condition  
    statement_list  
Wend
```

фактически идентична конструкции:

```
Do While condition  
    statement_list
```

### Loop

Цикл **While... Wend** может быть заменен одной из форм оператора **Do... Loop**. Использование в программе цикла **While... Wend** обуславливается стилистическими приверженностями каждого программиста.

### Пример:

```
Dim psum As Float, i As Integer  
Open Table "world"  
Fetch First From world  
i = 1  
While i <= 10  
    psum = psum + world.population  
    Fetch Next From world  
    i = i + 1  
Wend
```

### Смотрите также:

**Do... Loop, For... Next**

## Процедура WinChangedHandler

### Назначение:

Процедура, автоматически выполняющаяся при перемещении или увеличении/уменьшении изображения в окне Карты, а также при добавлении или удалении слоя.

### Синтаксис:

```
Declare Sub WinChangedHandler  
Sub WinChangedHandler  
  statement_list  
End Sub
```

где

*statement\_list* – список операторов процедуры.

### Описание:

**WinChangedHandler** – зарезервированное имя для процедуры MapBasic. Когда пользователь запускает программу, в которой есть такая процедура, программа не завершается после того, как выполнятся все операторы процедуры Main и других процедур, вызванных из нее. Программа будет пребывать в режиме ожидания до тех пор, пока не произойдет перемещение и масштабирование Карты в окне или изменение размеров самого окна Карты. После этого автоматически начнется выполнение процедуры с именем **WinChangedHandler**. После выполнения процедуры программа вновь переходит в режим ожидания. И так всякий раз до новых изменений на экране.

В процедуре **WinChangedHandler** может быть использована функция **CommandInfo( )** для распознавания идентификатора окна, в котором произошло изменение. Для завершения программы в теле процедуры **WinChangedHandler** используется оператор **End Program**. При этом полностью освобождается память, занимаемая программой, и после следующего изменения в окне Карты процедура **WinChangedHandler** уже не будет вызываться.

Одновременно в состоянии ожидания могут находиться несколько прикладных программ. Поэтому при изменении в окне Карты, автоматически выполняются все процедуры **WinChangedHandler** из этих программ, одна за другой.

В некоторых случаях MapBasic вызывают процедуру **WinChangedHandler** на события, не связанные с изменениями размера окна. Например, рисование нового объекта вызывает процедуру **WinChangedHandler**.

Чтобы закрыть обработчик и удалить его из памяти, применяйте оператор **End Program**.

### Автопрокрутка в окне Карты

MapInfo версии 4.0 автоматически сдвигает изображение в окне Карты, когда пользователь, используя, например, инструмент рисования прямоугольника, растягивая контур будущего объекта, подводит указатель мышки к краю окна. Если действия пользователя привели к автоматическому сдвигу изображения окна, MapInfo вызывает процедуру **WinChangedHandler** после выполнения или отмены действий инструмента.

Например, если Вы использовали инструмент Линейка и рисование каждого сегмента приводит к автопрокрутке, то MapInfo вызовет **WinChangedHandler** тогда, когда Вы закончите измерения двойным щелчком мышки или нажатием клавиши ESC. Если автопрокрутка произошла вследствие применения пользователем инструмента MapBasic, то MapInfo сначала вызовет обработчик инструмента, а затем процедуру **WinChangedHandler**.

MapInfo не будет вызывать процедуру **WinChangedHandler**, если пользователь вернет в предыдущее состояние изображение окна, в котором оно находилось перед автопрокруткой, или нажмет на клавишу ESC.

Автоматический сдвиг изображения Карты можно отключить с помощью оператора **Set Window**.

### Пример:

Пример использования процедуры смотрите в тексте программы OVERVIEW, которая входит в стандартную поставку MapBasic.

### Смотрите также:

**CommandInfo( ), WinClosedHandler**

### Процедура WinClosedHandler

#### Назначение:

Процедура автоматически выполняется при закрытии окна Карты, Списка, Графика, Отчета, Геогруппы или MapBasic.

#### Синтаксис:

