



# MapBasic

11.0

РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Информация содержащаяся в этом документе может быть изменена без уведомления и не представляет собой обязательств со стороны продавца или его представителей. Никакая часть этого документа не может быть воспроизведена или передана в какой-либо форме любыми средствами, механическими или электронными, включая фотокопирование, без письменного разрешения корпорации Pitney Bowes Software Inc., One Global View, Troy, New York 12180-8399.

© 2011 Pitney Bowes Software Inc. Все права защищены. MapInfo, Group 1 Software, и **MapBasic** являются торговыми марками Pitney Bowes Software Inc.. Все остальные марки и торговые марки являются собственностью их владельцев.

Соединенные Штаты:  
телефон: 518 285 6000  
факс: 518 285 6070  
отдел продаж: 800 327 8627  
отдел по работе с правительственными учреждениями: 800 619 2333  
служба технической поддержки: 518 285 7283  
факс отдела технической поддержки: (518) 285-6080  
**pbinsight.com**

Канада  
телефон: 416.594.5200  
факс: 416.594.5201  
отдел продаж: 800.268.3282  
служба технической поддержки: 518 285 7283  
факс отдела технической поддержки: (518) 285-6080  
**pbinsight.ca**

Европа/Великобритания:  
телефон: +44.1753.848.200  
факс: +44.1753.621.140  
служба технической поддержки: +44.1753.848.229  
**pbinsight.co.uk**

Азиатско-Тихоокеанский регион/Австралия  
телефон: +61.2.9437.6255  
факс: +61.2.9439.1773  
служба технической поддержки: 1.800.648.899  
**pbinsight.com.au**

Контактная информация по всем подразделениям Pitney Bowes Software Inc. размещена на сайте: <http://www.pbinsight.com/about/contact-us>.

© 2011 Adobe Systems Incorporated. Все права защищены. Adobe, логотип Adobe, Acrobat и логотип Adobe PDF являются либо зарегистрированными торговыми знаками, либо торговыми знаками Adobe Systems Incorporated в США и/или других странах.

Поставщики информации для © 2011 OpenStreetMap, CC-BY-SA; см. сайты OpenStreetMap <http://www.openstreetmap.org> и CC-BY-SA <http://creativecommons.org/licenses/by-sa/2.0>

libtiff © 1988-1995 Sam Leffler, © 2011 Silicon Graphics International. Все права защищены.

libgeotiff © 2011 Niles D. Ritter.

Amigo, часть авторских прав © 1999 3D Graphics, Inc. Все права защищены.

Halo Image Library © 1993 Media Cybernetics Inc. Все права защищены.

Часть авторских прав принадлежит LEAD Technologies, Inc. © 1991-2011. Все права защищены.

Часть авторских прав © 1993-2011 принадлежат Кену Мартину (Ken Martin), Виллу Шрёдеру (Will Schroeder), Билу Лоренсену (Bill Lorensen). Все права защищены.

ECW является продуктом ERDAS © 1993-2011 ERDAS Inc. и/или партнеров. Все права защищены.

Часть авторских прав © 2011 ERDAS Inc. Все права защищены.

MrSID, MrSID Decompressor и логотип MrSID являются торговыми знаками LizardTech, A Celatrem Company. Используются по лицензии. Часть авторских прав на эту программу принадлежит © 1995-1998 LizardTech компании Celatrem и/или университету штата Калифорния или охраняется патентами США: 5,710,835 и 5,467,110 и используются по лицензии. Все права защищены. MrSID защищен соглашениями США и международными о патентах и авторских правах, заявки на иностранные патенты поданы. Воспроизведение или несанкционированное использование запрещено.

Содержит FME® Objects © 2005-2011 Safe Software Inc., все права защищены.

© 2011 SAP AG, все права защищены. Crystal Reports® и Business Objects™ являются либо зарегистрированными торговыми знаками, либо торговыми знаками SAP AG в Германии и некоторых других странах.

Amunyi PDF Converter © 2000-2011, AMYUNI Consultants – AMYUNI Technologies. Все права защищены.

Civic England - Public Sector Symbols (условные знаки) охраняются авторским правом © 2011 West London Alliance. Эти условные знаки можно использовать бесплатно. Для того чтобы получить, подробную информацию об этих условных знаках, включая сведения о том, как получить их для использования в других приложениях, посетите веб-сайт альянса западных пригородов Лондона по адресу: <http://www.westlondonalliance.org/>

© 2006-2011 Tele Atlas. Все права защищены. Эти материалы являются собственностью и охраняются законами об авторском праве и другими законами об охране интеллектуальной собственности, которой владеет компания Tele Atlas или на которую имеются права, принадлежащие компании Tele Atlas. Использование этих материалов ограничено условиями лицензионного соглашения. Вы несете ответственность за несанкционированное копирование или незаконное распространение этих материалов.

Microsoft Bing: все содержимое службы Bing охраняются авторским правом © Microsoft Corporation и/или партнеров, One Microsoft Way, Redmond, WA 98052, USA. Все права защищены. Название, авторские права и прочая интеллектуальная собственность службы Bing принадлежит Microsoft или партнерам. Microsoft, Windows, Windows Live, логотип Windows logo, MSN, логотип MSN (бабочка), Bing и другие продукты и службы Microsoft могут быть либо зарегистрированными торговыми знаками, либо торговыми знаками Adobe Systems Incorporated в США и/или других странах.

В этом продукте содержится 7-Zip, который используется на условиях лицензии GNU Lesser General Public (Стандартной общественной лицензии ограниченного применения GNU) версии 3, 29 июня 2007, с дополнительными ограничениями unRAR. Текст лицензии можно получить по адресу: <http://www.7-zip.org/license.txt>. Исходный код программы можно получить по адресу: <http://www.7-zip.org>.

Программы и другие продукты, упомянутые здесь, могут являться торговыми марками компаний их производящих, что признается этим заявлением. Названия торговых марок используются только в тексте на пользу владельца торговой марки, без намерения нарушить права торговой марки.

October 19, 2011

# Содержание

---

- Глава 1: Введение . . . . . 14**
  - Требования к системе и компьютеру . . . . .15**
    - Совместимость между версиями . . . . .15
  - Установка среды разработчика MapBasic . . . . .15**
    - Перед тем как начать. . . . .15
    - Установка . . . . .15
    - Запуск MapBasic. . . . .16
  - Стандартные имена и типы файлов MapBasic . . . . .16**
  - Комплект документации MapBasic. . . . .17**
  - Принятые обозначения . . . . .18**
    - Термины . . . . .18
    - Принятые обозначений . . . . .18
  - Техническая поддержка . . . . .19**
    - Прежде чем позвонить. . . . .20
    - Следящая система поддержки (Support Tracking System) . . . . .20
    - Предполагаемое время ответа . . . . .20
    - Обмен информацией . . . . .21
    - Ошибки в программном обеспечении. . . . .21
    - Обучение и консультации . . . . .21
    - Другие информационные ресурсы . . . . .21
- Глава 2: Обзор языка MapBasic . . . . . 23**
  - Как создать и запустить MapBasic-программу? . . . . .25
  - Главные особенности языка MapBasic. . . . .25**
    - MapBasic позволяет настроить интерфейс MapInfo Professional . . . . .25
    - MapBasic позволяет автоматизировать работу MapInfo Professional . . . . .26
    - Средства доступа к базам данных . . . . .26
    - MapBasic поможет обращаться к другим программам из MapInfo Professional . . .26
  - Как осваивать MapBasic?. . . . .26**
  - Окно MapBasic в MapInfo Professional . . . . .28**

<b>Глава 3: Работа в интегрированной среде разработки программ</b>	<b>29</b>
<b>Как отредактировать программу</b>	<b>30</b>
Клавишные сочетания	30
Ограничения текстового редактора MapBasic	32
<b>Компиляция программ</b>	<b>33</b>
Об ошибках при компиляции	34
Запуск откомпилированного файла	35
Написание программ на MapBasic в других редакторах	35
<b>Сборка приложения из нескольких модулей</b>	<b>36</b>
Что такое файл проекта в среде MapBasic?	36
Создание файла проекта	38
Компиляция и сборка проекта	38
Вызов функций и процедур из других модулей	40
<b>Обзор меню среды разработки программ MapBasic</b>	<b>41</b>
Меню Правка	42
Меню Поиск	42
Меню Сборка	43
Меню Окно	44
Меню Справка	44
<b>Глава 4: Основы MapBasic</b>	<b>46</b>
Комментарии	47
Строчные и прописные буквы	47
Продолжение оператора на нескольких строках	47
Константы-коды, определенные в файле MAPBASIC.DEF	47
Как вводить операторы в окно MapBasic	48
Переменные	48
Строковые переменные	50
Массивы	51
Типы данных, заданные пользователем (структуры данных)	52
Глобальные переменные	52
Область определения переменных	53
<b>Выражения</b>	<b>54</b>
Что такое константа?	54
Что такое оператор?	54
Что такое вызов функции?	55
Константы в языке MapBasic	56
Правила преобразования типов переменных	59
Операторы языка MapBasic	59
Порядок применения операторов в языке MapBasic	64
<b>Циклы и другие управляющие операторы</b>	<b>65</b>

Оператор If...Then . . . . .	65
Оператор Do Case . . . . .	66
Оператор GoTo. . . . .	67
Оператор For... Next . . . . .	67
Оператор цикла Do... Loop . . . . .	68
Цикл While... Wend . . . . .	69
Завершение выполнения программы. . . . .	69
Завершение выполнения программы и сеанса работы с MapInfo Professional . . . . .	69
<b>Процедуры . . . . .</b>	<b>69</b>
Процедура Main . . . . .	69
Вызов процедуры . . . . .	70
Вызов процедур с параметрами . . . . .	71
Передача параметров ссылкой. . . . .	71
Передача параметров значением . . . . .	72
Рекурсивный вызов процедур . . . . .	73
<b>Процедуры-обработчики системных событий . . . . .</b>	<b>73</b>
Что такое системное событие? . . . . .	73
Что такое процедура-обработчик системных событий? . . . . .	73
Когда вызываются обработчики событий? . . . . .	76
<b>Рекомендации по использованию процедур-обработчиков . . . . .</b>	<b>76</b>
Делайте процедуры-обработчики короткими! . . . . .	76
Выбор без вызова SelChangedHandler . . . . .	76
Предотвращение бесконечных циклов. . . . .	76
Новые функции. . . . .	77
Область определения функций . . . . .	78
<b>Директивы компилятора . . . . .</b>	<b>78</b>
Директива Define . . . . .	78
Директива Include. . . . .	79
<b>Организация программ. . . . .</b>	<b>79</b>
<b>Глава 5: Поиск ошибок и отладка программ . . . . .</b>	<b>81</b>
<b>Отладка программ на языке MapBasic . . . . .</b>	<b>82</b>
Краткое описание процесса отладки . . . . .	82
Ограничения на оператор Stop . . . . .	83
Другие средства отладки. . . . .	84
<b>Поиск ошибок . . . . .</b>	<b>84</b>
Пример обработки ошибки . . . . .	85
<b>Глава 6: Создание интерфейса пользователя. . . . .</b>	<b>87</b>
<b>Программная обработка событий . . . . .</b>	<b>88</b>
Что такое событие? . . . . .	88
Что происходит, когда пользователь выбирает команду меню? . . . . .	88

Как программа обрабатывает нажатия на кнопки инструментальной панели? . . .	89
Как MapBasic обрабатывает события, происходящие в окнах диалога? . . . . .	90
<b>Меню . . . . .</b>	<b>90</b>
Основные принципы построения и работы с меню . . . . .	90
Добавление новых элементов в меню . . . . .	91
Удаление элементов из меню . . . . .	92
Создание нового меню . . . . .	92
Изменение элемента меню . . . . .	93
Переопределение строки меню . . . . .	94
Задание элементов меню на разных языках . . . . .	95
Настройка быстрых меню MapInfo . . . . .	96
Назначение одной обрабатывающей процедуры нескольким элементам меню . .	96
Команда MapBasic, эквивалентная выбору команды в меню . . . . .	97
Задание сочетаний клавиш . . . . .	97
Управление системой меню через файл MAPINFOW.MNU . . . . .	98
<b>Стандартные диалоговые окна . . . . .</b>	<b>100</b>
Показ простого сообщения . . . . .	100
Показ диалога с двумя кнопками . . . . .	100
Диалог открытия файла . . . . .	101
Показ диалога-индикатора процента выполнения . . . . .	102
Отображение одной записи таблицы . . . . .	102
<b>Новые диалоговые окна . . . . .</b>	<b>102</b>
Размеры и положение элемента диалога . . . . .	103
Элементы окна диалога . . . . .	104
Задание начального значения элемента . . . . .	107
Считывание установок диалога . . . . .	107
Реакция на действия пользователя . . . . .	107
Доступные и недоступные элементы . . . . .	108
Выбор строчки из списка . . . . .	109
Управление списком типа MultiListBox . . . . .	109
Сочетания клавиш, соответствующие элементам . . . . .	110
Заккрытие диалога . . . . .	110
<b>Окна . . . . .</b>	<b>111</b>
Размер и положение окна . . . . .	112
Окна Карт . . . . .	112
Использование слоя анимации для ускорения перерисовки Карты . . . . .	113
Окна Списка . . . . .	114
Окна Графиков . . . . .	115
Окна Отчётов . . . . .	116
Окна Районов . . . . .	116
Окна Сообщений . . . . .	116

Настройка окна Информации . . . . .	117
<b>Панели инструментов. . . . .</b>	<b>118</b>
Что происходит при нажатии кнопки? . . . . .	118
Операторы MapBasic, работающие с инструментальными панелями . . . . .	119
Создание кнопки типа PushButton. . . . .	120
Добавление новой кнопки в панель Операции . . . . .	121
Создание кнопки типа ToolButton . . . . .	122
Выбор пиктограммы для создаваемой кнопки. . . . .	123
Как выбрать объект, на который указали мышкой . . . . .	124
Вставка стандартных кнопок в панели, созданные в программе . . . . .	125
Добавление подсказок к кнопкам . . . . .	126
Закрепление панели в верхней части экрана. . . . .	126
Другие свойства инструментальных панелей . . . . .	126
<b>Курсоры . . . . .</b>	<b>127</b>
<b>Запуск программы в среде MapInfo. . . . .</b>	<b>127</b>
Запуск программ из Рабочего набора STARTUP. . . . .	128
Доступ к Рабочим наборам из программы MapBasic . . . . .	128
<b>Рекомендации по повышению производительности. . . . .</b>	<b>129</b>
Слои анимации. . . . .	129
Как избегать ненужных перерисовок Окна. . . . .	129
Очистка Окна Сообщения . . . . .	130
Подавление изображения индикатора выполнения (диалог “Минуточку”). . . . .	130
<b>Глава 7: Работа с таблицами. . . . .</b>	<b>131</b>
Имена таблиц во время выполнения программы . . . . .	132
Как открыть две таблицы с одинаковыми именами . . . . .	132
Как открыть файл, не являющийся таблицей MapInfo . . . . .	133
<b>Чтение значений из строк и колонок таблицы . . . . .</b>	<b>134</b>
Обращение к колонке с помощью переменной типа Alias . . . . .	135
Обработка составных имен колонок. . . . .	137
Обращение к записям с помощью поля “RowID”. . . . .	137
Использование колонки “Obj” для работы с графическими объектами. . . . .	138
Нахождение адресов в таблице . . . . .	138
Геокодирование . . . . .	138
SQL-запросы. . . . .	139
Ошибки при работе с таблицами и колонками . . . . .	139
<b>Запись значений в таблицу. . . . .</b>	<b>139</b>
<b>Создание новых таблиц. . . . .</b>	<b>140</b>
Изменение структуры таблицы . . . . .	140
Создание индексов и присоединение к таблицам графических объектов . . . . .	141
Информация о структуре таблицы . . . . .	141
Работа с таблицей Selection . . . . .	142

Изменение таблицы Selection . . . . .	143
Внесение изменений в выбранные записи . . . . .	143
Ввод данных пользователем с помощью таблицы Selection . . . . .	144
<b>Доступ к Косметическому слою . . . . .</b>	<b>144</b>
<b>Доступ к окнам Отчетов . . . . .</b>	<b>145</b>
<b>Редактирование в многопользовательской среде . . . . .</b>	<b>146</b>
Правила редактирования таблиц в многопользовательской среде . . . . .	146
Как избежать конфликтов при многопользовательской записи . . . . .	148
Открытие таблицы для записи . . . . .	149
<b>Файлы-компоненты таблицы . . . . .</b>	<b>150</b>
<b>Таблицы, содержащие растровые изображения . . . . .</b>	<b>150</b>
<b>Работа с метаданными . . . . .</b>	<b>152</b>
Что такое метаданные? . . . . .	152
Как выглядят ключи метаданных? . . . . .	152
Примеры работы с метаданными . . . . .	153
<b>Работа со сшитыми таблицами . . . . .</b>	<b>155</b>
Что такое сшитая (seamless) таблица? . . . . .	155
Как работать со сшитыми таблицами? . . . . .	155
Синтаксис MapBasic для сшитых таблиц . . . . .	156
Ограничения при работе со сшитыми таблицами . . . . .	156
<b>Доступ к удаленным базам данных . . . . .</b>	<b>156</b>
Как осуществлять доступ к удаленным данным . . . . .	157
Установка и разрыв связи . . . . .	157
Обработка геометрических объектов в PostGIS . . . . .	159
<b>Доступ к удаленным базам при помощи связанных таблиц . . . . .</b>	<b>159</b>
Прямой доступ к удаленным базам данных . . . . .	160
<b>Повышение производительности при работе с таблицами . . . . .</b>	<b>160</b>
Задайте стандартное представление для таблицы на удаленной СУБД . . . . .	160
Минимизация количества транзакций . . . . .	161
Разумное использование индексов . . . . .	161
Использование “вложенных” выборок . . . . .	161
Оптимизация оператора Select . . . . .	162
Применение оператора Update . . . . .	162
<b>Глава 8: Ввод/вывод в файлы . . . . .</b>	<b>163</b>
<b>Файлы последовательного доступа . . . . .</b>	<b>165</b>
Файлы произвольного доступа . . . . .	167
Двоичные файлы . . . . .	167
<b>Особенности работы с файлами в различных операционных системах и с национальными наборами символов . . . . .</b>	<b>168</b>
Функции ввода/вывода файлов . . . . .	168



<b>Глава 9: Географические и графические объекты</b>	<b>169</b>
<b>Работа с колонкой Obj</b>	<b>170</b>
Создание колонки Object	171
Ограничения на колонки географических объектов	171
<b>Определение атрибутов объекта</b>	<b>172</b>
Стили объектов (Pen, Brush, Symbol, Font)	173
Стили Шрифтов	174
Комбинированные стили	175
Переменные стилей	177
Выбор объектов с заданным стилем	178
<b>Создание новых объектов</b>	<b>180</b>
Операторы создания объектов	180
Функции создания объектов	180
Создание объектов с переменным числом узлов	181
Сохранение графических объектов в таблице	182
<b>Создание новых объектов на основе уже существующих</b>	<b>182</b>
Создание буферной зоны	183
Объединение, пересечение и слияние	183
Создание изограмм	184
Создание сдвинутых копий объектов	184
<b>Изменение объектов</b>	<b>185</b>
Общая процедура изменения графических объектов	185
Перемещение объекта	185
Перемещение объектов и их узлов	185
Изменение стилей графического объекта (Pen, Brush, Font, Symbol)	185
Преобразование областей и полилиний (ломаных)	186
Удаление части графического объекта	186
Точки пересечения	187
<b>Работа с подписями</b>	<b>187</b>
Показ подписей	187
Скрытие подписей	187
Редактирование подписей	188
Запрос к содержимому подписей	188
Другие примеры применения оператора Set Map	188
Разница между подписями и текстовыми объектами	189
<b>Координаты и единицы измерения</b>	<b>191</b>
Единицы измерения	193
<b>Географические запросы</b>	<b>193</b>
Работа с операторами географического анализа	193
Запросы к графическим объектам в таблицах	195
Географические SQL-запросы с промежуточными выборками (подзапросами)	196

Объединения таблиц по географическим критериям . . . . .	197
Пропорциональное обобщение данных . . . . .	198
<b>Глава 10: Особенности MapBasic в среде Microsoft Windows . . . . .</b>	<b>199</b>
Объявление внешней библиотеки . . . . .	200
Передача параметров . . . . .	201
Вызов стандартных библиотек . . . . .	201
Вызов DLL-процедур с помощью ключевого слова Alias . . . . .	201
Аргументы массива . . . . .	202
Типы данных, определенные пользователем . . . . .	203
Логические аргументы . . . . .	203
Уникальные номера (дескрипторы, handles) . . . . .	203
Пример: вызов процедуры из библиотеки KERNEL . . . . .	203
Советы по работе с DLL . . . . .	205
<b>Создание пиктограмм на кнопках и новых курсоров . . . . .</b>	<b>205</b>
Использование стандартных пиктограмм (иконки) . . . . .	206
Создание пиктограмм . . . . .	206
Создание новых курсоров в Windows . . . . .	207
<b>Связь между приложениями с использованием DDE . . . . .</b>	<b>207</b>
Обзор DDE-обмена . . . . .	208
MapBasic как DDE-клиент . . . . .	208
MapInfo Professional в роли DDE-сервера . . . . .	209
Как MapInfo Professional обрабатывает DDE-команды Execute . . . . .	213
Связь с приложениями Visual Basic с использованием DDE . . . . .	213
Пример DDE-обмена сообщениями . . . . .	213
Контроль глобальных переменных с помощью DDE . . . . .	213
<b>Добавление Справочной системы к Вашему приложению . . . . .</b>	<b>214</b>
<b>Глава 11: Интегрированная картография . . . . .</b>	<b>215</b>
<b>Концепции Интегрированной картографии . . . . .</b>	<b>216</b>
<b>Технические аспекты Интегрированной картографии . . . . .</b>	<b>218</b>
Системные требования . . . . .	218
Другие технические замечания . . . . .	218
<b>Простейший пример: “Hello, (Map of) World” . . . . .</b>	<b>218</b>
<b>Подробное обсуждение Интегрированной картографии . . . . .</b>	<b>219</b>
Передача команд в программу MapInfo . . . . .	220
Запрос данных из программы MapInfo Professional . . . . .	220
Настройка быстрого меню MapInfo . . . . .	225
Прерывание работы программы на Visual Basic . . . . .	227
Замечание о командных строках MapBasic . . . . .	227
О диалогах . . . . .	228
О клавишах-акселераторах . . . . .	228

<b>Использование Callback-вызовов</b>	<b>228</b>
Требования к функциям уведомления	228
Схема использования уведомлений в OLE	229
Обработка переданных данных	230
Синтаксис C/C++ для функций уведомления	231
<b>Другие способы использования OLE-уведомлений</b>	<b>232</b>
Обратные вызовы DDE	232
Обратные вызовы MBX	232
<b>Справочная система</b>	<b>233</b>
Вызов стандартного справочного файла MapInfo Professional	233
Запрещение вызова Справочной системы	233
Вызов пользовательского справочного файла	233
<b>Полезные операторы и функции языка MapBasic</b>	<b>234</b>
<b>Объектная модель механизма управления OLE</b>	<b>236</b>
Использование объектной модели OLE в процессах MapInfo Professional	238
Свойства объекта Application	238
Свойства объекта DockWindow	242
Свойства семейства MBAplications	244
Свойства объекта в семействе MBAplications	245
Свойства коллекции MBGlobals	246
Свойства объекта в семействе MBGlobals	246
Свойства объектов MIMapGen	247
Методы объекта MIMapGen	249
Свойства объекта MISearchInfo	250
Метод объекта MIRow	251
Свойства объекта MIField	251
Свойства объекта MISelection	252
<b>Аргументы командной строки MapInfo Professional</b>	<b>253</b>
Введение в Интегрированную картографию с поддержкой Visual C++ и MFC	255
<b>Добавление инструментальных кнопок</b>	<b>259</b>
Обработка ошибок MapInfo Professional	261
Добавление поддержки сервера OLE Automation	261
Добавление уведомления (callback) WindowContentsChanged	262
<b>Где получить дополнительную информацию</b>	<b>263</b>
<b>Глава 12: Работа в среде .Net</b>	<b>264</b>
О терминах	265
<b>Первые шаги</b>	<b>265</b>
Создание класса в .Net	266
Создание и копирование файла сборки	267
Объявление и вызов метода из MapBasic	267
Вызов метода по его псевдониму	269

Передача аргументов в .Net . . . . .	269
Замечания о скорости выполнения . . . . .	270
<b>Работа со структурами в .Net . . . . .</b>	<b>270</b>
Передача созданных пользователем типов (структур) в .Net . . . . .	270
Ограничения на передачу структур . . . . .	274
<b>Обработка ошибок. . . . .</b>	<b>274</b>
<b>Работа с GAC Suite . . . . .</b>	<b>275</b>
Загрузка файла сборки из кеша Global Assembly Cache (GAC) . . . . .	275
<b>Управление MapInfo Professional изнутри метода .Net . . . . .</b>	<b>276</b>
<b>Интегрированная картография в .Net . . . . .</b>	<b>278</b>
Доступ к MapInfo Professional посредством COM . . . . .	278
Методы обратного вызова (callback) . . . . .	279
Защита потоков . . . . .	282
<b>Приложение А: Примеры программ. . . . .</b>	<b>283</b>
Каталог Samples\DLLEXAMP . . . . .	284
Каталог Samples\DotNet . . . . .	284
Каталог Samples\MapBasic. . . . .	284
Каталог Samples\MFC . . . . .	291
Каталог Samples\PwrBldr . . . . .	291
Каталог Samples\VB4. . . . .	291
Каталог Samples\VB6. . . . .	291
<b>Приложение В: Сведения об операторах . . . . .</b>	<b>292</b>
Операторы сравнения . . . . .	293
Логические Операторы . . . . .	294
Географические операторы . . . . .	294
Старшинство выполнения операторов. . . . .	295
Автоматическое преобразование типов. . . . .	296
<b>Приложение С: Поддерживаемые типы данных в ODBC-таблицах. . . . .</b>	<b>297</b>
<b>Приложение D: Присоединение геоинформации к удаленной таблице . .</b>	<b>298</b>
<b>Приложение E: Создание Каталога Карт MapInfo_MapCatalog вручную . .</b>	<b>300</b>
Как присоединить геоинформацию к удаленной таблице . . . . .	302
<b>Приложение F: О служебных и вспомогательных файлах . . . . .</b>	<b>305</b>
Словарь терминов, используемых при обновлении программ . . . . .	307
Файлы и каталоги данных приложения . . . . .	309
Стандартные маршруты . . . . .	311
Изменения в реестре . . . . .	311
Требования для установки и политики групп . . . . .	312
MapBasic v.6.5 и 6.0 . . . . .	312

---

MapBasic v.7.0 и более новые версии .....	313
<b>Приложение G: Словарь MapBasic.....</b>	<b>314</b>
<b>Предметный указатель.....</b>	<b>325</b>

# Введение

Добро пожаловать в среду разработчика MapBasic!

MapBasic® – мощный и одновременно простой в использовании язык программирования, который позволит Вам создавать собственные приложения в среде MapInfo Professional.

В этой главе описана процедура установки MapBasic. Глава **Обзор языка MapBasic** содержит сведения о возможностях и устройстве языка MapBasic.

## В этой главе:

- ♦ Что нового .....15
- ♦ Требования к системе и компьютеру .....15
- ♦ Установка среды разработчика MapBasic .....15
- ♦ Стандартные имена и типы файлов MapBasic .....16
- ♦ Комплект документации MapBasic .....17
- ♦ Принятые обозначения .....18

## Что нового

Новое в этой версии MapBasic:

- Добавлена поддержка классов курсоров, см. раздел [Курсоры на стр. 127](#).

## Требования к системе и компьютеру

Убедитесь заранее, что компьютер, на который будет установлен MapBasic для Windows, отвечает следующим минимальным требованиям:

Требования	Следует придерживаться следующих параметров:
Операционная система	<ul style="list-style-type: none"><li>• Windows XP Professional 32-bit Service Pack 3 (SP3)</li><li>• Windows 7 Ultimate 32-bit SP1</li><li>• Windows 7 Ultimate 64-бит с режимом совместимости 32-битных программ</li><li>• Windows 2008 Server 32-bit SP2</li><li>• Windows 2008 Server 32-bit SP2 с поддержкой XenServer</li><li>• Windows 2008 Server R2 64-bit SP1</li></ul>
Дисплей	Любая видеокарта, поддерживаемая Windows
Мышь	Любая мышь, поддерживаемая Windows
Свободное место на диске	10 MB

## Совместимость между версиями

Программы, созданные в ранних версиях MapBasic, будут работать под MapInfo Professional более поздних версий.

## Установка среды разработчика MapBasic

### Перед тем как начать

Процедура установки MapBasic описана ниже. Прodelайте следующее:

- Установите MapInfo Professional до установки MapBasic. Внимательно прочитайте инструкцию в *Руководстве по установке MapInfo Professional*.

### Установка

Среда разработки MapBasic на английском языке предоставляется бесплатно с Web-сайта Pitney Bowes Software Inc. Чтобы получить копию, откройте сайт [www.mapinfo.com](http://www.mapinfo.com), введите "MapBasic Overview" поле поиска **Search**. Также Вы можете получить информацию о том, как создавать на языке MapBasic свои программы и запускать их в среде MapInfo Professional, по адресу [www.mapinfo.com/mapbasic](http://www.mapinfo.com/mapbasic).


Русская версия среды разработки MapBasic поставляется платно. Ознакомиться с условиями поставки русской версии MapBasic вы можете на сайте компании “ESTI MAP”, по адресам:

- [www.esti-map.ru](http://www.esti-map.ru)
- [www.mapinfo.ru](http://www.mapinfo.ru)

## Запуск MapBasic

Для запуска среды разработки MapBasic сделайте одно из следующих действий:

- дважды щелкните по иконке MapBasic Professional на рабочем столе.
- в меню **Пуск** ("Start") выберите MapBasic 11.0 из списка **Программы**.

 Всегда можно проверить наличие обновлений программы, если выполнить команду **Справка > Обновления**.

## Стандартные имена и типы файлов MapBasic

При установке языка MapBasic копируются следующие стандартные файлы:

Имя файла	Описание
ERRORS.DOC	Текстовый файл со списком сообщений об ошибках MapBasic
ICONS.DEF	Файл заголовков, содержащий объявления стандартных переменных, относящихся к виду кнопок и курсоров
MAPBASIC.BAS	Файл заголовков для программистов на Visual Basic; аналогичен MAPBASIC.DEF, но в нем использован синтаксис Visual Basic
MAPBASIC.CHM	Файл справки языка MapBasic
MAPBASIC.DEF	Файл заголовков, содержащий объявления стандартных переменных
MAPBASIC.EXE	Исполняемый файл, позволяющий Вам работать в среде MapBasic
MAPBASIC.H	Файл заголовков для программистов на языках C/C++; аналогичен MAPBASIC.DEF, но в нем использован синтаксис C/C++
MBLIB.DLL	Часть программной среды; содержит библиотеки общего доступа
MBRES.DLL	Часть программной среды; содержит ресурсы: диалоги и строки
MENU.DEF	Файл заголовков, содержащий объявления стандартных переменных, относящихся к системе меню
MISECUTIL.DLL	Часть программной среды; содержит исполняемые модули общего доступа



Имя файла	Описание
papersize.def	Включаемый в программу файл заголовков для разработчиков программ на MapBasic. Он содержит определения, необходимые для управлением печатью операторами MapBasic.
Каталог SAMPLES	В нем содержатся примеры готовых программ.
USRINFMB.LOG	Содержит информацию о процессе установки.

При работе в среде MapBasic используются следующие стандартные расширения:

Имя файла	Описание
<имя_файла>.MB	Файл с программой (исходный текст на языке MapBasic).
<имя_файла>.MBX	Откомпилированный (выполняемый) файл.
<имя_файла>.MBP	Файл описания модулей проекта (содержит список всех модулей, собираемых в единую программу).
<имя_файла>.MBO	Объектный файл (создается при сборке программы из нескольких модулей).
<имя_файла>.ERR	Список сообщений об ошибках, полученный при компиляции программы.

## Комплект документации MapBasic

Кроме *Руководства пользователя MapBasic*, в набор документации входят *Справочник MapBasic* и интерактивная *Справочная система*. Интерактивный *Справочник MapBasic* содержит описание всех команд MapBasic. *Справочная система* содержит всю информацию из *Справочника MapBasic* плюс описания диалогов и меню.

Чтобы открыть *Руководство пользователя MapBasic* и *Справочник MapBasic*, проделайте одно из следующих действий:

- найдите эти документы по адресу <http://go.pbinsight.com/mapbasicdocs>.
- установите их в Вашей системе и открывайте локально. Документация обычно устанавливается вместе с MapInfo Professional в папку **MapInfo\Professional\Documentation** (см. MapBasicUserGuide.pdf и MapBasicReference.pdf).

## Принятые обозначения

В данном Руководстве используются следующие термины и условные обозначения:

### Термины

В настоящем Руководстве мы обращаемся к разработчику программ на Вы, а тех, кто с ней будет работать, называем пользователями. Например:

Вы можете использовать оператор **Note** в языке MapBasic для выдачи сообщений пользователю.

Между понятиями программа и приложение проведено такое различие:

- *Программа* – это текстовый файл, созданный программистом. Как правило, файлы с программами на MapBasic имеют расширение "MB".
- *Приложение* – это двоичный исполняемый файл (в среде MapInfo). Для создания приложения необходим файл с текстом программы. MapBasic компилирует текст программы, создавая приложение. Затем пользователь запускает приложение (или "выполняет" его). Приложения, созданные с помощью MapBasic, обычно имеют расширение "MBX" (MapBasic eXecutable).
- *Командой* мы называем то, что можно выполнить, выбрав один из пунктов меню. Например, чтобы открыть файл, следует выбрать **Открыть** из меню **Файл**.
- *Оператор* – это действие, которое можно выполнить из программы на языке MapBasic. Например, программа на языке MapBasic может использовать оператор **Select** для выбора записей из таблицы.

### Принятые обозначений

Моноширинным шрифтом `Courier` выделены примеры программ на языке MapBasic:

```
Note "Привет от MapBasic!"
```

Полужирным шрифтом с заглавной буквы выделены ключевые слова языка MapBasic:

Оператор **Stop** используется при отладке.

Во всех примерах в данном руководстве первые буквы всех ключевых слов языка MapBasic являются заглавными. Однако, можно и не следовать такому стилю в своих программах. Вы можете набирать все большими буквами, все маленькими или большими и маленькими в любой комбинации.


Команды меню в интегрированной среде MapBasic приводятся с использованием знака "больше" (>), например:

- Выполните команду **Файл > Новый**, чтобы открыть новое окно.

Выражение "**Файл > Новый**" является сокращением от "команда **Новый** из меню **Файл**". Команды меню будут выделяться в тексте жирным шрифтом.

## Техническая поддержка

Служба технической поддержки всегда готова помочь вам. В этом разделе описывается информация, которую вам потребуется представить при звонке в службу технической поддержки. Здесь также объясняются некоторые технические процедуры, которые помогут в разрешении проблем.

 Пользователи MapInfo Professional v10.0, имеющие доступ к сервису MATS, могут обращаться по телефону к службе технической поддержки MapInfo по вопросам работы с MapBasic.

Выберите из этого списка службу технической поддержки вашего региона:

Ваш регион	Информация о контактах	Часы
<b>Америка</b>	телефон: 518.285.7283 факс: 518.285.6080 techsupport@mapinfo.com	Понедельник - пятница с 8:00 до 19:00 (EST), кроме выходных. По понедельникам, с 10:30 - 11:30 мы закрыты, в связи с проведением тренингов.
<b>Азия - Тихоокеанский регион</b>	телефон: 61.7.3844.7744 факс: 61.7.3844.2400 ozsupport@mapinfo.com	Понедельник - пятница с 9:00 до 17:00 EST по австралийскому восточному стандартному времени, кроме выходных.
<b>Европа/Средний Восток/Африка</b>	телефон: +44(0) 1634 880141 support-europe@mapinfo.com	Понедельник - пятница с 8:00 до 19:00 (GMT), кроме выходных.
<b>Германия</b>	телефон: +49 (0) 6142-203-400 факс: +49 (0) 6142-203-444 supportgermany@mapinfo.com	Понедельник - пятница с 8:00 до 19:00 (MEZ), кроме выходных.
<b>Россия и СНГ</b>	телефон: +7 (499) 241-00-57 факс: +7 (499) 241-57-32 support@mapinfo.ru	Понедельник - пятница с 10:00 по 17:00 (GMT+3)б кроме выходных.

Для получения технической поддержки вы должны зарегистрировать продукт. Это очень легко сделать во время установки. Чтобы получить дополнительную информацию о программах поддержки MapBasic, свяжитесь с представителем технической поддержки в вашем регионе. Иногда представителям технической поддержки может потребоваться пример ваших данных и небольшой фрагмент кода для воспроизведения ваших действий. Наиболее предпочтительный способ передачи этой информации через электронную почту или через FTP сайт.

Лучший способ обеспечить эффективное использование программного обеспечения MapInfo Professional – это обучение пользователей работе с продуктами. Семинары проводятся по методикам компании ESTI MAP в оборудованном учебном классе под руководством опытного преподавателя. Слушатели получают большой объем необходимой информации о возможностях ГИС MapInfo и языка программирования MapBasic. Они выполняют на компьютере комплекс учебных упражнений, позволяющих закрепить полученные знания и навыки по работе с ГИС.

Программу семинаров можно посмотреть на сайте [www.mapinfo.ru](http://www.mapinfo.ru)

телефон: +7(499) 241-57-32, +7(499) 241-25-42

адрес в Интернете: [edu@mapinfo.ru](mailto:edu@mapinfo.ru)

время работы: понедельник-пятница с 10:00 по 17:00 (GMT+3:00), кроме выходных

## Прежде чем позвонить

Пожалуйста, имейте под рукой эту информацию, когда связываетесь со службой технической поддержки MapInfo Professional

1. Серийный номер. Для получения технической поддержки вы должны иметь зарегистрированный серийный номер.
2. Наименование вашей организации и имя пользователя. Обращающийся в службу поддержки должен связаться с персоной указанной в соглашении о поддержке.
3. Номер версии продукта.
4. Наименование и версия операционной системы.
5. Краткое описание сути проблемы. Здесь может быть полезна следующая информация:
  - сообщения об ошибках
  - ситуация при которой возникают проблемы
  - часто или нет возникает подобная проблема?

## Следящая система поддержки (Support Tracking System)

Запросы пользователей в службу Технической поддержки фиксируются и обрабатываются системой контроля обращений Support Tracking System. Система также представляет возможность отслеживать все обращения. Эта система помогает службе технической поддержки отвечать на все запросы клиентов быстро и эффективно.

## Предполагаемое время ответа

Большая часть ваших вопросов, будет решена во время телефонного разговора. Если решения не будет найдено сразу, служба технической поддержки пришлет ответ до наступления конца рабочего дня. Работник службы технической поддержки будет сообщать вам, о том как решается проблема.

Техническая поддержка по электронной почте в целом следует тем же правилам, но возможна задержка в получении ответа.

## Обмен информацией

Иногда представителям технической поддержки может потребоваться пример ваших данных, для воспроизведения сценария. В случае, когда используются средства разработки (такие как MapXtreme), может потребоваться небольшой фрагмент кода.

Наиболее предпочтительный способ передачи этой информации через электронную почту или через FTP сайт. Используйте следующие адреса E-mail:

- в Соединенных Штатах – techsupport@mapinfo.com
- в Европе – support-europe@mapinfo.com
- в Австралии – ozsupport@mapinfo.com
- В России и СНГ – support@mapinfo.ru

## Ошибки в программном обеспечении

Если Вы обнаружите ошибку в программном обеспечении, свяжитесь с представителями Pitney Bowes Software Inc., ошибка будет занесена в базу данных, ей будет присвоен номер и вы сможете контролировать ситуацию с исправлением ошибки. В обновлённых версиях программы большинство указанных ошибок текущей версии будет исправлено.

## Обучение и консультации

При возникновении проблем с использованием языка MapBasic, сотрудники отделов технической поддержки корпорации MapInfo окажут вам необходимую помощь. Техническая поддержка для пользователей MapBasic включает в себя помощь в анализе сообщений об ошибках, указание на разделы документации, где приведены интересующие вас сведения, а также объяснение основ работы. Прежде чем обращаться за консультацией, найдите серийный номер вашего пакета. В России продажу и техническую поддержку осуществляет компания ESTI MAP (см. раздел **Техническая поддержка на стр. 19**).

Если требуется большой объем помощи в разработке программ на MapBasic, Вы можете обратиться в отдел консультаций. В отдельных случаях, можно организовать работу инженеров, владеющих опытом создания программ на MapBasic, с выездом к заказчику. Дополнительную информацию можно получить, обратившись в отдел работы с партнерами

## Другие информационные ресурсы

### Архив MapInfo-L

MapInfo-L (MapInfo List) – это независимый форум, в котором участвуют специалисты Pitney Bowes Software Inc. Чтобы подписаться на форум, зайдите на страничку:

<http://groups.google.com/group/mapinfo-l?hl=en>

и нажмите **Join this group**.



Любые сообщения, отправленные в этот список рассылки, могут быть прочитаны любым пользователем, подписанным на рассылку.

---

### Другие полезные web-сайты для пользователей MapInfo

Web-сайт **MapInfo Tools** организован Барбарой Кэрролл (Barbara Carroll) как хранилище бесплатных приложений и утилит:

<http://mapinfotools.com>

Web-сайт **GISnet** поддерживается партнером MapInfo Биллом Тоеном (Bill Thoen). Он содержит множество ссылок на ГИС-ресурсы и, в частности, ссылки на ресурсы MapInfo.

<http://www.gisnet.com/catalog/software/tools/index.php>

# Обзор языка MapBasic

MapBasic – это программный пакет, позволяющий создавать собственные приложения, работающие в среде MapInfo Professional.

## В этой главе:

- ♦ Первые шаги .....24
- ♦ Главные особенности языка MapBasic .....25
- ♦ Как осваивать MapBasic? .....26
- ♦ Окно MapBasic в MapInfo Professional .....28

## Первые шаги

Пакет MapBasic поставляется вместе с интегрированной средой разработки. В этой интегрированной среде Вы можете писать и компилировать программы на языке MapBasic.

Среда разработки включает в себя:

- Текстовый редактор, в котором Вы можете набирать тексты программ. (Если Вы привыкли к другому редактору, можете использовать его вместо редактора MapBasic. Дополнительную информацию Вы можете найти в разделе **Работа в интегрированной среде разработки программ на стр. 29**.
- Компилятор языка MapBasic. После написания программы ее надо откомпилировать, собрать “выполняемый” файл (т.е. такой, который можно запустить под MapInfo Professional).
- Сборщик MapBasic (linker). При создании больших приложений, состоящих из нескольких модулей, Вам потребуется "собирать" их в единое целое с помощью компоновщика.
- Справочную систему по языку MapBasic, содержащую сведения об операторах и функциях языка программирования MapBasic.
- Из названия языка Вы можете заключить, что MapBasic напоминает обыкновенный BASIC. На самом же деле программы на MapBasic не совсем похожи на программы на стандартном языке BASIC, хотя MapBasic близок к новым версиям BASIC, созданным в последние годы (например, к Microsoft Visual Basic).

Пример кода на традиционном BASICe	Пример кода на языке MapBasic
20 GOSUB 3000 30 IF DONE = 1 THEN GOTO 90 40 FOR X = 1 TO 10 50 GOSUB 4000 60 NEXT X 80 GOTO 30	Call Check_Status(quit_time) Do While Not quit_time For x = 1 To 10 Call Process_batch(x) Next Loop

Программы на MapBasic работают под управлением системы MapInfo Professional. Сначала нужно создать и скомпилировать программу в интегрированной среде MapBasic; затем запустить MapInfo Professional и после этого – Вашу программу. Таким образом, программы на языке MapBasic не являются полностью самостоятельными; их можно запускать только при наличии MapInfo Professional.

В то же время нельзя сказать, что MapBasic – только макроязык; MapBasic – мощный язык программирования, включающий в себя более 300 операторов и функций. Кроме того, поскольку MapBasic работает в среде MapInfo Professional, то он позволяет использовать всю гамму географических возможностей MapInfo Professional.



## Как создать и запустить MapBasic-программу?

**Глава 3: Работа в интегрированной среде разработки программ** предлагает подробные инструкции по созданию программ на MapBasic.

Если же Вы хотите начать прямо сейчас, то выполните следующие действия:

1. Запустите MapBasic.
2. Выполните команду **Файл > Новый**, чтобы открыть новое окно программы.
3. Наберите текст программы на языке MapBasic в этом окне. Например, можете просто набрать:  

```
Note "Привет от MapBasic!"
```
4. Выполните команду **Файл > Сохранить**, чтобы сохранить программу в файле. Укажите имя файла, например, `WELCOME.MB` (стандартное расширение программ на MapBasic – `.MB`).



Не закрывайте окно с текстом программы.

---

5. Выполните команду **Сборка > Компилировать файл**. MapBasic откомпилирует Вашу программу (`WELCOME.MB`) и создаст соответствующий исполняемый файл (`WELCOME.MBX`).
6. Запустите MapInfo.
7. Выполните команду **Файл > Запустить программу MapBasic**. MapInfo Professional запросит у Вас название программы, которую следует запустить.
8. Выберите файл **WELCOME.MBX**, чтобы запустить программу, которая выведет сообщение “Привет от MapBasic!” в диалоговом окне.

Таковы основные этапы создания, компиляции и выполнения приложений на языке MapBasic. На практике все выглядит, конечно, сложнее; так, в нашем примере ничего не сказано о том, что делать, если при компиляции была обнаружена ошибка. Как уже было сказано, подробные инструкции содержит **Глава 3: Работа в интегрированной среде разработки программ**.

## Главные особенности языка MapBasic

### MapBasic позволяет настроить интерфейс MapInfo Professional

Вы можете настраивать интерфейс MapInfo Professional. Программы на языке MapBasic могут изменять стандартные меню MapInfo, добавлять команды в основное меню MapInfo и создавать собственные диалоги.

Таким образом, MapBasic позволяет создавать готовые геоинформационные системы, которые позволяют пользователям решать свои задачи с минимальными затратами на обучение.

## MapBasic позволяет автоматизировать работу MapInfo Professional

Программы на языке MapBasic часто позволяют сэкономить время, затрачиваемое на некоторую "ручную" работу в MapInfo. Например, пользователю может потребоваться создать в MapInfo *градусную сетку* (набор параллелей и меридианов с заданным шагом). Рисование градусной сетки "вручную" является достаточно утомительным занятием, поскольку надо повторить процедуру рисования линии для каждой параллели и каждого меридиана. Программа же на MapBasic позволяет сгенерировать градусную сетку без утомительного рисования.

## Средства доступа к базам данных

Всего одного оператора MapBasic достаточно для выполнения сложнейших запросов к базе данных. Например, Вы можете с помощью оператора языка MapBasic **Select** (который работает по принципу операторов Select в SQL-языках) выполнить запрос к базе данных, выбрать записи, которые следует показать на экране, задать порядок их сортировки и обобщить результаты выполнения запроса. Все перечисленное выполняется с помощью одного единственного оператора MapBasic.

Такие операторы MapBasic, как **Select** и **Update**, заменяют собой десятки строк текста, которые Вам пришлось бы написать, если бы Вы использовали другой язык программирования.

## MapBasic поможет обращаться к другим программам из MapInfo Professional

Можно использовать не только операторы и функции, реализованные в языке программирования MapBasic. Поскольку MapBasic обеспечивает открытую архитектуру, Ваши программы могут использовать процедуры из внешних библиотек. Если потребуется реализовать функции, не включенные в состав стандартного набора команд MapBasic, открытая архитектура MapBasic позволит выполнить такую задачу.

Программы на MapBasic могут использовать динамический обмен данными (Dynamic Data Exchange – DDE) для взаимодействия с другими пакетами программ, включая программы на Visual Basic. Программы MapBasic могут использовать процедуры из динамически связанных библиотек Windows (Dynamic Link Library – DLL-файлов). Можно купить необходимые DLL-файлы, а можно и написать их самим, используя языки программирования такие как C или Pascal. MapBasic обеспечивает использование интегрированной картографии, с помощью которой можно добавить функции MapInfo Professional в программы, написанные в других средствах разработки, например, Visual Basic. Подробности содержит [Глава 11: Интегрированная картография](#).

## Как осваивать MapBasic?

Поскольку MapBasic предназначен для программирования в среде MapInfo, Вам следует освоить MapInfo прежде, чем начать работу с MapBasic. В данном Руководстве мы предполагаем, что Вы уже достаточно близко знакомы с такими концепциями MapInfo, как Таблицы, окна Списков, слои Карт и Рабочие Наборы.

Ознакомившись с работой MapInfo Professional, Вы можете приступить к изучению языка MapBasic с помощью данного Руководства и *Справочной системы*.

#### *Руководство пользователя MapBasic*

В этой книге изложены основные концепции программирования на языке MapBasic.

*Руководство пользователя MapBasic* – первая книга, с которой Вам следует ознакомиться при изучении языка MapBasic. В каждой главе описана своя область программирования. Следующие главы должен прочитать каждый программист: **Основы MapBasic. Создание интерфейса пользователя**, где рассказывается как создавать меню и диалоги, а также **Ввод/вывод в файлы**, где рассказывается, как работать с подсистемой ввода/вывода.

#### *Справочник MapBasic*

Содержит расположенные в алфавитном порядке подробные описания всех операторов и функций языка MapBasic. Обращайтесь к *Справочнику MapBasic*, когда у Вас возникнет вопрос по какому-либо конкретному элементу языка.

#### **Примеры программ**

Многие программисты считают, что лучшим способом освоения нового языка программирования является изучение примеров программ. Комплект поставки MapBasic включает библиотеку с примерами программ. Смотрите каталог "Samples" на компакт-диске MapBasic, в котором приведены примеры кода программ MapBasic.



В *Руководстве пользователя MapBasic* мы будем часто ссылаться на пример TEXTBOX.MB. Вы можете ознакомиться с ним, прежде чем изучать MapBasic.

---

#### **Рабочие Наборы MapInfo**

MapInfo Professional может сохранять информацию об использовавшихся окнах и таблицах в конце сеанса работы в виде так называемого Рабочего Набора. Если Вы откроете файл описания Рабочего Набора в любом текстовом редакторе, то увидите, что он содержит предложения языка MapBasic. Вы можете скопировать один или несколько операторов MapBasic из файла описания Рабочего Набора и вставить в свою программу. Ведь, по сути дела, любой Рабочий Набор MapInfo является программой на MapBasic.

Например, Вы хотите написать программу на MapBasic, которая создает шаблон печатной страницы. В этом случае Вы можете создать такой шаблон вручную в MapInfo Professional в виде окна Отчета и сохранить его в виде Рабочего Набора. Файл описания Рабочего Набора будет содержать Набор операторов языка MapBasic, создающих нужный Вам шаблон. Теперь просто скопируйте часть описания Рабочего Набора, относящуюся к окну Отчета, и вставьте ее в свою программу на MapBasic.

#### **Справочная система**

Интегрированная среда разработки программ на языке MapBasic включает в себя развитую *Справочную систему* (Help). Большая часть ее посвящена операторам и функциям самого языка. Кроме того, она содержит и описание среды MapBasic.



В любой момент в процессе написания программы Вы можете выбрать название оператора или функции и нажать **F1**. Появится окно *Справочной системы* с описанием того оператора или функции, которые Вас интересуют.

---

*Справочная система* содержит много фрагментов программ, которые Вы можете скопировать прямо из окна помощи в свою программу.

Если в момент просмотра окна *Справочной системы* Вы укажете на окно MapBasic, то окно *Справочной системы* исчезнет. Это определяется архитектурой *Справочной системы* Windows. Окно *Справочной системы* при этом не будет закрыто; оно просто окажется "под окном" MapBasic. Чтобы вернуть его "наверх", достаточно нажать **ALT+TAB**. Чтобы окно *Справочной системы* было всегда "над" всеми окнами, следует установить режим **Always on Top** ("Всегда наверху") в окне *Справочной системы*.

## Окно MapBasic в MapInfo Professional

MapInfo Professional позволяет получить доступ к окну MapBasic, чтобы помочь вам в изучении синтаксиса операторов языка MapBasic.

Чтобы открыть окно MapBasic:

1. Запустите MapInfo.
2. Выполните команду **Настройки > Показать окно MapBasic**.

На экране появится окно MapBasic. По мере выполнения различных операций в MapInfo Professional, в окне MapBasic будут появляться соответствующие операторы языка MapBasic.

Например, при составлении запроса в MapInfo с помощью диалога **Выбрать** в окне MapBasic автоматически появится оператор, позволяющий выполнить такой же запрос с помощью языка MapBasic.

Вы можете вводить свои операторы в окно MapBasic (хотя не все они будут работать в подобном диалоговом режиме). Чтобы узнать, можно ли выполнить оператор в окне MapBasic, изучите соответствующую главу в *Справочнике MapBasic* или в *Справочной системе*.

Операторы, которые не работают в окне MapBasic в среде MapInfo, имеют в описании раздел **"Предупреждение"**. В основном это управляющие операторы (такие, как **For...Next**).

Окно MapBasic можно использовать и для отладки. Дополнительную информацию содержит **Глава 5: Поиск ошибок и отладка программ**.

# Работа в интегрированной среде разработки программ

Комплект поставки MapBasic включает текстовый редактор, в котором Вы можете писать свои программы. Общепринятые средства (такие как отмена/повтор и копирование/ вырезание/вставка) упрощают редактирование программ. Кроме того, там же Вы можете компилировать (и, возможно, собирать или "линковать") программы в исполняемые файлы. Программирование поддерживается обширной *Справочной системой*.

Все вместе: текстовый редактор MapBasic, компилятор языка MapBasic и *Справочная система* по языку MapBasic составляют среду разработки.

## В этой главе:

- ♦ Введение в интегрированную среду MapBasic .....30
- ♦ Как отредактировать программу .....30
- ♦ Компиляция программ .....33
- ♦ Сборка приложения из нескольких модулей .....36
- ♦ Обзор меню среды разработки программ MapBasic .....41

## Введение в интегрированную среду MapBasic

Основой интегрированной среды разработки программ на языке MapBasic является текстовый редактор, позволяющий создавать и редактировать программы на MapBasic. Меню редактора – **Файл, Правка, Поиск, Сборка, Окно и Справка** – позволяют выполнять все необходимые операции по редактированию и компиляции программ, поиску и устранению ошибок, обнаруженных компилятором MapBasic.

Если Вы работали с текстовыми редакторами в Windows, у Вас не возникнет проблем с редактором MapBasic. Содержимое большинства его меню является стандартным: меню **Файл** содержит команды **Открыть, Закрыть, Печатать и Сохранить**; меню **Правка** – команды **Отменить, Вырезать, Копировать, Вставить**. Конечно, помимо этих операций MapBasic позволяет выполнять еще и особые действия (например, компиляцию).

## Как отредактировать программу

Если Вы еще не запустили MapBasic, сделайте это, указав дважды на иконку MapBasic. Затем, из меню **Файл** выполните команду **Открыть** (чтобы открыть существующую программу) или **Новый** (чтобы открыть пустое окно).

Наберите в появившемся окне текст программы. Если Вы просто экспериментируете, можете набрать простейшую программу на MapBasic:

```
Note "Привет от MapBasic!"
```

Набрав программу, сохраните ее на диске командой **Сохранить** из меню **Файл**. Дайте Вашей программе имя, например, WELCOME.MB.

MapBasic автоматически добавляет расширение "MB" к имени файла. Поэтому Вы можете просто набрать WELCOME, а на самом деле полное имя файла будет WELCOME.MB.

Поскольку MapBasic хранит программы в стандартном текстовом файле, их можно редактировать в любом текстовом редакторе.


## Клавишные сочетания

Следующая таблица перечисляет "горячие" клавиши, которые Вы можете использовать внутри окна редактирования MapBasic.

Клавиши	Результат
HOME / END	Переход в начало или конец строки
CTRL+HOME/ CTRL+END	Переход в начало или конец документа
CTRL+TAB/ CTRL+SHIFT+TAB	Переход вперед или назад на одно слово
CTRL+T	Вызов диалога <b>Перейти к строке</b>


Клавиши	Результат
<b>CTRL+O</b>	Вызов диалога <b>Открыть</b>
<b>CTRL+N</b>	Открывается новое окно редактора
<b>CTRL+S</b>	Сохранение содержимого активного окна
<b>CTRL+P</b>	Печать содержимого активного окна
<b>CTRL+A</b>	Выбор всего текста в окне
<b>CTRL+C</b>	Копирование выбранного текста в Буфер обмена
<b>CTRL+X</b>	Удаление выбранного текста и копирование его в Буфер обмена
<b>CTRL+V</b>	Вставка текста из Буфера обмена
<b>CTRL+DEL</b>	Удаление следующего после курсора слова
<b>DEL</b>	Удаление выбранного текста без копирования в Буфер Обмена
<b>CTRL+F</b>	Вызов диалога <b>Найти и Заменить</b>
<b>CTRL+G</b>	Повторяет самую последнюю команду поиска
<b>CTRL+R</b>	Заменяет выбранный текст (с использованием текста замены из диалогового окна <b>Найти и Заменить</b> ) и выполняет следующий поиск
<b>CTRL+J</b>	Вызов диалога <b>Выбор проекта</b>
<b>CTRL+K</b>	Компилирует программу в активном окне
<b>CTRL+E</b>	Команда <b>Следующая ошибка</b> ; пролистывает окно редактирования, чтобы показать строку, которая вызвала ошибку трансляции
<b>CTRL+L</b>	Сборка программы
<b>CTRL+U</b>	Посылает сообщение MapInfo, чтобы выполнить активную программу
<b>F1</b>	Открывает <i>Справочную систему</i>
<b>F8</b>	Отображает окно <b>Стиль текста</b>
<b>CTRL+F4</b>	Закрывает активное окно редактирования
<b>ALT+F4</b>	Выход из среды разработки MapBasic

Клавиши	Результат
<b>SHIFT+F4</b>	Расположение окон мозаикой
<b>SHIFT+F5</b>	Расположение окон стопкой

 Если Вы выбираете имя функции перед нажатием **F1**, окно *Справочной системы* откроется на разделе, описывающем эту функцию.

### Операции с использованием мышки

Действие с мышкой	Результат
<b>Двойной щелчок мышкой</b>	Выбирает слово. Двойное указание в списке сообщений об ошибках приводит к пролистыванию окна, чтобы показать строку программы, которая вызвала ошибку.
<b>Тройной щелчок мышкой</b>	Высвечивает всю строку текста (только 32-битной версии).
<b>Перенос мышкой</b>	Перемещение текста в другое окно копирует текст. Перемещение текста внутри того же самого окна перемещает текст. Если Вы держите нажатой клавишу <b>CTRL</b> нажатой во время перемещения, происходит копирование.

 Интерактивная *Справочная Система MapBasic* содержит примеры программ.

Вы можете перемещать мышкой фрагменты кода из окна Справки в окна редактирования.

1. Вызовите Справку, нажав F1.
2. В окне Справки выберите текст, который Вы хотели бы скопировать.
3. Щелкните по выбранному тексту. Не отпуская кнопку мыши, переместите текст за пределы окна Справки.
4. Наведите курсор мыши на редактируемое окно и отпустите кнопку мыши. Теперь текст перенесен в вашу программу.

### Ограничения текстового редактора MapBasic



Окно редактирования MapBasic может поместить только ограниченное количество текста. Если раздастся специальный звуковой сигнал при попытке вставить текст, значит, окно заполнено.

Есть три пути обойти это ограничение:

- Используйте другой текстовый редактор. Для компиляции переключитесь на MapBasic и выберите команду *Компиляция из файла*.
- Можно разделить файл программы (MB-файл) на два и более мелких файлов и использовать оператор MapBasic **Include** для включения различных файлов в одно приложение. Более подробную информацию об операторе **Include** смотрите в *Справочнике MapBasic*.
- Можно разделить файл программы (MB-файл) на два и более мелких файлов и затем создать *файл проекта* MapBasic, который связывает различные файлы программ в одно приложение. В некотором смысле, это действие аналогично применению оператора **Include** для объединения модулей программы. Это более эффективно, чем второе решение, с оператором **Include**. Каждый файл, включенный в проект, компилируется отдельно; это означает, что когда Вы редактируете отдельный модуль, то и перекомпилировать надо будет только этот модуль.

## Компиляция программ

Итак, Вы открыли окно с текстом программы. Чтобы откомпилировать программу, выполните команду **Компилировать файл** из меню **Сборка**.



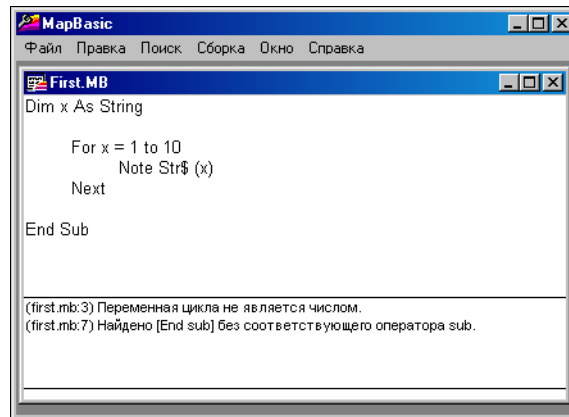
Вы можете открыть одновременно несколько окон с текстами разных программ. По команде **Компилировать файл** MapBasic компилирует ту программу, которая содержится в "самом верхнем", т.е. активном окне. Таким образом, чтобы откомпилировать одну из нескольких открытых программ, Вам нужно сначала сделать окно этой программы активным.

---

Компилятор языка MapBasic проверяет синтаксис программы. Если найдены ошибки синтаксиса, MapBasic покажет соответствующее сообщение и – внизу окна – перечень всех обнаруженных ошибок.

Каждое сообщение об ошибке начинается с номера строки, в которой обнаружена ошибка. Чтобы компиляция прошла успешно, Вам надо исправить все ошибки.

Рисунок: First.mb



Если Вы дважды укажете мышкой на сообщении об ошибке в нижней части окна, MapBasic пролистает окно так, чтобы строка с ошибкой стала видна на экране.

После исправления ошибок, снова выполните команду **Компилировать файл**. Если больше ошибок не найдено, MapBasic покажет сообщение о том, что компиляция закончена успешно.

В случае успешного завершения, компилятор MapBasic создает MBX-файл (MapBasic eXecutable). Для запуска готового приложения, необходим этот файл MBX. Так что, если Вы хотите передать пользователю приложение на MapBasic, но не исходные тексты, отошлите только файл MBX (без файлов MB).

## Об ошибках при компиляции

Существует несколько типов синтаксических ошибок, которые не могут быть выявлены компилятором MapBasic. Например, компилятор MapBasic выдаст сообщение об успешной компиляции следующей программы, хотя она и содержит опечатку во второй строке (вместо STATES набрано TATES):

```
Open Table "states"  
Map From tates
```

Компилятор MapBasic не может зафиксировать опечатку во второй строке. Это не недостаток компилятора; просто это результат того, что проверка существования некоторых переменных и таблиц осуществляется лишь при запуске программы на выполнение. Когда пользователь запустит приведенную выше программу, MapInfo попытается выполнить оператор `Map From TATES`. MapInfo Professional выдаст сообщение об ошибке (например, "Таблицу TATES открыть не удалось"), если только в программе не содержалось других операторов, открывающих таблицу с названием TATES.

## Запуск откомпилированного файла

Чтобы запустить откомпилированную программу, выполните команду **Запустить программу MapBasic** из меню MapInfo Professional. В диалоге **Запустить программу MapBasic** выберите файл с приложением на MapBasic (MBX-файл), который следует запустить.

Запустить программу можно и из среды MapBasic: после успешной компиляции программы следует выполнить команду **Запустить** из меню **Сборка** (или нажать **CTRL+U**). MapBasic пошлет сообщение MapInfo Professional о том, что MapInfo Professional следует запустить приложение.



Программа MapInfo Professional должна быть уже запущена.

---

## Написание программ на MapBasic в других редакторах

Если Вы предпочитаете работать в каком-то уже известном Вам текстовом редакторе, можно использовать его для создания программ на MapBasic. Только тогда сохраняйте программу в виде обычного текстового файла.

Большие текстовые процессоры, как правило, хранят тексты в собственных форматах. Вам следует убедиться, что Ваша программа будет сохранена как стандартный текстовый файл. Чаще всего для этого следует выполнить команду **Save As (Сохранить как)** вместо **Save (Сохранить)**. Подробнее о том, как сохранить файл в стандартном текстовом формате, написано в документации к соответствующему текстовому редактору.

### Компиляция программ, созданных во внешнем редакторе

Мы уже обсуждали использование команды **Компилировать файл** в MapBasic для компиляции программ из активного окна. MapBasic также предлагает альтернативные методы компиляции программы: **Файл > Компиляция из файла**.

Если Вы создавали программу во внешнем текстовом редакторе, Вы можете использовать команду **Компиляция из файла**. Эта команда (**Компиляция из файла**) компилирует программу, не показывая ее в окне редактора MapBasic.

При выполнении команды **Компиляция из файла** MapBasic предлагает Вам выбрать файл, который следует компилировать. Если в файле найдены ошибки, MapBasic запишет сообщения об ошибках в текстовый файл с расширением ERR. Например, если Вы выполнили команду **Компиляция из файла** для компиляции программы DISPATCH.MB, то сообщения об ошибках MapBasic сохранит в текстовом файле DISPATCH.ERR. Чтобы просмотреть этот файл, выполните команду **Файл > Открыть**.

### Компиляция и сборка программ с использованием командной строки

Если Вы используете внешний редактор для создания программ на MapBasic, то Вам может показаться неудобным переключаться на окно MapBasic каждый раз, когда требуется выполнить компиляцию или сборку. Поэтому предусмотрена возможность автоматического запуска компиляции и сборки с использованием командной строки, так что Вы можете выполнять компиляцию, не покидая внешний текстовый редактор.

Интегрированная среда разработки MapBasic активизируется командой:

```
mapbasic
```

Если командная строка также содержит параметр `-D`, за которым следует одно или несколько имен программ, то MapBasic автоматически компилирует эти программы. Например, следующая командная строка запускает MapBasic и компилирует две программы ("Main" и "Sub1"):

```
mapbasic -D main.mb subl.mb
```

Если же командная строка содержит параметр `-L`, за которым следует одно или несколько имен файлов проектов, то MapBasic автоматически собирает соответствующие программы. Сборка программ и файлы проектов обсуждаются в следующем разделе этой главы, [Компиляция и сборка проекта на стр. 38](#). Скажем, следующая командная строка выполняет сборку программы "TextBox":

```
mapbasic -L tbproj.mbp
```

Командная строка может содержать параметры `-D` и `-L` одновременно, как в примере ниже:

```
mapbasic -D textbox.mb -L tbproj.mbp
```

Если Вы запускаете MapBasic с помощью командной строки, содержащей параметр `-D` или `-L`, то окно MapBasic закрывается после компиляции или сборки соответствующих файлов.

Для того чтобы запустить программу MapBasic без стартовой заставки, воспользуйтесь параметром `-Nosplash`:

```
mapbasic -Nosplash
```

## Сборка приложения из нескольких модулей

### Что такое файл проекта в среде MapBasic?

Файл проекта – это текстовый файл, который описывает порядок объединения нескольких программных файлов (модулей) в единое приложение MapBasic. Если Вы разрабатываете большое и сложное приложение, то Ваша программа может содержать тысячи строк кода. Вы можете вводить всю эту программу в один файл. Однако большинство программистов не любит работать с большими программными модулями; при большом количестве кода они предпочитают разбивать его на сравнительно небольшие части (модули). Такой метод программирования известен под названием модульного.

Если Вы разбили текст на несколько модулей, Вы можете создать файл проекта. Такой файл содержит указания MapBasic, как собирать модули в приложение.

Создавать файл проекта не обязательно. Вы можете создать, откомпилировать и запустить приложение без использования файла проекта. Однако для отладки и поддержки сложного приложения на MapBasic лучше использовать преимущества, предоставляемые файлами проектов.

### В чем преимущество использования файла проекта?

- **Файл проекта позволяет Вам строить модульные программы.** Создав файл проекта, Вы можете разделять свою программу на любое количество небольших модулей. Модульные программы в целом легче поддерживать в долгосрочной перспективе. Кроме того, модульная программа позволит Вам обойти 64–килобайтный предел размера файла в окне MapBasic.
- **Использование файла проекта упрощает совместную работу** нескольких программистов над одним приложением. Создав такой файл, можно каждому из них выделить свой модуль, порядок связывания ("сборки") которых определяется в файле проекта.
- **Применение файла проекта может сократить время перекомпиляции** приложения. Если изменения внесены только в один из нескольких модулей, Вы можете перекомпилировать только этот модуль, а затем выполнить сборку нового варианта приложения. Это значительно быстрее, чем каждый раз перекомпилировать всю программу, что Вам придется делать, если не используется построение программы из модулей и файла проекта.

### Примеры файлов проектов

Программа "TextBox" использует файл проекта (TBPROJ.MBP) следующего вида:

```
[Link]
Application=textbox.mbx
Module=textbox.mbo
Module=auto_lib.mbo
```

А программа "ScaleBar" использует файл проекта (SBPROJ.MBP) следующего вида:

```
[Link]
Application=scalebar.mbx
Module=scalebar.mbo
Module=auto_lib.mbo
```

В обоих примерах последняя строка указывает на то, что MapBasic должен включить в приложение AUTO\_LIB. Это стандартный модуль, поставляемый с MapBasic.

Если в MapBasic-программу включить модуль AUTO\_LIB, то в диалог **About...** этой программы может быть добавлена специальная кнопка **Auto-Load** ("Автозапуск"). Нажатие кнопки **Auto-Load** позволяет пользователю указать, что данное приложение должно запускаться автоматически каждый раз, когда запускается MapInfo. Если не включать автозапуск, приложение MapBasic не будет автоматически загружаться в следующем сеансе работы с MapInfo.

Чтобы включить автозапуск в Вашу программу на MapBasic, изучите текст файла AUTO\_LIB.MB.

## Создание файла проекта

Если Вы уже создали все модули и хотите объединить их в файл проекта, выполните следующие шаги:

1. Выполните команду **Файл > Новый**, чтобы открыть новое окно программы.
2. Введите в появившемся окне:  
[Link]
3. Введите строку вида: `Application=appfilename` (где *appfilename* – это имя исполняемого файла). Например:  
`Application=C:\MB\CODE\CUSTOM.MBX`  
`Application=Local:MapBasic:custom.mbx`  
`Application=/MapBasic/mb_code/custom.mbx`
4. Добавьте строку с текстом `Module=modulename` (где вместо *modulename* укажите имя объектного файла MapBasic). Например:  
`Module=C:\MB\CODE\CUSTOM.MBO`  
`Module=Local:MapBasic:custom.mbo`  
`Module=/MapBasic/mb_code/custom.mbo`

Обратите внимание на расширение имени файла; объектный файл MapBasic имеет расширение MBO. MapBasic создает объектный файл при компиляции модуля, являющегося частью многомодульного проекта.

При выполнении команды **Сборка > Компилировать файл** MapBasic пробует скомпилировать активный файл в исполняемый файл (с расширением MBX). Однако, если в тексте компилируемого файла имеются вызовы внешних процедур или функций, MapBasic не может создать MBX-файл; в этих случаях MapBasic предполагает, что файл является частью проекта, и строит объектный файл (MBO) вместо исполняемого (MBX). Кроме того, MapBasic создает объектный файл, если компилируемый файл не содержит процедуры "Main".

5. Повторите **шаг 2** для каждого модуля Вашего приложения.
6. Выполните команду **Файл > Сохранить копию**, чтобы сохранить файл проекта.  
В диалоге **Сохранить файл** выберите тип **Проект** (из списка в левом нижнем углу диалога) так, чтобы Ваш файл получил расширение MBP (MapBasic Project).
7. Закройте окно редактирования (командой **Файл > Закреть** или, указав дважды на кнопку контрольного меню окна).

Если позднее Вы будете добавлять в проект новые файлы, не забудьте внести для них строки вида "Module=" в файл проекта.

## Компиляция и сборка проекта

Создав файл описания проекта, Вы можете компилировать и собирать приложение следующим образом:

1. Откомпилируйте каждый модуль, входящий в приложение.
  - Для этого выполните команду **Файл > Открыть**, а затем **Сборка > Компилировать файл**.
  - Чтобы откомпилировать модуль без открытия для него окна, выполните команду **Файл > Компиляция из файла**.

2. Выполните команду **Сборка > Выбрать файл проекта**, чтобы указать, какой файл проекта должен использовать MapBasic при сборке. Откроется диалог **Выбрать файл проекта**.
3. Выберите название файла проекта (MBP) и нажмите **ОК**. Заданный файл проекта появится в новом окне редактирования. Этот файл будет активным до тех пор, пока Вы не закончите сеанс работы в MapBasic, не закроете окно, содержащее файл проекта или не выполните еще раз команду **Сборка > Выбрать файл проекта**. В каждый момент времени может быть выбран только один файл проекта.



Файл проекта нельзя выбрать, просто сделав активным содержащее его окно. Файл проекта нельзя выбрать, выполнив команду **Файл > Открыть**. Чтобы выбрать файл проекта, выполните команду **Сборка > Выбрать файл проекта**.

---

4. Чтобы собрать приложение, выполните команду **Сборка > Собрать проект**. MapBasic проанализирует все объектные файлы (MBO), перечисленные в файле описания проекта. Если при сборке не выявлено ошибок, MapBasic построит исполняемый (MBX) файл. При обнаружении ошибок сборки, MapBasic выдаст соответствующее сообщение.

Выполнить сборку приложения можно и за один шаг, без явного открытия файла описания проекта, с помощью команды **Файл > Сборка из файла**.

Объектный файл, созданный MapBasic, не может участвовать в сборке приложения, которое проводится сборщиком других пакетов, например, сборщиком языка C. Только сборщик MapBasic может работать с этим объектным файлом.

### Открытие нескольких файлов

При использовании файла проекта в некоторых случаях приходится открывать все входящие в него файлы. Для упрощения в окне диалога **Открыть** имеется возможность сделать это “разом”.

Для того чтобы открыть несколько файлов за один раз:

1. Выполните команду меню **Файл > Открыть**.
2. Выберите имя файла в появившемся окне диалога.
3. Держите нажатой клавишу **SHIFT** или клавишу **CTRL** во время выбора следующего имени.  
Клавиша **SHIFT** позволяет выбирать подряд несколько имен файлов из списка,  
Клавиша **CTRL** – добавлять новые имена к уже выбранным, по одному за раз.

## Вызов функций и процедур из других модулей

Если программа состоит из нескольких модулей, то они могут обращаться к функциям или процедурам друг друга. Например, TEXT BOX.MB вызывает процедуру HandleInstallation, тело которой содержится в библиотеке AUTO\_LIB. Вызов функций или процедур из другого модуля называется вызовом внешней функции (процедуры) или внешней ссылкой.

Если модуль на MapBasic содержит внешнюю ссылку, то такая процедура должна быть объявлена с помощью оператора **Declare Sub**. Аналогично, внешняя функция должна быть объявлена оператором **Declare Function**. Операторы **Declare** нужны для правильной передачи параметров процедуры (функции) между модулями.

Модуль TEXTBOX.MB содержит оператор `Include "AUTO_LIB.DEF"`. Файл заголовков AUTO\_LIB.DEF содержит несколько операторов **Declare Sub** и **Declare Function**, относящихся к модулю AUTO\_LIB. Если в TEXTBOX.MB не включить файл заголовков AUTO\_LIB.DEF, то компилятор MapBasic посчитает вызов функции HandleInstallation синтаксической ошибкой ("Неправильное имя sub-процедуры").

### Глобальные (общие) переменные

Чтобы объявить глобальную переменную, которую можно использовать в различных модулях приложения:

4. Объявите ее оператором **Global** в файле заголовков (например, в GLOBALS.DEF).
5. С помощью оператора **Include** включите такой файл заголовков во все модули, где Вы хотите использовать данную переменную.

Например, файл заголовков AUTO\_LIB.DEF содержит объявления двух глобальных переменных типа String (строковых): "gsAppfilename" и "gsAppDescription". Модули AUTO\_LIB.MB и TEXTBOX.MB оба содержат оператор:

```
Include "auto_lib.def"
```

Таким образом, оба эти модуля имеют доступ к указанным глобальным переменным. Если в TEXTBOX.MB присвоить значение одной из указанных глобальных переменных, библиотека AUTO\_LIB.MB будет использовать это новое значение.

Глобальные переменные позволяют также обмениваться данными между работающими программами через механизм DDE.

### Локальные переменные

Программный модуль может содержать операторы **Dim**, располагающиеся вне функций и процедур. Такие операторы **Dim** объявляют локальные переменные. Все функции и процедуры этого модуля (т.е. MB-файла) имеют доступ к локальной переменной. Однако к ней нельзя обратиться из другого модуля.

Используйте эту возможность определения локальных переменных оператором **Dim**, если требуется переменная, доступная всем процедурам и функциям из данного файла и недоступная из других модулей, чтобы избежать конфликта из-за появления одинаковых имен у переменных).



## Обзор меню среды разработки программ MapBasic

### Меню Файл

Меню **Файл** содержит команды, позволяющие создавать, открывать, закрывать, сохранять и печатать программы на языке MapBasic, а также закончить сеанс работы.

- **Новый** – открывает новое окно, в которое можно вводить текст новой программы.
- **Открыть** – открывает окно с текстом уже существующей программы на MapBasic. Команда открывает окно с текстом уже существующей программы на MapBasic (например, DISPATCH.MB), списка сообщений об ошибках (DISPATCH.ERR) или текста файла Рабочего Набора (например, MAPINFOW.WOR). Напоминаем, что файл описания Рабочего набора – это просто текстовый файл с операторами языка MapBasic.

В окне диалога **Открыть** имеется возможность открыть сразу несколько файлов. Для этого при выборе имен файлов держите нажатой клавишу SHIFT или клавишу CTRL.



Файлы очень большого размера нельзя открыть в текстовом окне MapBasic. Способы обхода этого ограничения приведены в разделе **Ограничения текстового редактора MapBasic на стр. 32**.

---

- **Заккрыть** – закрывает активное окно редактирования. Если Вы вносили в него изменения, MapBasic запросит: сохранить ли эти изменения, прежде чем закрыть окно. Команда **Заккрыть** доступна, если открыто хотя бы одно окно.
- **Заккрыть все** – закрывает все открытые окна. Как и при выполнении команды **Заккрыть**, MapBasic предлагает Вам в диалоге сохранить изменения или отказаться от них. Команда **Заккрыть все** доступна, если хотя бы одно окно открыто.
- **Сохранить** – сохраняет содержимое активного окна на диске. Команда **Сохранить** доступна, если вносились какие-либо изменения.
- **Сохранить в** – сохраняет содержимое активного окна в новом файле на диске. Команда **Сохранить в** доступна, если хотя бы одно окно открыто.
- **Восстановить** – отменяет все изменения, внесенные со времени последнего сохранения содержимого окна в файл на диске. Команда **Восстановить** доступна, если вносились какие-либо изменения.
- **Компиляция из файла** – компилирует существующий MB-файл, считывая его прямо с диска, без открытия текстового окна для этого файла. (В отличие от команды **Компилировать файл** из меню **Сборка**, которая компилирует модуль из активного текстового окна.) Используйте команду **Компиляция из файла** для компиляции программ, написанных во внешнем текстовом редакторе. При обнаружении ошибок в программе, компилируемой командой **Компиляция из файла**, сообщения об ошибках сохраняются в текстовом файле с названием *имя\_файла*.ERR. Чтобы просмотреть этот файл, выполните команду **Файл > Открыть**.
- **Сборка из файла** – собирает приложение на основании заданного файла описания проекта без показа этого файла в текстовом окне. (В отличие от команды **Собрать проект** из меню **Сборка**, собирающей текущий проект.)
- **Настройка печати** – задает режимы печати (например, номер принтерного порта).

- **Печатать** – печатает содержания активного окна.  
Команда **Печатать** активна, если хотя бы одно окно открыто.
- **Выход** – завершает сеанс работы в среде MapBasic. MapBasic выдаст запрос, сохранять ли внесенные изменения.

## Меню Правка

Меню **Правка** содержит команды, позволяющие редактировать программу на MapBasic.

- **Отменить** отменяет результат последнего действия в активном текстовом окне. При выполнении команды **Отменить**, MapBasic удаляет результат последнего действия, а в меню этот пункт превращается в **Повторить**. Выполнение команды **Повторить**, возвращает отмененные изменения.  
  
Команда **Отменить** активна, если хотя бы одно окно открыто и внесено хотя бы одно изменение.
- **Вырезать** – переносит выбранный текст в Буфер обмена (Clipboard), удаляя его из текстового окна. Помещенный в Буфер Обмена текст можно потом вставить с помощью команды **Вырезать** (см. ниже). Команда **Вырезать** активна, если в одном из окон выбран текст.
- **Копировать** – копирует выбранный текст в Буфер обмена, не удаляя его с экрана. Команда **Копировать** активна, если в одном из окон выбран текст.
- **Вставить** – вставляет содержимое Буфера обмена в активное окно с позиции, где находится курсор. Если Вы выберете текст и выполните команду **Вставить**, то текст из Буфера Обмена заменит выбранный текст.  
  
Команда **Вставить** активна, если есть текст в Буфере Обмена и хотя бы одно окно открыто.
- **Удалить** – удаляет выбранный текст. Команда **Удалить** активна, если в одном из окон выбран текст.
- **Выбрать все** – выбирает все содержимое активного текстового окна. Команда **Выбрать все** активна, если открыто хотя бы одно окно.

## Меню Поиск

Меню **Поиск** позволяет проводить поиск и замену текста, а также облегчает поиск ошибок, а также облегчает поиск ошибок.

- **Найти** – осуществляет поиск заданной текстовой строки в активном окне. Команда **Найти** доступна, если открыто хотя бы одно окно. Чтобы найти заданную строку поступайте следующим образом: наберите строку, которую Вы хотите найти, в диалоге **Найти**. Если Вы хотите в процессе поиска различать большие и малые буквы, установите флажок **Различать строчные и прописные**.  
  
Нажмите на кнопку **Найти**. MapBasic начнет поиск с текущей позиции курсора. Если заданная строка будет найдена, MapBasic пролистает окно до того места, где была найдена эта строка. Если строка не найдена, MapBasic подаст звуковой сигнал.
- **Найти ещё** – ищет следующее совпадение со строкой, заданной в предыдущем диалоге поиска.  
  
Команда **Найти ещё** доступна если открыто хотя бы одно окно и операция поиска была уже однажды выполнена.

- **Заменить и найти ещё** – заменяет заданный в диалоге **Найти** текст, а затем ищет и выделяет следующее совпадение с заданной строкой.  
**Следующая ошибка** – позволяет находить и исправлять синтаксические ошибки. Если в программе обнаружены ошибки, MapBasic показывает список ошибок в нижней части окна. Команда **Следующая ошибка** листает окно до той строки программы, где была обнаружена заданная ошибка. Команда **Следующая ошибка** активна, если при компиляции были обнаружены ошибки.
- Команда **Предыдущая ошибка** похожа на команду **Следующая ошибка**, но листает окно назад, к предыдущей ошибке. Команда **Предыдущая ошибка** активна, если при компиляции были обнаружены ошибки.
- **Перейти к строке** – позволяет пролистать текстовое окно до строки с заданным номером. Возможно, что Ваша программа все еще содержит ошибки, хотя компиляция прошла успешно. При возникновении таких ошибок в момент выполнения приложения появляется диалог, в котором указана строка, где появилась ошибка. Обычно, после этого следует вернуться в среду MapBasic и изучить строку с указанным номером. Команда **Перейти к строке** активна, если хотя бы одно окно открыто.

Для замены всех образцов заданной строки поступайте следующим образом:

- наберите в окошке **Заменить на** строку, которой Вы хотите заменить ту, которую Вы задали в окошке **Найти**, и нажмите кнопку **Заменить все**.  
MapBasic заменит все образцы строки из окошка **Найти** на строку из окошка **Заменить на**.



Помните, что эта замена произойдет сразу, без дополнительных сообщений.

---

Чтобы провести замену с подтверждением каждого случая:

1. Выполните команду **Поиск > Найти**. Появится диалог **Поиск и замена**.
2. Заполните окошки **Найти** и **Заменить на**.
3. Нажмите кнопку **Найти**.

MapBasic найдет и выделит первый образец заданной строки.

Для того чтобы заменить этот образец, нажмите Ctrl+R (что аналогично выполнению команды **Заменить и найти ещё**).

Если же Вы не хотите заменить данный образец искомой строки, нажмите Ctrl+G (сокращение команды **Найти ещё**).

## Меню Сборка

Меню **Сборка** содержит команды компиляции и запуска программ на MapBasic, статистики программ и переключения активного окна.

- **Выбрать файл проекта** – открывает диалог, в котором Вы можете выбрать один из имеющихся файлов проектов. Файл описания проекта – это текстовый файл, в котором перечислены модули, из которых состоит приложение. Выполнив данную команду, Вы делаете один из файлов проекта активным и можете собирать соответствующий проект с помощью команды **Собрать проект**.

- **Компилировать файл** – компилирует программу (модуль) из активного окна. Команда **Компилировать файл** активна, если открыто хотя бы одно окно.  
Если компилятор обнаружил синтаксические ошибки, MapBasic покажет список ошибок в нижней части активного окна. Если ошибок не зафиксировано, то MapBasic построит MBX-файл (если компилируется программа, состоящая только из одного модуля) или объектный MBO-файл.
- **Собрать проект** – собирает модули, перечисленные в текущем файле проекта в исполняемый файл приложения (если только не были обнаружены ошибки при сборке). Команда **Собрать проект** активна, если открыт файл проекта.
- **Запустить** – посылает сообщение MapInfo Professional о том, что следует запустить на выполнение приложение из активного текстового окна.
- **Информация** – показывает статистическую информацию о программе из активного окна. **Информация** активна, если открыто хотя бы одно окно.
- **Показать/Скрыть ошибки** открывает или закрывает список сообщений об ошибках для активного окна. Если список ошибок показывается в окне, то меню содержит строку **Скрыть ошибки**. Если список ошибок не показывается, то меню содержит строку **Показать ошибки**. Команда **Показать/Скрыть ошибки** ошибки активны, если имеется окно, которому соответствует хотя бы одно сообщение об ошибке.

## Меню Окно

Если в MapBasic открыто несколько текстовых окон, то в меню **Окно** можно выбрать, какое из них сделать активным.

Команды этого меню активны, если открыто хотя бы одно окно.

- **Разложить все** – располагает окна так, чтобы все они были видны.
- **Сложить в стопку** – располагает окна друг над другом (каскадом).
- **Разложить иконки** – переставляет иконки, соответствующие минимизированным окнам. Чтобы временно свернуть окно в иконку, нажмите на кнопку минимизации справа от строки заголовка.
- **Стиль текста** – позволяет выбрать шрифт, которым следует показывать текст в окне. Этим шрифтом показывается весь текст в окне.
- Нижняя часть меню **Окно** содержит перечень открытых окон. Чтобы сделать окно активным (т.е. поместить его поверх других), выберите название окна в меню **Окно**.

