

A decorative graphic on the left side of the page consists of several rounded squares in various colors: a large green square at the top right, a yellow square below it, an orange square below that, a blue square to the left of the orange one, a purple square below the blue one, and a small teal square to the right of the purple one.

MapBasic

Руководство разработчика

Information in this document is subject to change without notice and does not represent a commitment on the part of the vendor or its representatives. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, without the written permission of MapInfo Corporation, One Global View, Troy, New York 12180-8399.

© 2006 MapInfo Corporation. All rights reserved. MapInfo, the MapInfo logo, MapBasic, and MapInfo Professional are trademarks of MapInfo Corporation and/or its affiliates.

© 2006 ООО “ЭСТИ МАП” Русский перевод. Владимир Журавлёв, Алексей Колотов, Камиль Мусин, Всеволод Николаев, Ольга Зислис.

MapInfo Corporate Headquarters:

Voice: (518) 285-6000

Fax: (518) 285-6070

Sales Info Hotline: (800) 327-8627

Government Sales Hotline: (800) 619-2333

Technical Support Hotline: (518) 285-7283

Technical Support Fax: (518) 285-6080

Contact information for all MapInfo offices is located at: <http://www.mapinfo.com/contactus>.

Adobe Acrobat® is a registered trademark of Adobe Systems Incorporated in the United States.

Products named herein may be trademarks of their respective manufacturers and are hereby recognized. Trademarked names are used editorially, to the benefit of the trademark owner, with no intent to infringe on the trademark.

СОДЕРЖАНИЕ

Глава 1: Введение	5
Требования к системе и компьютеру	6
Установка среды разработчика MapBasic	6
Стандартные имена файлов и типы файлов MapBasic	7
Установка документации MapBasic	8
Соглашения об обозначениях	9
Глава 2: Обзор языка MapBasic	13
Главные особенности языка MapBasic	16
Как осваивать MapBasic?	17
Глава 3: Работа в интегрированной среде разработки программ	21
Введение в интегрированную среду MapBasic	22
Как ввести или отредактировать программу	22
Компиляция программ	25
Проект: Сборка приложения из нескольких модулей	28
Обзор меню среды разработки программ MapBasic	33
Глава 4: Основы языка MapBasic	39
Общие замечания о синтаксисе языка MapBasic	40
Переменные	42
Выражения	48
Циклы и другие управляющие операторы	59
Процедуры	65
Процедуры-обработчики системных событий	69
Рекомендации по использованию процедур-обработчиков системных событий	74
Функции, созданные пользователем	74
Директивы компилятора	76
Организация программ	78
Глава 5: Отладка и обработка ошибок	79
Обработка ошибок	83
Глава 6: Интерфейс пользователя	87
Программная обработка событий	89
Меню	91
Стандартные диалоговые окна	102

Новые диалоговые окна	105
Окна.	116
Инструментальные панели	125
Запуск программы в среде MapInfo.	135
Рекомендации по повышению производительности	137
Глава 7: Работа с таблицами	139
Открытие таблиц с помощью MapBasic.	140
Создание файла отчета из открытой таблицы MapInfo	142
Запись значений в таблицу	149
Создание новых таблиц	150
Изменение структуры таблицы.	150
Работа с таблицей Selection	153
Доступ к Косметическому слою с помощью таблицы “CosmeticN”	156
Доступ к окнам Отчетов с помощью таблиц “LayoutN”	156
Редактирование в многопользовательской среде	157
Файлы-компоненты таблицы	162
Таблицы, содержащие растровые изображения	163
Работа с метаданными	166
Работа со сшитыми таблицами	169
Доступ к удаленным данным.	171
Доступ и изменения в удаленных базах данных при помощи связанных таблиц	174
Прямой доступ к удаленным базам данных	174
Рекомендации по повышению производительности при работе с таблицами	175
Глава 8: Ввод/Вывод в файлы	177
Обзор файлового ввода/вывода	178
Файлы последовательного доступа	180
Файлы произвольного доступа	182
Двоичные файлы	182
Особенности работы с файлами в различных операционных системах и с национальными наборами символов	183
Функции, имеющие дело с режимами файлового ввода/вывода	185
Глава 9: Графические объекты	187
Переменные типа Object	188
Работа с колонкой “Obj”	189
Определение атрибутов объекта	191

Создание новых объектов	199
Создание новых объектов на основе уже существующих	202
Изменение объектов	204
Работа с подписями	206
Координаты и единицы измерения	211
Географические запросы	214
Глава 10: Особенности MapBasic в среде	
Microsoft Windows	221
Добавление Справочной системы к Вашему приложению	237
Глава 11: Интегрированная Картография	239
Как выглядит приложение с Интегрированной Картой?	240
Концепции Интегрированной Картографии	241
Технические аспекты Интегрированной Картографии	243
Простейший пример	244
Подробное обсуждение Интегрированной Картографии	245
Использование уведомляющих вызовов (Callbacks) для получения информации из MapInfo	255
Другие способы использования уведомлений	260
Справочная система	261
Полезные операторы и функции языка MapBasic	262
Объектная модель механизма управления объектами OLE.	264
Аргументы командной строки MapInfo	276
Введение в Интегрированную Картографию с поддержкой Visual C++ и MFC	277
Добавление кнопок и процедур-обработчиков	283
Где получить дополнительную информацию	288
Приложение А: Примеры программ	291
Приложение В: Действия операторов	295
Приложение С: Что нового в MapBasic 7.0	299
Приложение D: Поддерживаемые типы таблиц ODBC	301
Приложение Е: Присвоение геоинформации удаленной таблице	303
Создание каталога Карт MapInfo	304
Приложение F: Установка и управление данными	307

Введение

Добро пожаловать в среду разработчика MapBasic 7.0, мощный и одновременно простой в использовании язык программирования, который позволит Вам создавать собственные приложения в среде MapInfo Professional. В этой главе будет описана процедура установки MapBasic. Общая информация о возможностях и назначении языка MapBasic начинается с главы 2: "Обзор языка MapBasic".

1

Глава

- >Требования к системе
 - >Установка MapBasic
 - >Стандартные имена файлов и типы файлов MapBasic
 - >Соглашения об обозначениях
-

Требования к системе и компьютеру

Требования к системе

Чтобы использовать MapBasic для Windows, Ваш компьютер должен удовлетворять следующим требованиям::

Требования

Операционная система	Microsoft Windows XP/2000/98 или Windows NT 4.0/2000
Дисплей	Любой адаптер дисплея поддерживаемый Windows
Мышь	Любая мышь поддерживаемая Windows
Жесткий диск	10 Mb

Совместимость между версиями

Программы, созданные в ранних версиях MapBasic, будут работать под Mapinfo Professional.

Смотрите более подробную информацию:

- Приложение C: Новое в версии MapBasic 7.0
- Приложение F: Установка и управление данными

Установка среды разработчика MapBasic

Перед тем как начать

Процедура установки MapBasic описана ниже. Сделайте следующее:

- Установите MapInfo перед установкой MapBasic. Внимательно прочитайте в Руководстве пользователя MapInfo как это правильно сделать.
- Запишите серийный номер MapBasic в месте, где его легко обнаружить в случае необходимости.

Установка

1. Вставьте компакт диск с MapInfo Pro CD дисковод, выберите Установку MapBasic 7.0 и следуйте инструкциям в экранных диалогах.

По умолчанию MapBasic устанавливается в директорию под MapInfo (т.е. C:\MAPINFO\MAPBASIC\MAPBASIC.EXE).

Если компакт диск автоматически не запускается, то нажмите Setup.

Запуск MapBasic

Чтобы запустить среду разработки MapBasic,

1. Выполните команду Windows ПУСК>ПРОГРАММЫ
2. Чтобы запустить MapBasic, выберите MapBasic из программной группы MapInfo.

Можно проверить, нуждается ли MapBasic в обновлении, для этого выполните команду СПРАВКА > ПРОВЕРИТЬ ВЕРСИЮ.

Стандартные имена файлов и типы файлов MapBasic

Процедура установки языка MapBasic переносит на Ваш компьютер следующие стандартные файлы:

errors.doc:	Текстовый файл со списком сообщений об ошибках MapBasic
mapbasic.exe:	Исполняемый файл, позволяющий Вам работать в среде MapBasic
mapbasic.def:	Файл заголовков, содержащий объявления стандартных переменных
menu.def:	Файл заголовков, содержащий объявления стандартных переменных, относящихся к системе меню MapInfo
icons.def:	Файл заголовков, содержащий объявления стандартных переменных, относящихся к виду кнопок и курсоров в MapInfo
mapbasic.hlp:	Справочный файл по языку MapBasic
mapbasic.h:	Файл заголовков для программистов на языках C/C++; аналогичен MAPBASIC.DEF
mapbasic.bas:	Файл заголовков для программистов на языке Visual Basic; аналогичен MAPBASIC.DEF
mapbasic65.isu	Файл для удаления MapBasic.
mbres650.dll	Часть программной среды; содержит ресурсы - диалоги и строки.
milib650.dll	Часть программной среды; содержит выполняемый код XVT

errors.doc:	Текстовый файл со списком сообщений об ошибках MapBasic
papersize.def	Файл для разработчиков программ в MB; содержит определения для управления печатью операторами MapBasic
usrinfmb.log	Содержит информацию для процесса установки.
samples folder	<i>содержит примеры программ filename.mb, filename.mbp;</i>

При работе в среде MapBasic используются следующие стандартные расширения:

filename.mb	Программный файл (исходный текст на языке MapBasic)
filename.mbx	Откомпилированный (выполняемый) файл
filename.mbp	Файл описания модулей (содержит список всех модулей, собираемых в единую программу)
filename.mbo	Объектный файл (создается при сборке программы из нескольких модулей)
filename.err	Список сообщений об ошибках, полученный при компиляции программы, записанной на диске.

Установка документации MapBasic

В дополнение к Руководству пользователя, документация MapBasic включает в себя электронную версию этой книги, электронную версию Справочника и электронную Справочную систему.

Справочник MapBasic®

Электронный справочник MapBasic является полным описанием команд и функций языка MapBasic. Смотрите Главу 24, Использование окна MapBasic, в этой главе обсуждается, какие команды MapBasic могут использоваться в этом окне.

Установка электронной документации

Можно пользоваться доступом к электронным Справочнику MapBasic или Руководству пользователя с компакт диска MapBasic, или установить программу Adobe® Acrobat Reader для локального доступа к документации.

Для локальной установки документации на Ваш компьютер:

1. Установите Acrobat® Reader.
2. Скопируйте с компакт диска файлы из папки [CD_ROM]:\PDF_DOCS в папку на компьютере.

mb70ug.pdf это Руководство, занимающее около ~8 МВ дисковой памяти.

mb_ref.pdf это Справочник MapBasic, занимающий около ~10 МВ of дисковой памяти.

3. Из Проводника Windows, дважды щелкните на любом из этих файлов и автоматически запустится Acrobat® Reader с электронной документацией.

Соглашения об обозначениях

В данном руководстве используются следующие термины и условные обозначения.

Терминология

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 Вами, программистом. Как правило, файлы с программами на MapBasic имеют расширение .MB.

Приложение – это двоичный исполняемый файл (в среде MapInfo). Его создает MapBasic при компиляции программы, и его пользователь запускает на выполнение. 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.

Обозначения

Шрифтом Courier выделены примеры программ на языке MapBasic:

Note "Привет от MapBasic-a!"

Жирным шрифтом 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, Вы можете 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 капителью.

Зарегистрируйтесь!

Заполните регистрационную карточку и отошлите ее в местное отделение MapInfo Corporation, чтобы получать информацию о новых версиях.

Работа с технической поддержкой

Техническая поддержка предназначена для помощи Вам, и Ваше обращение за такой поддержкой крайне важно. Далее будет дана информация о том как пользоваться технической поддержкой при обращении в региональные центры. Так же будут объяснены некоторые процедуры оказания технической поддержки.

Перед тем как обратиться за технической поддержкой

Подготовьте следующую информацию перед тем как обратиться к нам за технической поддержкой.

1. Серийный номер. Вы должны иметь зарегистрированный серийный номер, чтобы получить техническую поддержку.
2. Ваше имя и организация. Тот кто обращается должен быть внесен в специальный список.
3. Версия продукта, о котором Вы запрашиваете поддержку.
4. Ваша операционная система и ее версия.
5. Краткое описание проблемы. Вот некоторые детали, которые могут быть полезны при этом:
 - Сообщения об ошибках
 - Обстоятельства при которых возникла данная проблема
 - Часто или нет возникает подобная проблема?

Ожидается время оказания технической поддержки

Большинство проблем может быть разрешено во время первого же телефонного звонка. Если сразу не получается, ответ будет дан до конца данного рабочего дня.

Если запрос о поддержке отправлен по электронной почте, то он обрабатывается так же как и телефонный запрос и в те же сроки.

Обмен информацией

Иногда для технической поддержки требуются данные пользователя, чтобы воспроизвести сценарий действий.

Предпочтительный для нас метод обмена такой информацией - по электронной почте или через наш FTP сайт. Пользуйтесь следующими адресами:

- United States - techsupport@mapinfo.com
- Europe - support-europe@mapinfo.com
- СНГ - support@esti-map.ru

Ошибки в программном обеспечении

Если Вы обнаружите в программном обеспечении серьезную ошибку, "баг", то обратитесь к MapInfo Corporation, которая ведет базу по таким ошибкам, присваивает им номера. В последующих обновлениях или патчах эта ошибка в ряду других, накопившихся, будет исправлена.

Другие источники технической поддержки

MapInfo Test Drive Center

Веб сайт Test Drive Center корпорации MapInfo это форум пользователей, интересующихся техническими вопросами а так же место ознакомления с новыми программными продуктами. Вы можете загрузить отсюда оценочные версии программ, а так же получить последние патчи.

Вам надо заполнить регистрационную форму, чтобы получить доступ к услугам Test Drive Center. Это одноразовый процесс. Как только новые продукты и услуги становятся доступны в Test Drive Center, Вам не надо перерегистрироваться, чтобы получить к ним доступ. Вы можете просто обновить имеющуюся регистрационную информацию, указав в ней интересующий Вас новый программный продукт.

База данных архивов MapInfo-L

MapInfo Corporation вместе с Биллом Тоеном (Bill Thoen), предлагает интернетовский архив переписки MapInfo-L, снабженный системой поиска. Архив структурирован по датам и по темам.

MapInfo Corporation использует эту базу данных для помощи пользователям, управлением почтой MapInfo-L до сих пор занимается Bill Thoen. Больше информации о MapInfo-L может быть получено в MapInfo Test Drive Center (<http://testdrive.mapinfo.com>).

Обзор языка MapBasic

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

2

Глава

- >Что такое среда разработчика MapBasic?
- >Главные особенности языка MapBasic
- >Как осваивать MapBasic?



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. Подробнее см. Главу 3 "Работа в интегрированной среде разработки программ").
- Компилятор языка MapBasic. После написания программы ее надо откомпилировать, собрать "выполняемый" файл (т.е. такой, который можно запустить под MapInfo).
- 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 QBasic). Новые вариации языка BASIC, такие как QBasic и MapBasic, скорее больше напоминают Паскаль, чем общеизвестный BASIC.

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

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

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, то он позволяет использовать всю гамму географических возможностей MapInfo.

Как создать и запустить MapBasic-программу?

В главе 3 "Работа в интегрированной среде разработки программ" подробно описан процесс создания приложений на MapBasic. Если же Вы хотите начать прямо сейчас, то Вам надо выполнить 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:
4. Note "Привет от MapBasic-a!"
5. Выполните команду **Файл > Сохранить**, чтобы сохранить программу в файле. Укажите имя файла, например, WELCOME.MB (стандартное расширение программ на MapBasic – .MB).

Внимание: Не закрывайте окно с текстом программы.

6. Выполните команду **Сборка > Компилировать файл**. MapBasic откомпилирует Вашу программу (WELCOME.MB) и создаст соответствующий исполняемый файл (WELCOME.MBX).
7. Запустите MapInfo.
8. Выполните команду **Файл > Запустить программу MapBasic**. MapInfo запросит у Вас название программы, которую следует запустить. Если Вы выберете WELCOME.MBX, то MapInfo запустит программу, которая выдает сообщение "Привет от MapBasic-a!".

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 можно найти в главе 3 "Работа в интегрированной среде разработки программ".

Главные особенности языка MapBasic

MapBasic позволяет настроить интерфейс MapInfo

С помощью MapBasic Вы можете настраивать интерфейс MapInfo. Приложения 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 создавать свои программы в MapInfo

Приложения 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. Приложение же на MapBasic позволяет сгенерировать градусную сетку без утомительного рисования.

Средства доступа к базам данных

Всего одного оператора MapBasic достаточно для выполнения сложных 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.

Приложения MapBasic являются переносимыми

Программы, написанные на языке MapBasic, можно запускать на любой вычислительной платформе, в которой работает MapInfo. Другими словами, отлаженное приложение на MapBasic можно запускать и в среде MapInfo для Windows, и в среде MapInfo для Macintosh.

Переносимость сокращает усилия, которые требуется затратить Вам, программисту. Создав приложение один раз, Вы можете предложить его всем своим пользователям, независимо от того, какую систему они используют – Mac или Windows.

В целом, Вам не требуется вносить никаких изменений в программу для ее корректной работы в другой операционной системе. Имеются лишь некоторые исключения, связанные с использованием специфических средств этих систем (например, DDE-технологии, доступной только в Microsoft Windows, или Apple Events, используемых только на Macintosh). В главе 10 "Создание переносимых прикладных программ" будут более подробно описаны требования к переносимому коду.

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, Вы можете приступить к изучению языка MapBasic с помощью данного руководства и Справочной системы.

Руководство пользователя MapBasic

Здесь изложены основные концепции программирования на языке MapBasic. Руководство пользователя MapBasic – первая книга, с которой Вам следует ознакомиться при изучении языка MapBasic.

Каждая глава в Руководстве пользователя MapBasic посвящена определенной области в программировании; например, в главе 6 рассматривается создание интерфейса пользователя (своих меню и диалогов), а глава 8 рассказывает об операциях ввода/вывода. Обязательно следует прочитать главу 4 "Основы языка MapBasic".

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.

Внимание: В Руководстве пользователя MapBasic мы будем часто ссылаться на пример ТЕХТВОХ.МВ. Вы можете ознакомиться с ним, прежде чем изучать MapBasic. Текст программы ТЕХТВОХ.МВ приведен в Приложении В.

Рабочие Наборы MapInfo

MapInfo 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.

Предположим, например, что Вы хотите написать программу на MapBasic, которая создает шаблон печатной страницы. В этом случае Вы можете создать такой шаблон вручную в MapInfo 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.

Внимание: 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 подсказок 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+ТАВ. 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

В MapInfo можно открыть окно, которое называется MapBasic. Использование этого окна также может помочь Вам в изучении языка MapBasic.

Чтобы открыть это окно:

1. Запустите MapInfo
2. Выполните команду Настройка > Показать окно MapBasic.

На экране появится окно MapBasic. По мере выполнения различных операций в MapInfo, 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; описание операторов, которые не работают в окне MapBasic под MapInfo, имеется в разделе "Предупреждение". В основном это управляющие операторы (такие, как **For...Next**).

Окно MapBasic можно использовать и для отладки. Подробнее см. главу 5.

Техническая поддержка, обучение и консультации

При возникновении проблем с использованием языка MapBasic, сотрудники отделов технической поддержки 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 интересные 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 ЭСТИ МЭП.

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.

3

Глава

- > Введение в интегрированную среду MapBasic
 - > Как ввести или отредактировать программу
 - > Компиляция программ
 - > Проект: Сборка приложения из нескольких модулей
 - > Обзор меню среды разработки программ MapBasic
-

Введение в интегрированную среду MapBasic

Основой интегрированной среды разработки программ на языке MapBasic является текстовый редактор, позволяющий создавать и редактировать программы на MapBasic. Меню этого 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, Заккрыть, Печатать и Сохранить**; меню **Правка** – команды **Отменить, Вырезать, Копировать, Вставить**. Конечно, помимо этих операций MapBasic позволяет выполнять еще и особые действия (например, компиляцию).

Как ввести или 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**.

Внимание: MapBasic автоматически добавляет расширение **.MB** к имени файла. Поэтому Вы можете просто набрать **WELCOME**, а на самом деле полное имя файла окажется **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. В ней перечислены русские буквы; Вы можете нажимать клавиши с буквами независимо от языка, установленного на клавиатуре, т.е. “переключать” клавиатуру не обязательно.

Клавиши	Результат
Home / End	Переход в начало или конец строки

Клавиши	Результат
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 “Перейти к строке”
Ctrl-O	Вызов диалога “Открыть”
Ctrl-N	Открывается новое окно редактора
Ctrl-S	Сохранение содержания активного окна
Ctrl-P	Печать содержания активного окна
Ctrl-A	Выбор всего текста в окне
Ctrl-C	Копирование выбранного текста в Буфер Обмена
Ctrl-X	Удаление выбранного текста и копирование его в Буфер Обмена
Ctrl-V	Вставка текста из Буфера Обмена
Ctrl-Del	Удаление следующего после курсора слова
Del	Удаление выделенного текста; без копирования в Буфер
Ctrl-F	Вызов диалога “Найти и Заменить”
Ctrl-G	Повторяет самую последнюю команду поиска
Ctrl-R	Заменяет выбранный текст (с использованием текста замены из диалогового окна “Найти и Заменить”) и выполняет следующий поиск
Ctrl-J	Вызов диалога “Выбор проекта”
Ctrl-K	Компилирует программу в активном окне
Ctrl-E	Команда Next Error; пролистывает окно редактирования, чтобы показать строку, которая вызвала ошибку трансляции
Ctrl-L	Сборка программы
Ctrl-U	Посылает сообщение MapInfo, чтобы выполнить активную программу
F1	Отображает Справку. Совет: Если Вы выбираете имя функции перед нажатием F1, Справка показывает раздел, описывающий эту функцию.
F8	Вызов диалога “Выбор шрифта”
Ctrl-F4	Закрывает активное окно редактирования
Alt-F4	Выход из среды разработки MapBasic
Shift-F4	Расположение окон мозаикой

Клавиши	Результат
Shift-F5	Расположение окон каскадом

Действия с мышкой

Действие с мышкой	Результат
Двойное указание (“двойной щелчок”)	Выбирает слово. Двойное указание в списке сообщений об ошибках приводит к пролистыванию окна, чтобы показать строку программы, которая вызвала ошибку.
Тройное указание (“тройной щелчок”)	Высвечивает всю строку текста (только 32-битной версии).
Перенос мышкой	Перемещение текста в другое окно копирует текст. Перемещение текста внутри того же самого окна перемещает текст (если Вы не держите клавишу CTRL нажатой во время перемещения, иначе происходит копирование).

Совет: интерактивная справка MapBasic содержит примеры программ. Вы можете “перетаскивать” мышкой куски кода из окна Справки в окно редактирования.

1. Вызовите Справку (например, нажимая F1).
2. Нажмите на левую кнопку мышки и, не отпуская, переместите выделенный текст из окна Справки прямо в окно редактирования.

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 КОМПИЛИРОВАТЬ 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.
- 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. 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.

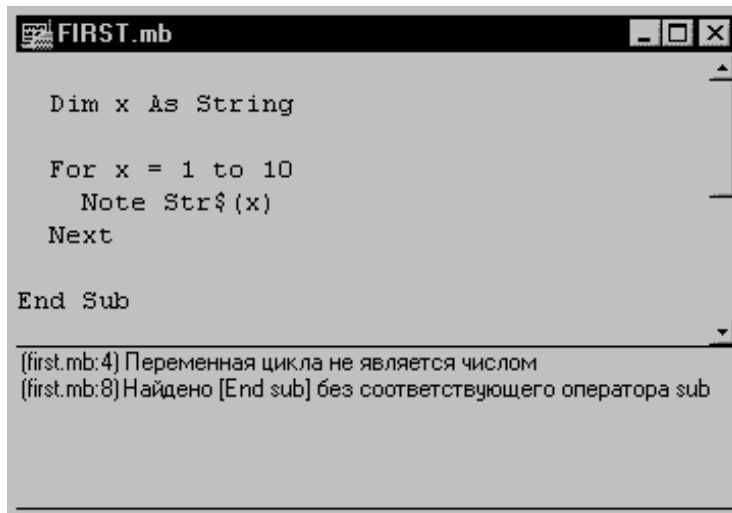
Компиляция программ

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

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

Компилятор языка MapBasic проверит синтаксис Вашей 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.



Внимание: Если Вы дважды укажете мышкой на сообщение 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). Такой .MBX-файл можно передавать пользователю как законченное приложение (если он, конечно, правильно работает). Так что, если Вы хотите отослать пользователю приложение на MapBasic, но не исходные тексты, отошлите только файл .MBX (без файлов .MB).

Об ошибках, выявляемых компилятором

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 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 из меню Файл в MapInfo. В диалоге "Запустить" 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. MapBasic пошлет сообщение MapInfo о том, что MapInfo следует запустить приложение.

Написание программ на MapBasic в других редакторах

Если Вы предпочитаете работать в каком-то уже известном Вам текстовом редакторе, Вы можете использовать его для создания программ на MapBasic. Только 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 вместо Save. Подробнее о том, как сохранить файл в стандартном текстовом формате, написано в документации к текстовому редактору.

Компиляция программ, созданных во внешнем редакторе

Мы уже обсуждали использование команды Компилировать Файл в MapBasic для компиляции программ из активного окна. В MapBasic имеется и другой способ компиляции: с помощью команды Компилировать из файла из меню Файл.

Если Вы создавали программу во внешнем текстовом редакторе, Вы можете использовать команду Компилировать из файла для компиляции программы. Эта команда компилирует программу, не показывая ее в окне редактора MapBasic.

При выполнении команды Компилировать 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 Компилировать 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 sub1.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.) Скажем, следующая командная строка выполняет сборку приложения 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 `-D` или `-L` параметр, то окно MapBasic закрывается после компиляции или сборки соответствующих файлов.

Проект: 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 файл проекта.

Примеры файлов проектов

Приложение TextBox использует файл проекта (tbproj.mbp) следующего вида:

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

А приложение SCALEBAR использует файл проекта (SBPROJ.MBP):

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

В обоих примерах последняя строка указывает, что MapBasic должен включить в приложение модуль AUTO_LIB. 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 он запускает 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.
2. Введите в появившееся окно:

```
[Link]
```

3. Введите строку вида `Application=appfilename` (где `appfilename` – это имя исполняемого файла). Например:
`Application=C:\MB\CODE\CUSTOM.MBX`
4. Введите строку вида `Module=modulename` (где `modulename` – название объектного файла MapBasic). Например:
`Module=C:\MB\CODE\CUSTOM.MBO`

Объектные файлы в MapBasic имеют расширение “.МВО”. MapBasic создает объектные файлы как результат компиляции модуля, являющегося частью проекта.

При выполнении команды Сборка > Компилировать Файл MapBasic пробует построить на основании заданного файла исполняемый файл (с расширением “.МВХ”). Однако, если в тексте компилируемого файла имеются вызовы внешних процедур или функций, MapBasic не может создать .МВХ-файл; в этих случаях MapBasic предполагает, что файл является частью проекта и строит объектный файл (.МВО) вместо исполняемого (.МВХ). Кроме того, MapBasic создает объектный файл, если компилируемый файл не содержит процедуры Main.

5. Повторите шаг 4 для каждого модуля Вашего приложения.
6. Выполните команду Файл > Сохранить в, чтобы сохранить файл проекта. В диалоге "Сохранить файл" выберите тип “Проект ” (из списка в левом нижнем углу диалога) так, чтобы Ваш файл получил расширение “.МВР” (MapBasic Project).
7. Закройте окно редактирования (командой Файл > Закрыть 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 > Компилировать из файла.
2. Выполните команду Сборка > Выбрать файл проекта, чтобы указать, какой файл проекта должен использовать MapBasic при сборке. В диалоге "Выбрать файл проекта" выберите название файла проекта (“.МВР”) 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.

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

3. Чтобы собрать приложение, выполните команду Сборка > Собрать проект. MapBasic проанализирует все объектные файлы (.МВО), перечисленные в файле описания проекта. Если при сборке не выявлено ошибок, MapBasic построит исполняемый (.МВХ) 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 С. 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”.

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 выбрать 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, TEXTBOX.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. TEXTBOX.MB вызывает внешнюю процедуру HandleInstallation, так как тело HandleInstallation не содержится в файле TEXTBOX.MB.

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:

1. Объявите ее оператором `Global` в файле заголовков (например, в `"GLOBALS.DEF"`).
2. 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 (он описан в главе 11).

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, если требуется переменная, доступная всем процедурам и функциям из данного файла и недоступная из других модулей (например, 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 (напр., 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.

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

Примечание: Каждое из открываемых окон в среде MapBasic может содержать до 64К текста. Программы большего размера нельзя открыть в текстовом окне MapBasic. Способы обхода этого ограничения приведены в разделе “Ограничения текстового редактора MapBasic”.

- **Заккрыть** закрывает активное текстовое окно. Если Вы вносили в него изменения, MapBasic запросит: сохранить ли эти изменения, прежде чем закрыть окно.
- Команда **Заккрыть** активна, если хотя бы одно окно открыто.
- **Заккрыть все** закрывает все открытые окна. Как и после команды **Заккрыть**, MapBasic выдает запрос, сохранять ли внесенные изменения.
- Команда **Заккрыть все** активна, если хотя бы одно окно открыто.
- **Сохранить** сохраняет содержимое активного окна на диске.
- Команда **Сохранить** активна, если вносились какие-либо изменения.
- **Сохранить в** сохраняет содержимое активного окна в новом файле на диске.
- Команда **Сохранить в** активна, если хотя бы одно окно открыто.
- **Восстановить** отменяет все изменения, внесенные со времени последнего сохранения содержимого окна в файл на диске. Команда **Восстановить** активна, 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 Собрать проект из меню Сборка, собирающего текущий проект.)
- **Настройка печати** задает режимы печати (например, номер принтерного порта).
- **Печатать** печатает содержимое 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 (Clipboard) текст можно потом вставить с помощью команды Вставить (см. ниже). Команда Вырезать активна, если в одном из окон выбран текст.
- **Копировать** копирует выбранный текст в буфер обмена (Clipboard), не удаляя его с экрана. Команда Копировать активна, если в одном из окон выбран текст.
- **Вставить** вставляет содержимое буфера обмена (Clipboard) 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 выделенный 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 операторов, содержащих ошибки.

- **Найти** осуществляет поиск заданной текстовой строки в активном окне. Команда Найти активна, если открыто хотя бы одно окно.

Чтобы найти заданную строку поступайте следующим образом: наберите строку, которую Вы хотите найти, в диалоге "Поиск и замена" (окошко "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.

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 подсказок.

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.

Если Вы хотите заменить этот образец, нажмите Ctrl+И (что аналогично выполнению команды Заменить и найти еще).

Если же Вы не хотите заменять данный образец искомой строки, нажмите Ctrl+E (сокращение команды Найти еще).

Продолжайте поиск, нажимая Ctrl+И для замены строки и Ctrl+E для продолжения поиска.

- **Найти еще** находит следующий образец заданной в диалоге поиска строки.
- Команда Найти еще активна, если открыто хотя бы одно окно и была выполнена команда Найти.
- **Заменить и найти еще** заменяет выбранный текст и находит следующий образец заданной строки.
- **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.
- **Предыдущая ошибка** похожа на команду Следующая ошибка, но листает окно назад, к предыдущей ошибке.
- Команда Предыдущая ошибка активна, если при компиляции 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 открыто хотя бы одно окно.

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 о том, что следует запустить на выполнение приложение из активного текстового окна. Команда Запустить запускает MapInfo, если оно было запущено.
- **Информация** показывает статистическую информацию о программе из активного окна.
- Команда 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.
- Команды Показать/Скрыть 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 Вы можете 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. Свертывание окон не поддерживается в версии для Macintosh.
- **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. Кроме того, механизм перекрестных ссылок позволяет быстро находить описание нужных операторов.

Содержание открывает окно "Contents" (Содержание), откуда Вы можете переходить к интересующим Вас темам, указывая на их заголовки или выбирая тему в диалоге (для этого нажмите кнопку "Search" ("Поиск")).