```
Declare Sub WinClosedHandler  
Sub WinClosedHandler  
  statement_list  
End Sub
```

где

*statement\_list* – список операторов процедуры.

#### Описание:

**WinClosedHandler** – зарезервированное имя для процедуры MapBasic. Когда пользователь запускает программу, в которой есть такая процедура, программа не завершается после того, как выполнятся все операторы процедуры Main и других процедур, вызванных из нее. Программа будет находиться в режиме ожидания до тех пор, пока пользователь не закроет какое-нибудь окно. Как только это произойдет, программа активизируется, выполняя процедуру с именем **WinClosedHandler**. После выполнения процедуры программа вновь переходит в режим ожидания.

В процедуре **WinChangedHandler** может быть использована функция **CommandInfo(CMD\_INFO\_WIN)** для распознавания идентификатора окна, в котором произошло изменение. Для завершения программы в теле процедуры **WinChangedHandler** используется оператор **End Program**. При этом полностью освобождается память, занимаемая прикладной программой, и после следующего изменения в окне Карты процедура **WinChangedHandler** уже не будет вызываться.

Одновременно в состоянии ожидания могут находиться несколько прикладных программ. Поэтому, как только окно будет закрыто, автоматически выполняются все процедуры **WinClosedHandler** из этих программ, одна за другой.

#### Смотрите также:

**CommandInfo( )**, **EndHandler**, **RemoteMsgHandler**, **SelChangedHandler**, **ToolHandler**, **WinChangedHandler**

## Функция WindowID( )

### Назначение:

Возвращает идентификатор окна, заданного его номером на экране.

### Синтаксис:

**WindowID**(*window\_num*)

где

*window\_num* – величина типа SmallInt, номер окна.

### Величина, полученная в результате:

Целое число. Величина типа Integer.

### Описание:

Функция **WindowID( )** возвращает уникальный номер окна. Некоторые операторы MapBasic, такие, как **Set Map**, используют идентификатор в качестве параметра.

В следующей таблице приводятся возможные способы задания параметра *window\_num*:

#### Значение *window\_num*

#### Результат

Положительное целое число (величина типа Smallint), например, 1, 2, ... *n*

MapInfo возвращает идентификатор документального окна, такого, как Карта, Список. Например, если задана единица, то MapInfo возвращает идентификатор первого документального окна. Заметим, что значение *n* можно получить с помощью функции **NumWindows( )**.

Отрицательное целое число (величина типа Smallint), например, -1, -2, ... -*m*

MapInfo возвращает идентификатор как документального окна, так и другого плавающего окна, такого как Информация. Заметим, что значение *m* можно получить с помощью функции **NumAllWindows( )**. Используя этот синтаксис, Вы можете вызывать функцию **WindowID( )** в цикле для построения списка всех открытых окон.

Ноль (0)

MapInfo возвращает ID-номер окна либо последнего из открытых документов, либо легенды, созданной пользователем, либо инструментальной панели; или ноль, если окна не открывались.

Код окна (например, WIN\_RULER)

Если Вы задали код от 1001 до 1013, то MapInfo вернет идентификатор соответствующего специального окна. Имена кодам назначены в файле стандартных определений MAPBASIC.DEF. Например, код WIN\_RULER, имеющий значение 1007, представляет окно Линейка.

### Ошибки:

ERR-BAD-WINDOW-NUM, если неверно значение аргумента.

### Смотрите также:

**FrontWindow( )**, **NumWindows( )**

### Функция WindowInfo( )

#### Назначение:

Возвращает информацию об открытом окне.

#### Синтаксис:

**WindowInfo**(*window\_id*, *attribute*)

где

*window\_id* – целочисленный идентификатор окна;

*attribute* – целое число, код необходимой информации.

#### Величина, полученная в результате:

Тип результата зависит от значения параметра *attribute*.

#### Описание:

Функция **WindowInfo( )** возвращает определенную информацию об открытом окне.

Параметр *window\_id* задает идентификатор окна. Значения идентификатора можно получить, используя функции **FrontWindow( )** и **WindowID( )**.

Многим кодам, используемых в параметрах, определены имена в файле MAPBASIC.DEF, которые Вы можете использовать в функции **WindowInfo( )**, если в начале программы есть строка **Include "MAPBASIC.DEF"**.

В следующей таблице приводятся возможные способы задания параметра *window\_num*:

#### Значение window\_num

Целочисленный идентификатор

Положительное целое число  
(величина типа Smallint),  
например, 1, 2, ... *n*

Отрицательное целое число  
(величина типа Smallint),  
например, -1, -2, ... -*m*

Ноль (0)

#### Описание

Для задания окна, о котором хотите получить информацию можно использовать его идентификатор, который, в свою очередь, можно получить с помощью функций **FrontWindow( )** или **WindowID( )**.

Функция возвращает информацию о документальном окне, таком, как Карта или Список. Например, если задана единица, то MapInfo возвращает информацию о первом документальном окне. Заметим, что значение *n* можно получить с помощью функции **NumWindows( )**.

Функция возвращает информацию как о документальном окне, так и о другом плавающем окне, таком, как Информация. Заметим, что значение *m* можно получить с помощью функции **NumAllWindows( )**. Используя этот синтаксис, Вы можете вызывать функцию **WindowInfo( )** в цикле для построения списка информации обо всех открытых окнах.

Функция опрашивает последнее из открывавшихся окон. Если окна не открыты, порождается ошибка.



Код окна  
(например, WIN\_RULER)

Если Вы задали код от 1001 до 1013, то MapInfo вернет информацию о соответствующем специальном окне. Имена кодам назначены в файле стандартных определений MAP-BASIC.DEF. Например, код WIN\_RULER, имеющий значение 1007, представляет окно Линейка.

Теперь рассмотрим зависимость результата от значения параметра *attribute*.

### Значение *attribute*

### Результат WindowInfo( *attribute* )

WIN\_INFO\_AUTOSCROLL

Логическая величина: "Да" (TRUE), если режим автоматического сдвига включен. Изменение этого режима делается оператором **Set Window**.

WIN\_INFO\_CLONEWINDOW

Строка с оператором MapBasic, который может быть использован в операторе **Run Command** для дублирования окна. Смотрите описание оператора **Run Command**.

WIN-INFO-HEIGHT

Число типа Float, высота окна в "бумажных" единицах измерения, установленных оператором **Set Paper Units**.

WIN\_INFO\_LEGENDS\_MAP

Целое число: если вы составляете запрос об окне Легенды, открытом оператором **Create Legend**, то результатом будет идентификатор соответствующего окна Карты или Графика. Если окно Легенды стандартно, то результатом будет 0.

WIN-INFO-NAME

Строка с именем окна.

WIN-INFO-OPEN

Логическая величина, определяющая, открыто ли окно (используется для таких окон как "Статистика").

WIN-INFO-STATE

Короткое целое число (тип SmallInt):  
WIN-STATE-NORMAL, если окно раскрыто, но меньше окна MapInfo;  
WIN-STATE-MINIMIZED, если окно свернуто в иконку;  
WIN-STATE-MAXIMIZED, если окно имеет максимальный размер.

WIN\_INFO\_SYSMENUCLOSE

Логическая величина: "Нет" (FALSE), если оператор **Set Window** нейтрализовал команду CLOSE из системного меню окна.

WIN-INFO-TABLE

Строка с именем временной таблицы:  
"Cosmeticn", если определено окно Карты;  
"Layoutn", если определено окно Отчета;  
где *n* – номер окна. Для окон Списка и Графика результатом будет имя таблицы, показанной в окне.

WIN-INFO-TOPMOST

Логическая величина. Если окно активно, значение будет истинно (TRUE).

## Функция WindowInfo( )

---

WIN-INFO-TYPE	Число типа SmallInt, определяющее тип окна (например, WIN_LAYOUT). Смотрите следующую таблицу.
WIN-INFO-WIDTH	Число типа Float, ширина окна в "бумажных" единицах измерения, установленных оператором <b>Set Paper Units</b> .
WIN_INFO_WINDOWID	Целое число, идентификатор окна. Результат такой же, как у функции <b>WindowID( )</b> . Это можно использовать для передачи нуля как <i>window_spec</i> .
WIN_INFO_WND	Целое число. В среде Windows представляет Windows HWND для опрашиваемого окна.
WIN_INFO_WORKSPACE	Строка с операторами MapBasic, с помощью которых запоминается Карта в Рабочем Наборе. Отличается от результата функции с кодом WIN_INFO_CLONEWINDOW тем, что результат включает в себя операторы <b>Open Table</b> и др.
WIN-INFO-X	Число типа Float, расстояние от левого края нашего окна до левого края рабочего поля MapInfo (в "бумажных" единицах).
WIN-INFO-Y	Число типа Float, расстояние от верхнего края нашего окна до верхнего края рабочего поля MapInfo (в "бумажных" единицах).
WIN_INFO_PRINTER_NAME	Возвращает строку с идентификатором принтера (например, <u>\\DISCOVERY\HP4_DEVEL</u> )
WIN_INFO_PRINTER_ORIENT	Возвращает WIN_PRINTER_PORTRAIT или WIN_PRINTER_LANDSCAPE
WIN_INFO_PRINTER_COPIES	Возвращает integer number of copies.
WIN_INFO_SNAPMODE	Возвращает a logical value. TRUE if snap mode is on. FALSE if snap mode is off. The value for WIN_INFO_SNAPMODE is 19.
WIN_INFO_SNAPTHRESHOLD	Возвращает короткое целое, устойчивость в пикселях. Значение для WIN_INFO_SNAPTHRESHOLD это 20.
WIN_INFO_PRINTER_PAPERSIZE	Целая величина. Смотрите в файле Papersize.def file объяснение значений возвращаемой величины.
WIN_INFO_PRINTER_LEFT_MARGIN	Вещественное: левое поле области печати в текущих единицах длины.
WIN_INFO_PRINTER_RIGHT_MARGIN	Вещественное: правое поле области печати в текущих единицах длины.
WIN_INFO_PRINTER_TOP_MARGIN	Вещественное: верхнее поле области печати в текущих единицах длины

WIN_INFO_PRINTER_BOTTOM_MARGIN	Вещественное: нижнее поле области печати в текущих единицах длины.
WIN_INFO_PRINTER_BORDER	Строковая величина: ON если рамка вокруг окна изображается при печати, OFF если нет.
WIN_INFO_PRINTER_TRUECOLOR	Строковая величина: ON если используется 24-битный true color для печати растров и сеток(grid). Это бывает когда изображение 24 битное и принтер поддерживает более 256 йветов, OFF в противном случае.
WIN_INFO_PRINTER_DITHER	Строковая величина: возвращает метод растеризации, который используется если надо конвертировать 24-битное изображение в 256 цветов. Возвращаемые значения HALFTONE и ERRORDIFFUSION. Эта настройка используется при печати растров и сеточных файлов (grid). Растеризация произойдет, если WIN_INFO_PRINTER_TRUECOLOR невозможно или если принтер поддерживает только 256 цветов или менее.
WIN_INFO_PRINTER_METHOD	Строковая величина: возвращает значения DEVICE и EMF.
WIN_INFO_PRINTER_TRANSPARENT	Строковая величина: возвращает значения DEVICE и INTERNAL.
WIN_INFO_PRINTER_TRANSPARENT	Строковая величина: возвращает значения DEVICE и INTERNAL.
WIN_INFO_EXPORT_BORDER	Строковая величина: возвращает значения ON и OFF
WIN_INFO_EXPORT_TRUECOLOR	Строковая величина: возвращает значения ON и OFF.
WIN_INFO_EXPORT_DITHER	Строковая величина: возвращает значения HALFTONE и ERRORDIFFUSION.
WIN_INFO_EXPORT_TRANSPARENT	Строковая величина: возвращает значения DEVICE и INTERNAL

Если Вы используете в качестве параметра *attribute* код WIN-INFO-TYPE, функция **WindowInfo( )** вернет код вида окна. Имена для этого кода перечислены в первой колонке в следующей таблице:

<u>Код в результате</u>	<u>Описание окна</u>
WIN_MAPPER	Окно Карты
WIN_BROWSER	Окно Списка
WIN_LAYOUT	Окно Отчета
WIN_GRAPH	Окно Графика

## Функция WindowInfo( )

---

WIN_HELP	Окно Справочной системы
WIN_MAPBASIC	Окно MapBasic
WIN_MESSAGE	Окно "Сообщение" (вызывается оператором <b>Print</b> ).
WIN_RULER	Окно "Линейка" (вызывается инструментом Линейка)
WIN_INFO	Окно "Информация" (вызывается инструментом Информация)
WIN_LEGEND	Окно "Легенда"
WIN_STATISTICS	Окно "Статистика"
WIN_MAPINFO	Рабочее окно программы MapInfo
WIN_BUTTONPAD	Окно инструментальной панели

Каждое окно Карты создает специальную временную таблицу, которая содержит данные для Косметического слоя Карты. Эти таблицы (имеющие имена "Cosmetic1", "Cosmetic2" и т.д.) пользователь не видит.

Аналогично, окна Отчетов также поддерживают временные скрытые от пользователя таблицы с именами вида "Layout1", "Layout2" и т.д. Вы можете получить эти имена от функции

**WindowInfo( )** используя код WIN-INFO-TABLE.

### Ошибки:

В результате выполнения функции могут генерироваться следующие коды ошибок:

ERR-FCN-ARG-RANGE, если неправильно значение аргумента *attribute*;

ERR-BAD-WINDOW, если неправильно значение аргумента *window\_id*.

### Пример:

В следующем примере открывается окно Статистика, если оно уже не открыто.