## Меню Справка

Меню **Справка** дает Вам доступ к *Справочной системе* (подсказкам). Файл *Справочной системы* содержит описания всех операторов и функций языка MapBasic. **Справка** также открывает доступ к механизму перекрестных ссылок, который позволяет быстро находить описание нужных операторов.

- **Содержание** – открывает одноименное окно. Для этого Вы указываете на их заголовки или выбирая тему в диалоге (для этого нажмите кнопку **Поиск**).
- **Поиск** – открывает одноименный диалог.
- **Как пользоваться** – открывает тему, описывающую порядок работы с подсказками.
- **Проверка обновлений** – открывает страницу web-сайта Pitney Bowes Software Inc., которая посвящена доступным обновлениям программы.

- **О программе MapBasic** – показывает диалог **О программе MapBasic**, в котором содержится информация об авторских правах и номер версии программы.



Многие подсказки содержат краткие примеры программ. Вы можете скопировать любой из этих примеров в Буфер Обмена, а затем вставить в свою программу. Для этого выполните команду **Правка > Копировать** в окне *Справочной системы* или переместите выбранный фрагмент мышкой в окно Вашей программы.

---

# Основы MapBasic

Каждому, кто собирается программировать на языке MapBasic, рекомендуется прочитать эту главу, в которой описаны основы синтаксиса языка MapBasic.

## В этой главе:

- ♦ Общие замечания о синтаксисе языка MapBasic .....47
- ♦ Выражения.....54
- ♦ Циклы и другие управляющие операторы .....65
- ♦ Процедуры.....69
- ♦ Процедуры-обработчики системных событий.....73
- ♦ Рекомендации по использованию процедур-обработчиков .76
- ♦ Директивы компилятора.....78
- ♦ Организация программ .....79

## Общие замечания о синтаксисе языка MapBasic

Прежде чем рассматривать отдельные операторы языка MapBasic, дадим общее описание синтаксиса программ на MapBasic.

### Комментарии

В MapBasic, как и в некоторых других BASIC-подобных языках, апостроф ( ' ) обозначает начало комментария. Если в программе встретился апостроф, MapBasic понимает оставшуюся часть строки как комментарий, кроме тех случаев, когда апостроф является частью строки символов (символьной константы).

### Строчные и прописные буквы

Компилятор языка MapBasic не различает большие (прописные) и малые (строчные) буквы. Вы можете использовать в своих программах СТРОЧНЫЕ БУКВЫ, прописные буквы или Строчные и Прописные Вместе.

В данном руководстве мы будем придерживаться следующего стиля: первые буквы ключевых слов языка MapBasic будут везде большими (прописными); переменные будут состоять только из маленьких (строчных) букв. Например, в следующем примере программы слова **If** и **Then** начинаются со строчной буквы, поскольку они являются ключевыми словами языка MapBasic, а слово "counter" содержит только малые буквы, так как это – имя переменной.

```
If counter > 5 Then
    Note "Значение counter слишком велико"
End If
```

### Продолжение оператора на нескольких строках

Операторы в программах на MapBasic могут располагаться на нескольких строках. Например, в следующем примере оператор **If\_Then** занимает несколько строк:

```
If counter = 55
    Or counter = 34 Then
    Note "Значение counter неправильное"
End If
```

### Константы-коды, определенные в файле MAPBASIC.DEF

Многие операторы и функции MapBasic не будут работать правильно, если в начале своей программы Вы не поставите строку:

```
Include "mapbasic.def"
```

MAPBASIC.DEF – это текстовый файл, который содержит определения многих стандартных констант MapBasic. Как правило, названия констант, определенных в MAPBASIC.DEF, состоят из больших букв (например, TRUE, FALSE, BLACK, WHITE, CMD\_INFO\_X, OBJ\_INFO\_TYPE и т.д.). При чтении примеров программ в документации по MapBasic Вы встретите много таких кодов. Например:

```
If CommandInfo( CMD_INFO_DLG_OK ) Then
```

Если программа использует стандартные константы (такие, как `CMD_INFO_DLG_OK` в приведенном примере), то в нее следует включить файл `MAPBASIC.DEF` с помощью оператора **Include**. Если пропустить оператор **Include**, то при выполнении программы будет зафиксирована ошибка (например, "Переменная или поле `CMD_INFO_DLG_OK` не определено").

## Как вводить операторы в окно MapBasic

В MapInfo Professional можно открыть окно, которое называется MapBasic. Вы можете использовать окно MapBasic для изучения синтаксиса операторов языка MapBasic. Однако, для окна MapBasic действуют некоторые ограничения:

- В окно MapBasic нельзя вводить некоторые операторы MapBasic, которые Вы тем не менее можете использовать в программах на MapBasic. Общее правило таково: управляющие операторы (такие как **If... Then**, **For... Next** и **GoTo**) не работают в окне MapBasic.
- Чтобы узнать, можно ли использовать тот или иной оператор в окне MapBasic, обратитесь к *Справочнику MapBasic*. Для каждого оператора, который нельзя использовать в окне MapBasic, в *Справочнике* сделано соответствующее замечание.
- Когда Вы вводите оператор непосредственно в Окно MapBasic программы MapInfo Professional, следует соблюдать особые соглашения переноса оператора на следующую строку. Вместо ENTER в таких случаях следует нажимать CTRL+ENTER. После того, как Вы наберете оператор полностью, выберите его и нажмите ENTER.
- Стандартные константы, определенные в `MAPBASIC.DEF` (например, `BLACK`, `WHITE` и т.д.) нельзя использовать в окне MapBasic. Однако каждой такой константе соответствует свое число (код), которое можно найти в файле `MAPBASIC.DEF`; например, константе `BLACK` соответствует число 0. В окне MapBasic Вы можете использовать числовые значения, соответствующие стандартным константам, вместо имен этих констант (т.е. ноль вместо "`BLACK`").
- Каждый оператор, который Вы вводите в окно MapBasic ограничен размером в 256 символов.

## Переменные

Синтаксис объявления переменных и операторов присваивания в языке MapBasic очень похож на синтаксис этих операторов в других современных BASIC-подобных языках. Однако в MapBasic имеется несколько типов переменных, которые отсутствуют в других языках программирования (например, тип `Object`; полный список типов переменных в языке MapBasic можно найти в главе *Справочника MapBasic*, посвященной оператору **Dim**).

### Что такое переменная?

Переменную можно определить как небольшой участок памяти компьютера, отведенный и предназначенный для временного хранения некоторого типа информации. Чтобы отвести подобный участок памяти, надо объявить переменную. Каждая переменная должна иметь уникальное имя (например, "`counter`", "`x`", "`y2`", "`customer_name`"). Для каждой объявленной переменной MapBasic отводит определенный участок в памяти. После этого переменную можно использовать для хранения информации.



## Объявление переменных и присвоение им значений

Для объявления переменных используется оператор **Dim**. Каждую переменную следует объявить, причем объявление переменной должно предшествовать ее использованию.

Оператор равенства (=) используется, чтобы присвоить переменной значение.

В следующем примере объявляется переменная типа Integer, и ей присваивается значение 23:

```
Dim counter As Integer  
counter = 23
```

Одним оператором **Dim** можно объявить несколько переменных, разделяя их запятыми. Следующий оператор **Dim** объявляет три вещественные переменные с плавающей точкой:

```
Dim total_distance, longitude, latitude As Float  
longitude = -73.55  
latitude = 42.917
```

Также в одном операторе **Dim** можно указывать переменные разных типов. Вот как объявить две переменные типа Date и две переменные типа String:

```
Dim start_date, end_date As Date,  
    first_name, last_name As String
```

## Имена переменных

В именах переменных следует соблюдать следующие правила:

- Имя переменной не должно содержать более 31 символа.
- В имени переменной не может быть пробелов.
- Имя переменной должно начинаться с буквы, подчеркивания (\_) или тильды (~).
- Имя переменной может содержать буквы, цифры, знак фунта(#) и подчеркивание (\_).
- Имя переменной может оканчиваться одним из следующих знаков: \$, %, &, ! или @. В некоторых BASIC-подобных языках эти символы определяют тип переменной; в MapBasic же они не имеют такого смысла.
- Нельзя использовать ключевые слова языка MapBasic в качестве имен переменных. Так, Вы не можете объявить переменные с именами **If**, **Then**, **Select**, **Open**, **Close** или **Count**. Список ключевых слов приводится в описании оператора **Dim** в *Справочнике MapBasic*.

## Типы данных

MapBasic поддерживает следующие типы данных:

Тип	Описание
SmallInt. Короткое целое число.	Целое число от -32767 до 32767; занимает два байта
Integer. Целое число.	Целое число от -2 миллиардов до 2 миллиардов; занимает четыре байта
Float. Вещественное число.	Вещественное число с плавающей точкой; хранится в восьми байтах в формате IEEE
String. Строка.	Строка произвольной длины (до 32767 символов).
String * <i>n</i> . Строка.	Строка фиксированной длины <i>n</i> (до 32767 символов).
Logical. Логическое значение.	TRUE или FALSE.
Date. Дата..	Дата
Object. Объект.	Графический объект; подробнее этот тип описан в главе <a href="#">Географические и графические объекты</a>
Alias. Псевдоним.	Ссылка на колонку таблицы; см. главу <a href="#">Работа с таблицами</a>
Rep. Линия.	Объект типа "Линия"; см. главу <a href="#">Географические и графические объекты</a>
Brush. Штрих.	Объект типа "Штриховка"; см. главу <a href="#">Географические и графические объекты</a>

## Строковые переменные

В языке MapBasic имеются два типа строковых переменных: фиксированной и произвольной длины. Строка произвольной длины может содержать до 32767 символов. Строка же фиксированной длины имеет предел числа символов, указанный в операторе **Dim**.

Чтобы объявить переменную типа String произвольной длины, надо просто применить ключевое слово "String". Чтобы объявить переменную типа String фиксированной длины, после ключевого слова "String" надо поставить звездочку (\*) и максимальный размер строки в байтах. В следующем примере объявляется строковая переменная "full\_name" произвольной длины и строковая переменная "employee\_id" длиной 9 символов:

```
Dim full_name As String,  
employee_id As String * 9
```



Как и в других BASIC-подобных языках, в MapBasic строки фиксированной длины дополняются справа пробелами таким образом, чтобы строка занимала все отведенное место. Так, например, если Вы объявили строковую переменную длиной в пять байт и присвоили ей строку "ABC", то в действительности переменная будет содержать строку "ABC " (т.е. "ABC" и два пробела в конце). Такой метод работы с фиксированными строками применяется для удобства форматного вывода.

## Массивы

Чтобы объявить массив, следует после имени переменной в круглых скобках указать размер (количество элементов) массива. Размер массива должен быть целой положительной константой или выражением из констант. Следующий оператор **Dim** объявляет массив из десяти переменных типа `Date`:

```
Dim start_date(10) As Date
```

Доступ к элементу массива осуществляется следующим образом:

```
array_name(element-number)
```

Так, следующий оператор присваивает значение первому элементу:

```
start_date(1) = "11.06.93"
```

Чтобы изменить размер массива, надо использовать оператор **ReDim**. В тех случаях, когда неизвестно, какого размера массив потребуется в Вашей программе – например, из-за того, что Вы не знаете, сколько данных введет пользователь – увеличивайте при необходимости размер массива с помощью **ReDim**. Текущий размер массива возвращает функция **UBound( )**.

Ниже приводится пример, в котором объявляется массив строковых переменных "name\_list", а затем размер массива увеличивается на десять элементов.

```
Dim counter As Integer, name_list(5) As String  
...  
counter = UBound(names) ' Найти размерность массива  
ReDim names(counter + 10) ' Увеличить размерность массива на 10
```

На массивы в языке MapBasic накладываются следующие ограничения:

- MapBasic поддерживает только одномерные массивы.
- В MapBasic первый элемент массива всегда имеет индекс 1; другими словами, в приведенном только что примере первым элементом массива имен является "names(1)".

Чтобы хранить больше данных, чем может вместить массив, Вы можете использовать для их размещения таблицы. Подробнее об использовании таблиц рассказывает глава **Работа с таблицами**.

MapBasic инициализирует объявленные числовые массивы и переменные, заполняя их нулевыми значениями. Строковые массивы и переменные заполняются пустыми строками.

## Типы данных, заданные пользователем (структуры данных)

Вы можете объявить свой тип данных с помощью оператора **Type... End Type**. Заданный пользователем тип данных представляет собой группу из одного или нескольких стандартных типов. Объявив свой тип данных, Вы можете объявлять переменные этого типа с помощью оператора **Dim**.

Приведем пример программы, в которой задается пользовательский тип данных, "employee" ("сотрудники"), и затем объявляются переменные типа "employee".

```
Type employee
    name As String
    title As String
    id As Integer
End Type
Dim manager, staff(10) As employee
```

Каждая компонента пользовательского типа данных называется полем. То есть тип "employee" в приведенном примере состоит из трех полей: "name", "title" и "id". Синтаксис обращения к полю таков:

```
variable_name.element_name
```

В следующем примере присваиваются значения всем полям переменной "manager":

```
manager.name = "Джо"
manager.title = "Директор издательства"
manager.id = 111223333
```

Вы можете объявить массив заданного (нового) типа. Вы можете объявить массив заданного (нового) типа. Ниже приведены операторы, присваивающие значения некоторым полям первого элемента массива "employee":

```
staff(1).name = "Эд"
staff(1).title = "Грузчик"
```

Операторы **Type... End Type** должны находиться вне тела процедуры. О процедурах речь пойдет далее в этой главе. Как правило, операторы **Type... End Type** размещают в начале программы. В операторе **Type** можно использовать поля любого типа, в том числе и ранее определенных пользовательских типов данных. Можно объявлять глобальные переменные и массивы пользовательских типов.

## Глобальные переменные

Переменные, объявленные оператором **Dim**, являются локальными. Локальные переменные можно использовать только внутри той процедуры, где они были объявлены. Наряду с этим MapBasic позволяет Вам объявлять глобальные переменные, которые можно использовать в любой процедуре, повсюду в Вашей программе.

Объявить глобальную переменную можно оператором **Global**. Синтаксис оператора **Global** похож на синтаксис оператора **Dim**, только вместо ключевого слова **Dim** употребляется ключевое слово **Global**. Следующий оператор **Global** объявляет две глобальные переменные типа Integer:

```
Global first_row, last_row As Integer
```

Операторы **Global** должны находиться вне тела процедуры. О процедурах речь пойдет далее в этой главе. Как правило, операторы **Global** размещают в начале программы.

Приведем пример программы, в которой объявляются несколько глобальных переменных, а затем эти переменные используются в процедурах.

```
Declare Sub Main
Declare Sub initialize_globals
Global gx, gy As Float ' объявили глобальные переменные типа Float
Global start_date As Date ' объявили глобальную переменную типа Date
Sub Main
    Dim x, y, z As Float ' объявили локальные переменные процедуры Main
    Call initialize_globals
    ...
End Sub
Sub initialize_globals
    gx = -1 ' назначение значений глобальным переменным
    gy = -1 ' назначение значений глобальным переменным
    start_date = CurDate() ' назначение значений глобальным переменным
End Sub
```

По возможности следует использовать локальные, а не глобальные переменные, так как память под глобальные переменные отводится на все время выполнения Вашей программы. Под локальные же переменные MapBasic отводит память только на время выполнения процедуры, в которой они объявлены.

Глобальные переменные можно использовать для обмена информацией с другими приложениями. Приложения для Windows используют механизм так называемого динамического обмена данными (Dynamic Data Exchange – DDE) для чтения и модификации глобальных переменных MapBasic-программ.

## Область определения переменных

В процедуре можно объявить локальную переменную с тем же именем, что и некоторая глобальная переменная. Даже если имеется, например, глобальная переменная "counter", в процедуре можно объявить локальную переменную "counter":

```
Declare Sub Main
Declare Sub setup
Global counter As Integer
...
Sub setup
    Dim counter As Integer
    counter = 0
    ...
End Sub
```

В случае совпадения имен локальной и глобальной переменных процедура теряет доступ к глобальной переменной. То есть в теле процедуры будет видима только локальная переменная. В приведенном выше примере оператор `counter = 0` не окажет влияния на значение глобальной переменной "counter".

Обработывая имя переменной, MapBasic пытается интерпретировать его как имя локальной переменной. Если нет локальной переменной с таким именем, MapBasic попытается найти глобальную переменную с таким именем. Если нет глобальной переменной с таким именем, Map Basic попытается найти открытую таблицу с таким именем. И, наконец, если во время выполнения программы не найдено открытой таблицы с таким именем, MapBasic выдаст сообщение об ошибке.

## Выражения

В этом разделе мы поговорим о том, что такое выражение. Под выражением мы понимаем группу из одной или нескольких переменных, констант, вызовов функций, имен таблиц или операторов.

### Что такое константа?

Выражение может иметь самый простой вид. Например, оператор: `counter = 23` присваивается значение целочисленного выражения, а именно, константа 23. Выражение 23 будем называть числовой константой. Можно сказать, что константа – это конкретное значение, которое можно присвоить переменной.

Следующий фрагмент программы содержит объявление строковой переменной и присваивание ей строковой константы (имя "Семен Семеныч"):

```
Dim name As String  
name = "Семен Семеныч"
```

Синтаксис числовых выражений отличается от синтаксиса строковых выражений: строковые константы следует заключать в двойные кавычки (как в случае с "Семен Семеныч"), а числовые – нет (например, 23). Значение строковой константы, например, "Семен Семеныч," нельзя присвоить числовой переменной. Подробнее о выражениях из констант см. в разделе **Константы в языке MapBasic на стр. 56**.

### Что такое оператор?

Оператор – это специальный символ (например, +, \*, >) или слово (**And**, **Or**, **Not** и т.п.), который связывает один или несколько параметров (констант, переменных или выражений). Выражение может содержать одно или несколько значений, объединенных оператором. Ниже приводится пример, в котором оператор сложения (+) используется для выполнения сложения в выражении "y + z". Результат сложения (сумма двух значений) присваивается переменной "x":

```
Dim x, y, z As Float  
y = 1.5  
z = 2.7  
x = y + z
```

В этом примере знак плюс (+) действует как оператор, а конкретнее – как числовой оператор. Среди прочих числовых операторов можно назвать минус (-), осуществляющий вычитание; оператор умножения (\*), а также возведения в степень (^). Полный список числовых операторов приводится ниже в этой главе.

Оператор сложения может также использоваться в строковых выражениях для соединения нескольких строк в одну. В следующем фрагменте строка составляется из трех частей и сохраняется в переменную "full\_name":

```
Dim first_name, last_name, middle_init, full_name As String
first_name = "Семен "
middle_init = "Семеныч "
last_name = "Горбунков"
full_name = first_name + middle_init + last_name

' В данный момент переменная full_name содержит строку:
' Семен Семеныч Горбунков
```

## Что такое вызов функции?

Язык программирования MapBasic поддерживает различные виды вызовов функций. Каждая стандартная функция выполняет особое действие, например, функция **Sqr( )** вычисляет квадратный корень от заданного значения, а **UCase\$( )** делает все символы в строке заглавными. Указав имя функции в программе, Вы тем самым определяете вызов функции, которая возвращает какое-либо значение.

Вызов функции может быть частью сложного выражения или единственным его элементом. Например, ниже переменной "x" присваивается минимальное значение, возвращаемое функцией **Minimum( )**:

```
x = Minimum( y, z )
```

Синтаксис вызовов функций в MapBasic таков же, как и в большинстве других современных BASIC-подобных языков. После имени функции (например, "Minimum" в последнем примере) следуют круглые скобки. Если у функции есть параметры, то в скобках перечисляются эти параметры. Если параметров более одного, между ними ставятся запятые (функция **Minimum( )** имеет два параметра).

Особенностью применения вызова функции в операторе является то, что функция возвращает значение. Стоящий отдельно в операторе вызов функции смысла не имеет; значение, возвращаемое функцией, должно как-то использоваться. Так следующий пример программы содержит два оператора: оператор **Dim** объявляет переменную x, а затем ей присваивается значение. Оператор присваивания включает в себя вызов функции **Sqr( )** для вычисления квадратного корня:

```
Dim x As Float
x = Sqr(2)
```

Аналогично, в следующем примере используется функция **CurDate( )**, которая возвращает значение типа Date, соответствующее текущей дате:

```
Dim today, yesterday As Date
today = CurDate( )
yesterday = today - 1
```

Функция **CurDate( )** не имеет параметров. При вызове функции в языке MapBasic, Вы должны ставить круглые скобки после имени функции даже в том случае, когда у функции нет параметров, как показано в последнем примере.

В языке MapBasic имеются многие стандартные функции BASIC подобных языков, такие как **Chr\$( )** и **Sqr( )**, а также различные специальные географические функции, такие как **Area( )**, **Perimeter( )** и т.п.

## Константы в языке MapBasic

Константами мы называем конкретные значения, которые не изменяются в процессе выполнения программы. На жаргоне программистов их называют также “защитыми” выражениями или “литералами.”

### Числовые константы

Различным числовым типам соответствуют различные типы констант. Например, константа 36 – обобщенная числовая константа; ее можно присвоить любой числовой переменной, независимо от того, какого типа переменная – Integer, SmallInt или Float. Значение же 86.4 – вещественная или десятичная константа.

### Шестнадцатеричные константы

В MapBasic 4.0 поддерживаются шестнадцатеричные константы в синтаксисе VisualBasic: *&Hчисло* (где *число* шестнадцатеричное число). В следующем примере переменной присваивается шестнадцатеричное значение 1A (равное десятичному 26):

```
Dim i_num As Integer  
i_num = &H1A
```

Числовая константа не может содержать запятых. Вот пример оператора, который будет работать неправильно:

```
counter = 1,250,000 ' Неправильно!
```

Символом десятичного разделителя должна быть точка, даже если компьютер настроен на другие десятичные разделители.

### Строковые константы

Строковые константы заключаются в двойные кавычки. Например:

```
last_name = "Горбунков"
```

Строковая константа может содержать до 256 символов.

Двойные кавычки не являются частью строковой константы, они просто обозначают ее начало и окончание. Чтобы употребить в строковой константе символ "двойная кавычка", Вам следует вставить в этом месте строки два символа кавычки (") подряд, как это сделано в данном примере:

```
Note "Таблица ""World"" уже открыта."
```



## Логические константы

Логическая константа может принимать одно из двух значений: единица (1), обозначающая истину, и ноль (0), обозначающий ложь. Во многих примерах программ на MapBasic используются значения TRUE и FALSE; на деле TRUE и FALSE – это константы, объявленные в файле заголовков MAPBASIC.DEF. Чтобы использовать стандартные константы TRUE и FALSE, Вы должны включить в свою программу с помощью оператора **Include** файл MAPBASIC.DEF. Например:

```
Include "mapbasic.def"  
Dim edits_pending As Logical  
edits_pending = FALSE
```

## Даты-константы

Можно использовать даты-константы в 8-байтовом формате (YYYYMMDD), где YYYY – это год, MM – месяц, а DD – день. Так, например, можно присвоить дату 31 декабря 1995:

```
Dim d_enddate As Date  
d_enddate = 19951231
```

Дату-константу можно задать в виде строки:

```
d_enddate = "31.12.95"
```

При использовании даты-константы в виде строки можно указать либо все 4 цифры года, либо только две последние :

```
d_enddate = "31.12.95"
```

Если Вы опускаете год, то используется текущий год:

```
d_enddate = "12/31"
```



Использование строковых констант для хранения даты в некоторых случаях оказывается ненадежным, поскольку результат будет зависеть от установок на данном компьютере. Если предполагается дата в виде “Месяц/День/Год”, то строка “06/11/95” будет соответствовать 11 июля 1995 года, если же “День/Месяц/Год”, то получается 6 ноября.

Если компьютер пользователя настроен так, что символ “-” используется в качестве разделителя, MapInfo Professional не сможет конвертировать строковые выражения, такие как “12/31”, в даты.

Чтобы гарантированно избежать неприятностей, используйте функцию **NumberToDate( )**, поддерживающую 8-байтовый формат даты, не зависящий от установок компьютера. (Числовые константы-даты, например, 19951231, не зависят от настроек компьютера.) Если же приходится иметь дело со строковым представлением, например, при чтении даты из текстового файла, используйте оператор **Set Format**. Подробнее об операторе **Set Format** см. в *Справочнике MapBasic* или в *Справочной системе*.

Для конфигурации формата даты используйте настройки даты и времени из Панели Управления Microsoft Windows.

Константы – имена таблиц

Псевдонимы таблиц – переменные типа "Alias", подробно обсуждаются в разделе **Работа с таблицами**. Переменной типа "Alias" можно присваивать строковые выражения. Например:

```
Dim column_name As Alias
column_name = "City"
```

В следующей таблице содержатся примеры констант различных типов.

Типы	Пример	Примечание
Integer. Целое число.	i = 1234567	
SmallInt. Короткое целое число.	m = 90	
Float. Вещест- венное число.	f = 4 size = 3.31 debt = 3.4e9	
String. Строка.	s_mesg = "Семен Семеныч"	Строка должна быть заключена в двойные кавычки. Внутри строки двойные кавычки обозначаются двумя двойными кавычками подряд. Специальные символы можно вставить в строку с помощью функции <b>Chr\$( )</b> .
Logical. Логическое значение.	edits_pending = 1 edits_pending = TRUE	1 = TRUE, 0 = FALSE. Константы TRUE и FALSE задаются в файл определений MapBasic.
Date. Дата..	d_starting = 19940105 date_done = "23.03.88" paidddate = "12-24-1993" yesterday = CurDate( ) - 1	

Типы	Пример	Примечание
Alias. Псевдоним .	col_name = "Pop_1990" col_name = "COL1"	Псевдонимом можно определить строку. Смотрите главу <b>Работа с таблицами</b> , где подробнее обсуждаются переменные типа "Alias".
Pen. Линия.	hwypen = MakePen(1, 3, BLACK)	Выражение только такого вида.
Brush. Штрих.	zbrush = MakeBrush(5, BLUE, WHITE)	Выражение только такого вида.
Font. Шрифт.	lbl_font = MakeFont("Helv", 1, 20, BLACK, WHITE)	Выражение только такого вида.
Symbol. Символ.	loc_sym = MakeSymbol(44, RED, 16)	Выражение только такого вида.
Object. Объект.	path = CreateLine(73.2, 40, 73.6, 40.4)	Выражение только такого вида.

Правила преобразования типов переменных

В языке MapBasic имеется набор функций преобразования типов данных. Например, можно для заданной числовой переменной получить представление ее в виде строки цифр с помощью функции **Str\$( )**:

```
Dim q1, q2, q3, q4, total As Float, s_message As String
...
total = q1 + q2 + q3 + q4
s_message = "Итого: " + Str$(total)
```

Операторы языка MapBasic

Операторы производят некоторое действие над одним или несколькими значениями. Операторы можно классифицировать по типу параметров, к которым они применяются, и по типу результата.

Числовые оператор

Все приведенные ниже в таблице операторы являются числовыми. На основании двух числовых значений они выдают числовой результат.

Оператор	Действие	Пример
+	сложение	x = a + b
-	вычитание	x = a - b
*	умножение	x = a * b
/	деление	x = a / b
\	деление нацело	x = a \ b
Mod	остаток от деления	x = a Mod b
^	возведение в степень	x = a ^ b

Операторы \ и **Mod** осуществляют целочисленное деление. Например:

10 / 8	возвращает	1.25
10 \ 8	возвращает	1 (целую часть числа 1.25)
10 Mod 8	возвращает	2 (остаток от деления 10 на 8)

Оператор вычитания (-) может использоваться для обозначения отрицательного значения:


x = -23

Строковые операторы

Оператор сложения (+) позволяет соединить несколько строк в одну.

Note "Служащий: " + служ\_имя + " " + служ\_фамилия

Можно использовать оператор (&) вместо (+). Этот оператор превращает оба операнда в строки и затем объединяет их в одну. Это не то же самое, что оператор +, который может работать с числами или датами, не выполняя принудительного преобразования в строки.



Оператор (&) используется также для обозначения шестнадцатеричных чисел (&число). Поэтому, используя его для объединения строк, не забывайте про пробелы до и после оператора &, тогда компилятор MapBasic не перепутает его с префиксом шестнадцатеричных чисел .

Оператор **Like** выполняет сравнение строк с заданным шаблоном. Приводимый ниже пример проверяет, начинается ли строковая переменная с подстроки "Север":

If s\_state\_name Like "Север%" Then ...

Оператор **Like** подобен функции **Like( )**. Описание функции **Like( )** см. в *Справочнике MapBasic*.

Операторы для работы с датами

С датами можно использовать операторы сложения и вычитания:

Выражение	Возвращает
<i>дата</i> + <i>целое число</i>	более поздняя дата
<i>дата</i> - <i>целое число</i>	более ранняя дата
<i>дата</i> - <i>дата</i>	целое число: длительность периода

В следующем примере с помощью функции **CurDate( )** находится текущая дата, затем вычисляется завтрашнее число и дата недельной давности:

```
Dim today, one_week_ago, tomorrow As Date,
    days_elapsed As Integer
today = CurDate( )
tomorrow = today + 1
one_week_ago = today - 7
' подсчет дней с первого января:
days_elapsed = today - StringToDate("1/1")
```

Операторы сравнения

Операторы этой группы сравнивают значения (как правило, одного типа) и возвращают логическое значение TRUE или FALSE. Операторы сравнения используются обычно в условных операторах (например, в **If... Then**).

Оператор	Возвращает TRUE, если выполняется условие	Пример:
=	равны	If a = b Then ...
<>	не равны	If a <> b Then ...
<	менее чем	If a < b Then ...
>	более чем	If a > b Then ...
<=	меньше или равно	If a <= b Then ...
>=	больше или равно	If a >= b Then ...
Between... And	в диапазоне между	If x Between f_low And f_high Then...

Каждый из перечисленных операторов сравнения может использоваться в строковых и числовых выражениях, а также в выражениях для дат. Заметим, что операторы сравнения нельзя применять в выражениях типа **Object, Pen, Brush, Symbol** или **Font**.

Оператор сравнения **Between... And** позволяет проверять, попадает ли значение в заданный диапазон. В следующем примере в операторе **If... Then** используется сравнение **Between... And**.

```
If x Between 0 And 100 Then
    Note "Значение попадает в диапазон."
Else
    Note "Значение не попадает в диапазон."
End If
```

Тот же фрагмент можно записать и иначе:

```
If x >= 0 And x <= 100 Then
    Note "Значение попадает в диапазон."
Else
    Note "Значение не попадает в диапазон."
End If
```

При использовании оператора **=** (равно) для сравнения строк, MapBasic проверяет обе строки полностью и возвращает TRUE, если они полностью совпадают. Отметим, что при сравнении строк большие и малые буквы не различаются; так, в следующем операторе **If... Then** два названия города ("Псков" и "ПСКОВ") будут считаться одинаковыми:

```
Dim city_name As String
city_name = "ПСКОВ"
If city_name = "Псков" Then
    Note "Название города совпадает."
End If
```

Для сравнения строк с учетом различия больших и малых букв используйте функцию **StringCompare( )**, описанную в *Справочнике MapBasic*.



Будьте внимательны при сравнении строк фиксированной и произвольной длины. MapBasic автоматически дополняет пробелами каждую строковую переменную фиксированной длины так, чтобы строка занимала все отведенное под нее место. А строки произвольной длины не дополняются таким образом. В связи с этим иногда строки, которые должны были бы по смыслу Вашей задачи быть одинаковыми, будут восприняты как различные.

Чтобы получить вариант строки без завершающих дополнительных пробелов, используйте функцию **RTrim\$( )**. Затем ее можно сравнить с результатом применения функции **RTrim\$( )** к строке произвольной длины, не беспокоясь о влиянии дополняющих пробелов.

## Логические Операторы

Логические операторы применяются к логическим значениям и возвращают TRUE или FALSE:

Оператор	Возвращает TRUE, если выполняется условие	Пример:
<b>And</b>	оба операнда истинны	If a And b Then...
<b>Or</b>	значение одного из операндов – истина	If a Or b Then...
<b>Not</b>	операнд ложен	If Not a Then...

Например, оператор **If...Then** выполняет две проверки: сначала, является ли значение переменной *x* меньше нуля, а затем проверяет, что значение *x* больше десяти. Если проверка не проходит, программа выдает сообщение об ошибке.

```
If x < 0 Or x > 10 Then
    Note "Число вне допустимого диапазона."
End If
```

## Географические операторы

Эти операторы применяются к выражениям типа Object и выдают логический результат TRUE или FALSE.

Оператор	Возвращает TRUE, если выполняется условие	Пример:
<b>Contains</b>	первый объект содержит центроид второго объекта	If a Contains b Then...
<b>Contains Part</b>	часть второго объекта содержится внутри первого объекта	If a Contains Part b Then...
<b>Contains Entire</b>	весь второй объект содержится внутри первого объекта	If a Contains Entire b Then...
<b>Within</b>	центроид первого объекта содержится во втором объекте	If a Within b Then...
<b>Partly Within</b>	второй объект содержит часть первого объекта	If a Partly Within b Then...
<b>Entirely Within</b>	второй объект полностью содержит первый объект	If a Entirely Within b Then...
<b>Intersects</b>	два объекта пересекаются хотя бы в одной точке	If a Intersects b Then...

Более подробные сведения о графических объектах содержатся в главе **Географические и графические объекты**.

## Порядок применения операторов в языке MapBasic

Некоторые операторы имеют более высокий приоритет, чем другие. Это означает, что при вычислении сложных выражений MapBasic следует определенному порядку применения операторов. Чтобы понять, как MapBasic обрабатывает сложные выражения, Вы должны знать относительные приоритеты операторов языка MapBasic.

Рассмотрим следующий оператор присваивания:

`x = 2 + 3 * 4`

В правой части используется две операции – сложение и умножение. Понятно, что результат вычисления зависит от последовательности применения операторов. Если сначала выполнить сложение ( $2 + 3 = 5$ ), а затем умножение ( $5 * 4$ ), то итоговым результатом будет 20. В действительности, умножение имеет более высокий приоритет, чем сложение. Это означает, что MapBasic сначала выполнит умножение ( $3 * 4 = 12$ ), а затем уже сложение ( $2 + 12 = 14$ ).

Чтобы изменить стандартный порядок применения операторов языка MapBasic, надо использовать группировку с помощью скобок. Например, чтобы в приведенном примере сначала выполнялось сложение, его нужно переписать следующим образом:

`x = (2 + 3) * 4`

В данной таблице приведены приоритеты операторов языка MapBasic.

В первую очередь:	скобки
	возведение в степень
	отрицательный знак
	умножение, деление, <b>Mod</b> , целочисленное деление
	сложение, вычитание, конкатенация строк (&)
	географические операторы, операторы сравнения, <b>Like</b>
	<b>Not</b>
	<b>And</b>
В последнюю очередь:	<b>Or</b>

Операторы, перечисленные в одной строке, имеют одинаковый приоритет. Операторы с более высоким приоритетом выполняются ранее других. Операторы с одинаковым приоритетом выполняются в выражении слева направо, исключая операторы возведения в степень, которые выполняются справа налево.



## Циклы и другие управляющие операторы

Управляющие операторы определяют порядок выполнения других операторов. В языке MapBasic имеется три типа управляющих операторов:

- Условные операторы позволяют пропускать выполнение некоторых операторов в программе на языке MapBasic (например, **If... Then, GoTo**).
- Операторы цикла позволяют повторять группу операторов несколько раз (например, **For... Next, Do... While**).
- Другие специальные управляющие операторы (**End Program** и др.).

### Оператор If...Then

Оператор **If... Then** языка MapBasic очень похож на условные операторы If... Then в других языках программирования. В операторе **If... Then** проверяется истинность условия; если оно истинно, MapBasic выполняет операторы, расположенные после ключевого слова **Then**. В следующем примере MapBasic выдает сообщение об ошибке и вызывает процедуру "reset\_counter", если значение счетчика слишком мало:

```
If counter < 0 Then
    Note "Ошибка: Счетчик слишком мал."
    Call reset_counter
End If
```

В операторе **If... Then** может также присутствовать предложение **Else**. Если условие ложно, то MapBasic выполняет операторы, расположенные после ключевого слова **Else**, вместо того, чтобы выполнять операторы, расположенные после ключевого слова **Then**.

Вот пример использования предложения **Else**.

```
If counter < 0 Then
    Note "Ошибка: Счетчик слишком мал."
    Call reset_counter
Else
    Note "Счетчик в порядке."
End If
```

В операторе **If... Then** можно также использовать одно или несколько предложений **Elseif**. Предложение **Elseif** предназначено для проверки дополнительных условий. Если в условном операторе присутствует предложение **Elseif** и проверявшееся условие ложно, то MapBasic проверит условие из предложения **Elseif**, например:

```
If counter < 0 Then
    Note "Ошибка: Счетчик слишком мал."
    Call reset_counter
ElseIf counter > 100 Then
    counter = 100
    Note "Ошибка: Значение счетчика слишком велико; сброшено до значения 100."
Else
```

```
Note "Счетчик в порядке."  
End If
```



Отметим, что ключевое слово **Elseif** пишется слитно. Оператор **If... Then** может содержать несколько предложений **Elseif**, указанные в них условия проверяются последовательно. Впрочем, если нужно проверить несколько условий, Вы можете использовать оператор **Do... Case** (см. описание ниже) вместо оператора **If... Then** со множеством предложений **Elseif**.

## Оператор Do Case


Оператор **Do Case** выполняет набор проверок, чтобы установить, какому из предусмотренных значений равно текущее значение выражения. Для каждого из предусмотренных значений выполняется своя последовательность операторов.

В следующем примере проверяется, к какому кварталу относится текущий месяц. Если текущий месяц относится к первому кварталу (январь-февраль-март), то текстовой переменной присваивается константа "За первый квартал года". Аналогично, если текущий месяц относится ко второму кварталу, то текстовой переменной присваивается константа "За второй квартал года" и т.д.

```
Dim current_month, quarter As SmallInt,  
    report_title As String  
current_month = Month( CurDate() )  
' В данный момент значение переменной current_month равно 1,  
' если текущим месяцем является январь, 2 - февраль и т.д.  
Do Case current_month  
    Case 1, 2, 3  
        ' Если текущий месяц - 1 (январь), 2 (февраль) или 3 (март),  
        ' Сейчас первый квартал.  
        ' Присвоить соответствующее значение.  
        report_title = "Первый квартал года"  
        quarter = 1  
    Case 4, 5, 6  
        report_title = "Второй квартал года"  
        quarter = 2  
    Case 7, 8, 9  
        report_title = "Третий квартал года"  
        quarter = 3  
    Case Else  
        ' Если текущий месяц имеет номер вне диапазона от 1 до 9,  
        ' то сейчас - четвертый квартал.  
        report_title = "Четвертый квартал года"
```

```
    quarter = 4  
End Case
```

---

 Обратите внимание на предложение **Case Else** в конце оператора **Do Case**. **Case Else** – необязательное предложение. Если оператор **Do Case** содержит предложение **Case Else** и текущее значение выражения не равно ни одному из значений, перечисленных в предложениях **Case**, MapBasic выполнит операторы, расположенные после предложения **Case Else**. Предложение **Case Else** должно быть последним предложением оператора **Do Case**.

---

## Оператор GoTo

Оператор **GoTo** используется для перехода в заданное место программы с того места, где встретился оператор **GoTo**. В операторе **GoTo** указывается метка перехода; оператор **GoTo** не будет работать, если в той же процедуре, где находится оператор перехода, нет метки с таким именем. Метка – это имя, сопоставленное некоторой строке программы и расположенное в начале этой строки; после имени метки должно стоять двоеточие (в операторе **GoTo** это двоеточие ставить не надо). Смотрите пример ниже.

```
If counter < 0 Then  
    GoTo get_out  
End If  
...  
get_out:  
End Program
```

Многие профессиональные программисты не одобряют использование операторов перехода типа **GoTo**. Полноценное использование возможностей других управляющих операторов, таких как **If\_Then**, обычно устраняет необходимость использования оператора **GoTo**. Так что Вы можете обходиться и без **GoTo**.

## Оператор For... Next

Оператор **For\_Next** определяет цикл, выполняющийся заданное число раз. Каждое повторение состоит в том, что MapBasic выполняет все операторы, находящиеся между **For** и **Next**. Перед использованием цикла **For... Next**, Вы должны объявить имя числовой переменной, которая будет использоваться в качестве счетчика. Надо указать начальное и конечное значения этого счетчика. После выполнения каждого повтора (после каждой итерации) MapBasic будет увеличивать счетчик на заданную величину (шаг цикла). Стандартный шаг цикла равен единице (1); чтобы задать другой шаг, следует использовать необязательное предложение **Step**.

Ниже приводится пример использования цикла **For... Next** для увеличения всех элементов некоторого массива:

```
Dim monthly_sales(12), grand_total As Float,  
    next_one As SmallInt  
...  
For next_one = 1 To 12  
    grand_total = grand_total + monthly_sales(next_one)  
Next
```

Начиная выполнять оператор **For... Next**, MapBasic присваивает переменной-счетчику начальное значение; в приведенном примере, MapBasic присвоит переменной "next\_one" единицу. Затем MapBasic выполняет операторы, расположенные до ключевого слова **Next**. После каждой итерации цикла MapBasic увеличивает переменную-счетчик. Если счетчик меньше либо равен заданному конечному значению (в примере, если "next\_one" меньше или равно 12), MapBasic выполняет следующую итерацию цикла.

Выполнение оператора **For... Next** прерывается, если обнаружен оператор **Exit For**. Это позволяет при необходимости немедленно остановить цикл.

Подробнее о цикле **For... Next** смотрите в *Справочнике MapBasic*.

## Оператор цикла Do... Loop

Оператор **Do... Loop** выполняет группу операторов, пока заданное в нем условие остается истинным, либо пока условие остается ложным.

Возможны различные виды оператора **Do... Loop**, в зависимости от того, когда Вы хотите проверять условие цикла: до или после выполнения операторов, находящихся в теле цикла. В следующем примере программы условие проверяется в конце цикла:

```
Dim sales_total, new_accounts(10) As Float,  
    next_one As SmallInt  
next_one = 1  
Do  
    sales_total = sales_total + new_accounts(next_one)  
    next_one = next_one + 1  
Loop While next_one <= UBound(new_accounts)
```

Обратите внимание, что такой цикл всегда выполняется хотя бы один раз, поскольку условие проверяется лишь в конце цикла (после выполнения первой итерации).

Следующий пример содержит проверку в начале цикла. Поскольку условие проверяется в начале цикла, операторы в теле цикла могут не выполняться ни разу. Если в момент начала выполнения цикла условие уже ложно, операторы в теле цикла **Do... Loop** не будут выполнены ни разу.

```
Dim sales_total, new_accounts(10) As Float,  
    next_one As SmallInt  
next_one = 1  
Do While next_one <= UBound(new_accounts)  
    sales_total = sales_total + new_accounts(next_one)  
    next_one = next_one + 1  
Loop
```

В обоих приведенных примерах оператора **Do... Loop** используется ключевое слово **While**; поэтому оба цикла выполняются до тех пор, пока указанное после этого ключевого слова условие остается истинным. С другой стороны, в операторе **Do... Loop** можно использовать ключевое слово **Until** вместо ключевого слова **While**. Если оператор **Do... Loop** содержит предложение **Until**, то цикл выполняется, пока контрольное условие остается ложным.

Выполнение оператора **Do... Loop** прерывается, если обнаружен оператор **Exit Do**. Это позволяет при необходимости немедленно остановить цикл.

## Цикл While... Wend

MapBasic поддерживает стандартный синтаксис **While... Wend** языка BASIC. Оператор **While... Wend** очень похож на оператор **Do While... Loop**.

Если вы опытный программист BASIC и привыкли пользоваться операторами **While... Wend**, вы можете продолжать использовать эти операторы так же, как это делали в MapBasic. Но помните, что синтаксис оператора **Do... Loop** более мощный, чем синтаксис **While...Wend**. Завершить цикл **Do... Loop** можно в любой момент при помощи оператора **Exit Do**, однако для завершения цикла **While... Wend** такого оператора нет.

Подробнее о цикле **While... Wend** смотрите в *Справочнике MapBasic*.

## Завершение выполнения программы

Оператор **End Program** прерывает выполнение MapBasic-приложения, удаляет все пользовательские меню, созданные приложением, и выгружает приложение из памяти. **End Program** также закрывает все файлы, открытые в процессе работы приложения (с помощью оператора **Open File**), но не закрывает открытые таблицы.

Оператор **End Program** не обязательно следует выполнять в каждой программе. В некоторых ситуациях Вам, наоборот, следует не выполнять оператор **End Program**. Например, Ваше приложение добавляет свои меню к системе меню MapInfo и Вы хотите, чтобы это приложение оставалось активным в течение всего сеанса работы с MapInfo, поскольку пользователь должен иметь доступ к меню Вашего приложения в течение всего сеанса работы. В таком случае следует проследить, чтобы Ваша программа не выполняла оператор **End Program**, поскольку **End Program** остановит выполнение приложения и удалит все созданные приложением меню. Подробное обсуждение пользовательских меню Вы найдете в главе [Создание интерфейса пользователя](#).

## Завершение выполнения программы и сеанса работы с MapInfo Professional

Оператор **End MapInfo** не только останавливает работу приложения MapBasic (так, как это делает оператор **End Program**), но также прекращает работу MapInfo Professional.

## Процедуры

Процедура (или подпрограмма) представляет собой основную структурную единицу программы на языке MapBasic. Типичная программа на MapBasic состоит из нескольких процедур; каждая такая процедура содержит операторы, вместе выполняющие некоторую конкретную задачу. Разбиение программы на процедуры придает программе модульную структуру, делает ее удобнее для дальнейшей разработки и поддержки.

## Процедура Main

Каждая программа на MapBasic должна содержать по крайней мере одну процедуру, а именно процедуру со стандартным именем "Main". При запуске приложения на MapBasic, автоматически начинается выполняться процедура **Main** из данного приложения.

Следующий пример программы демонстрирует синтаксис объявления и тела процедуры **Main**. В этом примере процедура **Main** выполняет один только оператор **Note**:

```
Declare Sub Main
Sub Main
    Note "Привет от MapBasic'a!"
End Sub
```

Оператор **Declare Sub** указывает MapBasic, что в программе встретится процедура с заданным именем. Каждой процедуре в программе должен соответствовать один и только один оператор **Declare Sub**. Причем оператор **Declare Sub** должен предшествовать телу объявляемой процедуры. Обычно все операторы **Declare Sub** располагаются в самом начале программы.

Вспомним, что ранее (см. [Работа в интегрированной среде разработки программ](#)) мы говорили, что программа на MapBasic может состоять только из одной строки. Например, фрагмент:

```
Note "Привет от MapBasic'a!"
```

является завершенной программой на MapBasic, которую можно откомпилировать и запустить. Даже такая простейшая программа все равно содержит процедуру **Main**, однако, в таких случаях мы говорим, что процедура **Main** задана неявно (в примере же выше она была задана явно).

## Вызов процедуры

При компиляции приложения MapInfo автоматически начинает с вызова процедуры **Main** (независимо от того, явно или неявно она присутствует в программе). Затем процедура **Main** может вызывать другие процедуры с помощью оператора **Call**.

Вот пример программы, состоящей из двух процедур: процедуры **Main** и процедуры с именем **"announce\_date"**.

```
Declare Sub Main
Declare Sub announce_date

Sub Main
    Call announce_date( )
End Sub

Sub announce_date
    Note "Сегодня " + Str$( CurDate( ) )
End Sub
```

## Вызов процедур с параметрами

Как и другие BASIC-подобные языки, MapBasic позволяет создавать процедуры с параметрами. Если процедура имеет параметры, то при ее объявлении их следует перечислять в круглых скобках после имени процедуры в операторе **Sub... End Sub**.

Ниже приводится пример, в котором процедура "check\_date" имеет один параметр (типа "Date"). Эта процедура проверяет, давно ли прошла дата, указанная в качестве параметра (прошло более 180 дней). Если прошло более 180 дней, то процедура устанавливает значение параметра равным текущей дате.

```
Declare Sub Main
Declare Sub check_date(last_date As Date)
Sub Main
    Dim report_date As Date
    report_date = "01.01.94"
    Call check_date( report_date )
    ' В данный момент переменная report_date
    ' может содержать текущую дату (в зависимости
    ' от результата работы процедуры check_date) .

End Sub
Sub check_date(last_date As Date)
    Dim elapsed_days As SmallInt
    elapsed_days = CurDate() - last_date
    If elapsed_days > 180 Then
        last_date = CurDate()
    End If
End Sub
```

## Передача параметров ссылкой

По умолчанию все параметры процедур в MapBasic передаются ссылкой. При этом применяются следующие правила:

- В операторе **Call** все параметры, передаваемые в процедуру, должны быть именами переменных.
- Если в вызываемой процедуре передаваемому ссылкой параметру присваивается новое значение, то изменяется значение соответствующей внешней переменной. Другими словами, процедура может использовать передачу параметров ссылкой для изменения значений параметров переменных.

Так, в последнем приведенном нами примере в операторе **Call** в качестве фактического параметра была указана переменная "report\_date" типа Date:

```
Call check_date( report_date )
```

В процедуре "check\_date" соответствующий (формальный) параметр носил имя "last\_date". Когда в процедуре "check\_date" формальному параметру "last\_date" присваивается текущая дата (last\_date = CurDate( ) ), MapBasic автоматически изменяет и значение фактического параметра "report\_date" в процедуре **Main**.

## Передача параметров значением

Иногда бывает неудобно передавать параметр ссылкой. Все параметры оператора **Call**, передаваемые ссылкой, должны представлять собой имя переменной, а иногда это бывает утомительно (например, если Вы не заводили переменных нужного типа).

Как и в других BASIC-подобных языках, в языке MapBasic Вы можете использовать передачу параметров значением вместо передачи ссылкой. Чтобы указать, что параметр передается значением, следует применить ключевое слово **ByVal** перед именем соответствующего параметра в операторе **Sub... End Sub**.

При передаче параметра значением действуют следующие правила:

- В операторе **Call** (фактические) параметры не обязательно должны быть именами переменных. Вы можете использовать в операторе **Call** как имена переменных, так и константы и вообще любые выражения.
- Если в вызываемой процедуре передаваемому значением формальному параметру присваивается новое значение, то это значение не влияет на соответствующую переменную (фактический параметр) в вызывающей процедуре. Иными словами, передачу значением нельзя использовать для изменения значения фактического параметра.

Следующий пример содержит процедуру ("display\_date\_range"), которая имеет два параметра типа "Date", передаваемых значением.

```
Declare Sub Main
Declare Sub display_date_range(ByVal start_date As Date,
    ByVal end_date As Date )

Sub Main
    Call display_date_range( "1/1", CurDate( ))
End Sub

Sub display_date_range(ByVal start_date As Date,
    ByVal end_date As Date )
    Note "Отчет охватывает период с " + Str$(start_date)
        + " по " + Str$(end_date) + "."
End Sub
```

В данном примере оба параметра в процедуру "display\_date\_range" передаются значением. Так, при вызове "display\_date\_range" из процедуры **Main**:

```
Call display_date_range( "1/1", CurDate( ))
```

ни один из параметров не обязан быть именем переменной. Первый параметр ("1/1") является константным выражением типа "Date", а второй – выражением, состоящим из вызова функции **CurDate( )**.



## Рекурсивный вызов процедур

MapBasic поддерживает рекурсивные вызовы процедур и функций. Другими словами, процедуры MapBasic могут вызывать сами себя.

Применение рекурсии ограничивается количеством доступной памяти. Каждый раз, когда происходит вызов процедуры или функции, MapInfo запоминает текущие данные в стеке; если таких вызовов слишком много, произойдет ошибка “Недостаточно памяти”. Количество требуемой памяти зависит от количества передаваемых параметров и локальных переменных процедуры.

## Процедуры-обработчики системных событий

В языке MapBasic имеются процедуры со специальными именами и особым режимом вызова. Мы уже говорили о процедуре **Main**, которая является специальной, поскольку MapBasic автоматически начинает выполнение приложения с вызова процедуры **Main**.

Кроме **Main**, MapBasic имеет еще несколько специальных процедур: **EndHandler**, **ForegroundTaskSwitchHandler**, **RemoteMapGenHandler**, **RemoteMsgHandler**, **RemoteQueryHandler( )**, **SelChangedHandler**, **ToolHandler**, **WinChangedHandler**, **WinClosedHandler** и **WinFocus ChangedHandler**. Каждая из них имеет имя, зарезервированное в языке MapBasic особым образом. Чтобы понять работу этих процедур, надо принять во внимание подход MapBasic к обработке системных событий.

### Что такое системное событие?

В графическом интерфейсе (Graphical User Interface) пользователь управляет программами нажатиями на клавиатуру и на кнопку мыши. Как только в ответ на действие пользователя было сгенерировано некоторое системное событие, Ваше приложение должно отреагировать соответствующим образом. Кроме перечисленных, событиями являются: выбор команд меню, открытие и закрытие окон и т.д.

### Что такое процедура-обработчик системных событий?

В языке MapBasic обработкой системных событий занимаются специальные процедуры. То есть Вы можете организовать свою программу таким образом, что MapBasic будет автоматически вызывать необходимые процедуры обработки при выявлении тех или иных системных событий. Например, когда пользователь генерирует событие выбора из меню, программе может понадобиться отобразить диалог. Как вариант, когда пользователь закрывает окно, программе может понадобиться заблокировать доступ к пункту меню или скрыть целое меню.

В языке MapBasic обработкой системных событий занимаются специальные процедуры. То есть Вы можете организовать свою программу таким образом, что MapBasic будет автоматически вызывать необходимые процедуры обработки при выявлении тех или иных системных событий.

Описание структуры процедур обработки событий в меню и инструментальных панелях смотрите раздел **Создание интерфейса пользователя**. Чтобы обрабатывать другие типы системных событий, Вы должны создать процедуры со специальными именами. Например, чтобы реагировать на закрытие окон пользователем, Ваше приложение должно содержать процедуру с именем **WinClosedHandler**.

В таблице приводится список всех специальных имен процедур обработки событий в языке MapBasic. Они подробно описаны в *Справочнике MapBasic*.

Имя обработчика	Процедура или функция обработчика
<b>EndHandler</b>	Вызывается в случае, если приложение заканчивает работу или пользователь закрывает MapInfo Professional. <b>EndHandler</b> может использоваться для корректного завершения работы (например, удаления временных файлов).
<b>ForegroundTaskSwitchHandler</b>	Вызывается, когда MapInfo Professional теряет или приобретает фокус.
<b>RemoteMapGenHandler</b>	Вызывается, когда клиент OLE Automation вызывает метод <b>MapGenHandler</b> ; ранее использовалась в приложениях MapInfo ProServer.
<b>RemoteMsgHandler</b>	Вызывается, когда приложение является сервером при обмене данными, и клиент присылает команду.
<b>RemoteQueryHandler( )</b>	Вызывается, когда приложение действует как сервер для обмена внешними процессами, и удаленный клиент посылает запрос.
<b>SelChangedHandler</b>	Вызывается при каждом изменении таблицы Selection. Так как Selection изменяется часто, процедура <b>Sel-ChangedHandler</b> должна быть небольшой, чтобы по возможности не увеличивать время работы приложения.
<b>ToolHandler</b>	Вызывается в том случае, когда пользователь применяет один из инструментов в окне Карты, Списка или Отчета.
<b>WinChangedHandler</b>	Вызывается, когда пользователь изменяет или листает содержимое окна Карты. Поскольку содержимое окна Карты может меняться очень часто, процедура <b>Win-ChangedHandler</b> должна быть небольшой, чтобы по возможности не увеличивать время работы приложения.
<b>WinClosedHandler</b>	Вызывается, когда пользователь закрывает окно Карты, Списка, Графика или Отчета.
<b>WinFocusChangedHandler</b>	Вызывается в тот момент, когда пользователь делает активным одно из окон.

Как правило, для вызова перечисленных процедур не используется оператор **Call**. Если процедура имеет одно из указанных специальных имен, MapBasic вызывает эту процедуру автоматически в момент появления соответствующего системного события. Например, если в Вашей программе имеется процедура **WinClosedHandler**, то MapBasic будет автоматически вызывать **WinClosedHandler** каждый раз, когда пользователь будет закрывать одно из окон.

Процедуры обработки событий могут и не присутствовать в программе. Например, Вам следует включать в приложение только процедуру **WinClosedHandler**, если Вы хотите контролировать только закрытие окон; или только процедуру **SelChangedHandler**, если Вам нужно контролировать только изменение таблицы Selection и т.д.

Следующий пример показывает, как использовать специальную процедуру обработки событий **ToolHandler**. Обратите внимание, что данная программа не содержит ни одного оператора **Call**. После запуска этой программы MapBasic вызывает процедуру **ToolHandler** автоматически каждый раз, когда пользователь применяет один из инструментов в окне Карты, Списка или Отчета.

```
Include "mapbasic.def"

Declare Sub Main
Declare Sub ToolHandler

Sub Main
    Note "Теперь подключена демонстрационная процедура ToolHandler. "
        + "Выберите инструмент MapBasic (+) и щелкните по Карте"
        + "чтобы увидеть распечатку координат карты."
End Sub
Sub ToolHandler
    If WindowInfo( FrontWindow(), WIN_INFO_TYPE ) = WIN_MAPPER Then
        Print "X: " + Str$( CommandInfo(CMD_INFO_X) )
        Print "Y: " + Str$( CommandInfo(CMD_INFO_Y) )
        Print " "
    End If
End Sub
```

В теле процедуры обработки событий вызывается функция **CommandInfo( )**, чтобы выяснить, с каким именно событием работает процедура обработки. В нашем примере в процедуре **ToolHandler** функция **CommandInfo( )** вызывается для того, чтобы определить координаты точки, в которую указал пользователь.

Следующий фрагмент процедуры **SelChangedHandler** взят из примера программы из комплекта поставки – TEXTBOX.MB. Эта процедура автоматически делает недоступным (рисует серым цветом) один из пунктов меню, когда пользователь отменяет выбор всех строк, и снова делает его доступным, когда пользователь выбирает какую-либо строку.

Смотрите текст программы TEXTBOX.MB.


```
Sub SelChangedHandler
    If SelectionInfo(SEL_INFO_NROWS) < 1 Then
        Alter Menu Item create_sub Disable
    Else
        Alter Menu Item create_sub Enable
    End If
End Sub
```

```
End If  
End Sub
```

## Когда вызываются обработчики событий?

По умолчанию, приложение MapBasic заканчивается после выполнения всех операторов в процедуре **Main**. Однако, если приложение содержит хотя бы один из перечисленных нами обработчиков событий (например, процедуру **ToolHandler**), приложение остается в памяти и после завершения выполнения процедуры **Main**. Такое состояние приложения называют неактивным. Неактивное приложение остается в памяти в состоянии ожидания до тех пор, пока не возникнет соответствующее событие (например, используется один из инструментов). В случае наступления подобного события MapBasic автоматически активизирует ожидающее приложение, а точнее – соответствующую процедуру обработки событий.

---

 Если в какой-нибудь из процедур будет выполнен оператор **End Program**, все приложение будет удалено из памяти, независимо от того, содержит ли приложение процедуры обработки событий. Чтобы оставить приложение в памяти, Вы не должны использовать оператор **End Program** в Вашей программе.

---

Аналогично работают пользовательские (новые) меню MapBasic. Если приложение на языке MapBasic добавляет свои меню к стандартной системе меню MapInfo Professional, то оно остается неактивным в памяти и ждет, пока пользователь выберет одну из команд в пользовательских меню. Подробнее о пользовательских меню под MapInfo Professional смотрите главу [Создание интерфейса пользователя](#).

## Рекомендации по использованию процедур-обработчиков

### Делайте процедуры-обработчики короткими!

Имейте в виду, что некоторые процедуры-обработчики событий вызываются часто. Например, если Вы создаете процедуру **SelChangedHandler**, MapInfo Professional вызывает ее каждый раз при изменении таблицы Selection. В типичном MapInfo сеансе таблица Selection часто изменяется; следовательно, Вы должны делать процедуры обработчики событий типа **SelChangedHandler** настолько быстрыми, насколько это возможно.

### Выбор без вызова SelChangedHandler

Если используется оператор **Select**, но требуется, чтобы оператор не переключал процедуру **SelChangedHandler**, добавьте ключевой параметр **NoSelect**. Например:

```
Select * From World Into EarthQuery NoSelect
```

### Предотвращение бесконечных циклов

Выполнение действий внутри процедуры-обработчика системных событий может иногда вызывать бесконечный цикл. Например, если Вы объявляете процедуру **SelChangedHandler**, MapInfo вызывает ее процедуру всякий раз, когда изменяется выборка. Если Вы вызываете

оператор **Select** внутри процедуры **SelChangedHandler**, оператор **Select** заставит MapInfo Professional вызывать процедуру рекурсивно. Конечным результатом может стать бесконечный цикл.

Оператор **Set Handler** может предотвращать бесконечные циклы. В начале Вашей процедуры-обработчика, поместите оператор **Set Handler Off**, чтобы предотвратить рекурсивный вызов обработчика. В конце процедуры поместите оператор **Set Handler On**, чтобы восстановить обработчик.

```
Sub SelChangedHandler
    Set Handler SelChangedHandler Off

    ' В этом месте можно применить в цикле оператор Select,
    ' не обрабатывая при этом изменения выбора.

    Set Handler SelChangedHandler On
End Sub
```

## Новые функции

В языке MapBasic используются различные функции: некоторые стандартные функции BASIC (такие как **Asc( )**, **Format\$( )**, **Val( )** и пр.), а также особые функции MapInfo и MapBasic (например, **Distance( )** или **ObjectGeography( )**). MapBasic также позволяет Вам создавать свои функции. Создав свою функцию, Вы можете вызывать ее так же, как и стандартные функции языка MapBasic.

Тело функции заключается между парой ключевых слов **Function... End Function**, что аналогично конструкции **Sub... End Sub** для процедур. Общий синтаксис конструкций **Function... End Function** таков:

```
Function имя_функции( параметры ) As тип_данных
    операторы
End Function
```

Отметим, что задается тип самой функции. Он определяет тип значения (например, Integer, Date, String), возвращаемого функцией.

Внутри конструкции **Function... End Function** имя функции доступно как параметр, переданный ссылкой. В операторе в теле конструкции **Function... End Function** можно присвоить значение; это значение позднее MapBasic вернет вызывающей процедуре как значение функции.

Ниже приводится пример функции с именем "money\_format( )". Функция "money\_format( )" имеет один числовой параметр, задающий сумму, и возвращает строку (полученную с помощью обращения к функции **Format\$( )**), в которой тысячи, миллионы и т.п. отделены запятыми.

```
Declare Sub Main
Declare Function money_format(ByVal num As Float) As String
Sub Main
    Dim dollar_amount As String
    dollar_amount = money_format( 1234567.89 )
    ' переменная dollar_amount сейчас содержит строку: "$1,234,567.89"
End Sub
```

```
Function money_format(ByVal num As Float) As String
    money_format = Format$(num, "$, #.##; ($, #.##)")
End Function
```

## Область определения функций

В программе можно ввести функцию, имя которой будет совпадать с именем одной из стандартных функций языка MapBasic. При вызове функции с таким именем будет выполнена Ваша функция вместо стандартной.

## Директивы компилятора

В языке MapBasic имеются две специальные директивы, которые упрощают процесс разработки больших приложений:

- Директива **Define** позволяет объявить имя-синоним константы, причем значение подставляется вместо имени-синонима во время компиляции.
- Директива **Include** позволяет компилировать несколько файлов, содержащих текст программы, в одну программу.

### Директива Define

С помощью директивы **Define** Вы можете сопоставить некоторой константе имя (идентификатор).

Директива **Define** используется в тех случаях, когда Вам часто приходится набирать в своей программе одно и то же выражение.

Например, если в программе активно используются объекты и цвета, Вам может понадобиться часто набирать значение 16711680, числовой код, соответствующий красному цвету. Довольно утомительно много раз набирать такое длинное число (и вспоминать его). Чтобы облегчить себе работу, Вы можете ввести следующую директиву **Define**:

```
Define MY_COLOR 16711680
```

Директива **Define** создает просто запоминающийся синоним (MY\_COLOR), соответствующий числу 16711680. После того, как была задана подобная директива **Define**, Вы можете набирать MY\_COLOR везде, где Вам нужно употребить значение 16711680. При компиляции программы MapBasic вместо каждого вхождения MY\_COLOR подставит значение 16711680.

Кроме того, имеются и более "долгосрочные" выгоды от использования синонимов. Предположим, что Вы разрабатываете большое приложение, в котором во многих местах используется имя MY\_COLOR. Предположим теперь, что Вам потребовалось заменить красный цвет на зеленый (65280). Вы сможете переключиться с красного на зеленый цвет, просто изменив директиву **Define** на:

```
Define MY_COLOR 65280
```

Стандартный файл заголовков MapBasic, MAPBASIC.DEF, содержит значительное число директив **Define**, в том числе директивы Define для некоторых наиболее часто используемых цветов (BLACK, WHITE, RED, GREEN, BLUE, CYAN, MAGENTA и YELLOW). С помощью директивы **Include** Вы можете включить файл MAPBASIC.DEF в свою программу.

## Директива Include

Директива **Include** позволяет объединять два или более отдельно набранных программных файлов в одно приложение на MapBasic. Директива **Include** имеет следующий синтаксис:

```
Include "имя_файла"
```

где *имя\_файла* – это имя текстового файла, содержащего операторы языка MapBasic. При компиляции программы, содержащей директиву **Include**, компилятор считает, что включаемый текст является частью файла, содержащего эту директиву.

Многие приложения на языке MapBasic используют директиву **Include** для того, чтобы включить в программу стандартный файл заголовков языка MapBasic – MAPBASIC.DEF:

```
Include "mapbasic.def"
```

MAPBASIC.DEF содержит директивы **Define**, определяющие многие стандартные имена языка MapBasic (TRUE, FALSE, RED, GREEN, BLUE, TAB\_INFO\_NAME и т.д.).

Имя файла в данной директиве может также включать имя диска и DOS-маршрут к каталогу. Если путь не указан, компилятор MapBasic ищет файл в текущем каталоге; если файл так и не обнаружен, компилятор ищет его в том каталоге, в котором установлен MapBasic.

По мере разработки приложений на MapBasic, у Вас могут накопиться часто используемые фрагменты исходных текстов. Возможно, Вы создадите из них библиотеку своих функций и будете включать эту библиотеку в каждую новую программу на MapBasic. Тексты таких функций можно вынести в отдельный текстовый файл, например, с именем FUNCTS.MB. Чтобы включить этот файл в любой текст программы, надо просто вставить директиву:

```
Include "functs.mb"
```

Применение директивы **Include** также позволяет обойти ограничения текстового редактора MapBasic. Как уже говорилось в главе **Работа в интегрированной среде разработки программ**, каждое текстовое окно в редакторе MapBasic может содержать не более 50K текста. В случае перехода через этот рубеж, Вы можете разбить текст своей программы на несколько файлов, а затем скомбинировать их вместе с помощью директивы **Include** (подробнее см. главу **Работа в интегрированной среде разработки программ**).

## Организация программ

Приложение на языке MapBasic может содержать любые или все типы операторов, описанных в этой главе. Однако, отдельные части MapBasic-программы должны быть оформлены определенным способом. Например, операторы **Global** должны быть вынесены за пределы процедур (вне конструкций **Sub... End Sub**).

Следующий пример показывает типичное расположение различных частей программы.

Сначала описываются операторы глобального уровня...

```
Include "mapbasic.def"  
другие операторы Include  
операторы Type... End Type  
операторы Declare Sub  
операторы Declare Function
```

```
операторы Define
операторы Global
... затем, процедура Main...

Sub Main
    операторы Dim
    ...
End Sub

... другие процедуры...

Sub ...
    операторы Dim
    ...
End Sub

... пользовательские функции...

Function ...
    операторы Dim
    ...
End Function
```



# Поиск ошибок и отладка программ

Даже если программа была успешно скомпилирована, в процессе ее выполнения могут появляться ошибки (так называемые ошибки при выполнении). Например, если в Вашей программе создается большая база данных, при выполнении программы может быть зафиксирована ошибка, если на жестком диске нет больше свободного места.

В этой главе рассказывается, как следует работать с ошибками при выполнении программы. Этот процесс можно разделить на два этапа: во-первых, Вы отлаживаете программу, чтобы выявить те места, где при выполнении появляются ошибки; затем Вы вносите изменения в программу, чтобы реагировать на исключительные (ошибочные) ситуации.

## В этой главе:

- ♦ Ошибки при выполнении .....82
- ♦ Отладка программ на языке MapBasic .....82
- ♦ Поиск ошибок .....84

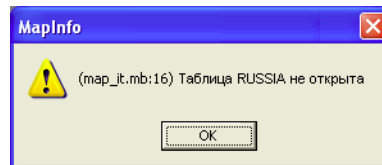
## Ошибки при выполнении

Существуют два основных типа ошибок в программах: ошибки при компиляции и ошибки при выполнении. Ошибки, фиксируемые во время компиляции, мы уже обсуждали (см.: "**Работа в интегрированной среде разработки программ**"). Обычно это синтаксические ошибки или опечатки, выявляемые при компиляции.

Ошибки при выполнении обнаруживаются только при запуске успешно скомпилированной программы. Причины возникновения таких ошибок различны. Как правило, они связаны с конкретными условиями (контекстом), сложившимися во время выполнения. Например, следующий оператор будет откомпилирован:

```
Map From RUSSIA
```

Однако, если при его выполнении не будет открыта таблица с именем "RUSSIA", то будет зафиксирована ошибка. При обнаружении ошибки выполнения MapInfo Professional прерывает выполнение MapBasic-программы и выдает диалог с сообщением об ошибке.



Сообщение содержит название программного файла и номер строки, где обнаружена ошибка. Пусть в приведенном выше примере название файла – MAP\_IT, а номер строки с ошибкой – 16. Номер строки является достаточно удобным идентификатором части программы, вызвавшей ошибку. Зная номер строки, Вы можете вернуться в редактор MapBasic и с помощью команды **Перейти к строке** (из меню **Поиск**) перейти к оператору, в котором возникли проблемы при выполнении.

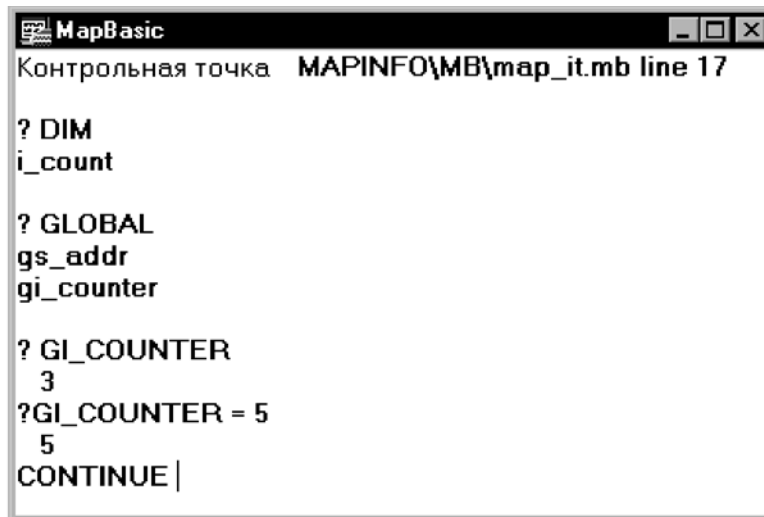
## Отладка программ на языке MapBasic

Некоторые ошибки при выполнении исправить довольно просто. Например, некоторые ошибки связаны просто с опечатками при создании программы (скажем, программист вместо названия таблицы RUSSIA набрал ROSSIA). Однако есть и такие ошибки, объяснить появление которых нелегко. При обнаружении ошибок в программе Вам помогут средства отладки языка MapBasic (операторы **Stop** и **Continue**), которые используются в окне MapBasic в MapInfo.

### Краткое описание процесса отладки

Чтобы выявить ошибки в неправильно работающем фрагменте Вашей программы, можно использовать следующую процедуру:

1. В редакторе MapBasic вставьте оператор **Stop** перед тем фрагментом программы, который содержит ошибку.
2. Перекомпилируйте и запустите Вашу программу.  
Когда выполнение программы дойдет до оператора **Stop**, MapBasic временно приостановит выполнение и выдаст специальное сообщение в окне MapBasic (например, "Контрольная точка в файле TEXTBOX.MB, строка 23").
3. В окне MapBasic (в среде MapInfo):  
Введите ? Dim – чтобы получить список всех локальных переменных.  
Введите ? Global – чтобы получить список всех глобальных переменных.  
Введите ? *имя\_переменной* – чтобы увидеть значение переменной.  
Введите ? *имя\_переменной* = *значение*, чтобы присвоить переменной новое значение.
4. Закончив работу по анализу значений переменных, наберите **Continue** в окне MapBasic, чтобы продолжить выполнение программы. Или же выполните команду **Продолжить программу MapBasic** из меню **Файл** программы MapInfo Professional. Заметьте, что при остановке выполнения приложения меню **Файл** содержит команду **Продолжить программу MapBasic** вместо **Запустить программу MapBasic**.



## Ограничения на оператор Stop

MapBasic не позволяет использовать оператор **Stop** для остановки программы в следующих случаях:

- Нельзя использовать оператор **Stop** внутри конструкции операторов **Function...End Function**.
- Нельзя использовать оператор **Stop** внутри оператора работы с диалогами (**Dialog**), поскольку этот оператор управляет работой с активным диалогом на экране.
- Нельзя использовать оператор **Stop** в операторе **ProgressBar**.
- Вы не можете отлаживать приложение, если уже выполняется другое приложение.

- С помощью оператора **Run Application** Вы можете запускать из выполняющейся программы другую программу MapBasic. Однако в таком случае Вы не сможете использовать оператор **Stop** для отладки этой программы.

Кроме того, возможно одновременное выполнение нескольких программ и без использования оператора **Run Application**. Например, если Вы запустите программу TEXTBOX (см. приложение 2), то оно создаст свое меню и перейдет в состояние ожидания, после чего Вы можете запустить другую MapBasic-программу. Однако в подобном случае Вы не должны использовать для отладки оператор **Stop**.

## Другие средства отладки

Операторы языка MapBasic **Note** и **Print** также могут использоваться при отладке программы. Например, если Вы хотите держать под контролем изменение значения некоторой переменной, просто воспользуйтесь оператором **Print**:

```
Print "Текущее значение счетчика: " + counter
```

чтобы выдавать текущее значение в окне Сообщений. С помощью программы AppInfo.mbx, из состава образцов программ, можно проверить глобальные переменные любой исполняемой MapBasic-программы.

## Поиск ошибок

В аккуратно написанной программе должна быть по возможности предусмотрена реакция на возможные ошибки при выполнении. Специальные процедуры для Прерывания исполнения программ и исследования ошибок иногда называют ловушками. В языке MapBasic реагировать на ошибки времени выполнения можно с помощью оператора **OnError**.

Для программистов, имеющих опыт работы с другими языками типа BASIC, заметим: в MapBasic **OnError** пишется слитно.

В любом месте программы можно включить режим реакции на ошибки. По умолчанию, в начале выполнения любой процедуры или функции этот режим отключен. Использование оператора **OnError** включает режим реакции на ошибки.

Обычно в операторе **OnError** указывается метка, которая должна присутствовать в той же процедуре или функции, что и этот оператор. Группу операторов, расположенных после такой метки, называют процедурой-обработчиком ошибки или просто обработчиком ошибки. При обнаружении ошибки во время выполнения программы MapBasic переходит на указанную метку и запускает обработчик ошибки вместо того, чтобы просто прервать выполнение программы.

При обработке ошибки можно обращаться к функции **Err( )**, чтобы узнать код выявленной ошибки (типа Integer). А функция **Error\$( )** возвращает строку с сообщением об ошибке. Полное описание возможных кодов ошибок языка MapBasic можно найти в текстовом файле ERRORS.DOC из комплекта поставки. Процедура-обработчик ошибки должна оканчиваться оператором **Resume**. Оператор **Resume** указывает, с какой строки MapBasic должен продолжить выполнение программы.

Подробнее обработка ошибок, возникающих при выполнении программы, описана в главах **OnError**, **Resume**, **Err( )** и **Error\$( )** в *Справочнике MapBasic*.

---

**i** В каждый конкретный момент времени возможна обработка только одной ошибки. Если используются ловушки и происходит ошибка, MapBasic переходит к процедуре обработки ошибки. Если новая ошибка случится в процедуре обработки ошибок (до оператора **Resume**), то работа приложения будет прекращена.

---

## Пример обработки ошибки

Ниже приведен фрагмент программы, в которой открывается таблица с именем ORDERS и ее содержимое отображается в окнах Карты и Списка. Обработчик исключительных ситуаций "bad\_open" предназначен для реакции на ошибки, которые могут возникнуть при выполнении оператора **Open Table**. Второй обработчик – "not\_mappable" – реагирует на ошибки при выполнении оператора **Map**.

```
Sub orders_setup
  ' В начале выполнения процедуры режим обработки ошибок отключен
  OnError Goto bad_open
  ' Здесь включается режим обработки еще одного типа.
  ' bad_open – метка начала процедуры обработки ошибки.
  Open Table "orders.tab"
  OnError Goto not_mappable

  ' Здесь включается режим обработки еще одного типа.
  ' not_mappable – метка начала обработки ошибки.
  Map From orders
  OnError Goto 0
  Browse * From orders

last_exit:
  Exit Sub
  ' позволяет избежать
  ' случайного запуска обработчиков ошибок.

bad_open:
  ' Следующие операторы выполняются, если найдена ошибка.
  ' в операторе Open.
  Note "Таблица Orders не была открыта... Конец программы."
  Resume last_exit

not_mappable:
  ' Следующие операторы выполняются, если найдена ошибка в операторе Map
  Note "Данные нельзя отобразить на карте; они доступны только в окне
  Списка."
  Resume Next
End Sub
```

Оператор `OnError Goto bad_open` включает режим обработки ошибок, которые могут случиться при выполнении оператора **Open Table**. В случае обнаружения ошибки MapBasic переходит к выполнению операторов, расположенных после метки "bad\_open". Обработчик ошибки состоит из сообщения об ошибке, а также оператора **Resume**, осуществляющего переход на метку "last\_exit".

Если же оператор **Open Table** был выполнен без ошибок, то выполняется следующий по порядку оператор, а именно, `OnError Goto not_mappable`. В нем включается режим обработки ошибки, которая может быть зафиксирована при выполнении оператора **Map**. В последнем случае MapBasic переходит на метку "not\_mappable". Обработчик "not\_mappable" выдает сообщение об ошибке (почему нельзя открыть окно Карты) и выполняет оператор **Resume Next**. **Resume Next** указывает, что MapBasic должен выполнить следующий оператор за тем, в котором была обнаружена ошибка.

Оператор `OnError Goto 0` отключает режим обработки исключительных ситуаций. Так, если будет выполнен с ошибкой оператор **Browse**, обработки ошибки не произойдет, выполнение программы просто будет прекращено.

# Создание интерфейса пользователя

Интерфейс пользователя является важной частью любой программы. MapBasic предоставляет в Ваше распоряжение все средства, необходимые для настройки интерфейса MapInfo Professional.

## В этой главе:

- ♦ Принципы построения интерфейса программ MapBasic . . . .88
- ♦ Программная обработка событий . . . . .88
- ♦ Меню . . . . .90
- ♦ Стандартные диалоговые окна . . . . .100
- ♦ Новые диалоговые окна . . . . .102
- ♦ Окна . . . . .111
- ♦ Панели инструментов . . . . .118
- ♦ Курсоры . . . . .127
- ♦ Запуск программы в среде MapInfo . . . . .127
- ♦ Рекомендации по повышению производительности . . . . .129

## Принципы построения интерфейса программ MapBasic

Создавая программу на языке MapBasic, Вы можете создать для нее интерфейс пользователя MapInfo Professional. Программа на языке MapBasic может управлять следующими элементами пользовательского интерфейса:

- **Меню:** Программы на языке MapBasic могут добавлять новые меню или новые команды в систему меню MapInfo и удалять существующие.
- **Диалоги:** Пользователь может создавать свои диалоги и вызывать их из своих программ.
- **Окна:** Программы на языке MapBasic могут показывать стандартные окна MapInfo (Карты, Списки и т.д.) с заранее определенным содержанием. MapBasic может также показывать сообщения в специальном окне "Сообщения" и в строке сообщений в нижней части окна MapInfo.
- **Панели инструментов:** Пользователь может создавать свои инструментальные панели со своими кнопками, изменять стандартные панели и их состав. Одна из стандартных панелей MapInfo Professional, **Программы**, предназначена для размещения кнопок, создаваемых программами MapBasic. Например, программа "Масштаб" ("Scalebar") помещает в эту панель свою кнопку.

Программа из пакета поставки MapBasic под названием "Обзор" ("Overview"), является хорошим пособием для изучения принципов создания интерфейса пользователя в MapBasic. После запуска программы "Overview" MapBasic добавляет новые элементы меню **Программы (Tools)**. Если пользователь выполнит команду **Настройка обзора**, MapBasic показывает диалоговое окно. Пользователь выбирает таблицу в диалоге, и MapBasic открывает ее в новом окне Карты.

## Программная обработка событий

Язык MapBasic работает под управлением событий. Для понимания того, как в MapBasic организуется интерфейс с пользователем, Вам нужно сначала познакомиться с концепцией программы, обрабатывающей события (event-driven).

### Что такое событие?

В графическом интерфейсе (Graphical User Interface) пользователь управляет программами нажатиями на клавиатуру и на кнопку мыши. С технической точки зрения нажатия на кнопку мыши, ее перемещения, нажатия на клавиши клавиатуры являются системными событиями. Кроме перечисленных, событиями являются: выбор команд меню, открытие и закрытие окон и т.д.

### Что происходит, когда пользователь выбирает команду меню?

Выбрав одну из команд меню, пользователь порождает событие, и программа реагирует на него: показывает диалог, выполняет операцию – другими словами, обрабатывает событие. Так и в программе, написанной на MapBasic и создающей новое меню, выбор команды порождает событие, которое MapBasic должен обработать, например, открыть или закрыть таблицу или окно. Мы говорим, что система обрабатывает события, если пользователь выполнил какие-либо действия.



Если в программе на MapBasic создано дополнительное меню, а пользователь выбирает один из пунктов этого меню, то программа на MapBasic обрабатывает это событие. Обычно, программа на MapBasic обрабатывает события при помощи вызова процедур. Тогда мы будем говорить, что процедура является обработчиком события или просто "обработчиком".

Таким образом, создание нового меню состоит из двух существенных этапов:

5. Настройка системы меню MapInfo Professional с помощью операторов **Create Menu** или **Alter Menu**.
6. Создание процедуры-обработчика для каждого нового элемента меню. Обработчик оформляется как подпрограмма, размещаемая в любом месте программы. Каждый обработчик выполняет свою задачу, соответствующую элементу меню. С другой стороны, Вы можете вместо процедуры определить в качестве обработчика стандартную команду MapInfo. Так можно включать в новое меню стандартные команды MapInfo, например, **Создать тематическую Карту** из меню **Карта**).

Как уже упоминалось в главе **Работа в интегрированной среде разработки программ**, оператор **Call** вызывает подпрограмму. Однако, если процедура является обработчиком, то оператор **Call** не нужен. Вместо оператора **Call** процедуру вызывает предложение **Calling**, входящее в состав оператора **Create Menu**.

Например, программа TEXTBOX содержит следующий оператор **Create Menu**:

```
Create Menu "Рамка" As
    "&Создать рамки..." Calling create_sub,
    "Выход" Calling Bye,
    "О программе ""Рамка""..." Calling About
```

Этот оператор создает новое меню с несколькими новыми командами, и каждой из них в предложении **Calling** сопоставлен обработчик (например, "Calling create\_sub"). Каждое предложение **Calling** задает имя процедуры, которая включена в текст программы TEXT-BOX.MB. То есть, имена "create\_sub", "Bye" и "About" являются именами Sub-процедур.

Как только пользователь выполнит команду **Создать рамки** из меню **Рамка**, MapBasic автоматически вызовет процедуру "create\_sub". Другими словами, процедура "create\_sub" является обработчиком для этой команды.

## Как программа обрабатывает нажатия на кнопки инструментальной панели?

Каждая кнопка на новой инструментальной панели должна иметь соответствующую процедуру-обработчик. Как и в операторе **Create Menu**, в аналогичном операторе **Create ButtonPad**, создающем инструментальную панель, есть предложение **Calling**, задающее обработчик. И точно так же, как и с командами меню, нажатие на кнопку инструментальной панели вызывает упомянутый в операторе **Create ButtonPad** обработчик.

MapBasic поддерживает несколько типов кнопок. При нажатии на кнопку типа PushButton MapBasic вызывает процедуру-обработчик сразу. При нажатии на кнопку типа ToolButton MapBasic вызывает процедуру только тогда, когда пользователь указал мышкой на окно. См. также раздел **Панели инструментов на стр. 118**.

## Как MapBasic обрабатывает события, происходящие в окнах диалога?

Диалоги, созданные пользователем MapBasic, могут вызывать процедуры-обработчики. Так, если в диалоге есть флажок, то MapBasic может обращаться к обработчику каждый раз, когда пользователь установит или сбросит флажок. При работе с диалогами связь с процедурами-обработчиками задается программистом, если она нужна, или может не задаваться вовсе. Более подробно о том, как создавать диалоги и работать с ними в главе **Новые диалоговые окна на стр. 102**.

## Меню

Меню являются существенным элементом любого графического интерфейса. MapBasic позволяет Вам управлять всеми элементами системы меню MapInfo. Вы можете перестроить систему меню MapInfo сравнительно легко, используя несколько команд.

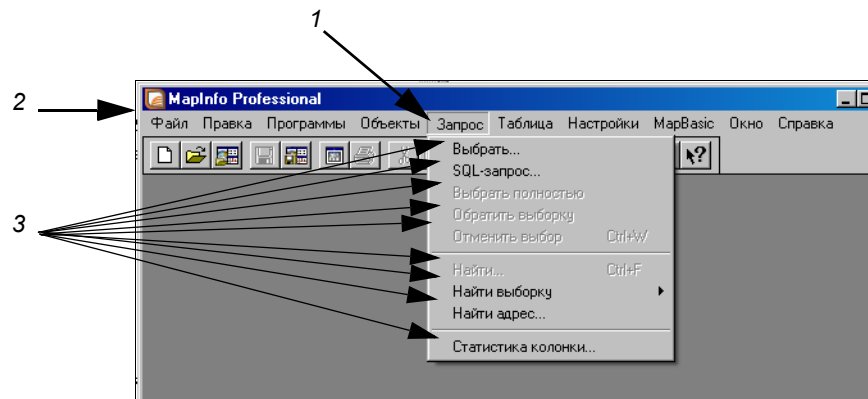
## Основные принципы построения и работы с меню

Система меню MapInfo состоит из следующих элементов:

Строка заголовков меню или просто строка меню – это горизонтальная полоска над рабочей областью MapInfo. Стандартная строка меню MapInfo содержит заголовки **Файл**, **Правка**, **Объект**, **Запрос** и т.д.

Меню – это вертикальный список команд, открывающийся при выборе одного из заголовков. Например, большинство программ содержат меню **Файл** и меню **Правка**.

Элемент меню (или пункт меню) – это отдельная команда из списка команд меню. Например, меню **Файл** обычно содержит элементы **Открыть**, **Заккрыть**, **Сохранить** и **Печатать**. Элементы меню иногда также называют командами (например, команда **Файл > Сохранить**).