Поиск открывает диалог "Search" ("Поиск").

Как пользоваться открывает тему, описывающую порядок работы с подсказками.

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

Внимание: 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 Edit > Copy (Правка > Копировать) из меню Правка в окне Справочной системы.

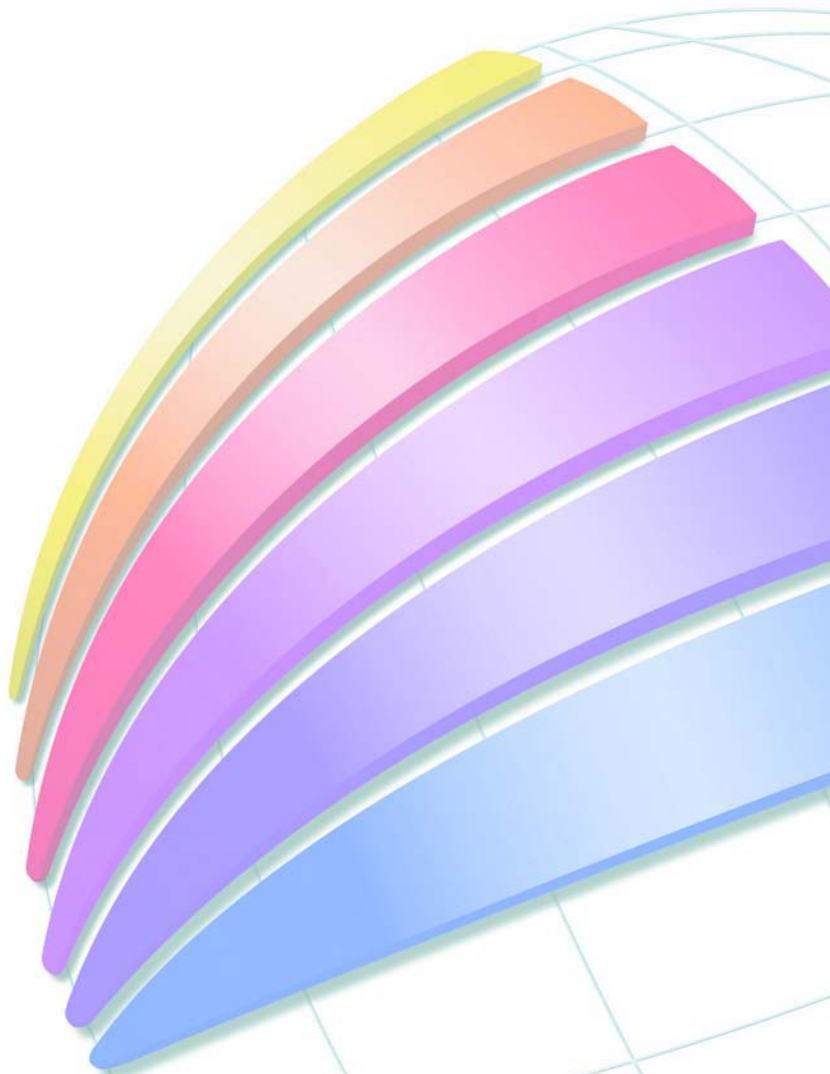
Основы языка MapBasic

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

4

Глава

- > Общие замечания о синтаксисе языка MapBasic
 - > Выражения
 - > Циклы и другие управляющие операторы
 - > Процедуры
 - > Процедуры-обработчики системных событий
 - > Рекомендации по использованию процедур-обработчиков системных событий
 - > Функции, созданные пользователем
 - > Директивы компилятора
 - > Организация программ
-



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).

Большие и малые буквы

Компилятор языка MapBasic не различает большие (прописные) и малые (строчные) буквы. Вы можете использовать в своих программах БОЛЬШИЕ-БУКВЫ, маленькие-буквы или Большие-и-Маленькие-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 с большой 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 "Счетчик переполнился"
End If
```

Размещение 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 "Неправильное значение счетчика"
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 констант-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 Вы этого не сделаете, при выполнении программы будет зафиксирована ошибка (например, “Переменная или поле **Field CMD_INFO_DLG_OK** не определена”).

Как вводить операторы в окно MapBasic

Программа MapInfo содержит так называемое Окно 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, 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** (напр., **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., например, 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	Целое число от -32767 до 32767; занимает два байта
Integer	Целое число от -2 миллиардов и 2 миллиардов; занимает четыре байта
Float	Вещественное число с плавающей точкой; хранится в восьми байтах в формате IEEE
String	Строка произвольной длины (до 32767 символов).
String * n	Строка фиксированной длины n (до 32767 символов).
Logical	TRUE или FALSE.
Date	Дата
Object	Графический объект; подробнее этот тип описан в главе 9
Alias	Ссылка на колонку таблицы; подробнее этот тип описан в главе 7
Pen	Объект типа линия; см. главу 9.
Brush	Объект типа штриховка; см. главу 9.

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 байт:

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

Замечание: 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(3+7) As Date
```

Доступ к элементу массива осуществляется следующим образом:

имя_массива(номер-элемента)

Так, следующий оператор присваивает значение первому элементу массива start_date:

```
start_date(1) = "6/11/93"
```

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)      ' Определим размер
массива
ReDim names(counter + 10)    ' Увеличим размер 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).
- Максимальный размер массива в приложениях на MapBasic, запускаемых под MapInfo для Windows, составляет 7000 элементов. При запуске приложений на MapBasic на других платформах (например, в MapInfo для Windows 95, Windows NT или Macintosh) максимальный размер массива составляет 32767 элементов.

Чтобы хранить больше данных, чем может вместить массив, Вы можете использовать для их размещения таблицы. Подробнее об использовании таблиц речь пойдет в главе 7 "*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 **оператора Type...End оператора Type**. Пользовательский 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:

имя_переменной.имя_поля

В следующем примере присваиваются значения всем полям переменной manager:

```
manager.name = "Гриша"  
manager.title = "Самый главный менеджер"  
manager.id = 111223333
```

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

```
staff(1).name = "Дима"  
staff(1).title = "Программист"
```

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      ' Объявление глобальных
                             переменных           ' типа Float
Global start_date As Date   ' Объявление глобальных
                             переменных           ' типа Date
Sub Main
    Dim x, y, z As Float ' Объявление локальных
                             переменных в         ' процедуре Main
    Call initialize_globals
    ...
End Sub
Sub initialize_globals
    gx = -1                  ' Присваивание значения
                             глобальной          ' переменной GX
    gy = -1                  ' Присваивание значения
                             глобальной          ' переменной GY
    start_date = CurDate() ' Присваивание значения
                             глобальной          ' переменной START_DATE
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 Microsoft Windows используют механизм так называемого динамического обмена данными (Dynamic Data Exchange – DDE) для чтения и модификации глобальных переменных MapBasic-программ; подробнее см. в главе 11.

Область определения

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`.

Обрабатывая 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, MapBasic 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:

```
counter = 23
```

Здесь переменной `counter` 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. ниже раздел “Константы в языке MapBasic”.

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). 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 склеивания 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 **АнаАА**, 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()** и **Perimeter()**.

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 – числовая константа с плавающей точкой.

Шестнадцатиричные константы: В MapBasic 4.0 поддерживаются шестнадцатиричные константы в синтаксисе VisualBasic: &Hчисло (где число - шестнадцатиричное число). В следующем примере переменной присваивается шестнадцатиричное значение 1A (равное десятичному 26):

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

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

```
counter = 1,250,000 ' Не будет работать!
```

Если число включает десятичные точки (десятичный разделитель), символы разделителя должны иметь период, даже если компьютер настроен на другие десятичные разделители.

Строковые константы: Строковые константы заключаются в двойные кавычки. Например:

```
last_name = "Nichols"

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 (") 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
```

Либо дата-константа должна быть заключена в двойные кавычки. Например:

```
d_enddate = "12/31/1995"
```

В дате-константе, использующей двойные кавычки, могут быть указаны как все 4 цифры года, так и только две последние :

```
d_enddate = "12/31/95"
```

Если Вы пропускает год, то используется текущий год:

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

Замечание. 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.

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

Для конфигурации формата даны используйте настройки Microsoft Windows, *Дата и время* из Панели управления.

Константы-имена таблиц: О псевдонимах таблиц мы поговорим подробно в главе 7 "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.

Тип	Пример	Замечания
Integer	i = 1234567	
SmallInt	m = 90	
Float	f = 4 size = 3.31 debt = 3.4e9	
String	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 Chr\$().
Logical	edits_pending = 1 edits_pending = TRUE	1 = истина, 0 = ложь Файл заголовков языка MapBasic содержит объявления TRUE и FALSE.
Date	d_starting = 19940105 date_done = "3/23/88" paidddate = "12-24-1993" yesterday = CurDate() - 1	
Alias	col_name = "Pop_1990" col_name = "COL1"	Присвоить можно строку. Подробнее см. главу 7.
Pen	hwypen = MakePen(1, 3, BLACK)	Выражение только такого вида.
Brush	zbrush = MakeBrush(5, BLUE, WHITE)	Выражение только такого вида.
Font	lbl_font = MakeFont("Helv", 1, 20, BLACK, WHITE)	Выражение только такого вида.
Symbol	loc_sym = MakeSymbol(44, RED, 16)	Выражение только такого вида.
Object	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 = "Общая сумма: " + 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 операторы: 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)

Знак минус (-) может использоваться для обозначения отрицательного значения:

```
x = -23
```


Строковые операторыСтр: Оператор сложения (+) позволяет соединить ("склеить") несколько строк в одну.

Note "Служащий: " + служ_имя + " " + служ_фамилия

Можно использовать оператор (&) вместо (+). Этот оператор превращает оба операнда в строки и затем склеивает их.

Внимание: Символ (&) используется также для обозначения шестнадцатиричных чисел (&число). Поэтому, используя его для склеивания строк, не забывайте про пробелы.

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	Результат
дата + целое число	более поздняя дата
дата - целое	более поздняя дата
дата - дата	значение типа Integer: 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
' А теперь посчитаем, сколько дней прошло
' с начала календарного года:
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	Пример
=	если равны	If a = b Then ...
<>	если не равны	If a <> b Then ...
<	если меньше	If a < b Then ...
>	если больше чем	If a > b Then ...
<=	если меньше или равно	If a <= b Then ...
>=	если больше или равно	If a >= b Then ...
Between...And...value is within range	Если x между 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 = 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.

Внимание: 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	Пример
And	оба операнда истинны	If a And b Then...
Or	значение одного из операндов – истина	If a Or b Then...
Not	операнд ложен.	If Not a Then..

Например, следующий оператор **If...Then** выполняет две проверки: меньше ли значение переменной x, чем ноль, и больше ли значение x, чем 10. 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	Пример
Contains	первый объект содержит центроид второго объекта	If a Contains b Then...
Contains Part	первый объект содержит часть второго объекта	If a Contains Part b Then...
Contains Entire	первый объект полностью содержит второй объект	If a Contains Entire b Then...
Within	центроид первого объекта содержит второй объект	If a Within b Then...
Partly Within	второй объект содержит часть первого объекта	If a Partly Within b Then...
Entirely Within	второй объект полностью содержит первый объект	If a Entirely Within b Then...
Intersects	два объекта имеют общую точку	If a Intersects b Then...

Более подробные сведения о графических объектах содержатся в главе 9.

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 (сложим 2 + 3 и получим 5), а затем умножение (5 * 4), то итоговым результатом будет 20. В действительности, однако, умножение имеет более высокий приоритет, чем сложение. Это означает, что MapBasic сначала выполнит умножение (3 * 4, получим 12), а затем уже сложение (2 + 12, получим 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.

Высший приоритет:	скобки возведение в степень отрицание сложение, деление, Mod, целочисленное деление, сложение, вычитание, склеивание строк (&) географические операторы, операторы сравнения, Like Not
Низший приоритет:	And Or

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 (100) слишком
велико."
Else
    Note "Счетчик в порядке."
End If
```

Внимание: Отметим, что ключевое слово **ElseIf** пишется слитно.

Оператор **If...Then** может содержать несколько предложений **ElseIf**,

указанные в них условия проверяются последовательно. Впрочем, если Вам 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 “Первый квартал года”. Аналогично, если текущий месяц относится ко второму кварталу, то текстовой переменной присваивается константа “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
```

Внимание: 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).

```
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 Вами конечному значению (в примере, если `next_one` меньше или равна 12), MapBasic выполняет следующую итерацию цикла.

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

Внимание: Если цикл **For...Next** использует явно заданные вещественные значения (например, **For i = 0.1 to 1.0 Step 0.1**), то такой цикл может выполняться в MapInfo для Macintosh иначе, чем в MapInfo для Windows или UNIX (т.е. может быть выполнено разное число итераций на Macintosh и PC под Windows). Это возможно из-за особого способа обработки плавающих вещественных чисел в компьютерах Macintosh.

Подробнее о цикле **For...Next** см. Справочнике MapBasic.

Оператор 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)
```

Обратим 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 выполниться 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.

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 Вам следует проследить, чтобы Ваша программа не выполняла оператор **End Program**, поскольку **End Program** остановит выполнение приложения и удалит все созданные Вашим приложением меню. Подробное обсуждение пользовательских меню Вы найдете в главе 6 "Создание пользовательского интерфейса".

Завершение выполнения программы и сеанса работы с MapInfo

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

Оператор **End Program** не требуется. Есть ситуации, когда надо осторожно пользоваться оператором **End Program**. Например, если Ваше приложение добавляет разделы меню к меню MapInfo, то может понадобиться оставить возможность запустить приложение во время данного сеанса MapInfo. В таком случае, не надо использовать оператор **End Program**, поскольку **End Program** завершит работу приложения и удалит его меню из главного меню. Подробнее об этом написано в Главе 6, Создание пользовательского интерфейса.

Завершение работы приложения и MapInfo

Оператор **End MapInfo** завершает работу приложения (как и **End Program**), и затем завершает сеанс работы MapInfo.

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 Вашей 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 начинает 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!"
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 в главе 3 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, имеющих параметры

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 типа `Date` `report_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 такие параметры, 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 "Период выдачи сообщений: 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, Map-Info 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 процедур.

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 **WinFocusChangedHandler**. 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 системах с графическим интерфейсом (Graphical User Interface), пользователь действует путем ввода с клавиатуры или указания мышью. Пользуясь техническими терминами, можно сказать, что действия пользователя вызывают системные события. Существуют разнообразные события; например, когда пользователь выбирает команду из меню, генерируется событие выбора из меню, когда пользователь закрывает окно, генерируется событие закрытия окна и т.д.

Что такое обработчик системных событий?

Обработчик системных событий – это часть программы на языке MapBasic, которая реагирует на системные события. Как только в ответ на действие пользователя было сгенерировано некоторое системное событие, Ваше приложение должно отреагировать соответствующим образом. Например, если пользователь выбрал команду из меню, от программы может потребоваться открыть диалог; или, если пользователь закрыл какое-то окно, могут стать недоступными некоторые команды в 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 см. в главе 6 "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.

Название специальной процедуры	Обрабатываемые события (подробно описано в Справочнике)
EndHandler	Вызывается в случае, если приложение заканчивает работу или пользователь закрывает MapInfo. EndHandler может использоваться для корректного завершения работы (например, удаления временных файлов).
ForegroundTaskSwitchHandler	Вызывается, когда MapInfo теряет или приобретает фокус.
RemoteMapGenHandler	Вызывается, когда клиент OLE Automation вызывает метод MapGenHandler; ранее использовалась в приложениях MapInfo ProServer.
RemoteMsgHandler	Вызывается, когда приложение является сервером при обмене данными, и клиент присылает команду.
RemoteQueryHandler()	Вызывается, когда приложение действует как сервер для обмена внешними процессами, и удаленный клиент посылает запрос.
SelChangedHandler	Вызывается при каждом изменении таблицы Selection. Так как Selection изменяется часто, процедура SelChangedHandler должна быть небольшой, чтобы по возможности не увеличивать время работы приложения.
ToolHandler	Вызывается в том случае, когда пользователь применяет один из инструментов в окне Карты, Списка или Отчета.
WinChangedHandler	Вызывается, когда пользователь изменяет или листает содержимое окна Карты. Так как содержимое окна Карты может меняться очень часто, процедура WinChangedHandler должна быть небольшой, чтобы по возможности не увеличивать время работы приложения.
WinClosedHandler	Вызывается, когда пользователь закрывает окно Карты, Списка, Графика или Отчета.
WinFocusChangedHandler	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 "Пример процедуры ToolHandler начал работу. "
    + "Выберите инструмент (+) и укажите им в окно
Карты "
    + "чтобы распечатать координаты точки на
карте."
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. (Текст программы **TEXTBOX.MB** приведен в Приложении 2).

```
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
```

Когда вызываются обработчики событий?

По умолчанию приложение на 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.

Замечание: 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, то оно остается неактивным в памяти и ждет, пока пользователь выберет одну из команд в пользовательских меню. Подробнее о пользовательских меню под MapInfo см. в главе 6 "Создание интерфейса пользователя".

Рекомендации по использованию процедур-обработчиков системных событий

Делайте процедуры-обработчики короткими!

Имейте в виду, что некоторые процедуры-обработчики событий вызываются часто. Например, если Вы создаете процедуру **SelChangedHandler**, MapInfo 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.

Selecting Without Calling SelChangedHandler

If you are using a **Select** statement, but you do not want the statement to trigger the **SelChangedHandler** procedure, include the **NoSelect** keyword. For example:

```
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 вызывать процедуру снова 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
  ' Далее может выполняться оператор Select,
  ' который не заиклится.
  Set Handler SelChangedHandler On
End Sub
```

Функции, созданные пользователем

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, если имеются) As
ResType
           операторы
End Function
```

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

Внутри конструкции **Function...End Function** имя функции доступно как параметр, переданный ссылкой. В операторе в теле функции имени функции 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 (отражающий 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 )
    ' теперь dollar_amount 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, имеющую то же имя, что и некоторая стандартная функция языка MapBasic. При вызове функции с таким именем будет выполнена Ваша функция вместо стандартной (стандартная не видна).

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 не обнаружен, компилятор ищет его в том каталоге, в котором установлен MapBasic.

По мере разработки Вами 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 3 "Работа в интегрированной среде разработки программ", каждое текстовое окно в редакторе MapBasic может содержать не более 50К 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. в главе 3.).

Организация программ

Приложения на MapBasic могут содержать некоторые или все операторы и другие конструкции, описанные в этой главе. Помните вместе с тем об особенностях организации некоторых секций программы на языке MapBasic. Например, операторы **Global** должны быть вынесены вне тел процедур (вне конструкций **Sub...End Sub**).

Следующий пример показывает типичное расположение различных частей программы.

Операторы глобального уровня:

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

Тело процедуры Main:

```
Sub Main  
    операторы Dim  
    ...  
End Sub
```

Другие процедуры:

```
Sub ...  
операторы Dim  
...  
End Sub
```

Пользовательские функции:

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


Отладка и обработка ошибок

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. 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.

5

Глава

- >Отладка и обработка ошибок
 - >Отладка программ на языке MapBasic
 - >Обработка ошибок
-

Ошибки при выполнении

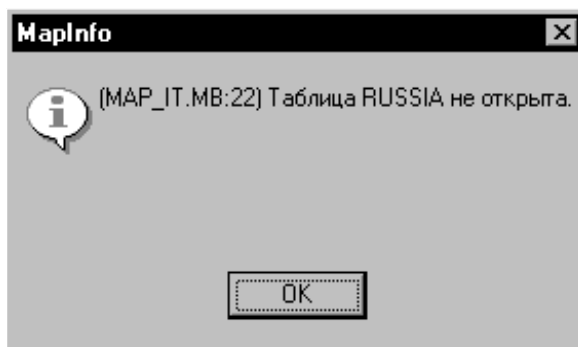
Существуют два основных типа ошибок в программах: ошибки при компиляции и ошибки при выполнении. Ошибки, фиксируемые во время компиляции, мы обсуждали в главе 3. Обычно это – синтаксические ошибки или опечатки, выявляемые при компиляции.

Ошибки же 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, оператор:

Map From RUSSIA

будет откомпилирован успешно. Однако, если при его выполнении не будет открыта таблица с именем “RUSSIA”, то будет зафиксирована ошибка. При обнаружении ошибки выполнения MapInfo 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 – 22. 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 ROSSIA набрал RUSSIA). 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 используют в сочетании с Окном 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)::
Наберите ? Dim, чтобы получить список всех локальных переменных.
Наберите ? Global, чтобы получить список всех глобальных переменных.
Наберите ?имя_переменной, чтобы увидеть значение переменной.
Наберите ?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. 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:

- Внутри конструкции операторов **Function...End Function**.
- Внутри оператора работы с диалогами **Dialog**, поскольку этот оператор управляет работой с активным диалогом на экране.
- 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 другой 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: " + counter
```

чтобы выдавать текущее значение в окне "Сообщения".

Обработка ошибок

В аккуратно написанной программе должна быть по возможности предусмотрена реакция на возможные ошибки при выполнении. 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 вместе.

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.

Внимание: В каждый конкретный момент времени возможна обработка только одной 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 режим обработки ошибок включается
'   bad_open - метка начала обработки ошибки.
    Open Table "orders.tab"
    OnError Goto not_mappable

' Здесь включается режим обработки еще одного типа
ошибок
'   not_mappable - метка начала обработки ошибки.
    Map From orders
    OnError Goto 0
    Browse * From orders
last_exit:
    Exit Sub
' Exit Sub позволяет избежать выполнения
обработчиков          'ошибок при правильном
выполнении программы.

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.

Интерфейс пользователя

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



6

Глава

- > Программная обработка событий
 - > Меню
 - > Стандартные диалоговые окна
 - > Новые диалоговые окна
 - > Окна
 - > Инструментальные панели
 - > Запуск программы в среде MapInfo
 - > Рекомендации по повышению производительности
-

Принципы построения интерфейса пользователя в MapBasic

Создавая программу на языке MapBasic, Вы можете создать для нее интерфейс пользователя MapInfo. 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 содержимым. MapBasic может также показывать сообщения в специальном окне "Сообщения" и в строке сообщений в нижней части окна MapInfo.
- **Инструментальные панели:** Пользователь может создавать свои инструментальные панели со своими кнопками, изменять стандартные панели и их состав. Одна из стандартных панелей MapInfo, "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 Windows 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. Обычно в MapBasic-программах для этого заготавливаются процедуры-обработчики или просто обработчики, которые срабатывают в ответ на определенные события.

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

1. Настройка системы меню MapInfo с помощью операторов **Create Menu**, **Alter Menu** и других.
2. Создание процедуры-обработчика для каждого нового элемента меню. Обработчик оформляется как sub-процедура, 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, ТЕМАТИЧЕСКАЯ КАРТА из меню **КАРТА**).

Как уже упоминалось в главе 3, оператор **Call** вызывает sub-процедуру. Если процедура является обработчиком, то оператор **Call** не нужен. Вместо него 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,  
"В&ыход" 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 TEXTBOX.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. Ниже в этой главе более подробно описывается процесс работы с новыми кнопками инструментальных панелей.

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. Ниже в этой главе более подробно описано, как создавать диалоги и работать с ними.

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.

В этом разделе рассматриваются:

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 – это отдельная команда из списка команд меню. Например, меню **ФАЙЛ** обычно содержит элементы **ОТКРЫТЬ**, **ЗАКРЫТЬ**, **СОХРАНИТЬ** и **ПЕЧАТАТЬ**. К элементам меню также относятся разделительные линии между группами команд; поскольку последние дальше почти не упоминаются, то в дальнейшем термины "элемент меню" и "команда меню" будут эквивалентны.

Как только пользователь выберет элемент меню и нажмет кнопку мыши или клавишу ENTER, 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 (sub-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 sub-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 sub-процедуру "report_sub". Если будет выбрана команда **Квартальный отчет**, то MapInfo вызовет sub-процедуру "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, следующий оператор определяет команду меню **Раскрасить карту**. Если пользователь выберет эту команду, MapInfo обратится по коду M_MAP_THEMATIC к своей внутренней процедуре, которая представлена стандартной командой **Карта > Выделить условно**.

```
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_PREVIOUS
```

Создание нового меню

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

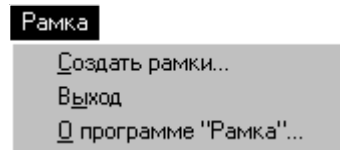
```
Create Menu "Рамка" As
    "&Создать рамки..." Calling create_sub,
    "В&ыход" Calling Bye,
    "&О программе ""Рамка""..." Calling About
```

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 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 меню в 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. Например, в системе Macintosh пользователь должен держать нажатой кнопку мыши во время всего процесса выбора команды.

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 задает 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

```

Меню СТИЛЬ РАМКИ вначале недоступно. (Этот режим задается с помощью символа “(”, предшествующего имени команды.)

Когда пользователю понадобится сделать доступной эту команду, 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 недоступной, а это делается следующим оператором:

```
Alter Menu Item PickFrame Disable
```

"PickFrame" – это имя подпрограммы (sub-процедуры) в 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, "PickFrame"), 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 > 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 меню в правую часть строки заголовков. В связи с этим возникает небольшое неудобство из-за того, что строка меню заканчивается меню СПРАВКА (Help). В большинстве программ принято, чтобы меню СПРАВКА (Help) было последним в строке заголовков. Если Вы хотите поместить новое меню слева от меню СПРАВКА, то последнее надо сначала удалить. Например, в программе ТЕХТВОХ используется следующий прием:

```
Alter Menu Bar Remove ID 6, ID 7
```

```
Alter Menu Bar Add "Программы", ID 6, ID 7
```

Первый оператор удаляет меню СПРАВКА (меню с ID-номером 7). Второй добавляет два меню сразу: ПРОГРАММЫ и СПРАВКА. В результате меню СПРАВКА снова оказывается последним в строке заголовков.

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 (ФАЙЛ, ЗАПРОС и т.д.) приведен в описании оператора **Create Menu Bar** 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, обращающемся 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. 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 7.0 обеспечивает быстрые меню – 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 "MapperShortcut" 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 ситуации, когда одна или две команды меню вызывают одну и ту же процедуру-обработчик, то в эту процедуру обычно помещается обращение к функции **CommandInfo()** для того, чтобы определить, какая именно команда была выбрана. 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** обращается к элементу меню по имени соответствующей процедуры-обработчика, то MapBasic изменит *все* элементы меню, которые обращаются к той же процедуре. 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 клавишных комбинаций

Клавишные комбинации 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 является из подчеркивания под буквой Ф. При создании элемента меню можно добиться подчеркивания буквы, поместив перед ней знак амперсанда (&). Амперсанд автоматически задает клавишное сокращение как в Windows, так и в Macintosh.

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
```

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

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

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"). При задании "горячей" клавиши нужно ориентироваться на английские буквы, которые им соответствуют стандартно; при этом действующая в системе Windows раскладка клавиатуры может быть другой, даже русской; например, в русской версии MapInfo комбинация CTRL+Я будет работать так же, как CTRL+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.

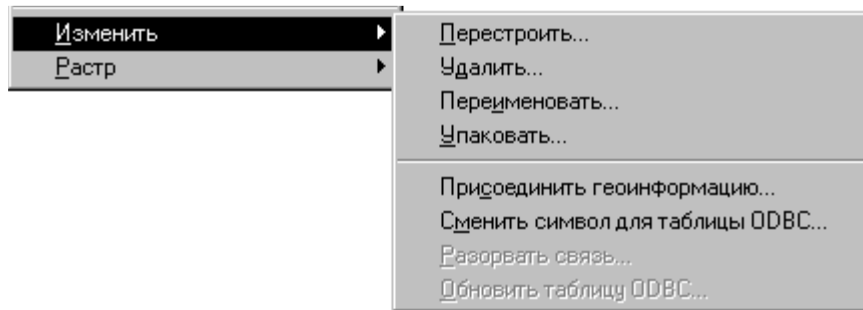
В системе UNIX этот файл называется MAPINFO.MNU; в MapInfo для Macintosh – Mapinfo.Menus; в MapInfo для Windows – MAP-INFO.MNU.

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.

Внимание: Прежде чем изменять что-либо в файле MAPINFOW.MNU, сохраните его резервную копию, например, в файле BACKUP.MNU. Ибо, 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**.



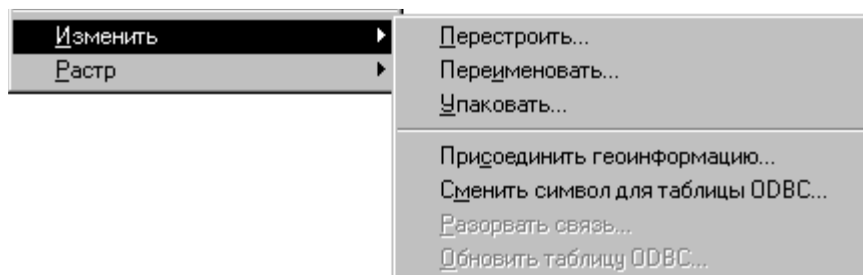
Изучив файл MAPINFOW.MNU, Вы обнаружите, что меню **ИЗМЕНИТЬ** определено следующим образом:

```
Create Menu "&Изменить" As
  "&Перестроить..."  calling 404,
  "У&далить..."      calling 409,
  "Пере&именовать..." calling 410,
  "&Упаковать..."     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..."  calling 404,
  "Пере&именовать..." calling 410,
  "&Упаковать..."     calling 403
  ...
```

При следующем запуске MapInfo меню **ТАБЛИЦА > ИЗМЕНИТЬ** предстанет в сокращенном виде:



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. Этим можно воспользоваться, например, для придания единообразия интерфейсу или для сокрытия от не в меру активных пользователей потенциально опасных команд типа **УДАЛИТЬ**.

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.

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.