```
If Not WindowInfo(WIN_STATISTICS,WIN_INFO_OPEN) Then  
    Open Window WIN_STATISTICS  
End If
```

### Смотрите также:

**Map, Browse, Graph**

## Процедура WinFocusChangedHandler

### Назначение:

Процедура, автоматически выполняющаяся при изменении фокуса окна.

### Синтаксис:

```
Declare Sub WinFocusChangedHandler
Sub WinFocusChangedHandler
    statement_list
End Sub
```

### Описание:

Если загруженная программа имеет процедуру **WinFocusChangedHandler**, то MapInfo автоматически выполняет процедуру, когда меняется фокус окна (фокус имеет активное окно). Фокус может перемещаться между всеми типами окон MapInfo (Списки, Карты и т.п.). Для определения идентификатора окна, которое стало активным, используйте в процедуре обработчика функцию **CommandInfo(CMD\_INFO\_WIN)**.

В процедуре **WinFocusChangedHandler** не может быть использован оператор **Note**, процедура обработчика также не может закрывать или открывать какие-либо окна. Эти ограничения подобны тем, которые имеют и другие обработчики, такие, как **SelChangedHandler**.

Процедура **WinFocusChangedHandler** должна быть как можно короче, чтобы не замедлять работу системы.

### Пример:

Следующий фрагмент текста программы показывает, как можно делать доступной или недоступной команду меню в зависимости от того, активно ли окно Карты или нет.

```
Include "mapbasic.def"
Include "menu.def"
Declare Sub Main
Declare sub WinFocusChangedHandler
Sub Main
    ' Здесь вместо комментариев должны быть операторы,
    ' создающие элемент меню, который может быть доступным,
    ' когда активно окно Карты.
End Sub
Sub WinFocusChangedHandler
    Dim i-win-type As SmallInt
    i-win-type=WindowInfo(CommandInfo(CMD-INFO-WIN),WIN-INFO-TYPE)
    If i-win-type = WIN-MAPPER Then
        ' Здесь вместо комментариев должен быть оператор,
        ' который делает элемент меню доступным.
    Else
        ' Здесь вместо комментариев должен быть оператор,
        ' который делает элемент меню недоступным.
    End If
End Sub
```

### Оператор Write #

#### Назначение:

Запись данных в открытый файл.

#### Синтаксис:

**Write #***file\_num* [, *expr ...* ]

где

*file\_num* – номер файла, который был присвоен ему при открытии;

*expr* – выражение для записи в файл.

#### Описание:

Оператор **Write #** записывает определенные данные в открытый файл. Он должен быть открыт в режиме последовательного доступа оператором **Open File**, который закрепляет за файлом номер, используемый в параметре *file\_num*.

Выражений *expr* может быть несколько, в операторе они должны быть разделены запятыми. В этом случае записанные значения в файле автоматически разделяются запятыми. Строчные значения при записи автоматически снабжаются кавычками. Если список выражений пуст, то записывается пустая строка.

Оператор **Write #** автоматически заключает строковые значения в кавычки при записи в файл.

Чтобы записать текст в файл, не используя кавычек, используйте оператор **Print #**.

Для чтения записей из файла используйте оператор **Input #**.

#### Смотрите также:

**Input #, Open File, Print #**

## Функция Year( )

### Назначение:

Извлекает год из значения даты.

### Синтаксис:

**Year**(*date\_expr*)

где *date\_expr* – выражение типа Date.

### Величина, полученная в результате:

Короткое целое число. Величина типа SmallInt.

### Описание:

Функция **Year( )** извлекает год из даты, заданной выражением *date\_expr*. Например, если дата задана в виде 12/17/95, то функция вернет целое число 1995.

### Примеры:

Пример демонстрирует, как Вы можете использовать функцию для извлечения из даты только компоненты года.

```
If Year( CurDate( ) ) = 1994 Then
    Note "В 1994 году..."
End If
```

Вы можете также использовать функцию **Year( )** в операторе **Select**, формирующем SQL-запрос. В этом примере выбираются строки из таблицы ORDERS, в которых есть данные о заказах, сделанных в декабре 1993.