1 Меню 2 Строка меню 3 Пункты меню

Меню, строка меню и элементы – взаимосвязанные понятия. Каждое меню состоит из набора элементов меню. Например, меню **Файл** содержит такие элементы меню как **Открыть**, **Заккрыть**, **Сохранить** и т.п. Строка меню содержит заголовки меню.

Как только пользователь выберет элемент меню, происходит системное событие. События могут носить различный характер: одни вызывают на экран диалоговое окно, другие сразу выполняют операцию и т.д.

Процедура, срабатывающая при выборе элемента меню, называется обработчиком. Обработчиком команды (элемента) меню может быть стандартный командный код MapInfo или подпрограмма (процедура) на языке MapBasic, которую может создать пользователь. Другими словами, как только будет выбран элемент меню, MapInfo обрабатывает это событие, либо выполняя одну из стандартных процедур, либо вызывая процедуру из MapBasic-программы.

## Добавление новых элементов в меню

Чтобы добавить один или несколько новых элементов в меню, используется оператор **Alter Menu**.

Например, следующий оператор добавляет две команды в меню **Запрос** (одна называется "Годовой отчет", а другая – "Квартальный отчет"):

```
Alter Menu "Запрос" Add
    "Годовой отчет" Calling report_sub,
    "Квартальный отчет" Calling report_sub_q
```

Для каждого из новых элементов меню в операторе **Alter Menu** задано предложение **Calling**. Это предложение является указанием на то, что нужно делать при выборе соответствующей команды. Если будет выбрана команда **Годовой отчет**, то MapInfo вызовет подпрограмму "report\_sub".

Если будет выбрана команда **Квартальный отчет**, то MapInfo вызовет подпрограмму "report\_sub\_q". Эти подпрограммы ("report\_sub" и "report\_sub\_q") должны содержаться в тексте той же MapBasic-программы.

Вы можете также создать новые команды меню, выполняющие те же действия, что и стандартные команды MapInfo, а не вызывать каждый раз процедуры на языке MapBasic. В файле определений MENU.DEF содержится список кодов, соответствующих командам меню (например, M\_FILE\_NEW соответствует команде **Файл > Новый**, M\_EDIT\_UNDO соответствует команде **Правка > Отменить** и т.д.). Если в предложении **Calling**, описывающем новую команду, будет указан код из файла MENU.DEF и пользователь выберет эту команду, то MapInfo выполнит команду, соответствующую заданному коду.

Например, с помощью следующего оператора можно создать пункт меню **Раскрасить Карту**. Если будет выбран пункт меню **Раскрасить Карту**, MapInfo Professional выполнит команду M\_MAP\_THEMATIC. Если пользователь выберет эту команду, MapInfo обратится по коду M\_MAP\_THEMATIC к своей внутренней процедуре, которая представлена стандартной командой **Карта > Создать тематическую Карту**.

```
Alter Menu "Запрос" Add
    "Раскрасить Карту" Calling M_MAP_THEMATIC
```

## Удаление элементов из меню

Программа может удалять элементы меню. Следующий оператор удаляет из меню MapInfo **Таблица > Изменить** команду Удалить. При этом удаляемая команда задается кодом M\_TABLE\_DELETE, определенным в файле MENU.DEF.

```
Alter Menu "Изменить" Remove M_TABLE_DELETE
```

Если нужно удалить несколько элементов меню, Вы можете выбрать один из двух способов: либо с помощью оператора **Alter Menu ... Remove** можно указать, какие именно элементы нужно удалить; либо оператором **Create Menu** полностью переопределить меню, задав новый набор команд.

Например, следующий оператор создает упрощенный вариант меню **Карта** из трех команд: **Управление слоями**, **Показать как было** и **Настройка**:

```
Create Menu "Карта" As
    "Управление слоями" Calling M_MAP_LAYER_CONTROL,
    "Показать как было" Calling M_MAP_PREVIOUS,
    "Настройка" Calling M_MAP_OPTIONS
```

## Создание нового меню

Для создания полностью нового меню используется оператор **Create Menu**. Например, программа TEXTBOX содержит следующий оператор **Create Menu**:

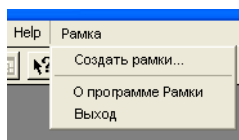
```
Create Menu "Рамка" As
    "&Создать рамки..." Calling create_sub,
    " (-",
    "&О программе ""Рамка""..." Calling About,
    "В&ыход" Calling Bye
```

Оператор **Create Menu** создает новое меню Рамка. Однако, создание нового меню отнюдь не влечет за собой его немедленный показ на экране. Для этого нужно предпринять дополнительные действия.

Чтобы сделать меню Рамка видимым, добавьте его в систему меню оператором **Alter Menu Bar**:

```
Alter Menu Bar Add "Рамка"
```

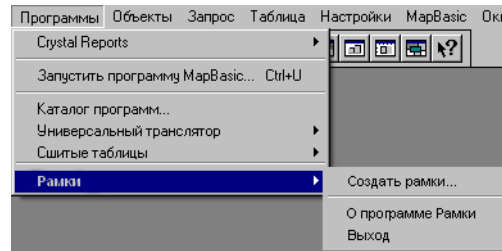
Оператор **Alter Menu Bar Add** добавляет новое меню с правой стороны строки заголовков меню. Новое меню будет выглядеть следующим образом:



На практике добавление нового меню может породить неудобства. Длина строки заголовков ограничена и, действуя таким образом, Вы ее быстро переполните. Поэтому в программе TEXTBOX используется другой прием: меню не добавляется в строку заголовков, а встраивается как подменю в другое меню **Программы** с помощью оператора **Alter Menu**.

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

В результате выполнения этого оператора в меню **Программы** появится некая иерархия команд с общим названием **Рамка**. В результате получается меню **Программы** следующего вида:



Программы из комплекта поставки, такие как "Scalebar" ("Масштаб") и "Overview" ("Обзор"), встраиваются в систему меню MapInfo тем же способом: добавляются в меню **Программы**. Так, если будут запущены программы "Рамка", "Масштаб" и "Обзор", то в меню **Программы** добавятся три новых элемента.

Если бы каждая новая программа добавлялась в строку заголовков меню, то последняя бы быстро переполнилась. Размещая новые элементы меню в меню **Программы**, Вы экономите много места. Однако, нужно заметить, что иерархическое построение меню не кажется удобным многим пользователям.

Каким из способов Вы воспользуетесь, зависит от самой Вашей программы. Если нужно, Вы можете, конечно, добавить много новых меню.

Однако, куда бы Вы не поместили новые меню, помните о том, что в MapInfo допускается не более 96 определений меню. Другими словами, в одно и то же время MapInfo может поддерживать 96 меню, включая свои стандартные меню. Это ограничение никак не связано с количеством показанных меню на экране.

## Изменение элемента меню

Язык MapBasic позволяет Вам совершать следующие действия с отдельными элементами меню:

- Вы можете показать элемент серым цветом, то есть сделать его недоступным.
- Вы можете сделать доступным ранее недоступный (серый) элемент.
- Вы можете пометить галочкой команду в меню; задать эту возможность нужно заранее на этапе определения. Для этого нужно перед первым символом в имени меню поставить восклицательный знак. Более подробные сведения см. в описании оператора **Create Menu** в *Справочнике MapBasic*.
- Галочку у элемента меню можно убрать.
- Команду можно переименовать.

Оператор **Alter Menu Item** используется для изменения элемента меню. Оператор **Alter Menu Item** содержит несколько предложений (**Enable**, **Disable**, **Check**, **UnCheck** и др.), манипулируя которыми Вы можете вносить перечисленные выше изменения.

Программа из комплекта поставки "Overview" ("Обзор") содержит пример создания и последующего изменения нового меню. Программа "Overview" создает следующее новое меню:

```
Create Menu "Обзор" As
  "&Настройка"Calling OverView,
  "(Фиксировать рамку"Calling MenuToggler,
  "(Стиль рамки"Calling PickFrame,
  " (-",
  "Заккрыть программу Обзор"Calling Bye,
  " (-",
  "О программе Обзор..."Calling About
```

Команда **Стиль рамки** вначале недоступна. (Этот режим задается с помощью символа "(", предшествующего имени команды.)

Когда пользователю понадобится сделать доступной команду **Стиль рамки**, например, при открытии окна с рамкой обзора, он применяет следующий оператор:

```
Alter Menu Item PickFrame Enable
```

Если пользователь закрывает окно с рамкой обзора, то снова нужно сделать команду **Стиль рамки** недоступной, а это делается следующим оператором:

```
Alter Menu Item PickFrame Disable
```

"PickFrame" – это имя подпрограммы (sub-процедуры) в файле OVERVIEW.MB. Обратите внимание на то, что "PickFrame" появляется как в операторе **Create Menu** (в предложении **Calling**), так и в операторе **Alter Menu Item**. В операторе **Alter Menu Item** Вы должны задать, какой элемент меню будет изменен. Если Вы определите имя процедуры (например, "Pick-Frame"), MapInfo изменит ту команду, которая ее вызывает.

Таким же образом, для изменения команды **Фиксировать рамку** можно воспользоваться следующим оператором:

```
Alter Menu Item MenuToggler Enable
```

Оператором **Alter Menu Item** можно также изменить имя команды меню. Например, программа "Overview" сначала задает команду под названием **Фиксировать рамку**. Как только эта команда будет выбрана, то программа изменит название команды на **Новая рамка**, для чего используется следующий оператор:

```
Alter Menu Item MenuToggler Text "Новая рамка"
```

MapInfo изменяет команды из стандартной системы меню самостоятельно. Например, команда **Окно > Новая Карта** активна только тогда, когда открыта хотя бы одна таблица, содержащая графические объекты. Рекомендуется оставить право изменения стандартного меню за MapInfo и не пытаться делать это из программы на языке MapBasic.

## Переопределение строки меню

Для удаления меню из строки заголовков используется оператор **Alter Menu Bar**. Например, следующий оператор удаляет с экрана меню **Запрос**:

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

Вы можете также с помощью оператора **Alter Menu Bar** добавить меню в строку заголовков. Например, следующий оператор добавляет меню **Карта** и **Список**. (Обычно эти меню не показываются одновременно; меню **Карта** появляется при открытом окне Карты, а меню **Список** – при открытом Списке).

```
Alter Menu Bar Add "Карта", "Список"
```

Оператор **Alter Menu Bar Add** всегда добавляет заголовок меню в строку меню с правого края. В связи с этим возникает небольшое неудобство из-за того, что строка меню заканчивается меню **Справка**. В большинстве программ принято, чтобы меню **Окно** и **Справка** были последними в строке заголовков. Поэтому Вы, возможно, захотите поместить новое меню слева от меню **Окно**. Например, в программе "Textbox" используется следующий прием для помещения меню перед меню **Окно**:

```
Alter Menu Bar Remove ID 6, ID 7  
Alter Menu Bar Add "Tools", ID 6, ID 7
```

Первый оператор удаляет меню **Окно** (ID 6) и **Справка** (меню с ID 7). Второй оператор добавляет в правый конец строки меню сначала заголовок меню **Программы**, а затем меню **Окно** и **Справка**. В результате меню **Программы** появится левее меню **Окно**.

Более радикально упорядочивает строку меню оператор **Create Menu Bar**. Например, следующий оператор переопределяет строку меню, оставляя в ней заголовки **Файл**, **Правка**, **Карта**, **Запрос**, **Справка** (в приведенном порядке):

```
Create Menu Bar As "Файл", "Правка", "Карта", "Запрос", "Справка"
```

Список стандартных заголовков меню MapInfo Professional ("Файл", "Запрос" и т.д.) приведен в описании оператора **Alter Menu** в *Справочнике MapBasic*. Восстановить стандартную строку меню MapInfo можно оператором **Create Menu Bar As Default**.

## Задание элементов меню на разных языках

В большинстве предыдущих примеров к меню обращались по имени ("Файл" и т.д.). Есть другой способ обратиться к стандартному заголовку меню MapInfo: по его номеру-идентификатору. Например, в любом операторе при обращении к меню **Файл** можно вместо слова "Файл" использовать ID 1. Следующий оператор удаляет меню **Запрос** (которому соответствует идентификатор 3) из строки заголовков:

```
Alter Menu Bar Remove ID 3
```

Если Вы планируете использовать Вашу программу более, чем в одной стране, то снабжайте заголовки меню соответствующими номерами идентификаторами. Если MapInfo Professional представляет собой локализованную версию, то имена меню будут изменены. В русской версии MapInfo обращение к меню "File" не пройдет (как и в любой другой локализованной версии), и будет порождена ошибка. Список идентификаторов, которые соответствуют стандартным меню MapInfo, приведен в описании оператора **Alter Menu** в *Справочнике MapBasic*.

## Настройка быстрых меню MapInfo

MapInfo поддерживает быстрые меню. Это меню, которые появляются, если пользователь нажимает правую кнопку мыши. Чтобы манипулировать такими меню, используйте те же самые операторы, которые Вы использовали бы, чтобы работать со стандартными меню: **Alter Menu**, **Alter Menu Item** и **Create Menu**.

Каждое быстрое меню имеет уникальное имя и ID-номер. Например, быстрое меню, которое появляется, когда Вы нажимаете правую кнопку мышки на Карте, называется "MapperShortcut" и имеет ID 17. Состав этих меню и ID-номера составляющих его команд, приведен в описании оператора **Alter Menu** в *Справочнике MapBasic*.

Чтобы отключить быстрое меню, используйте оператор **Create Menu**, который переопределяет систему меню MapInfo, и задайте управляющий код " (- " как новое определение меню. Например:

```
Create Menu "Спецоперации" ID 17 As "(-"
```

## Назначение одной обрабатывающей процедуры нескольким элементам меню

Операторы **Create Menu** и **Alter Menu** могут содержать предложение **ID**, которое назначает уникальный ID-номер, т.е. идентификатор, каждому новому элементу меню. Этот номер задавать не обязательно; однако с его помощью можно устроить так, чтобы разные команды вызывали одну и ту же процедуру.

В случаях, когда два или более пункта меню вызывают одну и ту же процедуру обработки, вызывается функция **CommandInfo( )** для определения, какой именно элемент меню был использован. Например, следующий оператор создает два новых элемента меню, обращающихся к одному обработчику:

```
Alter Menu "Запрос" Add  
  "Годовой отчет" ID 201 Calling report_sub,  
  "Квартальный отчет" ID 202 Calling report_sub
```

Обе команды обращаются к процедуре "report\_sub", но имеют разные ID-номера; поэтому, вызвав из тела обработчика функцию **CommandInfo( )**, Вы можете определить, какая из команд была выбрана, и поступить соответственно:

```
Sub report_sub  
  If CommandInfo(CMD_INFO_MENUITEM) = 201 Then  
    '  
    ' ... значит, выбрали Годовой отчет ...  
    '  
  ElseIf CommandInfo(CMD_INFO_MENUITEM) = 202 Then  
    '  
    ' ... значит, выбрали Квартальный отчет ...  
    '  
  End If  
End Sub
```



Номера-идентификаторы элементов меню также помогают при изменении элемента меню. Если оператор **Alter Menu Item** идентифицирует элемент меню в соответствии с названием процедуры обработки, MapBasic изменит все пункты меню, вызывающие эту процедуру. Так, следующий оператор отключает активность обеих команд из предыдущего примера (а это, может быть, совсем не то, что нужно):

```
Alter Menu Item report_sub Disable
```

В зависимости от характера Вашей программы, Вы можете захотеть изменить только одну из команд. Следующий оператор отключает активность только команды **Годовой отчет**, но не действует на другие:

```
Alter Menu Item ID 201 Disable
```

Номером-идентификатором может быть любое положительное число.

## Команда MapBasic, эквивалентная выбору команды в меню

Можно активировать команду MapInfo как если бы пользователь выбрал строку меню, для этого используйте оператор **Run Menu Command**. Например, следующий оператор открывает диалог MapInfo Открыть таблицу, как если бы пользователь выполнил команду **Файл > Открыть таблицу**:

```
Run Menu Command M_FILE_OPEN
```

Код M\_FILE\_OPEN определен в файле MENU.DEF.

## Задание сочетаний клавиш

Сочетания клавиш (комбинации клавиш) позволяют пользователю открывать меню, выполнять команды и операции прямо с клавиатуры. В меню обычно применяются клавишные сокращения, представленные подчеркнутой буквой в названии меню или команды. Например, клавишным сокращением для меню **Файл** является ALT+Ф, что явствует из подчеркивания под буквой Ф. При создании элемента меню можно добиться подчеркивания буквы, поместив перед ней знак амперсанда (&).

В следующем фрагменте программы создается команда **Создать рамки**, вызываемая с клавиатуры комбинацией ALT+C:

```
Create Menu "Рамка" As  
    "&Создать рамки..." Calling create_sub,  
    ...
```

"Горячие" клавиши (или акселераторы на жаргоне программистов) являются другой разновидностью клавишных комбинаций. "Горячие" клавиши позволяют выполнять некоторые команды и операции напрямую, без обращения к меню. В следующем примере команде меню сопоставляется "горячая" клавишная комбинация CTRL+Z:

```
Alter Menu "Запрос" Add  
    "Новый отчет" + Chr$(9) + "CTRL+Z/W^%122" Calling new_sub
```

Инструкция "+ Chr\$(9)" вставляет пробел размером в один табулятор между командой и клавишной комбинацией. Табуляторы позволяют единообразно выравнивать "горячие" комбинации с правой стороны меню.

Текст "CTRL+Z" появляется в меню, и пользователь может видеть, что команда снабжена "горячими" клавишами.

Инструкция `"/w^%122"` определяет клавишную комбинацию как одновременное нажатие на клавиши CTRL и Z. Код `"/w^%122"` MapInfo интерпретирует следующим образом: `"/w"` говорит о том, что это код для MapInfo для Windows, символ `"^"` определяет нажатие клавиши CTRL, а код `%122` определяет нажатие клавиши "z" (122 – это ASCII-номер буквы "z").

```
Alter Menu "Запрос" Add
    "Новый отчет /Mz" Calling new_sub
```

Инструкция `"/Mz"` определяет клавишную комбинацию как одновременное нажатие на клавиши CTRL и Z.

Список кодов, управляющих созданием "горячих" клавиш, приведен в описании оператора **CreateMenu** в *Справочнике MapBasic*.

## Управление системой меню через файл MAPINFOW.MNU

Стандартная система меню MapInfo версии 3 содержится в специальном текстовом файле. При желании Вы можете настроить меню MapInfo по своему вкусу, изменив этот файл.

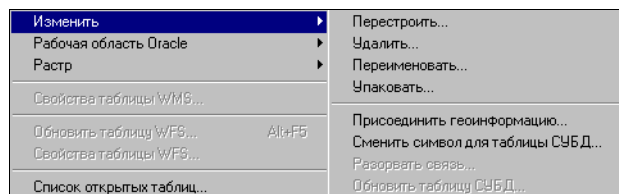
В MapInfo Professional файл меню называется MAPINFOW.MNU.

Это текстовый файл и его можно открыть в любом текстовом редакторе. Если Вы откроете этот файл в текстовом редакторе, то увидите, что он построен как фрагмент программы MapBasic. Если Вы измените описание элемента меню в файле MAPINFOW.MNU, то этот элемент будет выглядеть по-новому при следующем запуске MapInfo. Таким образом, манипулируя текстами в файле MAPINFOW.MNU, Вы можете переопределять стандартную систему меню напрямую, без участия программ MapBasic.

**i** Прежде чем изменять что-либо в файле MAPINFOW.MNU, сохраните его резервную копию. Если файл MAPINFOW.MNU будет испорчен или уничтожен, то MapInfo, возможно, не запустится. Запустить MapInfo можно будет только восстановив файл MAPINFOW.MNU из резервной копии. Если же файл MAPINFOW.MNU поврежден, а резервной копии нет, то Вам придется полностью переустановить пакет MapInfo.

Файл MAPINFOW.MNU содержит несколько операторов **Create Menu**; они и задают стандартную систему меню MapInfo (**Файл**, **Правка** и т.д.). Чтобы удалить один или несколько элементов меню, можно просто удалить соответствующую строку из оператора **Create Menu**.

Например, команда MapInfo **Таблица > Изменить** обычно открывает подменю с командами **Перестроить**, **Удалить**, **Переименовать** и **Упаковать**.



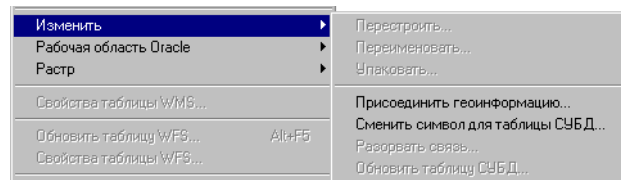
Изучив файл MAPINFOW.MNU, Вы обнаружите, что меню **Изменить** определено через команду **Create Menu** следующим образом:

```
Create Menu "&Изменить" As
    "&Перестроить..."
    HelpMsg "Изменить структуру таблицы."
    calling 404,
    "У&далить..."
    HelpMsg "Удалить таблицу и все связанные с ней компоненты. "
    calling 409,
    "Пере&именовать..."
    HelpMsg "Переименовать таблицу."
    calling 410,
    "&Упаковать..."
    HelpMsg "Упаковать таблицу
        и убрать из таблицы удалённые записи."
    calling 403,
    . . .
```

Команда **Удалить** может быть Вам не нужна, и Вы хотите удалить ее из меню **Изменить**. Для этого можно просто удалить соответствующую строку ("У&далить..." Calling 409) из файла MAPINFOW.MNU; в результате оператор **Create Menu**, описывающий меню **Изменить**, примет следующий вид:

```
Create Menu "&Изменить" As
    "&Перестроить..."
    HelpMsg "Изменить структуру таблицы."
    calling 404,
    "Пере&именовать..."
    HelpMsg "Переименовать таблицу."
    calling 410,
    "&Упаковать..."
    HelpMsg "Упаковать таблицу
        и убрать из таблицы удалённые записи."
    calling 403,
    . . .
```

При следующем запуске MapInfo меню **Таблица > Изменить** предстанет в сокращенном виде, без меню **Удалить**.



Аналогично, чтобы удалить меню полностью, со всеми командами, нужно найти соответствующий оператор **Create Menu Bar** в меню MAPINFOW.MNU и удалить его.

Если пакет MapInfo установлен на сети и Вы изменили файл MAPINFOW.MNU в том же каталоге, в котором установлена программа MapInfo, то изменения в системе меню распространятся на всех пользователей MapInfo в сети. Иногда требуется лишить

пользователей некоторых прав, но оставить эти права за собой или за администратором сети. Например, команду **Удалить** можно скрыть от пользователей, но оставить в распоряжении администратора.

Для того, чтобы создать персональный вариант файла MAPINFOW.MNU для отдельного пользователя, поместите его в личный каталог пользователя. В системе Windows личный каталог – это тот, в котором содержится файл WIN.INI.

Для того, чтобы создать персональный вариант файла MAPINFOW.MNU для отдельного пользователя, поместите его в личный каталог пользователя. Файл с новой структурой меню можно поместить в каталог System, или в каталог Preferences внутри каталога System.

Как только пользователь запускает MapInfo, то сначала проверяется наличие файла MAPINFOW.MNU в личном каталоге пользователя. Если файл MAPINFOW.MNU присутствует в личном каталоге, MapInfo загружает из него систему меню. Если же в личном каталоге MAPINFOW.MNU не найден, то MapInfo загружает систему меню из каталога, в котором установлен пакет MapInfo.

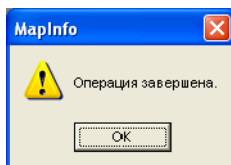
Таким образом, Вы можете добиться того, чтобы один вариант файла MAPINFOW.MNU был доступен всем пользователям, а другие – индивидуальным пользователям. Версия MAPINFOW.MNU, предназначенная для общего пользования, помещается в один каталог с MapInfo, а версии для индивидуальных пользователей – в их личные каталоги.

## Стандартные диалоговые окна

Диалоговые окна (или диалоги) являются существенным элементом интерфейса. В арсенале MapBasic содержится несколько операторов и функций, позволяющих создать диалоги для вашего приложения:

### Показ простого сообщения

Оператор **Note** показывает простейшее диалоговое окно с сообщением и кнопкой **ОК**.



### Показ диалога с двумя кнопками

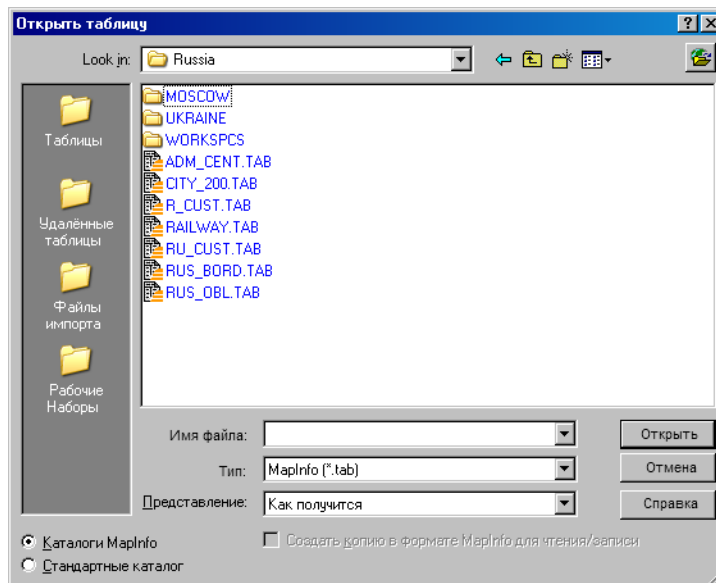
Функция **Ask( )** показывает диалог с вопросом и двумя кнопками. На двух кнопках обычно написано **ОК** и **Cancel** ("Отмена"), но Вы можете использовать свои надписи. Если пользователь выберет кнопку **ОК**, функция вернет логическую истину (TRUE) или вернет логическую ложь (FALSE) в противном случае.



## Диалог открытия файла

Функция **FileOpenDlg( )** показывает стандартный диалог команды **Файл > Открыть**. Если пользователь выберет один из файлов, то функция вернет его имя. Если пользователь нажмет кнопку **Отмена**, функция возвратит пустую строку.

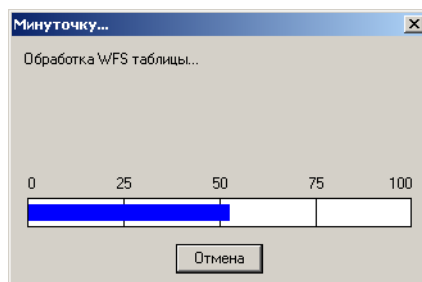
Функция **FileOpenDlg( )** создает диалог, который выглядит так:



Функция **FileSaveAsDlg( )** показывает стандартный диалог Сохранить как, и возвращает имя файла, введенное пользователем.

## Показ диалога-индикатора процента выполнения

Оператор **ProgressBar** показывает диалог с процентом выполнения и кнопкой **Отмена**.



## Отображение одной записи таблицы

В MapInfo нет стандартного диалога, который показывает только одну запись из таблицы. Для этого можно использовать окно Информация. О том, как управлять этим окном из программы MapBasic, смотрите в разделе [Настройка окна Информации на стр. 117](#).

Более подробно упомянутые в этом разделе операторы и функции описаны в *Справочнике MapBasic*. Если Вам недостаточно стандартных диалоговых окон, воспользуйтесь оператором **Dialog** и постройте собственное, новое диалоговое окно.

## Новые диалоговые окна

Оператор **Dialog** позволяет создать новый диалог. Когда оператор **Dialog** выполняется, MapInfo показывает диалог и пользователь может с ним взаимодействовать. Как только пользователь закрывает диалоговое окно (кнопками **OK** или **Cancel**), MapInfo выполняет операторы, следующие за оператором **Dialog**. После выполнения оператора **Dialog** можно с помощью функции **CommandInfo( )** определить, какой кнопкой (**OK** или **Cancel**) был закрыт диалог.

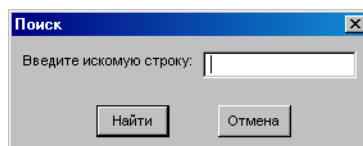
Все, что может появиться в диалоге, называется *управляющими элементами диалога*. Такими элементами, например, являются кнопки **OK** и **Cancel**. Каждый элемент описывается отдельным предложением **Control** в операторе **Dialog**. Например, следующий оператор создает диалог с четырьмя элементами: текст (элемент **StaticText**); окошко, в которое пользователь может ввести текст или числа (элемент **EditText**); кнопки типа "OK" (элемент **OKButton**) и "Отмена" (элемент **CancelButton**).

```
Dim s_searchfor As String

Dialog
  Title "Найти"
  Control StaticText
    Title "Введите искомую строку:"
  Control EditText
    Into s_searchfor
  Control OKButton
  Control CancelButton
```

```
If CommandInfo(CMD_INFO_DLG_OK) Then
    '
    ' ... если нажата кнопка "ОК", то переменная типа
    ' String: s_searchfor будет содержать
    ' значение, введенное пользователем.
    '
End If
```

Этот оператор **Dialog** показывает следующий диалог:



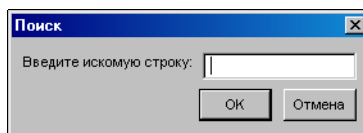
## Размеры и положение элемента диалога

Для изменения ширины и высоты элемента диалога Вы должны включить в его описательное предложение **Control** предложения **Width** и **Height** соответственно. Если Вы хотите сменить положение элемента, то включите в его описание предложение **Position**.

Например, Вы не удовлетворены размещением кнопок по умолчанию в диалоге описанном выше. Чтобы изменить положение кнопок, используем предложение **Position**:

```
Dialog
    Title "Найти"
    Control StaticText
        Title "Введите искомую строку:"
    Control EditText
        Into s_searchfor
    Control OKButton
        Title "Найти"
        Position 30, 30
    Control CancelButton
        Position 90, 30
```

Применение предложения **Position** в двух предложениях **Control** изменяет вид диалога:



Положение и размеры задаются в единицах измерения диалога. Горизонтальная единица равна одной четверти ширины буквы системного шрифта, а вертикальная – одной восьмой. За точку отсчета с координатами 0, 0 принимается верхний левый угол диалога. Следующее предложение **Position** определяет координаты в пять букв от левого края диалога и в две буквы от верхнего края:

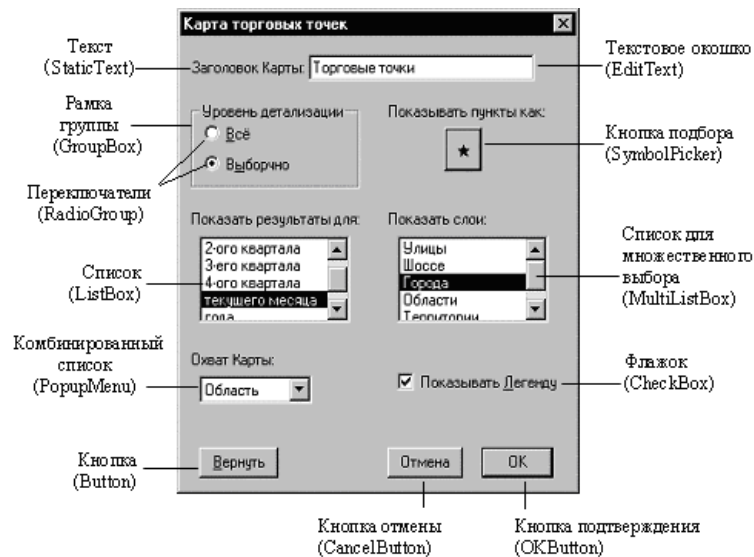
```
Position 20, 16
```

Значение горизонтальной позиции, равное 20, умноженное на четверть ширины символа, определяет ширину в пять символов. Вертикальное значение, равное 16, умноженное на одну восьмую высоты символа, определяет высоту в два символа.

Предложение **Position** может присутствовать в описании каждого элемента диалога. Так же и предложения **Width** и **Height** включаются в описание элемента, размеры которого нужно изменить.

## Элементы окна диалога

Предыдущие примеры представляли четыре вида элементов диалога (текст, окошко ввода, кнопки подтверждения и отмены). Следующий пример показывает все возможные в MapBasic элементы диалога.



### StaticText – неизменяемый текст

Элемент **StaticText** представляет из себя простую надпись. Например:

```
Control StaticText
  Title "Заголовок Карты:"
  Position 5, 10
```

### EditText – окошко ввода текста

Элемент **EditText** – это окошко, в которое можно вводить текст или цифры. Например:

```
Control EditText
  Value "Торговые точки"
  Into s_title
  ID 1
  Position 65, 8 Width 90
```



### Группа (GroupBox)

Элемент **GroupBox** создает рамку с надписью сверху. Рамки выполняют декоративную функцию; они используются для собирания в группы логически связанные элементы. Например:

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

### Кнопки-переключатели (RadioGroup)

Элементы **RadioGroup** работают как переключатели, из которых можно включить только один. Например:

```
Control RadioGroup
  Title "&Всё;В&ыборочно"
  Value 2
  Into i_details
  ID 2
  Position 12, 42 Width 60
```

### Выбор стиля (Picker)

Существует 4 типа кнопок: **PenPicker**, **BrushPicker**, **FontPicker** и **SymbolPicker**. Вызывает один из диалогов подбора пера, штриховки, шрифта или символа соответственно. В приведенном примере использована кнопка **SymbolPicker**, на которой изображен текущий символ. Например:

```
Control SymbolPicker
  Position 95, 45
  Into sym_variable ID 3
```

### Список (ListBox)

Элемент **ListBox** создает список, из которого пользователь может выбрать одну из строчек. MapBasic автоматически добавляет строку прокрутки, если строчек в списке слишком много. Например:

```
Control ListBox
  Title "1-ого квартала;2-ого квартала;3-его квартала;
  4-ого квартала;текущего месяца;года"
  Value 5
  Into i_quarter
  ID 5
  Position 5, 90 Width 65 Height 35
```

### Список множественного выбора (MultiListBox)

Элемент **MultiListBox** позволяет выбрать более чем одну строчку из списка, используя принятую в Windows технику выбора с нажатой клавишей SHIFT или CTRL. Например:

```
Control MultiListBox
  Title "Улицы;Дороги;Города;Области;Территории"
  Value 3
  ID 5
  Position 95, 90 Width 65 Height 35
```

### Комбинированные списки или "всплывающее" меню (PopupMenu)

Элемент **PopupMenu** можно открыть, нажав на кнопку со стрелочкой. Открывается обычный список для выбора значений, визуально похожий на меню. Например:

```
Control PopupMenu
  Title "Город;Область;Регион;Вся страна"
  Value 2
  Into i_scope
  ID 6
  Position 5, 140
```

### Флажок (CheckBox)

Элемент **CheckBox** – это квадратик с подписью. Щелчком на квадрате пользователь может включить или выключить некий режим. Например:

```
Control CheckBox
  Title "Показывать &Легенду"
  Into l_showlegend
  ID 7
  Position 95, 140
```

### Кнопки

Элементы **Button** – кнопки – присутствуют во всех диалогах; по крайней мере, одна из них почти всегда есть. MapBasic поддерживает стандартные кнопки **OKButton** и **CancelButton**, создающие соответственно кнопки **ОК** и **Cancel (Отмена)**, а также позволяет задавать свои собственные.

```
Control Button
  Title "&Вернуть"
  Calling reset_sub
  Position 10, 165
```

```
Control OKButton
  Position 65, 165
  Calling ok_sub
```

```
Control CancelButton
  Position 120, 165
```

В каждом диалоге должно быть не более одной кнопки типа **OKButton** и не более одной кнопки типа **CancelButton**. Можно, конечно, обойтись без них; однако, рекомендуется все же добавлять в диалоге хотя бы одну из двух кнопок **ОК** или **Cancel**, чтобы пользователь мог

закрыть диалог. С каждой из кнопок можно связать процедуру обработчик, и, если Вы нажмете такую кнопку, MapBasic сначала выполнит процедуру, а затем продолжит выполнение программы после оператора **Dialog**.

Каждый элемент диалога детально описан в *Справочнике MapBasic* и в *Справочной системе*. Например, элемент **ListBox** описан в разделе **Control Listbox**.

## Задание начального значения элемента

В описании большинства элементов может содержаться предложение **Value**. Оно определяет значение или установку элемента диалога сразу после его открытия. Например, если Вы хотите, чтобы при открытии диалога был выбран пятый элемент в списке, добавьте следующее предложение **Value** в описание элемента **ListBox**:

```
Value 5
```

Если предложение **Value** опущено, MapInfo самостоятельно подставляет стандартные значения. Например, флажок (элемент **CheckBox**) стандартно установлен. Более подробные сведения о предложениях **Value** приведены в соответствующих описаниях элементов (например, **CheckBox**) в *Справочнике MapBasic*.

## Считывание установок диалога

В описании большинства элементов может содержаться предложение **Info**. Оно определяет переменную, в которой запоминается значение или установка элемента диалога после его закрытия. Для того, чтобы MapBasic запомнил значение или установку в переменной, заданной предложением **Info**, нужно закрыть диалог нажатием на кнопку **OK**.

Переменная, задаваемая после слова **Info**, должна быть задана как локальная или глобальная в Вашей программе и, разумеется, должна по типу соответствовать элементу диалога. Например, переменная для флажка (**CheckBox**) должна быть логической (TRUE означает установку флажка, а FALSE – сброс). В *Справочнике MapBasic* содержится более подробная информация о задании переменных для каждого типа элемента диалога.



После того, как пользователь закроет диалог, нажав на кнопку **OK**, MapBasic помещает значения во все переменные, заданные предложениями **Info**; если же Вы хотите прочитать их в то время, пока диалог еще открыт, воспользуйтесь функцией **ReadControlValue( )**. Эта функция может вызываться только из процедуры-обработчика диалога, которая описана ниже.

---

## Реакция на действия пользователя

Вы можете сопоставить многим элементам диалога процедуру обработчик. Так называется подпрограмма (или процедура), автоматически выполняющаяся тогда, когда пользователь укажет мышкой на элемент диалога. Для сопоставления элементу диалога процедуры-обработчика используется предложение **Calling**; обработчик – это процедура без параметров; она срабатывает в тот момент, когда пользователь выберет или укажет мышкой на элемент диалога; как только процедура-обработчик закончит работу, пользователь может продолжить работу в диалоговом окне (если, конечно, он выбирал элементы, отличные от **OKButton** и **CancelButton**, автоматически закрывающие диалог).

Процедуры-обработчики позволяют программе выполнять действия, оставляя открытым диалог на экране. Например, в диалоге может быть кнопка **Вернуть**; если пользователь нажмет кнопку **Вернуть**, программа восстанавливает первоначальные значения элементов. Чтобы добиться такого эффекта, нужно сопоставить кнопке **Вернуть** процедуру-обработчик, из тела которой действуют операторы или операторы **Alter Control**, восстанавливающие значения элементов диалога.

Обработчики элементов **ListBox** и **MultiListBox** могут определять, сколько раз – один или два – была нажата кнопка мыши. Для этого обработчик использует функцию **Command Info(CMD\_INFO\_DLG\_DBL)**. Пример такого распознавания содержится в программе "Показ Карт" из комплекта поставки (NIEWS.MB). Диалог "Показ Карт" содержит список названий представлений Карт или Видов; если пользователь дважды укажет мышкой на название, то диалог закрывается. (другой способ, традиционный – это выбрать Вид, указав на него один раз и затем нажать кнопку **OK**).

Если два или более элементов в предложении **Calling** обращаются к одной и той же процедуре-обработчику, то эта процедура срабатывает для обоих элементов. Функция **TriggerControl( )** возвращает ID-номер последнего элемента, выбранного в диалоге, тем самым позволяя различать элементы диалога.

Элементы диалога **GroupBox**, **StaticText** и **EditText** не могут иметь обработчиков. Вы можете задать процедуру-обработчик, срабатывающую сразу при открытии диалога. Для этого нужно в оператор **Dialog** включить предложение **Calling**, не входящее ни в одно предложение **Control**, тогда предложение **Calling** будет определять процедуру-обработчик для самого диалога.

Оператор **Alter Control** может быть использован только в теле процедуры-обработчика. Оператор **Alter Control** используется для того, чтобы скрыть, показать, сделать доступным и недоступным элемент диалога или восстановить его начальное значение. С помощью оператора **Alter Control** также можно определить, какой элемент **EditText** имеет фокус (т.е. какой элемент активен). Смотрите также подробное описание оператора **Alter Control** в *Справочнике MapBasic*.

## Доступные и недоступные элементы

Когда элемент диалога появляется на экране, его либо можно использовать, либо нельзя. В последнем случае элемент показывается серым цветом. По умолчанию каждый элемент доступен. Чтобы сделать его недоступным, есть два способа:

- Включить в соответствующее элементу **Dialog** предложение **Control** ключевое слово **Disable**. Как только диалог открывается, элемент показывается серым цветом и недоступен.
- В процедуре-обработчике можно выполнить оператор **Alter Control** и сделать элемент недоступным. Если Вы желаете сделать элемент недоступным изначально, сделайте это в процедуре-обработчике самого диалога; для этого нужно задать имя процедуры в свободном (не входящем ни в какое предложение **Control**) предложении **Calling**. Эта процедура сработает сразу после открытия диалога. В теле этой процедуры должен быть помещен оператор **Alter Control**, отключающий элемент. Этот прием более сложен, но он и более гибок. Например, поместив оператор **Alter Control** в условный оператор **If... Then**, Вы можете делать элемент доступным в зависимости от некоторых условий.

**i** Если Вы планируете воздействовать на элемент диалога оператором **Alter Control**, предварительно присвойте этому элементу номер в предложении **ID** оператора **Dialog**. Пример использования оператора **Alter Control** описан в одноименной главе *Справочника MapBasic*.

---

## Выбор строчки из списка

Элемент **ListBox** представляет окошко списка. Есть два способа создать список в окошке элемента **ListBox**:

- Задать строчное выражение, состоящее из элементов списка, разделенных точками с запятой. Например:

```
Control ListBox  
Title "1-ого квартала;2-ого квартала;3-его квартала;  
4-ого квартала;текущего месяца;года"
```

- Объявить массив строчных переменных (типа String) и разместить каждый элемент списка в этом массиве. В предложении **Control** нужно задать ключевое слово **From Variable** и указать имя этого массива. Например, если Вы создали строчный массив "s\_list", то сделать из него список в элементе **ListBox** можно следующим оператором:

```
Control ListBox  
Title From Variable s_list
```

Предложение **From Variable** можно использовать в описании трех списочных элементов диалога MapBasic (**ListBox**, **MultiListBox** и **PopupMenu**).

## Управление списком типа MultiListBox

Если Ваш диалог содержит список типа **MultiListBox**, то Вы должны использовать процедуру-обработчик, чтобы определить, какие элементы (строчки) списка были выбраны пользователем. В большинстве случаев диалог, содержащий элемент **MultiListBox**, содержит кнопку **OKButton**, которой сопоставлен обработчик. Из тела обработчика кнопки **OKButton** в цикле вызывается функция **ReadControlValue( )**. Первый вызов функции **ReadControlValue( )** возвращает номер первой выбранной строчки списка; следующий вызов возвращает номер второй выбранной строчки и т.д. Как только функция **ReadControlValue( )** возвратит ноль, MapBasic решает, что список выбранных строчек исчерпан. Если же функция **ReadControlValue( )** возвратила ноль при первом же вызове, значит, ни одна строчка списка не была выбрана.

Из тела процедуры-обработчика Вы можете отменить выбор всех строчек в списке элемента **MultiListBox**, задав в операторе **Alter Control** нулевое значение (Value 0). Можно также добавить номера строчек к списку выбранных элементов в операторе **Alter Control**. В следующем примере формируется набор выбранных строчек в списке элемента **MultiListBox** из первой и второй строчки:

```
Alter Control 1 Value 1  
Alter Control 1 Value 2
```

Не забывайте, что функция **ReadControlValue( )** и оператор **Alter Control** требуют номера ID. Чтобы присвоить ID элементу **MultiListBox**, включите необязательный параметр ID в предложение **Control MultiListBox**.

## Сочетания клавиш, соответствующие элементам

В MapBasic-программе, работающей в среде Windows, элементам диалога могут быть назначены клавишные комбинации. Используя клавишные комбинации, пользователь может оперировать в диалоге, не пользуясь мышью.

Клавишные комбинации задаются с помощью знака амперсанта (&) в тексте элемента перед той буквой, которая будет использоваться в клавишной комбинации. В следующем примере в операторе **Control** клавиша "B" задается как клавишное сокращение:

```
Control Button  
    Title "&Вернуть"  
    Calling reset_sub
```

Кнопка "**Вернуть**" теперь может быть нажата без участия мыши, комбинацией ALT+B. Чтобы в тексте элемента знак амперсанта присутствовал как обычный символ, задавайте его как двойной амперсанта (&&).

Элементу **EditText** нельзя сопоставить клавишную комбинацию. Однако, если вы расположили надпись **StaticText** в левой части элемента **EditText** и указали сочетание клавиш для надписи **StaticText**, пользователь может воспользоваться элементом **EditText**, нажав соответствующее **StaticText** сочетание клавиш.

## Модальные и немодальные диалоги

Оператор **Dialog** создаёт модальный диалог. Это значит, что пользователь должен сначала закрыть диалог (например, нажатием кнопок **OK** или **Cancel**), прежде чем продолжить работу в MapInfo Professional.

Некоторые диалоги являются немодальными. Это значит, что пользователь может осуществлять другие действия, пока диалог открыт. Например, диалог **Регистрация раstra** в MapInfo Professional является немодальным. Оператор **Dialog** не может создать немодальный диалог. Если Вы хотите создать немодальный диалог, Вы можете создать приложение на другом языке программирования, таком, например, как Visual Basic, и вызывать это приложение из Вашей MapBasic-программы (например, используя оператор **Run Program**).

## Заккрытие диалога

После того, как программа MapBasic выполнит оператор **Dialog**, он будет пребывать на экране, пока не случится одно из четырех событий:

- пользователь нажмет на кнопку **OKButton** (если она есть)
- пользователь нажмет на кнопку **CancelButton** (если она есть)
- пользователь закроет диалог другим путем (например, нажатием на ESC)
- пользователь выбрал элемент диалога, в процедуре-обработчике которого выполняется оператор **Dialog Remove**.

Обычно диалог закрывается, как только пользователь нажмет кнопки **OKButton** или **CancelButton**. Иногда после нажатия кнопок **OK** или **Cancel** требуется сохранить диалог на экране. Например, после нажатия на кнопку **Cancel** (т.е. выбора элемента **CancelButton**), программа

может выдать запрос на подтверждение отмены (типа "Вы уверены, что хотите вернуться в окно исходного диалога?") Если пользователь ответит "Нет", то программа должна вернуться к исходному варианту диалога.

Оператор **Dialog Preserve** позволяет сохранять на экране диалоговое окно после того, как пользователь нажал на кнопки **OKButton** и **CancelButton**. Оператор **Dialog Preserve** может быть использован только в процедурах обработчиках элементов **OKButton** или **CancelButton**.

Оператор **Dialog Remove** закрывает диалог немедленно. Когда процедура-обработчик элементов управления выполнит оператор **Dialog Remove**, диалог исчезнет. **Dialog Remove** должен действовать из процедуры-обработчика. Оператор **Dialog Remove** может быть использован, например, при двойном указании мышкой на строчку списка элемента **ListBox**. В программе-примере из комплекта поставки **NVIEWS.MB** показано, как обрабатывается двойной щелчок в списке.

## Окна

Программа MapBasic может открывать и манипулировать любым стандартным окном MapInfo: (Карта, Список и т.д.).

Чтобы открыть новое окно, Вы можете воспользоваться одним из следующих операторов: **Map**, **Browse**, **Graph**, **Layout** или **Create Redistrict**. Так как в этих окнах показываются данные из таблиц, то Вы должны проследить за тем, чтобы перед выполнением этих операторов соответствующие таблицы были открыты.

Другие окна MapInfo (Справки, Статистики и т.д.) открываются оператором **Open Window**.

Многие режимы показа окон регулируются оператором **Set Window**. Например, Вы можете оператором **Set Window** задать размер и положение окна. MapBasic поддерживает также специализированные операторы, управляющие конкретными типами окон: Например, оператор **Set Map** позволяет изменять порядок слоев в окне Карты, а с помощью оператора **Set Browse** можно отключить показ сетки в окне Списка.

Каждому окну документа (Карте, Списку, Отчету, Графику или Списку Районов) MapInfo автоматически сопоставляет идентификатор (ID-номер). Этот ID-номер может использоваться различными операторами и функциями как параметр. Например, если открыты два окна Карты, то в операторе **Set Map** нужно задать идентификатор того окна, которое будет изменено.

Чтобы получить значение ID-номера для окна, нужно сразу после его открытия или активации вызвать функцию **FrontWindow( )**. Когда вы первый раз открываете окно (например, используя оператор **Map**), оно становится активным. Например, в программе из комплекта поставки "Overview" сначала выполняется оператор **Map**, открывающий окно Карты, и сразу же вызывается функция **FrontWindow( )**, которая возвращает ID-номер этого окна. Последующие операторы в программе "Overview" используют этот ID-номер.

**i** Несмотря на то, что идентификатор окна в сущности является числом, Вы не должны задавать его явно числами 1, 2 и т.д. Вы можете только заносить в переменную значение, возвращаемое функцией **FrontWindow( )** или **WindowID( )**. Например, ID-номер первого открытого окна возвращает функция `WindowID(1)`. Количество открытых окон можно получить, вызвав функцию **NumWindows( )**.

Функция **WindowInfo( )** возвращает информацию об открытом окне. Например, если Вы хотите узнать, является ли открытое окно окном Карты, вызовите функцию **FrontWindow( )** для определения ID-номера этого окна, а затем функцию **WindowInfo( )** для определения типа этого окна.

Оператор **Close Window** закрывает окно.

## Размер и положение окна

Изменить размер и положение окна можно двумя способами:

- Включить в оператор, открывающий окно, предложения **Position**, **Width** и **Height**. Например, следующий оператор **Map** не только открывает окно Карты, но и задает его положение и размер:

```
Map From world
Position (2,1) Units "in"
Height 3 Units "in"
Width 4 Units "in"
```
- Выполнить оператор **Set Window** уже после открытия окна. Окно в операторе **Set Window** должно быть задано ID-номером.

## Окна Карт

Окно Карты показывает графические объекты из одной или нескольких таблиц. При открытии окна Карты нужно определить таблицы, которые в нем будут показаны; каждая таблица должна быть уже открыта.

В следующем примере:

```
Map From world, worldcap, grid30
```

в окне Карты открываются таблицы WORLD, WORLDCAP и GRID30.

Чтобы добавить слой в окно Карты, используется оператор **Add Map Layer**. Удалить слой можно оператором **Remove Map Layer**. Чтобы временно скрыть слой, не удаляя его из окна Карты, выполните оператор **Set Map**, в котором задайте атрибуту **Display** значение **Off**.

Оператор **Set Map** позволяет управлять множеством режимов представления данных в окне Карты. Выполнение оператора **Set Map** эквивалентно заполнению диалога команды **Карта > Управление слоями** и команды **Карта > Режимы**. Более подробные сведения см. в описании оператора **Set Map** в *Справочнике MapBasic*.

Оператор **Shade** создает тематическую Карту, на которой данные выделены цветом или другим способом в зависимости от условий. Оператор **Shade** может построить тематические Карты следующих типов: диапазоны, круговые и столбчатые диаграммы, размерные символы,



плотность точек и индивидуальных значений. Создав тематическую Карту, MapInfo добавляет ее в виде слоя в активное окно Карты. Чтобы изменить тематическую Карту, используйте оператор **Set Shade**.

Можно также использовать оператор **Create Grid** для создания особого типа тематических Карт, позволяющий проводить новые виды анализа Карт. Тематические картодиаграммы, построенные с помощью регулярных поверхностей, позволяют наглядно визуализировать точечные данные, подобно тому, как раньше делалось с помощью точечных тематических карт или карт, построенных методом значков. Интерполятор дальностей IDW заполняет поверхность значениями, полученными из точек таблицы MapInfo Professional. Такие тематические Карты могут быть использованы в многих отраслях хозяйства, например, в телекоммуникационной, торговле, страховании, традиционных приложениях ГИС и многих других. Этот новый вид тематических Карт и формат растровой поверхности поддерживается открытым API для добавочных растровых grid-форматов и интерполяторов, которые могут быть созданы разработчиками. Подробнее смотрите описание оператора **Create Grid** в *Справочнике MapBasic*. Для изменения тематической растровой поверхности используйте предложение **Inflect** в операторе **Set Map**.

Чтобы изменить проекцию в окне Карты, Вы можете выполнить оператор **Set Map**, в котором задано предложение **CoordSys**. Другой способ изменить проекцию состоит в том, чтобы сохранить таблицу с заданием другой проекции с помощью оператора **Commit Table... As**.

Наличие или отсутствие в окне Карты строк прокрутки контролируется оператором **Set Window**.

## Использование слоя анимации для ускорения перерисовки Карты

Если в операторе **Add Map Layer** присутствует слово **Animate**, то добавленный слой становится анимационным. При перемещении объекта по такому слою перерисовка производится очень быстро, вне зависимости от сложности этого объекта.

Эффект анимации полезен в приложениях, отображающих процессы реального времени, в которых объекты Карты должны часто и быстро перерисовываться. Например, 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 и делают соответствующий слой анимационным:

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

Слой Анимации имеет следующие особенности:

- добавленный слой анимации не показывается в диалоге **Управление Слоями**
- пользователь не имеет интерактивного доступа к этому слою; в частности, он не может использовать инструмент **Информация**.
- каждое окно Карты может иметь только один анимационный слой анимационный слой автоматически становится самым верхним слоем Карты. Добавление второго приводит к тому, что он заменяет собой первый.

- в Рабочих наборах не сохраняется информация об этих слоях.
- Для завершения работы анимационного слоя используйте оператор **Remove Map Layer Animate**.

Просмотреть пример анимации можно, запустив программу ANIMATOR.MBX ("Анимация").

### Рекомендации по эффективному использованию слоя анимации

Слой анимации используется для ускоренной перерисовки небольших участков Карты. Для увеличения скорости работы рекомендуется:

- избегать ситуации, когда окно Карты отображается также в окне Отчета. В противном случае процесс замедлится.
- проверять, что слой анимации изображается только один раз.

Последнее проиллюстрируем на примере. Предположим, что Вы работаете с двумя таблицами: ROADS (таблица, содержащая карту улиц) и TRUCKS (таблица, содержащая точечные объекты, представляющая развозящие грузы автомобили). Пусть окно Карты уже содержит оба слоя. Если Вы захотите преобразовать слой грузовиков в слой анимации, следует использовать оператор:

```
Add Map Layer Trucks Animate
```

Однако, при этом слой Trucks появится в окне Карты дважды – как обычный слой и как слой анимации. Поскольку есть обычный слой, ускорения перерисовки Карты не произойдет и цель не будет достигнута. Другими словами, обновление Карты будет происходить по-прежнему медленно, что делает применение анимированного слоя бессмысленным.

Следующий пример показывает, как быть в таком случае. Перед добавлением указанного выше оператора отключите отображение слоя Truck:

```
' временно отключим обновление экрана
  Set Event Processing Off

' уберем изображение исходного слоя
  Set Map Layer "Trucks" Display Off

' добавим Trucks как слой анимации
  Add Map Layer Trucks Animate

' включим режим обновления экрана
  Set Event Processing On

' теперь все в порядке: есть два слоя Trucks
' а окне Карты. Но исходный слой Trucks
' не отображается и поэтому не замедляет отрисовку
' анимационного слоя Trucks.
```

### Окна Списка

Окно Списка показывает данные в табличной форме. Следующий оператор открывает окно Списка и показывает данные из таблицы WORLD:

```
Browse * From world
```

Звездочка задает показ всех колонок Списка. Если Вы хотите, чтобы в Списке показывались только определенные колонки, замените звездочку на список колонок или выражений, составленных с их участием. Например, следующий оператор открывает окно Списка, которое показывает две колонки из таблицы WORLD:

```
Browse country, capital From world
```

Оператор **Browse** может задавать выражения с участием колонок, задавая тем самым вычисляемые колонки. Например, следующий оператор использует функцию **Format\$ ( )** для форматирования колонки "Население" из таблицы WORLD. В результате вторая колонка окна Списка будет содержать более удобочитаемые данные о населении.

```
Browse country, Format$(Население, ",#") From world
```

Если в операторе **Browse** задано имя колонки, то в окне Списка эту колонку можно редактировать (если только таблица не была открыта только для чтения). Однако, если оператор **Browse** содержит выражение более сложное, чем просто имя колонки, то колонка будет закрыта для редактирования. Таким образом, если Вы хотите, чтобы какая-либо колонка открывалась только для чтения, сделайте из нее выражение.

Выражения, которые задаются в операторе **Browse**, появляются в качестве заголовков в окне Списка. В следующем примере показано, как можно задать свои заголовки колонок (т.е. синонимы):

```
Browse country, Format$(Население, ",#") "НАС" From world
```

Строка "НАС" (население), помещенная сразу после выражения, задающего значения в колонке, станет ее заголовком в окне Списка.

Вы также можете задавать координаты ячейки, которая будет показана в верхнем левом углу Списка; например, оператор в следующем примере помещает в верхний левый угол Списка значение из второй колонки и пятой строки:

```
Browse * From world Row 5 Column 2
```

## Окна Графиков

В окне Графика данные из таблицы и выражения, из них составленные, представляются графическими элементами. В следующем примере в окне Графика будет построен график населения, а названия стран будут надписями у оси:

```
Graph страна, население From world
```

Первый элемент после слова **Graph** интерпретируется как колонка, содержащая надписи у осей; остальные элементы интерпретируются как данные, которые нужно отобразить на Графике. В примере, приведенном выше, отображаемые на Графике значения задаются просто именем колонки, но Вы можете задать любое допустимое числовое выражение.

## Окна Отчётов

В окне Отчета показан эскиз листа. Для того чтобы открыть Отчет, используйте оператор **Layout**.

Обычно в Отчете существуют несколько объектов-рамок. Для того чтобы создать объект-рамку, используйте оператор **Create Frame**. В отчет можно включать любые объекты карты. Например, для того чтобы озаглавить Отчет, создайте оператором **Create Text** текстовый объект.

Отчет можно использовать для создания таблиц. Например, добавьте объекты в Отчет оператором **Insert**, в котором используется ссылка на таблицу с именем типа "Layout1". Однако, строго говоря, объекты отчетов не хранятся в виде таблиц (они сохраняются в Рабочих наборах). Дополнительную информацию о том, как получить доступ к Отчету подобно доступу к таблицам, смотрите в главе [Работа с таблицами](#).

Для хранения объектов в Отчете, требуется использовать систему координат отчета, в которой координаты объекты представлены "бумажными" единицами, такими как дюймы или миллиметры. Подробнее о системе координат Отчета смотрите в главе [Географические и графические объекты](#).

## Окна Районов

Работа с районами начинается с выполнения команды **Create Redistrict**. Оператор **Create Redistrict** управляет всеми режимами работы с районами, которые пользователь может видеть в диалоге команды **Окно > Районирование**.

Начав работу с районами, Вы можете управлять Списком Районов с помощью операторов **Set Redistrict**. Чтобы имитировать выполнение команд из меню **Районирование**, используйте оператор **Run Menu Command**.

Например, чтобы добавить объект в район (так, как это делается с помощью команды **Районирование > Добавить выборку в район**), выполните следующий оператор:

```
Run Menu Command M_REDISTRICT_ASSIGN
```

Завершить работу с районами и закрыть окно Списка Районов можно оператором **Close Window**. Не забывайте, что значения в базовой таблице могут изменяться по мере того, как объекты переходят из района в район. Поэтому после работы с районами нужно сохранить таблицу, если Вы хотите запомнить результаты районирования. Чтобы сохранить таблицу, выполните оператор **Commit**.

Работа с районами подробно описана в документации MapInfo.

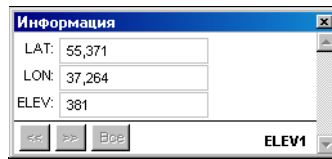
## Окна Сообщений

Оператор языка MapBasic **Print** выводит текст в окно Сообщений. Например:

```
Print "Работает Диспетчер."
```

## Настройка окна Информации

Окно Информации отображает строку из таблицы. Пользователь может редактировать строку, показанную в окне Информации. Чтобы управлять показом и настраивать окно Информации, используйте оператор **Set Window**. На рисунке показано такое окно:



Следующая программа создает настроенное окно Информации, показанное выше.

```
Include "mapbasic.def"
Open Table "World" Interactive
Select
  Country, Capital, Inflat_Rate + 0 "Инфляция"
From World
Into World_Query
Set Window Info
Title "Данные о стране"
Table World_Query Rec 1
Font MakeFont("Arial", 1, 10, BLACK, WHITE)
Width 3 Units "in" Height 1.2 Units "in"
Position (2.5, 1.5) Units "in"
Front
```

Обратите внимание на следующие моменты:

- Обычно окно Информации имеет заголовок "Информация". Эта программа использует предложение **Title**, чтобы изменить заголовок окна на "Данные о Стране."
- Чтобы определить, какая строка данных появится в окне, используйте предложение **Table... Rec** оператора **Set Window**. В примере выше отображается первая запись из таблицы WORLD\_QUERY. (WORLD\_QUERY – временная таблица, созданная оператором **Select**.)
- Окно Информации выводит строку для каждого поля в записи; полоса прокрутки на правой стороне окна позволяет пролистывать содержимое. Чтобы ограничить число отображаемых полей, пример выше использует утверждение **Select**, для формирования временной таблицы запроса, WORLD\_QUERY. Таблица WORLD\_QUERY имеет только три столбца; в результате окно Информации содержит только три поля.

Чтобы сделать некоторые, но не все, поля в окне Информации защищенными от модификации:

1. Используйте оператор **Select**, чтобы создать временную таблицу запроса.
2. Сформируйте оператор **Select** так, чтобы происходило вычисление выражения. Оператор **Select**, показанный выше, определяет выражение " Inflat\_Rate + 0 " для третьей колонки. (Строка "Инфляция", которая следует за выражением – псевдоним (alias) для выражения.)

```
Select
  Country, Capital, Inflat_Rate + 0 "Инфляция"
```

3. В операторе **Set Window Info** используйте предложение **Table... Rec**, чтобы определить, какую запись нужно отобразить. Определите строку из таблицы запроса, как в примере выше. Когда столбец в таблице запроса определен выражением, соответствующее поле в окне Информации – только для чтения. (В примере выше, поле "Инфляция" – только для чтения.)
4. Когда пользователь вводит новое значение в окно Информации, MapInfo автоматически сохраняет новое значение во временной таблице запроса, и в основной таблице, на которой запрос был основан. Вы не должны выполнять дополнительные операторы редактирования таблицы. (Однако необходимо использовать оператор **Commit**, если Вы хотите сохранять внесенные изменения.)

Чтобы сделать все поля в окне Информации недоступными для редактирования, используйте следующий оператор:

```
Set Window Info ReadOnly
```

---

**i** Все поля в окне Информации будут "только для чтения", если Вы отображаете таблицу, которая является объединением (типа таблицы StreetInfo) или таблицы запроса, которая использует предложение **Group By**, для вычисления сводных величин.

---

## Панели инструментов

Панели инструментов являются плавающими по экрану окнами, содержащими одну или более трехмерных кнопок. Нажимая на эти кнопки, пользователь запускает различные операции MapInfo.

Значение терминов "панель кнопок (ButtonPad)" и "панель инструментов (Toolbar)" абсолютно одинаково. Принято называть такие элементы интерфейса MapInfo Professional панелями инструментов. Например, в меню **Настройки** MapInfo Professional существует команда **Панели инструментов**, с помощью которой можно включать и выключать показ отдельных панелей инструментов. При этом, в языке MapBasic панели инструментов вызываются с использованием термина "ButtonPad". Например, оператор **Alter ButtonPad** позволяет показать или скрыть панель инструментов.

MapInfo Professional содержит несколько стандартных панелей инструментов, например панель Операции. MapBasic-программа может добавить новые кнопки к существующим панелям инструментов или создать полностью новую панель инструментов.

## Что происходит при нажатии кнопки?

Так же, как и с командами меню, с кнопками инструментальных панелей связаны соответствующие процедуры-обработчики, которые автоматически выполняются при нажатии пользователем на кнопку. Таким образом, если Вы хотите, чтобы MapBasic показывал некое диалоговое окно при нажатии на определенную кнопку, создайте sub процедуру, показывающую диалог и свяжите эту процедуру с кнопкой.

MapBasic-программа умеет создавать три разных типа кнопок: кнопки-инструменты (ToolButton), кнопки-переключатели (ToggleButton) и кнопки запуска (PushButton). Тип кнопки определяет условия, при которых MapBasic обращается к соответствующему ей обработчику.

- **Кнопки запуска (PushButton):** нажатием на кнопку этого типа пользователь вызывает соответствующую процедуру, а кнопка возвращается в ненажатое положение. Примером запускающей кнопки может служить кнопка **Управление слоями**, вызывающая на экран одноименный диалог. Внешний вид кнопки при этом не изменяется.
- **Кнопки-переключатели (ToggleButton):** нажатие на такую кнопку приводит к ее "залипанию" или "отлипанию". При каждом нажатии MapBasic вызывает процедуру-обработчик.

Примером переключающей кнопки может служить кнопка показа/скрытия окна Легенды. Нажатие на эту кнопку приводит к немедленному появлению окна Легенды, а кнопка "залипает"; вторичное нажатие на кнопку возвращает ее в исходное положение и скрывает окно Легенды.

- **Кнопки-инструменты (ToolButton):** нажатие на кнопку инструмента активизирует один из инструментов MapInfo, и этот инструмент остается активным до тех пор, пока не будет выбран другой. При этом MapBasic вызывает процедуру-обработчик не сразу, а как только пользователь укажет мышью на окно Карты, Списка или Отчета.

**Увеличительная лупа и Уменьшающая лупа** – это типичные кнопки-инструменты; выбор лупы не вызывает никакого действия, они срабатывают только тогда, когда пользователь укажет мышкой на окно Карты.

## Операторы MapBasic, работающие с инструментальными панелями

Следующие операторы и функции позволяют создавать и контролировать кнопки и панели инструментов:

### Create ButtonPad

Этот оператор создает новый элемент типа **ButtonPad** и создает соответствующую пиктограмму для кнопки. Следует определять большую и маленькую кнопку идентификатором файлов ресурсов в виде *N* и *N+1* соответственно.

### Alter ButtonPad

После создания новой инструментальной панели с помощью этого оператора можно изменить положение, размеры, набор кнопок и присутствие инструментальной панели. Оператор **Alter ButtonPad** позволяет изменить положение, показывает или скрывает инструментальную панель и позволяет добавить или удалить кнопки в инструментальной панели.

Оператор **Alter ButtonPad** позволяет также изменять стандартные панели (**Пенал** и др.). Если в программе нужны одна или две кнопки, то их можно добавить в панель **Пенал** или другую стандартную, а не создавать новую панель.

### Alter Button

Этот оператор **Alter Button** используется для изменения статуса активности или выбора кнопки. Используйте оператор **Alter Button** чтобы сделать кнопку доступной или недоступной, а также для изменения выбранной в данный момент кнопки.

## CommandInfo( )

Функция **CommandInfo( )** используется внутри процедуры, обрабатывающей нажатие кнопки и позволяет извлечь информацию о том, как нужно использовать кнопку. Например, если пользователь выбирает кнопку типа **ToolButton** (соответствующую инструменту) и указывает мышью на окно Карты, функция **CommandInfo( )** позволяет прочитать координаты точки, на которую указал пользователь.

Если Ваша программа создает одну или несколько новых кнопок, функция `CommandInfo( CMD_INFO_TOOLBTN )` позволяет определить, какая из них была нажата.

Таким образом, в процедуре обработки кнопки следует вызывать функцию **CommandInfo( )** несколько раз: один раз для определения выбранной пользователем кнопки; один раз для определения координаты X того места, где пользователь щелкнул курсором мыши; один раз для определения координаты Y; и один раз, чтобы определить была ли нажата клавиша SHIFT во время щелчка мыши.

## ToolHandler

**ToolHandler** – это имя специальной процедуры, соответствующей специальной кнопке. Если в программе есть процедура **ToolHandler**, то в инструментальную панель **Операции** будет добавлена новая кнопка типа **ToolButton**, последующие нажатия на которую запускают процедуру **ToolHandler**, которая срабатывает не сразу, а как только пользователь укажет мышью на окно Карты, Списка или Отчета.

MapBasic-программа не может изменить ни пиктограмму на кнопке (в виде знака +), ни присвоить ей другую функцию, кроме заданной процедурой **ToolHandler**. Создать новые кнопки или курсоры вместо процедуры **ToolHandler** можно операторами **Create ButtonPad** или **Alter ButtonPad**.

Если пользователь запустил несколько программ MapBasic одновременно и каждая из них содержит процедуру **ToolHandler**, то каждая из программ добавляет в панель **Операции** свою кнопку.

## Создание кнопки типа PushButton

Следующая программа создает новую инструментальную панель, содержащую кнопку типа **PushButton**. Процедура "button\_prompt" является обработчиком события, состоящего в нажатии этой кнопки, т.е. как только пользователь нажмет эту кнопку, MapBasic автоматически вызовет процедуру "button\_prompt".

```
Include "icons.def"
Declare Sub Main
Declare Sub button_prompt

Sub Main
    Create ButtonPad "Отчеты" As
        PushButton
        Icon MI_ICON_ZOOM_QUESTION
        Calling button_prompt
        HelpMsg "Нажатие на эту кнопку открывает диалог Запрос\Запрос"
```



```
Show
End Sub
Sub button_prompt
    ' Эта процедура выполняется всякий раз,
    ' когда пользователь нажимает на кнопку.
    ' ...
End Sub
```

Процедура **Main** содержит только один оператор: **Create ButtonPad**. Этот оператор создает новую инструментальную панель по имени "Мои Кнопки" и помещает в нее одну кнопку.

Ключевое слово **PushButton** задает MapBasic тип кнопки.

Предложение **Icon** инструктирует MapBasic, какую пиктограмму нужно поместить на кнопку. Идентификатор **MI\_ICON\_ZOOM\_QUESTION** определен в файле **ICONS.DEF**. В этом файле определены идентификаторы для стандартных пиктограмм MapInfo.

Предложение **Calling** сообщает MapBasic, что при нажатии кнопки нужно вызвать процедуру "button\_prompt".

Предложение **HelpMsg** определяет поясняющую строку для кнопки. Для просмотра в строке сообщений этих пояснений нужно указать мышью на кнопку и придержать ее нажатой.

Работа с подсказками обсуждается в разделе [Добавление подсказок к кнопкам на стр. 126](#).

Замечания о размере изображения на кнопке можно найти в описании оператора **Create ButtonPad** в *Справочнике MapBasic*.

## Добавление новой кнопки в панель Операции

В предыдущем примере оператор **Create ButtonPad** создавал новую панель. MapBasic может также добавлять кнопки в стандартные панели MapInfo. Для этого используется оператор **Alter ButtonPad**, а не **Create ButtonPad**:

```
Alter ButtonPad "Операции"
    Add Separator
    Add PushButton
        Icon MI_ICON_ZOOM_QUESTION
        Calling button_prompt
        HelpMsg "Нажатие на эту кнопку открывает диалог Запрос\Запрос"
    Show
```

Предложение **Add PushButton** добавляет в панель **Операции** новую кнопку, а предложение **Add Separator** добавляет пустое место между новой кнопкой и стандартными. Предложение **Add Separator** не обязательно и используется для собирания кнопок в группы.

Одна из стандартных панелей MapInfo, **Программы**, предназначена для размещения новых кнопок, создаваемых прикладными программами MapBasic. Например, программа "Масштабная линейка" (**ScaleBar**) добавляет кнопку в панель **Программы**.

## Создание кнопки типа ToolButton

В предыдущем примере мы создали кнопку типа **PushButton**. Кнопки типа **ToolButtons** предназначены для включения инструментальных средств MapInfo, таких, как обе **Лупы** или **Линия**. Если в программе создана кнопка типа **ToolButton**, пользователь может нажать эту кнопку и включить инструмент, после чего пользователь указывает инструментом (мышкой) или даже перемещает объекты в окнах Карты, Списка или Отчета.

В следующем примере создается новая кнопка типа **ToolButton**. После выбора инструмента пользователь оперирует мышкой в окне Карты. По мере того, как пользователь, нажав кнопку мыши, перемещает курсор, MapInfo рисует линию, динамически следующую за курсором, соединяющую указатель мыши и точку, в которой начал действовать инструмент.

```
Include "icons.def"
Include "mapbasic.def"
Declare Sub Main
Declare Sub draw_via_button
Sub Main
    Create ButtonPad "Отчеты" As
        ToolButton
        Icon MI_ICON_LINE
        DrawMode DM_CUSTOM_LINE
        Cursor MI_CURSOR_CROSSHAIR
        Calling draw_via_button
        HelpMsg "Этот инструмент рисует линию.\nСоздание линии"
    Show
End Sub
Sub draw_via_button
    Dim x1, y1, x2, y2 As Float
    If WindowInfo(FrontWindow(), WIN_INFO_TYPE) <> WIN_MAPPER Then
        Note "Инструмент используется только в окне Карты!"
        Exit Sub
    End If

    ' Запомнить место, в котором начал действовать инструмент:
    x1 = CommandInfo(CMD_INFO_X)
    y1 = CommandInfo(CMD_INFO_Y)
    x2 = CommandInfo(CMD_INFO_X2)
    y2 = CommandInfo(CMD_INFO_Y2)

    ' Используя значения x1, y1, x2 и y2, можно создавать объект.
End Sub
```

В этом примере оператор **Create ButtonPad** содержит слово **ToolButton** вместо **PushButton**. Это говорит MapBasic, что кнопка будет использоваться как инструмент.

В определении **ToolButton** входит предложение **DrawMode**. Это предложение сообщает MapBasic, может ли пользователь, нажав кнопку мыши, перемещать объекты. В предыдущем примере задается режим **DM\_CUSTOM\_LINE**; это значит, что пользователь может использовать инструмент точно так же, как стандартный инструмент **Линия**. Если бы был задан режим **DM\_CUSTOM\_POINT**, пользователь не смог бы перемещать объекты мышкой с нажатой кнопкой. Список режимов рисования можно найти в разделе об операторе **Alter ButtonPad** *Справочника MapBasic* или в *Справочной системе*.

Предложение **DrawMode** также позволяет определить, увидит ли что-нибудь пользователь на экране, перемещая мышь с нажатой кнопкой. В режиме DM\_CUSTOM\_LINE, MapBasic показывает линию между точкой начала рисования и текущим положением курсора до тех пор, пока пользователь не отпустит кнопку мыши. Если задан режим DM\_CUSTOM\_RECT, то MapBasic рисует динамический прямоугольник, следуя за перемещением курсора. Вне зависимости от того, какой задан режим **DrawMode**, MapInfo вызывает процедуру-обработчик для кнопки после того, как пользователь отпустит кнопку мыши. В процедуре обработчика может быть использована функция **CommandInfo( )** для определения места, на которое пользователь указал мышкой.

---

**i** Если пользователь отменит операцию (например, нажатием на ESC при перемещении указателя мыши), MapInfo не вызывает обработчик.

---

## Выбор пиктограммы для создаваемой кнопки

Когда Вы определяете новую кнопку, задавайте пиктограмму, которая должна появиться на кнопке. Определить, какая пиктограмма нужна, позволяет предложение **Icon**.

Ключевое слово Icon сопровождается кодом из файла ICONS.DEF. Например, следующий оператор определяет кнопку, которая использует пиктограмму для кнопки **Информация**. Код MI\_ICON\_INFO определен в файле ICONS.DEF.

```
Alter ButtonPad "Операции"  
  Add Separator  
  Add PushButton  
    Icon MI_ICON_INFO  
    Calling_procedure_name
```

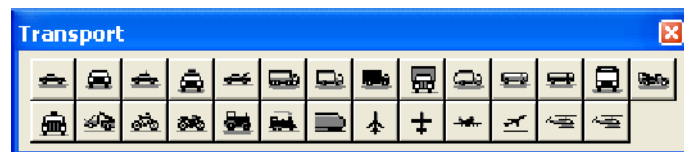
---

**i** MapInfo содержит много встроенных пиктограмм, большинство из которых не используются в интерфейсе пользователя MapInfo. Чтобы увидеть такую пиктограмму, запустите программу из комплекта поставки ICONDEMO.MBX. Чтобы видеть код для конкретной пиктограммы, задержите курсор мышки над нужной кнопкой.

---

Подсказка для этой кнопки покажет Вам код пиктограммы. Вы также можете скопировать код в Буфер Обмена:

1. Запустите приложение ICONDEMO.MBX.
2. Выполните команду **Пиктограммы > Показать**. Появится инструментальная панель:



3. Нажмите на соответствующую кнопку. Появится окно диалога.



4. Нажмите клавиши CTRL+C (клавишное сочетание для команды копирования в Буфер Обмена).
5. Нажмите **ОК**, чтобы закрыть диалог.
6. Переключитесь в окно MapBasic. Нажмите клавишу CTRL+V (клавишное сочетание для команды вставки из Буфера Обмена).

## Как выбрать объект, на который указали мышкой

Если пользователь выберет кнопку типа **ToolButton** и затем укажет на объект Карты, то объект не выбирается; однако, MapInfo вызывает для этой кнопки процедуру-обработчик. Если Вам нужно, чтобы объект, на который Вы указали, был выбран, выполните в теле обработчика оператор **Select**.

В следующем примере выбирается область, в границах которой находился указатель и была нажата кнопка мыши. Сначала функция **CommandInfo( )** определяет, в какой из точек окна была нажата кнопка мыши. Затем, для того чтобы выбрать объекты, используется оператор **Select** с предложением **Where** и указанием географического оператора, такого как **Contains**. В следующем примере выбираются все города региона на котором находился указатель и была нажата кнопка мыши.

```
Sub t_click_handle
  Dim fx, fy As Float

  fx = CommandInfo(CMD_INFO_X)
  fy = CommandInfo(CMD_INFO_Y)
  Select * From towns
    Where obj Contains CreatePoint(fx, fy)

End Sub
```

---

**i** Вместо использования оператора **Select** Вы могли бы сначала воспользоваться функциями **SearchPoint( )** или **SearchRect( )**, а затем с помощью **SearchInfo( )** обработать результат поиска. Пример такого подхода Вы можете найти в описании функции **SearchInfo( )** в *Справочнике MapBasic*.

---

Другой способ состоит в создании процедуры **SelChangedHandler**. Если в программе задана процедура с именем **SelChangedHandler**, MapInfo автоматически вызывает ее каждый раз, когда изменяется выборка. Пользователь может выбирать объект стандартным инструментом Стрелка из инструментальной панели **Операции**, и программа может реагировать на это событие из тела процедуры **SelChangedHandler**.

## Вставка стандартных кнопок в панели, созданные в программе


Вы можете вставить любую кнопку из стандартной панели MapInfo (например, кнопку **Выбор**) в свою инструментальную панель. Например, в следующем примере создается инструментальная панель, содержащая две кнопки: стандартную кнопку **Выбор** и созданную в программе.

```
Create ButtonPad "Мои Кнопки" As
  ' Здесь вставляется стандартная кнопка...
  ToolButton
    Icon MI_ICON_ARROW
    Calling M_TOOLS_SELECTOR
    HelpMsg "Выбор объекта для редактирования\nВыбор"
  ' Здесь вставляется новая кнопка...
  ToolButton
    Icon MI_ICON_LINE
    DrawMode DM_CUSTOM_LINE
    Calling sub_procedure_name
    HelpMsg "Новый маршрут\nНовый"
```

Первое предложение **Calling** задает номер инструмента M\_TOOLS\_SELECTOR, заданный в файле MENU.DEF. Этот номер соответствует инструменту **Выбор**. Каждая стандартная кнопка MapInfo имеет свой номер в MENU.DEF. Вторая кнопка – новая, поэтому предложению **Calling** для нее содержит имя процедуры, а не числовой код.

В описании новой кнопки включено предложение **DrawMode**, но в первой такого предложения нет, потому что для стандартной кнопки режим рисования менять нельзя. Если в дополнительную панель инструментов поместить стандартную кнопку, то необходимо исключить параметр оператора **DrawMode**, потому что для каждой стандартной кнопки MapInfo Professional заранее определены режимы рисования. Задавать режим рисования с помощью параметра оператора **DrawMode** следует, только при создании дополнительной кнопки инструмента **ToolButton**.


---

 Кнопки типов ToolButton и ToggleButton не взаимозаменяемы. Нельзя просто так в тексте программы заменить слово **ToolButton** на **ToggleButton** и наоборот. **ToolButtons** возвращает координаты X/Y места указания мышкой. **ToggleButtons** не возвращает координат, а срабатывает немедленно после нажатия на нее.

---

Если Вы добавляете стандартные кнопки MapInfo в новые Инструментальные панели, то следите за тем, чтобы не перепутать слова "ToolButton" и "ToggleButton". Вам следует ознакомиться с определениями кнопок и Инструментальных панелей в текстовом файле MAP-INFOW.MNU. Файл меню содержит оператор **Create ButtonPad** который определяет кнопки MapInfo.

---

 Можно скопировать определение кнопок из MAPINFOW.MNU и вставить код в свою программу.

---

## Добавление подсказок к кнопкам

Если пользователю не ясно назначение кнопки на Инструментальной панели. MapBasic позволяет Вам снабдить кнопку одним из двух видов подсказок:

- **Подсказка в строке состояния.** Появляется, если пользователь нажал и удерживает кнопку. Сообщение остается на экране все время, пока кнопка нажата.
- **Всплывающая подсказка.** Появляется, если указатель мышки некоторое время находится над кнопкой.

В ранних версиях MapInfo Professional подсказки в строке состояния появлялись только после того, как пользователь нажимал на кнопку. Начиная с версии 4.0, подсказки обоих типов появляются, если курсор пересекает область кнопки панели инструментов.

Оба типа подсказки определяются в предложении **HelpMsg** в операторах **Create Button Pad** и **Alter Button Pad**. Внутри предложения **HelpMsg** первая часть строки содержит сообщения в строке состояния, затем следует символ `\n` и сообщение для “всплывающей” подсказки.

Например:

```
Create ButtonPad "Отчеты" As
  PushButton
    Icon MI_ICON_ZOOM_QUESTION
    Calling generate_report
    HelpMsg "Эта кнопка создает Отчет\nСоздание Отчёта"
  Show
```

В этом примере текст в строке сообщений – “Эта кнопка создает Отчет”, а для подсказки – “Создание Отчета”. Чтобы показать или скрыть строку сообщений, используется оператор **StatusBar**.

## Закрепление панели в верхней части экрана.

Оператор **Alter Button Pad** позволяет превратить Инструментальную панель в строку из кнопок в верхней части экрана. Например:

```
Alter ButtonPad "Операции" Fixed
```

Ключевое слово **Fixed** позволяет зафиксировать панель вверх. Чтобы вернуть ее в прежнее “плавающее” состояние используйте ключевое слово **Float** вместо **Fixed**. Ключевые слова **Fixed** и **Float** допускаются в операторе **Create Button Pad**, так что Вы можете контролировать внешний вид панели уже при ее создании.

Текущий вид панели можно узнать при помощи функции **ButtonPadInfo( )**.

## Другие свойства инструментальных панелей

MapBasic также поддерживает следующие свойства инструментальных панелей:

- **Доступность/Недоступность кнопок.** Программа MapBasic может делать кнопки доступными или недоступными. Детали этого процесса описаны в *Справочнике MapBasic*, в разделе **Alter ButtonPad**.
- **Пиктограммы (рисунки) на кнопках.** С помощью редактора ресурсов Вы можете создавать пиктограммы для кнопок инструментальных панелей MapBasic.

- **Новые курсоры.** Курсор – это форма указателя мыши. Инструменты MapBasic обычно используют традиционный для Windows курсор в форме стрелки-указателя. Однако, с помощью редактора ресурсов можно создавать новые курсоры.

Редактор ресурсов не входит в комплект MapBasic. Однако программы MapBasic могут использовать файлы иконок и курсоров, созданных внешними редакторами. Более подробно создание и использование внешних графических ресурсов описано в главе [Интегрированная картография](#).

## Курсоры

Пользователи MapInfo Professional могут менять форму курсора-перекрестья нажатием на клавишу "X". Курсоры в MapInfo Professional независимы от курсоров в программах MapBasic, поэтому изменение курсора в MapInfo Professional не меняет текущий курсор в MapBasic и наоборот.

Нет никакого способа из MapBasic поменять форму курсора-перекрестья в MapInfo Professional, равно как и нет способа в программах MapBasic переключать курсоры клавишей "X". Однако, можно программно заменять курсоры в MapBasic. Файл ICONS.DEF содержит следующие определения курсоров-перекрестий:

MI_CURSOR_CROSSHAIR	138	малый синий курсор
MI_CURSOR_LRG_CROSSHAIR	164	большой синий курсор
MI_CURSOR_TINY_CROSSHAIR	165	малый черный курсор



В следующем примере задается текущий курсор:

```
Create ButtonPad "TestCursor" as ToolButton
    calling my_handler cursor MI_CURSOR_TINY_CROSSHAIR
```

## Запуск программы в среде MapInfo

В предыдущих разделах было описано, как пользователь может снабдить свою MapBasic-программу новыми меню, диалогами, окнами и инструментальными панелями. Осталось прояснить еще один момент: как запустить программу и задействовать все новые интерфейсные элементы?

В среде MapInfo можно запустить MapBasic-программу командой **Программы > Запустить программу MapBasic**. Вы можете также устроить свою программу так, что она будет запускаться автоматически при запуске MapInfo, чтобы каждый раз не использовать меню **Программы > Запустить программу MapBasic**. Например, в случаях, когда Вы создаете специализированные системы для пользователей, которые не имеют достаточного опыта работы в MapInfo.

В системе Windows Вы можете запустить MapInfo с дополнительными параметрами в командной строке, т.е. задать программу, которая будет запускаться при открытии MapInfo. Щелкните правой клавишей мыши по ярлыку, выберите команду **Свойства** и найдите закладку **Ярлык (Shortcut)**.

Обычно MapInfo показывает при открытии диалог **Открыть сразу** (Вы, кстати, можете отключить его показ в одном из диалогов команды **Настройка > Режимы**.) Если Вы добавите название MapBasic программы в командную строку MapInfo, то диалог **Открыть сразу** появляться не будет. В зависимости от приложения, такое поведение может либо быть желаемым, либо нет. Если Вы хотите, чтобы и этот диалог появлялся, и программа запускалась, то Вам нужно использовать другой прием. Вместо создания командной строки создать свой стартовый Рабочий набор, называемый Startup.

## Запуск программ из Рабочего набора STARTUP

“Startup” – это специальное имя для Рабочего набора, который автоматически загружается при старте MapInfo. Если в этом Рабочем наборе содержится оператор **Run Application**, MapInfo запускает заданную в нем программу.

Например, чтобы автоматически запускать программу "Масштаб" ("Scalebar"), нужно создать следующий Рабочий набор STARTUP.WOR:

```
!Workspace
!Version 600
!Charset Neutral
Run Application "scalebar.mbx"
```

По первым трём строчкам MapInfo распознает файл Рабочего набора. В четвёртой строчке оператором **Run Application** задается запуск программы MapBasic.

Наличие стартового Рабочего набора никак не влияет на диалог **Открыть сразу**, сопровождающий открытие MapInfo. Сначала загружается стартовый Рабочий набор (если он есть), а затем показывается диалог **Открыть сразу** (если только пользователь не отключил его показ).