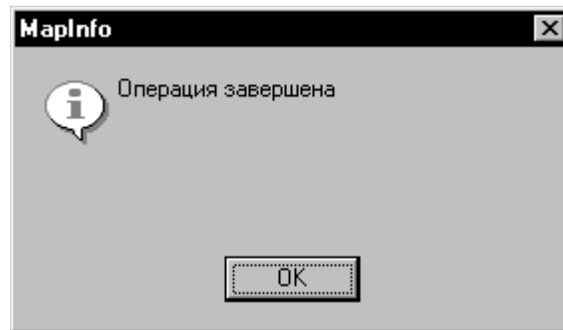
Таким образом, Вы можете добиться того, чтобы один вариант файла MAPINFOW.MNU был доступен всем пользователям, а другие – индивидуальным пользователям. Версия MAPINFOW.MNU, предназначенная для общего пользования, помещается в один каталог с MapInfo, а версии для индивидуальных пользователей – в их личные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

Стандартные диалоговые окна

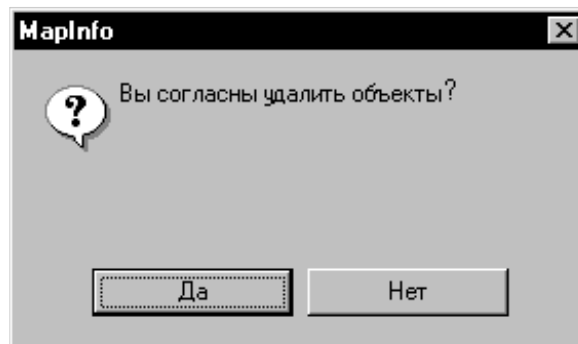
Диалоговые окна (или диалоги) являются существенным элементом интерфейса. В арсенале MapBasic содержится несколько операторов и функций, украшающих Вашу программу диалоговыми окнами:

Показ простого сообщения

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



Показ диалога с двумя кнопками

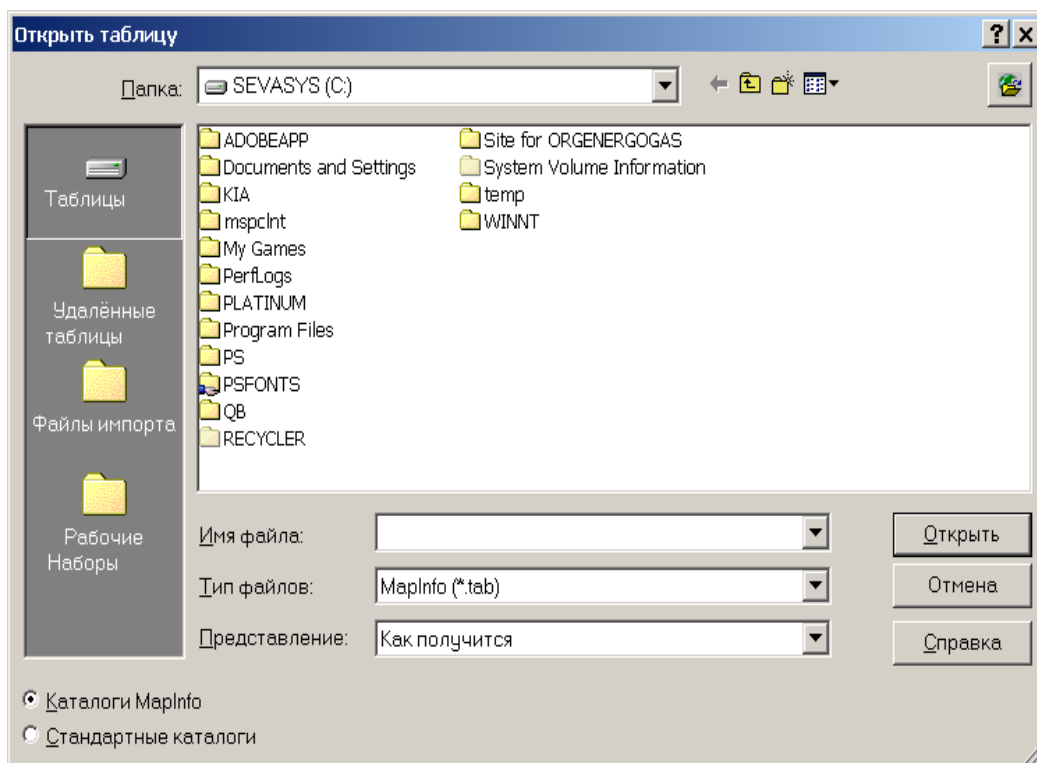


Функция **Ask()** показывает диалог с вопросом и двумя кнопками. На двух кнопках обычно написано "OK" и "Cancel", но Вы можете использовать свои надписи. Если пользователь выберет кнопку "OK", функция вернет логическую истину (TRUE) или вернет логическую ложь (FALSE) в противном случае.

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".



Показ только одной записи

В MapInfo нет стандартного диалога, который показывает только одну запись из таблицы. Для этого можно использовать окно Информация. Ниже в этой главе описано, как управлять этим окном из программы MapBasic.

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

Все, что может появиться в диалоге, называется *элементами*. 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 "ОК" (элемент **OKButton**) и "Отмена" (элемент **CancelButton**).

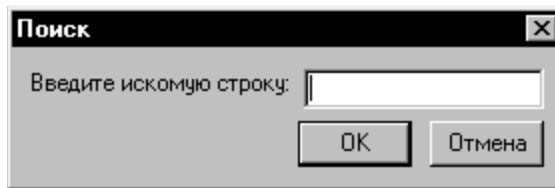
```
Dim строка_поиска As String

Dialog
  Title "Поиск"
  Control StaticText
    Title "Введите искомую строку:"
  Control EditText
    Into строка_поиска
  Control OKButton
  Control CancelButton

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

```
End If
```

Этот оператор **Dialog** показывает следующий диалог:



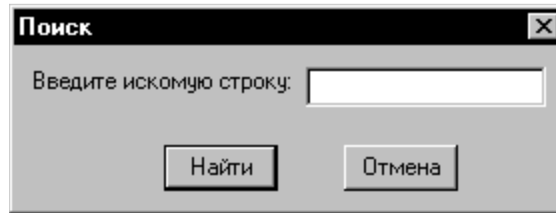
Размеры и положение элемента диалога

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 видом предыдущего диалога и хотите изменить положения и надписи на кнопках. Это достигается применением предложений **Position** и **Title** (последнее заменяет надпись на кнопке):

```
Dialog
  Title "Поиск"
  Control StaticText
    Title "Введите искомую строку:"
  Control EditText
    Into строка_поиска
  Control OKButton
    Title "Найти"
    Position 50, 30
  Control CancelButton
    Position 110, 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:

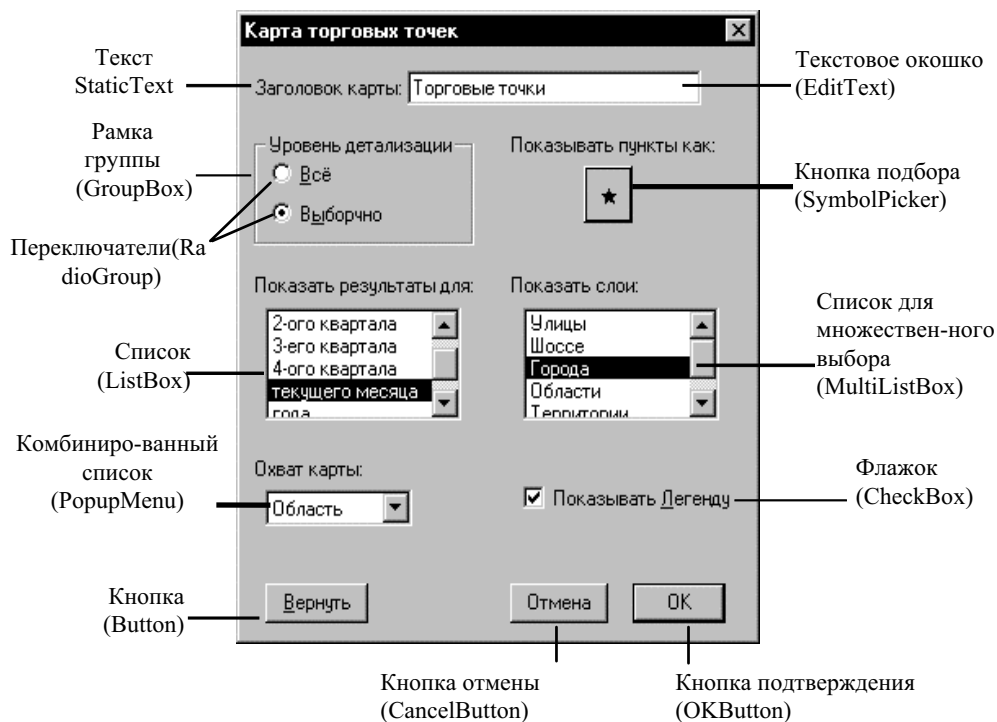
Position 20, 16

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.



Элемент **StaticText** представляет из себя простую надпись, например:

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

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

```
Control EditText
Value "Торговые точки"
Into maptitle
ID 1
Position 68, 8 Width 120
```

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

```
Control GroupBox
Title "Уровень детализации"
```

```
Position 5, 30
Height 40 Width 90
```

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

```
Control RadioGroup
Title "&Все;В&ыборчно"
Value 2
ID 2
Into details
Position 15, 42 Width 72
```

Нажав на одну из кнопок типа **PenPicker**, **BrushPicker**, **FontPicker** или **SymbolPicker**, можно включить один из диалогов подбора пера, штриховки, шрифта или символа соответственно. В приведенном примере использована кнопка **SymbolPicker**, на которой изображен текущий символ (квадрат). Например:

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

Элемент **ListBox** создает список, из которого пользователь может выбрать одну из строчек. MapBasic автоматически добавляет строку прокрутки, если строчек в списке слишком много. В нашем примере список задается следующей конструкцией:

```
Control ListBox
Title "Показать результат для"
"1-ого квартала;2-ого квартала;3-его квартала;4-ого
квартала;текущего месяца;года"
ID 4
Value 5
Into quarter
Position 10, 92 Width 65 Height 40
```

Другой вид списка, задаваемый элементом **MultiListBox**, позволяет выбрать более чем одну строчку из списка, используя принятую в Windows технику выбора с нажатой клавишей SHIFT или CTRL. В нашем примере список для множественного выбора задается следующей конструкцией:

```
Control MultiListBox
Title "Улицы;Шоссе;Города;Области;Территории"
ID 5
```

```
Value 3
```

```
Position 115, 92 Width 65 Height 40
```

Комбинированный список (элемент **PopupMenu**) можно открыть, нажав на кнопку со стрелочкой. Открывается обычный список. В нашем примере комбинированный список задается следующей конструкцией:

```
Control PopupMenu
```

```
Title "Город;Область;Территория;Регион;Вся страна"
```

```
Value 2
```

```
ID 6
```

```
Into mapscope
```

```
Position 10, 157
```

Флажок (**CheckBox**) может быть установлен или сброшен. В примере он задается следующей конструкцией:

```
Control CheckBox
```

```
Title "Показывать &Легенду"
```

```
Into showlegend
```

```
ID 7
```

```
Position 115, 155
```

Кнопки (элемент **Button**) присутствуют во всех диалогах; по крайней мере, одна почти всегда есть. MapBasic поддерживает стандартные кнопки **OKButton** and **CancelButton**, создающие кнопки "ОК" и "Отмена", а также позволяет задавать свои собственные. В примере заданы три кнопки с помощью следующих предложений:

```
Control Button
```

```
Title "&Вернуть"
```

```
Calling resetsub
```

```
Position 10, 190
```

```
Control OKButton
```

```
Title "ОК"
```

```
Calling oksub
```

```
Position 160, 190
```

```
Control CancelButton
```

```
Title "Отмена"
```

```
Position 110, 190
```


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 "Отмена", 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*. Например, элемент **ListBox** описан в главе **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 четвертый элемент в списке (элемент **ListBox**), добавьте следующее предложение **Value** в описание элемента **ListBox**:

Value 4

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 в *Справочнике MapBasic* (главы **Control...**).

Считывание установок диалога

В описании большинства элементов может содержаться предложение **Into**. 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 **Into**, нужно закрыть диалог нажатием на кнопку "ОК".

Переменная, задаваемая после слова **Into**, должна быть задана как локальная или глобальная в Вашей программе и, разумеется, должна по типу соответствовать элементу диалога. Например, переменная для элемента **EditText** должна быть строковой (типа **String**), а 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.

Внимание: После того, как пользователь закроет диалог, нажав на кнопку "ОК", MapBasic помещает значения во все переменные, заданные предложениями **Into**; 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 sub-процедура), автоматически выполняющаяся тогда, когда пользователь укажет мышкой на элемент диалога. Для сопоставления элементу диалога процедуры-обработчика используется предложение **Calling handler**; обработчик – это sub-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 “Восстановить”; если пользователь нажмет на эту кнопку, программа восстанавливает первоначальные значения элементов. Чтобы добиться такого эффекта, нужно сопоставить кнопке “Восстановить” 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 из комплекта поставки (NIEWS.MB). Диалог “Именованные Виды” из программы, входящей в комплект поставки MapBasic (см. Приложение 1), содержит список названий 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. Он используется для того, чтобы скрыть, показать, сделать доступным и недоступным элемент диалога или восстановить его начальное значение. С помощью оператора **Alter Control** можно также установить фокус (т.е. сделать элемент активным). См. 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 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.

Внимание: **Внимание:** 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-й квартал;2-й квартал;3-й квартал;4-й"+
"квартал;годовые итоги"
```

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 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 этого обработчика 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 предварительного задания номера элемента предложением **ID** в предложении **Control MultilistBox**.

Клавишные комбинации, 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 "&Восстановить"
  Calling reset_sub
```

Кнопка "&Восстановить" 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 зададите клавишную комбинацию для элемента **StaticText**, находящегося слева от окошка **EditText**, то при нажатии клавишной комбинации, заданной для текстового элемента, активным станет окошко ввода (в нем появится курсор).

Клавишные комбинации, заданные для среды Windows, не действуют в средах Macintosh и UNIX.

Заккрытие диалога

Все диалоги, которые создаются оператором **Dialog**, закрываются, как только пользователь нажмет на кнопки ОК или "Отмена". Другими словами, Вы не можете работать в MapInfo, пока на экране присутствует диалог, созданный оператором **Dialog**.

Если Вы хотите создать диалог, который присутствует на экране в то время, пока Вы работаете в среде MapInfo, то Вы должны создать приложение в другой программной среде (например, Visual Basic), и которое связывается с MapBasic-программой (например, посредством оператора **Run Command**).

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 этих кнопок требуется сохранить диалог на экране, например, после нажатия на кнопку "Отмена" (т.е. выбора элемента **CancelButton**), программа может выдать запрос на подтверждение отмены (типа "Вы уверены, что хотите вернуться в окно исходного диалога?").

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

Оператор **Dialog Remove** закрывает диалог немедленно. Он должен действовать из процедуры-обработчика. Оператор **Dialog Remove** может быть использован, например, при двойном указании мышкой на строчку списка элемента **ListBox**.

Программа из поставки примеров (NIEWS.MB) коказывает как работать с двойным щелчком.

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 Redistricter**. 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()**. 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.

Внимание: 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)**. А 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 "cm"
    Height 3 Units "cm"
    Width 4 Units "cm"
```

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**. Команда **Set Map** подробно описана в *Справочнике MapBasic*.

Оператор **Shade** создает условную или тематическую Карту, на которой данные выделены цветом или другим способом в зависимости от условий. Оператор **Shade** может провести условное выделение следующими, принятыми в MapInfo способами: диапазонами, круговыми и столбчатыми диаграммами, размерными символами, плотностью точек или индивидуально. Создав условную Карту, MapInfo добавляет ее в виде слоя в активное окно Карты. Чтобы изменить условную Карту, используйте оператор **Set Shade**.

Начиная с версии 5.0 можно использовать оператор **Create Grid** для создания нового типа тематических карт, позволяющий проводить новые виды анализа карт. Новый слой растровой поверхности имеет непрерывную цветовую раскраску. Интерполятор осуществляет вычисления в узлах растровой поверхности по нескольким алгоритмам. Данные для интерполяции берутся из таблицы MapInfo, содержащей точечные значения. 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 оператор **Add Map Layer** содержит ключевое слово **Animate**, слой становится слоем анимации. При перемещении объекта по такому слою перерисовка производится MapInfo очень быстро, вне зависимости от сложности этого объекта.

Эта новая возможность полезна в тех приложениях, где требуется частое обновление окна Карты. Например, Вы создаете приложение, управляющее движением автомобилей (или поездов), которое представляет каждый автомобиль точкой. Вы можете получать координаты при помощи технологии GPS (Global Positioning System) и перерисовывать точки в новом положении. В такого рода программах содержимое окна Карты постоянно обновляется и использование слоя анимации позволит делать это значительно быстрее, чем для обычного слоя.

В следующем примере открывается таблица, и создается слой анимации:

```
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 доступа к этому слою; в частности, он не может использовать инструмент “Информация” для указания точек на слое.
- Каждое окно Карты может иметь не более одного такого слоя. 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**.

Демонстрационная программа

В комплекте поставки имеется демонстрационная программа ANIMATOR.MBX.

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, представляющие 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.

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, но
' отображается только нужный
```

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. Так, 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 окно Графика построит 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 именами колонок, но Вы можете задать любое допустимое числовое выражение.

Окна Графиков

В окне Графика данные из таблицы и выражения, из них составленные, представляются графическими элементами. В следующем примере окно Графика построит 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 Районов

Работа с районами начинается с выполнения команды **Create Redistricter**. Этот оператор 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 Redistricter**. 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 "Inflation"
From World
Into WORLD_QUERY

Set Window Info
    Title "Данные о стране"
    Table WORLD_QUERY      Rec 1
    Font MakeFont("Arial Cyr", 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 для отредактирования 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

Внимание: Обратите внимание: Все поля в окне Информации будут "только читаемыми", если Вы отображаете таблицу, которая является объединением (типа таблицы **StreetInfo**) или таблицы запроса, которая использует предложение **Group By**, для вычисления сводных величин.

Инструментальные панели

Инструментальные панели (ButtonPads) являются важной частью интерфейса пользователя MapInfo. Инструментальные панели являются плавающими по экрану окнами, содержащими одну или более трехмерных кнопок. Нажимая на эти кнопки, пользователь запускает различные операции MapInfo.

В MapInfo есть несколько стандартных инструментальных панелей. С помощью программы на языке MapBasic можно добавлять в них новые кнопки либо создавать новые инструментальные панели.

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.

В программе MapBasic можно создавать три типа кнопок: кнопки-инструменты (ToolButtons), кнопки-переключатели (ToggleButtons) и кнопки запуска (PushButtons). Тип кнопки определяет условия, при которых MapBasic обращается к соответствующему ей обработчику.

- **Кнопки запуска (PushButton):** Нажатием на сигнальную кнопку пользователь вызывает соответствующую процедуру, а кнопка возвращается в ненажатое положение.
Примером запускающей кнопки может служить кнопка "Управление слоями", вызывающая на экран одноименный диалог.
- **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-инструмент 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

The following statements and functions let you create and control custom buttons and ButtonPads:

Create ButtonPad

Этот оператор создает новую инструментальную панель.

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 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.

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 прочесть координаты точки, на которую указал пользователь.

Если Ваша программа создает одну или несколько новых кнопок, функция **CommandInfo()** позволяет определить, какая из них была нажата. Таким образом, в процедуре-обработчике можно несколько раз вызывать **CommandInfo()**: сначала для определения нажатой кнопки, затем для определения координат (2 раза, по одному вызову на координату) и, наконец, для определения того, была ли нажата клавиша SHIFT при указании мышкой.

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. Операторами **Create ButtonPad** или **Alter ButtonPad** можно создать новые кнопки или курсоры, но не процедуру.

Если пользователь запустил несколько программ MapBasic одновременно и каждая из них содержит процедуру ToolHandler, то каждая из программ добавляет в панель "Операций" 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 "Мои Кнопки" As
        PushButton
            Icon    MI_ICON_ZOOM_QUESTION
            Calling button_prompt
            HelpMsg "Нажатие на эту кнопку открывает диалог
настроек"
            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 "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 придерживать ее нажатой.

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
настроек"
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 "Масштаб" (ScaleBar) добавляет кнопку в панель "Программы".

Создание кнопки типа ToolButton

Кнопки типа ToolButtons предназначаются для включения инструментальных средств MapInfo, таких, как Лупа или Линия. Если программа создала кнопку 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 "Мои Кнопки" As
    ToolButton
      Icon    MI_ICON_LINE
      DrawMode DM_CUSTOM_LINE
      Cursor  MI_CURSOR_CROSSHAIR
      Calling draw_via_button
      HelpMsg "Этот инструмент рисует линию."
    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)
  ' Запомнить место, в котором отпущена кнопка мыши:
  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 мышки, 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.

Предложение **DrawMode** также позволяет определить, увидит ли что-нибудь пользователь на экране, перемещая мышью 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. Если пользователь отменит операцию (например, нажатием на ESC при перемещении указателя мыши), MapInfo не вызывает обработчик. В теле процедуры-обработчика можно использовать функцию **CommandInfo()** для определения точки, в которой пользователь нажал кнопку мыши.

Режимы рисования и их коды перечислены в файле **ICONS.DEF**, а также в разделах **Alter ButtonPad** в *Справочнике MapBasic*.

Выбор пиктограммы для создаваемой кнопки

Когда Вы определяете новую кнопку, Вы указываете пиктограмму, которая появляется 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 Info MapInfo's. Код **MI_ICON_INFO** определен в файле **ICONS.DEF**.

```
Alter ButtonPad "Операции"
Add Separator
```

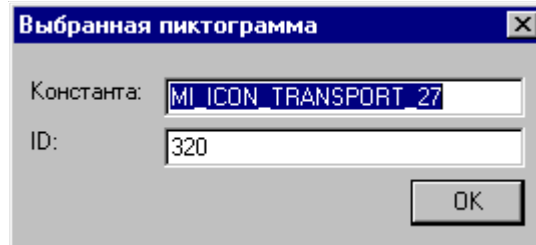
```
Add PushButton
Icon      MI_ICON_INFO
Calling procedure_name
```

Еще с версии 4.0 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**, и затем команду пункт из ее меню. Чтобы видеть код для конкретной пиктограммы, задержите курсор мышки над кнопкой с ней. Подсказка для этой кнопки покажет Вам код пиктограммы. Вы также можете скопировать код в Буфер Обмена:

1. Запустите приложение **ICONDEMO.MBX**.
2. Выберите команду из меню **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 **OK**, чтобы закрыть диалог.
6. Переключитесь в окно MapBasic. Нажмите клавиши **Ctrl+M** (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. Далее строится географический оператор **Select**, и в результате выбираются все области России, которые содержат эту точку.

```
Sub t_click_handle
    Dim fx, fy As Float
    fx = CommandInfo(CMD_INFO_X)
    fy = CommandInfo(CMD_INFO_Y)
    Select * From Russia
        Where obj Contains CreatePoint(fx, fy)
End Sub
```

Внимание: Совет: 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 Стрелка) в свою инструментальную панель. Например, в следующем примере создается инструментальная панель, содержащая две кнопки: стандартную кнопку Стрелка и созданную в программе:

```
Create ButtonPad "Рамка" As
' Здесь вставляется стандартная кнопка
ToolButton
    Icon MI_ICON_ARROW
    Calling M_TOOLS_SELECTOR
    HelpMsg "Выбор объекта для редактирования/
nСтрелка"
' Здесь вставляется новая кнопка
ToolButton
```

```

Icon MI_ICON_LINE
DrawMode DM_CUSTOM_LINE
Calling sub_procedure_name
HelpMsg "Рисование нового маршрута/нНовый
маршрут"

```

Первое предложение **Calling** задает номер инструмента **M_TOOLS_SELECTOR**, заданный в файле **MENU.DEF**. Этот номер соответствует инструменту Стрелка (каждая стандартная кнопка **Map-Info** имеет свой номер в **MENU.DEF**). Вторая кнопка – новая, поэтому предложени **Calling** для нее содержит вызов процедуры.

В описании новой кнопки включено предложение **DrawMode**, но в первой такого предложения нет, потому что для стандартной кнопки режим рисования менять нельзя.

Внимание: Внимание: кнопки типов **ToolButton** и **ToggleButton** не взаимозаменяемы. Нельзя просто так в тексте программы заменить слово **ToolButton** на **ToggleButton** и наоборот. Кнопка **ToolButton** возвращает координаты места указания мышкой, а **ToggleButton** 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 **MAPINFOW.MNU**. Эти определения можно копировать в текст Вашей программы.

Файл меню содержит оператор **Create ButtonPad** который определяет кнопки **MapInfo**.

Внимание: Можно скопировать определение кнопок из **MAPINFOW.MNU** и вставить код в свою программу.

Добавление подсказок для кнопок.

Если пользователю не ясно назначение кнопки на инструментальной панели, **MapBasic** позволяет Вам снабдить кнопку одним из двух видов подсказок:

1. Подсказка в строке состояния. Появляется, если пользователь нажал и удерживает кнопку. Сообщение остается все время, пока кнопка нажата.
2. “Парящая” подсказка. Появляется, если указатель мышки некоторое время находится над кнопкой.

Оба типа подсказки определяются в предложении **HelpMsg** в операторах **Create Button Pad** и **Alter Button Pad**. Внутри этого предложения первая часть строки содержит сообщения в строке состояния, затем следует символ /n и сообщение для “парящей” подсказки. Например:

```
Create ButtonPad "Мои дела" As  
PushButton  
Icon MI_ICON_ZOOM_QUESTION  
Calling generate_report  
HelpMsg "Эта кнопка создает Отчет/nСоздание Отчета"  
Show
```

В этом примере текст для строки 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. Пример:

```
Alter ButtonPad "Операции" Fixed
```

Ключевое слово **Fixed** позволяет зафиксировать панель вверху. Ключевое слово **Float** возвращает ее в прежнее “плавающее” состояние. Эти ключевые слова 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 **ButtonPad-Info()**.

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 главе **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 11.

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 **ФАЙЛ > ЗАПУСТИТЬ ПРОГРАММУ MAPBASIC**. Вы можете также устроить свою программу так, что она будет запускаться автоматически при запуске MapInfo, 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 командной строкой, т.е. задать программу, которая будет запускаться при открытии MapInfo. Для задания командной строки нужно проделать следующие действия:

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; если Вы хотите, чтобы и этот диалог появлялся, и программа запускалась, то Вам нужно использовать другой прием: создать свой стартовый Рабочий Набор.

Запуск программ из Рабочего Набора STARTUP

STARTUP – это имя специального 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 SCALEBAR, нужно создать следующий Рабочий Набор STARTUP:

```
!workspace
!version 600
!Charset Neutral
Run Application "scalebar.mbx"
```

По первым двум строчкам MapInfo распознает файл Рабочего Набора. В третьей 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.

В среде Macintosh стартовый Рабочий Набор называется "Startup Workspace" и располагается в одной папке с MapInfo, в системной (System) папке или папке настроек пользователя (Preferences).

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 3.x 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 **ФАЙЛ > 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 "Самозагрузка". 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. Теперь в системе меню Вы можете найти команду **РАМКА**.

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. Этот модуль используется также другими программами из комплекта поставки для организации автоматического запуска; Вы также можете использовать его в своих программах.

Текст программного модуля AUTO_LIB.MB включен в комплект поставки MapBasic. Инструкции по пользованию этим модулем Вы можете найти в комментариях в тексте программы.

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 этой главе.

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:

1. Используйте оператор **Set ... Redraw Off**. Затем напишите все операторы, изменяющие окно Карты, и, наконец, оператор **Set ... Redraw On**. Окно будет перерисовано один раз.
2. 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 резко 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 ProgressBars 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 происходить, 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.

7

Глава

- >Открытие таблиц с помощью MapBasic
 - >Создание новых таблиц
 - >Доступ к Косметическому слою с помощью таблицы “CosmeticN”
 - >Доступ к окнам Отчетов с помощью таблицы “LayoutN”
 - >Редактирование в многопользовательской среде
 - >Файлы-компоненты таблицы
 - >Таблицы, содержащие растровые изображения
 - >Работа с метаданными
 - >Работа со сшитыми таблицами
 - >Доступ к удаленным данным
 - >Доступ и изменения в удаленных базах при помощи связанных таблиц
 - >Рекомендации по повышению производительности при работе с таблицами
-

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 так долго, как открыта та таблица. Таблица не переименовывается навсегда. Для постоянного переименования, используйте оператор **Rename Table**. Если используется дополнительное предложение **Interactive** в операторе **Open Table**, и если таблица, которую вы определяете, не может быть размещена в указанной директории, то MapInfo откроет диалог, подсказывающий, где можно разместить таблицу. Если Вы пропускаете ключевое слово **Interactive** и таблица не может быть размещена 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. Поскольку заранее не известно, какую именно таблицу выберет пользователь, Вам придется 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:

Синтаксис колонки	Пример
имя_таблицы.имя_колонки	world.country
имя_таблицы.COLn	world.COL1
имя_таблицы.COL(n)	world.COL(i)

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)
```

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. Названия колонок в операторе **Select** в случае, когда он применяется только к одной таблице, также не обязаны содержать приставку имя_таблицы. Если же в предложении **From** оператора **Select** перечислены несколько таблиц, то названия колонок должны начинаться с приставки имя_таблицы. Более подробная информация об использовании оператора **Select** содержится в Руководстве пользователя MapInfo и главе "Select" Справочника 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 операторе

```
Select Country, Population/1000000 From World
```

имена колонок — "Country" и "Population" — указаны явно.

Иногда название колонки заранее не известно, оно определяется во время выполнения. Возможно, 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 такого типа используется в выражении на месте ссылки на колонку. Например:

```
Dim val_col As Alias
```

```
val_col = "Inflat_Rate"
Select * From world Where val_col > 4
```

MapBasic подставит значение val_col (псевдоним Inflat_Rate) при выполнении оператора **Select**, чтобы выбрать все страны с уровнем инфляции больше 4 процентов.

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", "popula-
tion", 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 вида 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:

```
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 BYPOP, переменная median_row содержит номер записи о штате со средним уровнем населения. Последний оператор выбора отбирает все штаты, идущие после него в упорядоченной таблице BYPOP.

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 незакодированные 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 9.

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 (напр., “Тверская && Лесная”). После выполнения оператора **Find** Вы должны дважды вызвать **CommandInfo()**: чтобы определить, найден ли адрес, и еще раз, 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: для этого используйте функцию **CreatePoint()** или оператор **Create Point**. Примеры см. в описании оператора **Find** в *Справочнике MapBasic* или интерактивной справке.

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

Если Вам нужно выполнить геокодирование большого объема, Вы можете приобрести специализированный пакет MapMarker, который продается отдельно. MapMarker геокодирует быстрее, чем MapInfo, и позволяет за один проход геокодировать все Соединенные Штаты. Приложение MapBasic может управлять пакетом MapMarker через его интерфейс. Для подробной информации о пакете MapMarker войдите в контакт с отделом сбыта MapInfo; номера телефонов есть в начале этого и других руководств MapInfo.

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 MapBasic 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 **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**. А 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, а также имена клиентов, их почтовые адреса, город, 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. Этот оператор 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 налегания 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 поиск и геокодирование 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 **НАЙТИ**, Вы должны проиндексировать колонку (поле) фамилий. Большинство запросов 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.

Замечание: 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.
- **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 Запрос1).

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  
table_count = NumTables() 'число открытых таблиц  
ReDim tablenames(table_count) 'Изменить размер массива,
```

```

чтобы
                                ' хранить имена всех таблиц.
For i = 1 To table_count 'Цикл по таблицам
                                'Считать имя таблицы
    tablenames(i) = TableInfo(i, TAB_INFO_NAME)
    Print tablenames(i)'показать имя таблицы в окне 'сообщений
Next
Print "*****"                'напечатать разделитель

```

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 "ЗапросN", где N — это целое число, большее или равное единице. Подобно Selection, ЗапросN является временной таблицей. Функция **SelectionInfo()** позволяет определить, какое стандартное имя-псевдоним присвоит MapInfo таблице Selection (скажем, Запрос1 или Запрос2).

SelectionInfo() также позволяет получить другую информацию о таблице Selection (число выбранных записей и т.п.).

Замечание: 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 "ЗапросN"

Работая в MapInfo, Вы можете заметить, что Вы открыли ряд таблиц с именами "ЗапросN" (Запрос1, Запрос2, и т.д.). Например, если Вы выбираете объекты Карты и затем просматриваете Список выбранных объектов, заголовок окна может выглядеть как "Запрос1 Список". Каждый новый ЗапросN — это "снимок" последней выборки. Программы MapBasic могут также могут открывать таблицы ЗапросN. Например, создание ссылки на колонку типа **Selection.Obj** заставляет MapInfo открывать таблицу ЗапросN. Если Вы хотите, чтобы ваша программа MapBasic закрыла таблицу ЗапросN, 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 ЗапросN, 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 ЗапросN. Следующий пример показывает, как закрыть таблицу ЗапросN.

