

Владимир Овчинников

# Программирование для MAPINFO

на примерах

Программирование

ДЛЯ

*Marlino*

на примерах

Москва 2011

© В.А.Овчинников, 2011

<b>ВВЕДЕНИЕ.....</b>	<b>6</b>
<b>ГЛАВА 1. ОРГАНИЗАЦИЯ ПРОГРАММЫ.....</b>	<b>7</b>
ТЕМА 1. СТРУКТУРА ПРОСТОЙ ПРОГРАММЫ.....	7
ТЕМА 2. СТРУКТУРА МНОГОМОДУЛЬНОЙ ПРОГРАММЫ.....	8
ТЕМА 3. ОБЛАСТЬ ВИДИМОСТИ ПЕРЕМЕННЫХ.....	10
ТЕМА 4. БАЗОВЫЕ УСТАНОВКИ ПРОГРАММЫ.....	12
<b>ГЛАВА 2. ПОСТРОЕНИЕ ИНТЕРФЕЙСА.....</b>	<b>14</b>
ТЕМА 5. СОЗДАНИЕ МЕНЮ.....	14
ТЕМА 6. РЕДАКТИРОВАНИЕ СУЩЕСТВУЮЩЕГО МЕНЮ.....	15
ТЕМА 7. КОНТЕКСТНЫЕ МЕНЮ.....	16
ТЕМА 8. СТАНДАРТНЫЕ ДИАЛОГИ.....	17
ТЕМА 9. ПОЛЬЗОВАТЕЛЬСКИЕ ДИАЛОГИ.....	20
ТЕМА 10. СОЗДАНИЕ ОКНА КАРТЫ.....	26
ТЕМА 11. СОЗДАНИЕ ОКНА СПИСКА.....	27
ТЕМА 12. СОЗДАНИЕ ОКНА ГРАФИКА.....	28
ТЕМА 13. СОЗДАНИЕ ОКНА ОТЧЕТА.....	29
ТЕМА 14. ОБРАЩЕНИЕ К ОКНУ «СТАТИСТИКА».....	31
ТЕМА 15. ОБРАЩЕНИЕ К ОКНУ «СООБЩЕНИЯ».....	31
ТЕМА 16. ОБРАЩЕНИЕ К ОКНУ «ИНФОРМАЦИЯ».....	32
ТЕМА 17. ОБРАЩЕНИЕ К ОКНУ «ЛЕГЕНДА».....	33
ТЕМА 18. ОБРАЩЕНИЕ К ОКНУ «MAPBASIC».....	35
ТЕМА 19. ДОБАВИТЬ КНОПКУ К СТАНДАРТНОЙ ПАНЕЛИ.....	36
ТЕМА 20. СОЗДАТЬ НОВУЮ ИНСТРУМЕНТАЛЬНУЮ ПАНЕЛЬ.....	37
ТЕМА 21. ИСПОЛЬЗОВАНИЕ КНОПКИ ПЕРЕКЛЮЧАТЕЛЯ.....	37
ТЕМА 22. ИСПОЛЬЗОВАНИЕ КНОПКИ ИНСТРУМЕНТА.....	38
ТЕМА 23. ИСПОЛЬЗОВАНИЕ ПРОЦЕДУРЫ TOOLHANDLER.....	39
ТЕМА 24. ИСПОЛЬЗОВАНИЕ ФУНКЦИЙ WIN32 API.....	40
<b>ГЛАВА 3. РАБОТА С ТАБЛИЦАМИ.....</b>	<b>48</b>
ТЕМА 25. СОЗДАНИЕ ТАБЛИЦЫ ЗАДАННОЙ СТРУКТУРЫ.....	48
ТЕМА 26. ИЗМЕНЕНИЕ СТРУКТУРЫ ТАБЛИЦЫ.....	50
ТЕМА 27. ПОЛУЧЕНИЕ ИНФОРМАЦИИ О ТАБЛИЦЕ.....	51
ТЕМА 28. ПОСЛЕДОВАТЕЛЬНЫЙ ПЕРЕБОР ЗАПИСЕЙ В ТАБЛИЦЕ.....	53
ТЕМА 29. ОРГАНИЗАЦИЯ ВЫБОРКИ ЗАПИСЕЙ.....	53
ТЕМА 30. ПОИСК.....	54
ТЕМА 31. ГЕОКОДИРОВАНИЕ.....	56
ТЕМА 32. РЕДАКТИРОВАНИЕ ТАБЛИЦ.....	59
ТЕМА 33. РАБОТА С МЕТАДААННЫМИ.....	61
<b>ГЛАВА 4. РАБОТА С КАРТОЙ.....</b>	<b>64</b>
ТЕМА 34. СЛОИ.....	64
ТЕМА 35. ТЕМАТИЧЕСКИЕ СЛОИ.....	64
<i>Создание слоя диапазонов.....</i>	<i>65</i>
<i>Создание слоя отдельных значений.....</i>	<i>65</i>
<i>Создание слоя плотности точек.....</i>	<i>67</i>
<i>Создание слоя размерных символов.....</i>	<i>67</i>
<i>Создание слоя круговых диаграмм.....</i>	<i>68</i>
<i>Создание слоя столбчатых графиков.....</i>	<i>68</i>
<i>Создание тематической растровой поверхности.....</i>	<i>69</i>
ТЕМА 36. ПОДПИСИ.....	71
ТЕМА 37. СОЗДАНИЕ ОБЪЕКТОВ.....	72
ТЕМА 38. ИЗМЕНЕНИЕ ОБЪЕКТОВ.....	79
ТЕМА 39. ИЗВЛЕЧЕНИЕ ИНФОРМАЦИИ ОБ ОБЪЕКТАХ.....	85
ТЕМА 40. УПРАВЛЕНИЕ СТИЛЯМИ ОБЪЕКТОВ.....	88
ТЕМА 41. ПОЛУЧЕНИЕ ИНФОРМАЦИИ О КАРТЕ.....	92
ТЕМА 42. ИЗМЕНЕНИЕ ИЗОБРАЖЕНИЯ В ОКНЕ КАРТЫ.....	94
ТЕМА 43. УПРАВЛЕНИЕ ПОВЕДЕНИЕМ КАРТЫ В ОКНЕ.....	95