В системе Windows стартовый Рабочий набор называется STARTUP.WOR и может быть помещен в один каталог с MapInfo или в личный каталог пользователя Windows (там, где находится файл WIN.INI). Если файлы STARTUP.WOR есть в обоих каталогах, то при запуске MapInfo загружаются оба Рабочих набора.

При работе в сети, если Вы хотите, чтобы стартовый Рабочий набор открывался для всех пользователей сети, помещайте его в один каталог с MapInfo. Если же Вы не хотите, чтобы все пользователи употребляли один и тот же Рабочий набор, поместите его в другой подходящий каталог (например, в Windows это может быть каталог, где пользователь содержит свой личный файл WIN.INI).

## Доступ к Рабочим наборам из программы MapBasic

Так как Рабочие наборы представляют из себя текстовые файлы, Вы можете их создавать и редактировать в любом текстовом редакторе. Более того, программа MapBasic может с помощью средств построчного ввода-вывода автоматически управлять содержимым Рабочего набора.



Чтобы воочию увидеть, как это делается, сделайте следующее:

1. Выполните команду MapInfo **Программы > Запустить программу MapBasic** и запустите программу "Textbox" ("Рамка").
2. Выполните команду **Программы > Рамка > О программе "Рамка"**, чтобы показать диалог **О программе "Рамка"**.
3. Нажмите на кнопку **Автозагрузка**. MapInfo покажет диалог, в котором Вы можете назначить режим автоматической загрузки программы "Рамка" при открытии MapInfo.
4. Нажмите кнопку **ОК** в диалоге **Автоматическая загрузка**. MapInfo выдаст сообщение о том, что программа "Рамка" стала автоматически загружаемой. Закройте диалог **О программе "Рамка"** кнопкой **ОК**.
5. Перезапустите MapInfo. В новой сессии MapInfo Professional программа "TextBox" запустится автоматически; вам не нужно выбирать меню **Программы > Запустить программу MapBasic**.

В тот момент, когда на шаге 4 Вы нажимаете на кнопку **ОК**, название программы "TextBox" появляется в операторе **Run Application** в стартовом Рабочем наборе. Если файл стартового Рабочего набора не существует, программа "TextBox" создает его.

Управление содержимым стартового Рабочего набора осуществляется через функции и процедуры модуля AUTO\_LIB.MB. Многие из примеров программ, поставляемых вместе с MapInfo Professional, используют такие функции; например, пользователь MapInfo Professional может настроить программу "Масштабная линейка" таким образом, чтобы эта программа запускалась автоматически, нажав кнопку **Автозагрузка** в диалоге **О программе**.

Текст программного модуля AUTO\_LIB.MB включен в комплект поставки MapBasic. Если потребуется включить в Вашу программу автоматическую загрузку, следуйте инструкциям в комментариях в самом начале файла AUTO\_LIB.MB.

## Рекомендации по повышению производительности

### Слой анимации

Если Вы часто вносите изменения в окне Карты, используйте слой анимации, это позволит делать перерисовку окна быстрее. Слой анимации описан выше в главе **Использование слоя анимации для ускорения перерисовки Карты на стр. 113**.

### Как избегать ненужных перерисовок Окна

При каждом изменении в окне Карты MapInfo перерисовывает его. Если происходит несколько изменений подряд, перерисовка производится после каждого, что неудобно.


Есть два способа избежать этого:

- Используйте оператор **Set ... Redraw Off**. Затем напишите все операторы, изменяющие окно Карты, и, наконец, оператор **Set ... Redraw On**. Окно будет перерисовано один раз.
- Предотвратить перерисовку всех окон MapInfo можно при помощи пары операторов **Set Event Processing Off** и **Set Event Processing On**.

## Очистка Окна Сообщения

Оператор **Print** печатает текст в окне Сообщения.

---

 Обратите внимание, что вывод большого количества текста может замедлять выполнение последующих операторов **Print**.

---

Если Ваша программа печатает много текста в окне Сообщения, Вы должны периодически очищать окно, используя оператор `Print Chr$(12)`.

## Подавление изображения индикатора выполнения (диалог “Минуточку”).

Если ваше приложение минимизировало окно MapInfo, Вам следует отключить показ индикатора выполнения (диалог **Минуточку**) при помощи оператора **Set ProgressBar Off**.

Если этот индикатор отображается на экране, в то время как окно MapInfo минимизировано, индикатор “зависает”. Если Вы скроете диалог **Минуточку**, работа по-прежнему будет выполняться, даже если окно MapInfo минимизировано.

# Работа с таблицами

MapBasic предоставляет полный набор средств работы с таблицами. Например, Вы можете изменять структуру таблицы с помощью оператора **Alter Table** или переходить к определенной записи в таблице с помощью оператора **Fetch**. Оператор **Import** позволяет создать таблицу в формате MapInfo из текстового файла, а оператор **Export** – перевести таблицу в другой формат.

В данной главе описаны операторы и функции языка MapBasic, использующиеся для работы с таблицами. Подробное описание каждого из этих операторов и функций можно найти в *Справочнике MapBasic*.

## В этой главе:

- ♦ Открытие таблиц с помощью MapBasic.....132
- ♦ Создание новых таблиц .....140
- ♦ Доступ к Косметическому слою.....144
- ♦ Доступ к окнам Отчетов .....145
- ♦ Редактирование в многопользовательской среде .....146
- ♦ Файлы-компоненты таблицы.....150
- ♦ Таблицы, содержащие растровые изображения .....150
- ♦ Работа с метаданными .....152
- ♦ Работа со сшитыми таблицами .....155
- ♦ Доступ к удаленным базам данных .....156
- ♦ Доступ к удаленным базам при помощи связанных таблиц159
- ♦ Повышение производительности при работе с таблицами.160

## Открытие таблиц с помощью MapBasic

Чтобы таблица стала доступна в приложении на языке MapBasic, эту таблицу надо сначала открыть. Это можно сделать с помощью оператора **Open Table**. Например, следующий оператор открывает таблицу WORLD:

```
Open Table "C:\mapinfo\data\world"
```

Обратите внимание, что оператор **Browse** идентифицирует таблицу по ее псевдониму "Earth". Псевдоним таблицы действует, пока таблица открыта. Сама таблица не переименовывается. Для переименования, используйте оператор **Rename Table**.

Если используется дополнительное предложение **Interactive** в операторе **Open Table**, и если таблица, которую Вы определяете, не найдена в указанном каталоге, то MapInfo откроет диалог, подсказывающий, где можно искать таблицу. Если Вы пропускаете ключевое слово **Interactive** и таблица не обнаруживается в указанном месте, то оператор **Open Table** сгенерирует сообщение об ошибке.

### Имена таблиц во время выполнения программы

Имя таблицы в MapBasic может представлять собой строковое выражение или константу. Например, если открыты таблицы States, Pipeline и Parcels, то в своей программе Вы можете явно использовать их имена:

```
Select * From States  
Browse * From Pipeline  
i = NumCols (Parcels)
```

Впрочем, Вы можете и не ограничиваться именами константами. Вы можете, например, предложить пользователю выбрать таблицу из списка всех открытых таблиц. Так как заранее не известно какая таблица будет открыта, то имя не может использоваться непосредственно в коде программы.

Можно использовать строковую переменную в качестве имени таблицы. Предположим, чтобы открыть таблицу Zoning, можно выполнить следующие действия:

```
Dim work_table As String  
work_table = "Zoning"  
Browse * From work_table
```

### Как открыть две таблицы с одинаковыми именами

Если Вы попытаетесь открыть две таблицы с одинаковыми именами, MapInfo присвоит им различные стандартные псевдонимы. Например, если Вы откроете таблицу "C:\data1994\sites", MapInfo присвоит ей стандартный псевдоним обычным образом ("sites"); но при открытии еще одной таблицы с таким же именем (допустим, "C:\backup\sites"), MapInfo присвоит второй таблице стандартный псевдоним, отличающийся от стандартного псевдонима для первой таблицы. Таким образом все таблицы будут иметь уникальные псевдонимы. В нашем примере MapInfo присвоит второй таблице псевдоним "sites\_2."

Если в операторе **Open Table** использовать предложение **Interactive**, то MapInfo покажет диалог, в котором пользователь сможет задать свой псевдоним. Если же не использовать предложение **Interactive**, MapInfo создаст стандартный псевдоним автоматически.

Вследствие этого, Вы не можете точно быть уверены, какой стандартный псевдоним будет сопоставлен той или иной открываемой таблице.

Однако узнать стандартный псевдоним можно с помощью функции **TableInfo( )**, например так:

```
Include "mapbasic.def"  
Dim s_filename As String  
Open Table "states" Interactive  
s_filename = TableInfo(0, TAB_INFO_NAME)  
Browse * from s_filename
```

Функция **TableInfo(0, TAB\_INFO\_NAME)** возвращает имя псевдоним последней открытой таблицы.

## Как открыть файл, не являющийся таблицей MapInfo

Вы можете получать доступ к файлам, в которых данные хранятся не в формате таблиц MapInfo (dBASE, Lotus, Excel или текстовым файлам). Но прежде чем Вы получите возможность работать с ними из MapBasic, Вы должны зарегистрировать файлы такого типа. При регистрации файла MapInfo строит файл таблицы (TAB) для файла во внешнем формате. Регистрацию достаточно провести один раз. После регистрации с файлом во внешнем формате можно будет работать так же, как с таблицей в формате MapInfo.

Следующий оператор регистрирует файл в формате dBASE:

```
Register Table "income.dbf" Type DBF
```

После регистрации файл можно считать обычной таблицей, Вы можете открыть его так же, как и обычную таблицу MapInfo с помощью оператора **Open Table**.

```
Open Table "income" Interactive
```

MapInfo проводит поиск по таблицам независимо от того, в каком именно формате они хранятся. Например, выбирать данные оператором **Select** (используя возможности языка запросов SQL) можно из любых таблиц, хранятся ли они в формате базы данных или электронной таблицы.

Что же касается внесения изменений в таблицы, то здесь действия MapInfo отчасти зависят от формата данных в таблице. Если таблица базируется на DBF-файле, то MapInfo сможет внести изменения в такую таблицу; при выполнении оператора **Update**. MapInfo реально произведет изменения в исходном DBF-файле. Но MapInfo не сможет выполнить то же самое с таблицами, базирующимися на файлах электронных таблиц или текстовых (ASCII) файлах. Чтобы внести изменения в файлы такого типа, следует создать копию таблицы (с помощью оператора **Commit Table... As**) и уже затем ее изменять.

## Создание файла отчета из открытой таблицы MapInfo

Отчеты по табличным данным высокого качества, теперь можно создавать прямо из MapInfo, для этого используется генератор отчетов, соответствующий промышленным стандартам Seagate Crystal Reports. Этот редактор отчетов имеет интуитивно понятный и дружелюбный интерфейс. Смотрите операторы **Create Report From Table** и **Open Report** в *Справочнике MapBasic*.

## Чтение значений из строк и колонок таблицы

Программа на MapBasic может оперировать значениями из отдельных строк (записей) и/или колонок (полей) таблицы. Для этого следует использовать следующие действия:

1. С помощью оператора **Fetch** указать, с какой строкой (записью) таблицы Вы будете работать. Этот оператор устанавливает текущую запись таблицы.
2. С помощью выражений над элементами таблицы (например, *имя\_таблицы.имя\_колонки*) Вы можете получать доступ к отдельным полям в текущей записи.

Рассмотрим пример программы, в которой читается содержимое поля "Country" из первой записи таблицы WORLD:

```
Dim s_name As String
Open Table "world" Interactive
Fetch First From world
s_name = world.Country
```

Каждой открытой таблице соответствует указатель текущей записи (не путать с указателем мыши, отображающим текущее положение мыши на экране). При выполнении оператора **Fetch** Вы передвигаете указатель текущей записи к заданной позиции. Дальнейшие операции производятся с той записью, перед которой находится указатель текущей записи.

Имеется несколько вариантов применения оператора **Fetch** для передвижения указателя текущей записи. Указатель можно передвинуть на одну запись вперед или назад, установить на запись с заданным номером, а также на первую или последнюю запись в таблице. Чтобы определить, не применяется ли оператор **Fetch** за пределами таблицы, используйте функцию **EOT( )**. Подробное описание оператора **Fetch** и функции **EOT( )** можно найти в *Справочнике MapBasic*.

В языке MapBasic используются три различных вида выражений, предоставляющих доступ к значениям полей:

В предыдущих примерах использовался синтаксис первого вида: *имя\_таблицы.имя\_колонки* (*world.country*).

Можно также использовать выражения вида *имя\_таблицы.col#*. Здесь указывается номер поля, а не его название (так, "col1" соответствует первому полю таблицы). Так как "Country" является первым полем таблицы World, то в приведенном нами примере оператор присваивания можно было бы записать по-другому:

```
s_name = world.col1
```

Третий вид выражений использует синтаксис *имя\_таблицы.col(числовое\_выражение)*. Здесь номер поля (колонки) задается числовым выражением, заключенным в круглые скобки. С использованием такого синтаксиса можно было переписать наш пример следующим образом:

```
Dim i As Integer
i = 1
s_name = world.col(i)
```

Синтаксис колонки	Пример:
<i>имя_таблицы.имя_колонки</i>	<code>world.country</code>
<i>имя_таблицы.colN</i>	<code>world.col1</code>
<i>имя_таблицы.col (N)</i>	<code>world.col(i)</code>

Данный подход позволит Вам определять, к какому полю необходим доступ во время выполнения программы.

Бывает, что указывать имя таблицы в выражениях внутри операторов не обязательно. Например, пусть в операторе **Browse** Вам надо задать названия колонок и имя таблицы. Поскольку ясно, с какой таблицей будет работать данный оператор (из предложения **From**), то в названия колонок не обязательно включать имя таблицы.

```
Select Country, Population/1000000 From World
Browse Country, Col2 From Selection
```

Оператор **Select** также содержит предложение **From**, где указано имя таблицы, к которой он применяется. Имена колонок в операторе **Select** могут не иметь префикса *имя\_таблицы*, если в операторе **Select** задана единственная таблица. Если же в предложении **From** оператора **Select** перечислены несколько таблиц, то названия колонок должны начинаться с *имя\_таблицы* в качестве префикса. Более подробная информация об использовании команды **SQL-запрос** содержится в *Руководстве пользователя MapInfo* и в описании оператора **Select** в *Справочнике MapBasic*.

В некоторых случаях Вы должны использовать определенный вид выражений: **colN** или **col(N)**. В последнем примере оператор **Select** работает с двумя колонками; причем вторая колонка является вычисляемой, поскольку значения для этой колонки получаются как результат вычисления выражения `Population/1000000`. В последующем операторе **Browse** на вычисляемую колонку можно ссылаться только как `col2` или как `col (2)`, поскольку `Population/ 1000000` не является допустимым названием колонки.

Обращение к колонке с помощью переменной типа Alias

В приводившихся до сих пор примерах использовались фиксированные имена колонок. Например, в нижеследующем операторе имена колонок – "Country" и "Population" – указаны явно.

```
Select Country, Population/1000000 From World
```

Иногда название колонки заранее не известно, оно определяется во время выполнения. Например, пользователь должен выбрать колонку из списка, а Ваше приложение должно ее обработать.

Язык MapBasic содержит тип переменных **Alias**, который позволяет хранить и формировать названия колонок во время выполнения программы. Как и переменным типа String, переменным **Alias** можно присваивать текстовые строки. MapBasic считает значение переменной типа **Alias** названием колонки, если переменная **Alias** используется в выражении на месте ссылки на колонку.

Например:

```
Dim val_col As Alias
val_col = "Инфляция"
Select * From world Where val_col > 4
```

MapBasic подставит значение `val_col` (псевдоним *Инфляция*) при выполнении оператора **Select**, чтобы выбрать все страны с уровнем инфляции больше 4%.

---

 Максимальная длина для *Alias* 32 символа.

---

Разберем пример процедуры `MapIt`, которая открывает таблицу, показывает ее в окне Карты и выбирает все записи из указанной колонки, имеющие значения большие или равные заданному пороговому значению. `MapIt` использует переменную типа **Alias** для формирования названия колонки во время выполнения программы.

```
Include "mapbasic.def"
Declare Sub Main
Declare Sub MapIt( ByVal filespec As String,
    ByVal col_name As String,
    ByVal min_value As Float )

Sub Main
    Call MapIt("C:\MAPINFOW\MAPS\WORLD.TAB", "population", 15000000)
End Sub
Sub MapIt( ByVal filespec As String,
    ByVal col_name As String,
    ByVal min_value As Float )

    Dim a_name As Alias
    a_name = col_name
    Open Table filespec
    Map From TableInfo(0, TAB_INFO_NAME)
    Select * From TableInfo(0, TAB_INFO_NAME)
    Where a_name >= min_value
End Sub
```

В процедуре `MapIt` оператор **Select** использует переменную типа **Alias** (`a_name`) вместо явного названия колонки. Заметим, что параметр `col_name` имеет тип `String` (а не **Alias**); это связано с тем, что MapBasic не позволяет передавать параметры типа **Alias** значением. Чтобы обойти это ограничение, название колонки передается значением как строковая переменная (типа `String`), а затем значение строковой переменной копируется в локальную переменную типа **Alias** (`a_name`).

Приведенный пример показывает, как переменной типа **Alias** присвоить название колонки в виде строковой переменной ("population"). Переменная типа **Alias** может содержать и сложное название колонки в виде *имя\_таблицы.имя\_колонки*. Вот пример использования подобного синтаксиса:

```
Dim tab_expr As Alias
Open Table "world"
Fetch First From world
```



```
tab_expr = "world.COL1"  
Note tab_expr
```

Оператор **Note** будет работать точно так же, если его переписать следующим образом:

```
Note world.COL1
```

## Обработка составных имен колонок

Форма записи *имя\_таблицы.имя\_колонки* (например, `world.population`) аналогична синтаксису, используемому при обращении к полю переменной типа Type (т.е. нового или “пользовательского”). MapBasic пытается интерпретировать любое выражение вида *имя.имя* сначала как ссылку на поле переменной пользовательского типа. Если же это не удастся, то MapBasic пробует интерпретировать такое выражение как обращение к колонке одной из открытых таблиц. Если и это не удастся, MapBasic выдает сообщение об ошибке при выполнении программы.

## Обращение к записям с помощью поля “RowID”

RowID – это название особой колонки, содержащей номера строк (записей) таблицы. RowID можно считать полем таблицы, хотя на самом деле в файле такое поле не содержится. Поэтому будем называть RowID виртуальной колонкой, то есть колонкой, которой можно пользоваться, но которая невидима и формируется особым образом. Значение RowID для первой строки (записи) таблицы равно 1, для второй – 2 и т.д.

Вот пример выделения первой записи из таблицы World:

```
Select * from world Where RowID = 1
```

В следующем примере RowID используется в операторе **Select** для выделения всех штатов, население в которых в 1990 году превосходило среднее по стране.

```
Dim median_row As Integer  
Select * From states Order By pop_1990 Into bypop  
median_row = Int(TableInfo(bypop, TAB_INFO_NROWS) / 2)  
Select * From bypop Where RowID > median_row
```

Так как функция **TableInfo( )** возвращает общее число записей в виртуальной таблице Вурор, переменная `median_row` содержит номер записи о штате со средним уровнем населения. Последний оператор **Select** отбирает все штаты, идущие после него в упорядоченной таблице Вурор.

Если удалить одну из записей таблицы, эта запись не будет считаться полностью удаленной до тех пор, пока Вы не выполните упаковку таблицы. Удаленные строки показываются в окнах Списков серыми полосками. Удаленным записям по-прежнему соответствуют значения RowID. Поэтому само по себе удаление записи из таблицы не влияет на значения RowID; лишь когда Вы после удаления записи сохраните изменения и упакуете таблицу, значения RowID изменятся. Чтобы упаковать таблицу, выполните команду **Таблица > Изменить > Упаковать** в MapInfo или оператор MapBasic **Pack Table**.

## Использование колонки "Obj" для работы с графическими объектами

В таблицах MapInfo имеется еще одна специальная колонка. Она называется "Obj" и предназначена для работы с графическими объектами. Любая таблица, содержащая графические объекты, содержит колонку Obj (хотя эта колонка и не показывается в окнах Списков). Если некоторой записи не сопоставлен графический объект, то такая запись содержит пустое поле Obj.

Пример выбора всех записей, не содержащих графических объектов:

```
Select * From sites Where Not Obj
```

В некоторых случаях, например, при геокодировании таблицы, когда не все записи геокодированы успешно, бывает удобно выбрать все негеокодированные записи.

Следующий пример показывает, как скопировать графический объект из таблицы в переменную типа Object:

```
Dim o_var As Object  
Fetch First From sites  
o_var = sites.obj
```

Подробнее работа с графическими объектами описана в главе [Географические и графические объекты](#).

## Нахождение адресов в таблице

Пользователи MapInfo могут находить адрес на карте с помощью команды **Запрос > Найти**. В программе на языке MapBasic аналогичный поиск выполняют операторы **Find** и **Find Using**. В операторе **Find Using** указывается таблица, по которой следует проводить поиск; оператор **Find** пробует определить географические координаты для указанного адреса (скажем, "Тверская 23 "). С помощью оператора **Find** можно также находить пересечение двух улиц, указывая в качестве параметра два названия через двойной амперсанд (например, "Тверская && Лесная").

После выполнения оператора **Find** Вы должны дважды вызвать **CommandInfo( )**: чтобы определить, найден ли адрес, и еще раз вызвать **CommandInfo( )**, чтобы узнать географические координаты. В отличие от команды **Запрос > Найти** в MapInfo оператор **Find** в языке MapBasic не передвигает автоматически изображение в окне Карты. Чтобы показать найденный объект в центре окна Карты, Вы можете использовать оператор **Set Map** с предложением **Center**. Аналогично, оператор **Find** не добавляет автоматически символ к Карте. Чтобы пометить найденный адрес, используйте функцию **CreatePoint( )** или оператор **Create Point**. Примеры кода смотрите в описании оператора **Find** в *Справочнике MapBasic* или в *Справочной системе*.

## Геокодирование

Для того, чтобы произвести геокодирование, сделайте следующее:

1. Используйте оператор **Fetch**, чтобы указать адрес в таблице.
2. Используйте оператор **Find Using** и оператор **Find**, чтобы найти адрес.

3. Вызовите функцию **CommandInfo( )** чтобы определить результат выполнения оператора **Find**; вызовите **CommandInfo( )** снова, чтобы определить координаты X и Y найденного места.
4. Создайте точечный объект, вызвав функцию **CreatePoint ( )** или оператор **Create Point**.
5. Используйте оператор **Update**, чтобы присоединить точечный объект к таблице.

Для того, чтобы произвести интерактивное (“ручное”) геокодирование, сделайте следующее:

```
Run Menu Command M_TABLE_GEOCODE
```

## SQL-запросы

Пользователи MapInfo могут выполнять сложные запросы с помощью диалога команды **Запрос > SQL-запрос**. Вся мощь диалога SQL-запрос доступна и в программах на языке Map-Basic и сосредоточена в операторе **Select**. Оператор **Select** можно использовать для фильтрации, сортировки, подсчета и объединения таблиц. Подробное описание оператора **Select** можно найти в *Справочнике MapBasic*.

## Ошибки при работе с таблицами и колонками

MapBasic не может проверить правильность ссылок на таблицы и колонки во время компиляции. Например, если в Вашей программе встречается обращение к колонке "states.pop", компилятор MapBasic не может определить, действительно ли в таблице States имеется колонка "pop". Это значит, что даже если Вами допущена опечатка в названии таблицы, компиляция пройдет успешно. Однако, если название колонки (вроде "states.pop") содержит опечатки, возникнет ошибка при выполнении программы.

Чтобы свести к минимуму возможность возникновения ошибок во время выполнения программы, используйте следующие приемы. Когда это возможно, употребляйте предложение **Interactive** в операторе **Open Table**; Если заданная Вами таблица не будет обнаружена во время выполнения, появится диалог, в котором пользователь сможет указать правильное расположение таблицы. Не следует ориентироваться на стандартные псевдонимы для открываемых Вами таблиц; после выполнения оператора **Open Table** вызывайте функцию `TableInfo(0, TAB_INFO_NAME)`, чтобы уточнить, какой псевдоним сопоставлен открытой таблице. Подробнее об открытии таблиц см. главу **Open Table** в *Справочнике MapBasic*.

## Запись значений в таблицу

Чтобы добавить новую запись в таблицу, следует использовать оператор **Insert**. Оператор **Update** позволяет изменять значения полей записей в таблице. Оба этих оператора описаны в *Справочнике MapBasic*.

Если Вы добавили новые записи в таблицу или внесли изменения в уже существующие записи, Вы должны сохранить изменения с помощью оператора **Commit**. Чтобы отменить внесенные изменения, выполните оператор **RollBack**.

## Создание новых таблиц

Оператор **Create Table** предназначен для создания новых (пустых) таблиц. Чтобы индексировать поля в таблице, применяется оператор **Create Index**, а для присоединения к таблице графических объектов – оператор **Create Map**.

В следующем примере создается таблица, содержащая графические объекты, имена клиентов, их почтовые адреса, размер и дату заказа, а также условный код. Поля "name" и "CustID" индексируются.

```
Create Table CUST
  (Name Char(20),
   Address Char(30),
   City Char(30),
   Amount Decimal(5,2),
   OrderDate Date,
   CustID Integer)
File "C:\customer\Cust.tab"
Create Map For CUST CoordSys Earth

Create Index On CUST (CustID)

Create Index On CUST (Name)
```

Вы также можете создавать таблицы путем сохранения копий существующих таблиц (например, таблицы Selection – результата выборки) с помощью оператора **Commit** или импорта таблиц с помощью оператора **Import**.

## Изменение структуры таблицы

Каждая таблица имеет определенную структуру. То есть список колонок (полей), их типов, список индексированных полей. Пользователи MapInfo могут изменять структуру таблиц командой **Таблица > Изменить > Структура**. В программе на MapBasic для изменения структуры таблицы используются операторы **Alter Table** и **Create Index**.

Как правило, структуру таблицы нельзя изменить, пока не сохранены все внесенные изменения. Если к таблице добавлялись записи, то после этого следует сохранить изменения (командой **Commit**) или отменить их (командой **Rollback**), прежде чем изменять структуру таблицы.

Оператор **Alter Table** изменяет структуру таблицы. В следующем примере изменяется название колонки "Address" на "ShipAddress", увеличивается длина поля "Name" до 25 символов, удаляется колонка "Amount", добавляются две новые колонки: "Zipcode" и "Discount", а также изменяется порядок колонок в таблице.

```
Alter Table CUST (Rename Address ShipAddress,
  Modify Name Char(25),
  Drop Amount
  Add Zipcode Char(10),
  Discount Decimal(4,2)
  Order Name, Address, City, Zipcode,
  OrderDate, CustID, Discount)
```

Не разрешается изменять структуру таблиц, базирующихся на файлах в формате электронных таблиц или текстовых (ASCII) файлов, нельзя изменять структуру таблицы `Selection`.

Оператор **Add Column** добавляет к таблице временную колонку. Оператор **Add Column** позволяет создать динамическую (вычисляемую) колонку, значения полей в которой вычисляются по данным из другой таблицы. **Add Column** также позволяет осуществлять расширенный набор операций над областями (полигонами) с пропорциональным обобщением данных, в зависимости от характера наложения объектов из одной таблицы на объекты другой таблицы. Предположим, например, что имеется таблица границ городов и таблица, отражающая риск затопления территорий. Некоторые города частично или полностью попадают в зоны возможного затопления, другие же полностью лежат вне таких зон. С помощью оператора **Add Column** Вы можете выделить демографическую информацию из таблицы городов, а затем использовать эту информацию при вычислении статистики для зон возможного затопления. Подробно оператор **Add Column** описан в *Справочнике MapBasic*.

## Создание индексов и присоединение к таблицам графических объектов

Индексирование полей позволяет оптимизировать время обработки запросов в MapInfo. Некоторые операции, такие как **Find** и **Geocode** в MapInfo, требуют наличия индексов для полей, по которым проводится поиск. Например, перед тем, как проводить поиск клиента по фамилии в базе данных с помощью команды **Find (Найти)**, Вы должны проиндексировать колонку (поле) фамилий. Команда **Select** выполняется значительно быстрее после проведения индексации. При обработке SQL-запросов создается временный индекс, если поля, перечисленные в предложении **Where**, не проиндексированы. На число колонок, которые можно проиндексировать, не накладывается никаких ограничений. Колонка `Obj` индексируется всегда.

Чтобы создать индекс из MapBasic программы, выполните оператор **Create Index**. Оператор **Drop Index** удаляет индекс. MapBasic не может использовать индексы, созданные в других программах; MapBasic также не индексирует вычисляемые колонки. Наличие индекса для того или иного поля не влияет на порядок записей в окне Списка. Оператор **Select** с предложением **Order By** позволяет упорядочить записи по некоторому полю и показать результат в окне Списка.

## Информация о структуре таблицы

Функции **TableInfo( )**, **ColumnInfo( )** и **NumTables( )** позволяют получать информацию об открытых на данный момент таблицах.

- **TableInfo( )** возвращает число записей в таблице, число колонок, данные о наличии графических объектов и т.д.
- **ColumnInfo( )** выдает сведения о каждой из колонок таблицы, в частности, название колонки, тип данных, проиндексирована ли колонка или нет.
- **NumTables( )** возвращает число открытых таблиц (включая временные таблицы, такие как `Query1`).

В следующем примере выясняется, какие таблицы открыты и их имена собираются в массив.

```
Include "mapbasic.def"
Dim i, table_count As Integer
Dim tablenames() As String

' определить число открытых таблиц
table_count = NumTables()

' Изменить размер массива так, чтобы он мог,
' хранить имена всех таблиц.
ReDim tablenames(table_count)

' Цикл по таблицам
For i = 1 To table_count

    ' Считать имя таблицы # i
    tablenames(i) = TableInfo(i, TAB_INFO_NAME)

    ' показать имя таблицы в окне Сообщений
    Print tablenames(i)

Next
```

## Работа с таблицей Selection


"Selection" – это специальное имя таблицы, содержащей набор выбранных в данный момент записей. Программа на языке MapBasic (как и конечный пользователь) может работать с таблицей Selection так же, как и с любой другой таблицей.

Например, Вы можете просматривать набор выбранных в данный момент записей с помощью следующего оператора:

```
Browse * From Selection
```

При подобном доступе к таблице Selection MapInfo делает копию текущего состояния таблицы-выборки и дает ей имя "QueryN", где N – это целое число, большее или равное единице. Подобно Selection, QueryN является временной таблицей. Функция **SelectionInfo( )** позволяет определить, какое стандартное имя-псевдоним присвоит MapInfo таблице Selection (скажем, Query1 или Query2). **SelectionInfo( )** также позволяет получить другую информацию о таблице Selection (число выбранных записей и т.п.).

---

 Слово "Selection" и некоторые другие не были переведены в русской версии MapInfo для того, чтобы обеспечить совместимость программ на языке MapBasic (прим. переводчика).

---

## Заккрытие таблиц запроса "QueryN"

Работая в MapInfo, Вы можете заметить, что Вы открыли ряд таблиц с именами "QueryN" (Query1, Query2, и т.д.). Например, если Вы выбираете объекты Карты и затем просматриваете Список выбранных объектов, заголовок окна может выглядеть как "Query1 Список". Каждый новый QueryN – это "снимок" последней выборки.

Программы MapBasic могут также могут открывать таблицы QueryN. Например, создание ссылки на колонку типа Selection.Obj заставляет MapInfo открывать таблицу QueryN. Если Вы хотите, чтобы ваша программа MapBasic закрыла таблицу QueryN, сделайте следующее:

- Когда Вы используете оператор **Select**, включайте необязательное предложение **Into**. Затем, вместо того, чтобы обращаться к имени таблицы "Selection" используйте имя таблицы, которое Вы указали в предложении **Into**. Если Вы используете предложение **Into**, MapInfo не будет открывать таблицы QueryN, когда Вы обращаетесь к результатам запроса. Когда работа выполнена, закройте таблицу, используя оператор **Close Table**.
- Если пользователь делает выбор (например, объекта на Карте), и затем Ваша программа работает с выбранным элементом, MapInfo откроет таблицу QueryN. Следующий пример показывает, как закрыть таблицу QueryN.

```
' Запомните, сколько таблиц уже открыто
i_open = NumTables()

' Получаем доступ к таблице Selection. Например:
Fetch First From Selection
obj_copy = Selection.obj

' Закроем таблицу QueryN, если она открыта.
If NumTables() > i_open Then
    Close Table TableInfo(0, TAB_INFO_NAME)
End If
```

## Изменение таблицы Selection

Оператор **Select** позволяет выбирать одну или несколько записей. Оператор **Select** представляет собой мощный оператор с большим числом возможностей. Вы можете применять оператор **Select** для фильтрации, сортировки, анализа данных, создания реляционных связей между таблицами. Все возможности команды **Запрос > SQL-запрос** в MapInfo доступны в программах на MapBasic благодаря оператору **Select**.

Если Вы хотите, чтобы в результате выполнения оператора **Select** была создана таблица с нестандартным именем (отличающимся от QueryN), Вы можете задать свое имя результирующей таблице. Оператор **Select** может содержать предложение **Into**, в котором указывается имя результирующей таблицы. Например, чтобы поместить выбранные записи в таблицу по имени "Active":

```
Select * From sites
Where growth > 15
Into Active
```

Введение в SQL-запросы можно найти в *Руководстве пользователя* MapInfo. Подробно оператор **Select** описан в *Справочнике MapBasic*.

## Внесение изменений в выбранные записи

Оператор **Update** можно использовать для внесения изменений в таблицу Selection. При внесении изменений в таблицу Selection, они вносятся и в базовые таблицы, на которых основана выборка.

Например, следующий оператор **Select** выбирает некоторые записи из таблицы "employees". После оператора **Select** выполняется оператор **Update**, с помощью которого в выбранные записи вносятся изменения.

```
Select * from employees
  Where department = "marketing" and salary < 20000

Update Selection
  Set salary = salary * 1.15
```

Оператор **Update** изменит значения в таблице "employees", поскольку выборка сделана из записей таблицы "employees".

## Ввод данных пользователем с помощью таблицы Selection

Таблица Selection может использоваться при обмене данных с пользователем. Некоторые приложения устроены таким образом, что пользователь должен выбрать одну или несколько записей, а затем выполнить команду. Когда пользователь делает свой выбор, он задает объект; а когда он выполняет команду, он задает действие, которое следует совершить над заданным объектом.

Программа TEXTBOX (из стандартного набора примеров) основана на такой модели "существительное/глагол". Пользователь выбирает один или несколько текстовых объектов, затем выполняет команду **Программы > Рамка > Создать рамки**. Приложение TEXTBOX обращается к таблице Selection и рисует рамки вокруг выбранных пользователем текстовых объектов.

Для доступа к выборке используется функция **SelectionInfo( )**. С помощью **SelectionInfo( )** Вы можете определить, сколько записей выбрано (если выборка непустая). Для непустой выборки с помощью **SelectionInfo( )** можно определить имя таблицы, откуда выбраны записи. Затем можно получить подробную информацию об этой таблице, вызвав **TableInfo( )**.

Если Ваше приложение содержит процедуру со специальным именем **SelChangedHandler**, MapInfo будет активизировать эту процедуру каждый раз, когда происходят изменения в выборке. Например, Вам может потребоваться контролировать доступность элементов меню, созданного Вашим приложением, в зависимости от того, пуста ли выборка или нет. Чтобы осуществлять контроль подобного рода, создайте процедуру **SelChangedHandler**. В теле этой процедуры вставьте обращение к **SelectionInfo(SEL\_INFO\_NROWS)**, чтобы узнать число выбранных записей. В зависимости от этого числа можно с помощью оператора **Alter Menu Item** делать доступными или недоступными соответствующие элементы меню. Подробнее работа с меню описана в главе **Создание интерфейса пользователя**.

## Доступ к Косметическому слою

Каждое окно Карты содержит особый слой, называемый Косметическим, на котором могут размещаться вспомогательные объекты. При выполнении операции поиска, MapInfo Professional пометит найденное специальным символом. Эти символы хранятся на Косметическом слое. Глава **Географические и графические объекты** подробно описывает подписи и их создание.



Для того чтобы управлять Косметическим слоем с помощью MapBasic, применяйте операторы для работы с таблицами (например, **Select**, **Insert**, **Update** или **Delete**) к таблицам с именами вида *CosmeticN* (где *N* – число типа Integer, больше или равное 1). Например, таблица *Cosmetic1* соответствует Косметическому слою первого по счету окна Карты на экране. Следующий оператор выбирает все объекты на Косметическом слое такого окна Карты:

```
Select * From Cosmetic1
```

Чтобы определить название таблицы, соответствующей Косметическому слою заданного окна, следует использовать функцию **WindowInfo( )** с параметром **WIN\_INFO\_TABLE**. Например, следующий оператор удаляет все объекты с Косметического слоя активного окна Карты:

```
Delete From WindowInfo(FrontWindow(), WIN_INFO_TABLE)
```

## Доступ к окнам Отчетов

Операторы работы с объектами языка MapBasic применимы к объектам в окнах Отчетов. Чтобы работать в окне Отчета, Вам следует указывать в операторах в качестве имени таблицы "*LayoutN*" (где *N* – целое число, большее или равное 1).

Например, таблица с именем "*Layout1*" соответствует первому по счету окну Отчета, которое Вы открыли. Следующий оператор выбирает все объекты в окне Отчета:

```
Select * From Layout1
```

Чтобы определить название таблицы, соответствующей заданному окну Отчета, следует использовать функцию **WindowInfo( )** с параметром **WIN\_INFO\_TABLE**.



Заметим, что объекты в окне Отчета используют особую систему координат, основанную на "бумажных" единицах измерения (единицах измерения бумажного носителя, отсчитываемых от верхнего левого угла листа). Любая программа на MapBasic, работающая с координатами объектов в окне Отчета, должна содержать оператор **Set CoordSys**, в котором устанавливается система координат Отчета.

---

Например, программа TEXTBOX (из набора примеров) рисует рамки (прямоугольные объекты) вокруг всех выбранных в данный момент текстовых объектов, независимо от того, лежат ли выбранные объекты в окне Карты или в окне Отчета. Если выбранные объекты лежат в окне Отчета, TEXTBOX выполняет оператор **Set CoordSys Layout**.

При работе с MapInfo окно Статистики дает наиболее простую возможность определения имени таблицы, соответствующей некоторому окну Отчета или Косметическому слою некоторого окна Карты. Если выбрать объект с Косметического слоя карты и затем открыть окно Статистики (например, командой **Настройка > Показать окно Статистики**), окно Статистики покажет сообщение вида "Таблица *Cosmetic1* содержит 1 выбранную запись." Аналогично, если выбрать объект в окне Отчета, то окно Статистики будет содержать сообщение вида "Таблица *Layout1* содержит 1 выбранную запись."

## Редактирование в многопользовательской среде

При работе в многопользовательской среде могут возникать конфликты из-за попыток одновременного доступа к одному и тому же файлу со стороны разных пользователей. MapInfo позволяет редактировать таблицу только одному пользователю.

В этом разделе приведены правила, позволяющие работать в такой ситуации. Изучите их, если Вы хотите создавать MapBasic-программы для работы в сети.

### Правила редактирования таблиц в многопользовательской среде

Многопользовательское редактирование таблиц в среде MapInfo Professional имеет три ограничения:

#### Правило 1

В каждый момент времени таблица может быть доступна для редактирования только одному пользователю.

Представьте себе, что имеется два пользователя, А и Б. Оба пытаются работать с одной и той же таблицей, доступной в сети.

Пользователь А начал редактировать таблицу. (Например, добавил в нее новую строку). Чуть позже пользователь Б попытался начать вносить изменения в эту таблицу. MapInfo не позволит сделать это и выведет сообщение “Нельзя редактировать. Кто-то другой сейчас редактирует таблицу”. Если попытка редактирования будет предпринята из программы MapBasic, произойдет ошибка времени выполнения.

Пока таблица редактируется пользователем А, MapInfo не позволит пользователю Б изменять ее. Так будет до тех пор, пока пользователь А не произведет операции сохранения или восстановления таблицы или не закроет ее.



Пользователь Б сможет читать таблицу, открытую А, а также отображать ее на Карте. Однако он не увидит изменений, произведенных в ней пользователем А до того, как таблица не будет последним сохранена.

---

#### Правило 2

Пользователь не может читать содержимое таблицы во время ее сохранения.

Окончив редактирование, пользователь А выполнил команду меню **Файл > Сохранить таблицу**. Во время сохранения таблицы пользователь Б попытался прочитать данные из нее. MapInfo не позволит сделать этого и выведет сообщение “Нет доступа на чтение к файлу <имя таблицы>.DAT”. Окно диалога будет содержать кнопки **Отмена** и **Повторить** со следующим назначением:

### **Повторить**

Если выбрана кнопка **Повторить**, MapInfo повторит попытку доступа к таблице. Доступа к таблице не будет, если операция сохранения ещё не завершилась. Кнопку **Повторить** можно нажимать до тех пор, пока таблица не станет доступной. После того как операция сохранения завершена, нажатие на кнопку **Повторить** приведёт к открытию таблицы.

### **Отмена**

Если пользователь Б нажмёт кнопку **Отмена**, операция будет отменена и диалоговое окно "Повторить / Отмена" исчезнет с экрана.



Если ошибка совместного доступа произошла в момент когда пользователь Б загружал Рабочий набор, нажатие кнопки **Отмена** может привести к прерыванию загрузки этого набора. Например, Рабочий Набор содержит оператор **Open Table**. Если оператор **Open Table** приведет к ошибке из-за конфликта доступа к таблице в сети и пользователь выберет кнопку **Отмена** в появившемся окне диалога, MapInfo не откроет таблицу. Последующие операторы могут оказаться ошибочными, так как таблица не открыта

---

## **Правило 3**

Если таблица открыта на чтение одним из пользователей, другой не может сохранять ее.

Если другие пользователи читают таблицу в момент, когда пользователь А выбрал команду меню **Файл > Сохранить таблицу**, процесс сохранения не будет запущен. MapInfo отобразит сообщение "Нельзя открыть файл <имя таблицы>.DAT на запись". Окно диалога будет содержать кнопки **Отмена** и **Повторить** со следующим назначением:

### **Повторить**

Если пользователь А нажмёт кнопку **Повторить**, MapInfo Professional повторит попытку сохранения таблицы. Кнопку **Повторить** можно нажимать до тех пор, пока таблица не станет доступной. Нажатие кнопки **Повторить** будет успешно только в том случае, если другие пользователи завершили чтение таблицы.

### **Отмена**

Если пользователь А нажмёт кнопку **Отмена**, операция будет отменена и диалоговое окно "Повторить / Отмена" исчезнет с экрана. В этом случае, таблица не будет сохранена и все изменения не будут внесены в неё до тех пор пока, пользователь не выполнит команду **Файл > Сохранить таблицу**.

## **Как избежать конфликтов при многопользовательском чтении**

Как указано в предыдущем разделе, некоторые конфликты совместного доступа к таблице приводят к появлению окна диалога с кнопками **Повторить** и **Отмена**. Обычно диалоговое окно появляется в момент возникновения конфликта. Однако программа MapBasic может подавлять появление окна, используя оператор **Set File Timeout**.

В той части Вашей программы, где открывается на чтение совместно используемая таблица, используйте оператор **Set File Timeout** со значением больше нуля. Например, если процедура открывает несколько файлов, поместите этот оператор в ее начале:


```
Set File Timeout 100
```

Оператор **Set File Timeout** устанавливает ограничение времени; в этом примере ограничение времени – 100 секунд. Другими словами, MapInfo автоматически повторит попытки выполнить любые операции таблицы, которые производят к конфликту при совместном использовании в течение 100 секунд. Обратите внимание, что MapInfo повторяет операции таблицы вместо того, чтобы отобразить диалог "Повторить / Отмена". Если конфликт все еще происходит после истечения 100 секунд, автоматическое повторение останавливается, и MapInfo отображает диалоговое окно "Повторить / Отмена".

## Как избежать конфликтов при многопользовательской записи

Некоторые операторы MapBasic изменяют содержание таблицы. Например, оператор **Insert** добавляет новые строки к таблице. Если Ваша программа пытается изменять содержание таблицы, и возникает конфликт совместного использования, то произойдет ошибка времени выполнения. Чтобы перехватить эту ошибку, используйте оператор **OnError**. Например, если Ваша процедура вставляет новые строки в таблицу (как в примере ниже), Вы должны создать процедуру обработки ошибок и разместить оператор **OnError** перед опасным участком (Обработка ошибок обсуждена более подробно в главе [Поиск ошибок и отладка программ.](#))

---

 Избегайте использовать операторы **Set File Timeout** и **OnError** вместе в одних и тех же программных сегментах. Там, где обработка ошибок разрешена, значение задержки должно быть равно нулю. В местах, где значение задержки не равно нулю, обработка ошибок должна быть заблокирована. Следующий пример демонстрирует это:

---

```
Function MakeNewRow (ByVal new_name As String) As Logical

' отключение автоматического повторения попыток
  Set File Timeout 0

' отключение перерисовки окна
  Set Event Processing Off

' разрешение обработки ошибок
  OnError Goto trap_the_error

' добавление новой строки и немедленное сохранение
  Insert Into Sitelist ("Name") Values ( new_name )
  Commit Table Sitelist

' установка признака успешного завершения
  MakeNewRow = TRUE

exit_ramp:
```

```
Set Event Processing On
Exit Function
trap_the_error:
' Переход сюда, если операторы Insert или Commit
' привели к ошибке времени исполнения (что случится,
' если другой пользователь уже редактирует таблицу).

If Ask("Ошибка доступа; попробовать еще раз?", "Да", "Нет") Then
' ... попробовать еще раз.
Resume 0
Else
' больше не пробовать.
' Если оператор Insert был успешным и возникла ошибка во время
' выполнения оператора Commit, возвратимся к исходному варианту.
Rollback Table Sitelist

' установим признак ошибки:
MakeNewRow = FALSE
Resume exit_ramp
End If
End Function
```

Отметим следующие моменты:

- Когда Вы изменяете "общедоступную" таблицу, старайтесь уменьшить количество времени, в течение которого таблица содержит несохраненные изменения. В примере, приведенном выше, оператор **Commit** следует немедленно после оператора **Insert** и указанное время очень мало.
- Пример использует оператор **Set Event Processing Off**, чтобы приостановить обработку событий; в результате MapInfo не будет перерисовывать окна во время редактирования. Если бы мы этого не сделали, оператор **Insert** мог бы привести к перерисовке одного или нескольких окон, а это могло бы, очевидно, вызывать конфликт совместного использования данных (например, потому что другие таблицы в том же самом окне Карты могут стать причиной конфликта).
- Оператор устанавливает задержку равную нулю. От процедуры, которая вызывает его, может потребоваться вернуть задержку к предыдущему значению.

## Открытие таблицы для записи

Когда Вы открываете таблицу в многопользовательской среде, есть вероятность, что MapInfo откроет таблицу с доступом только для чтения, даже если файлы, составляющие таблицу – не являются таковыми. Если программа MapBasic выдает оператор **Open Table** точно в момент, когда к таблице обращается другой пользователь, MapInfo может открыть таблицу только для чтения. Это не позволит внести в нее изменения.

Следующий пример показывает, как избежать этого. Вместо выдачи оператора **Open Table**, поместите его внутри цикла, который выполняет итерации, пока файл не будет открыт на чтение или запись.

```
Retry_point:

Open Table "G:\MapInfo\World"
If TableInfo("World", TAB_INFO_READONLY) Then
```

```
Close Table World  
Goto Retry_point  
End If
```

## Файлы-компоненты таблицы

Таблица состоит из нескольких файлов: один из них содержит информацию о структуре таблицы (названия колонок и т.п.); второй – значения полей в записях; третий – графические объекты таблицы (если таковые существуют); прочие – индексы. Значения записей (то есть собственно числовые данные) могут храниться в любом формате, который доступен в MapInfo: DBF, Lotus WKS или WK1, текстовый (ASCII с разделителями) или XLS и XLSX-форматы электронной таблицы Excel.

- *имя\_файла*.TAB: содержит описание структуры таблицы
- *имя\_файла*.DAT, или *имя\_файла*.DBF, или *имя\_файла*.WKS и т.п.: содержит числовые данные
- *имя\_файла*.MAP: содержит графические объекты
- *имя\_файла*.ID: содержит географические индексы
- *имя\_файла*.IND: содержит индексы для различных колонок таблицы.

Поскольку каждая таблица состоит из нескольких файлов, Вы должны быть осторожными при переименовании таблиц. Чтобы переименовать таблицу, выполните команду **Таблица > Изменить > Переименовать** в MapInfo или оператор MapBasic **Rename Table**.

## Таблицы, содержащие растровые изображения

Таблицы, содержащие растровые изображения (без векторных графических данных), имеют лишь некоторые из компонент, перечисленных нами в предыдущем разделе. Это связано с тем, что таблице с растровым изображением не соответствуют никакие числовые данные. Каждая растровая таблица состоит из двух или более файлов: TAB-файла (в нем хранится информация о контрольных точках изображения) и одного или нескольких файлов, содержащих собственно растровое изображение. Например, если растровая таблица основана на файле PHOTO.TIF, то такая таблица может состоять из двух файлов: PHOTO.TIF и PHOTO.TAB.

Во многих отношениях растровые таблицы сходны с обычными таблицами в MapInfo. Чтобы открыть растровую таблицу, надо выполнить оператор **Open Table**. Чтобы показать растровую таблицу в окне Карты, следует выполнить оператор **Map**. Чтобы добавить растровую таблицу в окно Карты, надо применить **Add Map Layer**. С другой стороны, к растровым таблицам неприменимы операции типа **Select**. Чтобы узнать, является ли таблица растровой, следует обратиться к функции **TableInfo( )** с параметром **TAB\_INFO\_TYPE**. Если таблица является растровой, **TableInfo( )** возвратит код **TAB\_TYPE\_IMAGE**. Как правило, MapInfo не вносит никаких изменений в исходный файл с растровым изображением, на котором основана та или иная растровая таблица. Поэтому:

- При использовании оператора **Drop Table** для удаления растровой таблицы, MapInfo удаляет TAB-файл, но не удаляет файл с исходным растровым изображением, на который этот файл ссылается.

- При использовании оператора **Rename Table** для растровой таблицы, MapInfo Professional переименует TAB-файл, но не файл с исходным растровым изображением, на который этот файл ссылается.
- При использовании оператора **Commit** для создания копии растровой таблицы MapInfo Professional скопирует TAB-файл, но не файл с исходным растровым изображением, на который этот файл ссылается.

TAB-файл для растровых таблиц создается в тот момент, когда пользователь заполняет поля диалога **Регистрация изображения**. Чтобы создать такой TAB-файл в программе MapBasic, Вам следует использовать стандартные операции файлового ввода/вывода: создать файл оператором **Open File** и поместить в него текст с помощью оператора **Print #**; см. пример ниже в этой главе.

В следующем примере программы показано, как создать TAB-файл для растровой таблицы. В данной программе контрольным точкам присвоены произвольные координаты (а не реальные географические координаты, которые следует задавать). Поэтому полученную таблицу нельзя использовать для совмещения растрового изображения с векторными картами. Однако при использовании изображения, не связанного с географическими координатами (скажем, Вашего фирменного знака), проблем не возникнет.

```
Include "mapbasic.def"
Declare Sub Main
Declare Function register_nonmap_image(ByVal filename As String,
    ByVal tablename As String) As Logical

Sub Main
    Dim fname, tname As String
    fname = "c:\data\raster\photo.gif" ' файл с растровым изображением
    tname = PathToDirectory$(fname)
        + PathToTableName$(fname) + ".tab" ' имя создаваемой таблицы
    If FileExists(tname) Then
        Note "Растровое изображение уже было зарегистрировано; остановка."
    Else
        If register_nonmap_image(fname, tname) Then
            Note "Создана таблица для изображения из файла: "
                + fname + "."
        Else
            Note "Таблица не может быть создана."
        End If
    End If
End Sub

Function register_nonmap_image( ByVal filename As String,
    ByVal tablename As String) As Logical
    register_nonmap_image = FALSE
    OnError GoTo handler
    Open File tablename For Output As #1 FileType "MIta"
    Print #1, "!Table"
    Print #1, "!Version 300"
    Print #1, "!charset Neutral"
    Print #1
    Print #1, "Definition Table"
    Print #1, " File "" + filename + """
```

```
Print #1, " Type ""RASTER"" "
Print #1, " (1,1) (1,1) Label ""Pt 1"", "
Print #1, " (5,1) (5,1) Label ""Pt 2"", "
Print #1, " (5,5) (5,5) Label ""Pt 3"" "
Print #1, " CoordSys NonEarth Units ""mm"" "
Print #1, " Units ""mm"" "
Print #1, " RasterStyle 1 45" ' Яркость; 50 по умолчанию
Print #1, " RasterStyle 2 60" ' Контраст; 50 по умолчанию
Close File #1
register_nonmap_image = TRUE ' зададим возвращаемое функцией значение
last_exit:
Exit Function
handler:
Close File #1
Resume last_exit
End Function
```

## Работа с метаданными

### Что такое метаданные?

Метаданные – это данные, хранящиеся в файле TAB не в виде строк или колонок. Например, если Вы хотите записать сведения типа “кто и когда работал с данной таблицей”, Вам следует запомнить их в виде метаданных.

Метаданные не отображаются в стандартном интерфейсе пользователя MapInfo. Пользователи не могут видеть метаданных таблицы (если они не просматривают TAB-файл в текстовом редакторе или не запускают программу-пример TABLEMGR). Однако, программа MapBasic может читать и записывать метаданные.

Каждая таблица может содержать (или не содержать) один или несколько ключей метаданных. Каждый ключ соответствует своему разделу, например, “имя автора”, “объявления об авторском праве”, и т.д. Например, ключ, именованный “\Copyright”, мог бы иметь значение “Copyright 1995 Acme Corp.”

### Как выглядят ключи метаданных?

Каждый ключ метаданных имеет имя, которое всегда начинается с символа “\” (наклонной черты влево). Имя ключа никогда не заканчивается этим символом. В имени ключа не различаются прописные и строчные буквы.

Значение ключа – это строка длиной не более 239 символов.

В таблице приведены примеры имен и значений ключей.

Ключ	Значение
"\Copyright Notice"	Copyright 2008 Pitney Bowes Mapinfo Corp."
"Info"	"Участки под застройку"



Ключ	Значение
"Info Author"	"Владимир Журавлев"
"Info\Date\Start"	"14.12.01"
"Info\Date\End"	"31.12.01"
"IsReadOnly"	"FALSE"

Отметим следующие моменты:

- Пробелы внутри имен и значений ключей разрешены.
- Вы можете определить иерархию ключей, используя имена, которые имеют два или больше символов (\). В приведенной выше таблице несколько ключей принадлежат иерархии, которая начинается с ключа "\Info". Устройство ключей в иерархиях разрешают работать со всей иерархией одновременно (например, Вы можете удалять всю иерархию одним оператором).
- "IsReadOnly" – специальный ключ, зарезервированный для внутреннего использования MapInfo Professional. Когда Вы добавляете метаданные к таблице, MapInfo автоматически создает ключ IsReadOnly. Не пытайтесь изменять его.
- В таблице выше каждая строка записана в кавычках, чтобы подчеркнуть, что она имеет строковое значение. Однако, когда Вы извлекаете ключи из таблицы, MapBasic не заключает их в кавычки.

## Примеры работы с метаданными

Функция **GetМетаданные\$( )** позволяет сделать запрос о метаданных таблицы, но только если Вы уже знаете точное имя ключа метаданных. Если Вы знаете, что таблица имеет ключ с именем "\Copyright", то следующее обращение к функции возвратит значение ключа:

```
s_variable = GetМетаданные$(table_name, "\Copyright")
```

Оператор **Метаданные** позволяет создавать, изменять или прочитывать метаданные таблицы, даже если Вы не знаете имена ключей. Следующие примеры показывают различные действия, которые Вы можете выполнять, используя оператор **Метаданные**.



Обратите внимание: в следующих примерах "table\_name" представляет строковую переменную, которая содержит имя открытой таблицы.

Следующий пример сохраняет значение ключа в таблице. Если ключ уже существует, это действие изменяет его значение; если не существует, то происходит добавление ключа к метаданным таблицы.

```
Метаданные Table table_name
  SetKey "\Info\Author" To "Алексей Колотов"
```

В следующем примере ключ "\Info\Author" удаляется.

```
Метаданные Table table_name
  Dropkey "\Info\Author"
```

Здесь удаляется целая иерархия ключей. Все ключи, имена которых начинаются с "Info", будут удалены.

```
Метаданные Table table_name  
    Dropkey "\Info" Hierarchical
```

Когда Вы используете оператор **Метаданные**, чтобы записывать или удалять метаданные, изменения происходят немедленно. Вы не должны выполнять операцию **Save**.

Вы также можете использовать оператор **Метаданные**, чтобы читать метаданные из таблицы, даже если Вы не знаете имена ключей. Для этого нужно:

1. Использовать оператор **Метаданные Table... SetTraverse** для инициализации просмотра имен ключей.
2. Использовать оператор **Метаданные Traverse... Next** чтобы получить значение следующего ключа. Оператор возвращает имя ключа в строковой переменной.
3. Продолжить использование оператора **Метаданные Traverse... Next** для получения значений других ключей. Обычно это следует повторять в цикле. Если ключа нет, оператор **Метаданные Traverse... Next** вернет пустое значение в качестве его имени.
4. Завершить поиск оператором **Метаданные Traverse... Destroy**. Память, использованная для просмотра метаданных, освобождается.

В следующем примере показано, как просматривать метаданные таблицы.

```
Sub Print_Метаданные(ByVal table_name As String)  
  
    Dim i_traversal As Integer  
    Dim s_keyname, s_keyvalue As String  
  
    ' Инициализировать путь обхода. ' Назначить "\"  
    ' начальным ключом, чтобы обход начинался  
    ' с первого ключа.  
    Метаданные Table table_name  
        SetTraverse "\" Hierarchical Into ID i_traversal  
    ' Попытаться получить первый ключ:  
    Метаданные Traverse i_traversal  
        Next Into Key s_keyname Into Value s_keyvalue  
  
    ' Выполнять цикл, пока не закончатся ключи;  
    ' на каждой итерации получать  
    ' один ключ и выводить его в окно Сообщений  
    Do While s_keyname <> ""  
        Print " "  
        Print "Key name: " & s_keyname  
        Print "Key value: " & s_keyvalue  
  
        Метаданные 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

Полный синтаксис оператора **Метаданные** приведен в *Справочнике MapBasic* и в *Справочной системе*.

## Работа со сшитыми таблицами

### Что такое сшитая (seamless) таблица?

Сшитые таблицы позволяют группировать несколько таблиц вместе и обрабатывать их как одну таблицу. Если Вы сгруппировали Ваши таблицы в сшитую, Вы сможете добавлять всю группу к окну Карты в диалоге **Управление Слоями**. Дополнительная информация о работе со сшитыми таблицами в *Руководстве пользователя MapInfo*.

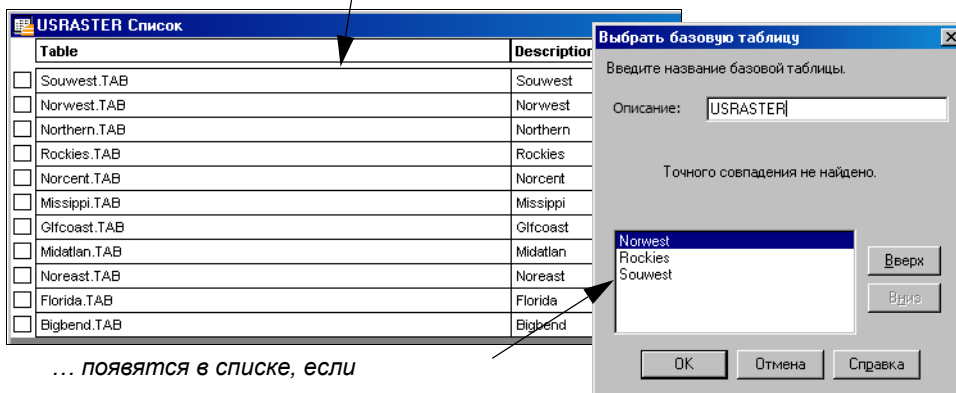
### Как работать со сшитыми таблицами?

Комплект поставки MapInfo содержит программу MapBasic “Сшитые таблицы” (SEAM-MGR.MBX), которая позволяет создавать такие таблицы и манипулировать ими. Чтобы увидеть, из чего составлена сшитая таблица, Вы должны действовать следующим образом:

1. Открыть сшитую таблицу, например USRASTER.TAB.
2. Запустить приложение “Сшитые таблицы”.
3. Выполнить команду **Программы > Сшитые таблицы > Отключить “сшитость”**, чтобы отключить соответствующий атрибут для таблицы DCMetroA.
4. Выберите **Окно > Новый Список**, чтобы открыть окно Списка.

Как и обычная таблица, сшитая содержит строки и колонки. Каждая строка соответствует включенной в нее таблице.

Описания таблиц из второй колонки...



Первый столбец в таблице содержит имена таблиц. Второй содержит описания, которые появляются в интерфейсе пользователя. Имена таблиц в первом столбце могут включать дисковый маршрут к каталогу. Вы можете опускать дисковый маршрут, если таблицы находятся в том же самом каталоге, что и сшитая таблица, или если каталоги с этими таблицами доступны для поиска.

Каждая строка в сшитой таблице имеет присоединенный графический объект, точно также как объекты присоединены к строкам в стандартных таблицах. Однако объекты в сшитой таблице не предназначены для отображения. Каждая строка в сшитой таблице содержит объект-прямоугольник, который определяет для соответствующей таблицы ее минимальный описывающий прямоугольник (МОП). Когда пользователь отображает сшитую таблицу в окне Карты, MapInfo сравнивает текущие размеры окна Карты с МОП. MapInfo открывает только те основные таблицы, для которых МОП виден хотя бы частично в окне Карты.

## Синтаксис MapBasic для сшитых таблиц

Используйте оператор **Set Table** для того, чтобы превратить сшитую таблицу в обычную. Например, если Вы хотите отредактировать описания в сшитой таблице, используйте оператор

```
Set Table USRaster Seamless Off
```

и затем откройте окно Списка для таблицы.

Вызывайте функцию `TableInfo(... , TAB_INFO_SEAMLESS)`, если нужно узнать, является ли данная таблица сшитой.

Вызывайте функцию **GetSeamlessSheet()**, если нужно вывести на экран диалог с приглашением выбрать таблицу из группы сшитых.

## Ограничения при работе со сшитыми таблицами

Все основные таблицы в сшитой таблице должны иметь ту же самую структуру (то есть, одинаковое число столбцов, одинаковые имена столбцов и т.д.).

Обратите внимание на то, что некоторые операции MapInfo не могут использоваться со сшитыми таблицами. Например:

- Вы не можете одновременно выбирать объекты более чем из одной базовой таблицы в сшитой таблице.
- Оператор MapBasic **Find** не может искать во всей сшитой таблице; оператор **Find** может работать только с одной основной таблицей в каждый момент времени.
- Вы не можете делать сшитую таблицу доступной для редактирования в окне Карты.
- Вы не можете создавать тематическую Карту для сшитой таблицы.

## Доступ к удаленным базам данных

В предыдущих обсуждениях мы описывали, как работать с “локальными” таблицами MapInfo (таблицами на Вашем жестком диске или, возможно, на сетевом файл-сервере). Этот раздел содержит сведения о том, как MapBasic может обращаться к удаленным базам данных типа Oracle или SQL Server.

Названия операторов и функций MapBasic, относящихся к этой теме, начинаются с ключевого слова **Server**, за исключением оператора **Unlink**. Подробности относительно синтаксиса можно найти в *Справочнике MapBasic*.

## Как осуществлять доступ к удаленным данным

MapInfo позволяет приложениям MapBasic соединяться со многими базами данных в одно и то же время и осуществлять SQL-запросы к различным базам. Это удастся сделать, используя так называемые дескрипторы соединения и дескрипторы оператора.

Дескриптор (или номер) соединения идентифицирует информацию о конкретном соединении. MapBasic определяет дескрипторы соединения как переменные целого типа. Приложение получает дескриптор соединения после соединения с источником данных. Дескриптор соединения используется, чтобы связать (сопоставить) последующие операторы с конкретным соединением.

Дескриптор (или номер) оператора идентифицируют информацию об SQL-операторе. MapBasic определяет дескрипторы соединения как переменные целого типа. Приложение должно получить дескриптор оператора после вызова функции **Server\_Execute( )**, чтобы передать SQL-запрос. Дескриптор оператора используется, чтобы связать с ним последующие SQL-запросы, например, операции **Fetch** и **Close** с конкретным оператором **Select**.

## Установка и разрыв связи

Прежде, чем приложение MapBasic сможет начать выполнять оператор SQL для доступа к удаленным базам данных, оно должно запросить соединение, используя функцию **Server\_Connect**. Только если соединение успешно установлено, функция возвращает дескриптор соединения (hdbc) для использования в последующих обращениях **SQLDataLink**.

```
Dim hdbc As Integer  
hdbc = Server_Connect("ODBC", "DLG=1")
```

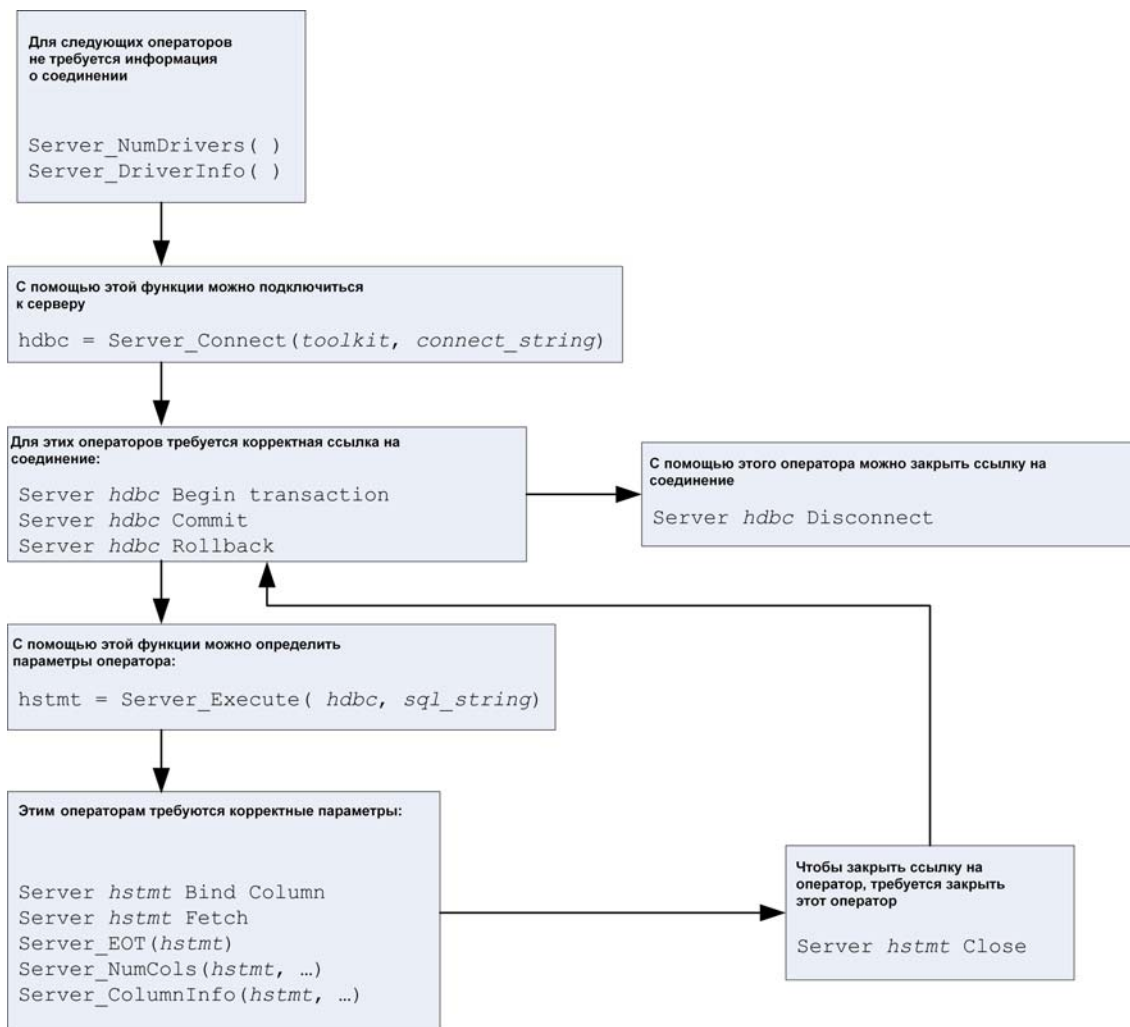
Когда драйвер выполнит фиксацию (commit) или откат (rollback), сбрасываются все операторные запросы, связанные с этим соединением. Диспетчер драйвера производит работу, связанную с переключением соединений, в то время как происходит обработка транзакции для текущего соединения.

Используйте следующий оператор для разъединения:

```
Server hdbc Disconnect
```

Этот оператор закрывает соединение и освобождает все связанные с ним ресурсы.

Следующая схема описывает последовательность, в которой могут быть использованы операторы SQL MapBasic **Server**. Есть некоторые операторы и функции, которые не требуют никакой информации о соединении, например, **Server\_NumDrivers( )**, некоторые требуют только дескриптора соединения, например, **Server Commit**, некоторым же нужен дескриптор оператора, например, **Server Fetch**.



С помощью ODBC (Open Database Connectivity), программного интерфейса доступа к базам данных, используя предложение **Into** оператора MapBasic **Server Fetch**, можно загрузить из источника данных целиком всю таблицу, отдельные строки или столбцы, и представить эту информацию в виде результирующего набора значений в таблице MapInfo. Однако никакие изменения в этой таблице не повлияют на таблицу базы данных на сервере. Обновление удаленных баз данных можно завершить оператором **Save File**.

## Обработка геометрических объектов в PostGIS

В PostGIS геометрические примитивы (дуга, эллипс, прямоугольник и т.п.) не поддерживаются, поэтому при попытке сохранить Карту содержащую геометрические примитивы, будут получены следующие варианты сообщений:

- **Геометрические примитивы всех не поддерживаемых типов** – при попытке сохранить в Oracle Карту, на которой присутствуют объекты со всеми типами не поддерживаемых геометрических примитивов, появится следующее сообщение:

Таблица содержит не поддерживаемые объекты (прямоугольники со скруглёнными углами, эллипсы или дуги) .  
Преобразовать в полигоны и/или полилинии?

Нажмите **Да**, чтобы преобразовать не поддерживаемые объекты в полигоны или полилинии; если требуется отменить преобразование не поддерживаемых объектов – нажмите **Нет**. Если отменить преобразование, то созданная Карта не будет сохранена в базе данных PostGIS.

- **Только полигональные геометрические примитивы** – при попытке сохранить в Oracle Карту, на которой присутствуют только полигональные объекты не поддерживаемых типов геометрических примитивов, появится следующее сообщение:

Таблица содержит не поддерживаемые объекты (прямоугольники со скруглёнными углами или эллипсы) .  
Преобразовать в полигоны?

Нажмите **Да**, чтобы преобразовать не поддерживаемые объекты в полигоны; если требуется отменить преобразование не поддерживаемых объектов, нажмите **Нет**. Если отменить преобразование, то созданная Карта не будет сохранена в базе данных PostGIS.

- **Только линейные геометрические примитивы** – при попытке сохранить в Oracle Карту, на которой присутствуют только линейные объекты не поддерживаемых типов геометрических примитивов, появится следующее сообщение:

Дуга – неподдерживаемый объект. Преобразовать в полилинии?

Нажмите **Да**, чтобы преобразовать не поддерживаемые объекты в полилинии; если требуется отменить преобразование не поддерживаемых объектов, нажмите **Нет**. Если отменить преобразование, то созданная Карта не будет сохранена в базе данных SQL Server Spatial.

- **Для прямоугольных объектов** – если Вы создали Карту с прямоугольными объектами и пытаетесь сохранить ее в PostGIS, появится сообщение:

Не удастся загрузить объект – прямоугольники не поддерживаются для этой таблицы. Операция отменена.

Нажмите **ОК**. Карта с прямоугольниками не будет сохранена в базе данных PostGIS.

## Доступ к удаленным базам при помощи связанных таблиц

Связанная таблица – специальный вид таблицы MapInfo, которая сохраняет связь с удаленной базой данных. Редактирование может производиться даже при наличии многих соединений. Так как обновление производится СУБД отдельно от транзакций, другие пользователи СУБД могут обновлять одни и те же строки одной и той же таблицы. Используется оптимистичный сценарий предотвращения потери данных. Управление параллельной обработкой данных реализовано с использованием предложения **Automatic/Interactive** оператора **Commit Table**. Когда данные сохранены, соединение с удаленной базой

данных переустанавливается, происходит разрешение конфликтов, связанных с внесением изменений в базу данных. Связанная таблица создается оператором MapBasic **Server Link Table**.

Связанные таблицы содержат информацию, позволяющую восстановить соединения и идентифицировать удаленные данные, в которые нужно внести изменения. Эта информация сохранена как метаданные в файле таблицы.

Неотредактированная связанная таблица может быть обновлена текущими данными из удаленной базы данных без повторного указания данных о соединении, запросе и таблице. Данные в связанной таблице обновляются оператором MapBasic **Server Refresh**.

Связанная таблица может быть отсоединена оператором MapBasic **Unlink**. Команда Разорвать связь разрывает связь таблицы MapInfo с таблицей СУБД удаленной базы данных. В результате Вы получите обыкновенную базовую таблицу MapInfo.

При использовании пространственной индексации MapInfo пользователи могут сохранять и отыскивать точки в любой базе данных. См. раздел **Присоединение геоинформации к удаленной таблице**.

## Прямой доступ к удаленным базам данных

Вы можете получить прямой доступ к данным из удаленных баз данных, используя оператор **Register Table**. Когда Вы определяете **Type** как ODBC, оператор **Register Table** сообщает MapInfo, что надо проверить таблицу ODBC и создать соответствующую таблицу (*имя\_файла*.TAB).

## Повышение производительности при работе с таблицами

### Задайте стандартное представление для таблицы на удаленной СУБД

Задайте стандартное представление, чтобы было видна только та часть Карты, которая Вас интересует. Это ускоряет доступ к большим таблицам.

Вид по умолчанию хранится в каталоге MAPINFO\_MAPCATALOG в четырех новых колонках, которые содержат границы (VIEW\_X\_LL, VIEW\_Y\_LL, VIEW\_X\_UR AND VIEW\_Y\_UR). MapCatalog – это таблица, которая содержит метаданные об удаленных таблицах. Если в MapCatalog нет записи о таблице, MapBasic вызовет всю Карту.

Для установки нового вида по умолчанию откройте удаленную таблицу (в полных границах), настройте требуемый вид и запустите утилиту MapBasic **Window Manager**, для вызова функции **Set Default View**. Новые границы будут внесены в MapCatalog и будут использованы при новом открытии таблицы.

Пользователь удаленной базы (определенный при подключении) должен иметь право вносить изменения в MapCatalog. Без наличия этого разрешения появится соответствующее предупреждение, и процесс изменения не состоится.

При использовании команды **Сохранить копию** для загрузки таблицы в удаленную базу данных представление по умолчанию оригинальной таблицы будет также использовано и для новой созданной таблицы.



## Минимизация количества транзакций

Обычно, когда пользователь редактирует таблицу, MapInfo сохраняет изменения во временном файле, называемом файл транзакции. Постепенно файл транзакции становится все больше и больше. Большой файл транзакции может замедлять некоторые операции; следовательно, если Ваша программа на MapBasic выполняет редактирование таблицы, полезно учитывать следующие рекомендации, чтобы предотвратить увеличение размера файла:

- Регулярно сохраняйте внесенные изменения (то есть выполняйте оператор **Commit**). Например, Вы могли бы настроить Вашу программу так, чтобы сохранение производилось после каждых 100 изменений. Сохранение Ваших изменений освобождает файл транзакции.
- Используйте оператор **Set Table... FastEdit**, чтобы включить режим Быстрого Редактирования. В этом режиме изменения сохраняются в таблице немедленно, вместо того, чтобы записываться в файл транзакции. Подробнее см. в *Справочнике MapBasic* описание оператора **Set Table... Undo Off**.

## Разумное использование индексов

Некоторые запросы работают быстрее, если Вы проиндексируете одну или большее количество колонок в таблице. Например, выполнение оператора **Select** может быть ускорено, если Вы проиндексируете столбцы, используемые в предложениях **Where**, **Order By** или **Group By**. Однако неразумно индексировать каждый столбец в таблице. Индексация всех столбцов может замедлять некоторые операции, потому что MapInfo должен будет тратить большее количество времени на работу с индексами.

Если Ваше приложение выполняет интенсивное манипулирование таблицей, которое не включает запросы, Вы можете увеличить быстродействие, делая следующее:

1. Удалите индексы из Вашей таблицы (используя оператор **Drop Index**).
2. Выполните редактирование таблицы.
3. Сохраните внесенные изменения.
4. Проиндексируйте заново, используя оператор **Create Index**.

Этот способ может ускорить работу с таблицей, потому что MapInfo больше не должен будет поддерживать индексы во время операций редактирования.

## Использование “вложенных” выборок

Оператор **Select** может включать предложение **Where**, которое позволяет описать вложенную выборку, (см. *Справочник MapBasic*). Однако Вы можете обнаружить, что быстрее выполняются два не вложенных оператора **Select** вместо одного вложенного **Select ... Where (Select...)**.

Если Вы выполняете подвыборку этого типа:

```
... Where X = Any( Select ... ) ...
```

то MapInfo оптимизирует производительность, но только если колонка X проиндексирована.

## Оптимизация оператора Select

Некоторые типы запросов **Select** оптимизированы для повышения эффективности. См. описание оператора **Select** в *Справочнике MapBasic* или в интерактивной *Справочной системе*.

## Применение оператора Update

MapBasic позволяет обновлять объекты Карты по одному за раз, используя оператор **Alter Object** и затем используя оператор **Update** для отдельных записей, чаще всего в цикле. Этот тип преобразования таблицы более медленный, поскольку операторы применяются к каждой записи, которую надо изменить.

В некоторых случаях, можно ускорить этот процесс, применяя **Update** ко всей таблице, а не к отдельной строке. Например, смотрите раздел “*Быстрое обновление символов*” в *Справочной системе MapBasic*.

# Ввод/вывод в файлы

В языке MapBasic существуют значительные отличия в работе с файлами и таблицами MapInfo. Предыдущая глава была посвящена тому, что в MapBasic можно делать с таблицами; в этой главе мы коснемся файлов, которые не являются таблицами.

## В этой главе:

- ♦ Обзор ввода/вывода в файлы. . . . .164
- ♦ Файлы последовательного доступа. . . . .165
- ♦ Особенности работы с файлами в различных операционных системах и с национальными наборами символов. . . . .168

## Обзор ввода/вывода в файлы

Ввод/вывод в файлы (англоязычное сокращение "I/O") – это процесс чтения информации из файлов (ввод) и/или записи информации в файлы (вывод). Язык программирования MapBasic содержит набор стандартных для BASIC-операторов ввода/вывода и функций, позволяющих читать и/или писать в текстовые или двоичные файлы. Более того, поскольку MapInfo и MapBasic построены таким образом, чтобы их можно было использовать на разных вычислительных платформах, файловый ввод/вывод в MapBasic позволяет обеспечивать перенос данных без потерь.

Существует три типа доступа к файлам: последовательный, произвольный и двоичный. Какой из этих типов следует использовать в каждом конкретном случае, зависит от характера хранимых в файле данных:

- **Последовательный** доступ применяется при чтении текста из текстовых файлов с произвольной длиной строки. Например, одна строка может содержать 50 символов, а следующая – больше или меньше 50 символов и т.д. К таким файлам следует осуществлять последовательный доступ.
- **Произвольный** доступ применяется при чтении из текстовых файлов с фиксированной длиной строки. Скажем, если все строки в файле имеют длину ровно 80 символов, то к такому файлу следует применять произвольный доступ.
- **Двоичный** доступ применяется по отношению к нетекстовым (двоичным) файлам данных. Если Вы производите запись, используя двоичный доступ к файлу, MapInfo сохраняет числовые данные в таком файле оптимальным образом (минимизирует размер файла). Двоичные файлы нельзя просматривать и редактировать в текстовом редакторе; но зато они позволяют более экономно хранить числовые данные.

Независимо от выбранного типа доступа, первое Ваше действие ввода или вывода – это открытие файла. В MapBasic файл можно открыть с помощью оператора **Open File**. Этот оператор может содержать несколько необязательных предложений, которые следует употреблять в зависимости от типа файла. Следующий оператор открывает текстовый файл для чтения с последовательным доступом:

```
Open File "settings.txt" For Input As #1
```

При открытии файла указывается номер файла; в нашем примере – это число 1.

В дальнейшем в программе на файл можно ссылаться по номеру, заданному в операторе **Open File**. Например, чтобы считать текст из рассматриваемого нами файла в переменную типа String, следует выполнить оператор **Line Input** и указать в качестве параметра **Line Input** тот же номер (#1), что и в операторе **Open File**:

```
Line Input #1, s_nextline
```

Если Вы одновременно открываете несколько файлов, проверьте, чтобы им соответствовали различные номера.

В некоторых ситуациях Вам может понадобиться создать новый файл. Создать новый файл можно оператором **Open File** с предложением **For Output**:

```
Open File "workfile.txt" For Output As #2
```

Вы можете также использовать предложение **For Append** в операторе **Open File**. В режиме Append MapBasic создает файл с указанным именем, если такой файл не существовал, или добавляет данные в конец файла, если файл с указанным именем уже существует. Окончив чтение из файла или запись в файл, выполните оператор **Close File**. Например:

```
Close File #1
```

Номер файла в этом операторе имеет тот же смысл, что и номер в операторе **Open File**. Знак (#) можно опустить. Для того, чтобы сохранить изменения, внесенные в созданный или существовавший файл, не обязательно выполнять особые операторы типа "сохранить"; изменения будут сохранены в момент выполнения оператора **Close File**. (В MapBasic имеется оператор **Save File**, но он предназначен для создания копии таблицы, а не для сохранения изменений в файле.)

Существует много причин, вызывающих ошибки времени выполнения, связанные с файловым вводом/выводом. Например, ошибка, если в операторе **Open File** указано неверное имя файла или если Вы попытаетесь открыть файл на запись, хотя ему присвоен режим доступа "только для чтения". При записи в файл ошибки времени выполнения могут возникать из-за того, что на диске больше нет свободного места. Ошибка будет также зафиксирована, если Вы станете открывать файл на запись в тот момент, когда этот файл уже открыт на запись другим пользователем в сети. При разработке приложений, использующих файловый ввод/вывод, следует включать в программу процедуры обработки исключительных ситуаций, которые бы контролировали ошибки и корректировали результаты, а также следует проверять состояние операционной среды перед выполнением подобных операций (например, размер свободного пространства на диске). Информацию об обработке исключительных (ошибочных) ситуаций содержит глава **Поиск ошибок и отладка программ**.

Иногда ошибку можно предотвратить, вызвав соответствующую информационную функцию. Например, перед тем, как выполнить **Open File**, можно обратиться к функции **FileExists( )**, чтобы выяснить, существует ли файл, который Вы собираетесь открывать. Также при создании временных рабочих файлов, чтобы узнать, какое имя присвоено файлу и в каком каталоге он расположен, можно обратиться к функции **TempFileName\$( )**. Также к операциям файлового ввода/вывода относятся следующие операторы и функции:

- Оператор **Kill** удаляет файлы.
- Оператор **Save File** создает копию файла.
- Оператор **Rename File** изменяет имя файла.
- Функции **ProgramDirectory\$( )**, **HomeDirectory\$( )** и **ApplicationDirectory\$( )** позволяют определить вид различных стандартных DOS-маршрутов во время выполнения программы. Например, Вам может потребоваться создать строку с названием файла, расположенного в каталоге MapInfo (скажем, для STARTUP.WOR), но полный DOS-маршрут к этому каталогу Вам заранее не известен. С помощью **ProgramDirectory\$( )** Вы можете определить, в каком каталоге установлен пакет MapInfo.

## Файлы последовательного доступа

Для осуществления последовательного доступа к файлу (чтения/ записи в текстовый файл со строками переменной длины), можно использовать один из трех режимов, задаваемых параметром **For** оператора **Open File**: **Input**, **Output** или **Append**.

Чтобы читать из уже существующего файла, используйте предложение **For Input**. Например, программа NVIEWWS.MB из набора примеров содержит следующий оператор открытия текстового файла на чтение:

```
Open File view_file For Input As #1
```

Строковая переменная "view\_file" содержит имя текстового файла.

Открыв текстовый файл, Вы можете осуществлять чтение из него оператором **Input #** или оператором **Line Input #**. Оператор **Line Input #** считывает целиком одну строку из текстового файла в переменную типа String. Оператор **Input #** позволяет рассматривать строку текста как набор значений, разделенных запятыми, и читать эти значения по отдельности.

Например, приложение NVIEWWS.MB считывает данные следующего вида:

```
"New York", -75.75, 42.83, 557.5  
"Texas", -100.2, 31.29, 1200
```

Каждая строка текстового файла содержит здесь четыре значения – название, координаты X и Y, а также размер изображения. Программа NVIEWWS.MB использует оператор **Input #** для того, чтобы считывать сразу по четыре значения из каждой строки в четыре различные переменные:

```
Input #1, vlist(tot).descript,  
        vlist(tot).x,  
        vlist(tot).y,  
        vlist(tot).zoom
```

Переменная "vlist" – это массив, определенный пользователем.

При последовательном считывании данных необходимо проверять результат чтения. После того, как считано содержимое последней строки файла, следующая операция чтения выдаст сообщение об ошибке. Чтобы избежать такой ошибки, проверяйте значение функции **EOF( )** (end-of-file – конец файла) после каждой операции чтения. Если функция **EOF( )** возвратила FALSE, то в файле еще остались несчитанные строки (то есть следующая операция чтения пройдет правильно). Если же **EOF( )** возвратит TRUE, то достигнут конец файла.



Чтение последней строки файла не означает, что будет установлен признак "конец файла". Функция **EOF( )** возвращает TRUE только после того, как Вы попытаетесь в первый раз считать строку за признаком конца файла.

---

Чтобы создать файл, содержащий значения, разделенные запятыми, надо применить оператор **Open File** с предложением **For Output** или **For Append**. После того, как файл будет открыт, с помощью **Write #** данные можно записывать в файл. В операторе **Write #** можно задать перечень значений, которые следует записывать в каждую строку файла. Например, в приложении NVIEWWS.MB из набора примеров оператор **Write #** используется (в цикле) для записи в файл четырех значений (названия, координат X, Y и размера):

```
Write #1, vlist(i).descript, vlist(i).x, vlist(i).y, vlist(i).zoom
```

Оператор **Write #** заключает в файл все строки в двойные кавычки, как Вы видели выше ("New York"...). Иногда бывает неудобно использовать **Write #**, поскольку мы не хотим заключать строки в кавычки. Чтобы избежать этого, применяйте **Print #** вместо **Write #**.

Чтобы считать всю строку в переменную типа String, можно использовать оператор **Line Input #**. С помощью оператора **Print #** можно создавать файлы, которые впоследствии будут читаться оператором **Line Input #**. Пример использования **Print #** и **Line Input #** для чтения или записи целой строки текста можно найти в программе примере AUTO\_LIB.MB. Программа AUTO\_LIB читает и записывает файлы описания Рабочих наборов MapInfo (в частности, Рабочий набор STARTUP).

Не допускается запись в последовательный файл, который был открыт для чтения, и чтение из последовательного файла, который был открыт для записи.

## Файлы произвольного доступа

Чтобы открыть произвольный доступ к файлу, следует употребить предложение **For Random** в операторе **Open File**:

```
Open File "datafile.dat" For Random As #1 Len = 80
```

При доступе к файлу в режиме **Random** следует указать длину строк в файле в предложении **Len**. Напомним, что любой текстовый файл содержит невидимые пользователю служебные символы конца строки. Длина строки в предложении **Len** (в приведенном примере – 80) должна включать в себя символы конца строки (обычно это символы перевода каретки и новой строки).

К файлу с произвольным доступом применимы операции чтения и записи с помощью операторов **Get** и **Put**; см. описание в *Справочнике MapBasic*.

## Двоичные файлы

Двоичные файлы предназначены для хранения числовых величин в двоичном формате. Следующий оператор показывает, как открыть двоичный файл:

```
Open File "settings.dat" For Binary As #1
```

Если файл открыт как двоичный, к нему применимы операторы чтения и записи **Get** и **Put**; см. *Справочник MapBasic*.

Хранение числовых данных в двоичном формате является наиболее эффективным по использованию дискового пространства. Например, целые числа (тип Integer) занимают ровно четыре байта, вне зависимости от значения. С другой стороны, целое значение 111222333, хранящееся в текстовом файле, будет занимать девять байт. Двоичное представление является наиболее компактным для нетекстовых данных. Однако, чтобы иметь возможность просматривать данные в текстовом редакторе, Вам придется использовать текстовый формат вместо двоичного.

В двоичных файлах могут присутствовать строки символов, но они должны быть фиксированной длины.

## Особенности работы с файлами в различных операционных системах и с национальными наборами символов

При возникновении проблем чтения текстовых файлов, созданных на другой вычислительной платформе или с использованием другого национального набора символов, может потребоваться применить в операторе **Open File** необязательный параметр **CharSet**. В компьютере каждому символу клавиатуры сопоставлен некоторый числовой код. Например, английскому символу "А" на Вашей клавиатуре соответствует код символа 65. Набором символов (или кодировкой символов) называется совокупность символов, доступных в Вашем компьютере и сопоставленных им числовых кодов.

В разных странах могут использоваться различные наборы символов. Например, в версии Windows для Северной Америки и Западной Европы код символа 176 соответствует символу градуса; однако, в Windows с другим национальным набором символов коду 176 может соответствовать другой символ. Поэтому при чтении файла, созданного с применением другого набора символов, могут возникнуть проблемы.

Чтобы устранить подобное несоответствие, указывайте параметр **Char Set** в операторе **Open File**. Предложение **CharSet** позволяет явно задать набор символов, который используется тем или иным файлом. Если правильно указать **CharSet** для файла, то MapInfo Professional гарантирует корректное чтение из файла (и запись в файл). Список имен наборов символов, которые могут быть указаны в предложении **CharSet**, приводится в главе **CharSet** в *Справочнике MapBasic*.

## Функции ввода/вывода файлов

С помощью следующих функций можно получить информацию об открытых файлах:

- **FileAttr( )** – возвращает код режима доступа к файлу (**Input**, **Out Put**, **Append**, **Random** или **Binary**).
- **EOF( )** – возвращает истину (TRUE), если Вы пытаетесь читать после конца файла или перемещаете указатель за конец файла.
- **Seek( )** – возвращает номер позиции в файле в виде смещения (в байтах). В файлах с произвольным доступом (**Random**) – это номер последней обрабатывавшейся записи, умноженный на длину записи.
- **LOF( )** – возвращает длину файла в байтах.

В качестве параметра в каждой из этих функций указывается номер файла, присвоенный в операторе **Open File**. Подробное описание функций можно найти в *Справочнике MapBasic*.



# Географические и графические объекты

Одно из наиболее существенных достоинств языка MapBasic – его способность обрабатывать и изменять объекты на Карте – дуги, эллипсы, рамки, линии, точки, ломаные, прямоугольники, области, скругленные прямоугольники и текстовые объекты. В этой главе мы рассмотрим, как из программы на MapBasic обращаться к объектам Карты, создавать и изменять их. Вместе с тем заметим, что Вы должны понимать принципы построения таблиц MapInfo прежде чем изучать, как с помощью MapBasic сохранять объекты в таблице. Поэтому прочтите **Работа с таблицами**, если Вы до сих пор этого не сделали.

## В этой главе:

- ♦ Переменные типа Object .....170
- ♦ Работа с колонкой Obj .....170
- ♦ Определение атрибутов объекта .....172
- ♦ Создание новых объектов .....180
- ♦ Создание новых объектов на основе уже существующих .182
- ♦ Изменение объектов .....185
- ♦ Работа с подписями.....187
- ♦ Координаты и единицы измерения.....191
- ♦ Географические запросы .....193

## Переменные типа Object

Тип данных **Object** в языке MapBasic позволяет работать как с простыми графическими объектами (такими как линии), так и со сложными объектами (например, областями). (для программистов на Visual Basic: **Object** тип MapBasic относится к графическим элементам, а не к OLE объектам.)

С переменными MapBasic типа **Object** можно работать почти так же, как и с переменными других типов: им можно присваивать значения переменных типа **Object**, передавать такие переменные как аргументы процедур и функций, а также сохранять значения переменных типа **Object** в таблицах MapInfo.

Определить переменную объект можно с помощью оператора **Dim**:

```
Dim Myobj, Office As Object
```

Вам не надо указывать конкретно, какой тип графических объектов будет храниться во вводимой переменной. Переменная типа **Object** может содержать любой графический объект Карты или Отчета.

Знак равенства (=) используется для присвоения значения переменной объекту, например:

```
Office = CreatePoint(73.45, 42.1)  
Myobj = Office
```

Можно присвоить объект, который является значением другой переменной объекта, значение функции, возвращающей объекты, а также значение табличного выражения вида *имя\_таблицы.Obj*. Константные выражения типа **Object** в MapBasic не применяются.

Переменная объект хранит всю информацию об объекте на Карте. Например, если сохранить линию в переменную типа **Object**, то такая переменная будет содержать не только географическую информацию о линии (т.е. координаты начала и окончания линии), но и информацию о представлении (цвет, толщину и тип линии). В языке MapBasic имеется также четыре типа переменных стиля (**Pen**, **Brush**, **Symbol** и **Font**), в которых можно хранить описание типов (стилей) графических объектов без конкретных координат.

## Работа с колонкой Obj

Специальная колонка с названием "Obj" предназначена для хранения информации о графических объектах таблицы. Каждая таблица, которой сопоставлены графические объекты, содержит колонку "Obj", хотя колонка "Obj" и не показывается обычно в окне Списка.

Обратиться к этой колонке можно следующим образом: *имя\_таблицы.obj* или *имя\_таблицы.object*. В следующем примере объявляется переменная типа **Object** (*current\_state*), а затем копируется первый объект из таблицы STATES в эту переменную.