```
Open Table "orders"
Select * From orders
Where Month(orderdate) = 12 And Year(orderdate) = 1993
```

### Смотрите также:

**CurDate( )**, **Day( )**, **Month( )**, **Weekday( )**, **DateWindow( )**

## Приложение А: Таблица кодов символов

Следующая таблица показывает часть символов Windows Latin 1. Диапазон символов от 32 (пробел) до 126 (тильда) идентичен большинству других наборов символов. Интересны так же символы: 9 табулятор, 10 перевод строки, 12 перевод страницы и 13 возврат каретки.

32	64 @	96 `	128 ■	160	192 À	224 à
33 !	65 A	97 a	129 ■	161 ì	193 Á	225 á
34 "	66 B	98 b	130 ■	162 ç	194 Â	226 â
35 #	67 C	99 c	131 ■	163 £	195 Ã	227 ã
36 \$	68 D	100 d	132 ■	164 ð	196 Ä	228 ä
37 %	69 E	101 e	133 ■	165 ¥	197 Å	229 å
38 &	70 F	102 f	134 ■	166 ì	198 Æ	230 æ
39 '	71 G	103 g	135 ■	167 \$	199 Ç	231 ç
40 (	72 H	104 h	136 ■	168 ~	200 È	232 è
41 )	73 I	105 i	137 ■	169 ©	201 É	233 é
42 *	74 J	106 j	138 ■	170 ®	202 Ê	234 ê
43 +	75 K	107 k	139 ■	171 «	203 Ë	235 ë
44 ,	76 L	108 l	140 ■	172 ~	204 Ì	236 ì
45 -	77 M	109 m	141 ■	173 ~	205 Í	237 í
46 .	78 N	110 n	142 ■	174 ®	206 Î	238 î
47 /	79 O	111 o	143 ■	175 ~	207 Ï	239 ï
48 0	80 P	112 p	144 ■	176 °	208 Ð	240 ð
49 1	81 Q	113 q	145 ´	177 ±	209 Ñ	241 ñ
50 2	82 R	114 r	146 ´	178 ²	210 Ò	242 ò
51 3	83 S	115 s	147 ■	179 ³	211 Ó	243 ó
52 4	84 T	116 t	148 ■	180 ´	212 Ô	244 ô
53 5	85 U	117 u	149 ■	181 µ	213 Õ	245 õ
54 6	86 V	118 v	150 ■	182 ¶	214 Ö	246 ö
55 7	87 W	119 w	151 ■	183 ·	215 ×	247 ÷
56 8	88 X	120 x	152 ■	184 ,	216 Ø	248 ø
57 9	89 Y	121 y	153 ■	185 ´	217 Ù	249 ù
58 :	90 Z	122 z	154 ■	186 ■	218 Ú	250 ú
59 ;	91 [	123 {	155 ■	187 >	219 Û	251 û
60 <	92 \	124	156 ■	188 ¼	220 Ü	252 ü
61 =	93 ]	125 }	157 ■	189 ½	221 Ý	253 ý
62 >	94 ^	126 ~	158 ■	190 ¾	222 Þ	254 þ
63 ?	95 _	127	159 ■	191 ÿ	223 ß	255 ÿ



### Приложение В: Арифметические и географические операторы

Оператор задает определенное арифметическое действие над одной или более величинами или задает отношение, имеющее результат. Операторы могут классифицироваться по типу величин, с которыми они действуют, и по типу результата.

Следующие арифметические операторы работают с числами и имеют численный результат.

Оператор	Действие	Пример
+	сложение	$a + b$
-	вычитание	$a - b$
*	умножение	$a * b$
/	деление	$a / b$
\	деление по модулю (без остатка)	$a \setminus b$
Mod	остаток деления по модулю	$a \text{ Mod } b$
^	возведение в степень	$a ^ b$

Два первых оператора также используются в другом контексте. Плюс используется как знак слияния (конкатенации строк). Вместо плюса можно использовать амперсанд. Минус может означать отрицательное значение.

Оператор	Действие	Пример
+	строковая конкатенация	$a + b$
&	строковая конкатенация	$a \& b$
-	перемена знака числа	$-a$

Операторы сравнения - это операторы, сравнивающие два элемента одного типа с получением логического результата: TRUE or FALSE. Численные данные не могут сравниваться с нечисленными (например, с строкой), но сравнение возможно между величинами типов Integer, SmallInt и Float. Выражения с операторами сравнения часто используются в операторах, управляющих выполнением программы (например, в операторе If...Then).

Оператор	TRUE, если	Пример
=	сложение	$a = b$
<>	вычитание	$a <> b$

## Приложение В: Арифметические и географические операторы

---

<	умножение	$a < b$
>	деление	$a > b$
<=	деление по модулю (без остатка)	$a \leq b$
>=	остаток деления по модулю	$a \geq b$

Логические операторы работают с логическими величинами и имеют в результате логическую величину: TRUE или FALSE:

Оператор	TRUE, если	Пример
And	обе величины TRUE	$a \text{ And } b$
Or	хотя бы одна величина TRUE	$a \text{ Or } b$
Not	отрицание	$\text{Not } a$

Географические операторы работают с объектными величинами и имеют в результате логическую величину: TRUE или FALSE:

Оператор	TRUE, если	Пример
Contains	если центроид объекта В лежит в границах объекта А	$\text{objectA Contains objectB}$
Contains Part	если границы объекта В частично лежат внутри границ объекта А	$\text{objectA Contains Part objectB}$
Contains Entire	если граница объекта В полностью лежит внутри границ объекта А	$\text{objectA Contains Entire objectB}$
Within	если центроид объекта А лежит в границах объекта В	$\text{objectA Within objectB}$
Partly Within	если границы объекта А частично лежат внутри границ объекта В	$\text{objectA Partly Within objectB}$
Entirely Within	если граница объекта А полностью лежит внутри границ В	$\text{objectA Entirely Within objectB}$
Intersects	если объекты имеют хотя бы одну общую точку	$\text{objectA Intersects objectB}$

### Приоритет

Скобки являются специальным оператором, который выделяет выражение внутри другого выражения. Использование скобок позволяет задавать порядок вычисления выражения, отличный от порядка, диктуемого приоритетом операторов. Ниже в виде колонки перечислены операторы в порядке убывания приоритета. Если скобок нет, то оператор, приоритет которого выше приоритета другого, выполняется первым. Операторы из одной строки имеют равный приоритет и в выражениях выполняются слева направо (за исключением степени, она вычисляется справа налево).

Например, выражение  $3 + 4 * 2$  имеет результат **11** (умножение вычисляется перед сложением). Выражение  $(3 + 4) * 2$  имеет результат **14** (скобки повышают приоритет сложения над умножением).

### Порядок приоритетов выполнения операторов MapBasic

(Высокий приоритет)

скобки

возведение в степень

перемена знака

умножение, деление, деление по модулю, остаток от деления по модулю

сложение, вычитание

географические операторы

операторы сравнения

Not

And

Or

(Низкий приоритет)

## Приложение В: Арифметические и географические операторы

---

### Автоматическое преобразование типов

Если Вы создали выражение с участием данных разных типов и с некоторыми одинаковыми, то MapInfo перед вычислением преобразует типы. Например, если Ваша программа вычитает из величины типа Date другую величину типа Date, MapBasic вычислит результат как целочисленный (разница дней между двумя датами).

В следующей таблице приведены правила автоматического преобразования типов MapBasic.

Оператор	Комбинации операторов	Тип результата
+	Величина типа Date + Число	Date
	Число + Величина типа Date	Date
	Величина типа Integer + Величина типа Integer	Integer
	Число + число	Float
	Что-то + Что-то (все другие комбинации)	String
-	Величина типа Date - Число	Date
	Величина типа Date - Величина типа Date	Integer
	Величина типа Integer - Величина типа Integer	Integer
	Число - Число	Float
*	Величина типа Integer * Величина типа Integer	Integer
	Число * Число	Float
/ (деление)	Число / Число	Float
\ (деление по модулю)	Число \ Число	Integer
MOD (остаток деления по модулю)	Число MOD Число	Integer
^ (степень)	Число ^ Число	Float

## **Приложение С: Техническая поддержка**

Перевод программного пакета MapInfo/MapBasic Professional выполнен фирмой ЭСТИ МАП.

Все права на русскую версию MapInfo и MapBasic принадлежат фирме ЭСТИМАП,  
РФ, Москва, 121019, пер. Сивцев Вражек, 29/16, тел./факс (095) 2415732.  
ESTI -M@.IBRAE.AC.RU

Техническая поддержка осуществляется специалистами ЭСТИ МАП и включает в себя следующее:

Консультирование пользователей MapInfo и MapBasic.

Информация о новейших разработках.

Сведения о новых версиях компонентов пакетов MapInfo и MapBasic.

Описание наиболее важных приемов работы.

Рассылка маркетинговой и технической информации.



# Новые и улучшенные операторы и функции MapBasic

# C

Это новые операторы и функции, доступные для программы MapInfo Professional 7.5.

## Разделы в этом Приложении:

---

### Новые операторы и функции

♦ Оператор Objects Move .....	656
♦ Оператор Objects Offset .....	658
♦ Функция Offset() .....	660
♦ Функция SphericalOffset() .....	662
♦ Функция CartesianOffset() .....	664
♦ Функция OffsetXY() .....	666
♦ Функция SphericalOffsetXY() .....	668
♦ Функция CartesianOffsetXY() .....	670

### Улучшенные операторы

♦ Функция CartesianOffsetXY() .....	670
♦ Оператор Register .....	673
♦ Оператор Server Create Map .....	680
♦ Функция TableInfo( ) .....	683

## Оператор Objects Move

---

### Назначение

**Objects Move** перемещает объекты, полученные из текущей выборки из исходной таблицы.

### Синтаксис

```
Objects Move  
  Angle angle  
  Distance distance  
  [Units unit]  
  [Type {Spherical | Cartesian}]
```

### Описание

**Objects Move** перемещает объекты в пределах исходной таблицы. Исходные объекты получены из текущей выборки. Результирующие объекты заменяют исходные объекты. Объединение данных не является необходимым условием и не осуществляется, так как данные, связанные с исходными объектами, являются неизменными.

Объект перемещен в направлении, заданным параметром *angle*, измеряемым от положительной оси X, указывающей восток (при этом угол измеряется против часовой стрелки), и смещен на расстояние, заданном параметром расстояния *distance*. Расстояние измеряется в единицах, указанных параметром *unit*, если он представлен. Если предложение **Units** пропущено, то текущая единица расстояния будет задана по умолчанию. По умолчанию, MapBasic использует для измерения расстояния мили, об изменении единиц, смотрите раздел, посвященный оператору **Set Distance Units**.

Дополнительная часть предложения **Type** позволяет задать тип расчета расстояния, используемого при смещении объектов. Если используется тип **Spherical**, то вычисления производятся в координатах “Широта/Долгота”, а расстояния рассчитываются на сфере. Если используется тип **Cartesian**, то вычисления производятся на плоскости, на которую спроектированные географические данные и расстояния рассчитываются по декартовым алгоритмам. Если часть предложения **Type** не задана, то используется сферический **Spherical** тип расчета расстояния. Если данные в проекции “Широта/Долгота”, то используется **сферический** тип расчетов независимо от настроек части предложения **Type**. Если данные представлены в проекции План-схема, используются **декартовые** вычисления независимо от настроек части предложения **Type**.



Измерения сделанные на сфере и на плоскости, различаются. Так, при перемещении объекта по карте, построенной в проекции “Долгота/Широта”, его форма не изменится, но поменяется площадь. Это происходит из-за того, что фактическое смещение измеряется в градусах, а соответствующее им расстояние в различных местах земного шара отличается.

Для функций *Offset*, фактическое смещения рассчитано в некоторой установленной точке на объекте (например, центр описывающего прямоугольника), и затем это значение преобразовано из исходных единиц в единицы текущей системы координаты. Если система координаты - Широта/Долгота, преобразование в градусы использует фиксированное число десятичных знаков. Фактическое преобразованное измерение расстояния могло измениться в различных местах объекта. Расстояние от исходного объекта до нового смещенного объекта будет вычислено с гарантированной точностью, если в них используются измерения с фиксированным числом знаков.

## Пример

```
Objects Move Angle 45 Distance 100 Units "mi" Type Spherical
```

## Оператор Objects Offset

---

### Назначение

**Objects Offset** копирует объекты, полученные из текущей выборки, перемещая их относительно исходных объектов.

### Синтаксис