## ВВЕДЕНИЕ

ТЕМА 44. ИЗМЕНЕНИЕ ПОВЕДЕНИЯ ОТДЕЛЬНОГО СЛОЯ .....	96
ТЕМА 45. ИЗМЕНЕНИЕ ПРЕДСТАВЛЕНИЯ ОТДЕЛЬНОГО СЛОЯ .....	97
ТЕМА 46. ИЗМЕНЕНИЕ РЕЖИМА ПОДПИСЫВАНИЯ ОТДЕЛЬНОГО СЛОЯ .....	98
<b>ГЛАВА 5. РАБОТА С ФАЙЛАМИ.....</b>	<b>100</b>
ТЕМА 47. ТИПЫ ДОСТУПА К ФАЙЛАМ .....	100
ТЕМА 48. РАБОТА С ФАЙЛАМИ ПОСЛЕДОВАТЕЛЬНОГО ДОСТУПА.....	100
ТЕМА 49. РАБОТА С ФАЙЛАМИ ПРОИЗВОЛЬНОГО ДОСТУПА .....	102
ТЕМА 50. РАБОТА С ФАЙЛАМИ В РЕЖИМЕ ДВОИЧНОГО ДОСТУПА.....	105
ТЕМА 51. ОПЕРАЦИИ С ФАЙЛАМИ И КАТАЛОГАМИ .....	107
<b>ГЛАВА 6. ПРОГРАММЫ .....</b>	<b>110</b>
ТЕМА 52. ФОРМИРОВАНИЕ БАЛАНСА ТЕРРИТОРИЙ.....	110
<i>Постановка задачи</i> .....	110
<i>Проект</i> .....	110
Настройка параметров программы .....	110
Присвоение кодов группам участков .....	111
Построение баланса .....	111
Формирование новой таблицы с заданной структурой.....	111
<i>Код</i> .....	111
Блок описаний .....	111
Главная процедура .....	112
Процедура настройки параметров программы .....	113
Процедура присвоения кодов.....	117
Процедура построения баланса территорий .....	119
Процедура формирования новой таблицы.....	121
Процедура окончания работы с программой.....	124
ТЕМА 53. УПРАВЛЕНИЕ РАСТРОВЫМИ СЛОЯМИ.....	125
<i>Постановка задачи</i> .....	125
<i>Проект</i> .....	125
<i>Код</i> .....	125
Блок описаний .....	125
Главная процедура .....	125
Процедура чтения файла префиксов .....	126
Процедура включения дополнительных кнопок в базовую панель.....	127
Процедуры визуализации слоев.....	127
Прочие процедуры .....	128
ТЕМА 54. ДИНАМИЧЕСКАЯ ЗАГРУЗКА РАСТРОВЫХ ТАБЛИЦ .....	128
<i>Постановка задачи</i> .....	128
<i>Проект</i> .....	128
<i>Код</i> .....	128
Блок описаний .....	128
Главная процедура .....	129
Процедура выбора папки с растровыми таблицами.....	130
Процедура загрузки растровых таблиц .....	131
Процедура выгрузки растровых таблиц .....	132
Процедура считывания координат из растровых таблиц.....	133
Процедура извлечения координат из строки .....	133
Прочие процедуры .....	134
<b>ГЛАВА 7. ИНТЕГРИРОВАННАЯ КАРТОГРАФИЯ .....</b>	<b>136</b>
ТЕМА 55. ОБЪЕКТНАЯ МОДЕЛЬ MAPINFO OLE AUTOMATION TYPE LIBRARY .....	137
<i>Объект Application</i> .....	137
<i>Коллекция MVAApplications</i> .....	138
<i>Объект MapBasicApplication</i> .....	138
<i>Коллекция MBGlobals</i> .....	139
<i>Объект MBGlobal</i> .....	139
ТЕМА 56. ПРОСТОЕ ПРИЛОЖЕНИЕ С ИНТЕГРИРОВАННОЙ КАРТОЙ.....	139
<i>Модуль Common</i> .....	139
<i>Модуль tbDEF.vb</i> .....	139
<i>Класс cMI80</i> .....	140
<i>Класс comCallBk</i> .....	144
<i>Класс Main</i> .....	145
Вариант 1 (с использованием таймера) .....	146
Вариант 2 (с использованием делегата) .....	146