```
Dim current_state As Object  
Open Table "states"  
Fetch First From states  
current_state = states.obj
```

С колонками графических объектов можно осуществлять те же операции, что и с обычными колонками. Можно использовать колонку объектов в SQL-запросах, производить обновление (update) значений (объектов) в такой колонке, присваивать содержимое колонки переменным.

Ниже в примере создается таблица, содержащая сокращенные названия штатов и их площади; колонка "Obj" используется в качестве одного из параметров функции **Area( )**:

```
Select state, Area(obj, "sq mi")
  From states
```

В следующем примере создается таблица из одной записи, которая содержит общую длину шоссе штата Калифорния в милях ("CA" – код штата):

```
Select Sum(ObjectLen(obj, "mi"))
  From highways
  Where obj Within (Select obj From states Where state = "CA")
```

Некоторым записям не соответствуют объекты на Карте. Например, если Вы станете геокодировать свою базу данных в MapInfo, то в процессе геокодирования некоторым записям таблицы будут сопоставляться точечные объекты; тем же записям, которые не были геокодированы, не будет сопоставлено никакого объекта на Карте. Чтобы выбрать записи, которым не соответствуют графические объекты, используйте условие **Not obj** в предложении **Where** оператора **Select**. Вот пример оператора, выбирающего все записи, которым не сопоставлены объекты на Карте:

```
Select *
  From sites
  Where Not obj
```

## Создание колонки Object

Не все таблицы отображаются на Карте. Например, если в качестве таблицы Вы используете файл базы данных или электронной таблицы, подготовленный не в MapInfo, то такую таблицу нельзя увидеть в окне Карты. Чтобы отобразить такую таблицу на Карте, следует выполнить оператор **Create Map**, который добавляет к таблице колонку графических объектов "Obj".

Чтобы удалить колонку "Obj" из таблицы, выполните оператор **Drop Map**. Запомните, что **Drop Map** удалит колонку объектов полностью. Иногда Вам может потребоваться удалить лишь некоторые объекты из таблицы, не убирая совсем колонку Obj; этот процесс называется "раскодированием" таблицы. Для удаления объектов из колонки "Obj" (но не ее самой), используйте оператор **Delete Object**.

Чтобы определить, имеется ли в таблице колонка графических объектов, вызовите функцию **TableInfo( )** с параметром **TAB\_INFO\_MAPPABLE**.

## Ограничения на колонки географических объектов

Объектные колонки имеют ограничения, которые не относятся к колонкам других типов. В частности, в каждой таблице может быть только одна колонка графических объектов. В связи с этим, важно помнить, что при создании выборки с объединением двух таблиц, обе из которых имеют колонку "Obj", результирующая таблица будет содержать только одну колонку с графическими объектами из этих таблиц (в результирующую таблицу попадут объекты из той таблицы, которая указана первой в предложении **From** оператора **Select**).

Следующий пример показывает, как осуществить запрос к двум таблицам, имеющим графические объекты: таблице States и таблице Outlets, которая содержит точечные объекты, обозначающие предприятия розничной торговли. В предложении **From** оператора **Select** указываются названия обеих таблиц. Но поскольку таблица States указана первой, результирующая таблица содержит графические объекты из таблицы States.

```
Select *
  From states, outlets
  Where states.state = outlets.state
Map From selection
```

Если в приведенном примере в предложении **From** указать первой таблицу Outlets (см. ниже), то оператор **Select** создаст таблицу, содержащую точечные объекты (торговые точки), а не границы штатов:

```
Select *
  From outlets, states
  Where outlets.state = states.state
Map From selection
```

Каждой записи таблицы может соответствовать не более одного графического объекта. Поэтому графические объекты могут быть составными. Объект типа "Область" может состоять из нескольких многоугольников (полигонов); так что группу островов можно представить единым графическим объектом (областью). Аналогично, и ломаная может состоять из нескольких сегментов. Чтобы определить, из скольких многоугольников состоит область или из скольких сегментов состоит ломаная, выберите этот объект и выполните команду MapInfo **Правка > Геоинформация**. Чтобы определить те же параметры в программе, используйте функцию **ObjectInfo( )** с кодом **TAB\_INFO\_NPOLYGONS**.

## Определение атрибутов объекта

Таблица MapInfo может содержать разные типы графических объектов. Скажем, Карта города может состоять из линий и областей. Чтобы определить тип графического объекта, обратитесь к функции **ObjectInfo( )** с параметром **OBJ\_INFO\_TYPE**. Подробно функция **ObjectInfo( )** описана в *Справочнике MapBasic*.

При работе с окном MapBasic в режиме диалога имеются и другие способы узнать тип графического объекта. Например, можно выполнить следующий оператор в окне MapBasic, чтобы выдать окно сообщения с типом объекта:

```
Fetch First From world
Note world.obj
```

Следующий оператор выбирает все объекты типа "Text" из окна Отчета.

```
Select *
  From Layout1
  Where Str$(obj) = "Text"
```

Чтобы узнать географические координаты объекта, обращайтесь к функции **ObjectGeography( )**. Например, если Вы хотите определить координаты X и Y концов линейного объекта, вызовите функцию **ObjectGeography( )**. Процедура определения

координат узлов ломаной или области сложнее, поскольку число узлов может быть произвольным. Для определения координат узлов ломаной или области используются функции **ObjectNodeX( )** и **ObjectNodeY( )**.

Положение центроида объекта определяется с помощью функций **Centroid( )** или **CentroidX( )** и **CentroidY( )**. Минимальный описывающий прямоугольник для объекта можно найти с помощью функции **MBR( )**.

Другие атрибуты графических объектов возвращает функция **ObjectInfo( )**. Например, после операции над графическими объектами из таблицы и присвоения значения выражения переменной типа "Object", можно вызвать **ObjectInfo( )**, чтобы определить тип объекта (линия, область и т.д.), либо обратиться к **ObjectInfo( )**, чтобы скопировать стиль объекта (**Pen**, **Brush**, **Symbol** или **Font**). Для текстовых объектов функция **ObjectInfo( )** выдает надпись, хранящуюся в текстовом объекте.

Многие стандартные функции языка MapBasic в качестве одного из аргументов используют графические объекты, возвращая при этом различные характеристики объектов. Например, функции **Area( )**, **Perimeter( )** и **ObjectLen( )** используют графические объекты в качестве параметров. Ниже в примере вычисляется площадь зоны затопления ("floodarea"):

```
Dim floodarea As Float
Open Table "floodmap"
Fetch First From floodmap
floodarea = Area(floodmap.obj, "sq km")
```

Обратите внимание, что подписи – это не то же самое, что текстовые объекты. Запрашивая текстовый объект, Вы вызываете функции типа **ObjectInfo( )**, а запрашивая подпись, Вы вызываете такие функции, как **Labelinfo( )**. Процедуры подписывания описаны в разделе: **Работа с подписями на стр. 187**.

Стили объектов (Pen, Brush, Symbol, Font)

Каждый графический объект имеет один или несколько параметров, условно называемых стилем. Например, каждая линия имеет стиль **Pen**, который определяет цвет линии, ее толщину и тип (или шаблон, скажем, пунктирная или непрерывная), каждый точечный объект имеет стиль **Symbol**, который определяет вид символа, его цвет и размер. Замкнутые (или площадные) объекты, такие как области, имеют стиль **Pen**, а также стиль **Brush** (штриховка).


В таблице ниже приведены определения четырех стилей объектов.

Объект	Компоненты стиля
Pen	Толщина, тип и цвет линии.
Brush	Шаблон штриховки, основной цвет и цвет фона для внутренней части области.
Font	Название шрифта, тип, размер, цвет, фон; для текстовых объектов.

Symbol	Условные знаки, совместимые с MapInfo Professional 3.0: вид, цвет, размер.  Для символов условных знаков шрифтов TrueType: вид, цвет, размер, имя шрифта, стиль (например, курсив), параметры поворота.  Для пользовательских символов, основанных на файлах с растровыми картинками: имя файла, цвет, размер и атрибуты стиля.
--------	---

Для получения подробной информации о стилях обратитесь к разделам: оператор **Brush**, оператор **Font**, оператор **Pen** и оператор **Symbol** в *Справочнике MapBasic* и *Справочной системе*.

Язык MapBasic имеет различные операторы и функции для создания и опроса объектов (такие, как оператор **Create Text**, функция **CreateLine( )** и другие). Каждый из операторов создания объектов имеет дополнительные предложения для настройки стилей для такого объекта. Например, оператор **Create Line** имеет дополнительное предложение **Pen** позволяющее настроить стиль линии. Если оператор создания объекта применяется без указания стиля, то MapInfo присваивает объекту текущий стиль.

 Вы не можете использовать оператор "=", чтобы сравнить два стиля. Например, следующая программа, которая пытается сравнивать две переменные типа **Brush**, генерирует ошибку времени выполнения.

```
Dim b1, b2 As Brush

b1 = MakeBrush(2, 255, 0)
b2 = CurrentBrush()

If b1 = b2 Then
    Note "Оба стиля Brush одинаковы."
End If
```

Если надо сравнить два стиля, используйте функцию **Str\$( )** для конвертации каждого стиля в строковое выражение. Например, следующий оператор сравнивает две переменные типа **Brush**:

```
If Str$(b1) = Str$(b2) Then ...
```

Если надо сравнить специфические элементы стиля (посмотреть, имеют ли два стиля **Symbol** один и тот же размер), используйте функцию **StyleAttr( )** для извлечения индивидуального элемента стиля (цвет и др.), и затем сравните отдельные элементы.

### Стили Шрифтов

Каждый текстовый объект имеет стиль (начертание). Стиль шрифта включает сам шрифт (например Times Roman или Helvetica), стиль текста (например полужирный, курсивный, и т.д.) и цвет. Стиль шрифта также содержит информацию о размере; однако, размер иногда игнорируется. В следующем списке перечислено, как размер шрифта действует на различные типы текста.

- Когда Вы создаете текстовый объект в окне Отчета, размер шрифта управляет высотой текста. Если стиль шрифта указывает размер 10 пунктов, текстовый объект определен как 10–пунктовый текст. Текст может не отображаться на экране высотой в 10 пунктов, но когда Вы напечатаете Отчет, высота текста будет 10 пунктов.
- Когда Вы создаете текстовый объект в окне Карты оператором **Create Text**, MapInfo игнорирует размер шрифта. В этой ситуации высота текста определяется координатами Карты, которые Вы устанавливаете в операторе **Create Text**. Когда выполняется оператор **Create Text**, Вы указываете координаты X и Y, которые определяют прямоугольную область на Карте; текстовый объект заполняет эту область. Из-за этого текстовые объекты, сохраненные в “графической” таблице, будут становиться больше, если Вы увеличиваете окно Карты, и меньше, если уменьшаете.
- Когда используется функцию **CreateText( )** для создания текстового объекта на Карте, текущий размер шрифта в пунктах управляет начальным размером текста. Таким образом, увеличение масштаба приводит к увеличению размера текста.
- Когда создается подпись в окне Карты, размер шрифта управляет высотой текста. Текст отображается на экране и печатается с указанной высотой. Обратите внимание, что подписи ведут себя по-другому, чем текстовые объекты, сохраненные в таблице. Процедуры подписывания описаны в разделе: **Работа с подписями на стр. 187**.

## Комбинированные стили

Вы можете применить к объекту на Карте несколько стилей, так чтобы изменения прорисовывались друг за другом, образуя интересные оформительские эффекты. Комбинированные стили применяются к точкам, полилиниям и полигонам. Наиболее удобно это бывает при оформлении полилиний.

Рисунок 1 показывает линии, к которым применены обычные стили. На рисунке 2 показаны те же линии после применения к ним комбинации стилей.

Рисунок 1

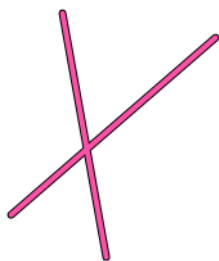
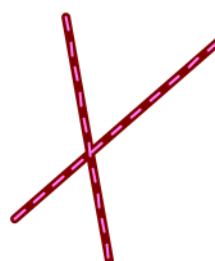


Рисунок 2



Комбинированные стили позволяют придавать Картам большую информационную и эстетическую насыщенность, избегая при этом создания множества слоев. Вы можете включать в комбинацию столько стилей, сколько пожелаете. Однако, при обилии стилей может замедлиться перерисовка экрана. Обычно для нужд картографии достаточно скомбинировать два или три стиля.

## Комбинация стилей как компонент оформления слоя

Чтобы задать комбинацию стилей, нужно установить флажок **Единообразно** в диалоге **Управление слоями > Свойства слоя** (или в диалоге **Стиль для масштабного диапазона**). Комбинированные стили становятся частью представления слоя и применяются к каждому объекту на слое. Комбинированные стили не сохраняются в TAB-файле, но сохраняются в Рабочем наборе, как часть настроек представления слоя. Это значит, что комбинацию стилей можно применять даже к TAB-файлам, открытым для чтения.

## Поддержка комбинированных стилей в MapBasic

MapBasic полностью поддерживает всю функциональность комбинированных стилей. Стили задаются в виде списка предложений **Line**, разделенного запятыми. Например, чтобы добиться эффекта, показанного в предыдущем примере, надо задать следующие предложения:

```
Line (7,2,8388608), Line (2,9,16744703)
```

Чтобы применить комбинацию стилей ко всем объектам слоя, добавьте предложение **Global** в оператор **Set Map**:

```
Set Map Layer 1 Display Global Global Line (7,2,8388608), Line  
(2,9,16744703)
```

Первое предложение **Line**, встретившееся в операторе MapBasic, прорисовывается первым, следующее за ним после запятой предложение **Line** – вторым и т.п.

О том, как создавать комбинацию стилей, читайте в описании оператора **Set Map** в *Справочнике MapBasic*. Функции, возвращающие атрибуты комбинированного стиля, описаны в главе *Новые и улучшенные операторы и функции MapBasic* в *Справочнике MapBasic*.

## Еще примеры комбинированных стилей

В следующих примерах показано, как еще можно комбинировать стили в MapBasic.

### Пример с точечными объектами (символами)

С помощью комбинации стилей можно создавать новые символы.

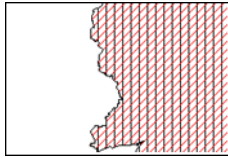


```
Set Map Layer 1 display Global Global  
Symbol (32,16777136,24),  
Symbol (36,255,14)
```



### Пример с полигоном (областью)

При оформлении региона заданы стили, позволяющие комбинировать два цвета и две штриховки.



```
Set Map Layer 1 Display Global Global  
Brush (4,0,16777215),  
Brush (5,16711680)
```

## Переменные стилей

В MapBasic имеются переменные стилей следующих типов: **Pen**, **Brush**, **Symbol** и **Font**, что соответствует стилям графических объектов. Присвоить значение переменной стиля можно четырьмя способами:

- Создать стиль с помощью **MakePen( )**, **MakeBrush( )**, **MakeSymbol( )**, **MakeFont( )**, **MakeCustomSymbol( )** или **MakeFontSymbol( )**, а затем присвоить его переменной стиля. Перечисленные функции позволяют явно задать нужный стиль. Так, в программе SCALE-BAR из набора примеров **MakeBrush( )** используется для построения штриховок из черного и белого цветов, которые используются для рисования шкалы.
- Вызвать **CurrentPen( )**, **CurrentBrush( )**, **CurrentFont( )**, или **CurrentSymbol( )** и присвоить значение функции переменной стиля. Указанные функции возвращают текущие стили (т.е. те стили, которые показываются в диалогах MapInfo **Настройка > Стиль Линий, Стиль Областей, Стиль Символов** и **Стиль Текста**, когда ни один графический объект не выбран).
- Вызвать **ObjectInfo( )**, определить с ее помощью стиль некоторого объекта и присвоить этот стиль переменной.
- Открыть диалог, в котором пользователь мог бы выбрать тот или иной стиль. В любом диалоге, содержащем кнопку **PenPicker**, **BrushPicker**, **SymbolPicker** или **FontPicker**, пользователь может выбрать стиль, указав мышью на такую кнопку. Подробные сведения о диалогах содержит глава **Создание интерфейса пользователя**.

Следующий пример демонстрирует, как создать стиль **Pen** с помощью обращения к функции **MakePen( )**. Значение стиля присваивается переменной типа **Pen**.

```
Dim p_var as Pen  
p_var = MakePen(1, 10, RGB(128, 128, 128))
```

Аргументы функции **MakePen( )** определяют стиль линии: 1 означает, что толщина линии составляет один пиксел; 10 задает номер шаблона (dotted); а функция **RGB( )** определяет цвет. Подробнее эти три параметра, с помощью которых создается стиль линий (со списком всех шаблонов линий), описаны в разделе **Предложение Pen** в *Справочнике MapBasic*. Аналогично, создание стилей **Brush**, **Font** и **Symbol** описано в разделах, описывающих предложения **Brush**, **Font** и **Symbol**.

В следующем примере показано, как присвоить переменной типа **Pen** стиль линии одного из существующих объектов:

```
p_var = ObjectInfo(obj_var, OBJ_INFO_PEN)
```

Присвоив значение переменной типа **Pen**, можно использовать эту переменную при создании графических объектов:

```
Create Line Into Variable obj_var
  (-73, 42) (-74, 43)
  Pen p_var
```

Функция **StyleAttr( )** возвращает одну из компонент заданного стиля. Например, в программе TEXTBOX из набора примеров показывается диалог, в котором пользователь может выбрать стиль линии; выбранный стиль сохраняется в переменной типа **Pen** с именем "pstyle". Затем в программе TEXTBOX выполняется оператор, который выделяет цвет стиля и присваивает его переменной типа Integer ("line\_color"):

```
line_color = StyleAttr(pstyle, PEN_COLOR)
```

Цвета в MapInfo хранятся в виде целых чисел. Например, черному цвету соответствует число 0, синему – 255. Функция **RGB( )** языка MapBasic вычисляет значение цвета на основании заданного соотношения красного, зеленого и синего оттенков. Например, RGB(0, 255, 0) возвращает зеленый цвет.

Для обозначения цветов используйте функцию **RGB( )**. Например:

```
highway_style = MakePen(2, 2, RGB(0, 0, 255))
```

Вместо обращения к **RGB( )** можно использовать одно из стандартных числовых обозначений цветов (BLACK, WHITE, RED, GREEN, BLUE, YELLOW, CYAN и MAGENTA), определенных в файле MAPBASIC.DEF.

## Выбор объектов с заданным стилем

Функция **ObjectInfo( )** позволяет извлекать значения типа **Pen**, **Brush**, **Symbol** или **Font** из объекта. Если только Вы извлекли **Pen**, **Brush**, **Symbol** или **Font**, Вы можете воспользоваться функцией **StyleAttr( )**, чтобы рассмотреть индивидуальные элементы (например, чтобы определить цвет стиля **Symbol**).

Вы можете использовать оператор **Select**, чтобы выбрать объекты заданного стиля. Как показывает следующий пример, предложение **Where** оператора **Select** может вызывать функции **ObjectInfo( )** и **StyleAttr( )**, чтобы выбрать только те объекты, которые имеют некоторые атрибуты (например, объекты некоторого цвета).

Следующий пример добавляет кнопку на инструментальную панель **Программы**. Если Вы выбираете точечный объект и затем нажимаете эту кнопку, программа выбирает все точечные объекты в таблице, которые имеют тот же самый цвет, что и выбранная точка.

```
Include "mapbasic.def"
Declare Sub Main
Declare Sub SelectPointsByColor()

Sub Main
  ' Добавим кнопку к панели "Программы".
  Alter ButtonPad "Программы" Add
  PushButton
    Calling SelectPointsByColor
    HelpMsg "Выбор точек того же цвета\nВыбор по цвету"
```

```
End Sub

Sub SelectPointsByColor
    Dim i_color, i_open As Integer
    Dim symbol_style As Symbol
    Dim object_name, table_name As String

    ' Запомните, сколько таблиц уже открыто
    i_open = NumTables()

    ' Определим имя текущей таблицы.
    table_name = SelectionInfo(SEL_INFO_TABLENAME)
    If table_name = "" Then
        ' ... ничего не выбрано; завершаем работу.
        Exit Sub
    End If
    ' Если таблица не содержит графики, то выход.
    If Not TableInfo(table_name, TAB_INFO_MAPPABLE) Then
        Exit Sub
    End If
    ' Выбранный объект точка?
    ' Если да, определим символ условного знака и цвет.
    Fetch First From Selection
    object_name = Str$(Selection.obj)
    If object_name = "Point" Then
        symbol_style = ObjectInfo(Selection.obj, OBJ_INFO_SYMBOL)
        i_color = StyleAttr(symbol_style, SYMBOL_COLOR)
    End If

    ' Обращение к "Selection.obj" может заставить MapInfo Professional
    ' открыть временную таблицу Query1 (или Query2...)
    ' просто закроем ее, для порядка.
    If NumTables() > i_open Then
        Close Table TableInfo(0, TAB_INFO_NAME)
    End If

    If object_name <> "Point" Then
        '...если объект не точка, то выход.
        Exit Sub
    End If
    ' Выберите все строки, содержащие точечные объекты.
    Select * From table_name
    Where Str$(Obj) = "Point"
    Into Color_Query_Prep NoSelect

    ' Выберите точечные объекты, имеющие тот же
    ' цвет, что и у исходных объектов.
    Select * From Color_Query_Prep
    Where
        StyleAttr(ObjectInfo(obj, OBJ_INFO_SYMBOL), SYMBOL_COLOR)
        = i_color
    Into Color_Query
```

```
Close Table Color_Query_Prep  
  
End Sub
```

Этот пример работает с объектами типа "Точки", но те же самые методы могут использоваться с другими типами объектов. Например, чтобы работать с площадными объектами вместо точек, Вы проверяли бы для имени объекта "Region" вместо "Point", и вызывали бы **ObjectInfo( )** с параметрами **OBJ\_INFO\_BRUSH** вместо **OBJ\_INFO\_SYMBOL** и т.д.

## Создание новых объектов

В языке MapBasic имеется набор операторов и функций, позволяющих создавать графические объекты. В данном разделе эти операторы и функции будут описаны кратко; подробные сведения о них можно найти в *Справочнике MapBasic*.

### Операторы создания объектов

Следующие операторы можно использовать для создания новых графических объектов. Их можно использовать в окнах Отчетов. Все эти операторы можно использовать в окнах Карт, за исключением **Create Frame**.

- **Оператор Create Arc**: создает дугу.
- **Оператор Create Ellipse**: создает эллипс или окружность. Окружность – это просто частный случай дуги, дуга с одинаковыми радиусами.
- **Оператор Create Frame**: создает рамку. Рамки – это объекты, существующие только в окнах Отчетов; каждая Рамка может содержать одно окно. Таким образом, чтобы разместить две Карты на одной странице, надо создать две Рамки.
- Оператор **Create Line**: создает линию.
- Оператор **Create Point**: создает точку.
- Оператор **Create Pline**: создает ломаную.
- Оператор **Create Rect**: создает прямоугольник.
- Оператор **Create Region**: создает область (многоугольник).
- Оператор **Create RoundRect**: создает скругленный прямоугольник.
- Оператор **Create Text**: создает текстовый объект.
- Оператор **AutoLabel**: позволяет создавать подписи (т.е. текстовые объекты) на Косметическом слое. Фактически этот оператор не создает подписи, он создает текстовые объекты. Для создания подписей, используйте оператор **Set Map**.

### Функции создания объектов

Следующие функции языка MapBasic возвращают значения-объекты:

- функция **CreateCircle( )**: возвращает окружность.
- функция **CreateLine( )**: возвращает линию.
- функция **CreatePoint( )**: возвращает точечный объект.
- функция **CreateText( )**: возвращает текстовый объект.

В некотором смысле функции создания графических объектов мощнее, чем соответствующие им операторы, поскольку эти функции можно встраивать в сложные операторы. Например, следующий оператор **Update** использует функцию **CreateCircle( )** для создания окружностей, соответствующих каждой записи таблицы:

```
Update sites  
  Set obj = CreateCircle(lon, lat, 0.1)
```

В данном примере предполагается, что таблица Sites содержит колонку "lon" со значениями долготы (координата X) и колонку "lat" со значениями широты (координата Y).

## Создание объектов с переменным числом узлов

Области и ломаные линии устроены сложнее, чем другие графические объекты. Такие объекты могут иметь произвольное число узлов (до 32 763 узлов в каждом объекте).

Можно создавать области с помощью оператора **Create Region**. В операторе **Create Region** можно явно указать число узлов создаваемого объекта. Однако зачастую заранее неизвестно, сколько узлов будет в создаваемом объекте. Скажем, программа может читать координаты объекта из текстового файла, а затем строить область, каждому узлу которой соответствует пара координат из указанного файла. В подобном случае в программе нельзя предусмотреть заранее, сколько узлов будет в той или иной области, поскольку число узлов зависит от данных, содержащихся во внешнем файле.

Создавать области и полилинии (ломаные) в программе можно в два этапа:

1. Выполнить оператор **Create Region** или **Create Pline**, чтобы создать "пустой" графический объект (объект без узлов).
2. Выполнить оператор **Alter Object**, чтобы добавить узлы к "пустому" графическому объекту. Оператор **Alter Object** обычно выполняется в цикле, так что на каждой итерации цикла к графическому объекту добавляется один узел.

Следующий пример иллюстрирует эту процедуру:

```
Include "mapbasic.def"  
  
Type Point  
  x As Float  
  y As Float  
End Type  
  
Dim objcoord(5) As Point  
Dim numnodes, i As Integer, myobj As Object  
numnodes = 3  
set CoordSys Earth  
objcoord(1).x = -89.213 objcoord(1).y = 32.017  
objcoord(2).x = -89.204 objcoord(2).y = 32.112  
objcoord(3).x = -89.187 objcoord(3).y = 32.096  
  
Create Pline Into Variable myobj 0  
  
For i = 1 to numnodes  
  Alter Object myobj Node Add (objcoord(i).x,objcoord(i).y)
```

Next

```
Insert Into cables (obj) Values (myobj)
```

## Сохранение графических объектов в таблице

После того, как графический объект создан и присвоен переменной типа "Object", Вам обычно требуется сохранить созданный графический объект в некоторой таблице. До тех пор, пока объект не помещен в таблицу, пользователь не может увидеть его на экране.

Чтобы сохранить графический объект в таблице, следует выполнить оператор **Insert** или оператор **Update**. Какой из двух операторов использовать, зависит от того, следует ли сопоставить графический объект уже существующей записи таблицы или надо создать новую запись.

Оператор **Update** используется для добавления графического объекта в уже существующую запись таблицы. Если этой записи уже соответствовал графический объект, новый объект будет вставлен вместо него. Оператор **Update** можно применять к любой колонке таблицы; чтобы работать с графическими объектами, надо указать название специальной колонки "Obj".

Например, следующий оператор сохраняет точечный объект в первой записи таблицы Sites:

```
Update sites  
  Set Obj = CreatePoint(x, y)  
  Where RowID = 1
```

Оператор **Insert** используется для добавления новой записи в таблицу. **Insert** позволяет добавлять за один раз одну запись в указанную таблицу или вставлять группу строк из другой таблицы. Следующий оператор вставляет одну новую запись в таблицу Sites, причем в колонку "Obj" этой записи помещается графический объект "Линия":

```
Insert Into sites (Obj)  
  Values (CreateLine(x1, y1, x2, y2))
```

Программа TEXTBOX из набора примеров содержит как операторы **Insert**, так и **Update**. Эта программа рисует квадрат (графический объект "Прямоугольник") вокруг каждой выбранной текстовой подписи; каждый квадрат записывается с помощью оператора **Insert**. Кроме того, если пользователь устанавливает флажок **Сделать одинаковыми цвета текста и рамки**, то программа также изменяет цвет выбранного текстового объекта и с помощью оператора **Update** обновляет информацию о текстовом объекте в таблице.

Операторы **Insert** и **Update** представляют собой мощное и гибкое средство работы с таблицами. В приведенных примерах эти операторы применялись только к одной колонке (колонке графических объектов "Obj"); однако, с помощью **Insert** и **Update** можно работать с любыми колонками таблиц. Подробное описание операторов **Insert** и **Update** Вы найдете в *Справочнике MapBasic*.

## Создание новых объектов на основе уже существующих

В программе на языке MapBasic можно создавать новые графические объекты на основе уже имеющихся объектов. В этом разделе содержится обзор различных операторов и функций языка MapBasic; подробное описание каждой из них можно найти в *Справочнике MapBasic*.

## Создание буферной зоны

Буферная область (или просто "буфер") – это область, охватывающая все возможные объекты, удаленные от заданного графического объекта не более чем на некоторое расстояние. Буферы полезны при поиске объектов, находящихся в пределах заданного расстояния от других объектов. Например, Вы можете создать буфер вокруг трассы оптического кабеля, чтобы выяснить, какие земляные работы велись на расстоянии менее 300 метров от кабеля. Создать буфер можно с помощью оператора **Create Object**.

Вот пример, как создать 300–метровую буферную зону вокруг заданного сегмента кабеля, а затем найти места проведения ремонтных работ:

```
Dim danger_zone As Object

Create Object As Buffer
  From selection
  Into Variable danger_zone
  Width 300 Units "m"

Select * From dig_sites Where dig_site.obj Within danger_zone
```

В MapBasic также есть функция **Buffer( )**, которая возвращает графический объект, представляющий собой буферную зону.

## Объединение, пересечение и слияние

Оператор **Create Object** позволяет также находить объединение и пересечение областей. Если задать этот оператор в форме **Create Object As Merge**, то MapInfo удалит общие подобласти пересекающихся областей, создав одну результирующую область (полигон). При слиянии двух областей с единой границей (например, Московской и Тверской областей России), результирующая область будет охватывать территорию обеих областей. Граница между соседними областями будет удалена.

Следующий пример показывает, как объединить два штата из таблицы States:

```
Select *
  From states
  Where state ="CA" Or state = "NV"

Create Object As Merge
  From selection
  Into Table territory
```

Операция "Merge" ("слияние") представляет собой наложение по правилу "исключающего ИЛИ" (XOR). При слиянии двух областей, одна из которых полностью содержит другую, результатом операции будет внешняя область без той ее части, которая соответствует внутренней области (т.е. область с дырой).

Слияние создает новый графический объект. Склеиваемые области остаются в исходной таблице. Вы можете удалить исходные области следующим образом:

```
Select * From Territory Where TerrName = "Western Territory" or TerrName =
"NV"
Delete From selection
```

Операторы **Create Object As Union** и **Create Object as Intersection** позволяют создавать области, являющиеся логической комбинацией двух и более областей. Эти операторы отличаются от **Create Object As Merge** тем, что они действуют на все фрагменты исходных областей, а не только на общие их части. "Union" – это объединение всех областей. "Intersection" – зона пересечения областей. Графический объект, полученный в результате объединения или пересечения областей, может содержать новые узлы (т.е. такие, которых не было в исходных областях). В MapBasic также есть функция **Combine( )**, которая возвращает графический объект, являющийся комбинацией двух других объектов.

## Создание изограмм

Изограмма – это Карта точек, удовлетворяющих заданным условиям на дальность и время. Изограммами являются изохроны и зоны транспортной доступности по расстоянию. Изохрона – эта изолиния одновременности того или иного явления. Например, полигон или последовательность точек, определяющих территорию, границ которой можно достичь от заданной точки за заданное время по данной сети дорог. Изограмма, представляющая зону транспортной доступности – это полигон или последовательность точек, определяющих территорию, любой точки которой можно достичь, от исходной точки, пройдя заданное расстояние по данной сети дорог.

С помощью операторов **Create Object As Isogram** можно создать одну или несколько таких областей, каждая оформленная индивидуальным стилем штриховки и линии, чтобы различать их на Карте. Чтобы создать изограмму требуется использовать дополнительную внешнюю службу, например Envinsa.

Чтобы создать изограмму:

1. Подключитесь к Envinsa, выполнив оператор **Open Connection**.  
Оператор возвратит идентификатор соединения и сохранит его в указанной переменной.
2. С помощью оператора **Set Connection Isogram** настройте соединение.
3. Создайте требующуюся область оператором **Create Object As Isogram**.

## Создание сдвинутых копий объектов

Группу функций и операторов сдвига Offset можно использовать для создания новых объектов сдвинутых на заданное расстояние относительно исходных объектов.

Следующие операторы можно использовать для создания сдвинутых копий существующих объектов.

- **Функция Offset( )**: возвращает копию исходного объекта, смещённого на заданное расстояние и угол.
- **Функция OffsetXY( )**: возвращает копию исходного объекта, смещённую на заданное расстояние по осям X и Y.
- **Функция SphericalOffset( )**: возвращает копию исходного объекта, сдвинутую на заданное расстояние и угол. В этом случае требуется использовать сферические единицы измерения расстояния.
- **Функция SphericalOffsetXY( )**: возвращает копию исходного объекта, передвинутую в точку и на заданное расстояние. В этом случае требуется использовать сферические единицы измерения расстояния.



- **Функция `CartesianOffset( )`**: возвращает копию исходного объекта, смещённого на заданное расстояние и угол. В этом случае требуется использовать декартовы единицы измерения расстояния.
- **Функция `CartesianOffsetXY( )`**: возвращает копию исходного объекта, смещённую на заданное расстояние по осям X и Y. В этом случае требуется использовать декартовы единицы измерения расстояния.

## Изменение объектов

### Общая процедура изменения графических объектов

MapBasic содержит несколько операторов, позволяющих вносить изменения в существующие графические объекты на Карте. Независимо от того, с помощью какого оператора Вы вносите изменения, процесс модификации графического объекта выглядит следующим образом:

1. Создается копия исходного объекта. (Как правило, для этого объявляют переменную типа "Object", выполняют оператор **Fetch**, чтобы переместить указатель файла, а затем выполняют оператор присваивания вида *имя\_переменной = имя\_таблицы.obj*).
2. Выполняются операторы или функции, которые изменяют графический объект. (Это обычно один или несколько операторов **Alter Object**.)
3. Выполняется оператор **Update**, чтобы сохранить измененный графический объект обратно в таблицу.

Программа TEXTBOX из набора примеров может служить иллюстрацией этого процесса. Если пользователь установил флажок **Сделать одинаковыми цвета текста и рамки**, то программа TEXTBOX использует оператор **Alter Object** для изменения цвета выбранного объекта, а затем – оператор **Update** для того, чтобы сохранить измененный текстовый объект в таблице.

### Перемещение объекта

Оператор **Objects Move** используется для перемещения объектов на заданное расстояние. Можно задать единицы измерения расстояний и тип. Оператор **Objects Offset** используется для создания копий объектов на заданном расстоянии. Можно задать тип и единицы измерения расстояний, способ хранения копий – в той же таблице или в другой.

### Перемещение объектов и их узлов

Для того чтобы изменить координаты объектов, используйте оператор **Alter Object**, в котором присутствует параметр **Geography**. Оператор **Alter Object**, скорее всего, придется использовать несколько раз (один раз, для того чтобы изменить X-координату, второй – Y-координату).

### Изменение стилей графического объекта (Pen, Brush, Font, Symbol)

С помощью оператора **Alter Object** можно изменить стиль любого графического объекта. В примере ниже оператор **Alter Object** используется для изменения выбранных объектов из таблицы:

```
Include "mapbasic.def"
Dim myobj As Object, mysymbol As Symbol
mysymbol = CurrentSymbol()
Fetch First From selection
myobj = selection.obj
If ObjectInfo(myobj, OBJ_INFO_TYPE) = OBJ_POINT Then
    Alter Object myobj
        Info OBJ_INFO_SYMBOL, mysymbol
    Update selection Set obj = myobj Where RowID = 1
Else
    Note "Выбранным объектом должна быть точка."
End If
```

- Чтобы изменить размер текстового объекта в окне Отчета, надо поменять стиль **Font** (оператором **Alter Object** с предложением **Info**).
- Чтобы изменить размер текстового объекта в окне Карты, надо поменять координаты X и Y графического объекта (с помощью оператора **Alter Object** с предложением **Geography**).
- Для того чтобы изменить высоту подписи на Карте, используйте оператор **Set Map**.

## Преобразование областей и полилиний (ломаных)

Преобразовать графический объект в область можно с помощью функции **ConvertToRegion( )**. Преобразовать графический объект в полилинию или ломаную можно с помощью функции **ConvertToPline( )**. Подробно эти функции описаны в *Справочнике MapBasic*.

## Удаление части графического объекта

Следующие операторы и функции предназначены для удаления фрагментов (частей) графических объектов:

- функция **Overlap( )** с двумя графическими объектами в качестве аргументов возвращает значение типа "Object". Результатом выполнения функции является область пересечения исходных объектов.
- функция **Erase( )** с двумя графическими объектами в качестве аргументов возвращает значение типа "Object". Результатом выполнения функции является первый объект, из которого удалены те элементы, которые входят во второй графический объект.
- **Objects Intersect** удаляет те фрагменты графического объекта (объектов), помеченного как "изменяемый", которые не являются выбранными в текущий момент.
- **Objects Erase** удаляет часть графического объекта (объектов), помеченного как "изменяемый", при этом удаляются выбранные объекты.

Оператор **Objects Erase** соответствует команде MapInfo **Объекты > Удалить часть**, а оператор **Objects Intersect** соответствует команде **Объекты > Удалить внешнюю часть**. Обе эти операции применяются к объектам со статусом "изменяемый". Этот статус устанавливается командой **Объекты > Объекты > Выбрать изменяемый объект** или оператором **Set Target** языка MapBasic. Принципы редактирования графических объектов описаны в *Руководстве пользователя MapInfo Professional*.

## Точки пересечения

Как уже говорилось ранее, добавить новые узлы к области или ломаной можно с помощью оператора **Alter Object**. Однако оператор **Alter Object** требует, чтобы Вы явно описывали каждый добавляемый узел. Чтобы добавить узлы в точках пересечения двух объектов, следует использовать оператор **Objects Overlay** или функцию **OverlayNodes( )**.

Вызовите функцию **IntersectNodes( )**, чтобы определить координаты точки или нескольких точек пересечения объектов. **IntersectNodes( )** возвратит объект-полилинию с узлом в точке пересечения объектов. Вызовите функцию **ObjectInfo( )**, для того чтобы определить число узлов полилинии. Определить координаты точек пересечения можно с помощью функций **ObjectNodeX( )** и **ObjectNodeY( )**.

## Работа с подписями

Подпись на Карте можно рассматривать как атрибут объекта Карты. Однако, MapInfo Professional до сих пор поддерживает оператор **AutoLabel**, чтобы сохранить совместимость с предыдущими версиями, в которых подписи были текстовыми объектами на косметическом слое.

### Показ подписей

Пользователь MapInfo Professional может настраивать режимы размещения подписей через окно **Управление Слоями**. Программа MapBasic может выполнять те же самые операции оператором **Set Map... Label**. Например, следующий оператор показывает подписи на первом слое:

```
Set Map Layer 1 Label Auto On Visibility On
```

### Скрытие подписей

В окне **Управление Слоями** сброс флажка **Подписывание** выключает автоматически созданные подписи, но не воздействует на измененные подписи (подписи, которые были добавлены или изменялись пользователем). Следующий оператор MapBasic имеет тот же самый эффект:

```
Set Map Layer 1 Label Auto Off
```



Оператор **Set Map... Auto Off** выключает автоматически созданные подписи, но не воздействует на измененные подписи (подписи, которые были добавлены или изменялись пользователем). Следующий оператор временно скрывает все подписи для слоя – и заданные по умолчанию подписи, и измененные подписи:

---

```
Set Map Layer 1 Label Visibility Off
```

Пользователь MapInfo Professional может восстанавливать подписи слоя к их заданному по умолчанию состоянию, выполняя команду **Карта > Восстановить подписи**. Следующий оператор MapBasic имеет тот же самый эффект:

```
Set Map Layer 1 Default Zoom
```

## Редактирование подписей

Пользователь MapInfo Professional может редактировать подписи в интерактивном режиме. Например, чтобы скрыть подпись, укажите на подпись мышкой, чтобы выбрать ее, и нажмите клавишу **DEL**. Подпись можно также переместить мышкой.

Чтобы изменять подписи, измененные пользователем, через MapBasic, используйте оператор **Set Map ... Label**, который включает одно или большее количество предложений **Object**. Например, следующий оператор скрывает две подписи в окне Карты:

```
Set Map Layer 1 Label  
    Object 1 Visibility Off  
    Object 3 Visibility Off
```

Для каждой подписи Вы можете включать предложение **Object**. В этом примере, Object 1 относится к подписи для первой строки таблицы, и Object 3 – к подписи для третьей строки таблицы. Чтобы сохранить измененные подписи, сохраните файл Рабочего Набора оператором **Save Workspace**.



Упаковка таблицы может сделать неверными отредактированные подписи, предварительно сохраненные в Рабочем наборе. Когда Вы сохраняете отредактированные подписи, сохраняя Рабочий набор, подписи представляются как операторы **Set Map ... Object**. Каждое предложение **Object** относится к номеру строки в таблице. Если таблица содержит строки, которые были помечены как удаленные, то упаковка таблицы устраняет удаленные строки.

Другими словами, если Вы упаковываете таблицу и затем загружаете Рабочий набор, любые отредактированные подписи, содержащиеся в нем, могут быть неправильны. Следовательно, если Вы предполагаете упаковывать таблицу, Вы должны делать это *перед* созданием собственных подписей.

Если удаленные строки в таблице находятся в самом конце таблицы (то есть внизу окна Списка), то все будет в порядке.

## Запрос к содержимому подписей

Запрос к подписям в окне Карты состоит из двух шагов:

1. Инициализация внутреннего указателя подписей MapBasic с помощью вызова **LabelFindFirst( )**, **LabelFindByID( )** или **LabelFindNext( )**.
2. Вызова **Labelinfo( )** для запроса “текущей” подписи. Пример кода смотрите в разделе **Labelinfo( )** электронной справки MapBasic или в программе-примере LABELER.MB (“Подписи”).

## Другие примеры применения оператора Set Map

Чтобы увидеть синтаксис MapBasic, который соответствует диалогу **Управление слоями**, сделайте следующее:

1. Откройте окно MapBasic.
2. Выберите окно Карты.

3. Выполните команду **Карта > Управление слоями**.

4. Выберите в диалоге нужные режимы

MapInfo Professional отобразит оператор **Set Map** в окне MapBasic. Вы можете скопировать текст из окна MapBasic и вставить его в Вашу программу.

Чтобы увидеть синтаксис MapBasic, который соответствует редактированию индивидуальной подписи, сделайте следующее:

1. Измените подписи в окне Карты (удалите или измените подпись, измените шрифт и т.п.).
2. Сохраните Рабочий набор.
3. Просмотрите файл Рабочего набора в текстовом редакторе, типа MapBasic редактора. Отредактируйте в рабочем наборе строки с оператором **Set Map ... Layer ... Label ... Object** ).

Разница между подписями и текстовыми объектами

Следующая таблица показывает различия между текстовыми объектами и подписями.

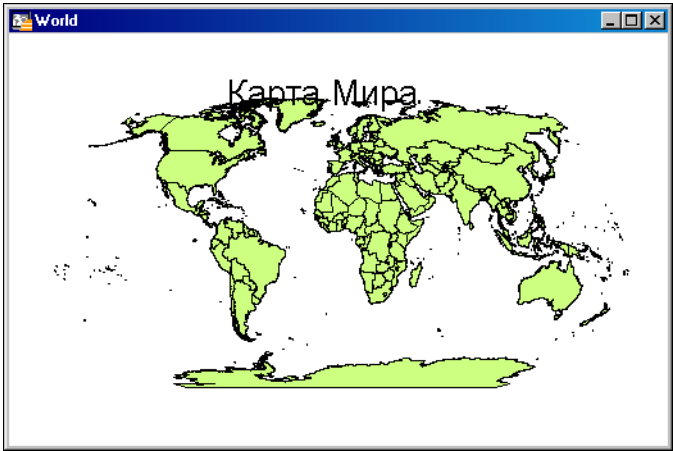
	Текстовые объекты	Подписи
Операторы MapBasic для создания текста:	<b>AutoLabel, Create Text, CreateText( )</b>	<b>Set Map</b>
Операторы MapBasic для изменения текста:	<b>Alter Object</b>	<b>Set Map</b>
Функции MapBasic, с помощью которых можно запрашивать текстовые объекты (например, чтобы определить цвет):	<b>ObjectInfo( ), ObjectGeography( )</b>	<b>LabelFindByID( ), LabelFindFirst( ), LabelFindNext( ), Labelinfo( )</b>
Операторы MapBasic для выбора текстовых объектов:	<b>Select</b>	С помощью программ на MapBasic нельзя выбирать подписи.
Сохранение текста на Карте:	Текстовые объекты можно хранить в таблицах с геоинформацией.	Подписи сохраняются только в Рабочих наборах.
Сохранение текста в Отчете:	Текстовые объекты, созданные в отчете, можно сохранить в Рабочем наборе.	Не применяется. Подписи не появятся в Отчете (кроме случая, когда Карта включена в состав Отчета).

	Текстовые объекты	Подписи
Настройки высоты текста:	Высота текста зависит от масштаба Карты. Размер текста растёт при приближении и уменьшается при удалении.	Размер текста подписи зависит от шрифта. Приближение или удаление не меняет размер текста подписей.
Преобразования текста и подписей:	Не применяется. Для текстового объекта не существует функции Map-Basic с помощью которой можно создать подпись.	Для подписи, функция <b>Labelinfo( )</b> возвращает текстовый объект из подписи. Пример использования в программе LABELER.MBX ("Подписи").

Когда Вы создаете подпись, Вы определяете точку привязки подписи. Например, если Вы рассматриваете Карту таблицы WORLD, оператор создает подпись, которая становится заголовком:

```
Set Map Layer 1 Label Object 1
Visibility On ' показать подпись к этой записи
Anchor (0, 85) ' привяжите подпись к этим координатам (x,y)
Text "Карта Мира" ' введите текст подписи
Position Center ' установите позицию по отношению к точке привязки
Font("Arial",289,20,0) ' установите стиль шрифта (20-пунктов, и т.д.)
```

Полученную подпись можно использовать в качестве заголовка Карты.



Размещение текста на Карте проще всего провести, используя механизм подписывания. Для этого Вы можете создать таблицу, единственным назначением которой является хранение подписей:

1. Создайте таблицу (оператором **Create Table**), которая содержит строковый столбец. Сделайте этот столбец достаточно широким, чтобы сохранить текст, который Вы хотите изобразить на Карте. Сделайте таблицу “картографической” (оператором **Create Map**).
2. Добавьте таблицу к Вашему окну Карты (оператором **Add Map**). Используйте оператор **Set Map**, чтобы установить параметры подписей таблицы (шрифт, режимы показа и т.д.).
3. Если Вы хотите добавить текст к Карте, вставьте точку или линию в таблицу, используя невидимый стиль символа (номер 31 в таблице символов) или невидимый стиль пера (шаблон 1). Объект будет невидим, но подпись появится.



Используйте линии, если хотите, чтобы текст можно было поворачивать. Программа из комплекта поставки COGOLINE.MB показывает, как создать объект-линию с данным углом.

---

Обратите внимание на то, что Вы не должны использовать оператор **Set Map... Object**, чтобы указать положение каждой подписи. Вы можете отображать подписи в позициях, заданных по умолчанию. Затем, если Вы хотите переместить подпись, переместите объект, которому она соответствует.

## Координаты и единицы измерения

Программа MapBasic может работать с одной определенной системой координат в каждый момент времени. MapBasic использует географические координаты, координаты плана и координаты Отчета. Возможность работы только с одной из этих систем в каждый момент времени диктует следующие требования к программисту:

- Прежде чем создавать, изменять или выбирать объекты из мировой Карты, убедитесь, что MapBasic работает в мировых координатах. Этот режим является стандартным. Поэтому во многих программах на MapBasic Вам не придется вспоминать о системе координат.
- Прежде чем создавать, изменять или выбирать объекты с плана, убедитесь, что MapBasic работает в координатах плана. Для этого следует выполнить оператор **Set CoordSys Non-earth**.
- Прежде чем создавать, изменять или выбирать объекты в окне Отчета, убедитесь, что MapBasic работает в координатах Отчета. Для этого следует выполнить оператор **Set CoordSys Layout**.

Каждая программа на языке MapBasic содержит установку параметра **CoordSys**, который определяет текущую систему координат в приложении. Стандартная система координат – мировая (широта, долгота). По умолчанию, все программы MapBasic работают с объектами мировых Карт, большинство таблиц MapInfo относится также к этой категории. Если же программа на языке MapBasic должна работать с объектами окна Отчета, Вам следует выполнить оператор **Set CoordSys Layout** вида:

```
Set CoordSys Layout Units "in"
```

С помощью оператора **Set CoordSys Layout** можно задать единицы измерения отчета, например, "in" (дюймы). От этих установок зависит, какую систему координат Отчета использует MapBasic. Чтобы работать в сантиметрах или миллиметрах, задайте в качестве единиц измерения "cm" или "mm", соответственно. Следующий фрагмент программы открывает окно Отчета, затем помещает в отчет заголовок, создавая текстовый объект. Поскольку объект создается в окне Отчета, оператору **Create Text** должен предшествовать оператор **Set CoordSys Layout**.

```
Include "mapbasic.def"

Dim win_num As Integer
Layout
win_num = FrontWindow()
Set CoordSys Layout Units "in"

Create Text
  Into Window win_num
  "Здесь располагается заголовок окна"
  (3.0, 0.5) (5.4, 1.0)
  Font MakeFont("Helvetica", 1, 24, BLUE, WHITE)
```

В данном примере используется система координат Отчета и дюймы в качестве единиц измерения. Следовательно, все значения в операторе **Create Text** указаны в дюймах. После того, как оператором **Set CoordSys** вводится новая система координат, является текущей до тех пор, пока не будет еще раз явно изменена. Каждое приложение MapBasic имеет свои установки системы координат. Это позволяет каждому приложению выполнять оператор **Set CoordSys**, не влияя на системы координат других выполняющихся приложений.

Система координат MapBasic не зависит от системы координат, использующейся в окнах Карт MapInfo. Стандартная система координат – широта/долгота (NAD 1927) (в десятичных градусах, а не в градусах, минутах и секундах.).

Все координаты в операторах и функциях MapBasic должны быть указаны в виде широты и долготы, если только не была изменена система координат MapBasic с помощью оператора **Set CoordSys**. Например, функция **Centroidx( )** возвращает долготу центраида объекта в стандартном виде (в десятичных градусах), даже если объект хранится в таблице или показывается в окне, которое имеет другую систему координат. Например, выборка, полученная в результате выполнения следующего примера, содержит значения: WY 107 554 43, широта и долгота центраида штата Вайоминг:

```
Select state, CentroidX(obj), CentroidY(obj)
  From states
  Where state = "WY"
```

Результатом же выполнения следующего оператора будет: WY 934612.97 2279518.38; координаты в проекции Алберса.

```
Set CoordSys Earth Projection 9, 62, "m", -96, 23, 29.5, 45.5, 0, 0
Select state, CentroidX(obj), CentroidY(obj)
  From states
  Where state = "WY"
```

Чтобы вернуться к стандартной системе координат MapBasic, выполните оператор:



```
Set CoordSys Earth
```

## Единицы измерения

Единицы площади, такие, как:

- единицы измерения площадей, например, квадратные километры и акры. Полный список единиц измерения площадей, поддерживаемых в языке MapBasic, приведен в главе **Set Area Units** *Справочника MapBasic*. Функция **Area( )** и другие функции измерения возвращают результаты в тех единицах, которые приняты в Вашей программе.
- Единицы расстояний, такие, как километры или мили. Полный список единиц измерения расстояний, поддерживаемых в языке MapBasic, приведен в главе **Set Area Units** *Справочника MapBasic*.
- Единицы макета Отчета, такие как дюймы и сантиметры. Например, при выполнении оператора **Set Window** для изменения высоты и ширины окна Карты Вы указываете новый размер окна на экране в единицах макета, например, в дюймах.

В любой момент сеанса работы с MapInfo действуют текущие единицы измерения расстояния, площади, а также текущие единицы макета. Стандартные единицы – это, соответственно, мили, квадратные мили и дюймы. Лучше всего показано, как работают текущие установки, на следующем примере. Оператор создает окружность:

```
obj_var = CreateCircle(x, y, 5)
```

Поскольку стандартные единицы измерения расстояний в MapBasic – это мили, то окружность будет иметь радиус пять миль. Однако, если Вы измените текущие единицы измерения расстояний с помощью оператора **Set Distance Units**, то значение радиуса (5) изменит смысл. Так, в следующем фрагменте создается окружность радиусом 5 километров:

```
Set Distance Units "km"  
obj_var = CreateCircle(x, y, 5)
```

Чтобы вернуть стандартные единицы измерения площадей, выполните оператор **Set Area Units**, а для единиц макета Отчета – оператор **Set Paper Units** соответственно.

## Географические запросы

Программы на языке MapBasic могут выполнять сложные запросы к данным на основании не только числовых, но и графических данных. Например, в программе с помощью оператора **Add Column** можно посчитать суммы и средние значения по областям, в зависимости от того, какие объекты из других слоев Карты попадают внутрь области или пересекаются с ней.

Чтобы понять, как MapBasic и MapInfo выполняют операции с данными при географическом анализе, Вам следует представлять, как программы MapBasic работают с таблицами. Сведения о таблицах содержит глава **Работа с таблицами**.

## Работа с операторами географического анализа

MapBasic не позволяет использовать оператор равенства (=) для логического сравнения географических объектов (If object\_a = object\_b). В то же время MapBasic поддерживает специальные географические операторы, которые позволяют выяснять

взаимное расположение объектов в пространстве. Операторы языка MapBasic **Contains**, **Within** и **Intersects**, а также предложения **Part** и **Entire** позволяют сравнивать географические объекты по аналогии со сравнением числовых величин.

Ниже приведен пример географического сравнения в операторе **If...Then**:

```
If Parcel_Object Within Residential_Zone_Obj Then
    Note "Ваша собственность расположена в жилом районе."
End If
```

А вот пример использования географического сравнения в операторе **Select**:

```
Select * From wetlands
Where obj Contains Part myproject
```

По крайней мере один из объектов, используемых в условиях **Within** и **Contains**, должен быть графическим площадным объектом: многоугольником (областью), эллипсом, прямоугольником или скругленным прямоугольником.

Какое условие использовать: **Within** или **Contains**, зависит от порядка объектов в выражении. Существуют следующие правила:

- **Within** используется для проверки, находится ли первый из объектов внутри второго
- **Contains** используется для проверки, содержит ли первый из объектов второй объект внутри себя

Например, в случае сравнения точки и области отношения вложения следующие:

- точка находится внутри (**Within**) области
- область содержит (**Contains**) точку

Приведем пример выбора штатов, содержащих точки – места расположения дистрибьюторов:

```
Select * From states
Where obj Contains distribution_ctr
```

Следующий оператор выбирает все свалки на территории указанного района:

```
Select * From landfill
Where obj Within county_obj
```

Оператор **Within** и оператор **Contains** проверяют, находится ли центроид объекта внутри другого объекта. Чтобы проверить, лежит ли весь объект целиком внутри другого объекта, следует употреблять предложение **Entire(ly)**. Предложение **Part(ly)** используется, чтобы определить, лежит ли хотя бы некоторая часть графического объекта внутри заданного.

Следующий оператор выбирает все участки шоссе, которые хотя бы частично проходят через территорию заданной области:

```
Select * From highway
Where obj Partly Within countyobj
```

Оператор **Partly Within** проверяет, находится ли хотя бы часть объекта, указанного первым, внутри второго объекта (соприкасаются ли они хотя бы в одной точке). С помощью оператора **Entirely Within** можно проверить, лежит ли некоторый графический объект внутри другого

графического объекта. Поскольку проверка всех сегментов графического объекта требует большего количества операций, чем проверка положения одного только центроида, проверка условий с предложением **Partly** или **Entirely** работает более медленно.

Оператор **Intersects** можно использовать со всеми типами графических объектов. Если два графических объекта имеют общую точку, соприкасаются или один из них лежит внутри другого, то считается, что эти объекты пересекаются. Области, соприкасающиеся в единственной точке, пересекаются. Точка, расположенная на узле ломаной, пересекается с ломаной, а точка внутри области пересекается с этой областью.

В таблице содержится сводка географических операторов языка MapBasic:

Оператор	Применение	Возвращает TRUE, если:
<b>Contains</b>	<code>objectA Contains objectB</code>	центроид второго объекта находится внутри первого объекта
<b>Contains Part</b>	<code>objectA Contains Part objectB</code>	Объект А содержит некоторую часть объекта В
<b>Contains Entire</b>	<code>objectA Contains Entire objectB</code>	Объект А содержит весь объект В
<b>Within</b>	<code>objectA Within objectB</code>	Центроид объекта А лежит внутри объекта В
<b>Partly Within</b>	<code>objectA Partly Within objectB</code>	Часть объекта А лежит внутри объекта В
<b>Entirely Within</b>	<code>objectA Entirely Within objectB</code>	Объект А находится полностью внутри объекта В
<b>Intersects</b>	<code>objectA Intersects objectB</code>	Два объекта пересекаются хотя бы в одной точке

Запросы к графическим объектам в таблицах

Географические операторы и функции языка MapBasic могут использоваться в запросах к таблицам (которые имеют колонку "Object"). Такие запросы очень похожи на все другие запросы, разница лишь в том, что не существует графических объектов констант. Вместо них в запросах для анализа графических объектов обычно используются географические операторы и функции (такие как **Entirely Within**).

В следующем примере функция **ObjectLen( )** используется для нахождения всех участков кабеля, которые длиннее 300 метров:

```
Select *
  From cable
  Where ObjectLen(obj, "m") > 300
```

Ниже подсчитывается общая площадь болот в штате Индиана:

```
Select Sum(Area(obj,"sq mi"))
  From wetlands
  Where obj Within (Select obj From states Where state = "IN")
```

Далее выбираются все хранилища в пределах одного километра от точки с долготой "lon" и широтой "lat":

```
Set Distance Units "km"
Select * From tanks Where obj Within
  CreateCircle(lon,lat, 1)
```

В следующем примере создается выборка данных о работниках с указанием, на каком расстоянии от офиса они проживают (начиная с тех, кто живет дальше):

```
Select
  Name, Distance(Centroidx(obj), Centroidy(obj),
                 office_lon, office_lat, "km")
  From employee
  Order By 2 Desc
```

## Географические SQL-запросы с промежуточными выборками (подзапросами)

MapBasic предоставляет возможность осуществлять выборку графических объектов из одной таблицы в зависимости от их расположения по отношению к объектам из другой таблицы. Например, можно составить запрос к таблице врачей, чтобы узнать, кто из них проживает в округе Мерион штата Индиана. Данные о врачах находятся в одной таблице, Doctors, а об округах – в другой, Counties.

С одной стороны, можно выбрать округ из таблицы округов, скопировать его значение в переменную, а затем выполнить запрос к таблице врачей с использованием этой переменной. Выглядеть это будет примерно так:

```
Dim mycounty As Object
Select *
  From counties
  Where name="Marion" and state="IN"
Fetch First From selection
mycounty = selection.obj
Select *
  From doctors
  Where obj Within mycounty
```

Если же в предложении **Where** использовать подзапрос, т.е. промежуточную выборку, вместо переменной "mycounty", то тот же результат можно получить, использовав меньшее число операторов:

```
Select *
  From doctors
  Where obj Within
    (Select obj From counties Where name="Marion" And state="IN")
```

Отметим, что подзапрос (второй оператор **Select**, заключенный в скобки) возвращает таблицу, содержащую единственную колонку и единственную строку – графический объект, соответствующий округу Marion штата Индиана ("IN"). MapInfo Professional проверяет каждую

запись таблицы врачей (DOCTORS), чтобы определить, не находится ли объект в округе Marion. В данном примере промежуточная выборка выполняет ту же роль, что и переменная в предыдущем примере ("myscounty"), поскольку она дает значение элементу выражения.

Чтобы гарантировать то, что промежуточная выборка (результат подзапроса) содержит только колонку "Object", в предложении **Select** указано имя только одной колонки – "obj". Данный оператор не будет правильно работать, если в промежуточную выборку помещать все колонки или если помещать колонку, отличающуюся от колонки "Object".

Если промежуточная выборка содержит несколько записей, используется оператор **Any( )**. В следующем примере показано, как обрабатывать промежуточную выборку из нескольких записей с помощью **Any( )**. Здесь выбираются все врачи в районах со средним доходом жителей менее \$15000. Место проживания врача в данном случае надо сравнить с каждой записью промежуточной выборки.

```
Select *
  From doctors
  Where obj Within
        Any (Select obj From counties Where inc_pcap < 15000)
```

Изменим порядок в операторе **Select**: будем выбирать районы, а не врачей. Вот пример, выбирающий районы, где проживают специалисты неврологи:

```
Select *
  From counties
  Where obj Contains
        (Select obj From doctors Where specialty = "Neurology")
```

Следующий пример находит все штаты, граничащие со штатом Небраска:

```
Select *
  From states
  Where obj Intersects (Select obj From states Where state = "NE")
```

## Объединения таблиц по географическим критериям

Процесс объединения таблиц заключается в том, что две таблицы связываются друг с другом путем сопоставления записей. Результатом объединения является таблица, содержащая колонки из обеих исходных таблиц и имеющая столько записей, сколько имелось сопоставленных пар в двух таблицах. MapBasic расширяет реляционную концепцию объединения таблиц, допуская использование географического критерия объединения. Например, при объединении таблицы демографических данных с Картой областей, результирующая таблица может содержать всю информацию Карты областей вместе с демографическими данными для каждой области.

MapInfo Professional позволяет задавать географические условия объединения таблиц. Например, вместо того, чтобы сравнивать в двух таблицах числовые параметры ID, можно объединять таблицы по результатам сравнения графических объектов: какие из объектов первой таблицы содержат объекты второй таблицы. Такой способ особенно удобен, когда нет числовой колонки, по которой можно было бы сопоставлять записи. Например, можно объединить таблицу строительных проектов с таблицей данных по районам (считая, что в

таблице проектов нет информации о том, в каких районах проекты осуществляются). Объединение таблиц может потребоваться для того, чтобы внести в таблицу проектов данные о районах. Для этого следует выполнить оператор SQL **Select** вида:

```
Select *  
  From projects, congdist  
 Where projects.obj Within congdist.obj
```

После географического объединения таблиц можно выполнить следующий оператор **Update**, чтобы ввести названия районов (из колонки "name") в таблицу проектов (колонку "cd"):

```
Update Selection Set cd = name
```

Полученная таблица проектов теперь содержит названия районов, в которых осуществляются проекты. Следующий пример подсчитывает общие затраты по проектам в рамках каждого из районов:

```
Select congdist.name, sum(project.amt)  
  From congdist, project  
 Where congdist.obj Contains project.obj  
 Group By 1
```

Поскольку порядок упоминания таблиц в предложении **Where** изменен, вместо условия **Within** используется **Contains**.

## Пропорциональное обобщение данных

Оператор **Add Column** может использоваться для выполнения сложных операций над областями (многоугольниками) при проведении пропорционального обобщения данных, в зависимости от того, каким образом расположены графические объекты одной из таблиц по отношению к объектам другой таблицы. Предположим, например, что имеется таблица границ городов и таблица, отражающая риск затопления территорий. Некоторые города частично или полностью попадают в зоны возможного затопления, другие же полностью лежат вне таких зон. С помощью оператора **Add Column** можно выделить демографическую информацию из таблицы городов, а затем использовать эту информацию при вычислении статистики для зон возможного затопления. Подробно оператор **Add Column** описан в *Справочнике MapBasic*.

# Особенности MapBasic в среде Microsoft Windows

В этой главе описывается, как в программе на MapBasic можно использовать возможности, предоставляемые средой Windows.


## В этой главе:

- ♦ Объявление и использование динамических библиотек . . .200
- ♦ Создание пиктограмм на кнопках и новых курсоров . . . . .205
- ♦ Связь между приложениями с использованием DDE . . . . .207
- ♦ Добавление Справочной системы к Вашему приложению .214

## Объявление и использование динамических библиотек

Динамические библиотеки Windows (Dynamic Link Library, DLL) – это файлы, содержащие выполняемые процедуры и другие ресурсы. Можно использовать библиотеки DLL в качестве источника внешних процедур, и вызывать эти процедуры из MapBasic-программ. Обращаться к DLL-библиотекам можно из программы MapBasic с помощью оператора **Call**, аналогично вызову оператора **Call** к процедуре на MapBasic. Многие DLL-библиотеки поставляются вместе с программами, а также отдельно на коммерческой основе. Каждая обычно снабжается документацией, описывающей процедуры и их параметры.

---

 Если из программы MapBasic вызываются внешние библиотеки DLL, то они должны быть доступны при исполнении программы. Другими словами, если Вы предлагаете другим пользователям использовать скомпилированную программу (MBX-файл), то необходимо обеспечить пользователей и внешними библиотеками DLL, которые могут быть вызваны Вашей MBX-программой.

---

Подробно DLL-файлы описаны в документации к Windows Software Developer's Kit (SDK), а также в книгах независимых авторов.

### Объявление внешней библиотеки

До того как MapBasic-программа сможет вызывать DLL-процедуры, необходимо объявить DLL-оператором **Declare** (точно также, как объявляются подпрограммы в коде программы MapBasic). Оператором **Declare** объявляются имя DLL-файла и имя процедуры в этой библиотеке.

```
Declare Sub my_routine Lib "C:\lib\mylib.dll"  
    (ByVal x As Integer, ByVal y As Integer)
```

Если в операторе **Declare** указан явный адрес библиотеки (например, "C:\lib\mylib.dll"), MapInfo Professional будет пытаться загружать DLL из этого места. Если этого файла там нет, то MapInfo его не загрузит, что повлечет ошибку. Если оператор **Declare** не задает явного пути к DLL-библиотеке (например, "mylib.dll"), то MapInfo будет искать его по следующим правилам:

1. если DLL находится в том же каталоге, что и MBX-файл, то MapInfo Professional загружает DLL, иначе
2. если DLL находится в том же каталоге, что и MapInfo, то MapInfo Professional загружает DLL, иначе
3. если DLL находится в каталоге WINDOWS\SYSTEM, то MapInfo Professional загружает DLL, иначе
4. если DLL находится в каталоге WINDOWS, то MapInfo Professional загружает DLL, иначе
5. MapInfo Professional проводит поиск по каталогам, заданным в системной переменной PATH.

Поиск растровых файлов для иконок и курсоров MapInfo Professional проводит по тем же правилам.



## Передача параметров

Многие DLL-процедуры требуют передачи параметров, как в приведенном выше примере оператора **Declare**, где DLL-процедуре передается два параметра.

MapBasic может передавать параметры двумя способами: значением (by value), при этом MapInfo помещает аргумент в стек; и ссылкой (by reference), при этом MapInfo помещает в стек адрес переменной MapBasic; в последнем случае DLL может это значение изменить. Более подробно о различиях в передаче параметров по ссылке или по значению, можно прочитать в главе **Основы MapBasic**.

Ключевое слово **ByVal** оператора **Declare** задает передачу параметра значением (by value). Вы должны явно задать слово **ByVal**, если нужно передать соответствующий параметр значением, иначе он будет передан ссылкой.

Следующие типы данных MapBasic нельзя передавать значением: массивы, новые типы данных (структуры) и псевдонимы. Строки переменной длины можно передавать значением, но только в том случае, если DLL обрабатывает этот параметр как структуру. См. ниже раздел **Строковые аргументы**.

## Вызов стандартных библиотек

В следующем примере показано, как программа на языке MapBasic может обращаться к процедуре "MessageBeep" из стандартной Windows-библиотеки "User".

```
Declare Sub MessageBeep Lib "user"  
    (ByVal x As SmallInt)
```

Обратите внимание на то, что оператор **Declare** обращается к стандартной библиотеке как к "user", а не "user.dll". Так можно обращаться еще только к двум другим библиотекам: "GDI" и "Kernel".

После объявления DLL-процедуры оператором **Declare Sub** ее можно вызывать оператором **Call**:

```
Call MessageBeep(1)
```

## Вызов DLL-процедур с помощью ключевого слова Alias

Некоторые имена DLL-процедур не могут быть использованы в MapBasic. Например, имя DLL-процедуры может конфликтовать с каким-нибудь стандартным словом языка MapBasic. Избежать конфликта можно, задав с помощью слова **Alias** синоним имени DLL процедуры.

В следующем примере показано, как задается синоним "Beeper" для имени процедуры "MessageBeep" из библиотеки "User":

```
Declare Sub Beeper Lib "user" Alias "MessageBeep"  
    (ByVal x As SmallInt)
```

Call Beeper(1)



Если Вы задаете DLL-процедуре имя-синоним – "Beeper", то синоним должен следовать сразу за словом **Sub**, а настоящее имя процедуры указывается после слова **Alias**.

---

### Строковые аргументы

При обращении к DLL-библиотекам программа MapBasic может передавать строки переменной длины ссылкой. Если вы создаете DLL-процедуру на языке C и желаете, чтобы MapBasic передавал в нее строку переменной длины, то определяйте соответствующий аргумент C-программы, как `char *`.



Когда MapBasic передает строчный аргумент ссылкой, DLL процедура может изменять значение строчной переменной. Однако, DLL-процедура не должна увеличивать размер строки, даже если она объявлена в MapBasic строкой переменной длины.

---

Программа MapBasic может передавать строку постоянной длины как ссылкой, так и значением. Однако, если Вы передаете аргумент-строку значением, то DLL-процедура может разобрать строку как структуру. Например, MapBasic передает значением 20-разрядную строку, а DLL процедура преобразует ее в пять четырехбайтовых целых чисел.

Когда MapBasic передает строку DLL-процедуре, MapInfo Professional автоматически добавляет символ ANSI-ноль в конец строки, независимо от того, фиксирована ли длина строки или нет.

Если DLL-процедура должна изменить Вашу строку-аргумент, то позаботьтесь о том, чтобы длина этой строки могла вместить все возможные изменения. Другими словами, предусматривайте в Вашей MapBasic-программе, строки достаточной длины, чтобы в них могли поместиться длинные значения передаваемых переменных.

Например, если возвращаемая (т.е. измененная) строка может достигать размера в 100 символов, то в программе MapBasic нужно задать этой строке длину в 100 символов, прежде чем передавать ее в DLL-процедуру. Функция MapBasic **String\$( )** позволяет легко создавать строки любой длины. Вы также можете объявить строку в начале программы (например, `Dim stringvar As String * 100`). MapBasic автоматически добавляет пробелы в конец строки, тем самым, сохраняя ее длину.

### Аргументы массива

MapBasic позволяет Вам передавать в качестве аргументов DLL процедур массивы так же, как они передаются в Sub-процедуры MapBasic. Вы можете передавать массив в DLL, который воспринимает массив как аргумент, задавая имя массива с пустыми круглыми скобками.

## Типы данных, определенные пользователем

Некоторым DLL-процедурам можно передавать сложные типы, заданные пользователем с помощью оператора **Туре**. MapBasic передает адрес первого элемента, подразумевая, что остальные элементы последовательно размещены в памяти.

---

**i** Чтобы DLL-процедура работала со сложными типами, ее нужно компилировать с заданием упаковки структур ("structure packing") однобайтовой границей – самой плотной. Например, при использовании компилятора Microsoft C, можно использовать аргумент `/Zp1`, чтобы предусмотреть самую плотную упаковку.

---

## Логические аргументы

Невозможно передать логическое значение переменной MapBasic в DLL.

## Уникальные номера (дескрипторы, handles)

Уникальные внутренние номера определяются самой операционной системой и используются для ссылки на сложные объекты. Стандартные библиотеки среды Windows используют уникальные номера окон (hWnd), контекстов устройств (hDC) и т.д. Уникальные внутренние номера можно использовать только как идентификаторы; участвовать в каких-либо вычислениях они не могут.

Если нужно передать уникальный номер как аргумент, то его нужно объявить как `ByVal Integer`.

DLL-функции, возвращающие уникальные номера, должны быть объявлять тип возвращаемого значения как `Integer`.

## Пример: вызов процедуры из библиотеки KERNEL

Следующий пример содержит вызов DLL-функции. Используется стандартная DLL-библиотека Windows KERNEL. Вызываемая процедура считывает установки из стартового файла Windows, WIN.INI.

```
Declare Sub Main
' Объявляется об использовании стандартной Windows-библиотеки
' "KERNEL".
Declare Function GetProfileString Lib "kernel" (
    lpszSection As String,
    lpszEntry As String,
    lpszDefault As String,
    lpszReturnBuffer As String,
    ByVal cbReturnBuffer As Smallint)
    As Smallint

Sub Main
    Dim sSection, sEntry, sDefault, sReturn As String
    Dim iReturn As Smallint

    ' чтение установки "sCountry"
    ' из секции "[intl]" файла WIN.INI.
```

```
sReturn = String$(256, " ")
sSection = "intl"
sEntry = "sCountry"
sDefault = "Not Found"
iReturn = GetProfileString(sSection, sEntry,
    sDefault, sReturn, 256)

' в этом месте строка sReturn содержит имя страны
' (например, "Russia")
Note "[" + sSection + "]" + chr$(10) + sEntry + "=" + sReturn
End Sub
```

Оператор **Declare Function** налаживает связь с библиотекой “KERNEL”. Обратите внимание на то, что библиотека “KERNEL” на самом деле находится в файле KRNL386.EXE. Windows сама будет использовать требующуюся библиотеку, если программа будет ссылаться на “kernel”. Поскольку эта библиотека является стандартной компонентой системы, Windows API в этом случае сам подставляет нужный файл. Однако, если Вы создаете свою DLL библиотеку, Вы должны точно задавать имя файла библиотеки в операторах **функции Declare**.

Если в DLL хранятся иконки и/или курсоры панелей инструментов, можно использовать такую же технику – вызов `SystemInfo(SYS_INFO_MIPLATFORM)` – чтобы определить, какую DLL следует использовать. Однако синтаксис MapBasic немного отличается: вместо оператора **Declare**, ресурсы DLL (растровые иконки и курсоры) объявляются параметром **File** оператора **Create ButtonPad**, как показано в следующем примере.

```
Declare Sub Main
Declare Function getDLLname() As String
Declare Sub DoIt

Sub Main
    Dim s_dllname As String

    s_dllname = getDLLname()

    Create ButtonPad "Новые кнопки" As
        ToolButton Calling doit
        Icon 134 File s_dllname
        Cursor 136 File s_dllname
End Sub

Function getDLLname() As String
    If SystemInfo(SYS_INFO_MIPLATFORM) = MIPLATFORM_WIN32 Then
        getDLLname = "..\icons\Test32.DLL"
    Else
        getDLLname = "..\icons\Test16.DLL"
    End If
End Function

Sub DoIt
    ' эта процедура будет вызвана, если
    ' используется самостоятельно разработанная кнопка.
End Sub
```

Смотрите также раздел [Создание пиктограмм на кнопках и новых курсоров на стр. 205](#), в котором обсуждается создание иконок панелей инструментов.

## Советы по работе с DLL

Следующие советы могут пригодиться, если при работе с Вашими DLL-библиотеками возникают ошибки.

- Если Вы используете язык C++ для написания DLL, учтите, что компилятор этого языка автоматически добавляет к названию функций дополнительные символы. Вам следует указать компилятору не производить эту операцию над именами экспортируемых функций (как в языке C).
- Компилятор Microsoft C (32-битный) поддерживает три типа передачи параметров: «стандартный» (ключевое слово `__stdcall`), «C» (ключевое слово `__cdecl`) «быстрый вызов» (ключевое слово `__fastcall`). Вызываемые из MapBasic процедуры не должны использовать третье соглашение.
- Если возникли проблемы при передаче созданных Вами типов данных (структур), проверьте, что структуры данных C «упакованы» (выровнены по границе байта).
- MapBasic может передавать аргументы как ссылкой (по умолчанию), так значением. Однако, передача аргументов значением для разных компиляторов несколько отличается (например, обработка значений типа "double", характерных для языка C). Может оказаться более надежным передавать аргумент ссылкой; большинство компиляторов, присутствующих на рынке, надежно обрабатывают в этом случае адреса и меньше оснований для неожиданностей.
- Следует придерживаться следующего правила: каждая функция из DLL сама выделяет и освобождает для себя память.
- Важно, чтобы объявление в программе на MapBasic оператора **Declare** было корректным; передаваемые параметры должны быть описаны в соответствии с «устройством» вызываемой из DLL функции. Если процедура DLL требует получения аргументов по значению, а Ваша программа передает аргументы ссылкой, процедура может не сработать или выдать ложные данные.

## Создание пиктограмм на кнопках и новых курсоров

Средства языка MapBasic позволяют управлять Инструментальными панелями – важным элементом пользовательского интерфейса MapInfo Professional. Подробно об этом рассказывает [Создание интерфейса пользователя](#).

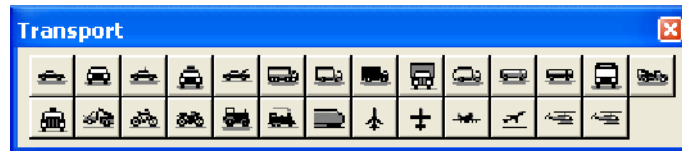
Маленькая пиктограмма (иконка) появится на каждой кнопке. Вы можете создавать свои кнопки и на каждую кнопку можно поместить пиктограмму. В каждой вычислительной среде поддерживается своя процедура создания кнопок и пиктограмм. В среде Windows пиктограммы помещаются как BMP-ресурсы в DLL файлы

Программа MapBasic может также создать свой курсор (изображение указателя мыши для окна Карты или Отчета). В этом разделе описано, как создавать курсоры в MapInfo для Windows.

## Использование стандартных пиктограмм (иконок)

Прежде чем заниматься созданием новых пиктограмм, познакомимся поближе со встроенными в MapInfo Professional. Начиная с версии 4.0, MapInfo Professional содержит большое количество различных пиктограмм, что должно облегчить Вам создание своих панелей инструментов.

Чтобы посмотреть на встроенные пиктограммы, запустите программу "Icon Sampler" (ICON-DEMO.MBX). Вот одна из создаваемых этой программой инструментальных панелей.



Каждая такая пиктограмма имеет свой числовой код. Список кодов есть в файле ICONS.DEF. Программа ICONDEMO.MBX покажет этот код, если задержать на некоторое время указатель мышки над кнопкой.

Если ни одна из включенных в состав MapInfo Professional иконок не подходит для Вашего приложения, может потребоваться разработать дополнительные иконки-пиктограммы самостоятельно, как описано ниже.

## Создание пиктограмм

Для создания пиктограмм MapInfo Professional Вам понадобится редактор ресурсов. Среда разработки MapBasic не содержит редактора ресурсов, однако MapBasic воспринимает ресурсы, созданные внешними редакторами ресурсов. Например, пиктограммы можно создавать программой AppStudio (редактор ресурсов из пакета Microsoft Visual C).

В среде Windows новые пиктограммы помещаются в файл DLL. Перед тем как разрабатывать свои пиктограммы, нужно создать DLL-файл или воспользоваться имеющимся. Такой DLL-файл может быть "пустышкой" (т.е. файл, который не содержит никаких процедур).

Для пиктограммы на кнопке можно создать две растровые картинki. Первая картинка должна быть размером 18 пикселей в ширину на 16 пикселей в высоту; эта картинка появится, если не установлен флажок **Большие кнопки** в диалоге **Инструментальные панели**. Вторая картинка имеет размеры 26 на 24 пикселей; она появится, если пользователь установит флажок **Большие кнопки**. Вы должны обязательно создать обе картинki, даже если одна Вам не нужна.

Создание пиктограммы состоит из следующих этапов:

- Подобрать или создать DLL-файл, в который будут помещены новые пиктограммы.
- Открыть этот DLL-файл в редакторе ресурсов, таких, как AppStudio.
- Для каждой пиктограммы создать две растровые картинki (BMP-ресурс): одну размером 18 пикселей в ширину на 16 пикселей в высоту и другую размером 26 на 24 пикселей.



Не забывайте, что требуется создать ресурс типа "bitmap", а не "icon".

- Назначить последовательные номера (ID) обоим растровым ресурсам. Например, если Вы назначили ID равный 100 картинке 18 X 16 пикселей, то картинке 26 x 24 пикселей нужно присвоить ID-номер равный 101.

Создав пару растровых ресурсов, Вы можете в приложении MapBasic использовать их с помощью операторов **Create ButtonPad** и **Alter ButtonPad**. Из текста программы Вы должны обращаться к меньшей (18 на 16) картинке. Например, если Вы назначите ID-номера 100 и 101 новой пиктограмме, то программа должна обращаться к номеру 100, как показано в следующем примере:

```
Alter ButtonPad "Программы"  
  Add PushButton  
    Icon 100 File "MBICONS1.DLL"  
    HelpMsg "Добавить новую запись"  
    Calling new_route  
  Show
```

DLL-файл, в котором помещаются пиктограммы (в примере выше это MBICONS1.DLL), должен быть установлен в Вашей системе по одному из следующих маршрутов:

- в том же каталоге, что и MBX-файл, который его использует;
- в личном каталоге пользователя; в каталоге, содержащем пакет файлов MapInfo;
- в каталогах WINDOWS;
- или WINDOWS\SYSTEM.
- Если MapInfo не находит DLL-библиотеку ни в одном из этих каталогов, то поиск продолжается по каталогам, описанным в системной переменной PATH.

Вы можете, разумеется, явно указать каталог (например, Icon 100 File "C:\GIS\MBICONS1.DLL"). Функции **ProgramDirectory\$( )** и **ApplicatopnDirectory\$( )** помогут Вам построить нужные имена каталогов из MBX-программы.

## Создание новых курсоров в Windows

Процесс создания нового курсора почти в точности повторяет процесс создания пиктограммы, описанный выше. Курсор, правда, имеет несколько особенностей, например, "точку указывания" ("hot spot").

Новый курсор помещается в виде CURSOR-ресурса в DLL-файл. Вы можете помещать в один DLL-файл как CURSOR-ресурсы, так и BMP ресурсы.

## Связь между приложениями с использованием DDE

Связь между приложениями является обобщенным термином для обмена информацией между отдельными пакетами программ. Windows поддерживает ее через протокол Динамического обмена данными, обычно известный как DDE.

Если два приложения Windows оба поддерживают DDE, приложения могут обмениваться командами и данными. Например, Windows программа типа Microsoft Excel, может дать команду MapInfo (например, **Map From world**).

## Обзор DDE-обмена

DDE-связь – процесс, который может происходить между двумя приложениями Windows. Оба приложения должны быть запущены, и оба должны поддерживать протокол DDE. В одном сеансе обмена могут участвовать только две программы; однако, программы (MapInfo, Excel и другие) могут одновременно участвовать во многих сеансах.

Одно приложение из двух начинает сеанс обмена посланиями. Это приложение называется клиентом. Другое, пассивное приложение называется сервером. Клиент управляет обменом; например, посылает инструкции-запросы серверу. Сервер только выполняет команды и отправляет запрошенные данные.

## MapBasic как DDE-клиент

Язык MapBasic поддерживает следующие операторы и функции, позволяющие приложению MapBasic действовать как клиент при DDE обмене.

Операторы и функции MapBasic	Действие
<b>DDEInitiate( )</b>	Открывает сеанс обмена;
<b>DDERequest\$( )</b>	Запрашивает информацию у сервера;
<b>DDEPoke</b>	Посылает информацию серверу;
<b>DDEExecute</b>	Посылает серверу команды;
<b>DDETerminate</b>	Закрывает один сеанс DDE-обмена.
<b>DDETerminateAll</b>	Закрывает все сеансы DDE-обмена, открытые программой MapBasic.

Подробно эти операторы и функции описаны в *Справочнике MapBasic* и в *Справочной системе*.

Чтобы начать сеанс DDE-обмена, нужно вызвать функцию **DDEInitiate( )**. Функции **DDEInitiate( )** нужно задать два параметра: имя приложения application и название объекта "topic".

Обычно параметр "application" – это имя сервера (например, имя "Excel" при DDE-обмене обозначает Microsoft Excel). Список параметров "topic" зависит от программы. Часто, параметром "topic" бывает имя файла или документа, открытого программой-сервером.

Например, пусть в Excel открыта таблица TRIAL.XLS, тогда приложение MapBasic может начать сеанс обмена следующими операторами:

```
Dim channelnum As Integer
channelnum = DDEInitiate("Excel", "TRIAL.XLS")
```

В этом примере Excel – имя приложения ("application"), а TRIAL.XLS – имя объекта ("topic").



Многие программы, поддерживающие DDE (и MapInfo тоже), поддерживают специальный объект по имени "System". Начав сеанс обмена с использованием объекта "System", Вы впоследствии можете в этом сеансе получить список всех доступных объектов

Каждый сеанс DDE обменивается данными через собственный уникальный канал ("channel"). Функция **DDEInitiate( )** возвращает номер канала в виде целого числа. Этот номер в дальнейшем используется другими операторами DDE-обмена.

После установления связи приложение MapBasic может посылать команды серверу посредством оператора **DDEExecute**. Например, приложение MapBasic может заставить программу-сервер открывать или закрывать файлы.

Используя функцию **DDERequest\$( )**, приложение MapBasic запрашивает информацию у сервера. При вызове **DDERequest\$( )** нужно определить имя элемента ("item"), чтобы сервер точно определил, какую именно информацию от него требуют. Например, если сервером является электронная таблица, то запрашиваемым элементом может быть имя ячейки.

Используя функцию **DDEPoke**, можно посылать информацию программе серверу. Пересылка данных из приложения MapBasic через DDE-связь имитирует ввод данных пользователем в соответствующий документ программы сервера. В следующем примере приложение MapBasic помещает фразу "СВ-округ" в ячейку электронной таблицы.