```
Objects Offset  
  [Into Table intotable]  
  Angle angle  
  Distance distance  
  [Units unit]  
  [Type {Spherical | Cartesian}]  
  [Data column = expression [, column = expression ...]]
```

### Описание

**Objects Offset** делает новую копию из объектов, смещенных от первоначальных исходных объектов. Исходные объекты получены из текущей выборки. Результирующие объекты помещаются в таблицу, задаваемую параметром *intotable*, если предложение **Into Table** представлено. Если оно пропущено, объекты помещаются в ту же самую таблицу, из которой получены исходные объекты (основная таблица, из которой выбираются объекты).

Объект перемещен в направлении, заданным *углом*, измеряемым от положительной оси X, указывающей восток (при этом угол измеряется против часовой стрелки), и смещен на расстояние, заданном параметром расстояния *distance*. Расстояние измеряется в единицах, указанных параметром *unit*, если он представлен. Если предложение **Units** пропущено, то текущая единица расстояния будет задана по умолчанию. По умолчанию, MapBasic использует для измерения расстояний мили, об изменении единиц измерения смотрите раздел, посвященный оператору **Set Distance Units**.

Дополнительная часть предложения **Type** позволяет задать тип расчета расстояния, используемого при смещении объектов. Если используется тип **Spherical**, то вычисления производятся в координатах “Широта/Долгота”, а расстояния рассчитываются на сфере. Если используется тип **Cartesian**, то вычисления производятся на плоскости, на которую спроектированные географические данные и расстояния рассчитываются по декартовым алгоритмам. Если часть предложения **Type** не задана, то используется сферический **Spherical** тип расчета расстояния. Если данные в проекции Широта/Долгота, то

используется **сферический** тип расчетов независимо от настроек части предложения **Type**. Если данные представлены в проекции План-схема, используются **декартовые** вычисления независимо от настроек части предложения **Type**.

Если Вы определяете предложение **Data**, то будет призведено объединение данных.

Измерения сделанные на сфере и на плоскости, различаются. Так, при перемещении объекта по карте, построенной в проекции “Долгота/Широта”, его форма не изменится, но поменяется площадь. Это происходит из-за того, что фактическое смещение измеряется в градусах, а соответствующее им расстояние в различных местах земного шара отличается.

Для функций *Offset*, фактическое смещение рассчитано в некоторой установленной точке на объекте (например, центр описывающего прямоугольника), и затем это значение преобразовано из исходных единиц в единицы текущей системы координаты. Если система координаты - Широта/Долгота, преобразование в градусы использует фиксированное число десятичных знаков. Фактическое преобразованное измерение расстояния могло измениться в различных местах объекта. Расстояние от исходного объекта до нового смещенного объекта будет вычислено с гарантированной точностью, если в них используются измерения с фиксированным числом знаков.

## Пример

```
Objects Offset Into Table c:\temp\table1.tbl Angle 45 Distance
100 Units "mi" Type Spherical
```

## Функция Offset()

---

### Назначение

Возвращает копию исходного объекта, смещенного на указанное значение расстояния в заданном направлении.

### Синтаксис

```
Offset(object, angle, distance, units)
```

где:

*object* - смещаемый объект,

*angle* - угол поворота объекта,

*distance* - расстояние перемещения объекта и

*units* - строка, представляющая единицы измерения расстояния.

### Возвращаемое значение

Объект

### Описание

Эта функция производит новый объект, который является копией исходного *объекта*, смещенного на некоторое расстояние, заданное параметром *distance*, в некотором направлении, заданном параметром *angle*. Угол измеряется от положительной оси X против часовой стрелки. Строка *units*, подобно такой же в операторах ObjectLen или Perimeter, является единицей измерения расстояния. Для параметра *distance* применяется способ расчета Spherical кроме тех случаев, когда используется план-схема. Для план-схемы автоматически используется **декартовый** алгоритм расчета расстояний. Используемая система координат - это система координат исходного объекта.

Измерения сделанные на сфере и на плоскости, различаются. Так, при перемещении объекта по карте, построенной в проекции “Долгота/Широта”, его форма не изменится, но поменяется площадь. Это происходит из-за того, что фактическое смещение измеряется в градусах, а соответствующее им расстояние в различных местах земного шара отличается.

Для функций *Offset*, фактическое смещение рассчитано в некоторой установленной точке на объекте (например, центр описывающего прямоугольника), и затем это значение преобразовано из исходных единиц в единицы текущей системы координаты. Если система координаты - Широта/Долгота, преобразование в градусы использует фиксированное число

десятичных знаков. Фактическое преобразованное измерение расстояния могло измениться в различных местах объекта. Расстояние от исходного объекта до нового смещенного объекта будет вычислено с гарантированной точностью, если в них используются измерения с фиксированным числом знаков.

### Пример

```
Offset(Rect, 45, 100, "mi")
```

## Функция SphericalOffset()

---

### Назначение

Возвращает копию исходного объекта, смещенную на заданное расстояние в заданном направлении, при этом используется сферический метод расчетов.

### Синтаксис

```
SphericalOffset(object, angle, distance, units)
```

где:

*object* - смещаемый объект,

*angle* - угол поворота объекта,

*distance* - расстояние перемещения объекта и

*units* - строка, представляющая единицы измерения расстояния.

### Возвращаемое значение

Объект

### Описание

Эта функция производит новый объект, который является копией исходного *объекта*, смещенного на некоторое расстояние, заданное параметром *distance* в некотором направлении, заданном параметром *angle*. Угол измеряется от положительной оси X против часовой стрелки. Строка *units*, аналогично параметрам в операторам ObjectLen или Perimeter, является единицей измерения расстояния. Для параметра *distance* применяется способ расчетов **Spherical**. Поэтому если объект построен в План-схеме, будет выведено сообщение об ошибке, так как тип расстояний *Spherical* не подходит для план-схемы. Это показано при возвращении пустого объекта NULL. Используемая система координат - это система координат исходного объекта.

Измерения сделанные на сфере и на плоскости, различаются. Так, при перемещении объекта по карте, построенной в проекции “Долгота/Широта”, его форма не изменится, но поменяется площадь. Это происходит из-за того, что фактическое смещение измеряется в градусах, а соответствующее им расстояние в различных местах земного шара отличается.

Для функций *Offset*, фактическое смещение рассчитано в некоторой установленной точке на объекте (например, центр описывающего прямоугольника), и затем это значение преобразовано из исходных единиц в единицы текущей системы координаты. Если система

координаты - Широта/Долгота, преобразование в градусы использует фиксированное число десятичных знаков. Фактическое преобразованное измерение расстояния могло измениться в различных местах объекта. Расстояние от исходного объекта до нового смещенного объекта будет вычислено с гарантированной точностью, если в них используются измерения с фиксированным числом знаков.

### Пример

```
SphericalOffset(Rect, 45, 100, "mi")
```

## Функция `CartesianOffset()`

---

### Назначение

Возвращает копию исходного объекта, смещенную на определенное расстояние и угол с использованием **декартового** типа измерения расстояний.

### Синтаксис

```
CartesianOffset(object, angle, distance, units)
```

где:

*object* - смещаемый объект,

*angle* - угол поворота объекта,

*distance* - расстояние перемещения объекта и

*units* - строка, представляющая единицы измерения расстояния.

### Возвращаемое значение

Объект

### Описание

Эта функция производит новый объект, который является копией исходного *объекта*, смещенного на некоторое расстояние, заданное параметром *distance* в некотором направлении, заданном параметром *angle*. Угол измеряется от положительной оси X против часовой стрелки. Строка *units*, аналогично параметрам в операторах `ObjectLen` или `Perimeter`, является единицей измерения расстояния. Для параметра *distance* имеет значение *Cartesian*. Поэтому если исходный объект построен в “Широта/Долгота”, будет выведено сообщение об ошибке, поскольку декартовый тип подсчета расстояний не подходит для “Широта/Долгота”. Это показано при возвращении пустого объекта `NULL`. Используемая система координат - это система координат исходного объекта.

Измерения сделанные на сфере и на плоскости, различаются. Так, при перемещении объекта по карте, построенной в проекции “Долгота/Широта”, его форма не изменится, но поменяется площадь. Это происходит из-за того, что фактическое смещение измеряется в градусах, а соответствующее им расстояние в различных местах земного шара отличается.

Для функций *Offset*, фактическое смещение рассчитано в некоторой установленной точке на объекте (например, центр описывающего прямоугольника), и затем это значение преобразовано из исходных единиц в единицы текущей системы координаты. Если система



координаты - Широта/Долгота, преобразование в градусы использует фиксированное число десятичных знаков. Фактическое преобразованное измерение расстояния могло измениться в различных местах объекта. Расстояние от исходного объекта до нового смещенного объекта будет вычислено с гарантированной точностью, если в них используются измерения с фиксированным числом знаков.

### Пример

```
CartesianOffset(Rect, 45, 100, "mi")
```

## Функция OffsetXY()

---

### Назначение

Возвращает копию исходного объекта, смещенного на заданные значения по осям X и Y.

### Синтаксис