## ВВЕДЕНИЕ

Процедуры обработки событий .....	147
ТЕМА 57. ИСПОЛЬЗОВАНИЕ MAPBASIC ПРОГРАММ В ПРИЛОЖЕНИЯХ С ИНТЕГРИРОВАННОЙ КАРТОГРАФИЕЙ...	148
ТЕМА 58. ФОРМИРОВАНИЕ ОТЧЕТОВ, ВКЛЮЧАЮЩИХ КАРТЫ.....	152
<i>Модуль forWord</i> .....	152
<i>Класс cMI80</i> .....	155
<i>Класс Main</i> .....	156
ТЕМА 59. ИСПОЛЬЗОВАНИЕ ПРОЦЕДУР NET В MAPBASIC ПРОГРАММАХ .....	157
<i>Формирование библиотеки</i> .....	157
<i>Использование процедур библиотеки</i> .....	159
Блок описаний .....	159
Главная процедура .....	160
Прочие процедуры .....	160
ТЕМА 60. ПРИМЕРЫ КОМПОНОВКИ ПРИЛОЖЕНИЯ .....	162
<i>Пример 1</i> .....	162
<i>Пример 2</i> .....	163
<i>Пример 3</i> .....	165
<i>Пример 4</i> .....	165
<b>ПРИЛОЖЕНИЯ.....</b>	<b>167</b>
1. ФАЙЛ MAPINFOW.MNU .....	167
2. ТОЧНОСТЬ ПРЕДСТАВЛЕНИЯ КООРДИНАТ В MAPINFO.....	168
3. ФОРМИРОВАНИЕ ЗАПРОСОВ .....	169
4. СПИСОК ИДЕНТИФИКАТОРОВ ДЛЯ СТАНДАРТНЫХ МЕНЮ .....	171
5. ВСПОМОГАТЕЛЬНЫЕ ОКНА .....	172
6. ЕДИНИЦЫ ИЗМЕРЕНИЙ.....	173
<i>Единицы измерения площадей</i> .....	173
<i>Единицы измерения расстояний</i> .....	173
<i>"Бумажные" единицы измерения</i> .....	173
7. СТИЛИ ОФОРМЛЕНИЯ.....	174
<i>Типы линий</i> .....	174
<i>Тип заливки</i> .....	175
<i>Стиль шрифта</i> .....	176
<i>Стиль символа</i> .....	176
8. ПАРАМЕТРЫ ПРЕДЛОЖЕНИЯ CHARSET.....	177
9. ОПИСАНИЕ ФОРМАТА MIF/MID .....	178

### Введение

Эта книга для тех, кто уже знаком с MapInfo и имеет желание более активно управлять этой средой. Все изложенное ниже не является руководством или справочником по программированию в конкретных языках (MapBasic, VB.NET). Здесь показано расширение возможностей среды MapInfo за счет программирования. И упор сделан на рассмотрение примеров кода языка MapBasic. Следует отметить, что если вы хотите серьезно работать со средой MapInfo, то знание языка MapBasic обязательно.