```
DDEPoke channelnum, "R1C2", "СВ-округ"
```

Когда все работы по DDE-связи выполнены, обмен должен быть завершен клиентом (приложением MapBasic), для чего выполняются операторы **DDETerminate** или **DDETerminateAll**. Оператор **DDETerminate** закрывает один канал DDE-обмена; **DDETerminateAll** закрывает все DDE-каналы, открытые из данного приложения. При этом другие приложения MapBasic, поддерживающие в это время свои сеансы DDE-обмена, не затрагиваются.

Когда приложение MapBasic является клиентом, то во время работы может порождаться ошибка, если сервер долгое время не отвечает.

Время, которое сервер может не отвечать клиенту без порождения ошибки, записывается в реестре Windows. Подробные сведения о том, какие параметры MapInfo Professional сохраняет в реестре, содержатся в *Справочной системе*.

## MapInfo Professional в роли DDE-сервера

MapInfo Professional выступает в роли сервера, когда другая Windows программа начинает сеанс DDE-обмена. Программа-клиент может передавать операторы MapBasic, читать значения глобальных переменных MapBasic и изменять их. Windows-программа, поддерживающая протокол DDE, может выполнять различные действия в MapInfo, такие, как открытие таблиц, показ Карт и Списков оператором MapBasic **Map**. (Однако, управляющие операторы MapBasic не могут быть выполнены MapInfo как сервером.)

Операторы и функции поддержки DDE-связи могут в других программах называться по-другому. Если в MapBasic'e используется оператор **DDEPoke**, в других программах такие же операции могут выполняться под другим названием. Выяснить, какие DDE-операторы используются в конкретном приложении Windows, можно в документации этого приложения.

Программы и приложения, выступающие в роли DDE-клиентов по отношению к MapBasic, должны задавать три основных параметра: "application", "topic" и "item".

**Имя приложения (Application name):** задайте "MapInfo Professional" в качестве имени приложения, если требуется начать DDE-обмен, в котором MapInfo Professional выступает сервером.

**Имя программы (Topic name):** задайте "System" или имя выполняемой программы MapBasic (например, SCALEBAR.MBX).

**Имя процедуры (Item name):** имя процедуры зависит от программы, в которой она используется. Например, следующая MapBasic-программа, которую Вы можете напечатать непосредственно в окне MapBasic, проводит простой DDE диалог, используя "MapInfo Professional" как имя приложения и "System" как имя программы.

В следующей таблице перечислены операции и процедуры, выполняемые в процессе DDE-обмена между приложением MapInfo и программным объектом System.

Команда DDE-обмена	Имя DDE-процедуры	Эффект
DDE-запрос Peek	"SysItems"	MapInfo Professional возвращает перечень имен, разделенных символом табуляции (TAB), принятых в объекте System. Topics SysItems Formats Version
DDE-запрос Peek	"Topics"	MapInfo Professional возвращает список разделенных символами табуляции доступных программ ("System" и все запущенные программы на языке MapBasic).
DDE-запрос Peek	"Formats"	MapInfo Professional возвращает список форматов буфера обмена, поддерживаемых MapInfo Professional (в виде текста).
DDE-запрос Peek	"Version"	MapInfo Professional возвращает текстовую строку, в которой записан номер версии MapInfo Professional, умноженный на 100. Например, MapInfo Professional 9.5.0 возвращает "950". Пример смотрите ниже.

Команда DDE-обмена	Имя DDE-процедуры	Эффект
DDE-запрос Peek	Выражение MapBasic	MapInfo Professional обрабатывает строку как выражение на языке MapBasic и возвращает полученное значение в виде строки. Если выражение содержит ошибку и не может быть выполнено, MapInfo Professional возвращает сообщение об ошибке. Такая возможность реализована, впервые, в MapInfo Professional 4.0 и во всех последующих версиях.
DDE-команда Execute	Текстовое сообщение	MapInfo Professional пробует выполнять сообщение как оператор MapBasic, словно пользователь набрал его в окне MapBasic. Оператор не может содержать обращения к нестандартным (пользовательским) функциям, хотя может содержать обращения к стандартным функциям. Оператор не может ссылаться на переменные, которые определены в оттранслированных прикладных программах (MBX-файлах). Однако оператор может ссылаться на переменные, которые были определены при помощи оператора <b>Dim</b> в окне MapBasic.

Например, следующая MapBasic-программа, которую Вы можете напечатать непосредственно в окне MapBasic, проводит простой DDE диалог, используя "MapInfo" как имя приложения и "System" как имя программы.

```
Dim i_channel As Integer
i_channel = DDEInitiate("MapInfo", "System")
Print DDERequest$(i_channel, "Version")
DDETerminate i_channel
```

Функция **DDEInitiate( )** начинает сеанс DDE-обмена. Затем функция **DDERequest\$( )** производит запрос, используя "Version" как имя процедуры ("item").

Если Вы используете имя приложения MapBasic (например, "C:\MB\SCALEBAR.MBX" или "SCALEBAR.MBX" или "SCALEBAR") как DDE программу, Вы можете использовать следующие имена в качестве процедур:

В следующей таблице перечислены действия и процедуры поддерживаемые во время DDE-сеанса между программой MapInfo ("application") и выполняемой MapBasic-программой ("topic").

Команда DDE-обмена	Имя DDE-процедуры	Эффект
DDE-запрос Peek	"{items}"	MapInfo Professional возвращает разделенный символами табуляции список глобальных переменных, определенных в приложении. Пример смотрите ниже.
DDE-запрос Peek	Имя глобальной переменной	MapInfo возвращает строку, представляющую собой значение глобальной переменной.
DDE-запрос Peek	Строка, не являющаяся именем глобальной переменной	Если приложение MapBasic содержит функцию с именем <b>RemoteQueryHandler()</b> , MapInfo вызовет ее. Внутри тела функции можно определить имя "item" посредством вызова <code>CommandInfo(CMD_INFO_MSG)</code> .
DDE-запрос на передачу данных (Poke)	Имя глобальной переменной	MapInfo Professional изменит значение переменной на новое.
DDE-команда Execute	Текстовое сообщение	Если в приложении MapBasic есть процедура с именем <b>RemoteMsgHandler</b> , MapInfo вызовет ее. Внутри тела процедуры можно узнать содержание текстового послания при помощи вызова <code>CommandInfo(CMD_INFO_MSG)</code> .

Например, следующая MapBasic-программа, которую Вы можете напечатать непосредственно в окне MapBasic, проводит простой DDE диалог, "SCALEBAR.MBX" как имя программы. При помощи этого обмена данными по DDE выводится список глобальных переменных, используемых приложением SCALEBAR.MBX.



Обратите внимание: обмен данными будет выполняться, если приложение SCALEBAR.MBX уже запущено.

```
Dim i_channel As Integer
i_channel = DDEInitiate("MapInfo", "SCALEBAR.MBX")
Print DDERequest$(i_channel, "{items}")
DDETerminate i_channel
```

## Как MapInfo Professional обрабатывает DDE-команды Execute

Существуют два способа, которыми клиентское приложение может посылать MapInfo Professional запрос-команду:

- Когда в DDE-обмене используется в качестве внешней программы "System" и клиентское приложение посылает запрос на выполнение действия, MapInfo Professional пробует выполнять заданное сообщение как оператор MapBasic.
- Когда в DDE-обмене используется имя программы MapBasic, а клиент посылает запрос на выполнение действия, MapInfo Professional вызывает процедуру **RemoteMsgHandler** приложения, которая может затем вызывать **CommandInfo ()**, чтобы определить текст команды.

Приложение MapBasic может действовать как клиент в одном DDE-обмене сообщениями, и, одновременно, как сервер в другом сеансе обмена сообщениями. Приложение MapBasic может инициализировать обмен сообщениями с другим приложением MapBasic или непосредственно с MapInfo Professional.

## Связь с приложениями Visual Basic с использованием DDE

Программист может расширить возможности MapBasic за счет более мощного языка Microsoft Visual Basic. Например, можно создавать средствами Visual Basic диалоговые окна более сложные, чем позволяет оператор MapBasic **Dialog**. Например, Visual Basic позволяет добавлять управляющие элементы диалога, которые нельзя задать оператором **Dialog**.

Приложение MapBasic может связываться с приложением Visual Basic с использованием DDE (или автоматизации OLE). Более подробно об этом рассказывает [Интегрированная картография](#).

## Пример DDE-обмена сообщениями

Пример чтения/записи в таблицу Excel с помощью DDE-связи приведен в описании функции **DDEInitiate ()** в *Справочнике MapBasic*.

Программа "Watcher" (APPINFO.MBX), поставляемая вместе с пакетом, содержит более сложный пример DDE-обмена. Программа "AppInfo" используется при отладке. Если Вы запустили программу MapBasic и, следом за этим, программу "AppInfo", то последняя проверяет глобальные переменные программы MapBasic. Процедура "WhatApps( )" запрашивает DDE-процедуру с именем "Topics", чтобы получить список исполняющихся MBX-файлов. Процедура "WhatGlobals( )" выполняет другой DDE-обмен данными, обращаясь к процедуре с именем "{Items}", чтобы получить список глобальных переменных.

## Контроль глобальных переменных с помощью DDE

Когда MapInfo выступает в роли сервера в DDE-обмене, во время сеанса можно поддерживать как "теплую", так и "горячую" связь. Иными словами, когда приложение Windows начинает сеанс DDE-обмена, контролирующий значения переменных MapBasic, то Windows может уведомлять клиента об изменении значения глобальной переменной автоматически или по запросу.

Когда приложение MapBasic выступает в роли DDE-клиента, подобный автоматический контроль невозможен.

## Добавление Справочной системы к Вашему приложению

Если Вы разрабатываете сложное приложение, Вы можете снабдить его своей Справочной системой. Чтобы создать собственную Справочную систему, Вам нужен специальный компилятор. Среда разработки MapBasic его не содержит. Вы можете воспользоваться Компилятором Справочников для Windows (Microsoft Windows Help Compiler), описание которого Вы можете найти в документации фирмы Microsoft Corp.



Сотрудники технической поддержки Pitney Bowes Software Inc. не смогут помочь в создании файлов Справочных систем.

---

Из программы Вы можете управлять Справочником с помощью операторов **Open Window**, **Close Window** и **Set Window**. Например, следующий оператор открывает окно Справочника и показывает Оглавление (Contents) справки:

```
Set Window Help Contents
```

Оператор **Set Window** можно использовать по-разному (см. описание этого оператора в *Справочнике MapBasic*). Многие формы оператора **Set Window** требуют задания целочисленного идентификатора окна; в случае же со *Справочной системой* достаточно указать слово **Help** или Справка.

Если Вы создали свой *Справочную систему* и назвали ее DISPATCH.HLP, то его можно открыть оператором

```
Set Window Help File "C:\MAPINFO\DISPATCH.HLP"
```

Следующий оператор открывает *Справочную систему* на теме, имеющей внутренний номер 500:

```
Set Window Help ID 500
```

Внутренние номера (ID-номера) определяются в разделе [MAP] файла проекта Справочника (например *имя\_файла*.HPJ). Подробно структура этого файла описана в документации к Windows Software Developers Kit (SDK).

Если нужно снабдить *Справочной системой* диалоговое окно, созданное в Вашей программе, поместите в диалог кнопку (элемент **Button**) с надписью "Help" или "Справка" с помощью следующей конструкции:

```
Control Button  
  Title "Справка"  
  Calling show_help_sub
```

Назначьте кнопке **Справка** процедуру-обработчик, в которую поместите оператор **Set Window**. Пользователь, нажав на кнопку «Справка», откроет окно *Справочной системы*. Подробнее о процедурах-обработчиках см. в главе **Создание интерфейса пользователя**.

# Интегрированная картография

Вы можете управлять пакетом MapInfo Professional, используя языки программирования, отличные от языка MapBasic. Например, если Вам хорошо знакомо программирование на языке Visual Basic, Вы можете включить (интегрировать) окно Карты MapInfo в Ваше приложение, написанное на языке Visual Basic, выполняя при этом основную часть или даже всю работу по программированию в среде Visual Basic. Такой способ разработки приложений известен как Интегрированная картография, так как при этом Вы интегрируете элементы MapInfo в другие программы.

Если Вы уже умеете программировать на таких языках, как C или Visual Basic, Вы увидите, что метод Интегрированной картографии обеспечивает простейший способ включения окон MapInfo в приложения, разработанные в других средах программирования, не использующих MapBasic.



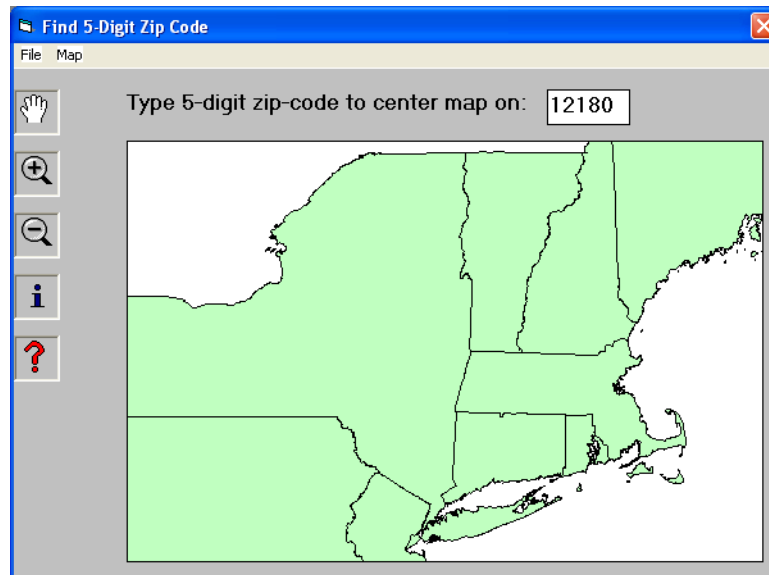
Создание приложений с Интегрированной картографией в среде .Net описано в разделе **Интегрированная картография в .Net на стр. 278**.

## В этой главе:


- ♦ Что такое Интегрированная картография .....216
- ♦ Концепции Интегрированной картографии .....216
- ♦ Технические аспекты Интегрированной картографии ....218
- ♦ Простейший пример: “Hello, (Map of) World” .....218
- ♦ Подробное обсуждение Интегрированной картографии ..219
- ♦ Использование Callback-вызовов .....228
- ♦ Другие способы использования OLE-уведомлений .....232
- ♦ Полезные операторы и функции языка MapBasic .....234
- ♦ Объектная модель механизма управления OLE .....236
- ♦ Аргументы командной строки MapInfo Professional .....253
- ♦ Добавление инструментальных кнопок .....259
- ♦ Где получить дополнительную информацию .....263

## Что такое Интегрированная картография

Вид на экране компьютера приложения с Интегрированной Картой определяется Вами. При желании, Вы можете создать интерфейс пользователя, радикально отличающийся от интерфейса MapInfo. Например, на следующем рисунке показано окно Карты MapInfo, интегрированное в форму окна диалога Visual Basic.



При интегрировании окна Карты MapInfo в Вашу программу, пользователь видит на экране оригинальное полнофункциональное окно MapInfo, а не растр, метафайл или графическое представление какого-либо другого типа. Вы можете разрешить пользователю интерактивно взаимодействовать с Картой (используя, например, инструменты Лупа для увеличения Карты). Интегрированное окно Карты имеет все возможности, присущие окну Карты в среде MapInfo.

 Когда пользователь запускает приложение с встроенной Картой, заставка MapInfo не демонстрируется.

## Концепции Интегрированной картографии

Для создания приложения с Интегрированной Картой, Вы должны написать программу – но не программу на языке MapBasic. Приложения с Интегрированной Картой могут быть написаны на нескольких языках программирования, среди которых наиболее часто используются C и Visual Basic. Примеры кода в этой главе даны на языке Visual Basic.

В Вашей программе должна присутствовать инструкция, запускающая MapInfo в фоновом режиме. Например, в программе на языке Visual Basic Вы можете запустить MapInfo вызовом функции CreateObject( ). Программа MapInfo Professional запускается в фоновом режиме незаметно для пользователя, не выводя заставку на дисплей.

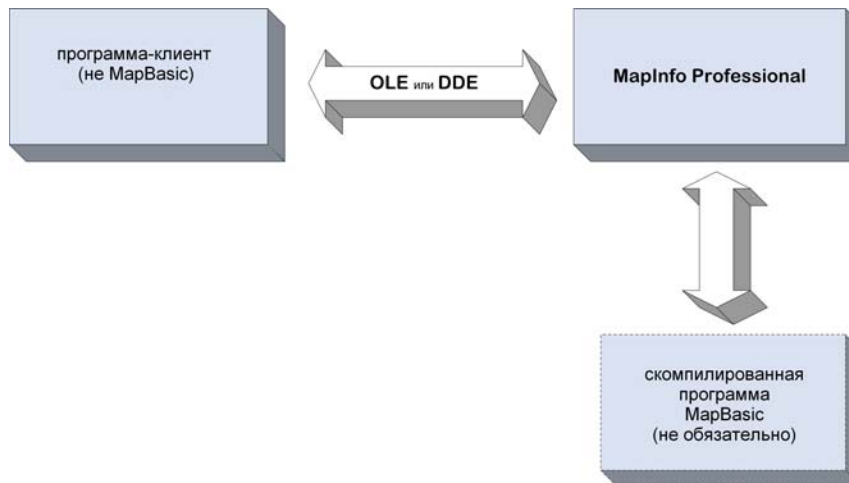


Ваша программа осуществляет управление программой MapInfo, конструируя строки, представляющие операторы языка MapBasic, которые затем передаются в MapInfo Professional посредством механизма управления объектами OLE (OLE Automation) или динамического обмена данными (DDE). MapInfo Professional выполняет эти операторы точно так же, как если бы пользователь вводил их с клавиатуры в окно MapBasic.

Если Вы хотите открыть окно Карты, используйте оператор **Map From** языка MapBasic точно таким же образом, как в обычной программе MapBasic. Однако, в приложении с Интегрированной Картой Вы должны также использовать дополнительные операторы (например, **Set Next Document Parent**), чтобы окно Карты могло стать подчиненным (порожденным) окном Вашего приложения. Этот процесс известен как “переподчинение” (parenting) окна. Вы можете переподчинить окна Карты, Списка, Графика, Отчета и Легенды.

**i** Переподчинение окон MapInfo другому приложению не дает программе MapInfo автоматического доступа к данным этого приложения. Для отображения данных приложения в окне MapInfo Professional, Вы должны предварительно записать эти данные в таблицу MapInfo.

На этом рисунке показаны основные элементы приложения с интегрированными Картами:



Заметьте, что наличие скомпилированной MapBasic программы (файл MBX) необязательно. Для некоторых приложений ее создание Вам может не понадобиться. Однако, если у Вас уже имеются написанные программы MapBasic, Вы можете использовать их в приложении с Интегрированной Картой.

## Технические аспекты Интегрированной картографии

### Системные требования

- Интегрированная картография требует наличия программы MapInfo 4.0 или более поздней версии. Вы можете использовать полную копию MapInfo Professional или т.н. исполнимый (runtime) модуль MapInfo (специальная версия MapInfo, поставляемая только в качестве основы для специализированных приложений).
- Компьютер должен иметь достаточно памяти и системных ресурсов для одновременного выполнения и программы-клиента, и MapInfo Professional.
- Ваша программа-клиент (например, Ваша программа на языке Visual Basic) должна быть способна действовать в качестве контроллера механизма управления объектами OLE (OLE Automation controller) или клиента динамического обмена данными (DDE-клиента). Рекомендуется применение механизма управления объектами OLE как более быстрого и надежного метода по сравнению с динамическим обменом данными. OLE Automation обеспечивает работу над ошибками лучше чем DDE. MapInfo Professional использует свойства OLE для отображения кодов ошибок при выполнении программы; если вместо OLE используется DDE, коды ошибок при исполнении программы получить невозможно.
- Ваша клиентская программа должна иметь возможность создавать элементы интерфейса (окна, кнопки и т.д.), т.е. элементы, содержащие окно Карты и средства управления ею. Клиентская программа должна уметь определять номера HWND для элементов пользовательского интерфейса.

Например, в Visual Basic'е можно разместить элемент управления или форму "PictureBox". Когда команда, которая должна заполнить "PictureBox" Картой, передается MapInfo Professional, необходимо указать HWND этого "PictureBox".

### Другие технические замечания

- Для разработки приложения с Интегрированной Картой Вы должны написать программу на языке, отличном от MapBasic. (В дальнейшем она будет называться программа-клиент). Вы можете написать программу-клиент, используя популярные среды программирования, такие как Visual Basic (3.0 или более новая версия), C, PowerBuilder или Delphi.
- Интегрированная картография использует механизм управления объектами OLE (OLE Automation), но не использует OLE-внедрение. Когда Вы хотите поместить окно Карта MapInfo в Ваше приложение, Вы не осуществляете его внедрение (embedding); напротив, Вы переподчиняете окно посредством пересылки программе MapInfo Professional серии командных строк. В результате окна MapInfo отображаются на дисплее как порожденные Вашим приложением (child window).
- Интегрированная картография не использует специализированные элементы управления VBX (Visual Basic Custom Control) или OCX. MapInfo Professional не предоставляет Вам какие-либо заголовочные файлы или библиотеки. (Программное обеспечение MapInfo включает в себя несколько динамически подключаемых библиотек DLL, но Вы не можете обращаться к ним непосредственно; эти библиотеки предназначены для внутреннего использования программой MapInfo Professional).

## Простейший пример: “Hello, (Map of) World”

Следующая программа на Visual Basic даст Вам представление о том, как легко встроить окно MapInfo в другую программу.

Сначала создадим новый проект Visual Basic. В процедуре "General Declarations" (общие определения) объявим переменную типа "Object". (В этом примере она будет называться "mi", но Вы можете назвать ее по своему.)

```
Dim mi As Object
```

Затем добавим в процедуру "Form\_Load" следующие строки:

```
Sub Form_Load()  
    Set mi = CreateObject("MapInfo.application")  
    mi.do "Set Application Window " & Form1.hWnd  
    mi.do "Set Next Document Parent " & Form1.hWnd & " Style 1"  
    mi.do "Open Table ""World"" Interactive Map From World"  
    mi.RunMenuCommand 1702  
    mi.do "Create Menu ""MapperShortcut"" ID 17 As ""(-"" "  
End Sub
```

Как только Вы запускаете программу на Visual Basic, она запускает MapInfo, которая создает окно Карты. При этом MapInfo действует как “скрытый” сервер, а окно Карты ведет себя как порожденное программой Visual Basic. В следующих разделах подробно объясняется каждый шаг встраивания Карты в другие программы.

## Подробное обсуждение Интегрированной картографии

Последующее обсуждение показывает, как интегрировать элементы MapInfo Professional в приложение, написанное на языке Visual Basic (в дальнейшем VB-приложение или VB-программа), и исходит из двух предположений:

- Вы понимаете основные термины и концепции программирования для операционной среды Windows. Например, Вы должны знать, что такое “порожденное окно”. Информацию о концепциях программирования для среды Windows Вы можете найти в документации по Вашему языку программирования.
- Вы должны быть знакомы с программированием на Visual Basic, т.к. в примерах используется синтаксис Visual Basic. Даже, если Вы не разработчик программ на Visual Basic, Вам следует прочесть этот раздел. Основные положения и процедуры, описанные в этом разделе, могут быть использованы и при программировании на других языках программирования.

### Запуск MapInfo Professional

Запуск уникального экземпляра программы MapInfo осуществляется вызовом функции **CreateObject( )** языка Visual Basic с присваиванием возвращаемого значения объектной переменной.



Вы можете декларировать объектную переменную как глобальную; в противном случае объект MapInfo освобождается после выхода из локальной процедуры.

---

Например: если назвать объектную переменную “mapinfo”, следующий оператор запустит MapInfo Professional:

```
Set mapinfo = CreateObject("MapInfo.Application")
```

Для подключения к ранее исполнявшемуся экземпляру MapInfo, который не был запущен вызовом функции **CreateObject( )**, используйте функцию **GetObject( )**.

```
Set mapinfo = GetObject("MapInfo.Application")
```



Если Вы работаете с Runtime-версией MapInfo, а не с полной копией, задавайте "MapInfo.Runtime" вместо "MapInfo.Application". Runtime-версия и полная версия могут работать одновременно.

---

Функции Visual Basic **CreateObject( )** и **GetObject( )** используют механизм OLE Automation, чтобы установить соединение с MapInfo Professional. Если Вам необходимо применить DDE-обмен вместо OLE-связи, используйте функцию **Shell( )** языка Visual Basic для запуска программы MapInfo Professional, а затем используйте свойство (property) **LinkMode** для установления DDE-обмена.

В Windows можно запускать несколько экземпляров MapInfo Professional. Если Вы запустите MapInfo Professional и вслед за этим программу, использующую Интегрированную картографию и вызывающую **CreateObject( )**, то будут работать два независимых экземпляра MapInfo.

## Передача команд в программу MapInfo

После запуска программы MapInfo Professional, необходимо сконструировать текстовые строки, представляющие операторы языка MapBasic. Например, для исполнения программой MapInfo MapBasic-оператора **Open Table** Вы можете задать в VB-программе следующую строку:

```
msg = "Open Table ""STATES.TAB"" Interactive "
```

Если Вы установили связь с MapInfo Professional, используя механизм управления объектами OLE (OLE Automation), передавайте командную строку программе MapInfo Professional методом **Do**. Например:

```
mapinfo.Do msg
```

При использовании метода **Do** программа MapInfo исполняет командную строку точно так же, как если бы пользователь ввел команду с клавиатуры в окно MapBasic.

Если Вы установили связь с MapInfo Professional, используя механизм управления объектами DDE, передавайте командную строку программе MapInfo Professional DDE-методом **LinkExecute**.

## Запрос данных из программы MapInfo Professional

Для получения значения выражения MapBasic, задайте в VB-программе строку, представляющую это выражение. Например, если Вы хотите определить значение, возвращаемое функцией MapBasic **WindowID(0)**, задайте следующую строку (в среде Visual Basic):

```
msg = "WindowID(0) "
```

Если Вы установили связь с MapInfo Professional, используя механизм управления объектами OLE (OLE Automation), передавайте командную строку программе MapInfo Professional OLE-методом **Eval**. Например:

```
Dim result As String  
result = mapinfo.Eval "WindowID(0)"
```

При использовании метода **Eval**, программа MapInfo Professional интерпретирует строку как выражение языка MapBasic, определяет значение выражения и возвращает это значение в виде строки.



Если выражение приводится к логическому значению (тип Logical), MapInfo возвращает односимвольную строку, "T" или "F" соответственно.

---

Если Вы установили связь с MapInfo Professional, используя динамический обмен данными (DDE), запросите значение выражения DDE-методом **LinkRequest**.

### Переподчинение окон MapInfo Professional

После запуска MapInfo, используйте оператор **Set Application Window** языка MapBasic, для обеспечения перехвата управления Вашей программой клиентом диалоговых окон и сообщений об ошибках программы MapInfo. (В следующем примере "FormName" является именем формы в среде Visual Basic.)

```
msg = "Set Application Window " & FormName.hWnd  
mapinfo.Do msg
```

Затем, в желаемой точке включения окна MapInfo в Ваше VB-приложение передайте MapInfo Professional оператор **Set Next Document**, за которым следует MapBasic-оператор, создающий окно.

Например, следующий фрагмент кода создает окно Карты MapInfo как подчиненное окно программы-клиента. (В этом примере "MapFrame" является именем элемента управления **Picture Box** в среде Visual Basic.)

```
msg = "Set Next Document Parent " & MapFrame.hWnd & " Style 1"  
mapinfo.Do msg  
  
msg = "Map From States"  
mapinfo.Do msg
```

Оператор **Set Next Document** позволяет Вам "переподчинять" окна документов. Синтаксис оператора **Set Next Document** требует указания уникального номера HWND элемента управления в Вашей VB-программе. При последующем создании окна документа MapInfo (с использованием операторов **Map**, **Graph**, **Browse**, **Layout** или **Create Legend**), создаваемое окно переподчиняется таким образом, что элемент управления программы-клиента с этим значением HWND становится для окна порождающим объектом.

Оператор **Set Next Document** содержит параметр **Style**, с помощью которого можно задавать тип создаваемого окна. В примере выше задано режим **Style 1**, который создает подчиненное окно без рамки. Можно задать **Style 2**, чтобы создать всплывающее окно со строкой заголовка половинной высоты (подобно окну легенды MapInfo Professional) или **Style 3**, чтобы создать всплывающее окно со строкой заголовка единичной высоты.

Для каждого переподчиняемого окна необходимо передать программе MapInfo из Вашей программы пару операторов – оператор **Set Next Document Parent**, а затем оператор, создающий окно. После создания окна Вам может понадобиться запросить из MapInfo значение функции WindowID(0) – целочисленный ID-номер окна (Window ID) в MapInfo, так как многие операторы языка MapBasic требуют задания этого идентификатора.

```
mapid = Val(mapinfo.eval("WindowID(0)"))
```

Заметьте, что даже после переподчинения окна Карты, MapInfo продолжает управлять им. Если часть окна нужно перерисовать, MapInfo автоматически обновляет его. Поэтому, клиентская программа может не обращать внимания на сообщения о перерисовке, адресованные подчиненному окну.

Если Вы программируете на C, то просто так игнорировать сообщения о перерисовке не удастся. В этом случае нужно добавить в описание порождающего окна стиль **WS\_CLIPCHILDREN**.

### Переподчинение окон Легенд, растровых диалогов и других окон MapInfo Professional

MapInfo Professional имеет несколько немодальных окон, включая окно Информации, окно Сообщений, диалогов, относящихся к растрам и окно Статистики. Чтобы изменить порождающее окно для одного из этих специальных “плавающих” окон, используйте оператор MapBasic **Set Window ... Parent**. Например, программа-пример FindZip использует следующее предложение:

```
mapinfo.do "Set Window Info Parent " & FindZipForm.hWnd
```

Заметьте, что способ переподчинения окна Информации другой, чем для окна Карты. В последнем случае не используется предложение **Set Next Document**. Дело в том, что может существовать несколько окон Карты.

Окна Легенды – особый случай. Обычно существует только одно окно Легенды, так же, как и одно окно Информации. Однако при помощи оператора MapBasic **Create Legend** Вы можете создавать дополнительные окна Легенды.

Для одного окна Легенды используйте оператор MapBasic **Set Window Legend Parent**.

Чтобы создать дополнительное окно Легенды, используйте оператор MapBasic **Set Next Document** и оператор **Create Legend**. Заметьте, что в этом случае Вы создаете Легенду, которая привязана к одному определенному окну Карты или окну Графика. Такое окно Легенды не изменяется, когда другое окно становится активным.

Вы можете создать “плавающее” окно Легенды внутри окна Карты. В операторе **Set Next Document** укажите идентификатор HWND окна Карты как порождающее окно. Легенда превратится в рамку, “вставленную” в Карту. Пример смотрите в программе FindZip.

## Разрешение пользователю изменить размеры окна Карты

Может ли пользователь изменить размеры окна Карты, зависит от того, как Вы настроите приложение. Типовая программа FindZip помещает окно Карты в элемент управления Visual Basic PictureBox, так что размер не может быть изменен. Однако Вы могли бы использовать интерфейс MDI, который позволяет пользователю изменить размеры окна.

---

**i** Когда пользователь изменяет размеры окна Карты, MapInfo не производит автоматически обновление содержания окна, чтобы заполнить его новым размером. Следовательно, если Ваше приложение разрешает, чтобы пользователь изменил размеры окна Карты, Вы должны вызвать функцию Windows API **MoveWindow**, чтобы заставить окно Карты соответствовать новому размеру.

---

Например, Вы можете использовать следующее оператор, описывающий доступ к API функции **MoveWindow**:

```
Declare Function MoveWindow Lib "user32" _  
    (ByVal hWnd As Long, _  
     ByVal x As Long, ByVal y As Long, _  
     ByVal nWidth As Long, ByVal nHeight As Long, _  
     ByVal bRepaint As Long) As Long
```

Когда пользователь изменяет размеры окна Карты, вызовите MoveWindow. В Visual Basic при изменении размера вызывается процедура "Form\_Resize( )"; Вы можете вызвать функцию **MoveWindow** внутри этой процедуры, как показано в следующем примере.

```
Dim mHwnd As Long  
mHwnd = Val(mapinfo.Eval("WindowInfo(FrontWindow(),12)"))  
MoveWindow mHwnd, 0, 0, ScaleWidth, ScaleHeight, 0
```

Номер 12 соответствует идентификатору MapBasic **WIN\_INFO\_WND**.

Заметьте, что "ScaleWidth" и "ScaleHeight" – стандартные свойства формы Visual Basic, содержащие текущую ширину и высоту формы.

---

**i** "ScaleMode" должен быть установлен в пикселах (Pixels), поскольку "ScaleWidth" и "ScaleHeight" указываются в пикселах.

---

## Интеграция панелей инструментов MapInfo Professional

Вы не можете переподчинить Инструментальные панели MapInfo. Если Вы хотите, чтобы Ваша клиентская программа имела такие панели, Вы должны создать кнопки на языке, который Вы используете. Например, если Вы используете Visual Basic, то должны создать Ваши кнопки при помощи Visual Basic.

Если Вы хотите, чтобы кнопка панели Visual Basic эмулировала стандартную кнопку MapInfo, используйте метод **RunMenuCommand**. (Этот метод действует так же, как оператор MapBasic **Run Menu Command**). Например, типовая программа FindZip содержит процедуру "InfoTool\_Click" с оператором:

```
mapinfo.RunMenuCommand 1707
```

Когда пользователь нажимает на соответствующую кнопку, программа FindZip вызывает метод MapInfo **RunMenuCommand**, который активизирует инструмент под номером 1707 (инструмент **Информация** MapInfo). В результате, инструмент **Информация** MapInfo Professional становится активным.

Число 1707 есть номер инструмента **Информация**. Вместо того, чтобы использовать такие числа, Вы можете использовать идентификаторы, более понятные в тексте программы. MapBasic определяет стандартный идентификатор **M\_TOOLS\_PNT\_QUERY**, который имеет значение 1707. Таким образом, следующий пример команды **RunMenuCommand** обеспечивает такой же эффект, как и в предыдущем примере:

```
mapinfo.RunMenuCommand M_TOOLS_PNT_QUERY
```

Использование идентификаторов (типа **M\_TOOLS\_PNT\_QUERY**) делает Вашу программу более легкой для чтения. Однако, если Вы планируете использовать идентификаторы в Вашем коде, то Вы должны включить соответствующий заголовочный файл MapBasic. Если Вы используете Visual Basic, используйте файл MAPBASIC.BAS. Для C используйте файл MAPBASIC.H.

В следующей таблице приведены идентификаторы инструментальных кнопок MapInfo. Они содержатся в файлах MAPBASIC.BAS (для Visual Basic), MAPBASIC.H (для C), и MENUS.DEF (для MapBasic).

Кнопки панели Операции	Номер	Идентификатор
<b>Выбор</b>	1701	M_TOOLS_SELECTOR
<b>Выбор-в-рамке</b>	1722	M_TOOLS_SEARCH_RECT
<b>Выбор-в-круге</b>	1703	M_TOOLS_SEARCH_RADIUS
<b>Выбор-в-области</b>	1704	M_TOOLS_SEARCH_BOUNDARY
<b>Увеличивающая Лупа</b>	1705	M_TOOLS_EXPAND
<b>Уменьшающая Лупа</b>	1706	M_TOOLS_SHRINK
<b>Сдвиг</b>	1702	M_TOOLS_RECENTER
<b>Информация</b>	1707	M_TOOLS_PNT_QUERY
<b>Геолинк</b>	1736	M_TOOLS_HOTLINK
<b>Подпись</b>	1708	M_TOOLS_LABELER
<b>Линейка</b>	1710	M_TOOLS_RULER
<b>Дубль окна</b>	1734	M_TOOLS_DRAGWINDOW
<b>Символ</b>	1711	M_TOOLS_POINT



Кнопки панели Операции	Номер	Идентификатор
Линия	1712	M_TOOLS_LINE
Полилиния	1713	M_TOOLS_POLYLINE
Дуга	1716	M_TOOLS_ARC
Полигон	1714	M_TOOLS_POLYGON
Эллипс	1715	M_TOOLS_ELLIPSE
Прямоугольник	1717	M_TOOLS_RECTANGLE
Скругленный прямоугольник	1718	M_TOOLS_ROUNDEDRECT
Текст	1709	M_TOOLS_TEXT
Рамка	1719	M_TOOLS_FRAME
Добавить узел	1723	M_TOOLS_ADD_NODE

Вы также можете создавать пользовательские кнопки, которые вызывают определенные процедуры при нажатии на них. Обзор соответствующих возможностей содержит глава [Создание интерфейса пользователя](#). О том, как можно использовать новые кнопки в Интегрированной картографии, рассказывается ниже в этой главе в разделе [Использование Callback-вызовов](#).

## Настройка быстрых меню MapInfo

MapInfo вызывает “быстрые” меню, если пользователь нажимает правую кнопку мышки в окне MapInfo. Эти меню появляются даже в приложениях, созданных при помощи технологий Интегрированной картографии. В зависимости от характера Вашего приложения Вы можете захотеть модифицировать или даже удалить такое меню. Например, Вы, возможно, захотите удалить команду **Дубль окна**, так как эта команда может не работать в приложении, использующем интегрированные Карты.

Чтобы удалить одну или несколько команд из локального меню, используйте оператор Map-Basic **Alter Menu... Remove** или переопределите меню целиком, используя оператор **Create Menu**. Подробнее см. в *Справочнике MapBasic* и *Справочной системе*.

Чтобы добавить команду к локальному меню, используйте оператор MapBasic **Alter Menu... Add** и синтаксис предложений **Calling OLE** или **Calling DDE**, см раздел [Использование Call-back-вызовов](#).

Чтобы удалить “быстрое” меню полностью, используйте оператор Map Basic **Create Menu** и управляющий код “(-” для определения нового меню. Например, следующий оператор разрушает “быстрое” меню для окон Карты:

```
mapinfo.do "Create Menu ""MapperShortcut"" ID 17 As ""(-"""
```

## Вывод на печать интегрированного окна MapInfo Professional

Вы можете использовать оператор **PrintWin** языка MapBasic для вывода окна MapInfo на печать даже в том случае, когда оно переподчинено. Например, Вы можете найти в программе FindZip, поставляемой в комплекте с MapBasic. Меню **Файл** программы FindZip включает в себя команду **Print Map**; при выборе **PrintMap**, программа выполняет следующую процедуру:

```
Private Sub Menu_PrintMap_Click()  
    mapinfo.do "PrintWin"  
End Sub
```

Заметьте, что оператор **PrintWin** печатает Карту на отдельной странице.

Вы также можете использовать оператор MapBasic **Save Window** для сохранения содержимого окна Карты в формате Windows Metafile (WMF). В качестве примера посмотрите типовую программу FindZip: Если пользователь выбирает **PrintForm**, программа создает метафайл с содержимым окна Карты, присоединяет его к форме и затем использует метод Visual Basic **PrintForm**. В результате получается отпечаток формы, содержащей Карту в виде метафайла.

## Обнаружение ошибок времени исполнения

Когда клиентская программа посылает в MapInfo командную строку, возможно возникновение ошибок. Например, команда **Map From World** потерпит неудачу, если таблица World не открыта. MapInfo Professional в этом случае сгенерирует код ошибки.

Чтобы обработать ошибку MapInfo Professional, установите обработчик. В Visual Basic, например, используется оператор **On Error**.

Чтобы определить, какая ошибка произошла в MapInfo, прочтите о свойствах **LastErrorCode** и **LastErrorMessage**, ниже в этой главе в разделе **Объектная модель механизма управления OLE на стр. 236**. Распечатка кодов ошибок приведена в текстовом файле ERRORS.DOC.



Заметьте, что свойство **LastErrorCode** возвращает значения, которые на 1000 больше, чем коды в ERRORS.DOC. Другими словами, если в скомпилированной программе MapBasic возникнет ошибка с кодом 311, то в приложении с интегрированной Картой эта же ошибка будет иметь код **LastErrorCode=1311**.

---

Когда запускается приложение MapBasic (MBX-файл) через механизм OLE Automation, не будет улавливать свои собственные ошибки. Можно запустить MBX используя метод **Do** для запуска оператора MapBasic **Run Application**. Таким образом, если случится ошибка приложения MapBasic внутри MBX, то MBX "повиснет", даже если MBX использует оператор MapBasic **OnError**. Если Вы создали MBX, которые будут вызываться через механизм OLE Automation, попытайтесь сделать MBX проще. Внутри MBX, избегайте использовать оператор MapBasic **OnError**; вместо этого, применяйте другие способы проверки и избегания ошибок перед запуском MBX.

## Завершение программы MapInfo Professional

Если Вы запустили новый экземпляр MapInfo вызовом функции **CreateObject( )**, то этот экземпляр MapInfo завершается автоматически при освобождении соответствующей объектной переменной. Если объектная переменная является локальной, она автоматически освобождается при выходе из локальной процедуры. Для освобождения глобальной объектной переменной необходимо явно присвоить этой переменной значение "Nothing":

```
Set mapinfo = Nothing
```

Если Вы используете для связи с MapInfo динамический обмен данными (DDE), Вы можете завершить исполнение MapInfo, используя DDE метод **LinkExecute** для пересылки командной строки "End MapInfo" из Вашей программы в программу MapInfo.

## Прерывание работы программы на Visual Basic

Если Вы создаете 16-битную программу на Visual Basic, которая использует DDE для связи с MapInfo, убедитесь, что Вы завершаете DDE-сеанс прежде, чем завершается выполнение программы. Если завершить программу Visual Basic, оставив DDE-связи активными, можно получить непредсказуемые результаты, включая сообщения об ошибках исполнения. Эта проблема возникает, если 16-битную программу Visual Basic запустить в 32-битной версии Windows (Windows 2000 или Windows XP). Чтобы избежать этой неприятности, закрывайте DDE-связи до того, как программа завершится.

## Замечание о командных строках MapBasic

Как показано ранее, Вы можете создавать строки в VB-программе, представляющие операторы языка MapBasic, и затем пересылать эти строки в программу MapInfo посредством OLE-метода **Do**. Обратите внимание, что можно объединить два и более оператора в одну командную строку, как показано ниже (в языке Visual Basic символ "&" выполняет конкатенацию строк).

```
Dim msg As String

msg="Open Table ""States"" Interactive "
msg=msg & "Set Next Document Parent " & Frm.hWnd & " Style 1 "
msg=msg & "Map From States "

mapinfo.do msg
```

При обработке командной строки в процессе исполнения MapInfo автоматически определяет, что данная строка содержит три отдельных оператора MapBasic: оператор **Open Table**, оператор **Set Next Document**, и оператор **Map From**. Программа MapInfo Professional способна различать в строке отдельные операторы, так как слова **Open**, **Set** и **Map** являются зарезервированными ключевыми словами языка MapBasic.

Отметьте наличие пробела после ключевого слова **Interactive**. Его присутствие необходимо, так как без этого пробела командная строка содержала бы подстроку "InteractiveSet", не имеющую смысла в синтаксисе языка MapBasic. Поскольку каждая командная строка оканчивается пробелом, MapInfo может определить, что подстроки **Interactive** и **Set** являются отдельными ключевыми словами.

Если Вы объединяете несколько операторов MapBasic в одной командной строке, удостоверьтесь, что они разделены пробелами.

## О диалогах

В приложениях Интегрированной картографии управление кнопкой **OKButton** будет неэффективным. Используйте обычное управление кнопкой и установите переменную, определяющую, нажал ли пользователь эту кнопку.

## О клавишах-акселераторах

В Интегрированной картографии акселераторы MapInfo (например, CTRL+C для копирования) игнорируются. Если Вы хотите, чтобы приложение поддерживало такие сочетания клавиш, то Вы должны определить их внутри клиентской программы.

Переключение режима совмещения узлов нажатием клавиши S поддерживается автоматически.

## Использование Callback-вызовов

Вы можете построить приложение так, чтобы MapInfo автоматически посылало информацию Вашей клиентской программе. Например, можно сделать так, чтобы всякий раз, когда изменяется активное окно Карты, MapInfo Professional вызывало клиентскую программу, чтобы сообщить ID-номер окна. Такой тип уведомления известен как обратный вызов или уведомление (callback).

- Обратные вызовы позволяют MapInfo Professional посылать информацию клиентской программе в следующих случаях:
- Пользователь применяет инструмент в окне. Например, если пользователь производит перемещение объекта мышкой в окне Карты, MapInfo Professional может вызвать Вашу клиентскую программу, чтобы сообщить X- и Y-координаты.
- Пользователь выбирает команду меню. Например, предположим, что приложение настраивает “быстрое” меню MapInfo (меню, возникающее при нажатии правой кнопки мышки). Когда пользователь выбирает команду из этого меню, MapInfo может вызвать клиентскую программу, чтобы сообщить ей о выборе.
- Изменяется окно Карты. Если пользователь изменяет содержание окна Карты (например, изменяя порядок слоев), MapInfo может послать клиентской программе идентификатор этого окна. (Это аналогично процедуре обработчика MapBasic **WinChangedHandler**.)
- Изменяется текст в строке сообщений MapInfo. Строка состояния MapInfo не появляется автоматически в приложениях Интегрированной картографии. Если Вы хотите, чтобы клиентская программа эмулировала строку состояния MapInfo, Вы должны построить приложение так, чтобы MapInfo сообщало вашей клиентской программе об изменениях текста в строке состояния.

## Требования к функциям уведомления

Если Вы планируете использовать обратные вызовы, клиентская программа должна быть способна функционировать, как DDE-сервер или как сервер OLE Automation. Visual Basic 4.0 Professional Edition и C++ могут создавать такие приложения. Приложения, созданные на Visual Basic 3.0, не могут являться серверами OLE Automation, но могут использовать механизм DDE.

## Схема использования уведомлений в OLE

Приведем краткую схему использования повторных вызовов посредством OLE:

1. Используя Visual Basic 4.0, C++, или другой язык, позволяющий создать OLE-сервер, напишите определение класса, включающее один или большее количество методов OLE. Подробности смотрите в документации для языка программирования.
2. Если Вы хотите имитировать строку состояния MapInfo, создайте метод, называемый **SetStatusText**. Определите этот метод так, чтобы у него был один строчный аргумент.
3. Если требуется чтобы программа MapInfo Professional предупреждала программу об изменениях Карт, создайте метод **WindowContentsChanged**. Определите этот метод так, чтобы у него был один аргумент: 4-х битное целое число.
4. Если Вы хотите, чтобы MapInfo сообщала клиентской программе о выборе команды меню или кнопки, напишите один или несколько дополнительных методов с произвольными именами. Каждый из этих методов должен иметь один аргумент: строку.
5. Создайте объект, используя Ваш класс. Например, если Вы назвали его "CMyClass", следующий оператор Visual Basic создает объект этого класса:

```
Public myObject As New CMyClass
```

6. После того, как Ваша программа запустит MapInfo, вызовите метод MapInfo **RegisterCallback** и точно укажите название объекта:

```
mapinfo.RegisterCallback myObject
```

Если Вы хотите, чтобы MapInfo сообщало Вашей клиентской программе, когда пользователь применяет инструментальную кнопку, создайте такую кнопку оператором **Alter ButtonPad... Add**. Определите кнопку в соответствии с *Calling OLE имя\_метода* (используя имя метода, созданного на [шаре 4](#)).

Заметьте, что панели инструментов MapInfo Professional скрыты, подобно остальной части интерфейса пользователя MapInfo. Пользователь не будет видеть новую кнопку. Вы можете добавить иконку, кнопку или другой видимый элемент управления к интерфейсу пользователя Вашей клиентской программы. Когда пользователь укажет на него мышкой, пошлите MapInfo оператор **Run Menu CommandID**, чтобы активизировать этот инструмент.

7. Если Вы хотите, чтобы MapInfo сообщала Вашей клиентской программе, когда пользователь выбирает созданную Вами команду меню, определите такую кнопку оператором **Alter Menu... Add** с указанием имени метода. Определите команду меню в соответствии с *Calling OLE имя\_метода* (используя имя метода, созданного на [шаре 4](#)).
8. Внутри метода обработайте аргументы, посланные MapInfo. Если Вы создали метод **SetStatusText**, MapInfo передает ему строку, содержащую текст строки состояния.
9. Если Вы хотите эмулировать строку состояния MapInfo, напишите код, чтобы поместить этот текст где-нибудь в Вашем интерфейсе пользователя. Если Вы хотите имитировать строку состояния MapInfo, добавьте код к этому методу, который будет показывать текст где-нибудь среди элементов управления в интерфейсе пользователя.
10. Если Вы создали метод **WindowContentsChanged**, MapInfo посылает четырехбайтовое целое число (ID окна MapInfo), чтобы указать, какое из окон Карты изменилось. Напишите код, делающий необходимую обработку. Например, если требуется следить за размером окна Карты, то можно использовать функцию MapInfo Professional **MapperInfo( )**.

11. Если Вы применяете пользовательские кнопки или команды меню, MapInfo посылает строку Вашему приложению, в которой данные разделены запятой. Внутри Вашего метода проанализируйте эту строку. Точный формат строки изменяется в зависимости от того, использовал ли пользователь команду меню, инструмент и т.д. Формат строки описан в разделе **Обработка переданных данных**.

### Обработка переданных данных

Ваше приложение может создавать пользовательские команды меню MapInfo и кнопки MapInfo. Когда пользователь использует команды или кнопки, MapInfo посылает Вашему OLE-методу строку, содержащую восемь элементов, разделенных запятыми. Например, строка, посланная MapInfo, может выглядеть так:

```
MI:-73.5548,42.122,F,F,-72.867702,43.025,202,
```

Содержание такой строки проще понять, если Вы уже знакомы с функцией MapBasic **CommandInfo( )**. Когда Вы пишете MBX-приложения, Вы можете создать новые команды меню и кнопки, вызывающие MapBasic-процедуры. Внутри процедуры обработчика вызовите функцию **CommandInfo( )**, чтобы получить информацию. Например, следующее обращение к функции определяет, держал ли пользователь нажатой клавишу SHIFT при использовании инструмента:

```
log_variable = CommandInfo(CMD_INFO_SHIFT)
```

Код **CMD\_INFO\_SHIFT** определен в файле MAPBASIC.DEF. В следующей таблице приведены эти данные.

Значение	Код событий, связанных с меню	Код событий, связанных с кнопкой панелей инструментов
1		CMD_INFO_X
2		CMD_INFO_Y
3		CMD_INFO_SHIFT
4		CMD_INFO_CTRL
5		CMD_INFO_X2
6		CMD_INFO_Y2
7		CMD_INFO_TOOLBTN
8	CMD_INFO_MENUITEM	

Разъяснение каждого кода смотрите в описании функции **Command Info( )** в *Справочнике MapBasic*.

Когда Вы создаете команду меню или кнопку, которая использует синтаксис вызова *Calling OLE имя\_метода*, MapInfo Professional создает строку, содержащую разделенные запятой все возвращаемые **CommandInfo( )** значения. Строка начинается с префикса "MI": чтобы Ваш OLE-сервер мог определять, что обращение метода было сделано MapInfo Professional.

Строка, которую MapInfo Professional посылает Вашему методу, выглядит следующим образом:

```
"MI:" +  
CommandInfo(1) + "," + CommandInfo(2) + "," +  
CommandInfo(3) + "," + CommandInfo(4) + "," +  
CommandInfo(5) + "," + CommandInfo(6) + "," +  
CommandInfo(7) + "," + CommandInfo(8)
```

Если каждой из созданных кнопок присвоен уникальный номер идентификатора ID, допускается чтобы все кнопки вызывали один и тот же метод. Этот метод сможет определить, от какой кнопки пришел вызов, проверяя седьмой аргумент строки, разделенной запятыми.

После того, как MapInfo Professional передаст методу разделенную запятыми строку, в Вашей воле добавить к методу код, разбирающий строку.

Предположим, что Ваше приложение добавляет команду меню к локальному меню MapInfo. Каждый раз, когда пользователь выбирает команду меню, MapInfo посылает OLE-методу строку. Если команда меню имеет номер 101, строка будет выглядеть следующим образом:

```
"MI:,,,,,,101"
```

В этом случае большинство элементов строки пусто, потому что функция **CommandInfo( )** может возвращать только эту одну часть информации. Из восьми "ячеек" строки, только ячейка номер восемь относится к обработке меню.

Теперь предположим, что Вы создаете кнопку панели MapInfo Professional, которая позволяет пользователю рисовать линии на Карте. Каждый раз, когда пользователь обращается к этому инструменту, MapInfo Professional передает OLE-методу строку с параметрами, разделенными запятыми, строка теперь будет выглядеть следующим образом:

```
"MI:-73.5548,42.122,F,F,-72.867702,43.025,202,"
```

В этом случае, разделенная запятыми строка содержит несколько значений, поскольку **CommandInfo( )** возвращает сведения о событиях с кнопками несколькими фрагментами, которые могут иметь существенное значение. Первые два элемента содержат X- и Y-координаты точки, на которую пользователь указал мышкой; следующие два элемента сообщают, была ли нажата клавиша SHIFT или CTRL; следующие два элемента содержат координаты точки, где пользователь отпустил кнопку мышки; и последний элемент указывает номер идентификатора кнопки. Последняя "ячейка" строки – пустая потому, что эта ячейка относится к событиям в меню, а не с кнопкам.

## Синтаксис C/C++ для функций уведомления

В предыдущем разделе были описаны обратные вызовы в контексте Visual Basic. Здесь мы рассмотрим синтаксис языка C для стандартных уведомлений MapInfo Professional: **SetStatusText** и **WindowContentsChanged**,

Если Вы используете метод MapInfo **SetCallback**, MapInfo может автоматически генерировать обратные вызовы для Вашего объекта **IDispatch**. Обратные вызовы стандарта MapInfo имеют следующий синтаксис:

```
SCODE SetStatusText(LPCTSTR lpszMessage)
```



MapInfo вызывает метод **SetStatusText** всякий раз, когда изменяется содержание строки сообщений в MapInfo. Единственный аргумент – текст сообщения в строке состояний.

```
SCODE WindowContentsChanged(Unsigned Long windowID)
```

MapInfo вызывает метод **WindowContentsChanged** всякий раз, когда изменяется содержание окна Карты. Единственный аргумент представляет собой идентификатор этого окна. Этот повторный вызов аналогичен процедуре MapBasic **WinChangedHandler**.

## Другие способы использования OLE-уведомлений

Как говорилось ранее, MapInfo может использовать обратные вызовы OLE, чтобы послать информацию клиентской программе. В некоторых случаях, однако, Вы должны применять повторные вызовы, которые не используют OLE. Например, если Вы пишете программы в Visual Basic 3.0, Вы не можете использовать OLE, потому что Visual Basic 3.0 не позволяет создавать собственные серверы OLE Automation.

MapInfo Professional поддерживает два типа уведомлений, которые не используют механизм OLE: использующие DDE и приложения MapBasic (файлы MBX).

### Обратные вызовы DDE

Когда Вы создаете кнопки на инструментальной панели или команды меню, Вы указываете предложение **Calling**. Чтобы пользоваться обратным вызовом DDE, используйте синтаксис вызова *Calling DDE сервер, программа*. Всякий раз, когда пользователь использует кнопку или команду меню, MapInfo открывает DDE-связь с DDE-сервером, и затем посылает строку объекту. Строка использует формат, обсужденный в предыдущем разделе **Обработка переданных данных на стр. 230** (например, "MI:,,,,,, 101 ").

Пример смотрите в программе FindZip. Процедура "Form Load" посылает MapInfo оператор **Alter ButtonPad... Add**, чтобы создать новую панель инструментов.

Новое определение панели инструментов содержит следующее выражение:

```
Calling DDE "FindZip", "MainForm"
```

Всякий раз, когда пользователь применяет инструмент, MapInfo открывает DDE связь с программой FindZip и посылает строку "Main Form" объекту. ("MainForm" – значение свойства формы "LinkTopic"). Подробную информацию содержит глава **Работа в интегрированной среде разработки программ**.

### Обратные вызовы MBX

Если Вы создаете приложение MapBasic (файл MBX), Вы можете сконструировать Ваши кнопки и команды меню так, чтобы они вызвали MapBasic процедуры в MBX. В предложении вызова **Calling** используйте синтаксис *Calling процедура* (где "процедура" – это имя программы MapBasic). После того, как Ваше приложение на языке Visual Basic запустит MapInfo, запустите MBX, послав MapInfo строку **Run Application**. Например:

```
mapinfo.do "Run Application ""C:\MB\MYAPP.MBX"" "
```

О создании кнопок и меню рассказывает глава **Создание интерфейса пользователя**.



## Справочная система

Приложение с Интегрированной Картой может активировать окна диалога MapInfo посредством OLE-метода **RunMenuCommand**. Если Ваше приложение, таким образом, активизирует окно диалога MapInfo, Вы можете контролировать доступность справки для этого окна.

### Вызов стандартного справочного файла MapInfo Professional

Вы можете разрешить Вашим пользователям вызов стандартного справочного файла MapInfo из диалогового окна. Такая конфигурация принимается по умолчанию. Если пользователь нажимает клавишу F1 в активном окне диалога MapInfo, *Справочная система* выводит на дисплей соответствующий раздел стандартного справочного файла MapInfo Professional MAP-INFOW.CHM.



После вывода на дисплей окна Справки MapInfo пользователь может щелкать по различным навигационным кнопкам для просмотра справочного файла. Пользователи могут посчитать такую конфигурацию неудобной, так как в справочном файле MapInfo описан пользовательский интерфейс MapInfo, а не пользовательский интерфейс Вашего приложения с Интегрированной Картой.

---

### Запрещение вызова Справочной системы

Вы можете полностью запретить вызов *Справочной системы* для диалоговых окон MapInfo Professional выдачей следующего оператора языка MapBasic:

```
Set Window Help Off
```

После исполнения оператора **Set Window Help Off** нажатие клавиши F1 в активном окне диалога MapInfo игнорируется.

### Вызов пользовательского справочного файла

Вы можете сконфигурировать MapInfo для вызова *Справочной системой* справочного файла, созданного пользователем. Например, следующая строка кода указывает MapInfo на необходимость использования файла CUSTOM.CHM при вызове *Справочной системы*:

```
Set Window Help File "CUSTOM.CHM" Permanent
```

После исполнения оператора **Set Window Help File...Permanent** нажатие клавиши F1 приводит к вызову приложением MapInfo *Справочной системы* Windows, но при этом на дисплей выводится заданный Вами справочный файл вместо справочного файла MAP-INFOW.CHM. Используйте такую конфигурацию, если Вы хотите обеспечить справку для одного или нескольких окон диалога MapInfo Professional, но при этом не желаете предоставлять пользователю доступ ко всему содержимому стандартного Справочного файла MapInfo.

Если Вы хотите предоставить специализированную справку для окон диалога MapInfo, Вы должны обеспечить соответствие контекстных ID-номеров (Context ID) в Вашем справочном файле ID-номерам диалоговых окон MapInfo.

Для определения ID-номера диалогового окна MapInfo:

1. Запустите MapInfo с аргументом `-helpdiag` в командной строке.
2. Активируйте диалог MapInfo, для которого Вы хотите создать справку.
3. Нажмите F1. Вследствие использования аргумента `-helpdiag` вместо отображения справочного файла MapInfo покажет ID-номер диалога, который Вам следует записать.
4. Используя программное обеспечение создания справочных файлов, отредактируйте Ваш справочный файл таким образом, чтобы ID-номер темы в Вашем файле совпадал с ID-номером соответствующего окна диалога MapInfo Professional.

Например, диалоговое окно MapInfo **Найти** имеет ID-номер 2202. Если Вы хотите обеспечить вывод Вашей собственной справки для этого окна, присвойте значение 2202 контекстному ID-номеру (Context ID) соответствующей темы Вашего справочного файла.

Отметим следующие моменты:

- В комплект поставки языка MapBasic не включен компилятор справочных файлов. Его можно свободно получить у Microsoft по адресу:  
<http://www.microsoft.com/downloads/details.aspx?FamilyID=00535334-C8A6-452F-9AA0-D597D16580CC&displaylang=en>
- ID-номера диалоговых окон MapInfo Professional могут быть изменены в будущих версиях программы.

## Полезные операторы и функции языка MapBasic

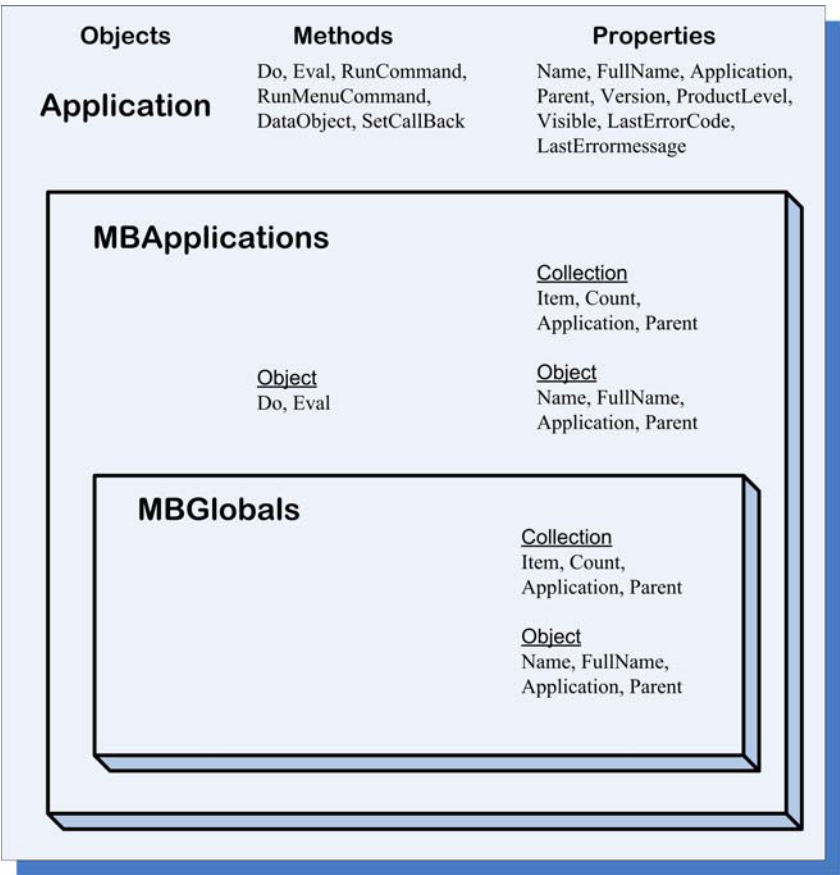
В этом разделе перечислены некоторые операторы и функции языка MapBasic, применение которых особенно полезно в приложениях с Интегрированной Картой. Детальное обсуждение этих операторов и функций смотрите в *Справочнике MapBasic* или *Справочной системе*.

Имя оператора или функции	Описание
<b>Create Legend</b>	Создает новое окно Легенды.
<b>Map</b>	Создает новое окно Карты.
<b>MenuItemInfoByID( )</b> <b>MenuItemInfoByHandler( )</b>	Возвращает статус команды меню в MapInfo Professional (отмечена галочкой или нет).
<b>Open Table</b>	Открывает таблицы MapInfo Professional.
<b>RemoteQueryHandler( )</b>	Позволяет MapBasic-программам обрабатывать запросы на чтение от DDE-клиентов.
<b>Run Menu Command</b>	Имитирует выбор пользователем команды меню MapInfo или кнопки на панели инструментов (ButtonPad).

Имя оператора или функции	Описание
<b>SearchPoint( ), SearchRect( )</b>	Осуществляют поиск в выбранных слоях окна Карта объектов в точке с заданными координатами (x,y) или в заданной прямоугольной области соответственно. Эти функции позволяют Вам эмулировать инструменты MapInfo Professional <b>Информация</b> и <b>Подпись</b> .
<b>SearchInfo( )</b>	Возвращает информацию о результатах исполнения функций <b>SearchPoint( )</b> и <b>SearchRect( )</b> .
<b>Set Application Window</b>	Обеспечивает переподчинение окон диалога и сообщений об ошибках MapInfo. Применяйте этот оператор в клиентской программе после запуска MapInfo Professional или подключения к работающему экземпляру MapInfo Professional.
<b>Set Map</b>	Управляет различными режимами представления окон типа Карты.
<b>Set Next Document</b>	Переподчиняет окно MapInfo-документа (например, окно Карты или Легенды) так, что оно становится порожденным окном программы-клиента.
<b>Set Window</b>	Управляет различными режимами представления окон MapInfo Professional.
<b>Shade, Set Shade</b>	Создает или изменяет тематические слои Карты.
<b>SystemInfo( )</b>	Некоторые значения, возвращаемые функцией <b>SystemInfo( )</b> , предназначены специально для Интегрированной картографии.  Пример: задайте в качестве аргумента <b>SYS_INFO_APPLICATIONWND</b> , чтобы получить Hwnd-идентификатор приложения.
<b>WindowID( ), WindowInfo( )</b>	Возвращают информацию об окнах MapInfo, в том числе переподчиненных.

## Объектная модель механизма управления OLE

На следующей диаграмме приведена схема библиотеки типов MapInfo Professional OLE Automation. Методы и свойства (Properties) детально описываются на следующих страницах.

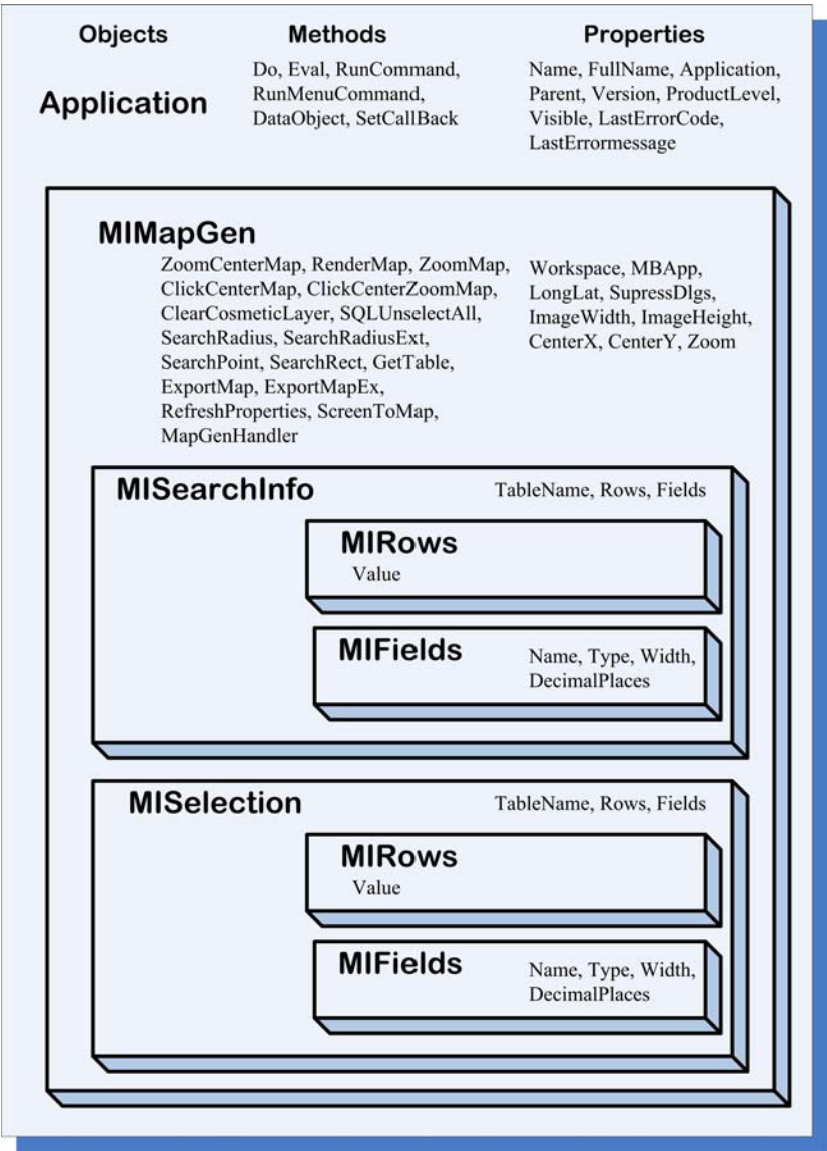


Объект **Application** (приложение) представляет работающий экземпляр MapInfo Professional.

Семейство **MBAApplications** (MapBasic-программы) представляет список MapBasic-программ, работающих в данный момент

Семейство **MBGlobals** (глобальные переменные в среде MapBasic) представляет список глобальных переменных, объявленных одной из работающих MapBasic-программ.

На следующей диаграмме перечислены дополнительные объекты библиотеки типов MapInfo Professional OLE Automation. Методы и свойства (Properties) детально описываются на следующих страницах.



## Использование объектной модели OLE в процессах MapInfo Professional

Объектная модель OLE Automation изначально предназначалась для использования в качестве отдельного процесса, вызываемого из клиентского приложения. Однако, Вы можете также пользоваться этой объектной моделью для выполнения команд из DLL-библиотеки, вызванной программой MapBasic. Если Вы захотите воспользоваться этой возможностью, не забывайте о следующих особенностях:


- Вы должны посылать команды процессу MapInfo Professional, в который загружалась Ваша DLL-библиотека. Это значит, что Вы не обратитесь к нужному OLE-объекту, выполнив вызов функции **CreateObject( )**, потому что это создаст второй экземпляр MapInfo Professional. Вы должны вместо этого использовать OLE-объект, представляющий текущий процесс MapInfo Professional. Чтобы определить его, воспользуйтесь MapBasic-функцией **SystemInfo** с атрибутом **SYS\_INFO\_APPDISPATCH**. Эта функция вернет целое число, представляющее указатель **IDispatch** на первичный OLE-объект. Далее, Вы передаете это значение из программы MapBasic в DLL и преобразуете его в переменную OLE, к которой применимы методы типа **Do** и **Eval** (конкретный механизм превращения указателя в OLE-объект зависит от языка программирования, на котором написана библиотека DLL).
- Если Вы хотите использовать обратные уведомления (callback notifications), зарегистрируйте объект обратных уведомлений методом **RegisterCallback**, но не более ранним **SetCallback**. Если Вы используете **SetCallback**, то может возникнуть ситуация, когда какая-то другая программа, запущенная изнутри MapInfo Professional, воспользуется участком памяти, зарезервированным за обратными уведомлениями. Метод **RegisterCallback** исключает этот риск.

### Свойства объекта Application

В следующей таблице перечислены свойства, применимые к объекту **Application**. Все свойства в этой таблице разрешены только для чтения, за исключением свойств **Visible** и **LastErrorCode**.

Свойства объектов приложения	
Имя свойства	Функциональность
Name	Возвращает имя приложения (например, "MapInfo Professional"). Стандартное свойство OLE. Свойство по умолчанию для объекта <b>Application</b> .
FullName	Возвращает полный маршрут к исполняемому файлу приложения. Стандартное свойство OLE.
Application	Возвращает значение <b>IDispatch</b> объекта <b>Application</b> . Стандартное свойство OLE.
Parent	Возвращает значение <b>IDispatch</b> порождающего объекта; для объекта <b>Application</b> возвращает себя самого. Стандартное свойство OLE.

Свойства объектов приложения (продолжение)

Имя свойства	Функциональность
<b>Version</b>	Возвращает текстовое представление номера текущей версии, умноженного на 100 (например, MapInfo Professional 9.0.0 возвращает "900").
<b>ProductLevel</b>	Возвращает целое значение, обозначающее "уровень" программы MapInfo. Для MapInfo Professional возвращает 200.
<b>Visible</b>	Логическое значение, контролирующее видимость окна приложения. Относится к категории свойств разрешение для записи/защиты от записи. Считывание позволяет определить видимость окна; присваивание свойству значения устанавливает видимость или невидимость окна.
<b>LastErrorCode</b>	<p>Целое значение, указывающее кодовый номер последней MapBasic-ошибки, случившейся в процессе вызова метода <b>Do</b>, <b>Eval</b> или <b>RunCommand</b>.</p> <hr/> <p> Коды ошибок, возвращаемые этим свойством, превышают на 1000 соответствующие коды ошибок в среде MapBasic.</p> <hr/> <p>Эти коды ошибок автоматически заменяются кодом следующей ошибки, но никогда автоматически не сбрасываются в 0. Относится к категории свойств разрешение для записи/защиты от записи.</p>
<b>LastErrorMessage</b>	Возвращает строку, представляющую текст сообщения об ошибке, соответствующей значению <b>LastErrorCode</b> .

Методы объектов приложения

Имя метода	Функциональность
<b>Do( string )</b>	Интерпретирует строку "string" как оператор языка MapBasic, затем выполняет этот оператор.
<b>Eval( string )</b>	Интерпретирует строку "string" как MapBasic-выражение, вычисляет выражение и возвращает его значение. Если выражение приводится к логическому значению (тип Logical), MapInfo возвращает односимвольную строку, "T" или "F" соответственно.

Методы объектов приложения (продолжение)

Имя метода	Функциональность
<b>RunCommand</b> ( <i>string</i> )	Интерпретирует строку "string" как оператор языка MapBasic; является синонимом "Do".
<b>RunMenuCommand</b> ( <i>menuid</i> )	<p>Исполняет команду меню, указанную аргументом "menuid" типа "Integer". Пример смотрите ниже.</p> <p>Этот метод активирует стандартную команду меню или кнопку панели инструментов; для того чтобы активировать специально разработанную команду меню или кнопку, используйте метод Do, использующий оператор <b>Run Menu Command ID</b>.</p>
<b>DataObject</b> ( <i>windowID</i> )	<p>Возвращает интерфейс <b>IUnknown</b>, представляющий это окно. Для получения графического представления окна в виде метафайла используйте <b>QueryInterface</b> для интерфейса <b>IDataObject</b>.</p> <p>Только значения IDataObject и IUnknown разрешены для этого объекта.</p> <div> Это "продвинутая" возможность для программистов, использующих язык программирования C.</div>
<b>SetCallback</b> ( <i>IDispatch</i> )	<p>Регистрирует объект механизма управления объектами OLE Automation в качестве "приемника" уведомлений, которые будет создавать MapInfo Professional. Только одна функция уведомления может быть зарегистрирована в каждый момент.</p> <p>Использование этого метода считается устаревшим. Лучше использовать методы <b>RegisterCallback</b> и <b>UnregisterCallback</b>.</p>
<b>RegisterCallback</b> ( <i>IDispatch</i> )	<p>Регистрирует объект механизма управления объектами OLE Automation в качестве "приемника" уведомлений, которые будет создавать MapInfo Professional. Используйте этот метод, если Вы регистрируете механизм уведомлений изнутри процесса MapInfo Professional (например, из DLL-библиотеки, вызванной из MapBasic). Такой подход гарантирует, что Ваш уведомляющий объект не будет конфликтовать с другими приложениями, запущенными изнутри процесса MapInfo Professional.</p>



Методы объектов приложения (продолжение)

Имя метода	Функциональность
<b>UnregisterCallback</b> ( <i>IDispatch</i> )	Отменяет регистрацию объекта механизма управления объектами OLE Automation, которая была объявлена с использованием метода <b>RegisterCallback</b> . Вы должны передать те же аргументы, что и при вызове <b>RegisterCallback</b> .
<b>SetCallbackEvents</b> ( <i>IDispatch</i> , <i>eventFlags</i> )	По умолчанию MapInfo Professional вызовет все корректные методы уведомления. С помощью этого метода Вы можете выбирать, какие виды уведомлений будут посланы уведомляемому объекту. Например, если Ваш объект использует методы <b>WinContentsChanged</b> и <b>SetStatusText</b> , но в некоторых ситуациях Вам нужно уведомлять только <b>SetStatusText</b> , Вы можете вызвать <b>SetCallbackEvents</b> ( <i>&lt;dispatch id&gt;</i> , <i>CallbackEvents.WindowChanged</i> ), чтобы MapInfo Professional посылала только уведомления об изменениях окна. См. ниже детали вызова <b>CallbackEvents</b> .
<i>IDispatch*</i> <b>RegisterDockWindow</b> ( <i>HWND hwnd</i> , <i>long domainId</i> )	Регистрирует данное окно в MapInfo Professional и возвращает объект, представляющий собой прикрепленное (docked) окно. Вызов этой функции также приводит к созданию прикрепляемого окна и размещению его в стандартном положении. Методами <b>Dock</b> и <b>Float</b> можно затем изменить положение и статус окна.  Аргумент <b>domainId</b> можно использовать только если Вы обращаетесь к COM API напрямую из кода .Net. В этом случае переменная представляет id-номер текущего объекта <b>AppDomain</b> .
<i>void</i> <b>UnregisterDockWindow</b> ( <i>IDispatch* dockWindow</i> )	Отменяет регистрацию прикрепляемого окна <b>dockWindow</b> . Дальнейшие попытки обратиться к объекту <b>DockWindow</b> порождают ошибку.

Например, следующая VB-инструкция использует метод **Do** для передачи программе MapInfo оператора **Map**, открывающего окно Карты:

```
mapinfo.Do "Карта Мира"
```

Следующая инструкция использует метод **RunMenuCommand** для выполнения команды меню MapInfo, имеющей значение кода 1702, которая выбирает MapInfo-инструмент **Сдвиг**. (Для определения конкретного значения кода интересующей Вас команды меню смотрите файл MENU.DEF или раздел **Интеграция панелей инструментов MapInfo Professional на стр. 223.**)

```
mapinfo.RunMenuCommand 1702
```

**CallbackEvents** использует следующие элементы для извещений:

- **None**,
- **MenuItem**,
- **WindowChanged**,
- **SetStatusText**.


Вы можете задавать эти флаги в произвольном порядке при обращении к **SetCallbackEvents**. Чтобы вызывать более одного такого элемента, их нужно разделять знаком "|". Например, чтобы вызвать в C++ уведомление **WindowContentsChanges** и **SetStatusText**, нужно задать **SetCallbackEvents( <dispatch id>, (int ) (WindowChanged | SetStatusText) )**.

### Свойства объекта DockWindow

**DockWindow** – это COM-интерфейс, представляющий прикрепляемое окно. В следующей таблице перечислены свойства, применимые к объекту **Application**.

Свойства объекта DockWindow	
Свойство	Описание
BOOL <b>Active</b>	Содержит индикатор видимости прикрепляемого окна.
long <b>id</b>	Содержит числовой идентификатор прикрепляемого окна.
BOOL <b>Closed</b>	Содержит индикатор закрытости прикрепляемого окна.
DockPosition <b>DockPosition</b>	Содержит значение положения прикрепляемого окна. Окно может быть прикреплено к одной из сторон экрана, а может быть неприкрепленным. Если окно не прикреплено, возвращается значение в предыдущем прикрепленном положении.  См. раздел <b>Значения DockPosition на стр. 244</b> .
int <b>DockSizeCX</b>	Содержит ширину окна, прикрепленного к левой или правой стороне экрана приложения. Если окно не прикреплено, возвращается значение в предыдущем прикрепленном положении.
int <b>DockSizeCY</b>	Содержит высоту окна, прикрепленного к верхней или нижней стороне экрана приложения. Если окно не прикреплено, возвращается значение в предыдущем прикрепленном положении.
BOOL <b>Floating</b>	Содержит индикатор неприкрепленного (плавающего) состояния прикрепляемого окна.

Свойства объекта DockWindow

Свойство	Описание
BOOL Pinned	Содержит индикатор свернутого состояния прикрепляемого окна. Когда окно свернуто, оно представлено небольшим ярлыком с именем окна около того края экрана, где оно было прикреплено. Если Вы помещаете указатель мыши на этот ярлык, прикрепленное окно разворачивается. Как только указатель перемещается вовне окна, оно сворачивается.
BSTR Title	Содержит и позволяет менять заголовок прикрепляемого окна.
LastErrorCode	<p>Целое значение, указывающее кодовый номер последней MapBasic-ошибки, случившейся в процессе вызова метода <b>Do</b>, <b>Eval</b> или <b>RunCommand</b>.</p> <hr/> <p> Коды ошибок, возвращаемые этим свойством, превышают на 1000 соответствующие коды ошибок в среде MapBasic.</p> <hr/> <p>Эти коды ошибок автоматически заменяются кодом следующей ошибки, но никогда автоматически не сбрасываются в 0. Относится к категории свойств разрешение для записи/защиты от записи.</p>
LastErrorMessage	Возвращает строку, представляющую текст сообщения об ошибке, соответствующей значению <b>LastErrorCode</b> .

Методы объекта DockWindow

Метод	Описание
void <b>Activate</b> ( )	Делает окно активным и передает в него фокус.
void <b>Close</b> ( )	Закрывает окно. Показать его снова можно, вызвав метод <b>Activate</b> ( ). Уничтожить окно можно вызовом <b>IMapInfo.UnregisterDockWindow</b> .
void <b>Dock</b> ( <b>DockPosition</b> , <b>cx</b> , <b>cy</b> )	Прикрепляет окно к одной из сторон экрана. См. раздел <b>Значения DockPosition на стр. 244</b> .
void <b>Float</b> ( <b>left</b> , <b>top</b> , <b>right</b> , <b>bottom</b> )	Делает окно плавающим и задает его размер и положение.

Методы объекта DockWindow

Метод	Описание
void FloatSize (*left, *top, *right, *bottom)	Возвращает размер и положение плавающего окна. Если окно прикреплено, возвращает последние значения для окна в плавающем положении.
void Pin( )	Сворачивает окно и прячет его за один из краев экрана.

Значения DockPosition

**DockPosition** использует следующие значения:

- **PositionFloat**,
- **PositionLeft**,
- **PositionTop**,
- **PositionRight**,
- **PositionBottom**.

Эти значения применяются в свойствах и методах объекта **DockWindow**.

Свойства семейства MBAplications

Семейство **MBAplications** включает в себя все MapBasic-приложения, работающие в среде MapInfo Professional в данный момент. Все свойства, перечисленные в следующей таблице, разрешены только для чтения.

Свойства семейства MBAplications

Имя свойства	Функциональность
<b>Item</b>	Возвращает значение <b>IDispatch</b> конкретного объекта "programobject". Аргумент имеет тип VARIANT, который может приводиться к целочисленному индексу в диапазоне от 1 до <b>Count</b> (количества объектов) или строковому значению (имя программы). Свойство по умолчанию для коллекции <b>MBAplications</b> .
<b>Count</b>	Возвращает длинное целое число объектов в семействе (т.е. число работающих приложений).
<b>Application</b>	Возвращает значение <b>IDispatch</b> приложения MapInfo Professional. Стандартное свойство OLE.
<b>Parent</b>	Возвращает значение <b>IDispatch</b> порождающего объекта; для семейства <b>MBAplications</b> это приложение MapInfo Professional. Стандартное свойство OLE.

## Свойства объекта в семействе MBAApplications

Каждый объект в семействе **MBAApplications** является работающим MapBasic-приложением. Все свойства, перечисленные в следующей таблице, разрешены только для чтения.

Свойства объекта семейства MBAApplications

Имя свойства	Функциональность
<b>Name</b>	Возвращает имя приложения (например, "FOO.MBX"). Стандартное свойство OLE. Свойство по умолчанию для объекта <b>MBAApplication</b> .
<b>FullName</b>	Возвращает полный маршрут к исполняемому MBX-файлу MapBasic-приложения. Стандартное свойство OLE.
<b>Application</b>	Возвращает значение <b>IDispatch</b> MapBasic приложения. Стандартное свойство OLE.
<b>Parent</b>	Возвращает значение <b>IDispatch</b> порождающего объекта; это приложение MapInfo Professional. Стандартное свойство OLE.

Например, следующий оператор определяет имя работающей MapBasic-программы:

```
Dim appsList As Object
Dim firstname As String

Set appsList = mapinfo.MBAApplications
If appsList.Count > 0 Then
    firstname = appsList(1).Name
End If
```

Методы объекта семейства MBAApplications

Имя метода	Функциональность
<b>Do( string )</b>	Строка string пересылается в процедуру <b>RemoteMsgHandler</b> MapBasic-программы.
<b>Eval( string )</b>	Строка string передается как аргумент в функцию <b>RemoteQueryHandler( )</b> MapBasic-приложения; возвращается значение функции <b>RemoteQueryHandler</b> .  <b>RemoteQueryHandler( )</b> обязательно следует определять как функцию, возвращающую строку. Если выражение приводится к логическому значению (тип Logical), MapInfo возвращает односимвольную строку, "T" или "F" соответственно.

## Свойства коллекции MBGlobals

Семейство MBGlobals включает в себя все глобальные переменные в среде MapBasic, объявленные конкретным работающим MapBasic приложением. Все свойства, перечисленные в следующей таблице, разрешены только для чтения.

Свойства коллекции MBGlobals

Имя свойства	Функциональность
<b>Item</b>	Возвращает значение <b>IDispatch</b> конкретного объекта "mbglobal". Аргумент имеет тип VARIANT, который может приводиться к целочисленному индексу в диапазоне от 1 до <b>Count</b> (количества объектов) или строковому значению (имя глобальной переменной). Свойство по умолчанию для коллекции <b>MBGlobals</b> .
<b>Count</b>	Возвращает длинное целое, число объектов в семействе (коллекции) (число глобальных переменных).
<b>Application</b>	Возвращает значение <b>IDispatch</b> приложения MapInfo Professional. Стандартное свойство OLE.
<b>Parent</b>	Возвращает значение <b>IDispatch</b> порождающего объекта; это объект "programobject". Стандартное свойство OLE.

## Свойства объекта в семействе MBGlobals

Каждый объект в семействе **MBGlobals** является глобальной переменной MapBasic-приложения. Все свойства, кроме свойства **Value**, перечисленные в следующей таблице, защищены от записи.

Свойства объекта в семействе MBGlobals

Имя свойства	Функциональность
<b>Value</b>	Чтение/запись. Чтение, чтобы получить строку определяющую значение глобальной переменной MapBasic; запись свойства, чтобы изменить значение переменной. Свойство по умолчанию для объекта <b>MBGlobal</b> .
<b>Name</b>	Возвращает имя переменной. Стандартное свойство OLE.
<b>Type</b>	Возвращает строку текста, описывающего тип переменной из списка стандартных типов MapInfo Professional ("Integer", "Date", и т.п.).
<b>Application</b>	Возвращает значение <b>IDispatch</b> MapBasic приложения. Стандартное свойство OLE.
<b>Parent</b>	Возвращает <b>IDispatch</b> порождающего объекта; для объекта MBglobal, это "programobject", в котором объявлена глобальная переменная. Стандартное свойство OLE.

В следующем примере на Visual Basic, сначала исследуется, а затем изменяется значение глобальной переменной "g\_status" программы MapBasic.