```
OffsetXY(object, xoffset, yoffset, units)
```

где:

*object* - смещаемый объект,

*xoffset* и *yoffset* - расстояния вдоль осей x и y на которые смещается объект и

*units* - строка, представляющая единицы измерения расстояния.

### Возвращаемое значение

Объект

### Описание

Эта функция создает новый объект, являющийся копией исходного объекта, заданного параметром *object*, смещенного на величину *xoffset* вдоль оси X и на величину *yoffset* вдоль оси Y. Строка *units*, аналогично параметрам в операторам ObjectLen или Perimeter, является единицей измерения расстояния. Для параметра *distance* применяется способ расчетов **Spherical**. Поэтому если объект построен в План-схеме, будет выведено сообщение об ошибке, так как тип расстояний *Spherical* не подходит для план-схемы. Это показано при возвращении пустого объекта NULL. Используемая система координат - это система координат исходного объекта.

Измерения сделанные на сфере и на плоскости, различаются. Так, при перемещении объекта по карте, построенной в проекции “Долгота/Широта”, его форма не изменится, но поменяется площадь. Это происходит из-за того, что фактическое смещение измеряется в градусах, а соответствующее им расстояние в различных местах земного шара отличается.

Для функций *Offset*, фактическое смещение рассчитано в некоторой установленной точке на объекте (например, центр описывающего прямоугольника), и затем это значение преобразовано из исходных единиц в единицы текущей системы координаты. Если система координаты - Широта/Долгота, преобразование в градусы использует фиксированное число десятичных знаков. Фактическое преобразованное измерение расстояния могло измениться в

различных местах объекта. Расстояние от исходного объекта до нового смещенного объекта будет вычислено с гарантированной точностью, если в них используются измерения с фиксированным числом знаков.

### Пример

```
OffsetXY(Rect, 92, -22, "mi")
```

## Функция SphericalOffsetXY()

---

### Назначение

Возвращает копию исходного объекта, смещенного на некоторые значения по осям X и Y, при этом используются сферические вычисления расстояний.

### Синтаксис

```
SphericalOffsetXY(object, xoffset, yoffset, units)
```

где:

*object* - смещаемый объект,

*xoffset* и *yoffset* - расстояния вдоль осей x и y на которые смещается объект и

*units* - строка, представляющая единицы измерения расстояния.

### Возвращаемое значение

Объект

### Описание

Эта функция создает новый объект, являющийся копией исходного *объекта*, заданного параметром *object*, смещенного на величину *xoffset* вдоль оси X и на величину *yoffset* вдоль оси Y. Строка *units*, аналогично параметрам в операторам ObjectLen или Perimeter, является единицей измерения расстояния. Для параметра *distance* применяется способ расчетов **Spherical**. Поэтому если объект построен в План-схеме, будет выведено сообщение об ошибке, так как тип расстояний *Spherical* не подходит для план-схемы. Это показано при возвращении пустого объекта NULL. Используемая система координат - это система координат исходного объекта.

Измерения сделанные на сфере и на плоскости, различаются. Так, при перемещении объекта по карте, построенной в проекции “Долгота/Широта”, его форма не изменится, но поменяется площадь. Это происходит из-за того, что фактическое смещение измеряется в градусах, а соответствующее им расстояние в различных местах земного шара отличается.

Для функций *Offset*, фактическое смещение рассчитано в некоторой установленной точке на объекте (например, центр описывающего прямоугольника), и затем это значение преобразовано из исходных единиц в единицы текущей системы координаты. Если система координаты - Широта/Долгота, преобразование в градусы использует фиксированное число десятичных знаков. Фактическое преобразованное измерение расстояния могло измениться в

различных местах объекта. Расстояние от исходного объекта до нового смещенного объекта будет вычислено с гарантированной точностью, если в них используются измерения с фиксированным числом знаков.

### Пример

```
SphericalOffsetXY(Rect, 92, -22, "mi")
```

## Функция `CartesianOffsetXY()`

---

### Назначение

Возвращает копию исходного объекта, смещенного на определенное расстояние по осям X и Y, рассчитанное по декартовым алгоритмам.

### Синтаксис

```
CartesianOffsetXY(object, xoffset, yoffset, units)
```

где:

*object* - смещаемый объект,

*xoffset* и *yoffset* - расстояния вдоль осей x и y на которые смещается объект и

*units* - строка, представляющая единицы измерения расстояния.

### Возвращаемое значение

Объект

### Описание

Эта функция создает новый объект, являющийся копией исходного объекта, заданного параметром *object*, смещенного на величину *xoffset* вдоль оси X и на величину *yoffset* вдоль оси Y. Строка *units*, аналогично параметрам в операторам `ObjectLen` или `Perimeter`, является единицей измерения расстояния. Для параметра *distance* имеет значение *Cartesian*. Поэтому если исходный объект построен в “Широта/Долгота”, будет выведено сообщение об ошибке, поскольку декартовый тип подсчета расстояний не подходит для “Широта/Долгота”. Это показано при возвращении пустого объекта `NULL`. Используемая система координат - это система координат исходного объекта.

Измерения сделанные на сфере и на плоскости, различаются. Так, при перемещении объекта по карте, построенной в проекции “Долгота/Широта”, его форма не изменится, но поменяется площадь. Это происходит из-за того, что фактическое смещение измеряется в градусах, а соответствующее им расстояние в различных местах земного шара отличается.

Для функций *Offset*, фактическое смещение рассчитано в некоторой установленной точке на объекте (например, центр описывающего прямоугольника), и затем это значение преобразовано из исходных единиц в единицы текущей системы координаты. Если система координаты - Широта/Долгота, преобразование в градусы использует фиксированное число десятичных знаков. Фактическое преобразованное измерение расстояния могло измениться в

различных местах объекта. Расстояние от исходного объекта до нового смещенного объекта будет вычислено с гарантированной точностью, если в них используются измерения с фиксированным числом знаков.

### Пример

```
CartesianOffsetXY(Rect, 92, -22, "mi")
```

# Функция LayerInfo( )

---

Мы добавили новый Layer\_Info\_Type, для корректной работы с WMS-таблицами.

## Назначение

Возвращает информацию о слое в окне Карты.

## Синтаксис

**LayerInfo**( *map\_window\_id* , *layer\_number* , *attribute* )

*map\_window\_id* - идентификатор окна Карты

*layer\_number* - число слоев в текущем окне Карты (например, 1 для верхнего слоя); для определения числа слоев в окне Карты, вызывайте MapperInfo( )

*attribute* - код, определяющий тип возвращаемой информации; смотрите таблицу в документации к MapBasic 7.5

**Внимание:** Для получения информации о значениях, об ограничениях, кодах атрибутов, для вывода описания и примеров, смотрите документацию MapBasic 7.5.

<i>attribute</i> код	LayerInfo( ) Возвращаемое значение
LAYER_INFO_TYPE	Короткое целое, обозначающее такие типы файлов:
	LAYER_INFO_TYPE_NORMAL для обычного слоя;
	LAYER_INFO_TYPE_COSMETIC для Косметического слоя;
	LAYER_INFO_TYPE_IMAGE для слоя с растровым изображением;
	LAYER_INFO_TYPE_THEMATIC для тематического слоя;
	LAYER_INFO_TYPE_GRID для слоя поверхности;
	LAYER_INFO_TYPE_WMS для слоя из Web Map Service.



## Оператор Register

**Внимание:** Разделы, выделенные жирным шрифтом в секции синтаксиса, - это измененная часть оператора

### Назначение

Создание таблицы MapInfo Professional из списка, базы данных, текстового файла, растра или изображения поверхности.

### Синтаксис