Книга состоит из нескольких глав, каждая из которых связана с определенным кругом вопросов по использованию MapBasic.

В последней главе рассматривается так называемая интегрированная картография. То есть использование языков высокого уровня, в частности VB.NET, для управления средой MapInfo с целью построения законченных геоинформационных систем настроенных на конечного пользователя. Нужно отметить, что и в этом случае, при использовании VB.NET, C# или Delphi мы в конечном итоге обращаемся к MapInfo через операторы языка MapBasic.

В Приложении приводятся некоторые справочные сведения по языку MapBasic.

Все изложение материала ведется применительно к операционной системе Windows XP и MapInfo/MapBasic – версии не ниже 8.0 и Visual Studio 2010<sup>1</sup>.

Автор не ставил перед собой цель детально описать весь круг вопросов связанных с программированием для MapInfo. В данном издании рассматриваются, в основном базовые возможности, можно сказать то, без чего нельзя обойтись. Все эти решения будут справедливы и для более поздних версий MapInfo/MapBasic.

Включенные в книгу материалы рассчитаны в первую очередь на начинающих разработчиков ГИС проектов в среде MapInfo.

Адрес электронной почты для вопросов и комментариев [ova\\_2011@list.ru](mailto:ova_2011@list.ru)

---

<sup>1</sup> За исключением темы 59, где используются MapInfo/MapBasic – версии 10 и Visual Studio 2008.

## Глава 1. Организация программы

Программа это последовательность инструкций, в терминах языка программирования, задающая порядок выполнения определенных действий, направленных на решение поставленной задачи. В данной главе рассмотрены общие вопросы построения MapBasic программы.

### Тема 1. Структура простой программы

Простейшая программа на MapBasic состоит из одного модуля (файл с расширением *MB*). В модуль записываются процедуры *Sub* и функции *Function* реализующие решение задачи. Обязательное условие, одна из процедур должна иметь имя *Main*, именно с выполнения этой процедуры стартует программа. К модулю могут подключаться файлы заголовков. Примером таких файлов может служить файл **MapBasic.def**. После успешной компиляции программного модуля будет сформирован исполняемый файл программы с расширением *MBX*. На рис. 1 показана общая схема одномодульной программы.