```
Dim globinfo As Object
Dim old_value As Integer

' Получаем список глобальных переменных,
' используемых приложением MapBasic:
Set globinfo = mapinfo.MBApplications(1).MBGlobals

' Получаем значение глобальной переменной,
' обращаясь к свойству "Value":
old_value = globinfo("g_status").Value

' Присвоим новое значение глобальной переменной:
globinfo("g_status") = old_value + 1
```

Выражение globinfo("g\_status") эквивалентно globinfo("g\_status").Value, потому что Value – стандартное свойство.

Свойства объектов MIMapGen

В следующей таблице перечислены свойства, применимые к объекту **MIMapGen**. Объект **MIMapGen** используется в основном в приложениях MapInfo ProServer; но и приложения MapInfo Professional могут использовать объект **MIMapGen**. Примеры использования объектной модели **MIMapGen** смотрите в документации к MapInfo ProServer.

Свойства объектов MIMapGen	
Имя свойства	Функциональность
Workspace	Маршрут к файлу Рабочего набора MapInfo. Если он задан, MapInfo Professional загружает Рабочий набор.
MBApp	Маршрут к исполняемому приложению MapBasic (MBX-файлу). Если он задан, MapInfo Professional запускает MBX.
LongLat	Логическое значение; определяет интерфейс координатной системы. Если TRUE, все значения, которые Вы вводите или получаете (используя <b>CenterX</b> и <b>CenterY</b> ) представляют широту и долготу. Если FALSE, то будет использоваться координатная система окна Карты.
SuppressDlg	Логическое значение; если TRUE, то вызывается диалог с сообщением об ошибке. Сюда включаются диалоги, действующие как результат оператора <b>Run Menu Command</b> .
ImageWidth	Ширина изображения, в пикселах.
ImageHeight	Высота изображения в пикселах.
CenterX	Х-координата (Долгота) центра Карты.

Свойства объектов MIMapGen (продолжение)

Имя свойства	Функциональность
CenterY	Y-координата (Широта) центра Карты.
Zoom	Ширина Карты (в единицах расстояния, например, количество миль). Это число показывается в единицах измерения, используемых в окне Карты (например, мили, километры).

Обратите внимание, что корректное задание Рабочего набора – это первый шаг в использовании объекта **MIMapGen**. **MIMapGen** настроен на работу в ситуации, когда есть одно окно Карты (то есть, когда Web-страница показывает одну Карту). Что бы начать использовать **MIMapGen**, определите настройки Рабочего набора, так, чтобы MapInfo загрузила такой Рабочий набор, который содержит одно окно Карты. Тогда Вы сможете использовать другие методы и возможности работы с окном Карты.



## Методы объекта MIMapGen


Следующие методы применяются к объекту MIMapGen.

### Методы работы с объектом MIMapGen

Метод	Функциональность
<b>ZoomCenterMap</b> ( )	Перерисовывает Карту, основываясь на текущих значениях <b>CenterX</b> , <b>CenterY</b> и <b>Zoom</b> . Карта перерисовывается, только если центр или масштаб изменились с момента последней прорисовки Карты
<b>RenderMap</b> ( )	Имеет тоже действие, что и <b>ZoomCenterMap</b> , кроме того, что Карта всегда перерисовывается.
<b>ZoomMap</b> (double <i>ZoomFactor</i> )	Изменяет масштаб Карты, в соответствии с указанием фактора масштаба. Положительное значение увеличивает масштаб; отрицательное уменьшает.
<b>ClickCenterMap</b> (long <i>MouseX</i> , long <i>MouseY</i> )	Перемещает центр Карты в зависимости от места, где будет Аргументы X/Y отражают положение на Карте в пикселах.
<b>ClickCenterZoomMap</b> (long <i>MouseX</i> , long <i>MouseY</i> , double <i>ZoomFactor</i> )	Перемещает центр Карты в зависимости от места, где будет щелчок мыши и изменяет масштаб в зависимости от масштабного фактора.
<b>ClearCosmeticLayer</b> ( )	Тот же эффект, что и от команды меню <b>Карта &gt; Удалить Косметику</b> .
<b>SQLUnselectAll</b> ( )	Тот же эффект, что и от команды меню <b>Запрос &gt; Отменить выбор</b> .
<b>SearchRadius</b> (double <i>CenterPointX</i> , double <i>CenterPointY</i> , double <i>Radius</i> )	Проводит операцию поиска в круге.
<b>SearchRadiusExt</b> (double <i>CenterPointX</i> , double <i>CenterPointY</i> , double <i>OuterPointX</i> , double <i>OuterPointY</i> )	Проводит операцию поиска в круге; радиус задается точкой на окружности.
<b>SearchPoint</b> (double <i>CenterPointX</i> , double <i>CenterPointY</i> )	Проводит операцию поиска в небольшой области вокруг указанного места.

Методы работы с объектом MIMapGen (продолжение)

Метод	Функциональность
<b>SearchRect</b> (double <i>x1</i> , double <i>y1</i> , double <i>x2</i> , double <i>y2</i> )	Проводит операцию поиска в прямоугольной области.
<b>GetTable</b> (string <i>Tablename</i> )	Возвращает объект <b>MISelection (IDispatch)</b> ; чтобы получить доступ к содержимому таблицы, используйте объект <b>MISelection</b> .
<b>ExportMap</b> (string <i>ImageType</i> , string <i>FileSpec</i> )	Создаёт растровый файл (JPEG, TIFF, PNG, PSD, BMP, WMF или GIF) окна Карты. Смотрите описание оператора MapBasic <b>Save Window</b> .
<b>ExportMapEx</b> (string <i>ImageType</i> , string <i>FileSpec</i> , string <i>CopyrightInfo</i> )	Создаёт растровый файл (JPEG, TIFF, PNG, PSD, BMP, WMF или GIF) окна Карты. Смотрите описание оператора MapBasic <b>Save Window</b> .
<b>RefreshProperties( )</b>	Обновляет параметры <b>CenterX</b> , <b>CenterY</b> , <b>Zoom</b> , <b>ImageHeight</b> и <b>ImageWidth</b> .
<b>ScreenToMap</b> (long <i>ScreenX</i> , long <i>ScreenY</i> , double <i>MapX</i> , double <i>MapY</i> )	Преобразует координаты экрана (пиксели) в координаты Карты (типа "Широта/Долгота").
<b>MapGenHandler</b> (string <i>Message</i> )	Вызывает процедуру <b>RemoteMapGenHandler</b> в приложении MBX, которая выполняется через <b>MBApp</b> . Используйте этот метод для запуска операторов MapBasic в файле MBX.

 Поисковые методы ищут только на самом верхнем слое. Для получения доступа к результатам поиска, смотрите описание объекта **MISearchInfo**.

Свойства объекта MIMapGen

Следующие свойства применяются к объекту **MISearchInfo**.

Свойства объекта **MISearchInfo**

Свойство	Функциональность
<b>Rows</b>	Это свойство возвращает коллекцию строк (объектов <b>MIRow</b> ). Эта коллекция представляет результаты поиска.
<b>Fields</b>	Это свойство возвращает коллекцию полей (объектов <b>MIField</b> ). Эта выборка представляет установки определенных полей (имена полей и др.) описывающих результат поиска.
<b>TableName</b>	Строка, имя таблицы, которая содержит результаты поиска.


Для получения объекта **MISearchInfo**, используйте методы поиска объектов **MIMapGen**: **SearchRadius**, **SearchRadiusExt**, **SearchPoint** или **SearchRect**.

Метод объекта **MIRow**

Следующий метод применяется к объекту **MIRow**. Каждый объект **MIRow** представляет одну запись, возвращаемую поисковым методом или одну строку в таблице определенной в методе поиска **GetTable**.

Метод объекта **MIRow**

Метод	Функциональность
<b>Value</b>	Возвращает указатель на значения для данной колонки, которая определяется использованием аргумента "arg". Допустимы следующие варианты VT_12, VT_14 и VT_BSTR (где VT_BSTR это имя колонки).

 Что бы получить коллекцию объектов **MIRow**, сошлитесь на свойства **Rows** объекта **MISearchInfo** или объекта **MISelection**.

Свойства объекта **MIField**

Следующие свойства применяются к объекту **MIField**. Каждый объект **MIField** описывает одну из колонок в последних результатах поиска или одну из колонок в таблице, определенной в методе вызова **GetTable**.

Свойства объекта **MIField**

Свойство	Функциональность
<b>Name</b>	Строка, имя колонки.
<b>Type</b>	Короткое целое: тип данных поля. Допустимы следующие значения: <ul style="list-style-type: none"><li>• (1) DT_CHAR</li><li>• (2) DT_DECIMAL</li><li>• (3) DT_INTEGER,</li><li>• (4) DT_SMALLINT</li><li>• (5) DT_TIME</li><li>• (6) DT_LOGICAL</li><li>• (8) DT_FLOAT.</li></ul>
<b>Width</b>	Короткое целое, ширина поля; применяется только к полям DT_CHAR и DT_DECIMAL.
<b>DecimalPlaces</b>	Короткое целое, число десятичных разрядов в поле DT_DECIMAL.

 Что бы получить коллекцию объектов **MIField**, сошлитесь на свойства **Fields** объекта **MISearchInfo** или объекта **MISelection**.

Свойства объекта **MISelection**

Следующие свойства применяются к объекту **MISelection**.

Свойства объекта **MISelection**

Свойство	Функциональность
<b>Rows</b>	Это свойство возвращает коллекцию строк (объектов <b>MIRow</b> ). Эта коллекция представляет выборку объектов (строк) таблицы.
<b>Fields</b>	Это свойство возвращает коллекцию полей (объектов <b>MIField</b> ). Эта коллекция представляет параметры полей (имена полей и т.п.) таблицы, определенной в методе <b>GetTable</b> .
<b>TableName</b>	Строка, имя таблицы, которая была определена в методе <b>GetTable</b> .

Для получения доступа к объекту **MISelection**, используйте метод **GetTable** из объекта **MIMap-Gen**.

## Аргументы командной строки MapInfo Professional

Если Вы используете для связи с MapInfo динамический обмен данными (DDE), Вы должны запустить MapInfo вручную (например, вызовом функции **Shell( )** языка Visual Basic) перед установлением DDE-связи. При запуске MapInfo Professional Вы можете использовать в командной строке любой из нижеперечисленных аргументов. Если Вы хотите оставить пользователя в неведении о работе программы MapInfo Professional в фоновом режиме, задайте один из следующих аргументов.

Аргумент командной строки	Эффект
<code>-nosplash</code>	MapInfo Professional запускается без вывода на дисплей заставки, показывающей заставку MapInfo Professional, номер версии и т.п.
<code>-server</code>	MapInfo Professional запускается без вывода на дисплей заставки или главного окна. Используйте этот аргумент, если Вы хотите, чтобы MapInfo работал как фоновый сервер для других приложений, использующих DDE.
<code>-automation</code> или <code>-embedding</code>	MapInfo Professional запускается без вывода на дисплей заставки или главного окна. Кроме того, MapInfo регистрирует свой Поставщик Объектов OLE (OLE Class Factory) в подсистеме OLE, что позволяет MapInfo работать фоновым OLE-сервером для другого приложения.
<code>-regserver</code>	MapInfo регистрирует свои OLE-возможности в регистрационной базе данных, после чего завершает работу. Запустите MapInfo с этим аргументом один раз после установки программы. Заметьте, что MapInfo автоматически регистрирует себя при обычном запуске. Особенно учтите, что при таком запуске регистрируются все механизмы MapInfo – управление объектами OLE (OLE Automation), OLE Embedding и т.д.
<code>-unregserver</code>	MapInfo Professional убирает все ссылки на себя из регистрационной базы данных, после чего завершает работу. Используйте этот аргумент при удалении программы с диска. Применение этого аргумента ликвидирует регистрацию всего, что было зарегистрировано при запуске с аргументом <code>-regserver</code> .
<code>-helpdiag</code>	Этот аргумент устанавливает специальный флажок в MapInfo, в результате чего MapInfo выводит диагностическое окно диалога при каждой попытке вызова <i>Справочной системы</i> нажатием клавиши F1. Более подробное обсуждение использования <i>Справочной системы</i> смотрите в разделе <b>Вызов стандартного справочного файла MapInfo Professional на стр. 233</b> .



Вместо знака минус можно использовать косую черту ("/").

## Введение в Интегрированную картографию с поддержкой Visual C++ и MFC

Оставшаяся часть этой главы описывает процесс создания программы на языке Visual C++ с поддержкой MFC, которая использует Интегрированную картографию. Примеры написаны для 32-битной версии Visual C++ (версия 2.0 и выше), но они работают и для 16-битной версии Visual C++ (версия 1.52); разница между версиями упоминается там, где это необходимо.

### Создание нового проекта

1. Запустите Visual C++ 2.x (32-битную) или 1.5x (16-битную).
2. Выполните команду **File > New** для создания нового проекта.
3. Запустите ассистирующую процедуру MFC **AppWizard** и задайте нужные режимы и параметры. Для первого раза выберите режим однодокументного окна (SDI), а не многодокументного интерфейса (MDI). Помните, что не обязательно сразу подключать стандартную поддержку OLE. Если Вам нужно использовать уведомления (callback) из Вашего приложения в MapInfo Professional, то закажите поддержку OLE Automation на шаге 3 из 6 процедуры **AppWizard**.
4. Соберите программу и запустите ее, чтобы убедиться в том, что она работает.

### Добавление клиентской поддержки OLE Automation

Если Вы не заказали поддержку OLE в процедуре **AppWizard**, можно добавить клиентскую поддержку OLE Automation следующим образом.

1. Откройте файл STDAFX.H и добавьте строки:  

```
#include <afxole.h>
#include <afxdisp.h>
```
2. Откройте главный файл текста программы (т.е. *имя\_проекта*.CPP) и добавьте следующие строки в начало *Симя\_проекта*App::InitInstance:  

```
if (!AfxOleInit()) {
    AfxMessageBox(IDP_OLE_INIT_FAILED);
    return FALSE;
}
```
3. Добавьте строчку в файл строчных ресурсов (с названием *имя\_проекта*.RC). Для этого откройте ресурс "String Table", выполните команду **Resource > New String**. Присвойте значение ID: "IDP\_OLE\_INIT\_FAILED" и значение **Caption**: "Не удалось инициализировать OLE. Проверьте правильность версии OLE библиотек." Закройте диалог **Properties**. Закройте и сохраните файл ресурсов.

## Создание класса поддержки MapInfo Professional Support Class и его экземпляра

В диалоге **Project > ClassWizard** откройте раздел OLE Automation и нажмите на кнопку **Read Type Library**. Найдите в Вашем каталоге MapInfo Professional файл MAPINFOW.TLB. Нажмите **ОК**, чтобы подтвердить создание классов. Будет создан класс, с помощью которого Вы можете обращаться к MapInfo Professional через механизм управления объектами OLE (OLE Automation).

Откройте главный файл с текстом программы (*имя\_проекта.CPP*) и добавьте в него следующие строки.

- После всех директив `#includes`:  
`#include "MapInfow.h"`
- Сразу за объявлением `"Симя_проектаApp theApp"` добавьте объявление переменной:  
`DMapInfo mapinfo;`
- Ближе к концу `Симя_проектаApp::InitInstance`, но перед вызовом **OnFileNew( )** добавьте:  
`mapinfo.CreateDispatch("MapInfo.Application");`

Откройте файл MAPINFOW.H и добавьте в конец файла следующие строки:

```
extern DMapInfo mapinfo;  
#include "маршрут-к-каталогу-mapbasic\mapbasic.h"
```

## Тестирование

Добавьте еще одну строчку в конец функции `Симя_проектаApp::InitInstance`, сразу после вызова **CreateDispatch** (его добавление описано выше):

```
::MessageBox(0, mapinfo.GetFullName(), mapinfo.GetName(), MB_OK);
```

Соберите снова Вашу программу. При ее запуске Вы должны увидеть сообщение с заголовком "MapInfo Professional" и полным DOS-маршрутом к MapInfo Professional. Это означает, что MapInfo успешно запустилась через механизм OLE Automation. Позже тестирующую строку `"::MessageBox..."` можно закомментировать или удалить.

## Переопределение "быстрых" или "плавающих" меню

Встраивая Карту в Ваше приложение, Вы можете украсить ее всеми сервисными средствами MapInfo. Иногда этот сервис не нужен и мешает; например, стандартное быстрое меню для окна Карты включает в себя команду **Дублирование окна**. Ее нужно удалить из быстрого меню, чтобы она не вводила в заблуждение пользователей Вашего приложения.

Для этого ближе к концу текста `Симя_проектаApp::InitInstance` сразу после вызова **CreateDispatch** добавьте следующие строки:

```
// отключить вызов Справочной системы  
mapinfo.Do("Set Window Help Off");  
// переопределение "быстрого меню"  
mapinfo.Do("Create Menu \"MapperShortcut\" ID 17 as \"(-)\");
```

Здесь же можно добавить другие инструкции, например, открыть необходимые таблицы.



## Переподчинение диалогов MapInfo Professional

Очень важно научиться переподчинять диалоги MapInfo, появляющиеся из окна Вашего приложения, особенно, если они предназначаются для заполнения пользователем. Этот прием дает уверенность в том, что диалог будет появляться над окном приложения и что окно приложения будет неактивным все время, пока пользователь заполняет диалог MapInfo. В следующем примере показано, как переподчинить два диалога MapInfo (например, используя **RunMenuCommand** с заданным аргументом) и сообщения об ошибках, которые MapInfo показывает, обнаруживая непонятные события.

В тексте MainFrm.CPP, для функции "CMainFrame::OnCreate" надо добавить:

- После всех директив `#includes`:  
`#include "MapInfow.h"`
- В конце текста "CMainFrame::OnCreate":  
`char str[256];  
sprintf(str, "Set Application Window %lu", (long) (UINT)m_hWnd);  
mapinfo.Do(str);`

Чтобы убедиться в том, что это сработало, добавьте строку: текст функции *Симя\_проектаApp::InitInstance*, сразу после вызова **OnFileNew()**. Это приведет к тому, что MapInfo Professional покажет один из своих стандартных диалогов изнутри прикладной программы

```
mapinfo.Do("Note \\"Привет от MapInfo\\"");
```

После организации подобного переподчинения рекомендуется провести промежуточное тестирование.

## Добавление окна Карты

Теперь, когда MFCСприложение заработало и Вы убедились, что к MapInfo можно обращаться через OLE Automation, пора сделать то, ради чего все это затевалось: добавить Карту в приложение.

Откройте диалог **Project > ClassWizard**. Выберите класс представлений (*Симя\_проектаView*) и раздел "Message Maps". В самом левом окошке списка выберите объект "Симя\_проекта-View".

В списке "Messages" выберите "WM\_CREATE", нажмите на **Add Function**; выберите "WM\_DESTROY", нажмите на **Add Function**; выберите "WM\_SIZE", и нажмите на **Add Function**.

В заголовочный файл для представлений (*имя\_проектаVW.H*) добавьте строки:

```
unsigned long m_windowid;  
HWND m_windowhwnd;
```

В файл ресурсов (*имя\_проектаVW.CPP*), добавьте следующее:

- После всех директив `#includes`:  
`#include "MapInfow.h"`
- В конструкторе (*Симя\_проектаView::Симя\_проектаView*) инициализируйте переменные:  
`m_windowid = 0;  
m_windowhwnd = 0;`
- В метод **OnCreate** добавьте следующий отрывок после вызова **CView::OnCreate**:

```
//must have ClipChildren style for integratable maps to work
SetWindowLong(m_hWnd, GWL_STYLE,
    GetWindowLong(m_hWnd, GWL_STYLE)
    |WS_CLIPCHILDREN);
char str[256];
mapinfo.Do("Open Table \"States\" Interactive");
sprintf(str,
    "Set Next Document Parent %lu Style 1 Map From States",
    (long)(UINT)m_hWnd);
mapinfo.Do(str);
m_windowid = atol(mapinfo.Eval("WindowID(0)"));
sprintf(str, "WindowInfo(0, %u)", WIN_INFO_WND);
m_windowhwnd = (HWND)atol(mapinfo.Eval(str));
```

В метод **OnDestroy** добавьте следующий отрывок до вызова **CView::OnDestroy**:

```
if (m_windowhwnd) {
    ::DestroyWindow(m_windowhwnd);
    m_windowhwnd = NULL;
    m_windowid = 0L;
}
```

• В метод **OnSize** добавьте следующий отрывок после вызова **CView::OnSize**:

```
if (m_windowhwnd && cx > 0 && cy > 0) {
    ::MoveWindow(m_windowhwnd, 0, 0, cx, cy, TRUE);
}
```

## Добавление команд меню для Карты

Все пункты меню могут быть добавлены описанным ниже способом. В примере показано, как добавить пункт меню **Карта > Управление слоями**.

1. Откройте файл ресурсов (*имя\_проекта*.RC), откройте ресурс "Menu" и выберите IDR\_MAINFRAME.
2. Добавьте новое меню "Карта". Ниже "Карты" добавьте команду "Управление слоями" и сохраните RC-файл.
3. В диалоге **Project > ClassWizard** откройте раздел "Message Map" и выберите *Симя\_проекта*View из списка "Class Name". В списке "Object ID" выберите ID-номер для создаваемой команды меню, по умолчанию это ID\_MAP\_LAYERCONTROL. Как только Вы это сделаете, появятся сообщения COMMAND и UPDATE\_COMMAND\_UI в окне "Messages". Добавьте прототипы функций, нажимая для каждого сообщения кнопку **Add Function**, соглашаясь с предложенными стандартными именами.
4. В тексте класса *Симя\_проекта*View Вы увидите, что добавлены обе функции. Добавьте к текстам функций следующие строки.

```
void CprojectnameView::OnMapLayercontrol()
{
    mapinfo.RunMenuCommand(M_MAP_LAYER_CONTROL);
}
void CprojectnameView::OnUpdateMapLayercontrol(CCmdUI* pCmdUI)
{
    CmdUI->Enable(m_windowid);
}
```

## Добавление инструментальных кнопок

Все кнопки на панель инструментов могут быть добавлены описанным ниже способом. Этот пример показывает, как добавить кнопки MapInfo: **Стрелку**, **Сдвиг** и обе **Лупы**. Для удобства мы также добавим их в новое меню **Программы** (или **Tools**); это позволяет добавить их на панель инструментов несколько более простым способом, используя **ClassWizard**.

1. Сначала, следуя приведенным выше инструкциям (**Добавление команд меню для Карты на стр. 258**), создайте новое меню "Программы" с четырьмя новыми элементами ("Выбрать", "Сдвинуть", "Увеличить" и "Уменьшить"). Для каждой команды определите функции UPDATE\_COMMAND\_UI и COMMAND, используя коды из файла MAPBASIC.H (M\_TOOLS\_SELECTOR, M\_TOOLS\_RECENTER, M\_TOOLS\_EXPAND, и M\_TOOLS\_SHRINK); эта процедура также описана выше. После этого скомпилируйте и протестируйте программу.
2. Открыв RC-файл для проекта, выберите растровый ресурс IDR\_MAINFRAME и создайте растр на 64 пиксела шире (чтоб поместились 4 кнопки шириной 16–пикселей). На этом растре нужно разместить изображения четырех кнопок справа от кнопки вставки. Нарисуйте подходящие изображения для четырех новых инструментов, например, стрелку (курсор), руку (сдвиг), увеличительное стекло (для увеличивающей и уменьшающей лупы).
3. Откройте раздел ресурсов "String" и добавьте описания для каждой кнопки. При этом нужно следить за тем, чтобы ID-номера строк описаний совпадали с номерами ранее заданных команд; строки можно задавать также, как и в файле MAPINFOW.MNU: "текст описания, за которым следует разделитель "\n", а затем "текст всплывающей подсказки". например, номеру ID\_TOOLS\_SELECTOR соответствует "Выбрать объект на Карте\nСтрелка"; ID\_TOOLS\_GRABBER – "Сдвинуть Карту\nСдвиг"; ID\_TOOLS\_ZOOMIN – "Показать детальнее\nУвеличивающая лупа"; и, наконец, ID\_TOOLS\_ZOOMOUT – "Уменьшить детализацию\nУменьшающая лупа".
4. В тексте MAINFRM.CPP найдите массив "UINT BASED\_CODE buttons[ ]" типа "static2 и вставьте ID-константы в этот массив в том порядке, в котором они появляются в растровом ресурсе.
5. Чтобы не было проблем при работе с интерфейсом пользователя, необходимо следить за тем, какой из инструментов используется в данный момент. В текст файла заголовков *Симя\_проектаView* добавьте объявление целой переменной, которая будет хранить значение выбранной кнопки:

```
int m_eMouseMode;
```

6. Эту переменную нужно инициализировать в конструкторе классов, чтобы задать начальную нажатую кнопку на экране. Для этого нужно пользоваться заданными в MapInfo Professional константами.

```
m_eMouseMode = M_TOOLS_SELECTOR;
```

7. Если Вы сначала задали команды меню, то у Вас уже есть описания функций COMMAND и UPDATE\_COMMAND\_UI; если нет, то добавим их способом, описанном в предыдущем примере.
8. Задайте обновление интерфейса, вызывая **CCmdUI::SetRadio** в каждой процедуре **OnUpdate** и задайте переменные "m\_eMouseMode" соответственно процедурам "OnTools*Имя\_инструмента*". Добавленные Вами процедуры должны выглядеть примерно так:

```
void Симя_проектаView::OnToolsSelector()  
{
```

```
        m_eMouseMoveMode = M_TOOLS_SELECTOR;
        mapinfo.RunMenuCommand(M_TOOLS_SELECTOR);
    }
void Сима_проектаView::OnToolsGrabber()
{
    m_eMouseMoveMode = M_TOOLS_RECENTER;
    mapinfo.RunMenuCommand(M_TOOLS_RECENTER);
}
void Сима_проектаView::OnToolsZoomin()
{
    m_eMouseMoveMode = M_TOOLS_EXPAND;
    mapinfo.RunMenuCommand(M_TOOLS_EXPAND);
}
void Сима_проектаView::OnToolsZoomout()
{
    m_eMouseMoveMode = M_TOOLS_SHRINK;
    mapinfo.RunMenuCommand(M_TOOLS_SHRINK);
}
void Сима_проектаView::OnUpdateToolsSelector(CCmdUI* pCmdUI)
{
    pCmdUI->SetRadio(m_eMouseMoveMode == M_TOOLS_SELECTOR);
    pCmdUI->Enable(m_windowid);
}
void Сима_проектаView::OnUpdateToolsGrabber(CCmdUI* pCmdUI)
{
    pCmdUI->SetRadio(m_eMouseMoveMode == M_TOOLS_RECENTER);
    pCmdUI->Enable(m_windowid);
}
void Сима_проектаView::OnUpdateToolsZoomin(CCmdUI* pCmdUI)
{
    pCmdUI->SetRadio(m_eMouseMoveMode == M_TOOLS_EXPAND);
    pCmdUI->Enable(m_windowid);
}
void Сима_проектаView::OnUpdateToolsZoomout(CCmdUI* pCmdUI)
{
    pCmdUI->SetRadio(m_eMouseMoveMode == M_TOOLS_SHRINK);
    pCmdUI->Enable(m_windowid);
}
```

## Обработка ошибок MapInfo Professional

MapInfo Professional пересылает информацию об ошибках в приложение, использующее Интегрированную картографию, посредством MFC-класса **COleDispatchException**. MapInfo Professional возвращает код ошибки в переменной "m\_wCode" класса **COleDispatchException** и описание ошибки в переменной "m\_strDescription" класса **COleDispatchException**. Обычно ошибки OLE общего характера передаются через класс **COleException**. Вы должны обработать эти ошибки внутри Вашего приложения, иначе ими займется обработчик MFC-ошибок более высокого уровня, а мы получим сообщение "Command failed". Вы можете добавить обработчик для каждой ошибки в каждый метод **DMapInfo**. В следующем примере показано, как это делается для метода **DMapInfo::Do**.

Оригинальный текст метода **DMapInfo::Do**, порожденный **ClassWizard**, выглядит так:

```
void DMapInfo::Do(LPCTSTR command)
{
    static BYTE BASED_CODE parms[] = VTS_BSTR;
    InvokeHelper(0x6001000b, DISPATCH_METHOD, VT_EMPTY,
        NULL, parms, command);
}
```

Усовершенствованный метод **DMapInfo::Do**, включающий обработку исключительных ситуаций, выглядит так:

```
void DMapInfo::Do(LPCTSTR command)
{
    static BYTE BASED_CODE parms[] = VTS_BSTR;
    try {
        InvokeHelper(0x6001000b, DISPATCH_METHOD, VT_EMPTY,
            NULL, parms, command);
    }
    catch(COleDispatchException *e) {
        // Обработка исключительной ситуации в Вашей.
        // программе.. Ошибка помещается в e->m_wCode.
        AfxMessageBox(e->m_strDescription);
        e->Delete();
    }
    catch(COleException *e) {
        AfxMessageBox("Fatal OLE Exception!");
        e->Delete();
    }
}
```

## Добавление поддержки сервера OLE Automation

В файл *Симя\_проектаDoc.cpp* добавьте раздел Dispatch map после Message map.

```
BEGIN_DISPATCH_MAP(Симя_проектаDoc, CDocument)
//{{AFX_DISPATCH_MAP(Симя_проектаDoc)
    // ClassWizard будет добавлять и удалять макро-команды для
    картографирования здесь
    // Не редактируйте ничего из того, что вы видите в этом коде!
//}}AFX_DISPATCH_MAP
END_DISPATCH_MAP()
```

```
В файле Симя_проектаДос.cpp добавьте в раздел Симя_проектаДос  
constructor:  
EnableAutomation();  
AfxOleLockApp();  
В файле Симя_проектаДос.cpp добавьте в раздел Симя_проектаДос  
AfxOleUnlockApp();  
В файле Симя_проектаДос.h добавьте раздел Dispatch после message map  
// Сгенерированный функции диспетчирования OLE  
//{{AFX_DISPATCH(Симя_проектаДос)  
// ClassWizard будет добавлять и удалять методы класса здесь.  
// Не редактируйте ничего из того, что вы видите в этом коде!  
//}}AFX_DISPATCH  
DECLARE_DISPATCH_MAP()
```



В приведенном выше фрагменте кода показано, как добавить поддержку механизма управления OLE-объектами в порожденный CDocument-класс. Используя MFC, можно также просто добавить поддержку механизма управления OLE-объектами любому классу, порожденному **CCmdTarget**. Так, для MDI-приложения вы можете добавить интерфейс механизма управления OLE-объектами либо порожденному классу **CWinApp**, либо порожденному классу **CMDIFrameWnd**; оба они были порождены классом **CCmdTarget**. Причиной этого является то, что указатель **IDispatch** для уведомлений MapInfo Professional можно устанавливать только один раз. В MDI-приложении окна документов уничтожаются при закрытии. Если указатель **IDispatch** будет установлен для документа, то он исчезнет вместе с окном.

## Добавление уведомления (callback) WindowContentsChanged

Если Вы создаете приложение с одним окном (SDI) и добавили в класс *Симя\_проектаДос* список сообщений для диспетчера, то тогда можно установить указатель на уведомление в конструкторе *Симя\_проектаДос* или в любом другом месте, где он будет вызван только один раз.

```
mapinfo.SetCallback(this->GetIDispatch(FALSE));
```

В диалоге **Project > Class Wizard** выберите раздел "OLE Automation" и из списка **Class Name** выберите класс, для которого разрешено использование OLE Automation (в нашем примере это *Симя\_проектаДос*). Выберите **Add Method** и задайте имя метода "WindowContentsChanged", возвращаемый тип "SCODE" и список аргументов "long lWindowID". После нажатия на **ОК** и выхода из диалога **ClassWizard** автоматически обновляет файлы *Симя\_проектаДос.CPP* и файлы заголовков. В CPP-файле заполните тело функции **WindowContentsChanged** и обеспечьте обработку сообщений. В этой функции удобнее всего обрабатывать Легенду.

## Где получить дополнительную информацию

Чтобы узнать больше об Интегрированной картографии, ознакомьтесь с примерами программ, поставляемыми в комплекте со средой разработки MapBasic. Следующие примеры программ включены в состав MapBasic:

- Samples\VB\FindZip: программа на языке Visual Basic, используемая образец разработки собственных приложений в этой главе.
- Samples\VB\VMapTool: программы, демонстрирующие более сложные примеры интеграции с Visual Basic 4.0 Professional Edition или более поздней версией.
- Samples\MFC\FindZip: Простое приложение, использующее MFC.
- Samples\PwrBldr\Capitals: Простое приложение на языке PowerBuilder. Внимание: это 16-битное приложение, и Вам нужно иметь runtime-версию PowerBuilder, чтобы увидеть, как оно работает.
- Samples\Delphi\TabEdMap: Простое приложение на языке Delphi.

В каталоге SAMPLES Вы можете найти другие приложения, в том числе и не указанные в данном *Руководстве*.

# Работа в среде .Net

Программы MapBasic могут вызывать функции и подпрограммы, созданные в среде разработки .Net корпорации Microsoft. Вы можете использовать языки C# (C-sharp) и VB.Net (Visual Basic для .Net) и среду разработки Microsoft Visual Studio. Эти .Net-компоненты Вы можете вызывать из программ MapBasic.

Некоторые сложные для MapBasic задачи – или даже не выполнимые для него – можно легко запрограммировать в среде .Net. Например, оператор Dialog в MapBasic не может создавать диалоги с закладками (Tab) или древовидными структурами (TreeView), но в среде .Net Вы можете это сделать.

Таким образом, Вы можете создавать прикладные программы в MapBasic, а вызываемые из них подпрограммы и функции – в среде .Net.

## В этой главе:

- ♦ Требования к программированию в среде .Net .....265
- ♦ Первые шаги .....265
- ♦ Работа со структурами в .Net .....270
- ♦ Обработка ошибок .....274
- ♦ Работа с GAC Suite .....275
- ♦ Управление MapInfo Professional изнутри метода .Net ....276
- ♦ Интегрированная картография в .Net.....278



## Требования к программированию в среде .Net

В этой главе подразумевается, что Вы уже отчасти знакомы с тем, как пишутся программы в среде .Net. В частности, Вы должны понимать, как создается класс в .Net, как назначается ему конструктор и какие шаги Вы должны предпринять далее, чтобы создать Вашу программу в среде Microsoft Visual Studio для .Net.

Чтобы иметь возможность вызывать .Net-компоненты из программы MapBasic, Вы должны использовать MapInfo Professional и MapBasic версии 10 или выше.

Файлы поддержки .Net версии 3.5 должны быть установлены на Вашем компьютере до того, как Вы запустите MapBasic-программу (MBX); однако, ниже мы будем полагать, что поддержка .Net уже установлена на компьютере с MapInfo Professional 10, поскольку при установке MapInfo Professional файлы поддержки добавляются автоматически.

Фрагменты программ, приведенные в примерах, написаны на VB.Net и C#. Эти примеры созданы в среде разработки Visual Studio 2005. Для разработки .Net-компонентов, к которым может обращаться MapBasic, Вы можете использовать версии Visual Studio 2003 и выше.

### О терминах

В этой главе слово "метод" (method) применяется для любой подпрограммы или функции, созданной средствами .Net. Например, sub-программа, написанная на языке VB – это один метод, а функция (Function) на языке VB – это уже другой метод. Оба таких метода Вы можете вызывать из программы MapBasic.

Некоторые методы могут быть вызваны без предварительного создания экземпляра класса. В языке C# такие методы принято называть "статическими" (static), в языке VB.Net их называют "общедоступными" (shared). Ниже мы будем использовать термин "статический метод" для обозначения метода, который можно вызывать без предварительного создания экземпляра класса.

## Первые шаги

В этом разделе приводится простой пример создания класса в среде .Net и использование этого класса в программе MapBasic.

Вызов .Net-метода из MapBasic требует выполнения шести шагов:

1. Создать класс в .Net, содержащий один или несколько статических методов.
2. Встроить .Net-класс в файл сборки (assembly).
3. Сделать файл сборки доступным для программы MapBasic.
4. Включить оператор Declare Method в MB-программу.
5. Вызвать метод, объявленный оператором Declare Method.
6. Компилировать и запустить MapBasic-программу.

## Создание класса в .Net

Все начинается с создания класса в .Net. Класс может быть очень простым. Необходимым требованием к классу является задание в нем хотя бы одного статического метода. MapBasic может вызывать только статические методы .Net.

В следующем примере создается простой класс с одним статическим методом, который показывает диалог с приветствием. Этот метод требует задания строкового аргумента и возвращает целое число, количество символов в вводимой строке.

Задание класса на языке C# может выглядеть примерно так:

```
using System;
using System.Windows.Forms;

namespace MapBasicMethods
{
    public class Demo
    {
        public static int SayHello(String strName)
        {
            MessageBox.Show("Привет, " + strName + "!");
            return strName.Length;
        }
    }
}
```

Большую часть этого кода порождает Visual Studio, так что Вам придется вводить от руки совсем немного. При создании нового проекта в Visual Studio Вы можете выбрать шаблон проекта, позволяющий Вам задавать новую библиотеку классов (Class Library), и тогда Visual Studio будет создавать большую часть кода автоматически.

На языке VB этот же класс может быть описан так:

```
Namespace MapBasicMethods

    Public Class Demo

        Public Shared Function SayHello(ByVal s As String) As Integer
            System.Windows.Forms.MessageBox.Show("Привет, " + s + "!")
            Return s.Length
        End Function

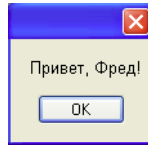
    End Class

End Namespace
```



В этом примере фрагмент программы на VB явно объявляет пространство имен. На практике в VB-коде оператор Namespace не указывается, поскольку он обычно задается в свойстве Root Namespace VB-проекта.

Метод SayHello использует .Net-класс MessageBox, чтобы показывать диалоги такого вида:



Так как в этом примере используется класс MessageBox, то в проекте нужно обеспечить доступ к стандартному файлу сборки, System.Windows.Forms. Это можно сделать в среде программирования Visual Studio.

## Создание и копирование файла сборки

Создав класс, Вы с помощью средств Visual Studio встраиваете его в EXE или в DLL-файл.

Далее Вы должны сделать файл сборки доступным из Вашей MapBasic-программы. Для этого проще всего перенести файл сборки (DLL) в один каталог с MBX-файлом. Другой способ, не требующий переноса файлов, состоит в том, что Вы регистрируете файл сборки в глобальном кеше сборок .Net Global Assembly Cache (GAC). Это можно сделать с помощью утилиты GACUTIL поставляемой Microsoft; подробности см. в документации по программированию в среде .Net.

---

**i** MapInfo Professional позволяет запускать MBX-программы из сети, используя UNC-маршрут. Однако, стандартные режимы безопасности среды .Net запрещают загрузку файлов сборки .Net с сети. Если Вы создали свой файл сборки .Net, и Вам нужно обращаться к нему из Вашей MapBasic-программы, установите этот файл сборки на клиентских машинах заранее. Другим вариантом решения этой проблемы является изменение режимов безопасности среды .Net Framework 2.0 с помощью утилиты Configuration, которую можно найти в окне **Control Panel > Administrative Tools** ("Панель управления" > "Администрирование"). Процедуры настройки режимов безопасности в этой документации не приводятся.

---

Теперь, когда у нас есть статический метод, класс и файл сборки .Net, мы можем составить обращение к этому классу из MapBasic.

## Объявление и вызов метода из MapBasic

Перед тем, как вызвать .Net-метод из MapBasic, Вы должны объявить его в Вашей MB-программе оператором **Declare Method**. Оператор **Declare Method** очень похож на оператор **Declare Function**, но содержит дополнительные предложения, позволяющие задать класс и файл сборки .Net.

Детали синтаксиса оператора **Declare Method** Вы можете найти в Справочнике MapBasic. Следующий код показывает, как может выглядеть простейший метод "SayHello" для нашего примера.

```
' Синтаксис MapBasic для объявления метода .Net как функции
Declare Method SayHello
    Class "MapBasicMethods.Demo" Lib "MBMethods.dll"
```

```
(ByVal strName As String) As Integer
```

В этом примере мы будем вызывать метод по его имени – "SayHello". (Можно также обращаться к методу по его псевдониму, используя предложение **Alias**; но в этом примере для простоты псевдоним не указывается. Псевдонимы нужны только если существует нужда различать несколько функций с одинаковыми именами.)

Предложение **Class** задает имя класса, например, "MapBasicMethods.Demo", в котором "Map-BasicMethods" является заданным нами пространством имен, а "Demo" – именем класса.

Предложение **Lib** задает имя файла сборки, например, "MBMETHODS.DLL"; имя файла сборки Вы можете задавать как атрибут проекта в Visual Studio.

Список аргументов совпадает со списком аргументов метода .Net. Это строка, передаваемая значением (by-value).

Тип возвращаемого значения совпадает с типом, возвращаемым методом .Net. В MapBasic это 4-байтное целое (4-byte integer), что эквивалентно в C# типу "int" (также называемому System.Int32).

Корректно задав оператор **Declare Method**, Вы можете обращаться к методу .Net, как к любой другой функции MapBasic. Например:

```
' Синтаксис MapBasic вызова метода .Net
Dim i As Integer
i = SayHello("Фред")
```

Если Ваш метод .Net не должен ничего возвращать (то есть Вы создаете подпрограмму, а не функцию; в языке C# это метод "void"), или Вы можете проигнорировать возвращаемое значение, опускайте предложение "As" в конце оператора **Declare Method**. Например:

```
' Синтаксис MapBasic для объявления .Net-метода как sub-программы
Declare Method SayHello
    Class "MapBasicMethods.Demo" Lib "MBMethods.dll"
        (ByVal strName As String)
```

```
Declare Sub Main
```

```
Sub Main
    Call SayHello("Фред")
End Sub
```

Прокомпилируйте и запустите программу MapBasic. Как только MBX-программа обратится к методу "SayHello", Ваша .Net-сборка будет загружена и будет вызван Ваш статический метод.

Обратите внимание на то, что если в записи оператора **Declare Method** была допущена ошибка (например, в имени класса), Ваша программа на MapBasic будет компилироваться, но при выполнении MBX-файла произойдет ошибка. Ошибки в именах файла сборки, класса и метода, а также список аргументов метода никак не обнаруживаются до тех пор, пока программа не будет запущена.

## Вызов метода по его псевдониму

Если имя метода, которое Вы зададите а операторе **Declare Method**, совпадает с именем уже существующей подпрограммы или функции в Вашей MapBasic-программе, то при компиляции это породит ошибку, потому что в программа MapBasic не может содержать две подпрограммы с одинаковым именем. Чтобы обойти это ограничение, в операторе **Declare Method** предусмотрено предложение **Alias**. В предложении **Alias** задается настоящее имя метода .Net; далее, Вы задаете в аргументе "fname" – он следует сразу за ключевым словом **Method** – уникальное имя (то есть, такое, которое нигде не используется). В следующем примере, мы вызываем метод .Net с именем "ShowDialog", но внутри MB-кода, он известен под именем "ShowPointDialog":

```
Declare Method ShowPointDialog
  Class "MyProduct.MyWrapper"
    Lib "MyAssembly.DLL"  Alias ShowDialog  () As Integer

Dim i As Integer
i = ShowPointDialog()
```

## Передача аргументов в .Net


Некоторые типы переменных, присущие только MapBasic, такие как **Pen** или **Brush**, не могут быть переданы в метод .Net. В следующей таблице приведены соответствия между типами, данных в MapBasic, и типами, используемыми в среде .Net.

Тип в MapBasic	Тип в .Net	Тип в VB.NET	Тип в C#
Короткое целое число. Величина типа SmallInt	System.Int16	Short	short
Целое число типа Integer	System.Int32	Integer	Int
Вещественное число типа Float	System.Double	Double	double
String (как фиксированной, так и переменной длины)	System.String	String	String
Логическое	System.Boolean	Boolean	bool
Типы и структуры, заданные пользователем	Зависит от конкретного случая; см. раздел ниже		
Другие типы MapBasic	нет	нет	нет


Аргументами могут быть массивы. Например, если Ваш метод .Net принимает в качестве аргумента массив целых чисел, то оператор **Declare Method** может выглядеть так:

```
Declare Method ProcessIntegerArray
  Class "MyProduct.MyWrapper" Lib "MyAssemblyName"
    (idNumbers() As Integer)
```

Аргументы могут передаваться ссылками (by-reference) или значениями (by-value). Описание таких аргументов в разных языках разное. В следующей таблице показывается, как можно передавать строчный аргумент (String) ссылкой в разных случаях.

 Вы не можете изменить размер массива MapBasic изнутри Вашего .Net-метода.

Язык	Синтаксис объявления ссылкой	Синтаксис объявления значением
MapBasic	str As String	ByVal str As String
VB.Net	ByRef str As String	ByVal str As String
C#	ref String str	String str

 MapBasic может передавать аргументы-массивы и аргументы-структуры только ссылкой (ByRef).

Замечания о скорости выполнения

Скорость вызова метода .Net зависит от того, сколько данных передается ему в списке аргументов. Чем больше данных Вы передаете, тем медленнее происходит вызов.

Если Вас не удовлетворяет скорость вызова метода .Net, попробуйте сократить суммарный объем данных, передаваемых методу в качестве аргументов.

Работа со структурами в .Net

Передача созданных пользователем типов (структур) в .Net

Программы MapBasic могут передавать структуры данных (переменные, созданные пользователем с помощью оператора **Type**) в среду .Net, но с некоторыми ограничениями. Для этого также Вы должны проделать некоторую подготовительную работу в .Net – создать соответствующее определение класса в .Net для представления этой структуры.

Чтобы передать структуру из MapBasic в метод .Net, поделайте в Вашем MB-коде следующее:

1. С помощью оператора MapBasic **Type** создайте MapBasic-структуру. (Если Вы хотите передать ее методу .Net, то, наверное, Вы уже это сделали.)
2. Выполните оператор **Declare Method**, чтобы задать сигнатуру .Net-метода, и включите тип MapBasic в список аргументов метода.
3. Объявите переменную-структуру того типа который нужно передать, и заполните ее поля значениями.
4. Вызовите метод и передайте ему в качестве аргумента эту структурную переменную.

В следующем фрагменте кода MapBasic определяется структурная переменная из 3 полей, а затем она пересылается в метод .Net.

**i** Оператор **Type** должен предшествовать оператору **Declare Method**, потому что имя типа (в этом примере, ParcelInfo) используется оператором **Declare Method**.

```
Type ParcelInfo
    idnum      As Integer
    descript As String
    area       As Float
End Type

Declare Method ShowParcelDialog
    Class "MapBasicMethods.Demo" Lib "MBMethods.dll"
    (p As ParcelInfo)

Declare Sub Main
Sub Main
    Dim p As ParcelInfo
    p.idnum = 23
    p.descript = "Участок-образец"
    p.area = 123.45

    Call ShowParcelDialog( p )
End Sub
```

В этом примере мы передали данные в структуре типа ParcelInfo в метод .Net. На следующем шаге мы решим, как описать метод .Net, чтобы он мог получить данные, посланные из MapBasic?

Чтобы метод .Net мог корректно принимать структурированные данные из MapBasic, сделайте следующее:

1. Создайте класс в .Net.
2. Назначьте этому классу конструктор типа "public" и дайте этому конструктору список аргументов, совпадающий со структурой MapBasic. Например, если Ваша MapBasic-структура содержит целое, строку и число с плавающей точкой, то и список аргументов конструктора должен содержать целое, строку и число с плавающей точкой. (Ваш класс может иметь и другие конструкторы, но MapInfo Professional/MapBasic их проигнорирует.)
3. В одном из Ваши .Net-классов создайте метод со атрибутами "public" и "static" (к этому методу и будет обращаться MapBasic). Добавьте аргументы к этому методу и определите тип аргумента как класс, который Вы создали на шаге 1.

Следующий фрагмент кода на C# показывает, как создать класс Parcel, соответствующий приведенной выше структуре ParcelInfo:

```
public class Parcel
{
    private int m_ID;
    private string m_Description;
    private double m_Area;
```

```

public Parcel(int idnum, string description, double area)
{
    m_ID = idnum;
    m_Description = description;
    m_Area = area;
}

// Вы можете также задать Свойства (Property) для
// ID, Description и Area.
// MapInfo/MapBasic не требуют задания Свойств.
}

```

Определив класс "Parcel", мы можем создать метод со атрибутами "public" и "static", который примет аргумент типа "Parcel".

```

public static int ShowParcelDialog(Parcel parc)
{
    // Здесь Вы поместите код, показывающий
    // компоненты структуры в диалоге.
    MessageBox.Show("Здравствуй, мир");
    return 0;
}

```

Теперь Ваша MapBasic-программа может передать структуру ParcelInfo в метод "ShowParcelDialog". Когда программа MapBasic посылает структуру методу, метод "ShowParcelDialog" получает объект подходящего типа; это происходит потому что MapInfo Professional преобразует структуру в приемлемый для метода .Net вид. (Поэтому Вы и должны присвоить Вашему .Net-классу public-конструктор – он требуется MapInfo Professional, чтобы преобразовывать структуры данных MapBasic в соответствующий объект .Net.)

Когда Вы передаете структуру MapBasic в Ваш .Net-метод, успешность вызова зависит от того, имеет ли класс-аргумент .Net конструктор типа "public", список аргументов которого совпадает со списком полей в структуре MapBasic. Если такого конструктора нет, Ваша MapBasic-программа породит ошибку во время исполнения при попытке вызова метода. Обратите внимание на то, что это ошибка обнаруживается только при выполнении; MapBasic не может ее обнаружить во время компиляции.

В некоторых случаях Вы начинаете программировать не с определения структуры MapBasic оператором **Type**, а наоборот, Вы отталкиваетесь от имеющейся сигнатуры .Net-метода. Представим, что Вы создаете вызов для уже существующего .Net-метода и этот метод принимает один аргумент: объект "System.Drawing.Point".

```

public static void ShowPointDialog(System.Drawing.Point p)
{
    MessageBox.Show("p.x is: " + p.X + ", p.y is: " + p.Y);
}

```

Такой аргумент метода не соответствует ни одному из стандартных типов переменных MapBasic, таких как Integer или String. Поэтому, если Вы хотите оформить вызов этого метода из MapBasic, то Вы должны создать в MapBasic структуру, приближающуюся к типу аргумента .Net (в нашем случае, "System.Drawing.Point"). В следующем примере приводится подходящий к этому случаю синтаксис MapBasic:



```
Type Location
  ix as Integer
  iy as Integer
End Type

Declare Method ShowPointDialog
  Class "MyProduct.MyWrapper"
  Lib "MyAssembly.DLL" (pnt As Location)

  . . .

Dim loc As Location
loc.ix = 23
loc.iy = 42

Call ShowPointDialog(loc)
```

В этом примере MapInfo Professional попытается проконвертировать структуру MapBasic в .Net-объект "System.Drawing.Point", вызывая public-конструктор класса "Point". Процесс преобразования аналогичен представленному в предыдущем примере, с одним важным различием: в последнем случае Вам не нужно создавать .Net-класс "Point", поскольку он уже существует – это класс, созданный Microsoft.

Так как MapBasic-структура "Location" содержит два целых поля, MapInfo Professional будет искать public-конструктор для класса "Point", принимающий два целых аргумента. Поля из структуры MapBasic передаются в .Net-конструктор, и при этом создается .Net-объект "Point". Объект "Point" затем передается в метод, завершая вызов.

Некоторые .Net-классы не имеют public-конструкторов. Например, таковых не имеет структура "System.Drawing.Color"; поэтому невозможно создать структуру MapBasic, имитирующую объект "System.Drawing.Color". Если Вы хотите передать информацию о цвете в метод .Net, то передавайте значения красного, зеленого и голубого (RGB) в отдельных аргументах. Далее, внутри .Net-метода эти значения можно скомбинировать в объект "Color".

```
Public Shared Sub ShowColorDialog(ByRef r As Integer, ByRef g As Integer,
ByRef b As Integer)

  Dim c As Color
  Dim dlg As ColorDialog
  dlg = New ColorDialog
  dlg.Color = Color.FromArgb(r, g, b)
  If (dlg.ShowDialog = DialogResult.OK) Then
    c = dlg.Color
    r = c.R
    g = c.G
    b = c.B
  End If

End Sub
```

В этом примере создается экземпляр объекта "ColorDialog", а затем производится вызов не-static (нестатического) метода для этих объектов. Выше указывалось, что в тексте программ MapBasic можно вызывать static-методы; однако, для .Net-кода, из которого состоит Ваш static-метод такое ограничение уже не действует. Внутри static-метода .Net Вы можете создавать экземпляры объектов и с их помощью вызывать не-static-методы.

## Ограничения на передачу структур

Если Вы передаете структуры из MapBasic в .Net, Вам может понадобиться дать Вашим .Net-методам уникальные имена. Помните, что в среде .Net допускается, чтобы класс имел несколько методов с одинаковым именем (они различаются по спискам переменных); в среде MapBasic, однако, это не пройдет. Если Вы передаете структуры из MapBasic в .Net, и если Ваш .Net содержит несколько методов с одинаковым именем и одинаковым количеством аргументов, то MapBasic может не определить, какой конкретный метод нужно вызвать. В этой ситуации программа MapBasic порождает ошибку исполнения при вызове .Net-метода. Проще всего избежать этой проблемы можно, задавая .Net-методам уникальные имена.

Если Вы передаете структуру в метод .Net, и этот .Net-метод изменяет переданный ему объект, то соответствующая MapBasic-структура не меняется. Чтобы заставить .Net-метод изменить значения передаваемых от MapBasic аргументов, используйте скалярные переменные (такие как "ix As Integer", а не структуры), и используйте способ ByRef.

## Обработка ошибок

Неперехваченная ошибка в методе .Net породит ошибку MapBasic с кодом 1666 (ошибка во время выполнения). Любая такая ошибка в программе MapBasic остановит выполнение MBX-программы; поэтому такие ошибки желательно перехватывать и обрабатывать. Вы можете создавать обработчики ошибок как в коде .Net, так и в коде MB.

Механизм обработки ошибок в .Net (блоки try-catch-finally) более удобен, чем соответствующий набор средств MapBasic. Поэтому, в общем случае лучше распознавать исключительные ситуации в .Net-фрагментах Ваших программ, чем полагаться на MapBasic. Однако, если Вы почему-либо не хотите использовать инструментарий .Net, то используйте в MapBasic-программе оператор **OnError**.

Следующий фрагмент кода показывает, как в MapBasic обрабатывается ошибка:

```
Sub MakeExternalCall

    OnError Goto caughtit
    Call DoSomething() ' Вызов метода .Net
    g_status = 0 ' задаем код результата; 0 = success
    Exit Sub
caughtit:

    ' Следующий код ошибки указывает на ошибку при вызове метода:
    if Err() = 1666 Then
        ' Код, назначаемый ниже, относится только в методу .NET
        ' и означает, что .NET-метод обнаружил необработанную ошибку.
        ' Используйте оператор Error$() для показа текста ошибки.
        g_status = -1
```

```

else
    ' Другие значения кода Err указывают на то, что метод не был
    ' вызван, например, из-за ошибки в синтаксисе Declare Method.
    Note "Ошибка в операторе Declare Method: Error: " + Error$()
    g_status = -2
end if

End Sub

```

## Работа с GAC Suite

### Загрузка файла сборки из кеша Global Assembly Cache (GAC)

В операторе **Declare Method** есть предложение **Lib**, в котором Вы можете задать файл сборки .Net. Если файл сборки находится в том же каталоге, что и файл MBX, то в предложении **Lib** можно просто задавать его имя (например, "MBTOOLS.DLL" или "MBTOOLS").

Если файл сборки был зарегистрирован в кеше GAC, то копировать DLL-файл в один каталог с файлом MBX не нужно. Однако, для обращения к файлу, зарегистрированному в GAC, в предложении **Lib** нужно указывать полное описание файла сборки, поскольку, в GAC может содержаться более одной версии такого файла.

В следующем примере показывается обращение к методу "System.IO.File.Delete", содержащемуся в файле сборки MSCORLIB от Microsoft:

```

Declare Method Delete
    Class "System.IO.File"
        Lib "mscorlib, Version=2.0.0.0, Culture=neutral,
        PublicKeyToken=b77a5c561934e089"
        (ByVal path as string)
    End Class
End Method

```

О том, как зарегистрировать файл сборки в GAC, а также полные описания файлов сборки, см. в документации по .Net-программированию от Microsoft.

Следующий пример показывает, как объявляются методы из файлов сборки, зарегистрированных в глобальном кеше GAC. Обратите внимание на то, что в предложении задается полное описание файла сборки. Чтобы составить полное описание файла сборки, можно воспользоваться разными вспомогательными средствами, например, утилитой GACUTIL, поставляемой Microsoft в составе Visual Studio.

```

'Объявление метода из файла сборки System.Windows.Forms.dll:
Declare Method Show
    Class "System.Windows.Forms.MessageBox"
        Lib "System.Windows.Forms, Version=2.0.0.0, Culture=neutral,
        PublicKeyToken=b77a5c561934e089"
        (ByVal str As String, ByVal caption As String)
    End Class
End Method

' Объявление метода из файла сборки mscorlib.dll:
Declare Method Move
    Class "System.IO.File"
        Lib "mscorlib, Version=2.0.0.0, Culture=neutral,
        PublicKeyToken=b77a5c561934e089"
    End Class
End Method

```

```
(ByVal sourceFileName As String, ByVal destFileName As String)

' Показ сообщения в .Net-диалоге с заголовком:
Call Show("Обновление таблицы завершено.", "Операция завершена")

' Вызов .Net-метода Move для перемещения файла
Call Move("C:\work\pending\entries.txt", "C:\work\finished\entries.txt")
```

## Управление MapInfo Professional изнутри метода .Net

При установке MapInfo Professional на Ваш диск добавляется файл сборки .Net MIADM.DLL, который поддерживает взаимодействие MapBasic и .Net. Некоторые из способов такого взаимодействия могут быть Вам полезны, поскольку они позволяют выполнять операторы MapBasic изнутри методов .Net.

Например, допустим, Вы создали в среде .Net диалог "Свойства Карты" с возможностью изменения режимов и кнопками **OK**, **Cancel** (Отмена) и **Apply** (Применить). Пусть диалог должен работать следующим образом:

1. MapBasic-программа вызывает метод .Net, чтобы показать этот диалог.
2. Пользователь устанавливает нужные ему режимы в диалоге, после чего нажимает на кнопку **Apply**.
3. Окно MapInfo Professional немедленно обновляется; при этом, поскольку была нажата кнопка **Apply** (а не **OK**), диалог остается на экране.
4. Затем, когда пользователь закроет диалог, метод .Net возвращает управление.

Для обновления окна используется соответствующий оператор MapBasic, например, **Set Map**. В этом примере оператор **Set Map** вызывается из кода программы .Net для демонстрации того, как изнутри метода .Net можно выполнять операторы MapBasic.

СОМ-интерфейс в MapInfo Professional поддерживает метод **Do**, позволяющий выполнять операторы MapBasic, и метод **Eval**, позволяющий собрать информацию о текущем состоянии MapInfo Professional. Если Вы когда-либо писали программу, использующую интегрированную картографию, то методы **Do** and **Eval** должны быть Вам знакомы; подробности см. в главе [Интегрированная картография на стр. 215](#).

.Net-класс "MapInfo.MiPro.Interop.InteropServices" предоставляет программистам простой доступ к методам MapInfo Professional **Do** и **Eval**. Класс "InteropServices" поддерживает свойство "MapInfoApplication", содержащее ссылку на объект "MapInfoApplication". В свою очередь, класс "MapInfoApplication" содержит методы **Do** и **Eval**.

Пример использования методов **Do** и **Eval** можно найти в тексте программы "Named Views" ("Показ Карт"), поставляемой с MapBasic (см. каталог Samples\DotNet\NamedViews). В тексте программы "Named Views" вызывается метод **Eval**, чтобы определить ID-номер активного окна Карты:

```
private static int GetFrontWindow()
{
    string evalResult =
    InteropServices.MapInfoApplication.Eval("FrontWindow()");
```

```
        return Int32.Parse(evalResult);  
    }
```

Аналогичным образом вызывается метод **Do** для выполнения оператора **Set Map** изнутри кода .Net:

```
InteropServices.MapInfoApplication.Do(string.Format(  
    "Set Map Window {0} Center ( {1}, {2} ) Zoom {3}",  
    windowId, centerX, centerY, mapperZoom));
```

Чтобы пользоваться классом "MapInfo.MiPro.Interop.InteropServices", включите в файл .Net-проекта указание на файл сборки. Этот файл находится в каталоге установки MapInfo Professional.



Если при создании программы "Named Views" возникают ошибки, то, возможно, их можно ликвидировать, обновив ссылку на MIADM.DLL в среде Visual Studio.

---

Класс "MapInfoApplication" – это "объемлющий" класс (wrapper), который дает возможность простого доступа в среде .Net к COM-интерфейсу MapInfo Professional. Этот класс добавлен для удобства, чтобы предоставить альтернативный доступ к методам, свойствам и событиям через стандартный .Net-класс вместо того, чтобы использовать напрямую COM-интерфейс. Кроме уже представленных методов **Do** и **Eval**, класс "MapInfoApplication" содержит компоненты, перечисленные ниже.

## Компоненты класса "MapInfoApplication"

### Методы

#### **Do**

Выполняет оператор MapBasic, как если бы Вы ввели его в окне MapBasic.

#### **Eval**

Выполняет выражение MapBasic и возвращает результат в виде строки.

### Properties

#### **FullName**

Возвращает полный дисковый маршрут к программе.

#### **LastErrorCode**

Возвращает целое число, номер последней ошибки MapBasic, происшедшей при вызове методов Do или Eval.

#### **LastErrorMessage**

Возвращает текст сообщения об ошибке по номеру LastErrorCode.

#### **Имя**

Возвращает имя программы.

#### **Версия**

Возвращает строку с номером версии, представляющую из себя номер версии, умноженный на 100.

### События

#### **MenuItemClick**

Срабатывает, когда пользователь выбирает пункт меню, заданный синтаксисом Calling OLE "MenuItemHandler".

#### **StatusBarTextChanged**

Срабатывает, когда меняется текст в строке состояний MapInfo Professional.

#### **WindowContentsChanged**

Срабатывает, когда изменяется содержимое окна Карты (например, при изменении масштаба).

## Интегрированная картография в .Net

Термином "Интегрированная картография" обозначают архитектуру программирования, позволяющую создавать программу с элементами картографии, которую пользователь может запускать вместо MapInfo Professional. Такая программа может запустить MapInfo Professional "на заднем плане", и пользователь получит доступ к Картам MapInfo прямо из этой программы.

В следующем разделе описано, как можно создать программу в .Net, использующую Интегрированную картографию. Более подробные сведения об Интегрированной картографии приводятся в одноименной главе этой книги.

При установке MapBasic добавляются примеры .Net-программ, представляющих Интегрированную картографию; см. каталог Samples\DotNet\IntegratedMapping. Эти примеры частично разобраны в следующем разделе.

## Доступ к MapInfo Professional посредством COM

В программе, использующей Интегрированную картографию, Вы будете пользоваться COM-интерфейсом MapInfo Professional. Чтобы это стало возможным, Вы должны создать ссылку на поле "Reference" в Вашем проекте в Visual Studio:

1. Создайте проект в Visual Studio.
2. В окне **Solution Explorer** поместите указатель мышки на папку "References", нажмите правую кнопку и выберите **Add Reference**.
3. В диалоге **Add Reference** откройте вкладку **COM**.
4. Выберите **MapInfo 10.0 OLE Automation Type Library** и нажмите **OK**.

Visual Studio создаст файл MAPINFO.INTEROP.DLL, который будет поставлять "объемлющие" (wrapper) .Net-классы для доступа к COM-интерфейсу MapInfo Professional. Теперь Ваша программа может использовать класс "MapInfoApplication". В программе-примере этот класс инициализируется в начале метода "MapForm.InitializeComObject":

```
_mapInfoApp = new MapInfoApplication();
```

Как только инициализирован объект "\_mapInfoApp", можно использовать его метод **Do** для выполнения операторов MapBasic (что аналогично вводу этих операторов в окно MapBasic в MapInfo Professional) или, пользуясь методом **Eval**, извлекать информацию о MapInfo Professional.

В частности, мы будем использовать метод **Do** для выполнения следующих операторов MapBasic:

1. **Set Application Window** – этот оператор позволяет использовать диалоги MapInfo в клиентских программах.
2. **Open Table** – этот оператор открывает таблицы MapInfo.
3. **Set Next Document Parent** – этот оператор переводит MapInfo Professional в специальное состояние, при котором каждое новое окно Карты будет "переподчинено" и появится в клиентской программе.
4. **Map From** – этот оператор создает окно Карты.

В программах-примерах демонстрируется использование этих операторов; ознакомьтесь также с тем, как класс "MapForm" вызывает метод **Do**.

## Методы обратного вызова (callback)

В некоторых случаях программы с Интегрированной картографией нуждаются в методах обратного вызова. Если программе нужно выполнить некие операции в ответ на определенные события – например, отреагировать на то, что пользователь изменил Карту – Вам придется создать метод обратного вызова, чтобы MapInfo Professional могла реагировать на такие события.

MapInfo Professional в примерах ниже использует следующие методы обратного вызова:

- Метод **WindowContentsChanged** вызывается MapInfo Professional при любом изменении содержимого окна Карты (например, при добавлении или удалении слоев).
- Метод **SetStatusText** вызывается MapInfo каждый раз, когда нужно изменить строку состояний в окне MapInfo Professional.
- Если в OLE-объекте создано меню и ему присвоен обработчик, реагирующий на изменение пункта этого меню; имя процедуры-обработчика задано в клиентской программе. В программе-примере создается одно меню в OLE-объекте и задается обработчик **MenuItemHandler**. Этот метод также присутствует в операторах MapBasic, задающих новые пункты меню (операторы **Create Menu** или **Alter Menu...Add**).

В примерах эти обратные вызовы представлены интерфейсом IMapInfoCallback. ерсия этого интерфейса на языке C# в файле MapInfoCallback.cs выглядит так:

```
public interface IMapInfoCallback
{
    // Метод вызывается MapInfo Professional при изменении окна
    int WindowContentsChanged(UInt32 windowID);

    // Метод вызывается MapInfo Professional при изменении текста в строке
    состояний
    int SetStatusText(string message);

    // Метод вызывается MapInfo Professional при изменении пункта меню в
    OLE-объекте
    void MenuItemHandler(string commandInfo);
}
```

Версия интерфейса на языке Visual Basic из программы MapInfoCallback.vb выглядит так:

```
Public Interface IMapInfoCallback
    ' Метод вызывается MapInfo Professional при изменении окна
    Function WindowContentsChanged(ByVal windowID As UInt32) As Integer

    ' Метод вызывается MapInfo Professional при изменении текста в строке
    состояний
    Function SetStatusText(ByVal message As String) As Integer

    ' Метод вызывается MapInfo Professional при изменении пункта меню в
    OLE-объекте
    Sub MenuItemHandler(ByVal commandInfo As String)
End Interface
```

В том же программном модуле содержится класс **MapInfoCallback**, который показывает как реализовать интерфейс "IMapInfoCallback". Заметим, что класс MapInfoCallback имеет атрибуты, позволяющие сдать класс COM-доступным, так чтобы MapInfo Professional могла вызывать эти методы. В языке C# атрибуты класса задаются так:

```
[ClassInterface(ClassInterfaceType.None)]
[ComVisible(true)]
public class MapInfoCallBack : IMapInfoCallback
```

В языке VB атрибуты класса задаются так:

```
<ClassInterface(ClassInterfaceType.None)> _
<ComVisible(True)> _
Public Class MapInfoCallBack
    Implements IMapInfoCallback
```

В том же файле, где определен интерфейс "IMapInfoCallback", задается и другой интерфейс, **ICallbackNotify**. Этот интерфейс можно использовать при создании программы типа Windows Forms. В программах-примерах, этот интерфейс описан в файлах MapForm.cs и в MapForm.vb.

Происходящие события могут понуждать MapInfo Professional обращаться к классу обратного вызова "IMapInfoCallback", который в свою очередь извещает класс клиента "ICallbackNotify". Чтобы лучше понять, как и когда вызываются те или иные методы интерфейса, рассмотрим следующую последовательность событий:

1. Пользователь запускает программу с клиентом Интегрированной картографии, который на заднем плане запускает MapInfo Professional. В следующем примере это происходит в методе **MapForm.InitializeComObject**. Этот метод запускает MapInfo Professional, создает экземпляр объекта для обратного вызова и регистрирует такой объект в MapInfo Professional:

```
private void InitializeComObject()
{
    // Создать объект MapInfo Professional
    _mapInfoApp = new MapInfoApplication();

    // Задать порождающие окно для диалогов MapInfo Professional
    _mapInfoApp.Do("Set Application Window " + this.Handle);

    // Создать объект для обратного вызова
    _callbackObject = new MapInfoCallBack(this);
}
```



```
// Зарегистрировать объект для обратного вызова
_mapInfoApp.RegisterCallback(_callbackObject);
}
```

2. Приложение-клиент вызывает метод **MapInfoApplication.Do**, чтобы открыть таблицы и окна Карт. В программе-примере это происходит в методе **MapForm.NewMap**, который вызывается, когда пользователь выполняет команду меню **File > Open** и открывает один или несколько TAB-файлов.
3. Пользователь вносит изменения в Карту. В программе-примере пользователь открывает диалог **Layer Control** ("Управление слоями") с помощью правой кнопки мыши, и изменяет Карту. MapInfo Professional управляет окном управления слоями.
4. Так как в Карте произошли изменения, MapInfo Professional извещает об этом программу-клиента, вызывая метод **MapInfoCallback.WindowContentsChanged**.
5. Метод **WindowContentsChanged** вызывает метод **MapForm.OnWindowContentsChanged**. Содержимое последнего метода зависит от характера программы. Например, если Ваша программа показывает информацию о Карте в строке состояний или еще где-либо, то Вы должны обновлять окно клиентской программы методом **OnWindowContentsChanged**.
6. Если в программе создан OLE-объект с меню, то MapInfo Professional при обращении к пунктам этого меню вызывает соответствующие методы-обработчики. В программе-примере оператор **Alter Menu** добавляет команду в контекстное меню и задает для нее обработчик "MenuItemHandler". Поэтому, когда пользователь выполнит эту команду, MapInfo Professional вызовет метод **MenuItemHandler**, который в свою очередь вызовет метод **MapForm.OnMenuItemClick**. Содержимое метода **OnMenuItemClick** будет зависеть от специфики выбранной команды.
7. Чтобы ни сделал пользователь такого, что бы вынудило MapInfo Professional изменить текст в строке состояний – например, выбор объектов на Карте мышкой – MapInfo Professional вызывает метод **MapInfoCallback.SetStatusText**. Этот метод вызывает метод **MapForm.OnStatusBarTextChanged**. Если Вы хотите сделать так, чтобы в Вашей программе строка состояний выглядела точно так же, как и строка состояний в MapInfo Professional, добавьте соответствующий код в метод **OnStatusBarTextChanged**.
8. По окончании действий мы должны отменить регистрацию объекта для обратного вызова. В программе-примере эту задачу выполняет метод **FormClosed**.