```
Register Table source_file
{Type "NATIVE" |
  Type "DBF" [Charset char_set] |
  Type "ASCII" [Delimiter delim_char] [Titles] [CharSet char_set] |
  Type "WKS" [Titles] [Range range_name] |
  Type "WMS" [Coordsys coordsys_string [CharSet char_set]
[Into destination_file]
  Type "XLS" [Titles] [Range range_name] |
  Type "Access" Table table_name [Password pwd] [CharSet char_set]
  Type ODBC Connection {Handle ConnectionNumber | ConnectionString}
  Toolkit toolkitname Cache {On | OFF}
  Type "GRID" |
  Type "RASTER" [ControlPoints (MapX1, MapY1) (RasterX1, RasterY1),
    (MapX2, MapY2) (RasterX2, RasterY2),
    (MapX3, MapY3) (RasterX3, RasterY3) [, ...]] [CoordSys ... ]
  Type "SHAPEFILE" [Charset char_set] CoordSys...
  [PersistentCache {On | Off}]
  [Symbol...] [Linestyle Pen(...)] [Regionstyle Pen(...)]
  Brush(...)] [Interactive] [Into destination_file]
```

*source\_file* - строка, определяющая имя существующей базы данных, списка, текстового файла, растра или изображения поверхности. Если Вы регистрируете таблицу Access, этот аргумент должен идентифицировать полноценную базу данных Access.

*char\_set* - имя символа из таблицы кодировки; смотрите отдельно обсуждение **CharSet**.

*delim\_char* - определяет символ используемый как разделитель колонок. Если файл использует в качестве разделителя табулятор, задайте **9**. Если файл использует разделитель запятую, задайте **44**.

*range\_name* - строка, идентифицирующая диапазон имен (например, "Таблица1") или диапазон ячеек (например, диапазон Excel может быть задан так "Лист1!R1C1:R9C6" или "Лист1!A1:F9").

*table\_name* - строка, определяющая таблицу Access.

*pwd* - пароль уровня базы данных для этой базы, задается при включении защиты доступа к базе данных.

*ConnectionNumber* - целое, идентифицирующее существующее соединение с базой данных ODBC.

*ConnectionString* - строка, используемая для соединения с сервером базы данных. Смотрите функцию **Server Connect**.

*toolkitname* - “ODBC” или “ORAINET.”

*SQLQuery* - SQL-запрос, используемый для определения таблицы MapInfo.

**ControlPoints** - дополнительный параметр, используемый если файла это Поверхность или растр. Он задает опорные (контрольные) точки, к ним нужны как минимум 3 пары координат точек карты и растра, которые используются для георегистрации изображения. Если такие точки заданы, то они заменяют любые контрольные точки, связанные с изображением или ассоциированным файлом World.

Предложение **CoordSys** является дополнительным, но может быть задано, если файла это Поверхность или растр. Если предложение CoordSys определено, то оно заменяет любую систему координат, ассоциированную с изображением. Это полезно при регистрации растрового изображения, имеющего ассоциированный файл World.

В случае шейп-файлов, предложение **CoordSys** является обязательным. Компилятор выдаст сообщение об ошибке, если это предложение будет пропущено.

**PersistentCache On** задается, если MAP и ID файлы генерируются во время открытия шейп-файлов и сохраняются на хард диске после закрытия таблицы. Если переключатель PersistentCache установлен на Off, то эти .MAP и .ID файлы будут удалены после закрытия таблицы и будут генерироваться каждый раз при открытии таблицы.

**Symbol (...)** - предложение определяет стиль символа, используемого для точечных объектов, созданных из шейп-файла

**Linestyle Pen (...)** - предложение определяет стиль линии, используемый для линейных объектов, созданных из шейп-файла

**Regionstyle Pen (...) Brush(...)** - предложение определяет стиль границы и стиль заливки, используемый для объектов-полигонов, созданных из шейпфайлов

Ключевое слово **Interactive** является дополнительным, но может быть задано, если файла это Поверхность или растр. Если слово **Interactive** задано, то пользователю будут выведены сообщения-подсказки в случае пропуска контрольной точки или информации о проекции.

Если ключевое слово **Interactive** не задано, файл TAB будет создан автоматически, как будто пользователь в диалоге, появляющемся после выбора файла поверхности или растра в диалоге "Открыть таблицу" выбрал настройку "Показать".

*destination\_file* - определяет имя, которое будет дано таблице MapInfo (TAB-файл). Эта строка может включать в себя путь к файлу; если путь не включен, файл будет создан в той же директории, где находится исходный файл.

## Описание

Перед тем как использовать отличные от таблиц MapInfo файлы (например, файл dBASE), необходимо их зарегистрировать. Оператор **Register Table** сообщает MI Pro что надо проверить такой файл (например, *filename.DBF*) и создать соответствующий ему файл таблицы (*filename.TAB*). После того, как оператор **Register Table** создаст файл таблицы, можно получить доступ к файлу как к таблице MapInfo.

Оператор **Register Table** не копирует и не изменяет файл исходных данных. Вместо этого, он сканирует данные, определяет тип данных в колонках и создает отдельный файл таблицы. Таблица не открывается автоматически. Чтобы открыть таблицу, используйте оператор **Open Table**.

**Внимание:** Каждый файл с данными нуждается в регистрации только *один раз*. После того как операция **Register Table** создала соответственный файл таблицы, в последующем сеансы MI Pro начинаются просто с команды **Открыть** для такой таблицы, а не с повтора процедуры **Регистрировать**.

Предложение **Type** определяет тип исходного файла. В нем есть ключевое слово **Type**, следующее за одной из следующих символьных констант: NATIVE, DBF, ASCII, WKS, XLS, Raster, Access или Grid. Другая информация необходима для обработки некоторых типов таблиц. Если регистрируется файл поверхности, то строка *Coordsys* будет считана из файла поверхности и будет создана таблица MapInfo TAB. Если регистрируется растровый файл, то генерируемый TAB файл будет таким же как если пользователь выбрал бы "Показать" при открытии растрового изображения из диалога "Открыть таблицу".

Если регистрируется файл поверхности, то строка *Coordsys* будет считана из файла поверхности и будет создана таблица MapInfo .TAB. Если регистрируется растр, то генерируемый TAB-файл зависит от того будет ли информация о георегистрации в файле изображения или будет ли ассоциированный файл World.

Предложение **CharSet** определяет тип кодировки шрифтов. Параметр *char\_set* должен быть строкой, такой как “MacRoman” или “WindowsLatin1”. Если предложение **CharSet** пропущено, MI Pro использует кодировку по умолчанию, которая задана для данной платформы в данном сеансе. Более подробно описание предложения **CharSet**, смотрите в соответствующем разделе документации.

Предложение **Delimiter** следует за строкой, содержащей символ разделителя. По умолчанию разделитель - табулятор. Предложение **Titles** показывает, что строка перед табличными данными должна использоваться как заголовок таблицы. Предложение **Range** позволяет задавать поименованный диапазон, данные из которого будут использоваться. Предложение **Into** используется для смены имени или местоположения создаваемого .TAB файла. По умолчанию имя таблицы будет совпадать с именем файла данных, и она будет храниться в той же директории. Таким образом, при считывании с устройства, открытого только для чтения (например, CD-ROM), надо будет сохранить TAB-файл на устройстве, которое доступно и для записи.

## Регистрация таблиц Access

Когда Вы регистрируете таблицу Access, MI Pro ищет колонку счетчика с уникальным индексом. Если колонка счетчика уже существует, MI Pro регистрирует эту колонку в TAB-файле. Колонка открыта только для чтения.

Если таблица Access не имеет колонки счетчика, MI Pro изменяет таблицу Access путем добавления колонки, называемой MAPINFO\_ID с типом данных “счетчик”. В этом случае колонка счетчика не показывается в MapInfo.

**Внимание:** Не изменяйте колонку счетчика ни в каком случае. Это может выполняться автоматически самой MapInfo.

Типы данных Access транслируются в ближайшие типы данных MapInfo. Специальные типы данных Access, такие как OLE-объекты и бинарные поля, не возможны в MapInfo Professional.

## Регистрация таблиц ODBC

Перед прямым доступом к таблице из удаленной базы данных, настоятельно рекомендуется, чтобы Вы сначала открыли таблицу карты (например, canada.tab) для таблицы базы данных. Если Вы не открыли таблицу карты, то целая таблица базы данных будет загружаться сразу целиком, что займет много времени.

Откройте таблицу карты и выберите ее масштаб так, чтобы в окно поместились все необходимые записи (объекты) из таблицы базы данных. Например, если Вы хотите загрузить строчки, относящиеся географически к Онтарио, выберите масштаб для отображения Онтарио в окне Карты. В результате, когда Вы открываете таблицу базы данных будут загружены из базы только те записи, которые попадают в описывающий прямоугольник окна Карты, в данном случае Онтарио.

Это список известных проблем, которые могут возникнуть при прямом доступе/issues при прямом доступе:

- Каждая таблица должна иметь одну уникальную ключевую колонку.
- FastEdit не поддерживается.
- Если в MS ACCESS ключ символьный, то не будут показаны записи, у которых значение ключа меньше, чем полная ширина колонки, если ключ имеет тип char(5) то значение 'aaaa' будет выглядеть как удаленная строка.
- Для прямого доступа, флажок “Только для чтения” в диалоге “Сохранить таблицу” будет неактивным.
- Изменения, сделанные другим пользователем, невидимы, пока браузер не будет прокручен или обновлен еще каким-нибудь способом. Вставки, сделанные другим пользователем не видны до тех пор, пока: 1). Поиск в МОП возвращает запись или 2). Команда PASC выпущена в добавление к тому, что если cache находится на обновлении других пользователей, и не может появиться пока cache не потеряет возможности применяться смещением карты или масштабированием.
- Если клиент объединит (через меню SQL-запрос или через MapBasic) две или более таблиц SPATIALWARE, которые хранятся в различных системах координат, то возникнет проблема. Это - не самое эффективное действие, (лучше присоединять в операторе SQL, который определяет таблицу), но это проблема в настоящее время еще решается.
- Таблицы Oracle 7, которые индексированы по десятичному полю, большему чем 8 байтов, приведет к обрушению MI при редактировании.
- Если оператор Cache OFF расположен перед строкой соединения, то будет генерироваться сообщение об ошибке во время компиляции.

## Регистрация шейп-файлов

Когда регистрируются шейп-файлы, они могут быть открыты в MapInfo Professional только для чтения. Поскольку сам шейп-файл не содержит информацию о проекции, то надо задать предложение **CoordSys**. Также возможно установить стили объектов, которые будут применяться, когда объекты шейп-файлов отображаются в MapInfo Professional. Информация о проекции и стилях хранится в виде метаданных в TAB-файле.

## Примеры

### Пример 1