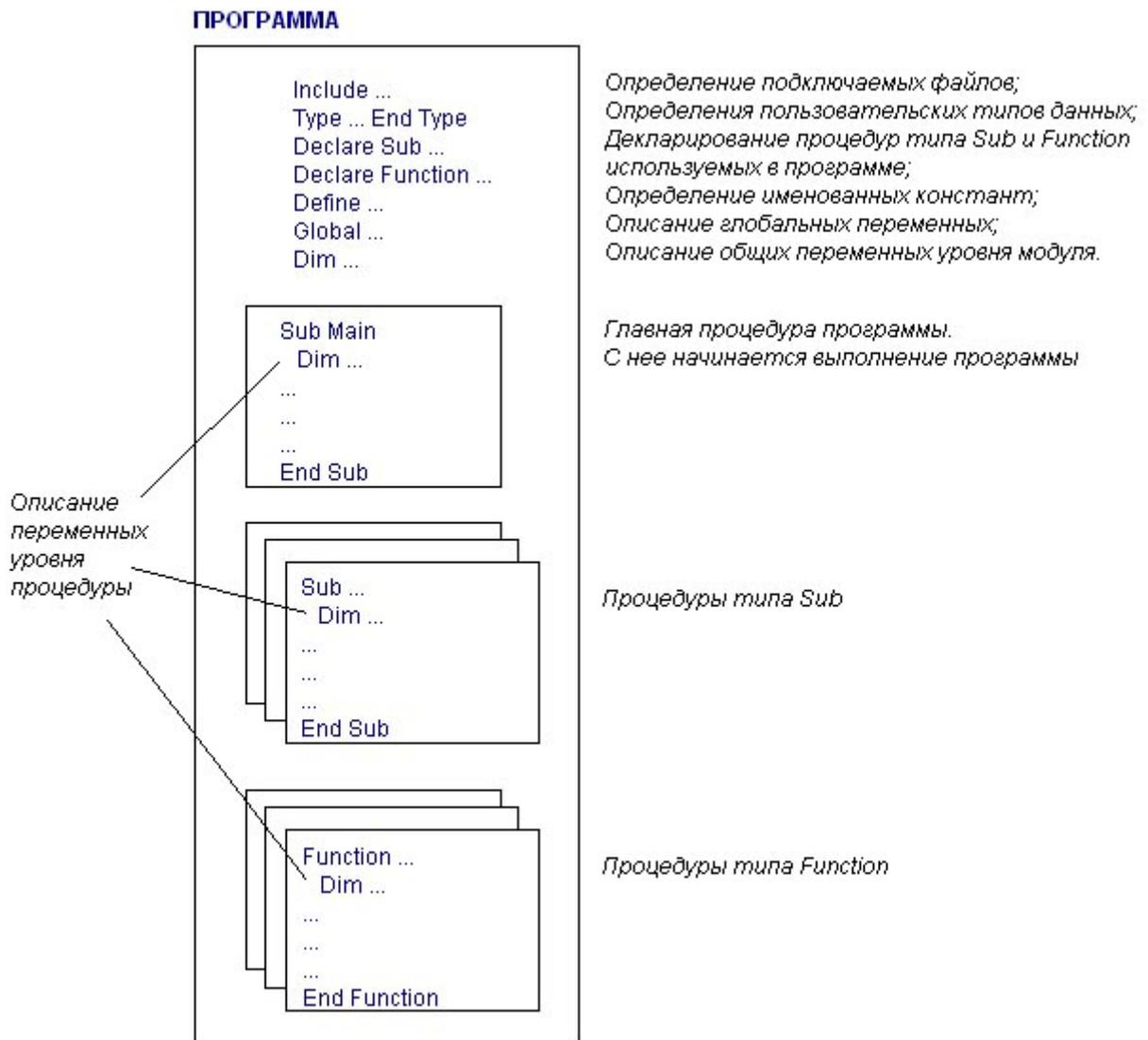


Рис. 1

Такая схема построения программы наиболее часто используется для реализации отдельных инструментов и небольших программ решающих некоторую локальную задачу.

Код: Простая программа

```

Declare Sub Main
Declare Sub tst1

Sub Main
call tst1
end sub

Sub tst1
print chr$(12)
print "Тест: Простая программа"
end sub
    
```

Приведенная программа имеет блок описаний, включающий декларирование процедур, и две процедуры. Далее остается только выполнить компиляцию программы и можно запускать ее на исполнение. Результатом работы программы будет текст, выведенный в окно **Сообщения**.

## Тема 2. Структура многомодульной программы

Сложные и объемные программы на MapBasic имеют многомодульную структуру<sup>2</sup>. Такая программа состоит из главного модуля, простых модулей и файла проекта. Физически модуль это текстовый файл с расширением *MB* а файл проекта имеет расширение *MBP*. Главный модуль программы отличается от простых модулей только наличием процедуры *main*. Структура главного модуля практически совпадает со схемой, представленной на рис. 1. Компиляция модулей выполняется независимо друг от друга. В результате компиляции будут получены объектные модули (файлы с расширением *MBO*) которые впоследствии, в соответствии с файлом проекта, объединяются в исполняемый файл *MBX* (Рис. 2).