' Сколько таблиц открыто?

```
i_open = NumTables()
```

' Доступ к таблице. Например:

```
Fetch First From Selection
```

```
obj_copy = Selection.obj
```

' Закроем последнюю таблицу ЗапросN

```
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 ЗапросN), 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 "Active":

```
Select * From sites
```

```
Where growth > 15
```

Into Active

Введение в SQL-запросы можно найти в Руководстве пользователя MapInfo. Подробное описание оператора **Select** см. в главе "Select" 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 (существительное); а когда он выполняет команду, он задает действие (глагол), которое следует совершить над заданным объектом. Программа ТЕХТВОХ (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 ТЕХТВОХ 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 не пуста). 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 6.

Доступ к Косметическому слою с помощью таблицы “CosmeticN”

Каждое окно Карты содержит особый слой, называемый Косметическим, на котором размещаются подписи и другие вспомогательные объекты. В программе на MapBasic Косметический слой рассматривается как обычная таблица с именем **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 с помощью таблиц “LayoutN”

Операторы работы с объектами языка MapBasic применимы к объектам в окнах Отчетов. Чтобы работать в окне Отчета, Вам следует указывать в качестве имени таблицы в операторах **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 **Layout**.

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 "Статистика" 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 "Статистика" (например, командой [НАСТРОЙКА > ПОКАЗАТЬ ОКНО СТАТИСТИКИ](#)), окно "Статистика" покажет сообщение вида "Таблица **Cosmetic1** содержит 1 выбранную запись." Аналогично, если выбрать объект в окне Отчета, то окно "Статистика" 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 не позволяет этого. 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

Работа в сети имеет особенности, перечисленные ниже.

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 Б попытался начать вносить изменения в эту таблицу. MapInfo не позволит сделать это и выведет сообщение “Нельзя редактировать. Кто-то другой еще сейчас редактирует таблицу”. Если попытка редактирования будет предпринята из программы MapBasic, произойдет ошибка времени выполнения. Пока таблица редактируется пользователем А, MapInfo не позволит Б изменять ее. Так будет до тех пор, пока А не произведет сохранение или восстановление таблицы или не закроет ее.

Замечание: Пользователь Б сможет читать таблицу, открытую А, в частности, изображать ее на Карте. Однако он не увидит изменений, произведенных в ней А до момента, когда тот ее сохранит.

Правило 2. Пользователь не может читать содержимое таблицы во время ее сохранения.

Окончив редактирование, А выполнил команду меню **ФАЙЛ > СОХРАНИТЬ ТАБЛИЦУ**. Во время сохранения таблицы пользователь Б попытался прочесть 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 эта кнопка, MapInfo повторит попытку доступа к таблице. Эту кнопку можно нажимать до тех пор, пока таблица не станет доступной.

Отмена

В этом случае диалоговое окно исчезнет с экрана, и операция будет отменена.

Замечание: Если ошибка случится при загрузке Рабочего Набора, нажатие на кнопку “Отмена” может привести к прерыванию загрузки этого набора. Например, Рабочий Набор содержит оператор Open Table. Если этот оператор 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, 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 выбрана эта кнопка, MapInfo повторит попытку сохранить таблицу. Эту кнопку можно нажимать до тех пор, пока таблица не будет сохранена.

Отмена

В этом случае диалоговое окно исчезнет с экрана, и операция будет отменена. Изменения в таблице не будут сохранены вплоть до следующей попытки сделать это.

Как избежать конфликтов при чтении

Как указано в предыдущем разделе, некоторые конфликты совместного доступа к таблице приводят к появлению окна диалога с кнопками “Повторить”/”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 любые операции таблицы, которые производят конфликт совместного использования, и MapInfo продолжит попытки повторить операцию в течение 100 секунд. Обратите внимание, что MapInfo повторяет операции таблицы вместо того, чтобы отобразить диалог “Повторить”/”Отмена”. Если конфликт все еще происходит после 100 секунд повторений, автоматическое повторение останавливается, и MapInfo отображает диалоговое окно.

Как избежать конфликтов при записи

Некоторые операторы MapBasic изменяют содержание таблицы. Например, утверждение **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 5).

Внимание: Используйте операторы **Set File Timeout** и **OnError** в разных местах. 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 предложения Insert или Com-
    mit
    ' привели к ошибке времени исполнения

    If Ask("Ошибка доступа; попробовать еще раз?",
    "Да", "Нет") Then
        ' ... попробовать еще.
        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, имеется возможность, 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-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-формат 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 географический индекс.

имя_файла.ind: Содержит индексы для различных колонок
таблицы.

Поскольку каждая таблица состоит из нескольких файлов, Вы должны быть осторожными при переименовании таблиц. Для переименования таблиц пользуйтесь командой **ТАБЛИЦА > ИЗМЕНИТЬ > ПЕРЕИМЕНОВАТЬ** в MapInfo или оператором **Rename Table** в MapBasic.

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: .ТАВ-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 .ТАВ-файл, но не удаляет файл с исходным растровым изображением.
- При использовании оператора **Rename Table** для растровой таблицы MapInfo переименует .ТАВ-файл, но не изменит название файла с исходным растровым изображением.
- При использовании оператора **Commit** для копирования растровой таблицы MapInfo скопирует только .ТАВ-файл, но не файл с исходным растровым изображением.

ТАВ-файл для растровых таблиц создается в тот момент, когда пользователь заполняет поля диалога "Регистрация изображения". Чтобы создать такой .ТАВ-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 .ТАВ-файл для растровой таблицы.

Внимание: 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" 'название
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 WindowsLatin1"
    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" ' Brightness;
default is 50
    Print #1, "  RasterStyle 2 60" ' Contrast; default
is 50
    Close File #1
    register_nonmap_image = TRUE ' установить
значение, которое вернет      ' функция
last_exit:
    Exit Function
handler:
    Close File #1
    Resume last_exit

```

Работа с метаданными

Что такое метаданные?

Метаданные – это данные, хранящиеся в файле .ТАВ 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 .ТАВ-файл в текстовом редакторе или не запускают программу-пример TABLEMGR). Однако MapBasic-приложение 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.

Имя ключа	Значение ключа (пример)
”\Copyright Notice”	”Copyright 1995 Горбунков.”
”\Info”	”Стамбул - город контрастов”
”\Info\Author”	”Семен Семеныч Горбунков”
”\Info\Date\Start”	”12/14/95”
”\Info\Date\End”	”12/31/95”
”\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 таблице выше 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. 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 `GetMetadata$()` 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 = GetMetadata$(table_name, "\Copyright")
```

Оператор **Metadata** 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 **Metadata**. **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.

```
Metadata Table table_name
      SetKey "\Info\Author" To "Семен Семеныч"
```

В следующем примере ключ удаляется.

```
Metadata Table table_name
      Dropkey "\Info\Author"
```

Здесь удаляется целая иерархия ключей. Все ключи, имена которых начинаются с "\Info\ ", будут удалены.

```
Metadata Table table_name
Dropkey "\Info" Hierarchical
```

Когда Вы используете оператор **Metadata**, чтобы записывать или удалять метаданные, изменения происходят немедленно. Вы не должны выполнять операцию **Save**.

Вы также можете использовать оператор **Metadata**, чтобы читать метаданные из таблицы, даже если Вы не знаете имена ключей. Для этого нужно:

1. Использовать оператор **Metadata Table ... SetTraverse** для инициализации просмотра имен ключей.
2. Использовать оператор **Metadata Table ... Next**, чтобы получить значение следующего ключа. Оператор возвращает имя ключа в строковой переменной. Обычно это следует повторять в цикле. Если ключа нет, оператор вернет пустое значение в качестве его имени.
3. Завершить поиск оператором **Metadata Traverse ... Destroy**.

Пример:

```
Sub Print_Metadata(ByVal table_name As String)

    Dim i_traversal As Integer
    Dim s_keyname, s_keyvalue As String

    ' Инициализация просмотра ключей. Укажите "\" как
    ' начальный ключ, просмотр начнется с первого
    ключа
    Metadata Table table_name
        SetTraverse "\" Hierarchical Into ID
    i_traversal
    ' Попытка доступа к первому ключу:
    Metadata Traverse i_traversal
        Next Into Key s_keyname Into Value s_keyvalue

    ' цикл по всем ключам
    Do While s_keyname <> ""
        Print " "
        Print "Имя ключа: " & s_keyname
        Print "Значение: " & s_keyvalue
```

```

Metadata Traverse i_traversal
Next Into Key s_keyname Into Value s_keyvalue
Loop

' Закончим и освободим память:
MetaData Traverse i_traversal Destroy

End Sub

```

Полный синтаксис оператора **Metadata** приведен в *Справочнике MapBasic* и в Справочной системе MapBasic.

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”.

Для введения в работу со сшитыми таблицами, см. *Руководство пользователя MapInfo*.

Как работать со сшитыми таблицами?

Комплект поставки MapInfo содержит программу MapBasic “Сшитые таблицы” (SEAMMGR.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, HIGHWAY.
2. Запустить приложение “Сшитые таблицы”.
3. Выполнить команду ПРОГРАММЫ > СШИТЫЕ ТАБЛИЦЫ > ОТКЛЮЧИТЬ “СШИТОСТЬ”, чтобы отключить соответствующий атрибут для таблицы HIGHWAY.TAB
4. Выполнить команду ОКНО > НОВОЕ ОКНО Окна просмотра, чтобы отобразить таблицу в окне Списка.

Как и обычная таблица, сшитая содержит строки и колонки. Каждая строка соответствует включенной в нее таблице



Первый столбец в таблице содержит имена таблиц. Второй содержит описания, которые появляются в интерфейсе пользователя. Имена таблиц в первом столбце могут включать путь к каталогу. Вы можете опускать путь, 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 минимальное прямоугольное покрытие (МПП). Когда пользователь отображает сшитую таблицу в окне Карты, MapInfo сравнивает текущие размеры окна Карты с МПП. MapInfo открывает только те основные таблицы, для которых МПП 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 DCMetroA 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 данным

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 таблицам в сетях типа Oracle или Sybase.

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 при помощи команд

MapInfo позволяет приложениям MapBasic соединяться со многими базами данных в одно и то же время и выдавать 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 дескриптор оператора как переменную 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 с последующими SQL-DataLink обращениями.

```
Dim hdbc As Integer  
hdbc = Server_Connect("ODBC", "DLG=1")
```

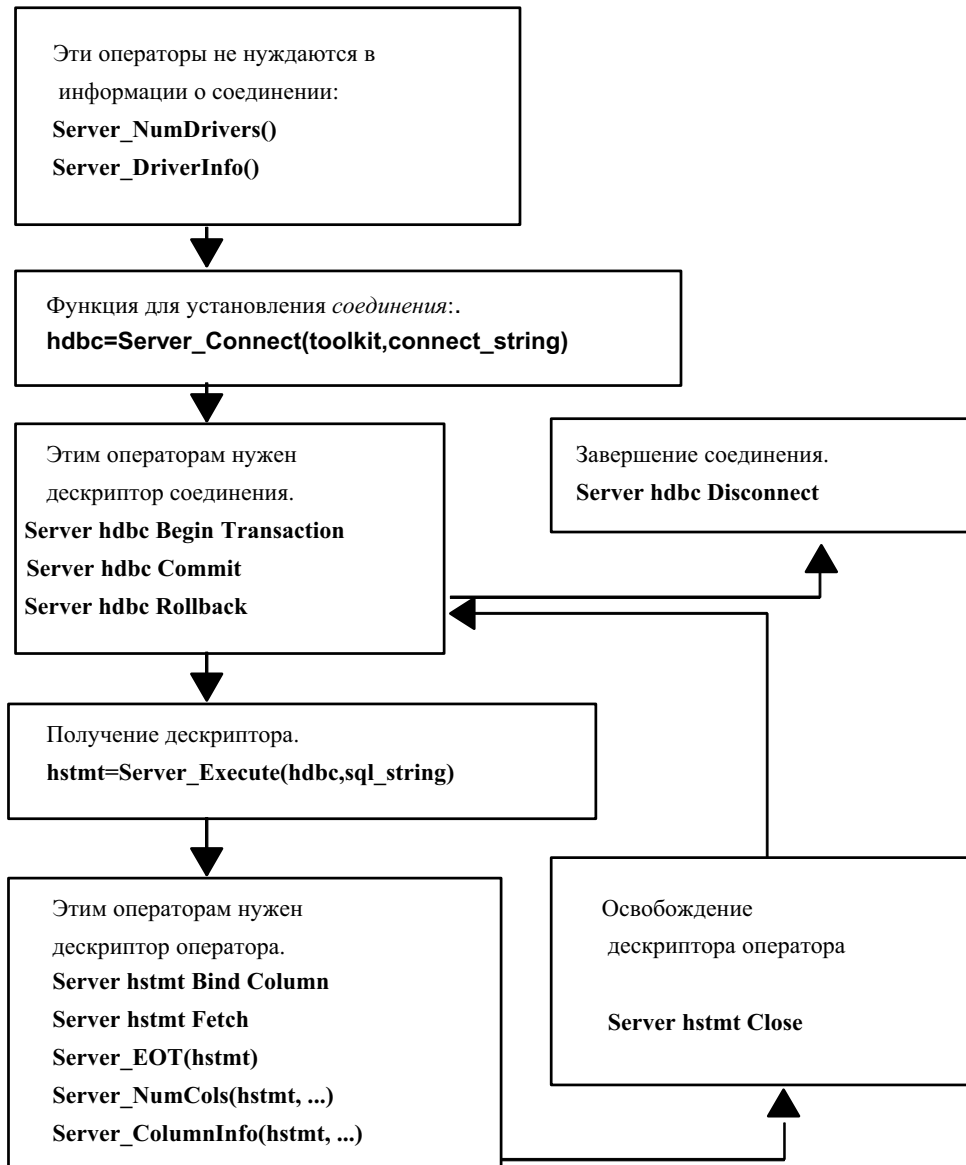
Когда драйвер выполнит фиксацию (**commit**) или откат (**rollback**), сбрасываются все операторные запросы, связанные с этим соединением. Диспетчер драйвера производит работу, связанную с переключением соединений, в то время как происходит обработка транзакции для текущего соединения.

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**).



Вы можете загрузить (**download**) из ODBC-источника данных всю таблицу, некоторые строки и столбцы, или *результатирующее множество*, используя предложение **Into** оператора MapBasic **Server Fetch**. Однако никакие изменения в загруженной таблице не повлияют на таблицу базы данных сервера. Модификация удаленных баз данных выполняется при помощи создания связанных таблиц.

Доступ и изменения в удаленных базах данных 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. Целостность данных обеспечивается механизмом транзакций; другие пользователи могут модифицировать те же самые строки в тех же самых таблицах. 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**. Отсоединение таблицы удаляет связь с удаленной базой данных. В результате получается обычная таблица MapInfo.

При использовании *пространственной индексации* MapInfo пользователи могут сохранять и отыскивать точки в любой базе данных. См. Приложение Е.

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*).

Рекомендации по повышению производительности при работе с таблицами

Минимизация количества транзакций

Обычно, когда пользователь редактирует таблицу MapInfo, 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*. См. там же описание оператора **Set Table ... Undo Off**.

Разумное использование индексов

Некоторые запросы работают быстрее, если Вы проиндексируете одну или большее количество колонок в Вашей 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 Вашей 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 быстроедействие, 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 это быстрее выполняются два не вложенных оператора **Select** вместо одного вложенного **Select ... Where**.

Если Вы выполняете подвыборку этого типа:

```
... Where x = Any( Select ... ) ...
```

то MapInfo оптимизирует производительность, но только если колонка **x** проиндексирована.

Оптимизация операторов Select

Некоторые типы запросов **Select** оптимизированы для повышения эффективности. См. оператор **Select** в *Справочнике MapBasic* или интерактивной справке.

Применение оператора Update

MapBasic позволяет обновлять объекты карты по одному за раз, используя оператор **Alter Object** и затем используя оператор **Update** для отдельных записей, часто в цикле. Таким образом, этот тип преобразования таблицы 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 *Справке 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.



8

Глава

- > Обзор файлового ввода/вывода
 - > Файлы последовательного доступа
 - > Особенности работы с файлами в различных операционных системах и с национальными наборами символов
-

Обзор файлового ввода/вывода

Файловый ввод/вывод (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 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, чем это возможно в текстовых файлах.

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 с последовательным доступом для чтения:

```
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 возможностей для ошибок времени выполнения, связанных 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). Подробно об обработке исключительных (ошибочных) ситуаций мы говорили в главе 5.

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 в предложении **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 #**. **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 NIEWS.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. Приложение NIEWS.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 **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.

Замечание: Чтение последней строки файла не означает, что будет установлен признак "конец файла". **EOF()** возвращает **TRUE** только после того, как Вы попытаетесь в первый раз считать строку за концом 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. Так, в приложении NIEWS.MB из набора примеров оператор **Write #** используется (в цикле) для записи в файл четырех значений (названия, 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**:

```
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.

При чтении двоичного файла, созданного на другой вычислительной платформе, у Вас могут возникнуть проблемы. Связано это с тем, что в разных компьютерах соблюдается различный порядок хранения байтов в двоичных файлах. Так, в DOS значения, состоящие из нескольких байт, хранятся в порядке от младшего байта к старшему, а в Macintosh и UNIX-компьютерах – в обратном порядке.

Из-за этого при работе с файлом, созданным приложением на Macintosh, в среде MapInfo для Windows можно получить неверные результаты. В подобной ситуации следует использовать предложение **ByteOrder** в операторе **Open File**. Предложение **ByteOrder** позволяет явно указать, какой порядок хранения байтов применен в том или ином файле.

```
Open File "datafile.dat" For Binary As #1 ByteOrder
LowHigh
```

Если предложение **ByteOrder** отсутствует, MapBasic полагает, что используется стандартное представление для той вычислительной платформы, на которой он установлен. Так, в MapInfo для Windows, чтобы читать файлы для Macintosh, следует указать **ByteOrder HighLow**, чтобы дать сигнал MapInfo, что файл использует порядок байт от старшего к младшему.

При создании приложений, использующих двоичные файлы и способных работать и под Windows, и под UNIX, и на Macintosh, может понадобиться явно указывать **ByteOrder**. Если предложение **ByteOrder** будет указано во всех операторах **Open File**, причем **ByteOrder** будет везде одинаковым (скажем, **LowHigh**), Ваше приложение будет без ошибок работать на всех платформах.

Если Вы помещаете в двоичный файл текстовые строки, то эти строки должны иметь фиксированную длину.

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** необязательное предложение **CharSet**.

В компьютере каждому символу 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 ° (градус); 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.

Более того, различные вычислительные платформы могут использовать различные наборы символов. В Windows код символа 176 соответствует знаку градуса, а на Macintosh код 176 соответствует другому символу. Такое несоответствие также может быть причиной проблем в использовании программ на различных платформах.

Если нет дополнительных указаний, то при чтении в программе на MapBasic текстового файла предполагается, что этот файл использует стандартный набор символов для той платформы, на которой выполняется программа. Например, если пользователь работает с MapInfo для Windows, причем для Windows установлен набор символов для Северной Америки и Западной Европы, то MapInfo считает, что открываемые текстовые файлы также используют набор символов Windows для Северной Америки и Западной Европы. Если же окажется, что текстовый файл создан для другого языка или на UNIX или Macintosh, то результаты чтения из такого файла могут быть некорректными.

Чтобы устранить подобное несоответствие, указывайте параметр **CharSet** 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 гарантирует корректное чтение из файла (запись в файл).

Например, чтобы приложение под MapInfo для Windows правильно прочитывало данные из файла, созданного на Macintosh, следует выполнить оператор вида:

```
Open File "settings.txt" As #1 CharSet "MacRoman"
```

Список имен наборов символов, которые могут быть указаны в предложении **CharSet**, приводится в главе **CharSet** в Справочнике MapBasic.

Функции, имеющие дело с режимами файлового ввода/вывода

С помощью следующих функций можно получить информацию о режиме, который назначен для ввода/вывода любому открытому файлу:

FileAttr() возвращает код режима доступа к файлу (INPUT, OUTPUT, 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.

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 главу 7, если Вы до сих пор этого не сделали.

9

Глава

- >Переменные типа Object
 - >Работа с колонкой “Obj”
 - >Определение атрибутов объекта
 - >Создание новых объектов
 - >Создание новых объектов на основе уже существующих
 - >Изменение объектов
 - >Работа с подписями
 - >Координаты и единицы измерения
 - >Географические запросы
-

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). С переменными 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 дорог штата California (в милях):

```
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 сначала нельзя увидеть в окне Карты. Чтобы отобразить такую таблицу на Карте, следует выполнить оператор **Create Map**, который добавляет к таблице колонку графических объектов (Object).

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 не колонки Obj, а значений из нее, используйте оператор **Delete Object**.

Чтобы определить, имеется ли в таблице колонка графических объектов, вызовите функцию **TableInfo()** с параметром **TAB_INFO_MAPPABLE**.

Ограничения, накладываемые на колонку Object

На колонки графических объектов накладываются некоторые особые ограничения. В частности, в каждой таблице может быть только одна колонка графических объектов. При создании выборки с объединением двух таблиц, обе из которых имеют колонку Object, результирующая таблица содержит только один из графических объектов (объект 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 предприятия), 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 описанный 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 **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 (flood – потоп, затопление, наводнение):

```
Dim floodarea As Float
Open Table "floodmap"
Fetch First From floodmap
floodarea = Area(floodmap.obj, "sq km")
```

Внимание: Обратите внимание, что подписи это не то же что и текстовые объекты. Запрашивая текстовый объект, Вы вызываете функции типа **ObjectInfo()**. Запрашивая подпись, Вы вызываете такие функции, как **Labelinfo()**.

Стили объектов (Pen, Brush, Symbol, Font)

Каждый графический объект имеет один или несколько параметров, условно называемых стилем. Например, каждая линия имеет стиль **Pen** ("перо"), 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 данной таблице приведены определения четырех стилей объектов.

Pen	толщина, тип и цвет линии
Brush	шаблон, основной цвет и цвет фона для внутренней части области
Font	название шрифта, тип, размер, цвет, фон; для текстовых объектов

Symbol	<p>В MapInfo 3.0: вид, цвет, размер.</p> <p>Для символов из шрифтов TrueType: вид, цвет, размер, имя шрифта, стиль (например, курсив), параметры поворота.</p> <p>Для пользовательских символов, основанных на файлах с растровыми картинками: имя файла, цвет, размер и атрибуты стиля.</p>
--------	--

Для получения подробной информации о стилях обратитесь к разделам **Brush clause**, **Font clause**, **Pen clause** и **Symbol clause** в *Справочнике MapBasic* и интерактивной справке.

Язык MapBasic имеет различные операторы и функции для создания объектов (таких как оператор **Create Text**, функция **CreateLine()**). 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.

Внимание: 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 "Два шаблона заполнения 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 Baltica), стиль текста (например полужирный, курсивный, и т.д.) и цвет. Стиль шрифта также содержит информацию о размере; однако, размер иногда игнорируется. Следующий список суммирует, как размер шрифта **Font** 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, MapInfo игнорирует размер шрифта. В этой ситуации высота текста определяется координатами карты, которые Вы устанавливаете в операторе **Create Text**. Когда Вы выполняете 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 большими, если Вы увеличиваете, и меньшими, если Вы уменьшаете окно Карты.
- Когда Вы используете 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 Вы создаете 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. Подписи обсуждены позже в этой главе.

Начертание шрифта включает имя шрифта, типа Courier или Helvetica. Имена шрифтов могут отличаться на различных аппаратных платформах; например, Geneva – общее имя шрифта на Macintosh, но не на других платформах. Helv и TmsRmn (или Times New Roman) в среде Microsoft Windows называются Helvetica и Times на платформах Sun и Macintosh. Helvetica, Times и Courier распознаются в предложении MapBasic **Font** независимо от текущей платформы.

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 SCALEBAR из набора примеров **MakeBrush()** используется для построения штриховок из черного и белого цветов, которые используются для рисования шкалы.
- Вызвать **CurrentPen()**, **CurrentBrush()**, **CurrentSymbol()** или **CurrentFont()** 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. Подробно диалоги описаны в главе 6.

Следующий пример демонстрирует, как создать стиль Pen с помощью обращения к функции **MakePen()**. Значение стиля **Pen** 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 Справочнике MapBasic.

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, Вы можете 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 **ТЕХТВОХ** из набора примеров показывается диалог, в котором пользователь может выбрать стиль линии; выбранный стиль сохраняется в переменную типа Pen, с именем "pstyle". Затем в программе **ТЕХТВОХ** 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 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, основанные на стилях. Как показывает следующий пример, предложение **Where** оператора **Select** может вызывать функции **ObjectInfo()** и **StyleAttr()**, чтобы MapInfo выбрал только те объекты, которые имеют некоторые атрибуты (например, объекты некоторого цвета).

Следующий пример добавляет кнопку на инструментальную панель “Программы”. Если Вы выбираете объект и затем нажимаете эту кнопку, программа выбирает все объекты в таблице, которые имеют тот же самый цвет.

```

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

    ' Сколько таблиц 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 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

' Если открылась таблица запроса Запрос1 (или
Запрос2...).
' закроем ее.
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

' Выберем все точечные объекты.
Select * From table_name
Where Str$(Obj) = "Point"
Into Color_Query_Prep NoSelect

' Отберем с нужным цветом.
Select * From Color_Query_Prep
Where
    StyleAttr(Object-
Info(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 объектами-областями вместо точек, Вы проверяли бы для имени объектного "Region" вместо "Point", и Вы вызывали **ObjectInfo()** с 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, за исключением **Create Frame**, в окнах Карт.

- **Create Arc**: создает дугу.
- **Create Ellipse**: создает эллипс или окружность. (Окружность – это просто частный случай дуги, дуга с одинаковыми радиусами.)
- **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.
- **Create Line**: создает линию.
- **Create Point**: создает точку.
- **Create Pline**: создает ломаную.
- **Create Rect**: создает прямоугольник.
- **Create Region**: создает область (многоугольник).
- **Create RoundRect**: создает прямоугольник.
- **Create Text**: создает текстовый объект.
- **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 типа Object:

- **CreateCircle()**: возвращает окружность.
- **CreateLine()**: возвращает линию.
- **CreatePoint()**: возвращает точечный объект.
- **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 колонки "lon" со значениями долготы (координата X) и "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).

Вы можете 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:

Выполнить оператор **Create Region** или **Create Pline**, чтобы создать пустой графический объект (объект без узлов).

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,objco-
ord(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 колонку "Obj" 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**. Программа TEXTBOX 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 буфера

Буферная область (или просто "буфер") – это область, охватывающая все возможные объекты, удаленные от заданного графического объекта не более, чем на некоторое расстояние. 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 две области 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.

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

Чтобы изменить значения координат графического объекта, выполните оператор **Alter Object** с предложением **Geography**. Вам может потребоваться выполнить более одного оператора **Alter Object** (например, один — чтобы изменить координату X, и еще один — чтобы изменить координату Y).

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**).

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.

- Оператор **Objects Erase** удаляет часть графического объекта (объектов), помеченного как "изменяемый", при этом удаляются выбранные объекты.
- Оператор **Objects Intersect** удаляет те фрагменты графического объекта (объектов), помеченного как "изменяемый", которые не являются выбранными в текущий момент.

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 > ВЫБРАТЬ ИЗМЕНЯЕМЫЙ ОБЪЕКТ** или оператором **Set Target** языка MapBasic. Принципы редактирования графических объектов описаны в Справочнике MapInfo.

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()**.

Работа с подписями

В более ранних версиях MapInfo, подписи на Карте создавались оператором **AutoLabel**, который создает текстовые объекты в Косметическом слое. В MapInfo 4.0 концепция подписывания была изменена: подпись теперь обрабатывается как атрибут элемента изображения объекта Карты, а не как отдельный текстовый объект. Однако, если Вы написали приложения, использующие MapBasic 3.x, Ваши приложения будут выполняться; MapInfo 4.0 поддерживает оператор **AutoLabel** для совместимости.

Показ подписей

Пользователь MapInfo может настраивать режимы размещения подписей через диалоговое окно "Управление Слоями". Программа MapBasic может выполнять те же самые операции оператором **Set Map ... Label**.

```
Set Map Layer 1 Label Auto On Visibility On
```


Скрытие подписей

В диалоге “Управление Слоями” сброс флажка подписывания в списке слоев выключает заданные по умолчанию подписи для этого слоя.

Следующий оператор MapBasic имеет тот же самый эффект:

```
Set Map Layer 1 Label Auto Off
```

Внимание: 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 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 Label Default
```

Редактирование подписей

Пользователь MapInfo 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**.

Внимание: 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 меток. Если удаленные строки в таблице находятся в самом конце таблицы (то есть внизу окна Окна Списка), то все будет в порядке.

Запрос подписей

Запрос к подписям в окне карты состоит из двух шагов:

1. Инициализация внутреннего указателя подписей MapBasic с помощью вызова **LabelFindFirst()**, **LabelFindByID()** и **LabelFindNext()**.
2. Вызова **Labelinfo()** для запроса “текущей” подписи.

Пример кода смотрите в разделе **Labelinfo()** электронной справки MapBasic или в программе-примере LABELER.MB.

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 нужные параметры и нажмите ОК.

MapInfo 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 (строки с оператором **Set Map ... Layer ... Label ... Object**).

Разница между подписями и текстовыми объектами

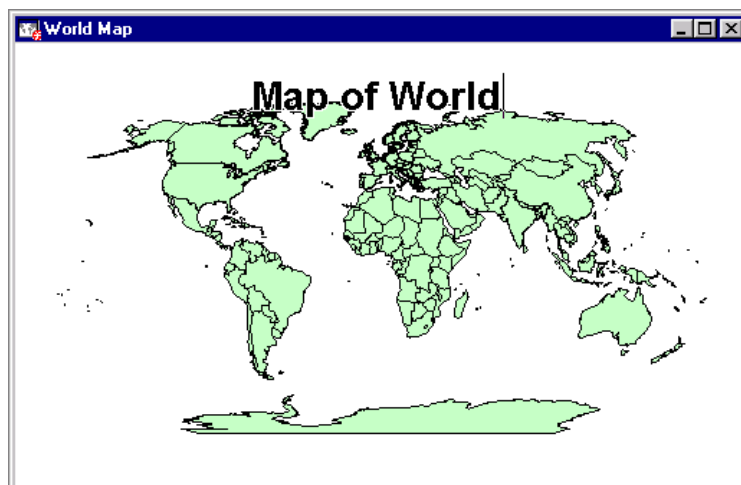
Следующая таблица показывает различия между текстовыми объектами и подписями.

	Текстовый объект	Подпись
Оператор MapBasic, используемый, чтобы создать текст:	AutoLabel, Create Text, CreateText()	Set Map
Оператор MapBasic, используемый, чтобы изменить текст:	Alter Object	Set Map
Функции MapBasic, используемые, чтобы сделать запрос о тексте (например, определить цвет):	ObjectInfo(), ObjectGeography()	LabelFindByID(), LabelFindFirst(), LabelFindNext(), Labelinfo()
Оператор MapBasic, используемый, чтобы выбрать текст:	Select	Программы MapBasic не могут выбирать подписи.
Сохранение текста в Карте:	Текстовые объекты могут быть сохранены в “картографических” таблицах.	Подписи могут быть сохранены только в Рабочих Наборах.
Сохранение текста в Отчете:	Текстовые объекты, созданные в окне Отчета могут быть сохранены в Рабочем Наборе.	Не применимо. Подписи не могут появляться в Отчете (за исключением того, когда в Отчете находится Карта).
Управление высотой текста из MapBasic:	При создании текстового объекта в “картографической” таблице, Вы указываете координаты x и y, чтобы определить прямоугольную область; текст заполняет ее. Высота текста изменяется при изменении размера окна.	Высота текста подписи контролируется шрифтом. Масштабирование на нее не влияет.
Конвертация между текстом и подписями:	Не применима. Данный текстовый объект не изменить, нет функции в MapBasic чтобы вернуть из него Подпись.	Если есть подпись, то функция Labelinfo() может вернуть текстовый объект, соответствующий подписи. Смотрите пример в LABELER.MBX.

Когда Вы создаете подпись, Вы не должны определять прямоугольную область. Вместо этого 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 подписи
  Anchor (0, 85)         'координаты (x,y)
  Text "Map of World"    'текст
  Position Center        'положение
  Font("Arial",289,20,0) 'стиль подписи (20 пунктов
                        и т.д.)
```

Возникающая в результате подпись выглядит так:.



Если Вы должны поместить текст в Вашу Карту, Вы можете обнаружить, что проще создать подписи, чем текстовые объекты. Вы можете создать таблицу с единственной целью использовать подписи. Для этого:

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, **Auto On**, 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.)

Внимание: 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 Nonearth**.
- 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 карт мира, 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
  "Title Goes Here"
  (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 центроида 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 центроида штата Wyoming:

```
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

В программах на MapBasic используются следующие единицы измерения:

- Единицы площади, такие, как 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 Distance 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. Для этого Вы сначала должны ознакомиться с главой 7 ("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-параметров внутри второго.

Contains используется для проверки, содержит ли первый из объектов-параметров второй объект внутри себя.