```
private void Form1_FormClosed(object sender, FormClosedEventArgs e)
{
    // Отменить регистрацию объекта обратного вызова
    MapInfoApp.UnregisterCallback(_callbackObject);
}
```

В программе-примере определен один пункт меню в OLE-объекте. Если Ваша программа задает несколько пунктов меню в OLE-объекте, то каждому нужно сопоставить отдельный метод. Другой способ состоит в том, что всем пунктам меню задается один метод, но тогда Вы должны каждому пункту меню задать уникальный ID-номер (что требует включения предложения ID в операторы **Create Menu** или **Alter Menu...Add**) с тем, чтобы уже внутри метода-обработчика с ними разобраться.

Напоследок, заметим, что класс **MapInfoCallback** универсален (re-usable). Вы можете создать несколько программ с интегрированной картографией и использовать в них один и тот же класс **MapInfoCallback**.

## Защита потоков

Чтобы понять логику действий класса **MapInfoCallback**, Вы должны прежде освоиться с концепцией многопоточности (multi-threading) и интерфейсом Windows Forms. Каждый раз, когда Вы выполняете код, который будет управлять интерфейсом Windows Forms – например, если нужно изменить в клиентской программе текст в строке состояний – Вы должны убедиться в том, что этот код действует в том же потоке, что и поток, в котором создавался интерфейс пользователя.

Следует следить за тем, чтобы методы обратного вызова функционировали в другом потоке, отличном от того, в котором действует интерфейс пользователя. Поэтому в методе обратного вызова желательно контролировать все, что может прямо или косвенно воздействовать на интерфейс пользователя.

Класс **Windows Forms Control** поддерживает свойство **InvokeRequired**. Если значение **InvokeRequired** истинно, это значит что текущий поток – это не тот поток, который имеет право обновлять элементы интерфейса (Control), и этом случае Вы должны использовать метод **Control.Invoke** для изменения любых элементов интерфейса. Метод **Invoke** дает гарантию того, что изменения применены в соответствующем потоке.

Например, метод из набора примеров "MapInfoCallback.SetStatusText" содержит следующий код, который гарантирует, что любые изменения в строке состояний происходят в соответствующем им потоке:


```
if (_callbackClient.InvokeRequired)
{
    _callbackClient.Invoke(this._onStatusBarTextChangedDelegate, new
Object[] { text });
}
else
{
    _callbackClient.OnStatusBarTextChanged(text);
}
```

Обратите внимание на то, что мы используем объект "\_callbackClient" (который реализует **ICallbackNotify**) для доступа к свойству **InvokeRequired** и методу **Invoke**. В программе-примере класс **Form** выступает как объект "ICallbackNotify":

```
public partial class MapForm : Form, ICallbackNotify
```

В примере компонент "\_callbackClient" является ссылкой на MapForm. Так как класс **Form** порожден классом **Control**, мы можем вызывать "\_callbackClient.Invoke".

---


 Не следует производить при обратном вызове (callback) операцию выхода из регистрации в методе-деструкторе, так как этот метод может быть вызван из некорректного потока.

---

# Примеры программ

В состав программного обеспечения MapBasic включены примеры программ.

---

 Дополнительные примеры могут быть добавлены после того, как этот документ уже был напечатан.

---

## В этой главе:

- ♦ Каталог Samples\Delphi .....284
- ♦ Каталог Samples\DLLEXAMP.....284
- ♦ Каталог Samples\DotNet.....284
- ♦ Каталог Samples\MapBasic .....284
- ♦ Каталог Samples\MFC.....291
- ♦ Каталог Samples\PwrBldr.....291
- ♦ Каталог Samples\VB4 .....291
- ♦ Каталог Samples\VB6 .....291

## Каталог Samples\Delphi

**tabmap:** программа на Delphi, которая запускает MapInfo Professional в качестве OLE-сервера.

## Каталог Samples\DLLEXAMP

### Каталог Samples\DLLEXAMP\Loadlib

**loadlib:** несколько файлов с исходными кодами на языке C для библиотеки DLL, которую можно скомпилировать либо для Win16, либо для Win32, и убедиться в работоспособности программы на MapBasic, из которой вызываются функции DLL.

### Каталог Samples\DLLEXAMP\ResDLL

Несколько программ, демонстрирующих совместимость Win16 и Win32.

## Каталог Samples\DotNet

### Каталог Samples\DotNet\HelloWorld

Простейший пример вызова .Net-метода из программы MapBasic (версии на языках C# и VB.Net ).

### Каталог Samples\DotNet\IntegratedMapping

Клиентское приложение Интегрированной Картографии, для использования в среде .Net (версии на языках C# и VB.Net ).

### Каталог Samples\DotNet\NamedViews

MapBasic-программа “Показ Карт” (Named Views), переписанное с использованием технологии .Net для управления XML-файлами и диалогами (версии на языках C# и VB.Net ). Эта программа позволяет присваивать имя Карте в текущем состоянии так, чтобы позже можно было ее открыть по этому имени в том же состоянии.

## Каталог Samples\MapBasic

В каталоге Samples\MapBasic находится несколько подкаталогов с файлами примеров программ. Содержимое каждого подкаталога описано в разделах ниже.

### Каталог Samples\MapBasic\Animator

**Animator.mb:** иллюстрирует ускорение перерисовки окон Карт со слоями анимации.

### Каталог Samples\MapBasic\Appinfo

**AppInfo.mb:** выводит информацию о работающих программах MapBasic.

### Каталог Samples\MapBasic\Autolbl

**AutoLbl.mb:** “Подписывает” Карту текстовыми объектами на Косметическом слое (эмуляция подписывания в ранних версиях MapInfo Professional).

### Каталог Samples\MapBasic\Cogoline

**COGOLine.mb:** рисует прямую заданной длины под заданным углом.

### Каталог Samples\MapBasic\Coordinateextractor

**Coordinateextractor.mb:** обновляет две колонки с координатами X- и Y-значениями координат объектов либо в проекции таблицы, либо в заданной пользователем проекции.

### Каталог Samples\MapBasic\Csb

**CoordSysBounds.mb:** проверяет и устанавливает границы рамки (минимальные и максимальны координаты объектов на Карте) любой базовой таблицы MapInfo.

### Каталог Samples\MapBasic\Database

**Autoref.mb:** обновляет связанные таблицы через заданный интервал времени в секундах

**BuildSQL.mb:** устанавливает соединение с базами данных СУБД; составляет, сохраняет и загружает запросы; выполняет запросы, допуская предварительный просмотр или полную загрузку результатов.

**Connect.mb:** выводит диалог **Управление соединением с СУБД** и некоторые другие функции, связанные с этим. В диалоге управления соединением можно выбрать установленное соединение, отключить соединение и установить новые.

**DescTab.mb:** выводит информацию о таблице.

**DLSUtil.mb:** возвращает элемент из окошка списка по его указателю в диалоге.

**GetMITab.mb:** диалог выбора таблиц MapInfo Professional.

**MIODbCat.mb:** программа для создания Каталога Карт. Позволяет администратору базу данных создавать таблицу Каталога Карт MAPINFO\_MAPCATALOG для пользователя MapInfo Professional. Кроме того, позволяет АБД удалять таблицу из Каталога.

**MIRowCnt.mb:** программа – счетчик записей таблицы СУБД. С помощью этой программы можно соединиться с базами данных СУБД и выполнить запрос count(\*) в таблицах, обновив Каталог Карт полученными результатами.

**MISetMBR.mb:** программа CoordSysBounds из каталога программ MapInfo Professional. С помощью этой программы Администраторы Базы Данных может изменить границы рамки таблицы в Каталоге Карт MapInfo\_MAPCATALOG.

**MIUpldDB.mb:** эта программа позволяет составлять SQL-запрос, специфичный для базы данных, с помощью которого можно загрузить таблицу MapInfo.

**MIUpLoad.mb:** программа **Присоединить геоинформацию к таблице SQL-сервера** из каталога программ MapInfo Professional. С помощью этой программы можно загрузить таблицу MapInfo в удаленную базу данных в специальную колонку с пространственной информацией. Колонки с пространственной информацией позволяют хранить геоинформацию MapInfo Professional в таблице удаленной базы данных.

**PickCol.mb:** с помощью этой утилиты можно интерактивно (в диалоговом режиме) выбирать колонки таблиц СУБД.

**PickSel.mb:** в этой утилите оформлен диалог выбора программы BuildSQL.mbx.

**PickTab.mb:** в этой утилите содержатся функции выбора из списка таблиц СУБД и их владельцев (схем), а также оформлен диалог выбора таблиц.

**PrepSQL.mb:** в этой утилите содержится функция подготовки SQL-запроса, обрабатывающая параметры запроса. Вычисляются границы действия всех параметров (обрабатываются и заменяются значениями).

**SQLPVW.mb:** в этой утилите параметры заменяются значениями, а в качестве результата выполнения возвращается строка с правильно подготовленным SQL-запросом с параметрами специфического для СУБД формата.

**SQLUtil.mb:** в этой утилите содержится несколько функций, позволяющих Mapinfo обеспечивать доступ к данным ODBC.

**SQLView.mb:** в этой утилите содержится готовое приложение SQL DataLink, предназначенное для проверки всех возможностей функции SERVER\_COLUMNINFO (кроме VALUE).

### Каталог Samples\MapBasic\Disperse

**disperse.mb:** с помощью этой программы можно рассеять точки вокруг заданной либо случайным образом, либо систематически.

### Каталог Samples\MapBasic\DistanceCalc

Программа DistanceCalc.MBX позволяет рассчитывать расстояния от объекта или группы объектов до ближайшего или наиболее удаленного объекта. Можно задать критерий, чтобы ограничить результаты.

### Каталог Samples\MapBasic\DMSCnvt

**DMSCnvt.mb:** с помощью этой программы можно пересчитывать десятичные значения координат в градусы/минуты/секунды.

### Каталог Samples\MapBasic\Geoset

**Geoset.mb:** эта программа позволяет создать набор данных Geoset для MapX или MapXtreme на основе слоев и оформления Карт MapInfo Professional, а также прочитать файлы MapX или MapXtreme Geoset, чтобы загрузить таблицы и параметры слоев в Карту MapInfo Professional.

### Каталог Samples\MapBasic\GridMakr

**GridMakr.mb**: эта программа создает сетку линий долгот/широт.

### Каталог Samples\MapBasic\HTMLImageMap

**HTMLImageMap.mb**: программа для создания HTML-Карт средствами MapInfo Professional для публикации на Web-страницах.

### Каталог Samples\MapBasic\IconDemo

**IconDemo.mb**: программа, демонстрирующая встроенные в MapInfo Professional пиктограммы панелей инструментов.

### Каталог Samples\MapBasic\Inc

**inc**: в этом каталоге находятся Include-файлы, которые можно использовать для создания MapBasic-программ.

Среди этих файлов:

- Файлы с определителями (DEF), используемыми многими программами, поставляемыми с MapInfo Professional. AUTO\_LIB.DEF и RESSTRNG.DEF требуются для модулей системы регистрации в каталоге программ и локализации ресурсов (оба файла хранятся в каталоге \LIB)
- MAPBASIC.DEF помимо прочего содержит определители универсальных макросов, логических констант, преобразований углов, цветов и длин строк. Все они используются в функциях MapBasic.
- MENU.DEF содержит определители необходимые для доступа и модификации диалогов, панелей инструментов и системы меню MapInfo Professional.
- MAPBASIC.H – вариант MAPBASIC.DEF плюс MENU.DEF для C++.
- MAPBASIC.BAS – вариант MAPBASIC.DEF плюс MENU.DEF для Visual Basic 6.0.

### Каталог Samples\MapBasic\Labeler

**labeler.mb**: программа, преобразующая подписи слоев в неизменяемые текстовые объекты, подписывающая выбранные объекты и преобразующая отдельные подписи, созданные при помощи утилиты "Автоподписи" в неизменяемые текстовые объекты.

### Каталог \MapBasic\Legends

**Legends.mb**: программа управляет показом в MapInfo Professional нескольких окон Легенды (стандартно в MapInfo Professional можно управлять единственным окном Легенды).

### Каталог Samples\MapBasic\Lib

**lib**: в этом каталоге находится библиотеки функций и подпрограмм, которые можно использовать для создания MapBasic-программ.

В частности, следующие два файла используются во многих MapBasic-программах, которые устанавливаются вместе с MapInfo Professional:

- AUTO\_LIB.MB используется многими программами для регистрации самих себя в каталоге программ.
- RESSTRNG.MB используется в локализованных программах, которые строки на соответствующем языке извлекают из STR-файлов.

### Каталог Samples\MapBasic\Linesnap

**linesnap.mb**: программа доводит линию до пересечения с другой линией или обрезает линию в точке пересечения с другой.

### Каталог Samples\MapBasic\Mapwiz

**mapwiz.mb**: утилита с шаблоном программы, который можно использовать в качестве менеджера программ.

### Каталог Samples\MapBasic\NorthArrow

**northarrow.mb**: MapBasic-программа, которая позволяет добавить в окно Карты или Отчёта стрелку направления на Север.

### Каталог Samples\MapBasic\Packager

**packager.mb**: программа сохраняет копию Рабочего набора со всеми данными, на которые в этом Рабочем наборе существуют ссылки.

### Каталог Samples\MapBasic\Regvector

**regvector.mb**: программа позволяет копировать векторные объекты таблицы (полигоны, полилинии, точки, и т.п.) в другое место Карты, указав три точки в исходной таблице.

### Каталог Samples\MapBasic\RingBuffer

**ringbuf.mb**: программа позволяет создавать множественные концентрические буферные зоны вокруг одного или более объектов. Кроме того, вычисляет суммы и средние значения для всех данных, попадающих в каждое их колец.

### Каталог Samples\MapBasic\RMW

**rotatemapwindow.mb**: программа, позволяющая вращать содержимое текущего окна Карты на заданное число градусов.

### Каталог Samples\MapBasic\RotateLabels

**rotatelabels.mb**: программа, позволяющая вращать подписи.



### Каталог Samples\MapBasic\RotateSymbols

**rotatesymbols.mb**: программа, позволяющая вращать условные знаки.

### Каталог Samples\MapBasic\SeamMgr

**seammgr.mb**: программа позволяет создавать Карту, сшитую из листов.

### Каталог Samples\MapBasic\Send2mxm

**send2mxm.mb**: программа создаёт MapX Geoset из активного окна Карты для передачи в мобильное устройство.

### Каталог Samples\MapBasic\Shields

**Shields.mb**: программа рисует декоративные рамки вокруг текстовых объектов (напр., дорожные знаки). Обратите внимание, что эта программа работает только с текстовыми объектами, а не с подписями.

### Каталог Samples\MapBasic\Snippets

В каталоге Snippets находятся примеры программ и их код, фрагменты которого можно включать в разрабатываемые программы MapInfo.



Кроме фрагментов готового кода программ в этом каталоге находятся три программы, которые устанавливаются в каталог программ MapInfo Professional. Это программы: "Показ Карт" [NIEWS.MBX], "Обзор" [OVERVIEW.MBX] и "Масштабная линейка" [SCALEBAR.MBX].

---

**acad.mb**: использует DDE для связи с AutoCAD for Windows.

**addnodes.mb**: фрагмент кода, в котором к объектам добавляются узлы. Может быть полезным при изменении проекции Карты; добавление узлов не позволяет появиться пустотам между областями при переходе в другую проекцию или масштаб, в случае если граница между областями была представлена одной длинной прямой линией.

**geocode.mb**: фрагмент кода, в котором показан процесс геокодирования с помощью MapBasic.

**geoscan.mb**: в этом фрагменте кода таблица проверяется на возможность геокодирования.

**get\_tab.mb**: это модуль, а не готовая программа. В программе содержатся процедуры с диалогом выбора таблицы из списка открытых таблиц. Пример использования этих процедур можно найти в программе OVERVIEW.MB ("Обзор").

**nviews.mb**: фрагмент содержит код программы "Показ Карт", с помощью которой можно переименовать окно Карты (с заданным центром и масштабом). После того, как создано новое именованное представление, к нему всегда можно будет вернуться, дважды щелкнув по его названию в диалоге "Показ Карт". Для того чтобы слинковать эту программу, используйте файл проекта nvproj.mbp.

**objinfo.mb:** в этом фрагменте кода выводится информация об объекте.

**overview.mb:** этот фрагмент кода открывает второе окно Карты с общим видом всей Карты в существующем окне. Если изменить масштаб показа исходной Карты, обзорная Карта автоматически изменится. Для того чтобы собрать эту программу, используйте файл проекта obproj.mbp.

**scalebar.mb:** этот фрагмент кода рисует масштабную линейку на Карте. Для того чтобы собрать эту программу, используйте файл проекта sbproj.mbp.

**textbox.mb:** пример программы, используемой в этом руководстве. Для того чтобы слинковать эту программу, используйте файл проекта tbproj.mbp.

**watcher.mb:** использует DDE для обмена данными с Microsoft Excel; книга Excel используется для контроля над глобальными переменными MapBasic-программы.

### Каталог Samples\MapBasic\Spider Graph

**Spider Graph:** эта программа строит линии между объектами одной или двух таблиц на основе общих значений данных. В полученной таблице можно хранить длину построенных линий.

### Каталог Samples\MapBasic\Srchrepl

**Srchrepl:** эта программа будет искать в символьной колонке указанную строку, и замещать ее другой заданной строкой.

### Каталог Samples\MapBasic\SWSpatialize

**sw\_spatialize:** эта утилита позволяет сделать таблицу SQL Server, ранее не настроенную на хранение координат, пространственной. После этого к таблице SQL Server можно будет присоединять геоинформацию, в ней можно будет хранить и извлекать пространственную информацию.

### Каталог Samples\MapBasic\Symbol

**symbol:** приложение позволяет Вам создавать, редактировать или удалять символы MapInfo. Символы, которые Вы создали или отредактировали, становятся частью стандартного набора символов MapInfo 3.0.

### Каталог Samples\MapBasic\SyncWindows

**syncwindows:** программа позволяет копировать окно Карты и синхронизировать открытые окна Карт таким образом, что масштаб и центр показа Карт совпадают для всех окон.

### Каталог Samples\MapBasic\Tablemgr

**tablemgr:** утилита, позволяющая получать информацию обо всех открытых таблицах, включая метаданные.

### Каталог Samples\MapBasic\Template

**toolapp.mb**: шаблон программы MapInfo Professional. Чтобы слинковать эту программу, используйте файл проекта toolapp.mbp.

### Каталог Samples\MapBasic\Winmgr

**winmgr**: эта программа позволяет задавать заголовок окна, или настраивать стандартный вид таблицы.

## Каталог Samples\MFC

**FindZip**: это приложение демонстрирует пример Интегрированной Картографии с элементами MapInfo Professional встроенными в C++ программу, написанную с использованием классов Microsoft Foundation (MFC).

**mdimfc**: C++ приложение Интегрированной Картографии.

## Каталог Samples\PwrBldr

**Capitals**: приложение Интегрированной Картографии написанное на PowerBuilder.



Runtime-библиотеки PowerBuilder не поставляются; для выполнения программы требуется наличие библиотек PowerBuilder.

---

## Каталог Samples\VB4

**Callback**: вызов функций OLE automation.

**FindZip**: это приложение демонстрирует пример Интегрированной Картографии с элементами MapInfo Professional, например, Картами, встроенными в программу Visual Basic. Требуется Visual Basic 3.0 или более поздняя версия этой программы.

**VMapTool**: пример использования сложных элементов Интегрированной Картографии, например, обратных вызовов (callback). Требуется Visual Basic 4.0 Professional Edition или более поздняя версия этой программы.

## Каталог Samples\VB6

**Callback**: вызов функций OLE automation.

**FindZip**: это приложение демонстрирует пример Интегрированной Картографии с элементами MapInfo Professional, например, Картами, встроенными в программу Visual Basic. Требуется Visual Basic 3.0 или более поздняя версия этой программы.

**VMapTool**: пример использования сложных элементов Интегрированной Картографии, например, обратных вызовов (callback). Требуется Visual Basic 4.0 Professional Edition или более поздняя версия этой программы.

# Сведения об операторах

Операторы производят некоторое действие над одним или несколькими значениями. Операторы можно классифицировать либо по типу значений, над которыми выполняются операции, либо по типу значения получаемого результата.

## В этой главе:

- ♦ Числовые операторы .....293
- ♦ Операторы сравнения .....293
- ♦ Логические Операторы .....294
- ♦ Географические операторы .....294
- ♦ Автоматическое преобразование типов .....296

## Числовые операторы

Следующие операторы выполняются с двумя числовыми значениями, а в результате должно быть получено числовое значение.

Оператор	Действие	Пример
+	сложение	$a + b$
–	вычитание	$a - b$
*	умножение	$a * b$
/	деление	$a / b$
\ (обратная наклонная черта)	целочисленное деление (остаток отброшен)	$a \setminus b$
Mod	остаток целочисленного деления	$a \text{ Mod } b$
^	возведение в степень	$a ^ b$

Два оператора из перечисленных выше могут быть использованы и в другом значении. Знак сложения (+) используется для конкатенации (объединения) двух строк в третью. Знак минус (–) вместе с единственным числом используется в качестве оператора отрицания, результатом выполнения будет числовое значение. Знак амперсанда (&) также используется для конкатенации строк.

Оператор	Действие	Пример
–	численное отрицание	$- a$
+	конкатенация строк	$a + b$
&	конкатенация строк	$a \& b$

## Операторы сравнения

С помощью операторов сравнения две величины одного типа сопоставляются друг другу, результатом выполнения оператора будет логическая величина TRUE или FALSE. Хотя операторы сравнения нельзя использовать для сопоставления числовых и нечисловых

данных (например, со строковыми выражениями), допустимо сравнение данных целых, коротких целых и вещественных типов. Операторы сравнения часто используются в выражениях с условием, например, **If...Then**.

Оператор	Возвращается TRUE если:	Пример
=	a равно b	a = b
<>	a не равно b	a <> b
<	a меньше b	a < b
>	a больше b	a > b
<=	a меньше или равно b	a <= b
>=	a больше или равно b	a >= b

## Логические Операторы

Логические операторы выполняются над логическими значениями, а результатом их выполнения является логическая величина либо TRUE, либо FALSE:

Оператор	Возвращается TRUE если:	Пример
And	оба операнда истинны	a And b
или	значение одного из операндов – истина	a Or b
Not	значение операнда FALSE	Not a

## Географические операторы

Географические операторы выполняются над объектами, а результатом их выполнения является логическая величина либо TRUE, либо FALSE:

Оператор	Возвращается TRUE если:	Пример
Contains	центроид второго объекта находится внутри первого объекта	objectA Contains objectB
Contains Part	Объект A содержит некоторую часть объекта B	objectA Contains Part objectB
Contains Entire	Объект A содержит весь объект B	objectA Contains Entire objectB

Оператор	Возвращается TRUE если:	Пример
Within	Центроид объекта А лежит внутри объекта В	<code>objectA Within objectB</code>
Partly Within	часть первого объекта помещается внутри второго	<code>objectA Partly Within objectB</code>
Entirely Within	первый объект полностью помещается внутри второго	<code>objectA Entirely Within objectB</code>
Intersects	Два объекта пересекаются хотя бы в одной точке	<code>objectA Intersects objectB</code>

## Старшинство выполнения операторов

Специальным типом операторов являются скобки, в которые можно заключать выражения внутри выражений. С помощью скобок можно изменить порядок выполнения операторов. В таблице ниже приведен порядок выполнения операторов MapBasic. Операторы из одной строки имеют одинаковый порядок выполнения. Операторы с более высоким приоритетом выполняются ранее других. Операторы имеющие одинаковый приоритет выполняются слева направо (кроме операторов возведения в степень, котрый выполняется справа налево).

Приоритет	Оператор MapBasic
(В первую очередь)	скобки
	возведение в степень
	отрицательный знак
	умножение, деление, Mod, целочисленное деление,
	сложение, вычитание
	географические операторы
	операторы сравнения, оператори Like
	Not
	And
(в последнюю очередь)	или

Например, результатом выполнения выражения  $3 + 4 * 2$  будет 11 (умножение выполняется до сложения). Результатом выполнения слегка измененного выражения  $(3 + 4) * 2$  будет 14 (сложение в скобках выполняется в первую очередь выполняется в первую очередь). Если есть сомнения, используйте скобки.

## Автоматическое преобразование типов

Если создать выражение обрабатывающее данные разных типов, то, для того чтобы получить осмысленный результат, MapInfo Professional будет автоматически преобразовывать выполнять типы данных. Например, если программа вычитает дату из другой даты, MapBasic вычислит целое значение (определяющее количество дней прошедшее между двумя датами). В таблице ниже приведен список правил, по которым MapBasic'ом выполняется автоматическое преобразование типов данных. В этой таблице слово "Целое" означает целое значение, которое может представлять собой целую переменную, короткую целую переменную или целую константу. Слово "Число" обозначает числовое выражение, которое обязательно является целочисленным.

Оператор	Комбинация операндов	Результат
+	Дата + Число	Дата типа Date
	Число + Дата	Дата типа Date
	Целое + Целое	Целое число типа Integer.
	Число + Число	Вещественное число типа Float.
	Другое + Другое	Строка
–	Дата – Число	Дата типа Date
	Дата – Дата	Целое число типа Integer.
	Целое – Целое	Целое число типа Integer.
	Число – Число	Вещественное число типа Float.
*	Целое * Целое	Целое число типа Integer.
	Число * Число	Вещественное число типа Float.
/	Число / Число	Вещественное число типа Float.
\ (обратная наклонная черта)	Число \ Число	Целое число типа Integer.
Mod	Число Mod Число	Целое число типа Integer.
^	Число ^ Число	Вещественное число типа Float.



# Поддерживаемые типы данных в ODBC-таблицах

Типы данных ODBC, которые поддерживает MapInfo Professional:

- SQL\_BIT
- SQL\_TINYINT
- SQL\_SMALLINT
- SQL\_INTEGER:
- SQL\_REAL
- SQL\_BIGINT
- SQL\_DECIMAL
- SQL\_DOUBLE
- SQL\_FLOAT
- SQL\_NUMERIC
- SQL\_BINARY
- SQL\_LONGVARBINARY
- SQL\_VARBINARY
- SQL\_LONGVARCHAR
- SQL\_DATE
- SQL\_TYPE\_DATE
- SQL\_TIMESTAMP
- SQL\_TYPE\_TIMESTAMP
- SQL\_TIME
- SQL\_TYPE\_TIME
- SQL\_CHAR
- SQL\_VARCHAR

# Присоединение геоинформации к удаленной таблице

## В этом приложении:

- ♦ Необходимые условия для хранения/получения пространственных данных<sup>299</sup>

## Необходимые условия для хранения/получения пространственных данных

Существует четыре необходимых условия для хранения и получения координат точек в таблицах СУБД.

1. Координаты пространственных объектов хранятся в числовых колонках или в колонках поддерживаемых пространственных типов данных.

Такие данные можно создать следующими способами:

- использовать существующие, заранее подготовленные данные;
- использовать программу "Easyloader" для загрузки в базу данных. Программа будет работать со всеми поддерживаемыми СУБД.

Эта процедура связана с созданием данных и может быть выполнена в любое удобное время.

2. Для того чтобы увеличить производительность обработки запросов к пространственным данным, можно создать колонку пространственного индекса. При необходимости это можно выполнить при загрузке. Эта процедура связана с созданием данных и может быть выполнена в любое удобное время.
3. MapInfo Professional хранит информацию о колонках, используемых для хранения координат, в специальной таблице СУБД, которая называется "Каталог Карт MapInfo" – MapInfo Map Catalog. Для каждой базы данных требуется единственный Каталог Карт. Создать Каталог Карт можно с помощью программы Easyloader или MIODBCAT.MBX. Для создания Каталога Карт вручную можно использовать процедуру, описанную в: разделе **Создание Каталога Карт MapInfo\_MapCatalog вручную в Приложении Е на стр. 300**. Эта операция выполняется единственный раз для каждой базы данных до того, как к таблицам этой базы данных будет присоединена геоинформация из MapInfo Professional.
4. MapInfo Professional обращается к Каталогу Карт за сведениями о таблицах с геоинформацией с помощью MapBasic-оператора **Server Create Map**. Эта операция выполняется единственный раз для таблицы до того, как к этой таблице будет присоединена геоинформация из MapInfo Professional.

# Создание Каталога Карт MapInfo\_MapCatalog вручную

Инструкции о том, как создать каталог MapInfo Map Catalog вручную и как представить удаленную таблицу в виде Карты (присоединить геоинформацию) – две процедуры, обязательные для геокодирования удаленных таблиц. Эта информация предназначена для тех пользователей, у которых нет доступа к MapInfo Professional.

Пользователи MapInfo Professional создают каталог MapInfo Map Catalog автоматически.

Вы или администратор вашей базы данных должны создать по одному каталогу MapInfo Map Catalog для каждой базы данных, в которую нужен доступ программам MapBasic.

Для того чтобы создать каталог Карт MAPINFO\_MAPCATALOG вручную:

1. Если СУБД требует наличия пользователей и паролей, то создайте в базе данных, в которой будут храниться таблицы с геоинформацией, пользователя с именем MAPINFO и паролем MAPINFO.
2. Создайте в этой базе данных таблицу MAPINFO\_MAPCATALOG. Оператор **Create Table** должен быть эквивалентным следующему SQL-оператору **Create Table**:

```
Create Table MAPINFO_MAPCATALOG (

    SPATIALTYPE           Float,
    TABLENAME            Char(32),
    OWNERNAME             Char(32),
    SPATIALCOLUMN          Char(32),
    DB_X_LL               Float,
    DB_Y_LL               Float,
    DB_X_UR               Float,
    DB_Y_UR               Float,
    VIEW_X_LL             Float,
    VIEW_Y_LL             Float,
    VIEW_X_UR             Float,
    VIEW_Y_UR             Float,
    COORDINATESYSTEM       Char(254),
    SYMBOL                Char(254),
    XCOLUMNNAME            Char(32),
    YCOLUMNNAME            Char(32),
    RENDITIONTYPE          Integer,
    RENDITIONCOLUMN        VarChar(32),
    RENDITIONTABLE         VarChar(32),
    NUMBER_ROWS            Integer.

)
```

Важно, чтобы структура в точности соответствовала описанной в этом операторе. Допускается только замена типов в базах данных, в которых поддерживаются типы данных varchar или text; этими типами данных можно заменить данные типа Char.

3. Создайте уникальный индекс этой таблицы по колонкам TABLENAME и OWNERNAME, тогда каждый пользователь сможет присоединять геоинформацию к единственной таблице.
4. Предоставьте право выполнения выборки (Grant Select) всем пользователям MAPINFO\_MAPCATALOG. После этого пользователи смогут присоединять геоинформацию к таблицам. Администратор базы данных может предоставлять пользователям права выполнять операции Update, Insert и Delete.

Типы пространственных индексов

В таблице перечислены типы пространственных индексов, поддерживаемые программой.

Тип пространственного индекса	Номер типа
Схема MapInfo MICODE (любая база данных)	1
Схема XY (любая база данных)	4
Геометрия Oracle Spatial	13
SpatialWare для SQL Server	14
Текст аннотации Oracle Spatial	16
SQL Server Spatial (с геометрией)	17
SQL Server Spatial (с географией)	18
PostGIS для PostgreSQL	19
SQL Server Spatial со значениями M и Z (с геометрией)	20
SQL Server Spatial со значениями M и Z (с географией)	21

Как присоединить геоинформацию к удаленной таблице

Для каждой таблицы из удаленной базы данных, к которой может обращаться MapBasic, необходимо добавить строку в таблицу MAPINFO\_MAPCATALOG. В MapInfo Professional это выполняется командой **Таблица > Изменить > Присоединить геоинформацию**.

Если для настройки Каталога Карт Map Catalog не используется MapInfo Professional, то необходимо вручную добавить записи о каждой таблице с географической информацией в таблицу Каталога Карт MAPINFO\_MAPCATALOG геокодированной базы данных. Каждая запись должна содержать следующую информацию о таблице.

Название колонки	Необходимые значения	Пример:
SPATIALTYPE	4.0 для таблиц географически проиндексированных по X,Y (планируется поддержка других пространственных серверов)	4.0
TABLERNAME	Название таблицы.	Drainage
OWNERNAME	Владелец.	Georgetown

Название колонки	Необходимые значения	Пример:
SPATIALCOLUMN	<p>Название колонки с пространственными данными, если такая существует. Имя:</p> <ul style="list-style-type: none"> <li>NO_COLUMN (для таблиц с присоединенной геоинформацией в виде пары координат X,Y)</li> </ul>	NO_COLUMN
DB_X_LL	X-координата левого, нижнего угла описывающего слой прямоугольника, в единицах заданных в MapInfo Professional параметром COORDINATESYSTEM.	-360
DB_Y_LL	Y-координата левого нижнего угла.	-90
DB_X_UR	X-координата правого, верхнего угла.	360
DB_Y_UR	Y-координата правого, верхнего угла.	90
VIEW_X_LL	X-координата левого, нижнего угла описывающего вид (view) прямоугольника, в единицах заданных в MapInfo Professional параметром COORDINATESYSTEM.	-360
VIEW_Y_LL	Y-координата левого нижнего угла.	-90
VIEW_X_UR	X-координата правого, верхнего угла.	360
VIEW_Y_UR	Y-координата правого, верхнего угла.	90
COORDINATESYSTEM	<p>Строка с названием проекции или координатной системы MapInfo, которая задает используемые для карты проекцию, единицы измерения и др. Допускаются следующие значения:</p> <ul style="list-style-type: none"> <li>Earth Projection 1,0 (для NAD27)</li> <li>Earth Projection 1,62 (для NAD27)</li> <li>Earth Projection 1,33 (для NAD 83) или</li> <li>Earth Projection 1,74 (для NAD 83)</li> </ul>	Earth Projection 1,0
SYMBOL	Symbol-оператор MapInfo с параметрами (для слоя с точками)	Symbol (35,0,12)
XCOLUMNNAME	Задаёт название колонки, содержащей X-координаты.	NO_COLUMN

Название колонки	Необходимые значения	Пример:
YCOLUMNNAME	Задаёт название колонки, содержащей Y-координаты.	NO_COLUMN
RENDITIONTYPE	Требуется ввести 1, если рендеринг включен, 0, если выключен.	1
RENDITIONCOLUMN	Задаёт название колонки для рендеринга.	MI_SYMBOLLOGY
RENDITIONTABLE	Задаёт название таблицы для рендеринга. Это поле не используется.	оставьте пустым
NUMBER_ROWS	Задаёт количество записей в таблице.	11



# О служебных и вспомогательных файлах

## В этом приложении:

- ♦ Обновление программ, созданных с использованием версий более ранних чем 6.5 .....312
- ♦ Файлы и каталоги данных приложения .....315
- ♦ Стандартные маршруты .....317
- ♦ Изменения в реестре .....317
- ♦ Требования для установки и политики групп .....318

## Обновление программ, созданных с использованием версий более ранних чем 6.5

Вспомогательные и служебные файлы – неисполняемые файлы, содержащие данные, которые используются MapInfo Professional в процессе работы. Начиная с версии 6.5 используются следующие файлы и папки:

Имя файла	Описание
MAPINFOW.PRF	Файл настроек, сделанных заранее
MAPINFOW.WOR	Стандартный Рабочий набор
STARTUP.WOR	Стартовый Рабочий набор
MAPINFOW.CLR	Файл раскраски
MAPINFOW.PEN	Файл штрихов линий и контуров
MAPINFOW.FNT	Файл символов условных знаков
CUSTSYMB	Каталог растровых символов
THMTMPLT	Каталог тематических шаблонов
GRAPHSUPPORT	Каталог с шаблонами графиков

В ранних версиях MapInfo (до 6.5) эти файлы содержались в каталоге Windows или в каталоге “Program”. Начиная с версии 6.5, установка служебных и вспомогательных файлов производится в личный каталог пользователя, что облегчает их поиск и совместное использование. Это удобно, например, в том случае, когда при работе с разным версиям MapInfo приходится использовать проекции из файла MAPINFOW.PRJ.

Следующие файлы остались в каталоге “Program”:

Имя файла	Описание
MAPINFOW.ABB	Файл сокращений
MAPINFOW.PRJ	Файл проекций
MAPINFOW.MNU	Файл меню

### Помните:

- Программа установки не спрашивает пользователя, где он хочет разместить эти файлы.
- Установщик всегда работает одинаково, независимо, установлена ли уже программа MapInfo Professional или нет.

- Не существует "обновляющей" установки до версии 6.5 и выше (т.е. Вы не можете установить Mapinfo Professional 10.0 в тот же каталог, где была установлена 9.5, установщик зафиксирует ошибку).
- Разработчики приложений могут перемещать или копировать файлы куда захотят, но Mapinfo Professional будет искать их в следующем порядке:
  - appdata\_dir
  - local\_appdata\_dir
  - pref\_dir
  - program\_dir

## Словарь терминов, используемых при обновлении программ

Полезно будет знать следующие определения:

*Личный каталог пользователя (user profile root)*

Структура папок пользователя. Каждый пользователь может производить запись в подчиненные папки. Их размещение зависит от версии Windows:

Windows XP : c:\Documents and Settings\имя\_пользователя

My Documents

Windows XP : c:\Documents and Settings\имя\_пользователя\My Documents

### **Pref\_dir**

По умолчанию Mapinfo Professional записывает в эту папку файлы "mapinfow.prf" и "mapinfow.wor".

- Версия 6.0: каталог операционной системы Windows
- Начиная с версии 6.5: *Личный каталог пользователя*\Application Data\MapInfo\MapInfo. Если такой папки не существует, то Mapinfo Professional создает ее.

### **home\_dir**

- Каталог Windows.

### **program\_dir**

MapInfo Professional 6.0 размещает большинство вспомогательных файлов в этой папке.

Версия 6.0: папка, где находится файл MAPINFOW.EXE

Начиная с версии 6.5: папка, где находится файл MAPINFOW.EXE

### **appdata\_dir**

Эта папка для хранения вспомогательных файлов, принадлежащих каждому пользователю, введена в версии 6.5. В ней размещаются многие служебные (вспомогательные) файлы.

- Версия 6.0: нет

- Начиная с версии 6.5: *Личный каталог пользователя*\Application Data\MapInfo\MapInfo\Professional\*nnn*.

где *nnn* – трехзначный номер версии MapInfo Professional (например, 850).



Если такой папки не существует при запуске программы, MapInfo Professional ее не создает. Разработчики программ не должны рассчитывать на обязательное существование этой папки.

---

### **local\_appdata\_dir**

Эта папка также определяется пользователем (аналогично appdata\_dir).

- Версия 6.0: нет
- Версия 6.5 и более новые: *Личный каталог пользователя*\Local Settings\Application Data\MapInfo\MapInfo\Professional\*nnn*,

где *nnn* – трехзначный номер версии MapInfo Professional (например, 850).



Если такой папки не существует при запуске программы, MapInfo Professional ее не создает. Разработчики программ не должны рассчитывать на обязательное существование этой папки.

---

### **common\_appdata\_dir**

Этот каталог предназначен для совместного доступа всех пользователей компьютера. По умолчанию, пользователи имеют неограниченный доступ к файлам в этого каталога: чтение всех файлов, возможность создания файлов и возможность изменения содержимого персонально созданных файлов. Файлы из этого каталога не переносятся в другие каталоги. Поддержка этого каталога появилась только в версии 7.0.

- Версии 6.5 и 6.0: нет
- Версия 7.0 и более новые: *Личный каталог пользователя*\All Users\Application Data\MapInfo\MapInfo\Professional\*nnn*,

где *nnn* – трехзначный номер версии MapInfo Professional (например, 850).

### **mydocs\_dir**

Соответствует каталогу “My documents” (Мои документы) пользователя.

- Версия 6.0: нет
- Версия 6.5 и более новые: My documents

### **search\_for\_file**

С помощью этой функции можно найти файлы приложения appdata. Она ищет папки для файлов в следующей очередности:

- Версия 6.0: pref\_dir, home\_dir, program\_dir
- Версия 6.5: appdata\_dir, local\_appdata\_dir, pref\_dir, program\_dir

- Версия 7.0 или более новые: appdata\_dir, local\_appdata\_dir, pref\_dir, common\_appdata\_dir, program\_dir

## Файлы и каталоги данных приложения

Следующий список описывает, как MapInfo Professional версий 6.0, 6.5 и выше проводит поиск вспомогательных файлов. В версии 9.5 появился вариант установки MapInfo Professional для рабочих групп, с помощью которого системный администратор может задавать совместно используемые каталоги и папки, в которых хранятся служебные файлы. В случае использования этого варианта установки стандартные адреса служебных файлов могут измениться.

### MAPINFOW.PRF

- Версия 6.0: используется функция search\_for\_file. Независимо от того, откуда был считан файл, он всегда записывает в pref\_dir.
- Начиная с версии 6.5: используется функция search\_for\_file. Если файл найден, то программа читает файл и запоминает его местоположение.

При завершении работы, если файл был найден при загрузке, а пользователь имеет право записи, то файл будет сохранен здесь. Иначе, файл будет записан в pref\_dir.

### MAPINFOW.WOR

- Версия 6.0: ищет в pref\_dir, затем в home\_dir. Будет загружен первый найденный файл.
- Начиная с версии 6.5: используется функция search\_for\_file. Если файл найден, то программа читает файл и запоминает его местоположение.

При завершении работы, если файл был найден при загрузке, а пользователь имеет право записи, то файл будет сохранен здесь. Иначе, файл будет записан в pref\_dir.

### STARTUP.WOR

- Версия 6.0: загружаются из каталогов в заданном порядке: program\_dir, pref\_dir.
- Начиная с версии 6.5: загружает из каталогов по порядку: pref\_dir, appdata\_dir, local\_appdata\_dir, pref\_dir, program\_dir. В отличие от других файлов приложений, каждый найденный STARTUP.WOR обрабатывается.

### MAPINFOW.CLR

- Версия 6.0: используется функция search\_for\_file. Если файл не находится, то открывается диалог с предложением пользователю найти его самостоятельно.

Если пользователь решил записать собственные цвета, то новый файл с палитрой цветов будет записан в pref\_dir (или будет перезаписан существующий там файл).

- Начиная с версии 6.5: используется функция search\_for\_file. Если файл не находится, то открывается диалог с предложением пользователю найти его самостоятельно.

Если пользователь хочет записать собственные цвета и файл цветов находится в пользовательской директории, то MapInfo Professional обновит существующий файл. Если файл цветов был считан из программного каталога или если пользователь не имеет права записи в этот файл, то MapInfo Professional запишет этот файл в pref\_dir.

### MAPINFOW.PEN

- Версия 6.0: используется функция `search_for_file`. Если файл не находится, то открывается диалог с предложением пользователю найти его самостоятельно.
- Начиная с версии 6.5: используется функция `search_for_file`. Если файл не находится, то открывается диалог с предложением пользователю найти его самостоятельно.

### MAPINFOW.FNT

- Версия 6.0: используется функция `search_for_file`. Если файл не находится, то открывается диалог с предложением пользователю найти его самостоятельно.
- Начиная с версии 6.5: используется функция `search_for_file`. Если файл не находится, то открывается диалог с предложением пользователю найти его самостоятельно.

### Каталог CUSTSYMB

- Версия 6.0: предполагается, что файл находится в `program_dir`.
- Начиная с версии 6.5: ищет каталог символов, используя `search_for_file`.
- Если файл не находится, то предполагается, что файл находится в `program_dir`.

### Каталог THMTMPLT

- Версия 6.0: Если каталог тематических шаблонов, указанный в файле настроек, существует, то используется он. В противном случае программа попытается создать каталог шаблонов под папкой `program_dir`. Если программа не сможет создать каталог под `program_dir`, то такого каталога не будет.

Во всех случаях, включая последний, MapInfo Professional обновляет маршрут к файлу настроек.

- Версия 6.5: Если каталог тематических шаблонов, указанный в файле настроек, существует, то используется он. Иначе, ищется папка с шаблонами с помощью `search_for_file`. Если она находится, то она и используется. В противном случае, программа пытается создать каталог шаблонов под `appdata_dir`, а затем в `program_dir`. Если это не удастся, каталог шаблонов не будет создан. В любом случае, MapInfo Professional не устанавливает путь к файлу настроек.

### Каталог GRAPHSUPPORT

- Версия 6.0: использует каталог, указанный в настройках, независимо от того, существует ли файл настроек. Если указанный каталог отсутствует, то при попытке создать новый график появится сообщение об ошибке.
- Версия 6.5: Если каталог тематических шаблонов, указанный в файле настроек, существует, то используется он. Иначе, ищется папка с шаблонами графиков с помощью `search_for_file`. Если она находится, то она и используется. Если она не находится, то предполагается, что ее нет в `program_dir`, и пользователь получит сообщение об ошибке при попытке создать график.



В версии 7.0 программа `search_for_file` включает в поиск `common_appdata_dir`.

---

## Стандартные маршруты

В версии 9.5 появился вариант установки MapInfo Professional для рабочих групп, с помощью которого системный администратор может задавать совместно используемые каталоги и папки, в которых хранятся служебные файлы. В случае использования этого варианта установки стандартные адреса некоторых служебных файлов могут измениться.

В следующей таблице перечислены настраиваемые адреса каталогов и адреса, используемые по умолчанию:

Адрес каталога или папки	Версии 6.5 и 6.0	Начиная с версии 7.0
Таблицы	mydocs_dir	mydocs_dir
Рабочие наборы	mydocs_dir	mydocs_dir
Программы MapBasic	program_dir\Tools	program_dir\Tools
Импортированные файлы	mydocs_dir	mydocs_dir
SQL-запросы	mydocs_dir	mydocs_dir
Шаблоны тематических карт	appdata_dir\thmtmpl, если существует, иначе, program_dir\thmtmpl	используется search_for_file, при ошибке program_dir
Сохраненные запросы	mydocs_dir	mydocs_dir
Новые регулярные поверхности	mydocs_dir	mydocs_dir
Файлы Crystal Report	mydocs_dir	mydocs_dir
Файлы, обеспечивающие работу с графиками	local_appdata_dir, если существует, иначе, program_dir otherwise	используется search_for_file, при ошибке program_dir
Поиск каталогов и таблиц	mydocs_dir	mydocs_dir

## Изменения в реестре

Использование реестра MapInfo Professional должно быть организовано так, чтобы каждый пользователь мог работать со своими данными. Следующие изменения были сделаны, чтобы поддержать такую организацию работы:

- Ключи Каталога Программ (Tool Manager) теперь устанавливаются в следующей ветке реестра – HKEY\_CURRENT\_USER.
- Ключи настройки цветов и форматов численных значений, сделанные пользователями, теперь хранятся в HKEY\_CURRENT\_USER.

## Требования для установки и политики групп

### MapBasic v.6.5 и 6.0

Вспомогательные файлы MapBasic версий 6.5 и 6.0 должны быть установлены в каталоги, как указано в следующей таблице:

Имя файла	Версии 6.5 или 6.0*
MAPINFOW.CLR	Application Data\MapInfo\MapInfo\Professional\ <i>nnn</i>
MAPINFOW.PEN	Application Data\MapInfo\MapInfo\Professional\ <i>nnn</i>
MAPINFOW.FNT	Application Data\MapInfo\MapInfo\Professional\ <i>nnn</i>
MAPINFOW.ABB	Каталог программы
MAPINFOW.PRJ	Каталог программы
MAPINFOW.MNU	Каталог программы
CUSTSYMB	Application Data\MapInfo\MapInfo\Professional\ <i>nnn</i>
THMTMPLT	Application Data\MapInfo\MapInfo\Professional\ <i>nnn</i>
GRAPHSUP- PORT	Local Settings\Application Data\MapInfo\MapInfo\Professional\ <i>nnn</i>

\* где *nnn* – трехзначный номер версии MapInfo Professional (например, 850).




## MapBasic v.7.0 и более новые версии

Начиная с версии 7.0 вспомогательные файлы MapBasic должны быть установлены в директории, как указано в следующей таблице:

Имя файла	Начиная с версии 7.0*
MAPINFOW.CLR	Application Data\MapInfo\MapInfo\Professional\nnn
MAPINFOW.PEN	Application Data\MapInfo\MapInfo\Professional\nnn
MAPINFOW.FNT	Application Data\MapInfo\MapInfo\Professional\nnn
MAPINFOW.ABB	Каталог программы
MAPINFOW.PRJ	Каталог программы
MAPINFOW.MNU	Каталог программы
CUSTSYMB	Application Data\MapInfo\MapInfo\Professional\nnn
THMTMPLT	Application Data\MapInfo\MapInfo\Professional\nnn
GRAPHSUPPORT	All Users Application Data\MapInfo\MapInfo\Professional\nnn

\* где *nnn* – трехзначный номер версии MapInfo Professional (например, 850).

 В версии 9.5 появился вариант установки MapInfo Professional для рабочих групп, с помощью которого системный администратор может задавать совместно используемые каталоги и папки, в которых хранятся служебные файлы. В случае использования этого варианта установки стандартные адреса служебных файлов могут измениться.

Следующие функции MapBasic были добавлены, чтобы помочь разместить вспомогательные файлы:

- Функция **GetFolderPath\$ ( )** – возвращает путь к папке MapInfo Professional или Windows.
- Функция **LocateFile\$ ( )** – возвращает путь к одному из вспомогательных файлов MapInfo Professional.

# Словарь MapBasic

Если вы не найдете термин в этом словаре, попробуйте найти его в словаре терминов *Руководства пользователя MapInfo Professional*.

## Термины

Термин	Определение
Automation, OLE Automation	OLE Automation – технология, позволяющая Windows-приложениям взаимодействовать друг с другом. Например, приложение написанное на Visual Basic может управлять MapInfo Professional используя методы и свойства OLE объекта MapInfo Professional. См. <b>Интегрированная картография</b> .
DDE	См. <b>Динамический обмен данными (Dynamic Data Exchange) – DDE</b> .
MBX	Исполняемый файл MapBasic, который может быть запущен в MapInfo Professional с использованием команды <b>Программы &gt; Запустить программу MapBasic</b> . Любой пользователь MapInfo Professional может запускать файлы MBX. Чтобы создать файл MBX, необходимо использовать среду разработки программ – MapBasic.
Object Linking and Embedding (OLE)	Технология, позволяющая объект созданный в одном приложении, использовать в другой программе. Объектом может быть Карта, график, электронная таблица, звук, текст и т.п. Внедрение (Embedding) – процесс вставки объекта из приложения-сервера в приложение-контейнер.
Runtime	Специальная версия MapInfo Professional, в которой сохранены все возможности работы с географическими данными и базами данных полной версии, но исключены меню и Панели Инструментов. Используется для создания специально подготовленных версий MapInfo Professional.
Анимационный слой	Специальный “плавающий” слой, добавленный на Карту, объекты которого перерисовываются отдельно от объектов других слоев Карты. Изменение положения или оформления объекта на анимационном слое не приводит к перерисовке объектов объектов на остальных слоях.
Аргумент	Иногда применяется синоним – параметр. Часть оператора или функции. Если оператор или функция требует применения одного аргумента или большего их количества, необходимо составить соответствующее каждому аргументу выражение. Заданный аргумент передается оператору или функции. На синтаксических диаграммах в <i>Справочнике MapBasic</i> и <i>Справочной системе</i> аргументы выделены курсивом.
Всплывающее меню	Меню, которое появляется после нажатия на правую кнопку мыши.

Термин	Определение
Выражение	Группа из одной или более переменных, констант, функций, ссылок на таблицы и операторы.
Вычисляемые колонки	Колонка в таблице запросов, содержащая значения рассчитанные на основе выражения и данных исходной таблицы. Смотрите также оператор <b>Add Column</b> .
Географическое объединение (Geographic Join)	Реляционная связь между двумя таблицами, использующая географические критерии для объединения (например, определением какие точки одной таблицы находятся внутри полигонов другой таблицы).
Глобальная система позиционирования (GPS)	Программно-аппаратная система, принимающая спутниковые сигналы для определения точного местоположения на местности.
Глобальная переменная	Переменная, определяемая в начале программы и доступная в любой процедуре или функции. Создается с использованием оператора <b>Global</b> .
Градусы	Обычно на бумажных Картах координаты задаются в градусах, минутах и секундах (например, 55 градусов, 36 минут северной широты); в операторах MapBasic, однако, используются десятичные координаты (например, 42.6 градусов). Смотрите также: <b>Широта, Долгота</b> .
Данные СУБД, хранящиеся удаленно данные	Данные, которые хранятся в базе данных, например, на сервере Oracle или SYBASE.
Динамический обмен данными (Dynamic Data Exchange) – DDE	Специальный протокол Microsoft Windows, позволяющий различным приложениям обмениваться инструкциями и данными. Оба приложения должны поддерживать протокол DDE для успешного обмена.
Динамически-связанная библиотека (Dynamic Link Library) – DLL	Файлы Microsoft Windows, содержащие подпрограммы и другие ресурсы для исполнимых файлов. Файлы DLL обычно вызываются из программы для того, чтобы решить некоторую задачу, и вернуть в основную программу результат.
Долгота	Тип координат, исчисляемый в градусах, указывающий положение к востоку-западу от Гринвича. Размещение западней Гринвича будет иметь отрицательные значения долготы.
Запись	Строка в таблице или базе данных. Каждая запись выглядит как одна строка в окне Списка.

Термин	Определение
Изограмма	Изограмма – это линия их точек, удовлетворяющих заданным условиям на дальность и время. Изограммами являются изохроны и изолинии зоны транспортной доступности по расстоянию.
Изолиния дистанционная (зона транспортной доступности)	Граница зоны транспортной доступности по расстоянию – полигон или последовательность точек определяющих территорию, любую точку которой можно достичь от исходной точки, пройдя заданное расстояние по данной сети дорог.
Изохрона	Граница зоны транспортной доступности по времени – полигон или последовательность точек, определяющих территорию, любую точку которой можно достичь от исходной точки за заданное время по данной сети дорог.
Индикатор выполнения	Стандартный диалог с горизонтальной бегущей полоской, отображающей процент выполнения задачи.
Интегрированная Картография	Технология, позволяющая помещать окно Карты MapInfo Professional в другое приложение (например, программу написанную на языке Visual Basic). См. <b>Интегрированная картография</b> .
Клиент	Приложение, которое получает информацию от другой программы. Часто используется в отношении соединений с базами данных или по DDE.
Ключевое слово	Слово являющееся частью языка программирования, например оператор или название функции. В документации по MapBasic ключевые слова выделяются <b>жирным</b> шрифтом.
Код программы, исходный код	Некомпилированный текст программы. В MapBasic – это MB-файл.
Колонка	Часть таблицы или базы данных. Таблица состоит из одной или нескольких колонок, содержимое каждой из которых отображает категорию информации (например, имя, адрес, номер телефона и т. п.). Иногда колонки называются “поля”. Таблицы с растровыми изображениями не содержат колонок.
Комментарии	Замечания программиста в тексте программы. Замечания не влияют на синтаксис программы, требующийся для компиляции. В языке MapBasic для выделения начала комментария используется символ апострофа (одиночная кавычка – ‘). При появлении апострофа в тексте программы, MapBasic пропускает остаток строки (кроме случая, когда апостроф появляется внутри литерала строкового выражения).

Термин	Определение
Компилятор	Программа-транслятор, обрабатывающая текст программы, проверяет синтаксис и, если нет ошибок, переводит текст кода в результирующую программу на языке машинных команд.
Косметический слой	Временный слой, существующий на каждой Карте. Этот слой всегда самый верхний и перечислен первым в списке диалога <b>Управления слоями</b> . Команда <b>Найти</b> MapInfo Professional при поиске наносит условные знаки на косметическом слое.
Круговая диаграмма	Круг, разделённый на сектора, представляющие процентные отношения величин. MapInfo Professional может отображать круговую диаграмму в окне Графика или на тематической Карте.
Курсор, курсор мыши, курсор строк	Курсор мыши (указатель мыши), обычно отображается в виде стрелки, перемещается по экрану повторяя движения мыши или другого манипулятора (например джойстика). Нажатие кнопок и другие действия мыши применяются к текущему положению курсора. Курсор строк определяет положение текущей строки таблицы; перемещение по строкам выполняется оператором <b>Fetch</b> .
Локальная переменная	Переменная, определённая и используемая в определённой функции или процедуре. Локальные переменные имеют приоритет над глобальными переменными с тем же названием. Создаётся с использованием оператора <b>Dim</b> .
Локальная переменная модуля	Переменная, доступная из любой функции или процедуры MB-файла программы, но которая не может быть доступной из других MB-файлов, входящих в тот же проект. Создается оператором <b>Dim</b> , помещённым за пределами функций и процедур.
Массив	Последовательность однотипных элементов, число которых фиксировано и которым присвоено одно имя.
Метаданные	Информация о таблице (такая как дата создания, информация об авторских правах и т.п.), сохранённая в TAB-файле, в специальном разделе (не в строках и колонках). См. <b>Работа с таблицами</b> .
Методы, OLE-методы	Часть механизма OLE Automation. Вызов методов приложения подобен вызову процедур в программе. См. <b>Интегрированная картография</b> .
Модуль	Файл с текстом программы (MB-файл), являющийся частью проекта.

Термин	Определение
Недоступно (disabled)	Режим, при котором часть элементов интерфейса программы (команды меню, элементы управления диалога, кнопки Панели Инструментов) недоступна для пользователя. Заблокированные, недоступные элементы обычно выглядят светло-серыми, указывая что этот элемент недоступен в данное время. Смотрите также: <b>Разрешенно (enabled)</b> .
Область определения переменных	Определяет: может ли переменная вызываться из любой части программы (глобальные переменные) или только из определенной функции или процедуры (локальные переменные). Если для процедуры задана локальная переменная с именем совпадающим с именем глобальной переменной, то локальная переменная будет иметь преимущество; при любом обращении к переменной из процедуры будет использоваться локальная переменная.
Обработчик событий (Handler)	Процедура в программе. Когда в программе происходит определённое событие (например, пользователь выбрал команду меню), обработчик определяет, какое действие произвести в ответ на это событие.
Объект	Графический объект, который может быть помещён в окне Карты или Отчета (например, линии, точки, окружности и т.п.). Переменная объект языка MapBasic – это переменная, которая может содержать графический объект. В специальной колонке "Object" хранятся характеристики объектов таблицы. OLE-объект – это часть технологии Windows (например, "ухватить и перетащить").
Окно MapBasic.	Окно в интерфейсе MapInfo Professional. В MapInfo Professional вызывается из меню <b>Настройки</b> командой <b>Показать окно MapBasic</b> . Вы можете ввести операторы MapBasic в окне MapBasic и выполнить их без компиляции программы.
Оператор	Специальный символ или слово, которое обрабатывает одну или более констант, переменных или других значений. Например, оператор минус (–) вычитает одно число из другого.
Оператор MapBasic	Инструкция MapBasic-программы. Для компиляции MapBasic-программы можно использовать оператор, разнесенный по двум или нескольким строкам.
По ссылке, по значению	Два разных способа передачи параметров функции или процедуре. При передаче параметра по ссылке (используется по умолчанию) при вызове функции необходимо указать имя переменной, при этом, вызываемая функция может менять заданную переменную. При передаче параметра по значению (пользуясь зарезервированным словом <b>ByVal</b> ), задавать имя переменной не требуется.

Термин	Определение
Панель Инструментов	Набор кнопок. Пользователь может “закрепить” панель инструментов, перетаскив её к верхнему краю окна MapInfo Professional. В документации MapBasic панель инструментов часто называется “ButtonPads” потому что <b>ButtonPad</b> – ключевое слово языка MapBasic, используемое для изменения Панели Инструментов.
Панель кнопок (ButtonPad)	Иное название Панели Инструментов.
Папка	Область для хранения файлов; также используется название – директория или каталог.
Параметр	Синоним для “аргумент.”
Переменная	Область памяти, выделенная для хранения значения.
Период исполнения	Период, в течении которого выполняется программа. Ошибки исполнения возникают при выполнении программы (MBX-файл).
Платформа	Операционная система компьютера (например, Windows, Linux).
Подпрограмма	Группа операторов; в MapBasic также называется процедурой или стандартной процедурой.
Подпрограмма-обработчик	Группа операторов, с помощью которой решается определенная задача; например, можно использовать оператор <b>OnError</b> , чтобы задать группу операторов, которые будут выполняться в случае возникновения ошибки.
Подсказка в строке состояния.	Подсказка, появляющаяся в строке сообщений при выборе пользователем команд меню или при помещении курсора над кнопкой Панели Инструментов.
Подсказка для инструмента	Короткое описание назначения кнопки Панели Инструментов; появляется после задержки курсора над кнопкой.
Процедура	Группа операторов, помещённая в конструкцию <b>Sub... End Sub</b> . Иногда называется подпрограммой.
Проект, файл Проекта	Проект – это набор модулей. Файл проекта (MBP-файл) – это текстовый файл, содержащий перечень модулей. Результатом компиляции всех модулей в проекте и их сборки является приложение (MBX-файл).
Прозрачная штриховка	Это, например, линейная или сетчатая штриховка, через которую можно видеть, что находится под ней.



Термин	Определение
Псевдоним (Alias)	Название или имя, которое используется в MapInfo Professional (или в MapBasic-программе) для обращения к открытой таблице. Например, если полным именем таблицы является "C:\MapInfo\Parcels.Tab", то ее синонимом будет "Parcels". Псевдонимы таблицы могут не содержать пробелов; все пробелы в имени таблицы в псевдониме будут заменены символами подчеркивания. Псевдоним является также типом данных MapBasic; в переменной псевдонима можно хранить строковое выражение с названием (именем) колонки (например, "World.Population"). Максимальная длина псевдонима 32 символа.
Разрешенно (enabled)	Режим, обратный режиму <b>Недоступно (disabled)</b> ; условие при котором команды меню, элементы диалога и кнопки Панели Инструментов доступны для использования.
Растр	Формат изображения, образованного строками и колонками маленьких точек (пикселей).
Растровая подложка	Таблица, содержащая растровое изображение. Эта таблица не содержит колонок и строк списка и, следовательно, некоторые операторы MapBasic не применимы к растровым таблицам.
Рекурсия	Вызов функции или процедуры из неё же самой (обычно с другими значениями входных параметров). Несмотря на то, что рекурсия может быть полезной в некоторых случаях, программисты должны учитывать, что рекурсия может возникать непреднамеренно, особенно, в процедурах-обработчиках событий таких, как <b>Self-ChangedHandler</b> .
Сборщик (Linker, компоновщик)	Программа, объединяющая отдельные модули файла проекта в единый файл MBX.
Свойства, свойства OLE	Часть OLE Automation. Свойство – это поименованный атрибут OLE-объекта. Чтобы определить статус объекта, необходимо прочитать его свойства. Если свойство доступно не только для чтения, можно изменить статус объекта, присвоив свойству новое значение. См. <b>Интегрированная картография</b> .
Сервер	Программный компонент вычислительной системы, выполняющий сервисные функции по запросу клиента, предоставляя ему доступ к определённым ресурсам. Часто используется в отношении соединений с базами данных или по DDE.

Термин	Определение
Связанная таблица	Тип таблицы MapInfo, загружаемой из удалённой базы данных. Данные, взятые из удалённой базы данных, сохраняются локально. Следующий раз при открытии этой связанной таблицы, MapInfo Professional проверит время создания таблиц, чтобы увидеть есть ли разница между этими таблицами. Если различие обнаружено, таблица обновится новыми значениями.
Система координат	Набор параметров, однозначно определяющий способ обработки координат объектов. Системы координат могут быть проективными (например, координаты в градусах долготы/широты) или плоскими (например, координаты план-схемы, выраженные в метрах); проективные координаты определяют положение объекта относительно всей поверхности Земли.
Служебные файлы и каталоги	Текстовые файлы, содержащие переменные конфигурации, названия шаблонов, каталогов символов пользователя и прочие основные режимы настройки Карт. Эти файлы обычно располагаются на машине пользователя, но могут быть также доступны пользователям группы, работающей над общим картографическим проектом.
Собственный формат	Стандартный (родной) формат файла. Команда MapInfo Professional <b>Файл &gt; Новая таблица</b> создаёт собственную таблицу MapInfo; таблица на основе тестового файла или файла электронной таблицы не является файлом собственного формата MapInfo.
Стандартные команды и панели	Стандартные команды меню и стандартные Панели Инструментов являются частью используемого по умолчанию интерфейса MapInfo Professional (например, <b>Файл &gt; Новая таблица</b> – это стандартная команда меню). Стандартные диалоги и диалоги с заранее определенным набором элементов управления (например, после выполнения оператора <b>Note</b> появится стандартный диалог со статическим текстовым элементом и кнопкой <b>ОК</b> ). Если программа MapBasic создает собственный элемент управления (диалог, кнопка на Панели Инструментов и т.п.), то такие элементы будут называться пользовательским диалогом, пользовательской кнопкой и т.п.
Стиль линии	Набор параметров, определяющих стиль линии для объекта. Стиль включает в себя следующие параметры: ширина, собственно стиль (вид линии) и цвет.
Стиль штриховки	Стиль штриховки объекта. Стиль состоит из штрихов определенной формы и цвета и цвета фона.

Термин	Определение
Столбчатая картодиаграмма	Тип графика или диаграммы, построенного по данным из таблицы. Столбчатая диаграмма может быть построена в окне графика или на тематической Карте.
Строка	Термин, аналогичный термину “запись”.
Строка сообщений	Полоска вдоль нижней части окна программы MapInfo Professional, в которой отображаются подсказки, имя редактируемого слоя и т.п.
Сшитые таблицы	Тип таблицы, в которой другие таблицы объединены в группу, что позволяет более простым способом открывать и показывать на Карте одновременно несколько таблиц. См. <a href="#">Работа с таблицами</a> .
Точка прерывания	Средство отладки программ. Чтобы заставить программу остановиться на заданной строке, задайте точку прерывания перед этой строкой. В MapBasic-программе точку прерывания можно задать, если вставить оператор <b>Stop</b> и перекомпилировать программу.
Файловый ввод/вывод	Процесс чтения информации из файла или записи информации в файл. Обратите внимание, язык MapBasic имеет один набор операторов для файловых операций ввода/вывода и один набор операторов для манипуляций с таблицами.
Фокус	Про активный элемент диалога (элемент который в данный момент используется пользователем) говорят, что он имеет фокус; нажатие клавиши Tab перемещает фокус от одного элемента к другому. Про фокус также говорят имея ввиду активное приложение. Переключение с одного приложения на другое (например, при нажатии комбинации клавиш Alt-Tab) приводит к смене фокуса с одного приложения на другое.
Функции обобщения	Функции, такие как <b>Sum( )</b> и <b>Count( )</b> , с помощью которых можно вычислить обобщенные по группам строк таблицы данные. См. оператор <b>Select</b> в <i>Справочнике MapBasic</i> или интерактивной справке.
Шестнадцатеричная	Шестнадцатеричная система исчисления, часто используется в программировании. Символами в шестнадцатеричном числе могут быть 0-9 и A-F. В MapBasic необходимо начинать каждое шестнадцатеричное число с префикса &H (например, &H1A – шестнадцатеричное число эквивалентное десятичному 26).
Широта	Тип координат исчисляемый в градусах указывающий положение к северу-югу от экватора. Размещение южнее экватора будет иметь отрицательные значения широты.

Термин	Определение
Цикл	Управляющая структура программы, которая позволяет повторять набор операторов, помещённых в тело цикла. Некорректный код программы с использованием цикла может привести к бесконечному циклу (ситуация когда цикл никогда не закончится).
Элемент управления	Компонент диалога, например, кнопка или флажок.
Явные значения	Выражение определяющее заданное, явное значение. Например, 23.45 это явно заданное число, а "Hello, World" – явно заданная строка. Также, иногда, называется значением "прописанным в коде".

# Предметный указатель

---

## Символы

### & (амперсанд)

- конкатенация строк **60, 293**
- пересечение улиц **138**
- сочетания клавиш в диалогах **110**
- сочетания клавиш в меню **97**
- шестнадцатичные числа **56**

### \* (звездочка)

- в строках фиксированной длины **50**
- умножение **59, 293**

### + (плюс) **293**

- конкатенация строк **60**
- сложение **60**
- сложение дат **61**

### , (запятая) символ

- разделитель тысяч **56**

### . (точка) символ

- разделитель целой части десятичной дроби **56**

### .Net-программирование

- введение **265**
- вызов метода по псевдониму **269**
- начала **265**
- обработка ошибок **274**
- объявление и вызов метода из MapBasic **267**
- ограничения на передачу структур **274**
- передача аргументов **269**
- передача структур **270**
- скорость выполнения **270**
- создание и копирование файла сборки **267**
- создание класса **266**
- терминология **265**

### / (косая)

- деление **60, 293**
- формат даты **57**

### < (меньше) **61**

### < > (не равно) **61**

### <= (меньше или равно) **61**

### = (знак равенства) **61**

### > (больше) **61**

### >= (больше или равно) **61**

### \ (обратная косая) **60, 293**

### ^ (каретка) **60**

- возведение в степень **293**

### – (минус) **293**

- вычитание **60**
- вычитание даты **61**

### ' (апостроф) **47**

## A-Z

### Add Column, оператор **198**

### Add Map Layer, оператор **112**

### Alter Button, оператор **119**

### Alter ButtonPad, оператор **119, 207**

### Alter Control, оператор **108**

### Alter Menu Bar, оператор **92**

### Alter Menu Item, оператор **93**

### Alter Menu, оператор **91**

### Alter Object, оператор **181, 185**

### Alter Table, оператор **140**

### And, оператор **63**

### Any(), оператор **197**

### Area(), функция **173, 195**

### Ask(), функция **100**

### AUTO\_LIB.MB (пример программы) **129**

### AutoLabel, оператор **180**

### Automation **315**

- объектная модель **236**

### Between, оператор **61**

### BIL-файлы (SPOT image) **150**

### Brush, стиль заливки **173**

### BrushPicker, элемент управления **105**

### Button, элемент управления **106**

### CancelButton, элемент управления **106**

### CheckBox, элемент управления **106**

### Close Window, оператор **112, 214**

### CommandInfo(), функция **96, 102, 108, 120** DDE **213**

### Commit Table, оператор **113**

**Commit**, оператор 116, 139  
**Contains**, оператор 64, 195  
**Continue**, оператор 82  
**Create ButtonPad**, оператор 119, 122, 207  
**Create Frame**, оператор 116, 180  
**Create Index**, оператор 140  
**Create Map**, оператор 140, 171  
**Create Menu Bar**, оператор 95  
**Create Menu**, оператор 92  
**Create Table**, оператор 301  
**Create Text**, оператор 116, 175  
**CreateCircle()**, функция 181  
**Crystal Report**, отчеты 133  
**CurDate()**, функция 61  
**CurDate()**, функция 55  
**Date**  
    операторы 61  
**DBF-файлы** 133  
**DDE**  
    функционирование в качестве клиента 208, 213  
    функционирование в качестве сервера 213  
**Declare Function**, оператор 77, 200  
**Declare Sub**, оператор 69, 200  
**Define**, оператор 78  
**Delphi**, примеры программ 263  
**Dim**, оператор 49  
**Do Case**, оператор 66  
**Do...Loop**, оператор 68  
**Drop Map**, оператор 71  
**EditText**, элемент управления 104  
**End Program**, оператор 69  
**EndHandler**, процедура 74  
**EOF()**, функция (конец файла) 166  
**EOT()**, функция 134  
**Err()**, функция 84  
**Error\$()**, функция 84  
**ERRORS.DOC** 226  
**Excel-файлы** 133  
**Fetch**, оператор 134, 185  
**FileExists()**, функция 165  
**FileOpenDlg()**, функция 101  
**FileSaveAsDlg()**, функция 101  
**FontPicker**, элемент управления 105  
**For...Next**, оператор 67  
**ForegroundTaskSwitchHandler**, процедура 74  
**Format\$()**, функция 115  
**FoxBase-файлы** 133  
**Frame**, объект 180  
**FrontWindow()**, функция 111  
**Function...End Function**, оператор 77  
**Get**, оператор (файловый ввод/вывод) 167  
**GetMetaData\$()**, функция 153  
**GetSeamlessSheet()**, функция 156  
**GIF-файлы** 150  
**Global Assembly Cache (GAC)** 275  
**GoTo**, оператор 67  
**GPS** 316  
**GPS**, приложения 113  
**GroupBox**, элемент управления 105  
**If...Then**, оператор 65  
**Include**, оператор 79  
**Input #**, оператор 166  
**Insert**, оператор 116, 139, 182  
**IntersectNodes()**, функция 187  
**Intersects**, оператор 64, 195  
**JPG-файлы** 150  
**Kernel (Windows DLL)** 203  
**Kill**, оператор 165  
**LabelFindByID()**, функция 188  
**LabelFindFirst()**, функция 188  
**LabelFindNext()**, функция 188  
**Labelinfo()**, функция 188  
**Like**, оператор 60  
**Line Input #**, оператор 166  
**ListBox**, элемент управления 105, 109  
**Lotus-файлы** 133  
**Main**, процедура 69  
**MakePen()**, функция 177  
**MapBasic**, введение 23  
**MapInfo Runtime**  
    запуск через OLE 220  
**MapInfo-L**, архив 21  
**MapInfo\_Mapcatalog**, создание 301  
**MAPINFOW.MNU** 98  
**MBX-файл** 315  
**MFC**  
    начало работы 255  
    примеры программ 263  
**Microsoft Excel** 133  
    взаимодействие по протоколу DDE 208  
**Mod** (целочисленная математика) 60  
**Mod**, оператор 293  
**MultiListBox**, элемент управления 105, 109  
**NoSelect**, ключевое слово 76  
**Not**, оператор 63  
**Note**, оператор 100  
**NumberToDate()**, функция 57  
**Object**, переменные 170  
**ObjectGeography()**, функция 173  
**ObjectInfo()**, функция 172, 177–178  
**ObjectLen()**, функция 173, 195  
**ODBC**, поддерживаемые типы данных 297  
**OKButton**, элемент управления 106  
**OLE Automation** 236, 315  
**OLE**, внедрение 218  
**OnError**, оператор 84

Open File, оператор 164  
Open Window, оператор 111, 214  
Or, оператор 63  
Pack Table, оператор 137  
PCX-файлы 150  
Pen, стиль линии 173  
PenPicker, элемент управления 105  
Perimeter(), функция 173  
PowerBuilder, примеры программ 263  
Print #, оператор 166  
Print, оператор 116  
ProgressBar, оператор 102  
PushButtons 119  
Put, оператор (файловый ввод/вывод) 167  
RadioGroup, элемент управления 105  
ReadControlValue(), функция 107, 109  
ReDim, оператор 51  
RemoteMsgHandler, процедура DDE 213  
RemoteQueryHandler(), функция 212  
Remove Map Layer, оператор 112  
Rename File, оператор 165  
Resume, оператор 84  
RGB, значение цвета 178  
RollBack, оператор 139  
RowID 137  
RTrim(), функция 62  
Run Application, оператор 128  
Run Menu Command, оператор 97, 116  
Save File, оператор 165  
Seagate Crystal Report, отчеты 133  
SearchInfo(), функция 124  
SelChangedHandler, процедура 74, 124  
Select Case (Do Case) 66  
Select, оператор 171–172, 179, 194, 196  
Selection  
    запрос 144  
    изменение таблицы 143  
SelectionInfo(), функция 142  
Set CoordSys, оператор 145, 191  
Set Event Processing, оператор 114  
Set File Timeout, оператор 147  
Set Format, оператор 57  
Set Map, оператор 112, 114, 180  
Set Redistricter, оператор 116  
Set Shade, оператор 113  
Set Table, оператор 156  
Set Target, оператор 186  
Set Window, оператор 112, 214  
Shade, оператор 113  
SPOT-файлы 150  
SQL-запросы 139  
StaticText, элемент управления 104  
Stop, оператор 82

StringCompare(), функция 62  
StyleAttr(), функция 174, 178  
SymbolPicker, элемент управления 105  
TableInfo() function 171  
TableInfo(), функция 137, 156, 171  
Targa-файлы 150  
TempFileName\$(), функция 165  
TIFF-файлы 150  
ToggleButtons, определение 119  
ToolButtons, определение 119  
ToolHandler, процедура 74, 120  
TriggerControl(), функция 108  
Type...End Type, оператор 52  
UBound(), функция 51  
Update, оператор 139, 181–182, 185  
User (библиотека Windows) 201  
Visual Basic, примеры программ 218, 263  
Visual C++  
    начало работы 255  
    примеры программ 263  
While...Wend, оператор 69  
WIN.INI, настройки запроса 203  
WinChangedHandler, процедура 74  
WinClosedHandler, процедура 74  
WindowID(), функция 112  
WindowInfo(), функция 112, 145  
WinFocusChangedHandler, процедура 74  
Within, оператор 64, 195  
WKS-файлы, открытие 133  
Write #, оператор 166  
XLS-файлы, открытие 133

## A

Адрес, поиск 138  
Анимационный слой 113  
Аргументы  
    передача в среду .Net 269  
    передача по значению 72  
    передача по ссылке 71  
Аргументы командной строки 35, 253  
Арифметические операторы 59

## Б

Бесконечные циклы, предотвращение 76  
Библиотеки DLL  
    библиотека Kernel 203  
    библиотека пользователя 201  
    декларирование 200  
    определение 200  
    передача параметров 201  
    путь поиска 200  
    сохранение пиктограмм для панели

инструментов **206**  
строковые параметры **202**  
**Бинарные файлы** **164, 167**  
**Битовые изображения** **150**  
**"Бумажные" единицы измерения** **193**  
**Буферные зоны, создание** **183, 288**  
**Быстрое меню**  
    изменение **96**  
    отключение **96**  
    удаление **96**

**В**

**Ввод/вывод в файл, См. Файловый ввод/вывод**  
**Вершины, См. Узлы**  
**Ветвление** **67**  
**Включенный режим** **321**  
**Внедрение** **218**  
**Внешние ссылки**  
    Windows DLLs **200**  
    подпрограммы в других модулях **40**  
**Вращение графического объекта** **288**  
**Всплывающее меню (PopupMenu)** **106, 109**  
**Всплывающие подсказки** **126**  
**Вставка**  
    колонок в таблицу **140**  
    строк в таблицу **139**  
    узлов в объекте **181**  
**Выбор команды меню, имитация** **97**  
**Выбор щелчком на объекте** **124**  
**Вызов внешних подпрограмм** **40, 201**  
**Вызов метода из MapBasic** **267**  
**Вызов процедур** **70**  
**Выполнение программы, ошибки** **82**  
**Выполнение процедур-обработчиков** **76**  
**Высота текста** **174**

**Г**

**Генератор отчетов** **133**  
**Географические объекты, См. Объекты**  
**Географические операторы** **64, 193**  
**Геокодирование**  
    автоматические **138**  
    интерактивное **139**  
**Глобальные переменные** **52**  
**Градусная секта, программа** **287**  
**Градусы** **316**  
**Градусы в ГМС, преобразование** **286**

**Д**

**Данные, структура** **52**  
**Даты, константы** **57**  
**Действия с мышью** **32**

**Десятичные разделители в числовых константах** **56**  
**Диалог "Retry/Cancel"** **146**  
**Диалог "Открыть сразу"** **128**  
**Диалоги, созданные пользователем**  
    выбор строчки из списка **109**  
    закрытие **110**  
    модальные, немодальные **110**  
    начальное значение элемента **107**  
    недоступные элементы управления **108**  
    примеры **102–103**  
    размер элемента диалога **103**  
    реакция на действия пользователя **107**  
    сочетания клавиш **110**  
    списки **109**  
    считывание значений **107**  
    элементы управления **104**  
    элементы управления, размещение **103**  
**Диалоги, стандартные**  
    "OK/Cancel", запрос **100**  
    открытие файла **101**  
    пример сообщения **100**  
    процент выполнения **102**  
    скрытие индикатора выполнения **130**  
    сохранение файла **101**  
**Длина объекта** **195**  
**Добавление колонок к таблице** **140**  
**Добавление узлов** **181**  
**Долгота** **316**  
**Доступ к удаленным базам данных** **156**

**Е**

**Единицы измерения**  
    листа **193**  
    листа ("бумажные") **193**  
    площади **193**  
    расстояния **193**

**З**

**Закрепление Инструментальной панели** **126**  
**Закрытие**  
    таблицы QueryN **142**  
**Записи, См. Строки**  
**Запрос значения для переменной** **49**  
**Запуск программы**  
    из MapInfo **25, 35**  
    из среды разработки **44**  
    из стартового Рабочего набора **128**  
**Звуковой сигнал о переполнении окна** **32**  
**Значение, передача параметров** **72**  
**Значения цвета**  
    RGB( ), функция **178**



выбор объектов по цвету 178

## И

Идентификатор выбранной команды меню 96

Идентификатор окна 111

Идентификаторы, определение 78

Изменяемый объект 186

Имя каталога 165

Имя класса

MapInfo.Application 219

MapInfo.Runtime 220

Индексы, создание 140–141

Индикатор процесса выполнения 317

диалог 102

скрытие 130

Инструментальные панели 320

ICONDEMO.MBX, программа 123

добавление новых кнопок 121

закрепление 126

кнопки PushButtons 119

кнопки ToggleButtons 119

кнопки ToolButtons 119

пиктограммы пользователя 206

создание кнопки 121

создание новой панели 120

сообщения-подсказки для кнопок 126

Интегрированная картография 317

MFC 255

введение 216

выход из MapInfo 227

запуск MapInfo 219

изменение размеров окна 223

кнопки обратных вызовов (callback) 228

кнопки панели инструментов 223

объектная модель 236

переподчинение окна документа 221

переподчинение окна Легенды 222

перехват ошибок 226

печать 226

примеры программ 218, 263

системные требования 218

электронная Справочная система 233

Интерфейс пользователя

диалоги пользователя 102

диалоги, стандартные 100

Инструментальные панели 118

курсоры 127

меню 90

обзор 88

окна 111

Информация, окно

настройка 117

параметр только для чтения 117

Исполняемый Runtime-модуль

запуск через OLE 220

Исходный код 317

## К

Карта градуированных символов 113

Картографические объекты, См. Объекты

Карты, См. Слои

Километры 191

Клавишные комбинации 30

Класс, создание в среде .Net 266

Клиент/сервер

доступ к базам данных 156

протокол DDE 208

Количество

выбранных строк 142

объектов в строке 172

открытых окон 112

полигонов в регионе 172

сегментов в полилинии 172

узлов в объекте 181

Колонки

Obj (object) 138, 170

RowID 137

псевдонимы 135

синтаксис 135

Комбинации клавиш в диалогах 110

Комментарии 47

Компилятор 318

директивы 78

Компиляция программы

без открытия файла 41

в активном окне 25, 33

из командной строки 35

Комплект документации MapInfo и MapBasic 17

Конкатенация строк

оператор & 293

оператор + 293

Константы

арифметические 56

даты 57

логические 57

определение 54

строковые 56

Конфликты совместного доступа 146

Координатные системы

в окне Отчета 145

географические координаты 191

координаты Отчёта 191

координаты план-схемы 191

Косметический слой 318

выбор объектов **145**  
удаление объектов **145**  
**Круговые диаграммы**  
на Графиках **115**  
на тематических картах **113**  
**Курсор (в таблице)** **134**  
установка **134**  
**Курсор (пиктограмма инструмента Рисование)"**  
**127**  
**Курсор, изменить стиль** **127**

## Л

**Легенда, управление окном** **287**  
**Линейки прокрутки, показать или скрыть** **113**  
**Линии, См. Объекты**  
**Литерал** **324**  
**Логические операторы** **63**  
**Локальные переменные** **49**

## М

**Маска (сравнение строк)** **60**  
**Массив переменных**  
изменение размера **51**  
объявление **51**  
**Математика, целочисленная** **59**  
**Меню**  
Окно **44**  
Поиск **42**  
Правка **42**  
Сборка **43**  
Справка **44**  
Файл **41**  
**Меню, настройка**  
MAPINFOW.MNU **98**  
добавление элементов меню **91**  
изменение элемента меню **93**  
переопределение строки меню **94**  
создание нового меню **92**  
сочетания клавиш **97**  
удаление элементов меню **92**  
**Метаданные** **152**  
**Методы** **318**  
вызов по псевдониму **269**  
объект Application **239**  
объект MApplication **245**  
объект MIMapGen **249**  
объявление и вызов из MapBasic **267**  
**Метрические единицы** **191**  
**Модальный диалог** **110**  
**Модули** **318**  
вызов функций/процедуры из других **40**  
глобальные (общие) переменные **40**

локальные переменные **40**

## Н

**Найти и заменить**  
в редакторе MapBasic **42**  
программа **290**  
**Наборы символов, национальные** **168**  
**Наложение полигонов** **198**  
**Настройка окна**  
График **115**  
Информация **117**  
Карта **112**  
Отчёт **116**  
Районирование **116**  
размер и положение **112**  
Сообщение **116**  
Список **114**  
**Нежесткие связи** **213**  
**Номер строки в программе** **43**

## О

**Области, См. Объекты**  
**Область определения переменных** **53**  
**Область определения функций** **78**  
**Обновление в удаленных базах данных** **159**  
**Обобщение данных, функции, См. Справочник**  
*MapBasic*  
**Обработка событий, См. События, обработка**  
**Обратные вызовы (callbacks)** **228**  
**Обучение пользователей** **21**  
**Объединение таблиц** **197**  
**Объект, удаление части** **186**  
**Объектная модель** **236**  
**Объекты, запросы**  
координаты **172**  
стили **173**  
типы **172**  
**Объекты, редактирование**  
добавление узлов **181, 187**  
объединение **183**  
положение **185**  
сохранение в таблице **182**  
стиль **185**  
тип объекта **186**  
удаление части объекта **186**  
**Объекты, создание**  
буферные зоны **183**  
на основе существующих объектов **182**  
Операторы создания объектов **180**  
сохранение в таблице **182**  
Функции создания объектов **180**  
**Объекты, стиль** **173**

**Объекты, удаление** 171  
**Объявление метода в MapBasic** 267  
**Ограничение размера текста** 32  
**Ограничения на память** 32  
**Окна Отчета**

табличный подход 145

**Окно**

MapBasic 48

Графика 115

Карты 112

подписывание 187

Отчёта

координаты объекта 192

открытие 116

Районирование 116

Сообщений 116

Списка 114

**Окно, меню** 44

**Окружности, См. Объекты**

**Оператор, дескриптор** 157

**Оператор, номер** 157

**Операторы** 319

Add Column 198

Add Map Layer 112

Alter Button 119

Alter ButtonPad 119, 207

Alter Control 108

Alter Menu Bar 92

Alter Menu Item 93

Alter Object 181, 185

Alter Table 140

And 63

AutoLabel 180

Between 61

Close Window 112, 214

Commit 116, 139

Contains 64, 195

Continue 82

Create ButtonPad 119, 122, 207

Create Frame 116, 180

Create Index 140

Create Map 140, 171

Create Menu 92

Create Menu Bar 95

Create Text 116, 175

Declare Function 77, 200

Declare Sub 69, 200

Define 78

Dim 49

Do Case 66

Do...Loop 68

Drop Map 171

End Program 69

Fetch 134, 185

For...Next 67

Function...End Function 77

GoTo 67

If...Then 65

Include 79

Input # 166

Insert 116, 139, 182

Kill 165

Line Input # 166

Not 63

Note 100

OnError 84

Open File 164

Open Window 111, 214

Or 63

Pack Table 137

Print 116

Print # 166

ProgressBar 102

ReDim 51

Remove Map Layer 112

Rename File 165

Resume 84

RollBack 139

Run Application 128

Run Menu Command 97, 116

Save File 165

Select 171–172, 179, 194, 196

Set CoordSys 145, 191

Set Event Processing 114

Set File Timeout 147

Set Format 57

Set Map 112, 114, 180

Set Redistricter 116

Set Shade 113

Set Table 156

Set Target 186

Set Window 112, 214

Shade 113

Stop 82

Type...End Type 52

Update 139, 181, 185

While...Wend 69

Write # 166

арифметические 59

географические 64, 193

даты 61

логические 63

определение 54

порядок выполнения 64

порядок применения 64

сравнения 61

- строковые **60**
- числовые **59**
- Определение двойного щелчка в списке 108**
- Определение нажатия пользователем кнопки ОК в диалоге 102**
- Определенная пользователем функция 77**
- Оптимизация производительности**
  - интерфейс пользователя **129**
  - процедуры-обработчики **76**
  - управление таблицами **160**
- Организация программ 79**
- Остановка программы 69**
- Отключенный режим 319**
- Открытие нескольких файлов 39**
- Открытие таблиц 132**
- Отладка программ 82**
- Ошибки 139**
  - времени исполнения **82**
  - времени компиляции **34**
  - перехват **84**
  - поиск **84**
- П**
- Панели инструментов, См. Инструментальные панели**
- Панель управления, эффект формата даты 58**
- Параметры**
  - передача по значению **72**
  - передача по ссылке **71**
  - страницы **116**
- Передача структур**
  - в среду .Net **270**
- Передача структур в среду .Net, ограничения 274**
- Переменные**
  - глобальные **52**
  - имена, ограничения **49**
  - область определения **53**
  - объявление **49**
  - определение **48**
  - стилей **177**
  - строковые **50**
  - типа Object **170**
  - типы данных **50**
  - чтение глобальных переменных другого приложения **212**
- Перемещение объекта 185**
- Пересечение**
  - область перекрытия объектов **183**
  - оператор Intersects **64**
  - точка пересечения линий **187**
  - улиц **138**
- Пиктограммы для кнопок 205**
- Поддержка пользователей**
  - web-сайты **22**
- Подзапросы 196**
- Подписывание**
  - в программе **67**
  - на Карте **180, 187**
  - преобразование в текст **190**
- Подпрограммы, См. Процедуры**
- Подсказки в строке сообщений 126**
  - в интегрированной картографии **228**
- Подсказки для кнопок 126**
- Подтверждение ввода 100**
- Поиск и замена, программа 290**
- Поиск пути к DLL 200**
- Поиск уличного адреса 138**
- Поиск, меню 42**
- Полилинии, См. Объекты**
- Порядок применения операторов 295**
- Правка, меню 42**
- Преобразование градусов, программа 286**
- Преобразование типов 59**
- Привязка к узлу 228**
- Приложения реального времени 113**
- Примеры программ**
  - интегрированная картография **263**
  - на языке C **263**
- Принятие решений**
  - Do Case, оператор **66**
  - If...Then, оператор **65**
- Принятые обозначения 18**
- Приоритет операторов 64**
- Программирование в среде .Net, введение 265**
- Проекция, изменение 113**
- Проекция Карты 113**
- Прозрачная заливка 320**
- Производительность**
  - интерфейс пользователя **129**
  - процедуры-обработчики **76**
  - управление таблицами **160**
- Промежуточные итоги 139**
- Пропорциональное обобщение данных 198**
- Процедуры**
  - вызов **70**
  - главная процедура (Main) **69**
  - обработчики событий **73**
  - определение **69**
  - передача параметров **71**
  - рекурсия **73**
- Прямой доступ к базам данных 160**
- Прямой доступ к удаленным базам данных 160**
- Прямое перемещение объектов мышкой 122**
- Псевдонимы 321**
  - использование при вызове метода **269**

колонок **135**  
переменных **135**

## **Р**

### **Рабочие наборы**

STARTUP **128**

как примеры программ **27**

### **Разделители в числовых константах** **56**

### **Размер шрифта** **174**

### **Раскодирование (отмена геокодирования)** **171**

### **Расстояния**

единицы изменения **193**

### **Растровая подложка** **321**

### **Растры** **150**

регистрация в таблицах **150**

### **Расширения имен файлов** **16**

### **Редактирование в многопользовательской среде** **146**

### **Режим рисования** **122**

### **Рекурсия** **73, 321**

### **Реляционная связь** **172, 197**

## **С**

### **Сборка проекта**

без открытия файла **41**

из командной строки **35**

после выбора текущего проекта **38**

### **Сборка, меню** **43**

### **Сборщик программ** **321**

### **Связанные таблицы** **159**

### **Свойства** **321**

Application, объект **238**

MVApplication, объект **245**

MVApplications, коллекция **244**

MBGlobal, объект **246**

MBGlobals, коллекция **246**

MIMapGen, объект **247**

### **Связанные таблицы** **322**

### **Скорость выполнения** **129**

оптимизация управления таблицами **160**

процедуры-обработчики **76**

### **Слияние объектов** **183**

### **Слои**

добавление/удаление **112**

Косметический слой **144**

тематические **113**

### **События**

обработка

изменение выборки **124**

изменение интерфейса пользователя **88**

определение **73**

специальные процедуры **74**

производимые мышкой

выбор команды меню **90**

двойной щелчок в списке **108**

захватить и перетащить **122**

### **Соединение с удаленной базой данных** **156**

### **Соединение, номер** **157**

### **Соединение. дескриптор** **157**

### **Создание каталога MapInfo\_Mapcatalog** **301**

### **Создание класса в среде .Net** **266**

### **Создание объектов Карты** **180**

### **Сортировка** **139**

### **Сохранение точек в удаленной базе данных** **159, 299**

### **Сочетания клавиш**

в диалогах **110**

в интегрированных Картах **228**

в меню **97**

### **Справка, меню** **44**

### **Справочная система**

использование **27**

копирование фрагментов программ **27**

создание **214**

### **Сравнение по шаблону** **60**

### **Сравнение, операторы** **61**

### **Ссылка, передача параметров** **71**

### **Ссылки** **213**

### **Стартовый Рабочий набор** **128**

### **Стили**

линии **173**

объектов (Pen, Brush, Symbol, Font) **173**

символов **173**

сравнение **174**

текста (Font) **173**

точечных объектов (Symbol) **173**

шрифта **173–174**

### **Столбчатые диаграммы**

на Графиках **115**

на тематических Картах **113**

### **Строки в таблице**

в окне Информация **117**

вставка новых **139**

номера строк (RowID) **137**

обновление **139**

сортировка **139**

текущая **134**

### **Строки фиксированной и произвольной длины** **50**

### **Строки, конкатенация**

оператор & **293**

оператор + **293**

### **Строковые константы** **56**

### **Строковые операторы** **60**

### **Строковые переменные**

произвольной длины **50**

Строковые переменные фиксированной длины **50**  
Структура данных **52**  
Структуры, передача в среду .Net **270**  
Суммирование **139**  
Сшитые таблицы **155**

## Т

### Таблицы

Query/V  
    заккрытие **142**  
    открытие **142**  
Selection **142**  
    внесение данных **139**  
    выражения для колонок **135**  
    для растров **150**  
    добавление временных колонок **141**  
    добавление динамических колонок **141**  
    добавление постонных колонок **140**  
    картографируемые **140**  
    количество открытых **141**  
    колонок Obj **170**  
    колонок Obj (object) **138**  
    косметические **144**  
    метаданные **152**  
    на основе электронных таблиц и баз данных **133**  
    номера строк **137**  
    объединение **197**  
    открытие **132**  
    Отчет **145**  
    создание **140**  
    структура, запрос **141**  
    структура, изменение **140**  
    файлы-компоненты **150**  
    чтение значений **134**  
Таблицы, в удаленных базах данных **301**  
    каталог mapinfo\_mapcatalog **301**  
Текстовый редактор **35**  
Текстовые объекты **173, 185**  
Тексты, См. Текстовые объекты  
Тематическая Карта **113**  
Тематическая растровая поверхность, поддержка **113**  
Техническая поддержка  
    службы **19–21**  
Типографские обозначения **18**  
Типы данных, определенные пользователем **52**  
Точки останова (отладка) **83**  
Точки пересечения **187**  
Точки, См. Объекты  
Точки, сохранение в удаленной базе данных **159**

## У

### Удаление

    индексов **141**  
    колонок из таблицы **140**  
    команд меню **92, 98**  
    меню **94, 98**  
    файлов **165**  
    части объекта **186**  
Удаленные базы данных  
    каталог mapinfo\_mapcatalog **301**  
Удаленные базы данных, прямой доступ **160**  
Удаленные данные **316**  
Узлы  
    добавление **181, 187, 289**  
    максимальное число **181**  
    определение координат **187**  
Уличный адрес, поиск **138**  
Улучшения интерфейса пользователя **129**  
Установка программы **15**

## Ф

Файл меню MapInfo **98**  
Файл сборки, создание и копирование **267**  
Файл, меню **41**  
Файл, удаление **165**  
Файловый ввод/вывод  
    бинарные файлы **167**  
    копирование файла **165**  
    национальные наборы символов **168**  
    определение **164**  
    переименование файла **165**  
    удаление файла **165**  
    файлы последовательного доступа **165**  
    файлы произвольного доступа **167**  
Файлы DBF **133**  
Файлы Excel **133**  
Файлы FoxBase **133**  
Файлы Lotus **133**  
Файлы внешние  
    DBF (dBASE) **133**  
    WKS (Lotus) **133**  
    XLS (Excel) **133**  
Файлы заголовков (header) **16**  
Файлы поверхности, поддержка **113**  
Файлы последовательного доступа **164–165**  
Файлы проекта  
    определение **36**  
    преимущество **37**  
    примеры **37**  
    сборка **38**  
    создание **38**  
Файлы произвольного доступа **164, 167**

---

## Файлы Справочной системы

использование **27**

создание **214**

## Файлы, внешние

BIL (SPOT image) **150**

GIF **150**

JPG **150**

PCX **150**

Targa **150**

TIFF **150**

## Фокус **323**

в диалоге **108**

## Функции

Area( ) **173**

CommandInfo( ), разбор результатов поиска **138**

CreateCircle( ) **181**

CurDate( ) **61**

CurDate( ) **55**

EOF( ) (конец файла) **166**

EOT( ) (конец таблицы) **134**

Err( ) **84**

Error\$( ) **84**

FileExists( ) **165**

FileOpenDlg( ) **101**

FileSaveAsDlg( ) **101**

Format\$( ) **115**

FrontWindow( ) **111**

GetMetaData\$( ) **153**

GetSeamlessSheet( ) **156**

IntersectNodes( ) **187**

LabelFindByID( ) **188**

LabelFindFirst( ) **188**

LabelFindNext( ) **188**

LabelInfo( ) **188**

MakePen( ) **177**

NumberToDate( ) **57**

ObjectGeography( ) **173**

ObjectInfo( ) **172, 177–178**

ObjectLen( ) **173, 195**

Perimeter( ) **173**

ReadControlsValue( ) **107**

ReadControlValue( ) **109**

RemoteQueryHandler( ) **212**

RTrim\$( ) **62**

SearchInfo( ) **124**

SelectionInfo( ) **142**

StringCompare( ) **62**

StyleAttr( ) **174, 178**

TableInfo( ) **137, 156, 171**

TempFileName\$( ) **165**

TriggerControl( ) **108**

UBound( ) **51**

WindowID( ) **112**

WindowInfo( ) **112, 145**

область определения **78**

определенная пользователем **77**

## Функции обобщения **323**

## Ц

Целочисленное деление **293**

## Циклы

Do...Loop, оператор **68**

For...Next, оператор **67**

MapInfo DDE-сервер **213**

While...Wend, оператор **69**

## Ч

Числовые константы **56**

Числовые операторы **59**

Чтение переменных другого приложения **212**

Чувствительность к регистру **47**

## Ш

Шестнадцатеричные числа **323**

синтаксис **56**

Широта **323**

## Э

Электронные таблицы, открытие **133**

Элементы меню, помечаемые галочкой **93**

Элементы управления **104–106, 109**