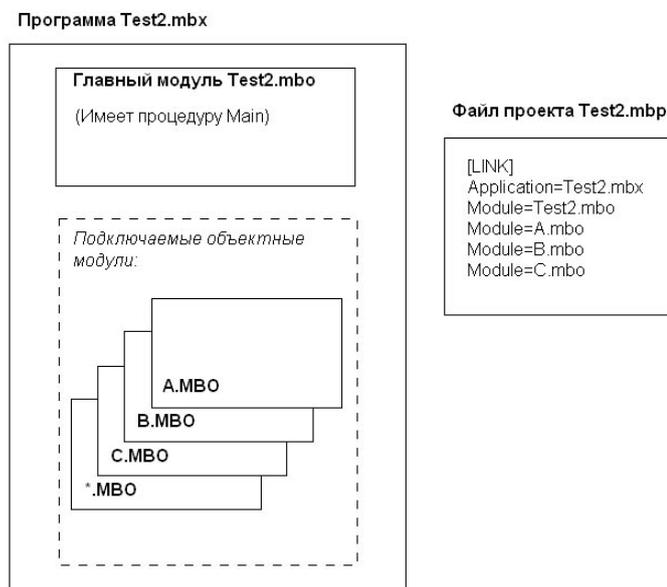


Рис. 2

Рассмотрим некоторый гипотетический пример. Наша программа состоит из четырех модулей.

<sup>2</sup> Это связано с ограничением среды MapBasic на размер модуля (64 Кбайт). Кроме того многомодульные программы легче поддерживать, а также это упрощает совместную разработку приложения.

Код: Многомодульная программа

Главный модуль **Test2.mb** имеет вид:

```
Include "mapbasic.def"  
Declare Sub Main  
Declare Sub A1  
Declare Sub A2  
Declare Sub B1  
Declare Sub B2  
Declare Sub C1  
Declare Sub C2
```

```
Sub Main  
print chr$(12)  
call A1  
call A2  
call B1  
call B2  
call C1  
call C2  
end sub
```

**Модуль А:**

```
Include "mapbasic.def"  
Declare Sub A1  
Declare Sub A2
```

```
Sub A1  
print "модуль А: процедура А1"  
end sub
```

```
Sub A2  
print "модуль А: процедура А2"  
end sub
```

**Модуль В:**

```
Include "mapbasic.def"  
Declare Sub B1  
Declare Sub B2
```

```
Sub B1  
print "модуль В: процедура В1"  
end sub
```

```
Sub B2  
print "модуль В: процедура В2"  
end sub
```

**Модуль С:**

```
Include "mapbasic.def"  
Declare Sub C1  
Declare Sub C2
```

```
Sub C1  
print "модуль С: процедура С1"  
end sub
```

```
Sub C2  
print "модуль С: процедура С2"  
end sub
```

После компиляции этих модулей сформируем файл проекта **Test2.mbp**<sup>3</sup>.

```
[Link]
Application=Test2.mbx
Module=Test2.mbo
Module=A.mbo
Module=B.mbo
Module=C.mbo
```

Следующий шаг это сборка проекта и после сборки получим исполняемый файл **Test2.mbx**. Результат работы этой программы показан на рис. 3.

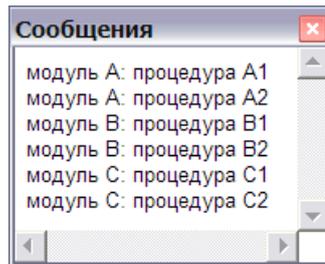


Рис. 3

Пожалуй, это все. Единственно, что можно добавить это возможность замены объявлений процедур в главном модуле на подключение соответствующих файлов *DEF*, которые и будут содержать все необходимые описания.

```
Include "mapbasic.def"
Include "A.def"
Include "B.def"
Include "C.def"
Declare Sub Main

Sub Main
print chr$(12)
call A1
call A2
call B1
call B2
call C1
call C2
end sub
```

Нужно также отметить, что хотя, в приведенном примере, все процедуры используются в главном модуле, ничто не мешает использовать процедуры из модуля **B** в модуле **A** и т.д. нужно только включить в модуль соответствующие объявления.

### **Тема 3. Область видимости переменных**

Обсуждая многомодульные программы нужно несколько слов сказать об области видимости переменных. По области видимости различают:

- ✓ Локальные переменные. Определяются в пределах конкретной процедуры с помощью оператора `Dim`, и никак не связаны с одноименными переменными вне этой процедуры.
- ✓ Локальные переменные уровня модуля. Определяются в модуле до кода процедур с помощью оператора `Dim`. Они доступны во всех процедурах этого

<sup>3</sup> Здесь в определении файла не указан путь, так как все файлы размещены в одной папке.

модуля кроме случая, когда в процедуре определена локальная переменная с тем же именем.

- ✓ Глобальные переменные уровня приложения. Определяются в модуле до кода процедур с помощью оператора `Global`. Они доступны во всех модулях программы. Глобальные переменные позволяют также обмениваться данными между работающими программами через механизмы DDE и OLE.