Например, для случая сравнения точки и области:

1. Точка **Within** (внутри) области.
2. Область **Contains** (содержит) 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 центрост объекта внутри другого объекта. Чтобы проверить, лежит ли весь объект целиком внутри другого объекта, следует употреблять предложение **Entirely**. Предложение **Partly** 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** Вы можете 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 центростид, 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	Синтаксис	Возвращает TRUE, если:
Contains	objectA Contains objectB	Объект А содержит центроид объекта В
Contains Part	objectA Contains Part objectB	Объект А содержит некоторую часть объекта В
Contains Entire	objectA Contains Entire objectB	Объект А содержит весь объект В
Within	objectA Within objectB	Центроид объекта А лежит внутри объекта В
Partly Within	objectA Partly Within objectB	Часть объекта А лежит внутри объекта В
Entirely Within	objectA Entirely Within objectB	Объект А находится полностью внутри объекта В
Intersects	objectA Intersects objectB	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 Indiana:

```
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 Marion штата Indiana. 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. Это будет выглядеть 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 Indiana. MapInfo 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 ("mycounty"), 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 Nebraska:

```
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 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 риска затопления (в виде областей разной степени риска). Некоторые города частично или полностью попадают в зоны риска, другие же полностью лежат вне этой зоны. С помощью оператора **Add Column** можно выделить демографическую информацию из таблицы городов, а затем на ее основе просчитать статистику для городов, попадающих в зону риска затопления.

Подробное описание оператора **Add Column** см. в Справочнике MapBasic.

Особенности MapBasic в среде MS Windows

В этой главе описывается, как MapBasic может использовать возможности, предоставляемые средой Windows. В этой главе:

- Объявление и использование динамических библиотек (DLL)
- Создание новых курсоров и пиктограмм на кнопках
- Связь между программами через механизм динамического обмена данными (DDE)
- Добавление собственной Справочной системы к приложению

10

Глава

- > **Объявление и использование динамических библиотек (DLL)**
 - > **Создание пиктограмм на кнопках и новых курсоров**
 - > **Связь между приложениями с использованием DDE**
 - > **Добавление Справочной системы к Вашему приложению**
-

Объявление и использование динамических библиотек (DLL)

Динамические библиотеки Windows (Dynamic Link Library, DLL) – это файлы, содержащие выполняемые процедуры и другие ресурсы. Вы можете использовать DLL-файл как библиотеку внешних процедур и обращаться к ним из программы MapBasic с помощью оператора **Call**. Многие DLL-библиотеки поставляются вместе с программами, а также отдельно. Каждая обычно снабжается документацией, описывающей процедуры и их параметры. Для того, чтобы можно было вызывать DLL-процедуру из Вашей MapBasic-программы, Вы должны объявить ее оператором **Declare** так же, как и собственные процедуры. В операторе **Declare** Вы объявляете имя DLL-файла, имя процедуры и описываете ее параметры. Предложение **Alias** позволяет, если нужно, переименовать DLL-процедуру для избежания конфликтов с уже существующими в программе именами.

Вы можете вызвать DLL-функцию так же, как и собственную функцию MapBasic, объявив ее сначала с помощью оператора **Declare Function** и затем обращаясь к ней из текста. (См. в *Справочнике MapBasic* описание операторов **Declare Sub** и **Declare Function**.) DLL-файл должен быть доступен программе во время ее работы.

Подробно строение DLL-файлов описано в документации к *Windows Software Developer's Kit (SDK)*, а также в книгах независимых авторов.

Объявление внешней библиотеки

Предложение **Lib libname** в операторе **Declare** сообщает MapBasic, где находится динамическая библиотека "libname". Среда Windows содержит несколько стандартных библиотек, входящих в состав системы. Вы можете в качестве параметра *libname* объявить "User", "GDI", "Kernel" или любую из стандартных DLL-библиотек Windows. При объявлении других DLL-библиотек параметр *libname* должен включать в себя DOS-маршрут:

```
Declare Sub mydll Lib "C:\dlls\mydll.dll" (ByVal x
    As Integer,
    ByVal y As Integer)
```

Если явно задать DOS-маршрут в операторе **Declare** (например, "C:\LIB\MYLIB.DLL"), MapInfo постарается загрузить именно этот DLL-файл. Если этого файла там нет, то MapInfo его не загрузит, что повлечет ошибку.

Если оператор **Declare** не задает явного пути к DLL-библиотеке, то MapInfo будет искать его по следующим правилам:

1. Если DLL находится в том же каталоге, что и MBX-файл, то MapInfo загружает DLL, иначе

2. Если DLL находится в том же каталоге, что и MapInfo, то MapInfo загружает DLL, иначе
3. Если DLL находится в каталоге WINDOWS\SYSTEM, то MapInfo загружает DLL, иначе
4. Если DLL находится в каталоге WINDOWS, то MapInfo загружает DLL, иначе
5. MapInfo проводит поиск по каталогам, заданным в системной переменной PATH.

Поиск растровых файлов для иконок и курсоров MapInfo проводит по тем же правилам.

Передача параметров

Многие DLL-процедуры требуют передачи параметров, как в приведенном выше примере, 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 об этих способах можно прочитать в главе 4.

Ключевое слово **ByVal** задает передачу параметра значением (by value). Вы должны явно задать слово **ByVal** в операторе **Declare**, если нужно передать соответствующий параметр значением, иначе он будет передан ссылкой.

Передача параметра ссылкой (by reference) в процедуру, требующую передачу значением, может привести к ошибке в работе процедуры или возвращению неверных значений.

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 переменные”.

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 “Beep” для имени процедуры “MessageBeep” из библиотеки “User”:

```
Declare Sub Beep Lib "user" Alias "MessageBeep"  
  (ByVal x As SmallInt)  
Call Beeper(1)
```

Если Вы задаете DLL-процедуре имя-синоним, 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 в C-программе как **char ***.

Внимание: 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 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.

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.

Передача 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 **Type**. MapBasic передает адрес первого элемента, подразумевая, что остальные элементы последовательно размещены в памяти. Для того, чтобы DLL-процедура работала со сложными типами, ее нужно компилировать с заданием упаковки структур (“structure packing”).

Логические переменные в качестве аргумента

Если передается логическое значение из MapBasic в DLL, то аргумент передается в виде двухбайтового значения.

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. Если нужно передать уникальный номер как аргумент, то его нужно объявить как **ByVal Integer**. DLL-функции, возвращающие уникальные номера, должны быть объявлены **Integer**-функциями. Уникальные внутренние номера можно использовать только как идентификаторы; участвовать в каких-либо вычислениях они не могут.

Пример: Вызов процедуры из библиотеки “KERNEL”

Следующий пример содержит вызов DLL-функции из стандартной Windows-библиотеки “KERNEL”. Вызываемая процедура считывает установки из стартового файла Windows, WIN.INI.

```
Declare Sub Main
```

```
' Сначала 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)

    ' теперь переменная sReturn содержит имя страны
    ' (например, "United States")
    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. Так как эта библиотека является стандартной компонентой системы, Windows в этом случае сам подставляет нужный файл. Однако, если Вы создаете свою DLL-библиотеку, Вы должны в операторе **Declare Function** точно задавать имя файла библиотеки.

32-разрядные DLL

DLL написанные для Windows95 и Windows NT, имеют 32-разрядную архитектуру.

Ограничения:

Если Ваши пользователи работают на 32-разрядной версии MapInfo, любая DLL, используемая вашей MapBasic-программой, должна быть 32-разрядной.

Эти ограничения остаются в силе, даже если Вы используете DLL только для хранения иконок инструментальных панелей.

MBX-файлы не имеют "16-разрядной" или "32-разрядной" архитектуры, т.е. два таких файла совершенно одинаковы.

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, передача типа double). Может оказаться более надежным передавать аргумент ссылкой; в этом случае передается адрес и меньше оснований ожидать неожиданностей.
- Следует придерживаться следующего правила: каждая функция из DLL сама выделяет для себя и освобождает память.
- Важно, чтобы объявление **Declare** было корректным, передаваемые параметры должны быть описаны в соответствии с “устройством” вызываемой из DLL функции (в частности,

следует обратить внимание на передачу аргументов ссылкой или значением).

Создание пиктограмм на кнопках и новых курсоров

Средства языка MapBasic позволяют управлять инструментальными панелями MapInfo. Подробно об этом можно прочитать в главе 6. 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; в среде Macintosh пиктограммы помещаются в PICT-ресурсах; в среде UNIX пиктограммы помещаются в XPM-файлах. В этой главе описано, как создавать пиктограммы в MapInfo для Windows. Программа MapBasic может также создать свой *курсор* (изображение указателя мыши для окна Карты или Отчета, например). В этом разделе описано, как создавать курсоры в MapInfo для Windows.

Использование стандартных пиктограмм (иконки)

Прежде чем заниматься созданием новых пиктограмм, познакомимся поближе со встроенными в MapInfo. Начиная с версии 4.0, MapInfo содержит большое количество различных пиктограмм, что должно облегчить Вам создание своих инструментальных панелей.

Чтобы посмотреть на встроенные пиктограммы, запустите программу Icon Sampler (ICONDEMO.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.

Создание пиктограмм для Windows

Для создания пиктограмм для MapInfo для Windows 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 файле 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 картинки. Первая картинка должна быть размером 18 пикселей в ширину на 16 пикселей в высоту; эта картинка предназначена для показа на экране с относительно низким разрешением. Вторая картинка имеет размеры 26 на 24 пикселей; она предназначена для экранов с высоким разрешением (например, 1024 x 768). Вы должны обязательно создать обе картинки, даже если одна Вам не нужна. В MapInfo можно включать режим “Большие кнопки” в диалоге команды [НАСТРОЙКИ > ИНСТРУМЕНТАЛЬНЫЕ ПАНЕЛИ](#).

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 картинки (BMP-ресурс): одну размером 18 пикселей в ширину на 16 пикселей в высоту и другую размером 26 на 24 пикселей.
- Не забывайте, что создается ресурс типа “курсор”, а не “иконка”.
- 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.

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.

В диалоге одно приложение активно; это начинает диалог. 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.

DDEInitiate()	Открывает сеанс обмена;
DDERequest\$()	Запрашивает информацию у сервера;
DDEPoke	Посылает информацию серверу;
DDEExecute	Посылает серверу команды;
DDETerminate DDETerminateAll	Закрывает один или все сеансы DDE-обмена.

См. также описание этих операторов и функций в *Справочнике MapBasic*.

Для того, 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. Обычно, 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. Послание данных из приложения MapBasic через DDE-связь имитирует ввод данных пользователем в соответствующий документ программы-сервера. В следующем примере приложение MapBasic помещает фразу “Привет, друзья!” в ячейку электронной таблицы.

```
DDEPoke channelnum, "R1C2", "Привет, друзья!"
```

При вызове функции **DDEPoke** нужно точно задавать имя элемента, принимающего данные (в примере это R1C2, т.е. Строка 1, Столбец 2). 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, устанавливается в WIN.INI – стартовом файле Windows. MapInfo создает раздел [MAPINFO] в WIN.INI и задает время задержки в строке DDETimeout:

```
DDETimeout=10000
```

Число представляет собой количество миллисекунд; стандартное значение времени задержки равно десяти тысячам (десять секунд). Если приложение-клиент часто исчерпывает время ожидания в сеансах DDE-связи, и порождается ошибка, Вы можете увеличить значение задержки.

MapInfo в роли DDE-сервера

MapInfo 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. Однако, управляющие операторы MapBasic не могут быть выполнены MapInfo как сервером.

Операторы и функции поддержки DDE-связи могут в других программах называться по-другому. Вы должны изучить документацию каждой из программ, чтобы можно было наладить DDE-связь.

Программы и приложения, выступающие в роли DDE-клиентов по отношению к MapBasic, должны задавать параметры **application**, **topic** и **item**, значения которых описаны выше.

Чтобы начать сеанс DDE-связи с MapInfo как с сервером, нужно задать "MapInfo" как значение параметра **application**.

В качестве параметра **topic** можно задать "System" или имя действующего в настоящий момент приложения MapBasic (например, "C:\MAPBASIC\GRIDS.MBX"). При этом нужно точно задавать строчные и прописные буквы.

Параметр **item** зависит от темы (**topic**). В следующей таблице приведены различные сочетания объектов и элементов, которые можно использовать, если MapInfo выступает в роли DDE-сервера.

Application:"MapInfo"

Topic:"System"

Команды и темы, поддерживаемые в сеансе обмена DDE:

Действие	Параметр item	Результат
Запрос на считывание данных	"SysItems"	На запрос клиента с именем элемента Sysitems Map-Info возвращает разделенный табуляторами список элементов объекта System: Sysitems Topics Formats
Запрос на считывание данных	"Topics"	На запрос клиента с именем элемента Topics Map-Info возвращает разделенный табуляторами список доступных объектов (System и названия всех приложений MapBasic, действующих или пребывающих в режиме ожидания).

Запрос на считывание данных	"Formats"	На запрос клиента с именем элемента Formats MapInfo возвращает список всех поддерживаемых MapInfo форматов обмена через Clipboard (TEXT и другие).
Запрос на считывание данных	"Version"	MapInfo возвращает текстовую строку, представляющую номер версии MapInfo, умноженный 100. Например, MapInfo 4.0.0 возвратит "400". См. пример ниже.
Запрос на считывание данных	Выражение MapBasic	MapInfo интерпретирует строку как выражение MapBasic и возвращает значение в виде строки. Выражение "If" недопустимо, MapInfo возвращает ошибку. Эта функциональная возможность появилась в MapInfo 4.0.
Execute	Текстовое послание	MapInfo 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 Dim в окне MapBasic.

Например, следующая MapBasic-программа, которую Вы можете напечатать непосредственно в окне MapBasic, проводит простой DDE-диалог, используя "MapInfo" как **application** и "System" как **topic**.

```
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 **item**. Если Вы используете имя приложения MapBasic (например, "C:\MB\SCALEBAR.MBX" или "SCALEBAR.MBX" или "SCALEBAR") как DDE-topic, Вы можете использовать следующие имена в качестве **item**:

Application: "MapInfo"**Topic:** *Имя запущенной MapBasic-программы***Команды и темы, поддерживаемые в сеансе обмена DDE:**

Действие	Параметр item	Результат
Запрос на считывание данных	"{items}"	MapInfo возвращает разделенный символами табуляции список глобальных переменных, определенных в приложении. См. пример ниже.
Запрос на считывание данных	Имя глобальной переменной	MapInfo возвращает строку, представляющую собой значение глобальной переменной.
Запрос на считывание данных	Строка, не являющаяся именем глобальной переменной	Если приложение MapBasic содержит функцию с именем RemoteQueryHandler() , MapInfo вызовет ее. Внутри тела функции можно определить имя item посредством вызова: CommandInfo(CMD_INFO_MSG) . Это новое свойство в MapInfo 4.0.
Поместить данные (Poke)	Имя глобальной переменной.	MapInfo изменит значение переменной на новое.
Execute	Текстовое послание	Если в приложении MapBasic есть процедура с именем RemoteMsgHandler , MapInfo вызовет ее. Внутри тела процедуры можно узнать содержание текстового послания при помощи вызова CommandInfo(CMD_INFO_MSG) .

Например, следующая программа MapBasic, которую Вы можете напечатать непосредственно в MapBasic-окне, проводит простой DDE диалог, используя "SCALEBAR.MBX" как topic.

При помощи этого диалога выводится список глобальных переменных, используемых приложением SCALEBAR.MBX. **Обратите внимание:** Этот диалог будет только работать, если приложение SCALEBAR.MBX уже выполняется.

```
Dim i_channel As Integer
i_channel = DDEInitiate("MapInfo", "SCALEBAR.MBX")
Print DDERequest$(i_channel, "{items}" )
DDETerminate i_channel
```

Как MapInfo обрабатывает сообщение DDE-Execute

Имеются два способа, которыми клиентское приложение может посылать MapInfo запрос на выполнение действий:

- Когда DDE-диалог использует "System" как topic и клиентское приложение посылает запрос на выполнение действия, MapInfo пробует выполнять определенное сообщение как оператор MapBasic.
- Когда диалог использует имя приложения MapBasic как topic, и клиент посылает запрос на выполнение действия, MapInfo вызывает процедуру **RemoteMsgHandler** приложения, которая может затем вызывать **CommandInfo ()** чтобы определить текст сообщения.

Приложение MapBasic может действовать как клиент в одном DDE-диалоге, и одновременно как сервер в другом диалоге. Приложение MapBasic может инициализировать диалог с другим приложением MapBasic или непосредственно с MapInfo.

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 подробная информация приведена в главе 12.

Пример DDE-обмена

Пример чтения/записи в таблицу Excel с помощью DDE-связи приведен в описании функции **DDEInitiate()** в *Справочнике MapBasic*. Программа Watcher (WATCHER.MB), поставляемая вместе с пакетом, содержит более сложный пример DDE-обмена. WATCHER используется при отладке: если Вы запустили приложение MapBasic и следом за ним программу Watcher, то последняя открывает таблицу в Excel и помещает в нее текущие значения глобальных переменных приложения, подвергаемого отладке. Другими словами, WATCHER позволяет просматривать глобальные переменные программы в таблице Excel.

Программа WATCHER проводит несколько сеансов DDE-обмена:

- Сначала WATCHER проводит сеанс DDE-обмена, используя "MapInfo" и "System" как параметры **application** и **topic**, и запрашивает список действующих приложений MapBasic. Затем WATCHER показывает диалоговое окно, в котором Вы можете выбрать одно из приложений для отладки.

- Затем WATCHER начинает второй сеанс DDE-обмена, обращаясь к выбранному в первом сеансе приложению MapBasic, как к объекту (**topic**). Используя оператор **DDERequest\$()**, Вы получаете список глобальных переменных этого приложения.
- И, наконец, WATCHER проводит третий сеанс, на этот раз с Excel. Используя оператор **DDEPoke**, WATCHER, с помощью макропрограммы Excel, размещает значения переменных и их названия в таблице Excel.

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. 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.

Если вы создадите Справочник для работы в среде MapInfo для Windows, то он не будет работать в MapInfo для Macintosh или в MapInfo для UNIX. Последние две среды используют свои Справочные системы.

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):

```
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**. Если Вы создали свой Справочный файл и назвали его 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 *filename.hpj*). Подробно структура этого файла описана в документации к Windows Software Developers Kit (SDK).

Если нужно снабдить Справочником диалоговое окно, созданное в Вашей программе, поместите в диалог кнопку (Button control) 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 Справочника. Подробнее о процедурах-обработчиках см. главу 6.

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 другое приложение.

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.

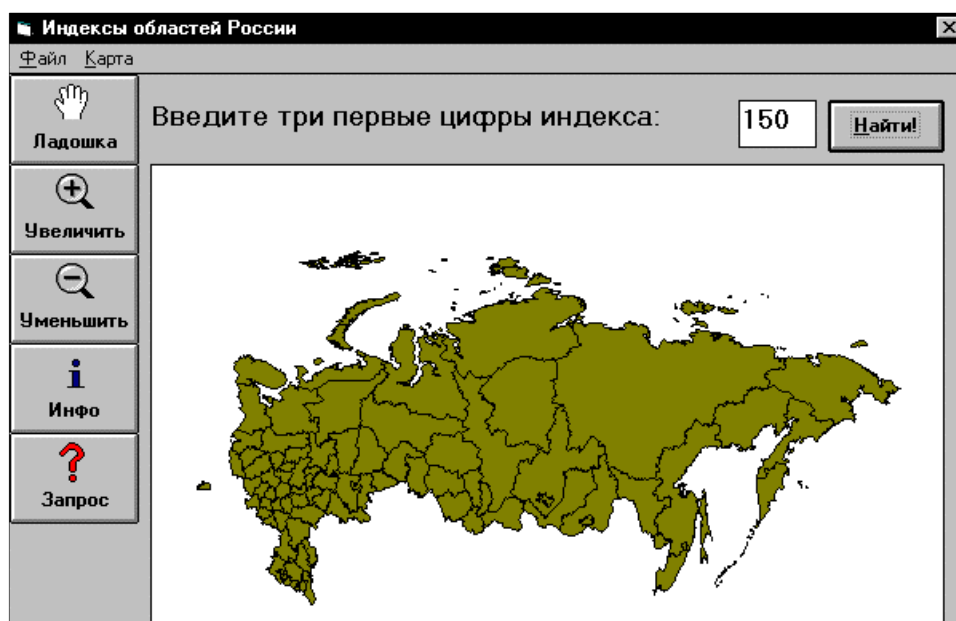
11

Глава

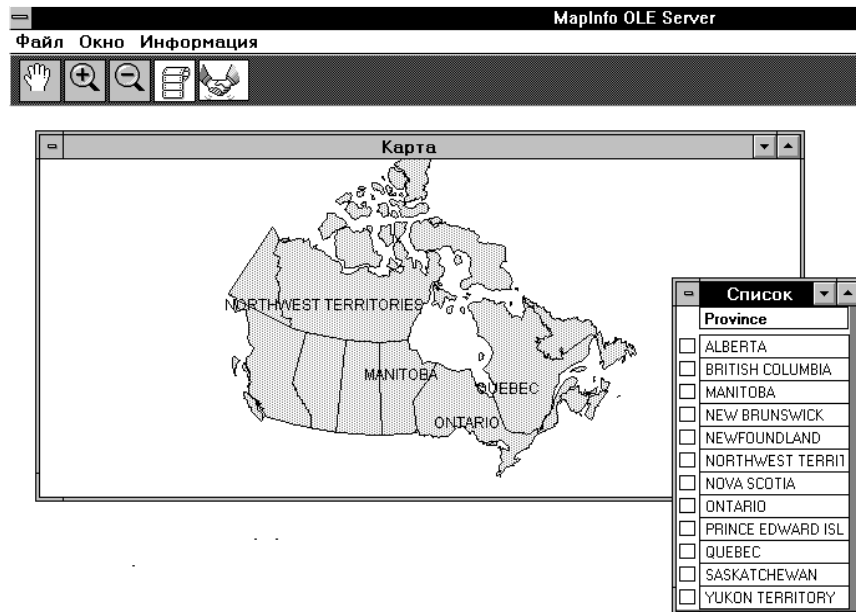
- >Что такое Интегрированная Картография
 - >Концепции Интегрированной Картографии
 - >Технические аспекты Интегрированной Картографии
 - >Простейший пример
 - >Подробное обсуждение Интегрированной Картографии
 - >Использование уведомляющих вызовов (Callback) для получения из MapInfo
 - >Другие способы использования уведомлений
 - >Полезные функции и операторы MapBasic
 - >Аргументы командной строки MapInfo
 - >Добавление кнопок и процедур-обработчиков
-

Как выглядит приложение с Интегрированной Картой?

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.



На следующем рисунке показано приложение с мультидокументным (MDI) интерфейсом, также написанное на языке Visual Basic, которое показывает окна Карты и Списка MapInfo.



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.

Внимание: 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 С и Visual Basic. Примеры кода в этой главе даны на языке Visual Basic.

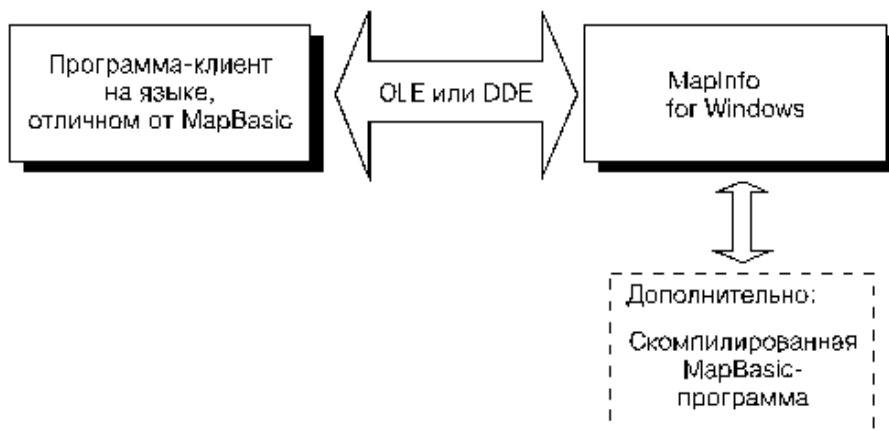
В Вашей программе должна присутствовать инструкция, запускающая MapInfo в фоновом режиме. Например, в программе на языке Visual Basic Вы можете запустить MapInfo вызовом функции **CreateObject()**. Программа MapInfo запускается в фоновом режиме незаметно для пользователя, не выводя заставку на дисплей.

Ваша программа осуществляет управление программой MapInfo, конструируя строки, представляющие операторы языка MapBasic, которые затем передаются в MapInfo посредством механизма управления объектами OLE (OLE Automation) или динамического обмена данными (DDE). MapInfo 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 MapBasic-программе. 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” (reparenting) окна. Вы можете переподчинить окна Карты, Списка, Графика, Отчета и Легенды.

Замечание: Переподчинение окон MapInfo другому приложению не дает программе MapInfo автоматического доступа к данным этого приложения. Для отображения данных приложения в окне MapInfo Вы должны предварительно записать эти данные в таблицу MapInfo.

На следующей схеме показаны основные элементы приложения с Интегрированной Картой:



Внимание: 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 MapBasic-программы, 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 или т.н. исполнимый (runtime) модуль MapInfo (специальная “усеченная” версия MapInfo, поставляемая только в качестве основы для специализированных приложений).
- 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). Рекомендуются применение механизма управления объектами OLE как более быстрого и надежного метода по сравнению с динамическим обменом данными.
- Компьютер пользователя должен иметь достаточно свободной памяти для одновременной работы Вашей программы-клиента и программы MapInfo.
- 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 HWND-номера для элементов пользовательского интерфейса.

Другие технические замечания

- Для разработки приложения с Интегрированной Картой Вы должны написать программу на языке, отличном от MapBasic, называемую в дальнейшем **программа-клиент**. Вы можете написать программу клиент, используя популярные среды программирования, такие как Visual Basic, 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 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 Corporation не предоставляет Вам какие-либо заголовочные файлы или библиотеки. (Программное обеспечение MapInfo включает в себя несколько динамически подключаемых библиотек (файлы .DLL), но Вы не можете обращаться к ним непосредственно; эти библиотеки предназначены для внутреннего использования программой MapInfo.)

Простейший пример

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 в приложение, написанное на языке Visual Basic (в дальнейшем VB-приложение или VB-программа), и исходит из двух предположений:

- Вы уже знаете технику программирования на языке Visual Basic. (Примеры в данном обсуждении используют синтаксис языка Visual Basic.)
- 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.

Запуск MapInfo

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:

```
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")
```

Внимание: 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 **CreateObject()** и **GetObject()** используют механизм управления объектами OLE (OLE Automation) для связи с MapInfo. Если Вам необходимо применить DDE-связь вместо OLE-связи, используйте функцию **Shell()** языка Visual Basic для запуска программы MapInfo, а затем используйте свойство (property) **LinkMode** для установления DDE-связи.

В 32-разрядной версии Windows (Windows95 или Windows NT) можно запускать несколько экземпляров MapInfo. Если Вы запустите MapInfo и вслед за этим программу, использующую Интегрированную Картографию и вызывающую **CreateObject()**, то будут работать два независимых экземпляра MapInfo. Однако в 16-разрядной версии может работать только один экземпляр MapInfo; если Вы запустите MapInfo и вслед за этим программу, использующую Интегрированную Картографию и вызывающую **CreateObject()**, то она не сможет запустить второй экземпляр MapInfo.

Пересылка команд в программу MapInfo

После запуска программы MapInfo 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 ""RUSSIA.TAB"" Interactive "
```

Если Вы установили связь с MapInfo, используя механизм управления объектами OLE (OLE Automation), передавайте командную строку программе MapInfo 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, используя динамический обмен данными (DDE), передавайте командную строку программе MapInfo DDE-методом **LinkExecute**.

Внимание: Вы можете передать оператор в программу MapInfo, если этот оператор допустим в окне MapBasic. Например, Вы не можете переслать MapBasic-оператор **Dialog**, поскольку его использование не разрешено в окне MapBasic.

Для определения допустимости использования оператора языка MapBasic в окне MapBasic обратитесь к *Справочнику MapBasic* или откройте Справочную систему; искомая информация находится под заголовком "Предупреждение". Например, в Справке по оператору

Dialog дано следующее ограничение: “Вы не можете использовать оператор **Dialog** в окне **MapBasic**”. Как правило, операторы языка **MapBasic**, управляющие последовательностью исполнения (такие, как **For...Next** и **Goto**), не разрешены для исполнения в окне **MapBasic**.

Запрос данных от программы **MapInfo**

Для выполнения запроса из Вашей программы-клиента значения **MapBasic**-выражения задайте в **VB**-программе строку, представляющую выражение. Например, если Вы хотите определить значение, возвращаемое **MapBasic**-функцией **WindowID(0)**, задайте следующую строку (в среде **Visual Basic**):

```
msg = "WindowID(0)"
```

Если Вы установили связь с **MapInfo**, используя механизм управления объектами **OLE (OLE Automation)**, передайте строку выражения программе **MapInfo** **OLE**-методом **Eval**. Например:

```
Dim result As String
result = mapinfo.Eval "WindowID(0)"
```

При использовании метода **Eval** программа **MapInfo** интерпретирует строку как выражение языка **MapBasic**, определяет значение выражения и возвращает это значение в виде строки. **Замечание:** Если выражение приводится к логическому значению (тип **Logical**), **MapInfo** возвращает односимвольную строку, “T” или “F” соответственно.

Если Вы установили связь с **MapInfo**, используя динамический обмен данными (**DDE**), запросите значение выражения **DDE**-методом **LinkRequest**.

Переподчинение окон **MapInfo**

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** оператор **Set Next Document**, за которым следует **MapBasic**-оператор, создающий окно. Например, следующий фрагмент кода создает окно Карта **MapInfo** как подчиненное окно программы-клиента. (В этом примере “**MapFrame**” является именем элемента управления **Picture Box** (Поле Иллюстрации) 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 этого оператора 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.

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 номера. Этот запрос выполняется следующим образом:

```
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.

Внимание: Совет: Если Вы программируете на С, 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

MapInfo 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 " & FindZip-
Form.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 **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.

Совет: Вы можете создать “плавающее” окно Легенды внутри окна Карты. В операторе **Set Next Document** укажите окно Карты как порождающее окно. 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.

Внимание: Когда пользователь изменяет размеры окна Карты, MapInfo не производит автоматически обновление содержания окна, чтобы заполнить его новый размер. 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, если Ваша программа Visual Basic выполняется под управлением 32-битной версии Windows, Вы можете использовать следующее описание для доступа к 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.

Интеграция инструментальных панелей MapInfo

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).

“Магический” номер 1707 ссылается на инструмент 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, этот пример можно записать так:

```
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
Стрелка	1701	M_TOOLS_SELECTOR
Выбор в рамке	1722	M_TOOLS_SEARCH_RECT
Выбор в круге	1703	M_TOOLS_SEARCH_RADIUS
Выбор в области	1704	M_TOOLS_SEARCH_BOUNDARY
Увеличивающая лупа	1705	M_TOOLS_EXPAND
Уменьшающая лупа	1706	M_TOOLS_SHRINK
Ладонка	1702	M_TOOLS_RECENTER
Информация	1707	M_TOOLS_PNT_QUERY
Подпись	1708	M_TOOLS_LABELER
Линейка	1710	M_TOOLS_RULER
Переноска	1734	M_TOOLS_DRAGWINDOW

Вы также можете создавать пользовательские кнопки, которые вызывают определенные процедуры при нажатии на них. Обзор соответствующих возможностей смотрите в главе 6. Ниже в этой главе рассказывается, как новые кнопки можно использовать в Интегрированной Картографии.

Настройка “быстрых” меню MapInfo

MapInfo вызывает “быстрые” меню, если пользователь нажимает правую кнопку мышки в окне MapInfo. Эти меню появляются даже во внедренных приложениях. В зависимости от характера Вашего приложения Вы можете захотеть модифицировать или даже удалить такое меню. Например, Вы, возможно, захотите удалить команду **ДУБЛИРОВАТЬ ОКНО**, так как эта команда не работает в OLE-приложении.

Чтобы удалить одну или несколько команд из локального меню, используйте оператор MapBasic **Alter Menu ... Remove** или переопределите меню целиком, используя оператор **Create Menu**. Подробнее смотрите в *Справочнике MapBasic*.