```
Register Table "c:\mapinfo\data\rpt23.dbf"  
  Type "DBF"  
  Into "Report23"  
  
Open Table "c:\mapinfo\data\Report23"
```

### Пример 2

```
Open Table "C:\Data\CANADA\Canada.tab" Interactive  
Map From Canada  
set map redraw off  
Set Map Zoom 1000 Units "mi"  
set map redraw on  
Register Table "odbc_cancaps"  
  TYPE ODBC  
  TABLE "Select * From informix.can_caps"  
  CONNECTION  
  
DSN=ius_adak;UID=informix;PWD=informix;DATABASE=sw;HOST=adak;  
  SERVER=adak_tli;SERVICE=sqlexec;PROTOCOL=onsoctcp;"  
Into  
  "D:\MI\odbc_cancaps.TAB"  
Open Table "D:\MI\odbc_cancaps.TAB" Interactive  
Map From odbc_cancaps
```

### Пример 3

Регистрация полностью геопривязанного растрового изображения (растровый обработчик может вернуть по крайней мере три контрольных точки и проекцию)

```
Register Table "GeoRef.tif" type "raster" into "GeoRef.TAB"
```

### Пример 4

Регистрация растрового изображения, имеющего ассоциированный файл World, содержащий информацию о контрольных точках, но не о проекции.

```
Register Table "RasterWithWorld.tif" type "raster" coordsys  
earth projection 9, 62, "m", -96, 23, 29.5, 45.5, 0, 0 into  
"RasterWithWorld.TAB"
```

### Пример 5

Регистрация растрового изображения, не имеющего информации о контрольных точках или проекции.

```
Register Table "NoRegistration.BMP" type "raster" controlpoints
(1000,2000) (1,2), (2000,3000) (2, 3), (5000,6000) (5,6)
coordsys earth projection 9, 62, "m", -96, 23, 29.5, 45.5, 0, 0
into "NoRegistration.tab"
```

### Пример 6

Следующий пример показывает регистрацию шейпфайла.

```
Register Table "C:\Shapefiles\CNTYLN.SHP" TYPE SHAPEFILE
Charset "WindowsLatin1" CoordSys Earth Projection 1, 33
PersistentCache Off linestyle Pen (2,26,16711935) Into
"C:\Temp\CNTYLN.TAB"
Open Table "C:\Temp\CNTYLN.TAB" Interactive
Map From CNTYLN
```

### Смотрите также

Open Table, Create Table

## Оператор Server Create Map

---

**Внимание:** Разделы выделенные жирным шрифтом в секции синтаксиса - это измененная и усовершенствованная часть оператора

### Назначение

Эта функция идентифицирует пространственную информацию для таблицы на сервере. Она не изменяет таблицу при добавлении пространственных колонок.

Новое предложение **Object Type** в операторе Server Create Map, позволяет Вам определять объекты как области, линии или объекты всех типов. Если предложение **Object type** не определено, по умолчанию будут возвращены точечные объекты.

### Синтаксис

```
Server ConnectionNumber Create Map
For DBMSTableName
Type { смотрите Maptypes ниже }
CoordSys ...
[MapBounds {Data|Coordsys|Values (x1, y1) (x2, y2)} ]
[ObjectType { Point | Line | Region | ALL } ]
[Symbol (...) ]
[LineStyle Pen(...) ]
[Regionstyle Pen(...) Brush(...) ]
[StyleType style_number (0 или 1) [ Column column_name ]
```

*ConnectionNumber* - целое, идентификатор указанного соединения.

*DBMSTableName* - идентификатор таблицы для таблицы СУБД. Зависит от регистра символов и должно содержать информацию об имени схемы и владельца.

*MapTypes* - один из типов данных СУБД с картографическими данными, перечисленными здесь:

- **MICODE** *MICODEColName (XCoordColName, YCoordColName)* — MICODE это ключевая колонка пространственного индекса и числовые колонки координат X и Y. Имена колонок зависят от регистра символов.
- **XYINDEX** *(XCoordColName, YCoordColName)* — Колонки числовых координат X и Y.
- **ORA\_SP** *SDO\_Spatial\_Column\_Name* — Oracle Spatial
- **IUS\_SW** *ST\_Spatial\_Column\_Name* — SpatialWare IUS Blade
- **IUS\_MM\_SW** *columnname* — MapInfo MapMarker Geocoding DataBlade for SpatialWare



- IUS\_MM\_XY *columnname* — MapInfo MapMarker Geocoding DataBlade for XY
- SPATIALWARE — SpatialWare for SQL Server

**CoordSys ...** Это предложение определяет используемую систему координат и проекцию. Для Oracle Spatial этого не требуется, поскольку информация берется из метаданных Oracle Server.

предложение **ObjectType** - предложение определяет тип объектов в таблице, по умолчанию это точечные объекты.

предложение **Symbol (...)** - предложение определяет стиль символа, используемого для точечного объекта

предложение **Linestyle Pen (...)** - предложение определяет стиль линии, используемого для линейного объекта

предложение **Regionstyle Pen (...) Brush(...)** - предложение определяет стиль линии и стиль заливки, используемых для объектов типа полигон

**StyleType** устанавливает символы для записей таблицы. Должны быть представлены символ Колонки и аргумент, если **StyleType** установлен на 1 (единица). Когда *style\_number* установлен на ноль, символ Колонки игнорируется, и создается исполняемая колонка в MAPCATALOG.

## Описание

Оператор **Server Create Map** делает таблицу связанной с удаленной картографической базой данных. Для таблиц SpatialWare или Oracle Spatial, можно присоединять географические объекты - точки, линии или полигоны. Для всех других таблиц, можно присоединять географическую информацию к таблице только в виде точечных объектов. Любая таблица MapInfo может быть показана в виде списка, но только геокодированная таблица может иметь прикрепленные географические объекты. Только таблица с географическими объектами может отображаться в окне Карты.

**Внимание:** Если Oracle9i - это сервер, а система координат - “Долгота/Широта” без указания топоцентрической системы координат, по умолчанию ставится в соответствие референц-эллипсоид World Geodetic System 1984(WGS 84). Такой процесс совместим с оператором **Server Create Table** и **EasyLoader**.

Настройка *MapBounds* позволяет Вам настроить тот район, который надо сохранить для целой таблицы, чтобы в пределах этого района таблица отображалась в MapCatalog. По умолчанию имеется Data, которая рассчитывает границы района для всех данных на слое. (Для программ, скомпилированных в версиях ранее 7.5, по умолчанию используется **CoordSys**.)

Настройка *Coordsys* в MapBounds сохраняет границы в которых определена система координат. Это не рекомендуется, так как слой, показанный в окне может оказаться без объектов, если прямоугольник, в котором определена система координат, значительно больше чем границы реальных данных. Большинство пользователей делают масштаб слишком крупным, чтобы посмотреть, как выглядят все их данные.

Настройка *Values* позволяет настроить собственные границы применения системы координат для MapCatalog.

## Функция TableInfo( )

### Назначение

Возвращает информацию об открытой таблице.

### Синтаксис

**TableInfo(** *table\_id* , *attribute* )

*table\_id* - строка, имя таблицы, положительное целое, номер таблицы, или 0 (ноль)

*attribute* - целочисленный код, показывающий какой аспект таблицы возвращается

**Внимание:** Для более подробной информации о возвращаемых значениях, ограничениях, кодах атрибутов, описаниях и примерах, смотрите документацию MapBasic 7.5.

<i>attribute</i> код	TableInfo( ) возвращает
TAB_INFO_TYPE	Короткое целое, показывает тип таблицы. Возвращаемое значение будет сравниваться с одной из следующих величин: TAB_TYPE_BASE (если это нормальная сшитая таблица) TAB_TYPE_RESULT (если это результат запроса) TAB_TYPE_IMAGE (если таблица это растр) TAB_TYPE_VIEW (если таблица составная, StreetInfo например, являются составными) TAB_TYPE_LINKED (если таблица связанная). TAB_TYPE_WMS (если таблица из Web Map Service)