Чтобы добавить команду к локальному меню, используйте оператор MapBasic **Alter Menu ... Add** и синтаксис предложений **Calling OLE** или **Calling DDE**.

Чтобы удалить “быстрое” меню полностью, используйте оператор MapBasic **Create Menu** и управляющий код “ (- ” как новое определение меню. Например, следующий оператор разрушает “быстрое” меню для окон Карты:

```
mapinfo.do "Create Menu ""MapperShortcut"" ID 17 As  
""(-"" "
```

Вывод на печать интегрированного окна MapInfo

Вы можете использовать оператор **PrintWin** языка MapBasic для вывода окна MapInfo на печать даже в том случае, когда оно переподчинено. Пример Вы можете найти в программе FindZip, поставляемой в комплекте с MapBasic. Меню Файл программы FindZip включает в себя команду **Print Map**, при выборе которой пользователем 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**.

Обнаружение ошибок времени исполнения

Когда Ваша клиентская программа посылает в MapInfo командную строку, возможно возникновение ошибок. Например, команда Map From World потерпит неудачу, если таблица World не открыта. MapInfo в этом случае сгенерирует код ошибки.

Чтобы обработать ошибку MapInfo, 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. Распечатка кодов ошибок приведена в текстовом файле `ERRORS.DOC`.

Внимание: Заметьте, что свойство `LastErrorCode` возвращает значения, которые на 1000 больше, чем коды в `ERRORS.DOC`. Другими словами, если возникла ошибка с кодом 311, то `LastErrorCode=1311`.

Когда запускается приложение MapBasic (MBX файл) через 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 которое будет вызываться через Automation, попытайтесь сделать MBX проще. Внутри MBX, избегайте использовать оператор MapBasic **OnError**; вместо этого, применяйте другие способы проверки и избежания ошибок перед запуском MBX.

Завершение программы MapInfo

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 сеанс DDE прежде, чем завершается выполнение программы. Иначе возможны разнообразные ошибки. Эта проблема встречается, когда Вы запускаете 16-битную программу на Visual Basic в 32-битной версии Windows (Windows95 или WindowsNT).

Чтобы избежать этой неприятности, устройте в программе Visual Basic так, чтобы DDE-связи закрывались до того, как она завершится.

Замечание о командных строках MapBasic

Как показано ранее, Вы можете в VB-программе создавать строки, представляющие операторы языка MapBasic, и затем пересылать эти строки в программу MapInfo посредством OLE-метода **Do**. Заметьте, что Вы можете 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 MapBasic-оператора – оператор **Open Table**, оператор **Set Next Document**, и оператор **Map From**. Программа MapInfo 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 их 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 Вашей клиентской программы.

Переключение режима совмещения узлов нажатием клавиши S поддерживается автоматически.

Использование уведомляющих вызовов (Callbacks) для получения информации из MapInfo

Вы можете построить Ваше приложение так, чтобы MapInfo автоматически посылало информацию Вашей клиентской программе. Например, можно сделать так, чтобы всякий раз, когда изменяется активное окно Карты, MapInfo вызывало Вашу клиентскую программу, чтобы сообщить ID-номер окна. Такой тип уведомления известен как *обратный вызов* или *уведомление* (callback).

Обратные вызовы позволяют, чтобы MapInfo посылало информацию вашей клиентской программе в следующих случаях:

- **Пользователь применяет инструмент в окне.** Например, если пользователь производит перемещение объекта мышкой в окне Карты, MapInfo может вызвать Вашу клиентскую программу, чтобы сообщить x- и y-координаты.
- **Пользователь выбирает команду меню.** Например, предположим, что Ваше приложение настраивает “быстрое” меню MapInfo (меню, возникающее при нажатии правой кнопки мышки). Когда пользователь выбирает команду из этого меню, MapInfo может вызвать Вашу клиентскую программу, чтобы сообщить ей о выборе.
- **Изменяется окно Карты.** Если пользователь изменяет содержание окна Карты (например, добавляя или передвигая слои), MapInfo может послать Вашей 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 Ваша клиентская программа эмулировала строку состояния MapInfo, то 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, Ваша клиентская программа должна быть способна функционировать, как DDE-сервер или как сервер Автоматизации OLE. Visual Basic 4.0 Professional Edition и C++ могут создавать такие приложения. Однако приложения Visual Basic 3.0 не могут являться серверами Автоматизации OLE, поэтому они должны использовать 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 Вашего языка программирования.
2. Если Вы хотите имитировать строку состояния MapInfo, создайте метод, называемый **SetStatusText**. Определите этот метод так, чтобы у него был один аргумент: строка.
3. Если Вы хотите, чтобы MapInfo сообщало Вашей 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.
4. Создайте объект, используя ваш класс. Например, если Вы назвали класс "CMyClass", следующий оператор Visual Basic создает объект этого класса:

```
Public myObject As New CMyClass
```
5. Создайте объект, используя Ваш класс. Например, если Вы назвали класс "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 SetCallback и точно укажите название объекта:

```
mapinfo.SetCallback myObject
```

7. 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 именем метода (см. шаг 4).

Заметьте, что инструментальные панели MapInfo 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**, чтобы активизировать этот инструмент.

8. 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 (см. шаг 4).
9. Внутри метода обработайте аргументы, посланные MapInfo. Если Вы создали метод **SetStatusText**, MapInfo передает ему строку, содержащую текст строки состояния. Если Вы хотите эмулировать строку состояния MapInfo, напишите код, чтобы поместить этот текст где-нибудь в Вашем интерфейсе пользователя.

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 Вы следите за размером окна Карты, то Вы можете вызвать функцию MapInfo **MapperInfo()**.

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, рисующий инструмент и т.д. В следующем разделе описан формат строки.

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 для событий, связанных с меню	Код для событий, связанных с кнопкой
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 OLE, MapInfo создает строку, содержащую разделенные запятой все восемь возвращаемых **CommandInfo()** значений. Строка начинается с префикса "MI:", чтобы Ваш OLE-сервер мог определять, что обращение метода было сделано MapInfo.

Строка, которую MapInfo посылает Вашему методу, выглядит следующим образом:

```
"MI:" +  
CommandInfo(1) + "," + CommandInfo(2) + "," +  
CommandInfo(3) + "," + CommandInfo(4) + "," +
```

```
CommandInfo(5) + "," + CommandInfo(6) + "," +  
CommandInfo(7) + "," + CommandInfo(8)
```

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.

Теперь предположим, что Вы создаете кнопку панели MapInfo, которая позволяет пользователю линии на Карте. Строка теперь будет выглядеть следующим образом:

```
"MI:-73.5548,42.122,F,F,-72.867702,43.025,202,"
```

Теперь строка включает несколько элементов. 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.

Внимание: Если Вы приписываете уникальный идентификатор каждой из Ваших кнопок, Вы можете сделать так, что все кнопки будут вызывать один и тот же метод. Ваш метод может определять, какая из кнопка вызвала его, используя восьмой аргумент в переданной строке.

Разбор содержания строки возлагается на Ваше приложение.

Синтаксис C/C++ для функций уведомления

В предыдущем разделе были описаны обратные вызовы в контексте Visual Basic. Здесь мы рассмотрим синтаксис языка C.

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 уведомлений

Как говорилось ранее, MapInfo может использовать обратные вызовы OLE, чтобы послать информацию Вашей 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 Ваши собственные серверы Автоматизации OLE.

MapInfo поддерживает два типа уведомлений, которые не используют механизм OLE: использующие DDE, и использующие 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 посредством DDE, используйте синтаксис вызова DDE-сервера. Всякий раз, когда пользователь использует кнопку или команду меню, MapInfo открывает DDE-связь с DDE-сервером, и затем посылает строку Вашему объекту. Строка использует формат, обсужденный в предыдущей секции (например, " MI:,,,,, 101 ").

Пример смотрите в программе FindZip. Процедура Form Load посылает MapInfo оператор **Alter ButtonPad ... Add** с предложением

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). Подробнее смотрите в Главе 11.

Обратные вызовы MBX

Если Вы создаете приложение MapBasic (файл MBX), Вы можете сконструировать Ваши кнопки и команды меню так, чтобы они вызвали MapBasic п-процедуры в MBX. В предложении вызова используйте синтаксис вызова процедуры **Calling MapBasic-** программы.

После того, как Ваше приложение на языке Visual Basic запустит Map-Info, запустите MBX, послав MapInfo строку вида:

mapinfo.do "Run Application ""C:\MB\MYAPP.MBX"" "

Подробности о создании кнопок и команд меню смотрите в главе 6.

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 Справочной системы для этого окна.

Вызов стандартного Справочного файла MapInfo

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 MAPINFOW.HLP.

Внимание: 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 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 специализированного Справочного файла

Вы можете сконфигурировать MapInfo для вызова Справочной системой специализированного Справочного файла. Например, следующий MapBasic-оператор указывает MapInfo на необходимость использования Справочного файла CUSTOM.HLP вместо MAPINFOW.HLP:

```
Set Window Help File "CUSTOM.HLP" 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 специфицированный Вами Справочный файл вместо Справочного файла MAPINFOW.HLP. Используйте такую конфигурацию, если Вы хотите обеспечить Справку для одного или нескольких окон диалога MapInfo, 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 клавишу 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.

Например, диалоговое окно MapInfo “Поиск” 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.

Учтите следующие факторы:

- В комплект поставки языка MapBasic не включен компилятор Справочных файлов в формате Windows (.HLP).
- ID-номера диалоговых окон MapInfo могут быть изменены в будущих версиях программы.

Полезные операторы и функции языка MapBasic

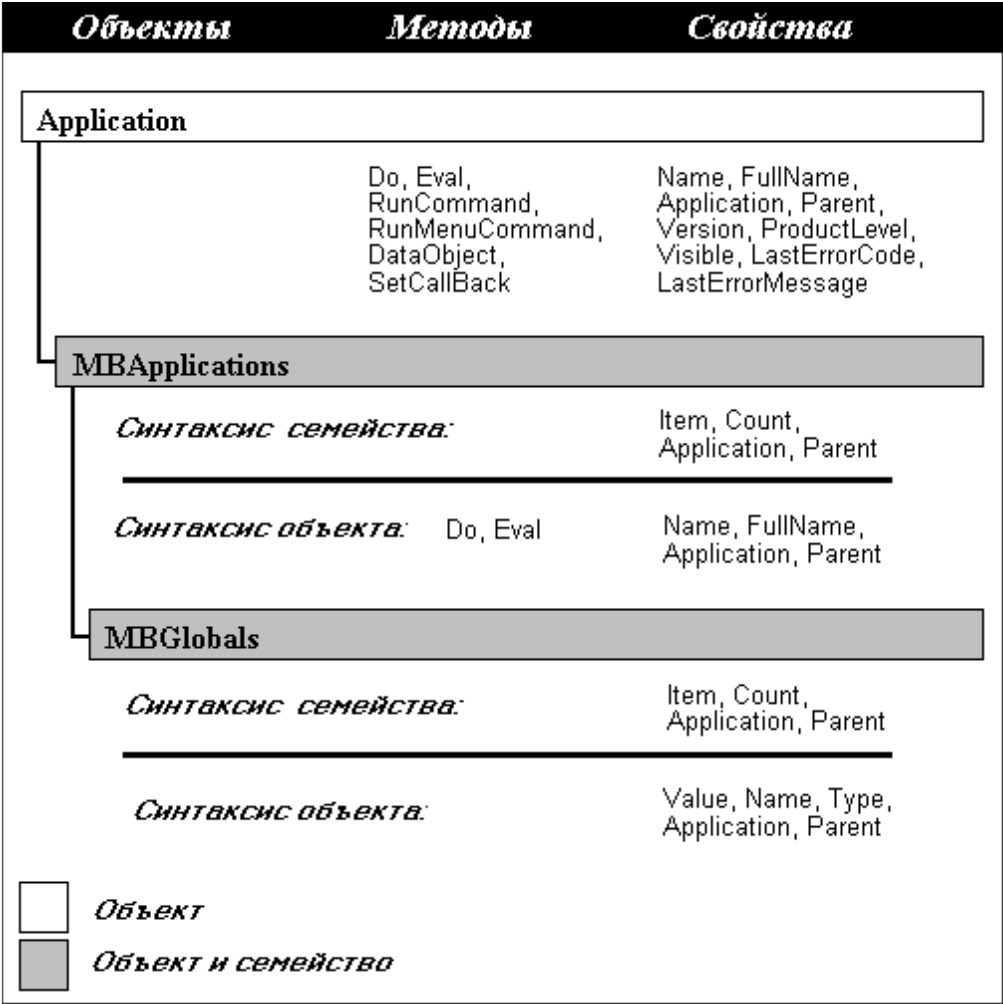
В этом разделе перечислены некоторые операторы и функции языка MapBasic, применение которых особенно полезно в приложениях с Интегрированной Картой. Детальное обсуждение этих операторов и функций смотрите в *Справочнике MapBasic* или Справочной системе.

Имя оператора или функции	Описание
Create Legend	Создает новое окно Легенды; позволяет Вам создать окно Легенды как подчиненное окно Вашей программы.
Map	Создает новое окно Карты.
MenuItemInfoByID() MenuItemInfoByHandler(),	Возвращает статус команды меню в MapInfo (отмечена галочкой или нет).

Имя оператора или функции	Описание
Open Table	Открывает таблицы MapInfo.
RemoteQueryHandler()	Позволяет MapBasic-программам обрабатывать запросы на чтение от DDE-клиентов.
Run Menu Command	Имитирует выбор пользователем команды меню MapInfo или кнопки на инструментальной панели (ButtonPad).
SearchInfo()	Возвращает информацию о результатах исполнения функций SearchPoint() и SearchRect().
SearchPoint(), SearchRect()	Осуществляют поиск в выбранных слоях окна Карта объектов в точке с заданными координатами (х,у) или в заданной прямоугольной области соответственно. Эти функции позволяют Вам эмулировать инструменты MapInfo Информация и Подпись.
Set Application Window	Обеспечивает переподчинение окон диалога и сообщений об ошибках MapInfo. Выдайте этот оператор в Вашей клиентской программе после запуска MapInfo или подключения к работающему экземпляру MapInfo.
Set Map	Управляет различными режимами представления окон типа Карты.
Set Next Document	Переподчиняет окно MapInfo-документа (например, окно Карты или Легенды) так, что оно становится порожденным окном Вашей программы-клиента.
Set Window	Управляет различными режимами представления окон MapInfo.
Shade, Set Shade	Создает или изменяет тематические слои Карты.
SystemInfo()	Некоторые значения, возвращаемые этой функцией, предназначены специально для Интегрированной Картографии. Пример: задайте в качестве аргумента SYS_INFO_APPLICATIONWND, чтобы получить HWND-номер приложения.
WindowID() WindowInfo()	Возвращают информацию об окнах MapInfo, в том числе переподчиненных.

Объектная модель механизма управления объектами OLE

На следующей диаграмме приведена схема MapInfo 6.5 OLE Automation Type Library. Методы и Свойства (Properties) подробно описываются на следующих страницах.

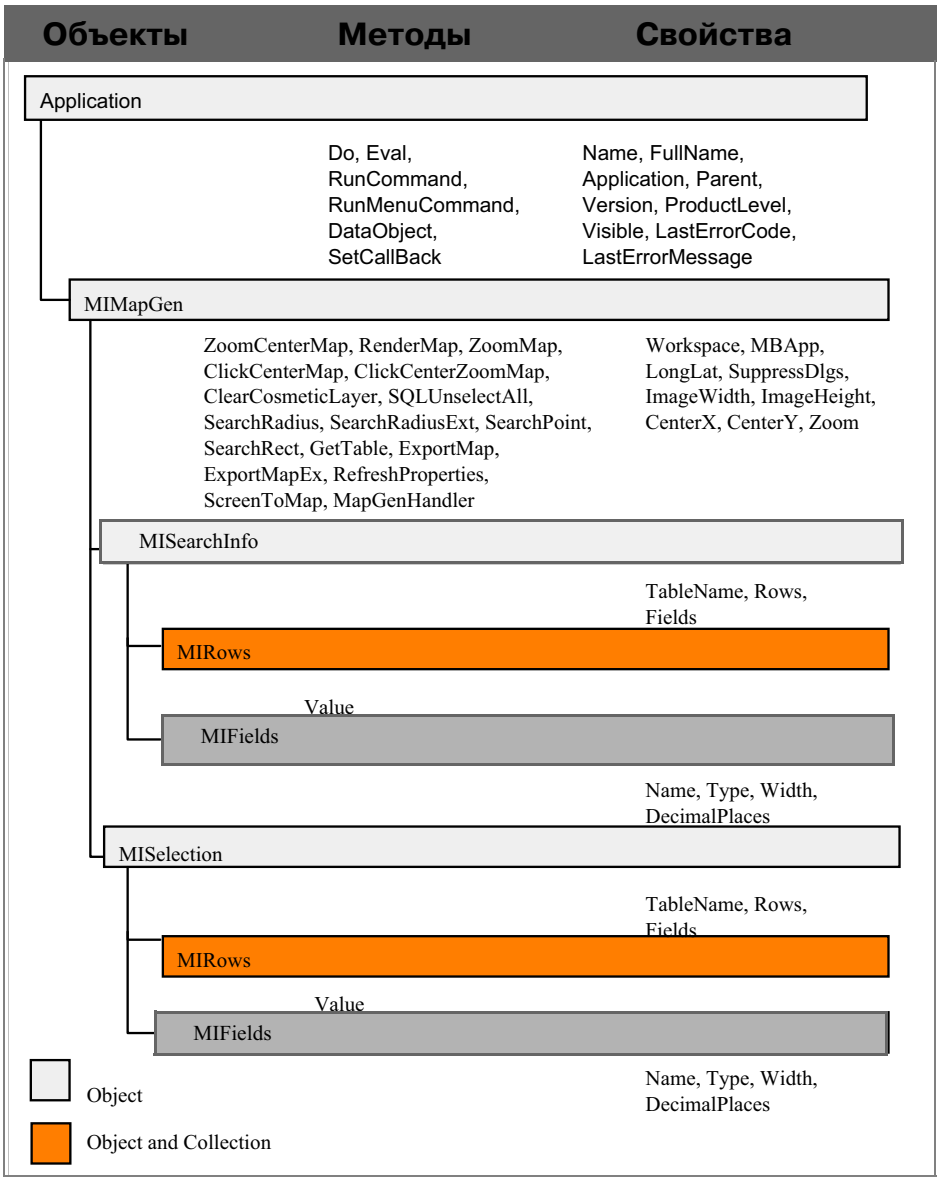


Объект **Application** (Приложение) представляет работающий экземпляр MapInfo.

Семейство **MBApplications** (MapBasic-Приложения) представляет список MapBasic-приложений, работающих в данный момент.

Семейство **MBGlobals** (Глобальные переменные в среде MapBasic) представляет список глобальных переменных, объявленных одним из работающих MapBasic-приложений.

Следующая диаграмма показывает дополнительные объекты возможные в MapInfo's 6.5 OLE Automation Type Library. Методы и Свойства описываются в дальнейшем.



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.

Имя свойства	Функциональность
Name	Возвращает имя приложения (например, “MapInfo”). Стандартное свойство OLE. Свойство по умолчанию для объекта Application.
FullName	Возвращает полный маршрут к исполняемому файлу приложения. Стандартное свойство OLE.
Application	Возвращает значение IDispatch объекта Application. Стандартное свойство OLE.
Parent	Возвращает значение IDispatch порождающего объекта; для объекта Application возвращает себя самого. Стандартное свойство OLE.
Version	Возвращает текстовое представление номера текущей версии, умноженного на 100 (например, MapInfo 6.0.0 возвращает “600”).
ProductLevel	Возвращает целое значение, обозначающее “уровень” программы MapInfo. Для MapInfo Professinal возвращает 200.
Visible	Разрешена для записи. Логическое значение, контролирующее видимость окна приложения. Считывание позволяет определить видимость окна; присваивание свойству значения устанавливает видимость или невидимость окна.
LastErrorCode	Разрешено для записи/чтения. Целое значение, указывающее кодовый номер последней MapBasic-ошибки, случившейся в процессе вызова метода Do, Eval или RunCommand. Коды ошибок, возвращаемые этим свойством, превышают на 1000 соответствующие коды ошибок в среде MapBasic. Эти коды ошибок автоматически заменяются кодом следующей ошибки, но никогда автоматически не сбрасываются в 0.
LastErrorMessage	Возвращает строку, представляющую текст сообщения об ошибке, соответствующей значению LastErrorCode.

Методы объекта Application

Имя метода	Имя метода
Do(string)	Интерпретирует строку <i>string</i> как оператор языка MapBasic, затем выполняет этот оператор.
Eval(string)	Интерпретирует строку <i>string</i> как MapBasic-выражение, вычисляет выражение и возвращает его значение. Если выражение логическое, то возвращает “Т” или “F”.
RunCommand(string)	Интерпретирует строку <i>string</i> как оператор языка MapBasic; является синонимом “Do”.
RunMenuCommand(menuid)	Исполняет команду меню, указанную аргументом <i>menuid</i> типа Integer. Пример смотрите ниже
DataObject(windowID)	По целочисленному ID-номеру окна <i>windowID</i> возвращает интерфейс IUnknown, представляющий это окно. Для получения графического представления окна в виде метафайла используйте QueryInterface для интерфейса IDataObject. Только значения Idata и ObjectIUnknown разрешены для этого объекта. Замечание: Это “продвинутая” возможность для программистов, использующих язык программирования C. Смотрите далее в этой главе “Уведомления вызывающего приложения”.
SetCallBack(IDispatch)	Регистрирует объект механизма управления объектами OLE (OLE Automation) как получатель уведомлений, генерируемых программой MapInfo. Только одна функция уведомления может быть зарегистрирована в каждый данный момент. Замечание: Это “продвинутая” возможность для программистов, использующих язык программирования C. Смотрите далее в этой главе “Уведомления вызывающего приложения”.

Например, следующая VB-инструкция использует метод Do для передачи программе MapInfo оператора Map, открывающего окно Карта:

```
mapinfo.Do "Map From World"
```

Следующая инструкция использует метод RunMenuCommand для выполнения команды меню MapInfo, имеющей значение кода 1702, которая выбирает MapInfo-инструмент Ладощка. (Для определения конкретного значения кода интересующей Вас команды меню смотрите файл MENU.DEF; 1702 является значением кода M_TOOLS_RECENTER.)

mapinfo.RunMenuCommand 1702

Свойства семейства MBAplications

Семейство MBAplications включает в себя все MapBasic-приложения, работающие в среде MapInfo в данный момент. Все свойства, перечисленные в следующей таблице, разрешены только для чтения.

Имя свойства	Функциональность
Item	Возвращает значение IDispatch конкретного объекта <i>programobject</i> . Аргумент имеет тип VARIANT, который может приводиться к целочисленному индексу в диапазоне (1..Count) или строковому значению (имя программы). Является стандартным свойством для семейства MBAplications.
Count	Возвращает длинное целое число объектов в семействе (т.е. число работающих приложений).
Application	Возвращает значение IDispatch приложения Map-Info.Стандартное свойство OLE..
Parent	Возвращает значение IDispatch порождающего объекта; для семейства MBAplications это приложение MapInfo. Стандартное свойство OLE.

Свойства объекта в семействе MBAplications

Каждый объект в семействе MBAplications является работающим MapBasic-приложением. Для все свойств, перечисленных в следующей таблице, разрешено только чтение.

Имя свойства	Функциональность
Name	Возвращает имя приложения (например, "FOO.MBX"). Стандартное свойство OLE для объекта MBApapplication.
FullName	Возвращает полный DOS-маршрут к исполняемому .MBX файлу MapBasic-приложения. Стандартное свойство OLE.
Application	Возвращает значение IDispatch MapBasic-приложения. Стандартное свойство OLE.
Parent	Возвращает значение IDispatch порождающего объекта; для <i>programobject</i> это приложение MapInfo. Стандартное свойство OLE.

Например, следующий оператор определяет имя работающей MapBasic-программы:

```
Dim appsList As Object
Dim firstname As String

Set appsList = mapinfo.MBApplications
If appsList.Count > 0 Then
    firstname = appsList(1).Name
End If
```

Методы объекта в семействе MBApplications

Имя метода	Функциональность
Do(<i>string</i>)	Строка <i>string</i> пересылается в процедуру RemoteMsgHandler MapBasic-приложения.
Eval(<i>string</i>)	Строка <i>string</i> передается как аргумент в функцию RemoteQueryHandler() MapBasic-приложения; возвращается значение этой функции. Замечание: Если выражение имеет значение типа Logical, MapInfo возвращает односимвольную строку, “T” или “F”.

Свойства семейства 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.

Имя свойства	Функциональность
Item	Возвращает значение IDispatch конкретного объекта <i>mbglobal</i> . Аргумент имеет тип VARIANT, который может приводиться к целочисленному индексу в диапазоне (1...Count) или строковому значению (имя глобальной переменной). Является стандартным свойством для семейства MBGlobals.
Count	Возвращает длинное целое, число объектов в семействе (коллекции) (число глобальных переменных).

Application	Возвращает значение IDispatch MapBasic-приложения. Стандартное свойство OLE.
Parent	Возвращает значение IDispatch порождающего объекта; для объекта <i>mbglobal</i> это объект <i>programobject</i> , который декларировал глобальную переменную. Стандартное свойство OLE.

Свойства объекта в семействе MBGlobals

Каждый объект семейства MBGlobals является глобальной переменной в среде MapBasic. Все свойства, перечисленные в следующей таблице, разрешены только для чтения, за исключением свойства Value.

Имя свойства	Функциональность
Value	Разрешена для чтения/записи. Считывание свойства возвращает строку, представляющую значение глобальной переменной; присваивание значения свойству изменяет значение глобальной переменной. Свойство по умолчанию для объекта MBGlobal.
Name	Возвращает имя переменной. Стандартное свойство OLE.
Type	Возвращает текстовую строку, представляющую тип переменной как один из стандартных типов MapInfo (“Integer”, “Date” и т.д.).
Application	Возвращает значение IDispatch MapBasic-приложения. Стандартное свойство OLE.

Имя свойства	Функциональность
Parent	Возвращает значение IDispatch порождающего объекта; для объекта <i>mbglobal</i> это объект <i>programobject</i> , который декларировал глобальную переменную. Стандартное свойство OLE.

Следующий пример программы на Visual Basic определяет и изменяет значение глобальной переменной (g_status) MapBasic-программы.

```
Dim globinfo As Object
Dim old_value As Integer

' Извлечение глобальных переменных первой
' из работающих MapBasic-программ
Set globinfo = mapinfo.MBApplications(1).MBGlobals

' Извлечение конкретного значения
old_value = globinfo("g_status").Value

' Присвоение глобальной переменной нового значения:
globinfo("g_status") = old_value + 1
```

Обратите внимание, что `globinfo("g_status")` эквивалентно `globinfo("g_status").Value`.

Свойства объекта MIMapGen

Следующая таблица содержит список свойств, применяемых к объекту MIMapGen. Первоначально объект MIMapGen используется в приложениях MapInfo ProServer; таким образом, приложения MapInfo Professional могут использовать объект MIMapGen. Например, для использования объектной модели MIMapGen, смотрите документацию к ProServer.

Имя свойства	Функциональность
Workspace	Путь к файлу рабочего набора MapInfo.
MBApp	Маршрут к исполняемому приложению MapBasic (MBX файл). Когда он установлен, MapInfo запускает MBX.

Имя свойства	Функциональность
LongLat	Логическое: Определяет интерфейс координатной системы. Если TRUE, все значения, которые Вы вводите или получаете (используя CenterX и CenterY) представляют широту и долготу. Если FALSE, то будет использоваться координатная система окна карты.
SuppressDlgs	Логическое: Если TRUE, то вызывается диалог с сообщением об ошибке. Сюда включаются диалоги, действующие как результат оператора Run Menu Command.
ImageWidth	Ширина изображения, в пикселах.
ImageHeight	Высота изображения в пикселах.
CenterX	X-координата (Долгота) центра карты..
CenterY	Y-координата (Широта) центра карты.
Zoom	Ширина карты (в единицах расстояния).

Обратите внимание, что 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, что бы MapInfo загрузила рабочий набор - такой набор, который содержит одно окно карты. Тогда Вы сможете использовать другие методы и возможности работы с окном карты.

Методы работы с объектом MIMapGen

Следующие методы применяются к объекту MIMapGen.

Метод	Функциональность
ZoomCenterMap()	Перерисовывает карту, основываясь на текущих значениях CenterX, CenterY и Zoom. Карта перерисовывается, только если центр или масштаб изменились с момента последней прорисовки карты.
RenderMap()	Имеет тоже действие, что и ZoomCenterMap, кроме того, что карта всегда перерисовывается.
ZoomMap (double ZoomFactor)	Изменяет масштаб карты, в соответствии с указанием фактора масштаба. Положительное значение увеличивает масштаб; отрицательное уменьшает.

Метод	Функциональность
ClickCenterMap (long MouseX, long MouseY)	Перемещает центр карты в зависимости от места, где будет щелчок мыши. Аргументы x/y отражают положение на карте в пикселах.
ClickCenterZoomMap (long MouseX, long MouseY, double ZoomFactor)	Перемещает центр карты в зависимости от места, где будет щелчок мыши и изменяет масштаб в зависимости от масштабного фактора.
ClearCosmeticLayer()	Те же эффекты, что и от команды Map menu: Удалить все объекты с косметического слоя.
SQLUnselectAll()	Тот же эффект, что и от команды Запрос: Отменить выделение всех строк.
SearchRadius (double CenterPointX, double CenterPointY, double Radius)	Определение радиуса поиска.
SearchRadiusExt (double CenterPointX, double CenterPointY, double OuterPointX, double OuterPointY)	Изменяет радиус поиска, определяет круг поиска, центральную точку и точку, лежащую на радиусе.
SearchPoint (double CenterPointX, double CenterPointY)	Ищет небольшую область вокруг указанного места.
SearchRect (double x1, double y1, double x2, double y2)	Ищет в прямоугольной области.
GetTable (string Tablename)	Возвращает объект MISElection (IDispatch); чтобы получить доступ к содержимому таблицы, используйте объект MISElection.
ExportMap (string ImageType, string FileSpec)	Генерирует растровый файл (JPEG, TIFF, PNG, PSD, BMP, WMF или GIF) окна Карты. Смотрите оператор MapBasic Save Window.
ExportMapEx (string ImageType, string FileSpec, string CopyrightInfo)	Генерирует растровый файл (JPEG, TIFF, PNG, PSD, BMP, WMF или GIF) окна Карты. Смотрите оператор MapBasic Save Window.
RefreshProperties()	Обновляет параметры CenterX, CenterY, Zoom, ImageHeight и ImageWidth.

Метод	Функциональность
ScreenToMap (long ScreenX, long ScreenY, double MapX, double MapY)	Преобразует координаты экрана (пиксeлы) в координаты карты (типа широта / долгота).
MapGenHandler (string Message)	Вызывает подпроцедуру RemoteMapGenHandler в приложении MBX, которая выполняется через MApp. Используйте этот метод для запуска операторов MapBasic в файле MBX.

Внимание: Поисковые методы делают поиск только на самом верхнем слое. Для получения доступа к результатам поиска, смотрите объект MISearchInfo.

Свойства объекта MISearchInfo

Следующие свойства применяются к объекту MISearchInfo.

Свойства	Функциональность
Rows	Это свойство возвращает выборку MIRows (выборку объектов MRow). Эта выборка представляет результаты поиска.
Fields	Это свойство возвращает выборку MFields (выборку объектов MField). Эта выборка представляет установки определенных полей (имена полей и др.) описывающих результат поиска.
TableName	Строка: Имя таблицы, которая содержит результаты поиска.

Внимание: Для получения объекта MISearchInfo, используйте методы поиска объектов MMapGen: SearchRadius, SearchRadiusExt, SearchPoint или SearchRect.

Методы работы с объектом MRow

Следующий метод применяется к объекту MRow. Каждый объект MRow представляет одну запись, возвращаемую поисковым методом или одну строку в таблице определенную в методе поиска GetTable.

Метод	Функциональность
-------	------------------

Value	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).
-------	--

Внимание: Что бы получить выборку 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	Функциональность
Name	Строка: Имя колонки..
Type	Short: Тип данных поля. Допустимы следующие значения: (1) DT_CHAR, (2) DT_DECIMAL, (3) DT_INTEGER, (4) DT_SMALLINT, (5) DT_TIME, (6) DT_LOGICAL, (8) DT_FLOAT.
Width	Short: Ширина поля; применяется только к полям DT_CHAR и DT_DECIMAL.
DecimalPlaces	Short: Число десятичных разрядов в поле DT_DECIMAL..

Внимание: Для получения выборки объектов MIField, обратитесь к свойствам Fields объектов MISearchInfo или MISelection.

Свойства объекта MISelection

Следующие свойства применяются к объекту MISelection.

Свойства	Функциональность
Rows	Это свойство возвращает выборку MIRows (выборку объектов MIRow). Эта выборка представляет все строки в таблице.
Fields	Это свойство возвращает выборку MIFields (выборку объектов MIField). Эта выборка представляет определения полей (имена полей и др.) для таблицы, которая определена в методе GetTable.

Свойства	Функциональность
TableName	Строка: Имя таблицы, которая была определена в методе GetTable.

Внимание: Для получения доступа к объекту MISElection, используйте метод GetTable из объекта MIMapGen.

Аргументы командной строки MapInfo

Если Вы используете для связи с MapInfo динамический обмен данных (DDE), Вы должны запустить MapInfo вручную (например, вызовом функции Shell() языка Visual Basic) перед установлением DDE-связи. При запуске MapInfo 4.0 для Windows Вы можете использовать в командной строке любой из нижеперечисленных аргументов. Если Вы хотите оставить пользователя в неведении о работе программы MapInfo в фоновом режиме, задайте один из следующих аргументов.

Аргумент командной строки	Эффект
-nosplash	MapInfo запускается без вывода на дисплей заставки, показывающей заставку MapInfo, номер версии и т.п..
-server	MapInfo 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.
-automation or -embedding	MapInfo 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.
-regserver	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 способности MapInfo – механизм управления объектами OLE (OLE Automation), Внедрение OLE и т.д.

Аргумент командной строки	Эффект
-unregserver	MapInfo 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 -regserver..
-helpdiag	Этот аргумент устанавливает специальный флажок в MapInfo, в результате чего MapInfo выводит диагностическое окно диалога при каждой попытке вызова Справочной системы нажатием клавиши F1. Более подробное обсуждение использования Справочной системы смотрите далее в этой главе.

Внимание: 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.

Так как эта информация предназначена для программистов, в тексте, для минимального искажения смысла, частично оставлены английские термины.

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 или (PROJECT > APPWIZARD... в версии 1.5).
3. Запустите ассистирующую процедуру MFC AppWizard. 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 из Вашего приложения в MapInfo, то закажите поддержку OLE Automation на шаге 3 из 6 процедуры MFC 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. *имяпроекта*.CPP) и добавьте следующие строки в начало *Симяпроекта*App::InitInstance:

```
if (!AfxOleInit( )) {
    AfxMessageBox(IDP_OLE_INIT_FAILED);
    return FALSE;
}
```
3. Добавьте строчку в файл строчных ресурсов (с названием *имяпроекта*.RC). Для этого откройте ресурс “String Table”, выполните команду Resource > New String (в версии 1.5 для этого используется AppStudio). Присвойте значение ID: “IDP_OLE_INIT_FAILED” и значение Caption: “Не удалось инициализировать OLE. Проверьте правильность версии OLE-библиотек.” Закройте и сохраните файл ресурсов.

Создание класса поддержки MapInfo

В диалоге PROJECT > CLASSWIZARD (BROWSE > CLASSWIZARD в версии 1.5) откройте раздел OLE Automation и нажмите на кнопку “Read Type Library”. Найдите в Вашем каталоге MapInfo файл MAPINFOW.TLB. Нажмите ОК, и будет создан класс, с помощью которого Вы можете обращаться к MapInfo через механизм управления объектами OLE (OLE Automation).

Откройте главный файл с текстом программы (с названием .CPP) и добавьте в него следующие строки.

После всех директив #includes:

```
#include "MapInfow.h"
```

Сразу за объявлением “*Симяпроекта*App theApp” добавьте объявление переменной:


```
DMapInfo mapinfo;
```

Ближе к концу *СимьяпроектаApp::InitInstance*, но перед вызовом *OnFileNew()*:

```
mapinfo.CreateDispatch("MapInfo.Application");
```

Откройте файл MAPINFOW.H и добавьте в конец файла следующие строки:

```
extern DMapInfo mapinfo;
#include "маршрут-к-каталогу\mapbasic.h"
```

Если Вы работаете в Visual C++ 1.5, Вы должны еще вручную добавить названия OLE-библиотек в командную строку сборки (link command line); Visual C++ 2.x делает это самостоятельно. Для этого нужно выполнить команду Options > Project... и нажать на кнопку Linker.... Выберите переключатель "Common to both" в окошко "Libraries:" слова:

```
comobj, storage, ole2, ole2disp, ole2nls, mfco-  
leui
```

Тестирование

Добавьте еще одну строчку в конец функции *СимьяпроектаApp::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. Это означает, что MapInfo успешно запустилась через механизм OLE Automation. Позже тестирующую строку "::MessageBox..." можно закомментировать или удалить.

Глобальная инициализация

Переопределение "быстрых" 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 дублирования окна. Ее нужно удалить из быстрого меню, чтобы она не вводила в заблуждение пользователей Вашего приложения.

Для этого ближе к концу текста *СимьяпроектаApp::InitInstance* сразу после вызова *CreateDispatch* добавьте следующие строки:

```
// удалить вызов Справочной системы
mapinfo.Do("Set Window Help Off");
// Удаление команды дублирования из "быстрого меню"
mapinfo.Do("Create Menu \"MapperShortcut\" ID 17 as
\"(-\\\"");
```

Здесь же можно добавить другие инструкции, сокращающие объем сервиса, идущего от MapInfo.

Переподчинение диалогов MapInfo

Очень важно научиться переподчинять диалоги MapInfo, появляющиеся из окна Вашего приложения, особенно, если они предназначены 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:

```
mapinfo.Do("Note \"Привет от MapInfo\"");
```

в текст функции `СимяпроектаApp::InitInstance` сразу после вызова `OnFileNew()`. Это приведет к тому, что `MapInfo` покажет один из своих стандартных диалогов изнутри прикладной программы

После организации подобного переподчинения рекомендуется провести промежуточное тестирование.

Добавление окна Карты

Теперь, когда MFC-приложение заработало и Вы убедились, что к `MapInfo` можно обращаться через OLE Automation, пора сделать то, ради чего все это затевалось – добавить окно Карты в приложение.

Откройте диалог `Project > ClassWizard` (или `Browse > ClassWizard` в версии 1.5). Выберите класс представлений (`СимяпроектаView`) и раздел “`Message Maps`”. В самом левом окошке списка выберите объект “`СимяпроектаView`”.

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 (`имяпроектаVW.H`) добавьте строки:

```
unsigned long m_windowid;
HWND         m_windowhwnd;
```

В текст файла представлений (`имяпроектаVW.CPP`) добавьте строки после всех директив `#includes`:

```
#include "MapInfow.h"
```

В конструкторе (`СимяпроектаView::СимяпроектаView`) инициализируйте переменные:

```
m_windowid = 0;
m_windowhwnd = 0;
```

В метод `OnCreate` добавьте следующий отрывок после вызова `CView::OnCreate`:

```
//стиль для окна Карты должен быть ClipChildren
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 текст метода OnDestroy добавьте перед вызовом CView::OnDestroy:

```

if (m_windowhwnd) {
    ::DestroyWindow(m_windowhwnd);
    m_windowhwnd = NULL;
    m_windowid = 0L;
}

```

В текст метода OnSize добавьте 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 элементы меню добавляются с помощью описанной ниже процедуры. Пример показывает, как добавить команду Карты > Управление слоями.

1. Откройте файл ресурсов (*имяпроекта.RC*), откройте ресурс "Menu" и выберите IDR_MAINFRAME. (В версии Visual C++ 1.5 нужно воспользоваться AppStudio.)
2. Добавьте новое меню "Карта" и команду "Управление слоями" и сохраните RC-файл.
3. В диалоге Project > ClassWizard (Browse > ClassWizard... в версии 1.5) откройте раздел "Message Map" и выберите *СимяпроектаView* 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” и принимая стандартные имена.

4. В тексте класса *СиянпроектаView* Вы увидите, что добавлены обе функции. Добавьте к текстам функций следующие строки.

```
void СиянпроектаView::OnMapLayercontrol( )
{
    mapinfo.RunMenuCommand(M_MAP_LAYER_CONTROL);
}

void СиянпроектаView::OnUpdateMapLayercontrol(CCcmdUI*
pCmdUI)
{
    pCmdUI->Enable(m_windowid);
}
```

Добавление кнопок и процедур-обработчиков

Все кнопки на инструментальные панели могут быть добавлены описанным ниже способом. Этот пример показывает, как добавить кнопки MapInfo: Стрелку, Ладощку и обе Лупы. Для удобства мы также добавим их в новое меню “Программы” (или “Tools”); позволяет добавить их в инструментальную панель несколько более простым способом, используя ClassWizard.

1. Сначала, следуя приведенным выше инструкциям добавления команды меню, 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.
3. Откройте строчный ресурс 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,

например, номеру ID_TOOLS_SELECTOR соответствует “Выбрать объект на Карте\пСтрелка”.

4. В тексте MAINFRM.CPP найдите массив UINT BASED_CODE buttons[] типа static и вставьте ID-константы в этот массив в том порядке, в котором они появляются в растровом ресурсе.
5. Чтобы корректно работать с кнопками, мы должны следить за тем, какая из них выбрана в данный момент. В текст файла *СимьяпроектаView* добавьте объявление целой переменной, которая будет хранить значение выбранной кнопки:

```
int m_eMouseMove;
```

6. Эту переменную нужно инициализировать в конструкторе классов, чтобы задать начальную нажатую кнопку на экране. Для этого нужно пользоваться заданными в MapInfo константами.

```
m_eMouseMove = 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 OnUpdate-процедуре и задайте переменные m_eMouseMove соответственно процедурам OnTools*ИмяИнструмента*. Ваши добавления должны выглядеть примерно так:

```
void СимьяпроектаView::OnToolsSelector( )
{
    m_eMouseMove = M_TOOLS_SELECTOR;
    mapinfo.RunMenuCommand(M_TOOLS_SELECTOR);
}

void СимьяпроектаView::OnToolsGrabber( )
{
    m_eMouseMove = M_TOOLS_RECENTER;
    mapinfo.RunMenuCommand(M_TOOLS_RECENTER);
}

void СимьяпроектаView::OnToolsZoomin( )
{
```

```

m_eMouseMove = M_TOOLS_EXPAND;
mapinfo.RunMenuCommand(M_TOOLS_EXPAND);
}
void СумяпоектаView::OnToolsZoomout( )
{
m_eMouseMove = M_TOOLS_SHRINK;
mapinfo.RunMenuCommand(M_TOOLS_SHRINK);
}
void СумяпоектаView::OnUpdateToolsSelector(CCmd-
dUI* pCmdUI)
{
pCmdUI->SetRadio(m_eMouseMove == M_TOOLS_SELECTOR);
pCmdUI->Enable(m_windowid);
}
void СумяпоектаView::OnUpdateToolsGrabber(CCmdUI*
pCmdUI)
{
pCmdUI->SetRadio(m_eMouseMove == M_TOOLS_RECENTER);
pCmdUI->Enable(m_windowid);
}
void СумяпоектаView::OnUpdateToolsZoomin(CCmdUI*
pCmdUI)
{
pCmdUI->SetRadio(m_eMouseMove == M_TOOLS_EXPAND);
pCmdUI->Enable(m_windowid);
}
void СумяпоектаView::OnUpdateToolsZoomout(CCmdUI*
pCmdUI)
{
pCmdUI->SetRadio(m_eMouseMove == M_TOOLS_SHRINK);
pCmdUI->Enable(m_windowid);
}
}

```

Обработчики ошибок MapInfo

MapInfo пересылает информацию об ошибках в приложение, использующее Интегрированную Картографию, посредством MFC-класса COleDispatchException. MapInfo 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 Class Wizard, 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 Вашем
        // приложении.
        // 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 файле *СимяпроектаДос.сpp* добавьте настройку диспетчера (Dispatch map) после настройки сообщений (Message map):

```
BEGIN_DISPATCH_MAP(СимяпроектаДос, CDocument)
//{{AFX_DISPATCH_MAP(СимяпроектаДос)
//ВНИМАНИЕ: здесь ClassWizard добавит или удалит настроечные
макросы
//Не редактируйте содержимое этих автоматически созданных
фрагментов
// кода
//}}AFX_DISPATCH_MAP
END_DISPATCH_MAP()
```

Добавьте в текст файла *СимяпроектаДос.сpp* конструктор *СимяпроектаДос*:

```
EnableAutomation( );
AfxOleLockApp( );
```

Добавьте в текст файла *СимяпроектаДос.сpp* деструктор *СимяпроектаДос*:

```
AfxOleUnlockApp( );
```

Добавьте в текст файла *СимяпроектаДос.h* раздел "Dispatch" после разметки сообщений:

```
// Автоматически созданные функции OLE-диспетчера
//{{AFX_DISPATCH(СимяпроектаДос)
//ВНИМАНИЕ: здесь ClassWizard добавит или удалит функции
//Не редактируйте содержимое этих автоматически созданных
фрагментов
// кода
//}}AFX_DISPATCH
DECLARE_DISPATCH_MAP()
```

Внимание: 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 можно устанавливать только 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 *СимяпроектаДос* список сообщений для диспетчера, то тогда можно установить указатель на уведомление в конструкторе *СимяпроектаДос* 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 *СимяпроектаДос*). Выберите "Add Method" и задайте имя метода "WindowContentsChanged", возвращаемый тип "SCODE" и список аргументов "long lWindowID". После нажатия на ОК и выхода из диалога Class Wizard автоматически обновляет файлы *СимяпроектаДос.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 примерными программами, поставляемыми в комплекте со средой разработки MapBasic.

- SAMPLES\VB\FINDZIP: программа на языке Visual Basic, используемая как примерный образец для разработки собственных приложений.
- SAMPLES\VB\VMAPTOOL: Visual Basic-программы, демонстрирующие более сложные примеры интеграции с Visual Basic 4.0 Professional Edition.
- 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.

Приложение А: Примеры программ

В поставку MapBasic включены следующие программы:

Внимание: К моменту печати этой книги могут появиться новые программы, распространяемые MapInfo вместе с исходными текстами.

Примеры из каталога (папки) \MAPBASIC

ANIMATOR: демонстрация использования слоя анимации для ускорения перерисовки окна Карты.

APPINFO: выводит информацию о загруженных в данный момент приложениях MapBasic.

AUTOLBL: создает “подписи”, помещая текстовые объекты на Косметический Слой (так, как это делали ранние версии MapInfo).

COGOLINE: рисует линию заданной длины под заданным углом.

DMSCNVRT: преобразование данных из колонки с градусами-минутами-секундами в колонку с градусами в десятичной форме и обратно.

GEOREG: открывает растровые изображения GeoTIFF, регистрирует изображение, после чего оно может быть показано в MapInfo и отображает изображение формата GeoTIFF.

GRIDMAKR: создает координатную сетку из параллелей и меридианов.

ICONDEMO: демонстрация встроенных пиктограмм MapInfo 4.0, используемых для создания инструментальных панелей.

LEGENDS: дает возможность работать с несколькими окнами Легенды. (Стандартный интерфейс MapInfo позволяет иметь только одно окно Легенды.)

ROTATE: поворачивает выбранные объекты типа линии, многоугольника или области.

R_BUFER: создает множественные кольцевые буферы.

SEAMMGR: создание и работа со сшитыми таблицами.

SHIELDS: рисует декоративные рамки вокруг текстовых объектов, по форме похожие на щит.

SRCHREPL: производит поиск и замену в таблице.

SYMBOL: редактор, позволяющий просматривать символы MapInfo 3.0 и добавлять или удалять новые.

Примеры из каталога (папки) SAMPLES\VB

FINDZIP: демонстрация использования интегрированных элементов MapInfo, таких как окно Карты, в программе, написанной на VisualBasic.

VMARTOOL: демонстрация дополнительных возможностей встраивания MapInfo, таких как обратные вызовы функций (callbacks). Для работы требуется VisualBasic 4.0 Professional Edition.

Примеры из каталога (папки) SAMPLES\MFC

FINDZIP: демонстрация использования интегрированных элементов MapInfo в программе, написанной на C++ с использованием библиотеки классов MFC.

Примеры из каталога (папки) SAMPLES\PWRBLDR

CAPITALS: демонстрация использования интегрированных элементов MapInfo в программе, написанной с использованием пакета Power Builder. Runtime-модули Power Builder в комплекте поставки не прилагаются, так что для запуска этого приложения нужно, чтобы библиотеки Power Builder были уже установлены на компьютере.

Примеры из каталога(папки) SAMPLES\DELPHI

TABEDMAP: демонстрация использования интегрированных элементов MapInfo в программе, написанной с использованием Delphi.

Примеры из каталога(папки) SAMPLES\MAPBASIC\SNIP-PETS

Здесь содержатся программы, включавшиеся в комплекты поставки предыдущих версий MapInfo.

ACAD: связь с AutoCAD для Windows с помощью DDE.

ADDNODES: добавляет узлы к объектам. Это может оказаться полезным при изменении проекции карты; большее количество узлов сокращает вероятность появления промежутков между большими областями, граница между которыми является длинной прямой линией.

AE_Handler: демонстрирует как приложение MapBasic может действовать в качестве Apple Event server.

Bigdialog: демонстрирует XFNCN которое возвращает размер экрана в момент работы программы.

Checkmem: демонстрирует Проверку Памяти XCMD которая определяет свободную память.

examples directory (rpcclient.mb, rpcserver.mb, mi*.c): простые примеры как использовать RPC в качестве клиента, сервера.

flags directory (flags.mb, micli_flags.c): показывает растровые символы флагов стран.

GEOCODE: демонстрирует геокодирование в MapBasic.

GEOSCAN: просмотр таблицы для оценки объема работы при геокодировании.

GET_TAB: это модуль, а не программа. GET_TAB содержит подпрограммы для показа в диалоге списка открытых таблиц. Пример использования модуля GET_TAB Вы можете найти в тексте программы **OVERVIEW**.

greatcircle directory (greatcir.mb, micli_gcir.c): определяет путь по дуге большого радиуса на поверхности Земли между 2-мя точками.

LONGLATS: открывает диалог, в котором значения градусов-минут-секунд преобразуются в десятичные единицы. Для того, чтобы создать выполняемый файл, используйте файл проекта **LLPROJ.MBP**.

nfslist: показывает использование RPC для коммутации с NFS mount daemon.

NVIEWS: программа, позволяющая создавать Именованные Виды Карты (запоминается центр и размер окна). Списком именованных видов можно управлять в диалоге "Именованные Виды". Для того, чтобы создать выполняемый файл, используйте файл проекта **NVPROJ.MBP**.

OBJINFO: показывает информацию о выбранном объекте.

OVERVIEW: открывает второе окно Карты, в котором рамкой показана область, охватываемая первым окном Карты; таким образом, Вы можете видеть, какую часть всей Карты показывает первое окно. Увеличение или уменьшение первого окна сопровождается соответствующим изменением рамки во втором. Для того, чтобы создать выполняемый файл, используйте файл проекта **OVPROJ.MBP**.

pline directory (pline.c, grid.c): строит полилинию по точкам с координатами из файла.

SCALEBAR: показывает масштабную шкалу в окне Карты. Для того, чтобы создать выполняемый файл, используйте файл проекта **SBPROJ.MBP**.

SendEvent: показывает как посылать Apple Events в другие приложения.

TEXTBOX: программа, используемая как пример в данном руководстве. Текст ее приведен в Приложении 2. Для того, чтобы создать выполняемый файл, используйте файл проекта **TBPROJ.MBP**.

WATCHER: связь через DDE с Microsoft Excel; показывает таблице в Excel глобальные переменные MapBasic-программы.

Примеры из каталога (папки) SAMPLES\MAPBASIC\LIB

AUTO_LIB: это модуль, а не программа. Он включается в другие программы упоминанием в файле проекта MapBasic, и позволяет программе загружаться автоматически каждый раз при запуске Map-Info. Автоматический запуск технически регулируется в Рабочем

Наборе с названием **STARTUP**. Пример использования модуля **AUTO_LIB** приведен в тексте программы **TEXTBOX**; в диалоге этой программы "О программе "Рамка"" есть кнопка, управляющая автоматической загрузкой.

MISTLIB: библиотека различных вспомогательных процедур и функций.

STR_LIB: библиотека различных функций для работы со строками.

Примеры из каталога (папки) `SAMPLES\MAPBASIC\DATA-BASE`

Различные процедуры для работы с удаленными базами данных. Информация об их использовании приведена в Приложении Е.

Приложение В: Действие операторов

Операторы действуют на одну или более величин с целью произведения определенного результата. Операторы могут классифицироваться по типам данных, которые они используют и типам результатов, которые они производят.

Следующие *числовые операторы* действуют на две числовых величины, производя числовой результат.

Оператор	Действие	Пример
+	сложение	$a + b$
-	вычитание	$a - b$
*	умножение	$a * b$
/	деление	a / b
\	деление нацело (без остатка)	$a \setminus b$
Mod	остаток от деления нацело	$a \text{ Mod } b$
^	возведение в степень	$a ^ b$

Два из этих операторов используются в другом контексте. Знак плюс применяется для объединения двух строковых величин в одну. Знак минус действует как показатель того, что число отрицательное.

Амперсанд так же может означать объединение строковых величин.

Оператор	Действие	Пример
-	для отрицательных чисел	$- a$
+	объединение строк	$a + b$
&	объединение строк	$a \& b$

Операторы сравнения производят само сравнение двух величин одного и того же типа и производят логическую величину TRUE или FALSE. Хотя нельзя непосредственно сравнить числовые данные и нечисловые данные (строковые выражения), тем не менее сравниваться могут данные типов Integer, SmallInt, и Float. Операторы сравнения часто используются в выражениях условий, таких как **If...Then**.

Оператор	Возвращает TRUE если:	Example
=	a равно b	a = b
<>	a не равно b	a <> b
<	a меньше чем b	a < b
>	a больше чем b	a > b
<=	a меньше или равно b	a <= b
>=	a больше или равно b	a >= b

Логические операторы действуют с логическими величинами и дают логический результат в виде TRUE или FALSE:

Оператор	Возвращает TRUE если:	Пример
And	оба операнда TRUE	a And b
Or	один из операндов TRUE	a Or b
Not	операнд является FALSE	Not a

Географические операторы действуют на объекты и производят логический результат TRUE или FALSE:

Оператор	Возвращает TRUE если:	Пример
Contains	первый объект содержит центрoид второго объекта	objectA Contains objectB
Contains Part	первый объект содержит часть второго объекта	objectA Contains Part objectB
Contains Entire	первый объект полностью содержит второй объект	objectA Contains Entire objectB
Within	центрoид первого объекта содержит второй объект	objectA Within objectB
Partly Within	второй объект содержит часть первого объекта	objectA Partly Within objectB
Entirely Within	второй объект полностью содержит первый объект	objectA Entirely Within objectB
Intersects	два объекта имеют общую точку	objectA Intersects objectB

Порядок применения операторов в языке MapBasic

Специальный тип операторов это скобки, которые заключают одни выражения внутри себя. Использование скобок позволяет изменять порядок применения операторов в выражениях. Таблица ниже показывает приоритет применения операторов в MapBasic.

Приоритет операторов MapBasic
(Высший приоритет)
скобки
возведение в степень
отрицание
сложение, деление, Mod, целочисленное деление
сложение, вычитание
географические операторы
операторы сравнения, Like
Not
And
Or
(Низший приоритет)

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.

Например, выражение $3 + 4 * 2$ производит результат **11** (умножение старше по приоритету и производится первым). Если поставить скобки $(3 + 4) * 2$ то получим **14** (скобки приведут к тому, что сложение будет выполнено первым).

Автоматическое превращение одних типов данных в другие

Когда Вы создаете выражение из данных разных типов, MapInfo осуществляет автоматическое преобразование типов данных, сохраняя смысл результата. Например, если программа вычитает Дату из другой Даты, то MapBasic будет рассчитывать результат как величину типа Integer (соответствующую числу дней между двумя датами).

Таблица ниже суммирует правила, по которым MapBasic автоматически переводит типы данных. В этой таблице, тип *Integer* представляет целые значения, которые могут быть и целыми переменными и переменными типа короткого целого *SmallInt* или целыми константами *Integer*. Тип *Number* в этой таблице представляет числовое выражение, которое не обязательно является целым.

Оператор	Комбинация операндов	Результат
+	<i>Date</i> + <i>Number</i> <i>Number</i> + <i>Date</i> <i>Integer</i> + <i>Integer</i> <i>Number</i> + <i>Number</i> <i>Other</i> + <i>Other</i>	Date Date Integer Float String
-	<i>Date</i> - <i>Number</i> <i>Date</i> - <i>Date</i> <i>Integer</i> - <i>Integer</i> <i>Number</i> - <i>Number</i>	Date Integer Integer Float
*	<i>Integer</i> * <i>Integer</i> <i>Number</i> * <i>Number</i>	Integer Float
/	<i>Number</i> / <i>Number</i>	Float
\	<i>Number</i> \ <i>Number</i>	Integer
MOD	<i>Number</i> MOD <i>Number</i>	Integer
^	<i>Number</i> ^ <i>Number</i>	Float

Приложение С: Что нового в MapBasic 7.0

В этом разделе суммируются все усовершенствования, сделанные в нынешней версии MapBasic. Более подробно о новинках смотрите в *Справочнике MapBasic* или *Электронной Справке*.

Усовершенствования, представленные в MapBasic 7.0

Новые операторы и Функции

Оператор Control Document Window

- Оператор Create Cutter
- Функция CurrentBorderPen()
- Функция CurrentLinePen()
- Функция Rotate()
- Функция RotateAtPoint()
- Оператор ServerCreateTable
- Функция SessionInfo()
- Оператор Set Style
- Функция TextSize()

Улучшенные операторы и функции

Поддержка шейпфайлов:

- Оператор Register Table

Дублирование структуры таблицы:

- Оператор Create Map

Создание полигонов Вороного:

- Оператор Create Object

Создание новой таблицы на сервере СУБД и Объединение объектов по колонке:

- Оператор Commit Table
- Оператор Create Table
- Оператор Server Create Map

Драйверы СУБД

Функция Server Connect()

Текущие параметры линии:

Функция CurrentPen()

Местоположение файлов:

Функция GetFolderPath\$()

Импорт GML файлов

Оператор Import

Подписывание части объектов:

Функция LayerInfo()

Прозрачность растров:

Оператор Set Window

Дополнительные типы ReadControlValue():

Функция ReadControlValue()

Пользовательская настройка разрешения и поддержка JPEG 2000:

Оператор Save Window

Приложение D: Поддерживаемые типы таблиц ODBC

Типы данных ODBC, поддерживаемые MapInfo:

- 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

Приложение Е: Присвоение геоинформации удаленной таблице

Предварительные условия для хранения/извлечения пространственных данных

Есть четыре предварительных условия для хранения и извлечения точек в удаленной таблице.

1. Сначала координаты для точек должны быть сохранены в столбцах таблицы как числа.
Это можно сделать следующим образом:
 - Использовать существующие данные.
 - Использовать Easyloader для загрузки данных в базу данных. Это приложение должно работать для большинства баз данных. Это задача создания данных и она может быть решена в любой момент.
2. Увеличить эффективность запросов о координатах, проиндексируйте соответствующую колонку. Это задача создания данных, и она может быть выполнена в любое время.
3. MapInfo сохраняет информацию стол, какие именно столбцы являются координатами, в специальной таблице на удаленной базе данных, называемой MapInfo Map Catalog. На каждую БД должен быть один каталог. С помощью приложения **MIODBCAT.MBX** можно создать этот каталог для Oracle7, Sybase, SQL Server, SQL Base или MSAccess. Это приложение можно настроить для любой другой системы управления удаленными БД. Вы можете также создать этот каталог вручную, следуя процедуре, описанной в следующем разделе. Это “одноразовое мероприятие”, и его нужно проделать прежде, чем любые таблицы из такой базы данных могут быть отображены в MapInfo..
4. MapInfo извлекает информацию из каталога о таблицах, используя оператор MapBasic **Server Create Map**. Это делается один раз для каждой задачи и требуется проделать это перед отображением таблицы в MapInfo.

Создание каталога Карт MapInfo

Вы не можете делать ODBC-таблицу “картографической”, если MapInfo Map Catalog не был создан для базы данных, в которой постоянно находится таблица. MapInfo Map Catalog должен быть создан администратором базы данных.

1. Создайте пользователя MAPINFO со значением пароля PASSWORD ***** в конкретной базе данных, где размещены нужные таблицы.
2. Создайте таблицу MAPINFO_MAPCATALOG в базе данных.
3. Оператор **Create table** должен быть эквивалентен следующему оператору MapBasic:

```
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,
                                COORDINATESYSTEM
Char(254),
                                SYMBOL
Char(254),
                                XCOLUMNNAME
Char(32),
                                YCOLUMNNAME
Char(32),
                                RENDITIONTYPE     integer
                                RENDITIONCOLUMN
Char(32)
                                RENDITIONTABLE    Char(32)
```

Важно, чтобы структура таблицы была точно такой, как в этом операторе. Единственное изменение, которое может быть сделано – для баз данных, которые поддерживают varchar или

текстовые типы данных. Эти типы данных могут заменяться Char datatype.

4. Создайте уникальный индекс в TABLENAME в OWNERNAME; только одна таблица для каждого владельца может быть сделана “картографической”.
5. Выберите приоритет, позволяющий исполнять операторы **Select**, **Update** и **Insert** для таблицы MAPINFO_MAPCATALOG. Право удаления должно быть оставлено за администратором базы данных.

Приложение F: Установка и управление данными

Служебные файлы MapInfo Professional и вспомогательные файлы настроек

Вспомогательные файлы - это неисполняемые файлы, содержащие данные, которые используются MapInfo Professional в процессе работы. В версии 6.5 используются следующие файлы и папки:

Имя файла	Описание
mapinfow.prf	Файл настроек
mapinfow.wor	Стандартный рабочий набор
startup.wor	Стартовый рабочий набор
mapinfow.clr	Файл цветов
mapinfow.pen	Файл стилей линий
mapinfow.fnt	Файл символов
custsymb	Директория пользовательских символов
thmtmpl	Директория тематических шаблонов
graphsupport	Директория поддержки графиков

В ранних версиях MapInfo (до 6.5) эти файлы содержались в директории Windows или в директории Program. Начиная с версии 6.5 пользователь сам выбирает папки, в которые следует размещать вспомогательные файлы. Это удобно, например, в том случае, когда при работе с программой приходится использовать проекции, относящиеся к разным версиям MapInfo, - файл "mapinfow.prj". Следующие файлы остались в директории Program:

Имя файла	Описание
mapinfow.abb	Файл сокращений

Имя файла	Описание
mapinfow.prj	Файл проекций
mapinfow.mnu	Файл меню

Запомните:

- Установщик не спрашивает пользователя где он хочет разместить эти файлы приложений.
- Установщик всегда запускается одинаково, независимо, имеет ли пользователь MI Pro 6.0 или нет.
- Здесь не происходит "обновления" установки до версии 6.5 (т.е. Вы не можете установить 6.5 в ту же директорию, где и 6.0, установщик выдаст ошибку).
- Разработчики приложений могут перемещать или копировать файлы куда хотят, но MI Pro 6.5 будет искать их в следующем порядке:
appdata_dir, local_appdata_dir, pref_dir, program_dir

Глоссарий терминов

Полезно будет знать следующие определения:

<user profile root>

Пользовательская иерархия папок. Каждый пользователь имеет доступ производить запись в подчиненные папки. Их размещение зависит от версии Windows:

Windows 2000 : c:\Documents and Settings\

Windows 98: <Windows dir>

Windows NT 4.0: <Windows dir>\profiles\

<My Documents>

Windows 2000: c:\Documents and Settings\\My Documents

Windows 98: c:\MyDocuments

Windows NT 4.0: <Windows dir>\profiles\<username>\personal

Pref_dir

По умолчанию MI Pro записывает в эту папку файлы "mapinfow.prf" и "mapinfow.wor".

6.0: папка Windows

6.5: <user profile root>\Application Data\MapInfo\MapInfo.

Если такой папки не существует, то MI Pro создает ее.

program_dir

В MI Pro 6.0 в этой папке размещено большинство вспомогательных файлов.

6.0: папка, где находится файл mapinfow.exe

6.5: папка, где находится файл mapinfow.exe

appdata_dir

Это папка для конкретного пользователя. В нее можно разместить служебные (вспомогательные) файлы.

6.0: не существует

6.5: <user profile root>\Application Data\MapInfo\MapInfo\Professional\650.

Если такой папки не существует при запуске программы, MI Pro ее не создает - будет выдано сообщение об ошибке, что указан неверный путь.

local_appdata_dir

Эта папка также определяется пользователем (аналогично appdata_dir).

6.0: не существует

6.5: <user profile root>Local Settings\Application Data\MapInfo\MapInfo\Professional\650. Если такой папки не существует при запуске программы, MI Pro ее не создает - будет выдано сообщение об ошибке, что указан неверный путь.

common_appdata_dir

Эта директория предназначена для доступа всем пользователям на компьютере (появилась только в версии 7.0).

6.0: не существует

6.5: не существует

7.0 : <profile root>\All Users\Application Data\MapInfo\MapInfo\Professional\700

mydocs_dir

Отсылает к папке My Documents текущего пользователя.

6.0: не существует

6.5: <My Documents>

search_for_file

Эта функция размещения файлов приложений. Она ищет папки для файлов в следующей очередности:

6.0: pref_dir, home_dir, program_dir

6.5: appdata_dir, local_appdata_dir, pref_dir, program_dir

7.0: appdata_dir, local_appdata_dir, pref_dir, common_appdata_dir, program_dir

Файлы приложений и директории

Следующий список описывает как ищут MI Pro 6.0 и 6.5 файлы приложений и директории.

mapinfow.prf

6.0: Использует `search_for_file`. Независимо от того, откуда был считан файл, всегда записывает в `pref_dir`.

6.5: Использует `search_for_file`. Если найдет, тогда читает файл и запоминает его местоположение.

mapinfow.wor

6.0: Ищет в `pref_dir`, затем в `home_dir`. Загружает первый, который найдет.

6.5: Использует `search_for_file`. Если найдет, тогда читает файл и запоминает его местоположение.

startup.wor

6.0: Загружает из следующих директорий по порядку: `program_dir`, `pref_dir`.

6.5: Загружает из следующих директорий по порядку: `pref_dir`, `appdata_dir`, `local_appdata_dir`, `pref_dir`, `program_dir`. В отличие от других файлов приложений, каждый `startup.wor` который найден, обрабатывается.

mapinfow.clr

6.0: Использует `search_for_file`. Если не находит, то открывает диалог с предложением пользователю найти его самостоятельно.

Если пользователь решил записать собственные цвета, то новый файл с цветами записывается в `pref_dir` (или переписывает существующий файл).

6.5: Использует `search_for_file`. Если не находит, то открывает диалог с предложением пользователю найти его самостоятельно.

Если пользователь хочет записать собственные цвета и файл цветов находится в пользовательской директории, то MI Pro обновит существующий файл. Если файл цветов был считан из программной директории или если пользователь не имеет

доступа, чтобы записать файл, то MI Pro запишет этот файл в the pref_dir.

mapinfow.pen

6.0: Использует search_for_file. Если не находит, то открывает диалог с предложением пользователю найти его самостоятельно.

6.5: Использует search_for_file. Если не находит, то открывает диалог с предложением пользователю найти его самостоятельно.

mapinfow.fnt

6.0: Использует search_for_file. Если не находит, то открывает диалог с предложением пользователю найти его самостоятельно.

6.5: Использует search_for_file. Если не находит, то открывает диалог с предложением пользователю найти его самостоятельно.

custsymb directory

6.0: Предполагает, что файл в program_dir.

6.5: Ищет директорию символов используя search_for_file. Если не находит, то предполагает что файл в program_dir.

thmtmpl directory

6.0: Если директория тематических шаблонов указанная в файле настроек существует, то используется она. В противном случае ропытается создать директорию шаблонов под папкой program_dir. Если не сможет создать под program_dir, то такой директории не будет.

Во всех случаях, включая последний, MI Pro обновляет путь к файлу настроек.

6.5: Если директория тематических шаблонов указанная в файле настроек существует, то используется она. С другой

стороны, ищется папка с шаблонами с помощью `search_for_file`. Если находится, то она и используется. В противном случае, пытаются создать директорию шаблонов под `appdata_dir`, а затем в `program_dir`. В противном случае, директория шаблонов не будет создана. В любом случае, MI Pro не устанавливает путь к файлу настроек.

graphsupport directory

6.0: Использует директорию указанную в настройках, не зависимо от того, существует ли файл настроек. Если указанная директория отсутствует, появится соответствующее сообщение и надо создавать ее пользователю самому.

6.5: Если директория шаблонов, указанная в файле настроек существует, то используется она. С другой стороны, ищется директория поддержки графиков с использованием `search_for_file`. Если она находится, то она и используется. Если не находится, предполагается, что ее нет в `program_dir`, и пользователь получит сообщение об ошибке при попытке создать график.

Внимание: В версии 7.0 программа `search_for_file` включает в поиск `common_appdata_dir`.

Пути к файлам настроек по умолчанию

Следующая таблица показывает настраиваемые пути и сами пути по умолчанию в версиях 6.0/6.5 /7.0:

Путь	6.0 по умолчанию	6.5 по умолчанию
Таблицы	<program_dir>\Data	mydocs_dir
Рабочие наборы	<program_dir>\Data	mydocs_dir
Программы MapBasic	<program_dir>\Tools	<program_dir>\Tools
Файлы импорта	<program_dir>\Data	mydocs_dir
SQL Запросы	<program_dir>\Data	mydocs_dir
Тематические шаблоны	<program_dir>\thmtmpl	appdata_dir\thmtmpl если существует, иначе program_dir\thmtmpl
Сохраненные запросы	<program_dir>\Data	mydocs_dir
Навые поверхности	<program_dir>\Data	mydocs_dir
Файлы Crystal Report	<program_dir>\Data	mydocs_dir
Файлы поддержки графики	<program_dir>\GraphSupport	local_appdata_dir if exists, program_dir otherwise
Поиск папки с таблицами	<program_dir>\Data	mydocs_dir

Путь	7.0 по умолчанию
Таблицы	mydocs_dir
Рабочие наборы	mydocs_dir
Программы MapBasic	<program_dir>\Tools
Файлы импорта	mydocs_dir
SQL Запросы	mydocs_dir
Тематические шаблоны	использует search_for_file затем ищет в program_dir
Сохраненные запросы	mydocs_dir
New Grids	mydocs_dir
файлы Crystal Report	mydocs_dir
Файлы поддержки графиков	использует search_for_file затем program_dir если не находит
Поск папки с таблицами	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 теперь устанавливаются под HKEY_CURRENT_USER.
- Ключи настройки цветов и форматов численных значений, сделанные пользователями, теперь хранятся в HKEY_CURRENT_USER.

Требования установки

Вспомогательные файлы должны быть установлены в директории, как указано в следующей таблице:

Имя файла	версия 6.5
mapinfow.clr	Application Data\MapInfo\MapInfo\Professional\650
mapinfow.pen	Application Data\MapInfo\MapInfo\Professional\650
mapinfow.fnt	Application Data\MapInfo\MapInfo\Professional\650
mapinfow.abb	директория Program
mapinfow.prj	директория Program
mapinfow.mnu	директория Program
custsymb	Application Data\MapInfo\MapInfo\Professional\650
thmtmplt	Application Data\MapInfo\MapInfo\Professional\650
graphsupport	Local Settings\Application Data\MapInfo\MapInfo\Professional\650

Имя файла	версия 7.0
mapinfow.clr	Application Data\MapInfo\MapInfo\Professional\700
mapinfow.pen	Application Data\MapInfo\MapInfo\Professional\700
mapinfow.fnt	Application Data\MapInfo\MapInfo\Professional\700
mapinfow.abb	директория Program
mapinfow.prj	директорияProgram
mapinfow.mnu	директорияProgram
custsymb	Application Data\MapInfo\MapInfo\Professional\700
thmtmpl.t	Application Data\MapInfo\MapInfo\Professional\700
graphsupport	All Users Application Data\MapInfo\MapInfo\Professional\700

Следующие функции MapBasic были добавлены чтобы помочь разместить вспомогательные файлы:

Функция GetFolderPath\$()

Возвращает путь к специальной папке MI Pro или Windows.

Функция LocateFile\$()

Возвращает путь к одному из вспомогательных файлов MI Pro.

Индекс

Символы

\ (backslash)
 деление нацело 295
 \ (backward slash)
 деление нацело 54
 & (амперсанд) 99, 115
 соединение строковых 295
 ' (апостроф) 40
 * (астерикс)
 строки фиксированной длины 43
 умножение 295
 * (умножение) 54
 + (сложение) 54
 + (плюс) 295
 суммирование 54
 прибавление даты 55
 . (период) символ
 десятичный разделитель 51
 / (деление) 54
 / (forward slash)
 формат даты 52
 деление 54
 / (slash)
 деление 295
 = (равенство) 55
 = (равно) 56
 > (больше) 55
 > (больше чем) 56
 >= (больше или равно) 55-56
 ^ (возведение в степень) 54
 ^ (перевод строки) 54
 возведение в степень 295
 p (меньше или равно) 56
 ≠ (не равно) 56

А

Аргументы
 передача значением 68
 передача ссылкой 68
 Арифметические операторы 54
 Адреса, поиск 148

Б

Бесконечные циклы, предотвращение 74
 Битовые (растровые) карты 163
 Буфер, создание 202, 291
 Быстрые меню
 изменение 97
 удаление 97

В

Введение в MapBasic 13
 Ввод/вывод в файлы
 двоичные файлы i/o 182
 копирование файла 180
 наборы символов 183
 определение 178
 произвольный доступ 182
 переименования файла 180
 удаление файла 180
 Ввод/вывод в файлы последовательный
 доступ 180
 Внешние ссылки
 процедуры из других модулей 31
 Windows DLL 222
 Внешние файлы
 DBF (dBASE) 141
 WKS (Lotus) 141
 XLS (Excel) 141
 Вставка
 записи в таблицу 149
 колонок в таблицу 151
 узлов в объект 200
 Выбор
 указание на объект 131
 Вызов внешних процедур 31, 224
 Вызов процедур 66
 Выполнение программы
 ошибки 80
 Высота текста 194

Г

Галочка у элемента меню 94
 Геокодирование 292
 автоматическое 148
 интерактивное 148

MapMarker 149
Географические операторы 58, 214
Глобальные переменные 46
Горячая связь 237
График (окно) 121, 122

Д

Данные
 структуры 45
Даты и их обработка 55
Даты-константы 52
Двоичные файлы 182
Двоичный доступ (ввод/вывод) 178
Действия мышью
 указание и перемещения 128
 двойное указание на окошко списка 112
Деление нацело 295
Дескриптор оператора
 определение 171
Дескриптор соединения
 определение 171
Дескрипторы 225
Десятичный разделитель
 в числовых константах 51
Директивы компилятора 76
Диалог "Открыть сразу" 42 135
Диалог с процентом выполнения
"Минуточку" 104
Диалоги
 элементы окна 107
Диалоги, присутствие на экране 115
Диалоги, новые
 задание начальных значений 111
 закрытие 115
 клавишные комбинации 114
 недоступные элементы 113
 примеры 105
 положение элементов 106
 размеры элемента 106
 реакция на действия пользователя 112
 считывание установок 111
 списки, созданные из массивов 113
Диалоги, пользовательские 105
 примеры 106
Диалоги, стандартные
 запрос с кнопками ОК/Отмена 103
 открытие файла 103

 скрыть индикатор выполнения 137
 простое сообщение 102
 процент выполнения 104
Диалоги, новые
 списки, составленные из строчек 113
Диалоги, стандартные
 сохранение файла 104
Длина объекта 216
Добавление колонок в таблицу 150
Добавление узлов к объекту 200
Доступ к удаленной базе данных 171

Е

Единицы измерения
 бумажные 213
 площади 213
 расстояний 213

З

Завершение программы 65
Задержка
 при DDE-обмене 232
Запрос с подтверждением 103
Запуск программы 26
 из MapInfo 15
 из среды MapBasic 37
 стартовый Рабочий Набор 135
Звуковой сигнал 223
Значение цвета
 выбор объектов по цвету 196

И

Идентификаторы, определения 76
Идентификатор окон 116
Изменения в удаленных базах данных 174
Изменяемый объект 205
Изображения (растровые) 163
Изображения формата SPOT 163
Имена и расширения файлов 7
Имена каталогов 180
Индекс, создание 150, 152
Индикатор выполнения, скрыть 137
Инструментальные панели
 добавление кнопок 128

- кнопки запуска (PushButtons) 125
- кнопки-инструменты (ToolButtons) 125
- кнопки-переключатели (ToggleButtons) 125
- ICONDEMO.MBX 131
- новые пиктограммы 228
- размещение на экране 134
- создание новых панелей 127
- Интегрированная Картграфия
 - изменение размера окна 249
 - Инструментальные панели 250
 - обработка ошибок 252
 - пример 244
 - примеры 288
 - переподчинение окна Легенды 248
 - уведомления 255
- Интерфейс
 - диалоги, новые 105
 - диалоги, стандартные 102
 - меню 91
 - обзор 88
 - окна 116
- Использование справочной системы 18

К

- Карта (окно) 117
- Каталог Карт 304
- Километры 211
- Клавиши-акселераторы 254
- Клавишные комбинации
 - в диалогах 114
 - в меню 99
- Клавишные сочетания 22
- Клиент/сервер
 - доступ к данным 171
 - протокод DDE 230
- Кнопки
 - добавление подсказок 133
- Кнопки-инструменты) 125
- Кнопки-переключатели 125
- Количество
 - объектов 191
 - полигонов в области 191
 - сегментов в ломаной 191
- Количество выбранных строк 153
- Количество открытых окон 117
- Колонки
 - создание имени-синонима 144
 - Obj (object) 147, 189
 - RowID 146

- Командная строка 27, 135
- Координатная сетка 291
- Координатные системы
 - карты мира 211
 - планы 211
 - Отчет 211
- Копирование программ из Справки 18
- Комментарии 40
- Компиляция
 - без открытия файла 34
 - из командной строки 27
- Компиляция программ
 - в активном окне 25
- Компиляция программы
 - в активном окне 15
- Константы
 - числовые 51
 - даты 52
 - логические 51
 - определение 48
 - строчные 51
- Контроль переменных
 - MapInfo в роли DDEсервера 237
- Контрольные точки при отладке 81
- Косметический слой 156
- Круговые диаграммы
 - в окне Графика 121, 122
 - условное выделение 118
- Курсор (позиция в таблице) 143
- Курсоры 134

Л

- Линковка 27
- Личный каталог 102
- Логические константы 51
- Логические операторы 57
- Локальные переменные 32, 42

М

- Меню Окно 37
- Меню под правой кнопкой
 - изменение 97
 - удаление 97
- Меню Поиск 34
- Меню Правка 34
- Меню Сборка 36

Меню Справка 37
Меню Файл 33
Меню, настройка
 изменение строки меню 96
 клавишные комбинации 99
 файл MAPINFOW.MNU 100
Меню, создание
 добавление элементов меню 91
 изменение элемента меню 94
 новое меню 93
 удаление элементов меню 92
Массивы
 размер 44
 объявление 44
Метаданные 166
 ключи 166
Метрические единицы измерения 211
Метки в программах 62
Многопользовательская среда 157

Н

Наборы символов 183
Наложение областей 220
Найти и заменить 34
Национальные наборы символов 183
Непрерывная тематическая заливка, поддержка 118
Номер оператора
 определение 171
Номер соединения
 определение 171
Номера строк в программе 36

О

Области
 наложение 220
Область видимости функции 76
Обработка ошибок при выполнении программы 83
Объединение
 областей 203
Объединение строковых переменных
 & (оператор) 295
 + (оператор) 295
Объединение строк
 & оператор 295

 + оператор 295
Объединение таблиц 219
Объектная модель механизма управления объектами OLE 264
Объекты
 атрибуты 191
 координаты 191
 стили 192
Объекты типа "Рамка" 199
Объекты, изменение
 размещение в таблице 201
 добавление узлов 200
 добавление узлов 206
 комбинация 203
 положения 204
 стирание части 205
 стиль 205
 типа объекта 205
Объекты, создание
 буфер 202
 размещение в таблице 201
 на основе других объектов 202
 операторы 199
 функции 199
Объекты, удлинение 190
Ограничения размеров 24
Ограничения памяти 24
Окна, настройка
 График 121, 122
 Районы 122
 размер и положение 117
 перерисовка 137
 окно Информации 122
 слой анимации 118, 119
 Список 120
 Сообщения 122
Окна,настройка
 Карта 117
Окно Информации
 настройка 122
 только для чтения 124
Окно Карты
 подписи 206
Окно легенды,работа 291
Окно MapBasic 41
Окно Списка 120
Операторы
 сравнения 56
 даты 55

- географические 58
 - логические 57
 - числовые 54
- Оператор Select 214, 215
- Оператор Set Table 170
- Операторы
 - числовые 54
 - географические 58, 214
 - логические 57
 - обработка дат 55
 - определение 49
 - порядок выполнения 58
 - сравнение 55
 - строчные 55
- Оптимизация
 - работа с таблицами 175
- Оптимизация производительности
 - интерфейс пользователя 137
 - процедуры-обработчики 74
- Организация программы 78
- Остановка программы 65
 - неявные таблицы 156
- Открытие нескольких файлов 31
- Открытие таблицы 140
- Отладка программ 81
- Отчет
 - координаты объектов 212
- Ошибки
 - компиляции 25
 - обработка 83
 - при выполнении 80, 149

П

- Параметры
 - передача значением 68
 - передача ссылкой 68
- Перемещение объектов 204
- Пересечение
 - линий 206
 - областей 203
 - оператор Intersects 58
 - улиц 148
- Пиктограммы для кнопок Инструментальной панели 228
- Пиктограммы для кнопок инструментальных панелей 128
- Пиктограммы” 291
- Переменная

- объявление 42
 - определение 42
- Переменные
 - глобальные 46
 - ограничения на имена 43
 - область определения 47
 - объектные 188
 - список типов 43
 - стиль переменной 195
- Переменные типа Alias 144
- Пересечение
 - Intersects оператор 58
- Плотность точек 118
- Подзапрос 217
- Поворот графических объектов 291
- Повышение скорости
 - процедуры-обработчики 74
- Поддержка тематических поверхностей 118
- Подписи
 - преобразование в текст 209
 - на картах 199
 - на Картах 206
- Поиск и замена 34
 - пример программы 291
- Поиск по адресу 148
- Поиск по номеру строки 36
- Порядок исполнения операторов 297
- Последовательный доступ 180
- Последовательный доступ (ввод/вывод) 178
- Преобразование ГМС в Градусы 291
- Преобразование типов 54
- Примеры программ
 - Интегрированная Картография 288
- Присвоение значений переменным 42
- Проекция 118
- Произвольный доступ 182
- Произвольный доступ (ввод/вывод) 178
- Прописные и строчные буквы 40
- Пропорциональное обобщение данных 220
- Процедуры
 - рекурсия 69
 - вызов 66
 - передача параметров 68
 - обработка событий 69
 - определение 66
 - Main 66
- Прямой доступ к базам данных 174

Р

- Районирование (окно) 122
- Разорвать соединение с удаленной базой 174
- Раскодирование 190
- Растровые изображения и таблицы 163
- Рабочие наборы
 - использование в как примеров программ 18
- Рабочий Набор
 - стартовый 135
- Размер текста 194
- Размерные символы 118
- Режимы рисования 128
- Рекомендации по производительности
 - работа с таблицами 175
 - процедуры-обработчики 74
- Рекомендации по повышению производительности
 - интерфейс пользователя 137
- Рекурсия 69
- Реляционное объединение 190, 219

С

- С язык
 - примеры программ 288
- Сборка
 - без открытия файла 34
 - из командной строки 27
- Сборка проекта
 - после выбора проекта 30
- Свойства
 - объекта Application 266
- Связанные таблицы 174
- Связь с удаленной базой данных 171
- Связывание 29
- Система координат
 - координаты окна Отчета 157
- Системные события 70
- Системы координат 211
- Скорость работы, повышение
 - работа с таблицами 175
- Скорость, увеличение
 - интерфейс пользователя 137
- Слияние
 - областей 203
- Слияние объектов 203

Слои

- добавление/удаление 118
- Косметический слой 156
- условное выделение 118
- Слой анимации 118, 119, 137, 291
- Совмещение узлов 254
- Соглашения 9
- События
 - выбор команды меню 91
- События, обработка
 - специальные процедуры 71
 - изменение выборки 132
 - интерфейс пользователя 89
 - определение 70
 - специальные процедуры 70
- Создание
 - новых инструментальных панелей 125
- Сообщения
 - окно 122
- Сортировка строк в таблице 149
- Сохранение в удаленных базах данных 174
- Справочная система, создание
 - в среде Windows 237
- Сравнение (операторы) 55
- Сравнение строк по шаблону 55
- Стартовый Рабочий Набор 135
- Стирание части объекта 205
- Стили (Pen, Brush, Symbol, Font) 192
- Стили пера (Pen) 192
- Стили символов (Symbol) 192
- Стили, сравнение 193
- Стили шрифта (Font) 192-194
- Стили штриха (Brush) 192
- Столбчатые графики
 - в окне Графика 121, 122
- Строчные константы 51
- Строка сообщений
 - в Интегрированной Картографии 255
- Строки в таблице
 - вставка новых 149
 - номер (RowID) 146
 - обновление 149
 - отображение в окне Информации 122
 - сортировка 149
 - установка текущей строки 142
- Строки постоянной длины 44
- Строки фиксированной длины 44

Структуры 45
 тематическое выделение 118
 Строчные переменные
 фиксированной и постоянной длины 44
 Строчные операторы 55
 Сшитые таблицы 169
 Схема
 SQL Server-операторы MapBasic 172

Т

Таблица
 колонка Obj (object) 189
 Таблицы
 добавление временных колонок 151
 добавление вычисляемых колонок 151
 добавление новых колонок 151
 закрытие таблиц QueryN 153
 запись значений 149
 количество открытых таблиц 152
 колонка Obj (object) 147
 Косметический слой 156
 метаданные 166
 на основе электронных таблиц и баз данных
 141
 номера строк 146
 объединение 219
 Отчеты 156
 открытие 140
 присоединение графических объектов 150
 растровые изображения 163
 редактирование 157
 создание 150
 структура 152
 структура, изменение 150
 чтение значений 142
 файлы-компоненты 162
 Selection 153
 Таблицы QueryN
 закрытие 153
 Текстовые редакторы 26
 Текстовые объекты 192, 205
 Текущая запись, установка 143
 Тематические карты 118
 Техническая поддержка
 автоматическая по факсу 12
 MapInfo Test Drive Center 11
 сервисы 10–11
 Техническая поддержка 19
 Типографские соглашения 9

Типы данных, задаваемые пользователем 45
 Точки пересечения 206
 Точки, сохранение в удаленных базах данных
 174
 Требования 6
 Требования к компьютеру 6
 Требования к оборудованию 6
 Требования к системе 6

У

Уведомления 255
 Удаление
 элементов меню 92, 100
 части объекта 205
 индекса 152
 колонок из таблицы 150
 меню 96, 100
 файла 180
 Узлы, добавление 200, 206, 292
 Узлы, максимальное число 200
 Указание и перемещения с помощью мышки
 128
 Управление программой 62
 оператор Do Case 61
 Установка, инструкции 6
 Установки в WIN.INI 225

Ф

Файл проекта
 определение 28
 примеры 29
 сборка 30
 создание 29
 Файл MAPINFOW.MNU 100
 Файлы электронных таблиц, открытие 141
 Файлы DBF(dBASE) 141
 Файлы Excel 141
 Файлы FoxBase 141
 Файлы Lotus 141
 Файлы Справки, использование 18
 Файлы WKS, открытие 141
 Файлы XLS, открытие 141
 Файлы, внешние
 формата
 BIL (SPOT) 163

формата GIF 163
формата JPG 163
формата PCX 163
формата Targa 163
формата TIFF 163

Функции, созданные пользователем 74

Функция CommandInfo()
 идентификатор элемента меню 98

Ц

Цвет, значения 196

Целочисленная математика 54

Целочисленные операторы 54

Циклы

 Do Case 61
 Do...Loop 64
 For...Next 63

Ч

Число

 узлов в объекте 200

Числовые константы 51

 шестнадцатиричные 51

Числовые операторы 54

Числовые операторы 54

Ш

Шаблоны при сравнении строк 55

Э

Элемент диалога ListBox 113

Элемент диалога MultiListBox 113

Элемент диалога PopupMenu 113

Элементы диалога

 CheckBox 110
 EditText 108
 кнопка BrushPicker 109
 кнопка CancelButton 110
 кнопка FontPicker 109
 кнопка OKButton 110
 кнопка PenPicker 109
 кнопка SymbolPicker 109
 кнопки 110
 GroupBox 108
 ListBox 109
 MultiListBox 109
 PopupMenu 110

 RadioGroup 109
 StaticText 108
Элементы окна диалога 107

А

Add Column (оператор) 220

Add Map Layer (оператор) 118

Alter Button (оператор) 126

Alter ButtonPad (оператор) 126, 229

Alter Control (оператор) 113

Alter Menu (оператор) 91

Alter Menu Bar (оператор) 93

Alter Menu Item (оператор) 94

Alter Object (оператор) 200, 204

Alter Table (оператор) 150

And (оператор) 57

Any() (оператор) 218

Area() (функция) 192, 217

Ask() (функция) 103

auto_lib.mb (программа-пример) 137

AutoLabel оператор 199

В

Beeping (звуковой сигнал)

 потому что окно заполнено 24

Between (оператор) 55- 56

BIL-файлы (SPOT-изображение) 163

Brush (стиль) 192

С

Close Window (оператор) 117, 237

CommandInfo() (функция)

 DDE 236

 инструментальные панели 126

 определение результатов поиска 148

 определение нажатия на ОК в диалоге 105

 определение двойного щелчка мыши 112

Commit (оператор) 122, 149

Commit Table (оператор) 118

Comparison (оператор) 56

Contains (географический оператор) 216

Contains (оператор) 58

Continue (оператор) 81
 Create ButtonPad (оператор) 126, 129, 229
 Create Frame (оператор) 199
 Create Index (оператор) 150
 Create Map (оператор) 150, 190
 Create Menu (оператор) 93
 Create Menu Bar (оператор) 96
 Create Text (оператор) 194
 CreateCircle() (функция) 200
 CurDate() (функция) 50, 55

D

Date константы 52
 Date операторы 55
 DDE, поведение клиента 231, 236
 DDE, поведение сервера 233, 236
 Declare Function (оператор) 75, 222
 Declare Sub (оператор) 66, 222
 Define (оператор) 76
 Dim (оператор) 42
 DLL
 библиотека User 223
 библиотека Kernel 225
 объявление 222
 определение 222
 передача параметров 223
 размещение пиктограмм 228
 DLLs
 16 vs. 32 bit 227
 Do Case (оператор) 61
 Do...Loop (оператор) 64
 Drop Map (оператор) 190

E

End Program (оператор) 65
 EndHandler (процедура) 70
 EndHandler (процедура) 71
 EOF() (функция) 181
 EOT() (функция) 143
 Err() (функция) 83
 Error\$() (функция) 83
 ERRORS.DOC 253

F

Fetch (оператор) 142, 204
 FileExists() (функция) 180
 FileOpenDlg() (функция) 103
 FileSaveAsDlg() (функция) 104
 Font (стиль) 192
 For...Next (оператор) 63
 ForegroundTaskSwitchHandler (процедура) 71
 Format\$() (функция) 121
 FrontWindow() (функция) 117
 Function...End Function (оператор) 75

G

Get (оператор) 182
 GetMetaData\$() (функция) 167
 GetSeamlessSheet() (функция) 170
 GIF-файлы 163
 GoTo (оператор) 62

H

Help-файлы, в среде Windows 237

I

Include (оператор) 77
 Input # (оператор) 181
 Insert (оператор) 149, 201
 Intersects (географический оператор) 216
 Intersects (оператор) 58

J

JPG-файлы 163

K

Kernel (Windows DLL) 225
 Kill (оператор) 180

L

LabelFindByID() (функция) 208
 LabelFindFirst() (функция) 208
 LabelFindNext() (функция) 208
 Labelinfo() (функция) 208

Like (оператор) 55
Line Input # (оператор) 181
Logical (оператор) 57

M

Main (процедура) 66
MapInfo
 установка документации 8–9
MapInfo Test Drive Center 11
MapInfo-L
 архив 12
MAPINFOW.MNU (файл) 100
MapMarker (пакет) 149
Методы
 объект Application 267
MFC
 примеры программ 288
Microsoft Excel
 DDE -обмен 231
 открытие файла 141
Microsoft Visual Basic 236
Mod (деление по модулю) 54
Mod оператор 295

N

NoSelect ключевое слово 74
Not (оператор) 57
Note (оператор) 102

O

Object (переменные) 188
ObjectGeography() (функция) 191
ObjectInfo() (функция) 191, 195–196
ObjectLen() (функция) 192, 216
ODBC соединение
 поддерживаемые типы данных 301
OnError (оператор) 83
Open File (оператор) 178
Open Window (оператор) 116, 237
Or (оператор) 57

P

Pack Table (оператор) 147

PCX-файлы 163
Pen (стиль) 192
Perimeter() (функция) 192
PowerBuilder
 примеры программ 289
Print # (оператор) 182
Print (оператор) 122
ProgressBar (оператор) 104
PushButtons (кнопки запуска) 125
Put (оператор) 182

R

ReadControlValue() (функция) 111, 114
ReDim (оператор) 44
RemoteMsgHandler
 DDE-процедура 236
RemoteQueryHandler() (оператор) 235
Remove Map Layer (оператор) 118
Rename File (оператор) 180
Resume (оператор) 83
RGB-цвета, значения 196
RollBack (оператор) 149
RTrim\$() (функция) 57
Run Application (оператор) 135
Run Menu Command (оператор) 99, 122

S

Save File (оператор) 180
SelChangedHandler (процедура) 70, 132
Select (оператор) 189, 190, 198, 217
Select Case (Do Case) 61
Selection
 запрос 155
 изменения 154
Server-операторы MapBasic 171
Set CoordSys (оператор) 157, 211
Set Map (оператор) 118, 199
Set Redistricter (оператор) 122
Set Shade (оператор) 118
Set Target (оператор) 205
Set Window (оператор) 117, 237
Shade (оператор) 118
SQL-запросы 149

Stop (оператор) 81

StringCompare() (функция) 57

StyleAttr() (функция) 196

StyleAttr() (функция) 193

StyleAttr() (функция) 196

Symbol (стиль) 192

T

TableInfo() (функция) 147, 170, 190

Targa-файлы 163

TempFileName\$() (функция) 180

TIFF-файлы 163

ToolHandler (процедура) 70, 126

TriggerControl() (функция) 112

Type...End Type (оператор) 45

U

UBound() (функция) 44

Update (оператор) 149, 200, 201, 204

User (Windows DLL) 223

V

Version 7.0 новости 299

Visual Basic 236

 пример 244

 пример программ 288

Visual C++

 примеры программ 288

W

WIN.INI

 задержка DDEобмена 232

 установки 225

WinChangedHandler (процедура) 70-71

WinClosedHandler (процедура) 70-71

WindowID() (функция) 117

WindowInfo() (функция) 117, 157

WinFocusChangedHandler (процедура) 70-71

Within (географический оператор) 216

Within (оператор) 58

Write # (оператор) 182