В качестве примера приведем рассмотренную выше многомодульную программу. Но добавим в нее определения переменных разного уровня, и все описания разместим в файлах *DEF*.

### Код: Использование переменных с разным уровнем видимости

Главный модуль **Test2.mb** теперь имеет вид:

```
Include "A.def"  
Include "B.def"  
Include "C.def"  
Declare Sub Main  
Define CLS Print Chr$(12)
```

```
Sub Main  
CLS  
call A1  
call A2  
call B1  
call B2  
call C1  
call C2  
Print AA  
Print BB  
end sub
```

#### Модуль А:

```
Include "A.def"  
'локальная переменная уровня модуля  
dim sA1 as string  
Sub A1  
sA1="модуль А: процедура A1"  
print sA1  
end sub
```

```
Sub A2  
'локальная переменная уровня процедуры  
dim sA2 as string  
sA2="модуль А: процедура A2"  
print sA2  
AA="глобальная переменная AA"  
end sub
```

#### Модуль В:

```
Include "B.def"  
Sub B1  
print "модуль В: процедура B1"  
end sub
```

```
Sub B2  
print "модуль В: процедура B2"  
BB="глобальная переменная BB"  
end sub
```

#### Модуль С:

```
Include "C.def"
Sub C1
print "модуль C: процедура C1"
end sub

Sub C2
print "модуль C: процедура C2"
end sub
```

### Файл A.def

```
Declare Sub A1
Declare Sub A2
Global AA as string
```

### Файл B.def

```
Declare Sub B1
Declare Sub B2
Global BB as string
```

### Файл C.def

```
Declare Sub C1
Declare Sub C2
```

Файл проекта остался без изменений<sup>4</sup>.

После компиляции модулей и сборки проекта получим исполняемый файл **Test2.mbx**. Результат работы этой программы показан на рис. 4.

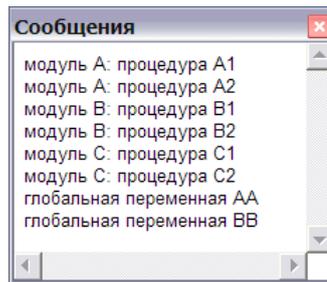


Рис. 4

## Тема 4. Базовые установки программы

Координатная система в программе задается с помощью оператора `Set CoordSys`. Все операции с координатами будут выполняться в соответствии с установленной системой координат. Если система координат не назначалась, то будет использоваться система координат по умолчанию (Долгота/Широта).

Для работы с окнами Отчета необходимо установить соответствующую систему координат с помощью оператора вида `Set CoordSys Layout Units "mm"`.

Разумно также сразу определиться в программе с единицами измерения площадей и расстояний.

<code>Set Area Units "sq m"</code>	Устанавливает единицы измерения площади. По умолчанию кв. мили (sq mi).
<code>Set Distance Units "m"</code>	Устанавливает единицы измерения расстояний. По умолчанию мили (mi).
<code>Set Paper Units "mm"</code>	Устанавливает так называемые «бумажные» единицы измерения (размеры и положение окон на экране и т.л.). По умолчанию дюйм (in).

<sup>4</sup> Предполагается, что все файлы размещены в одной папке.

Иногда, хотя и редко, возникает необходимость в изменении, используемой в программе координатной системы, на систему выбранную пользователем. В этом случае поможет следующий код.

```
dim newCS as string
newCS=ChooseProjection$("", True)
newCS= "Set "+newCS
Run Command newCS
```

Функция `ChooseProjection$` открывает окно выбора проекции и возвращает строку с выбранной проекцией. Первый аргумент, имя проекции, определяет, какая проекция будет показана при открытии окна. Если этот аргумент пустая строка или имя проекции указано неверно, то при открытии активной будет текущая проекция, установленная в программе. Второй аргумент это логическое значение определяющее отображать ли диалог задания границ, если выбранная проекция план-схема.

Получить информацию о действующих установках можно с помощью следующих функций.

<code>SessionInfo</code> <code>(SESSION_INFO_COORDSYS_CLAUSE)</code>	Возвращает строку с установленной в программе системой координат.
<code>SessionInfo</code> <code>(SESSION_INFO_DISTANCE_UNITS)</code>	Возвращает строку с установленной в программе единицей измерения расстояний.
<code>SessionInfo</code> <code>(SESSION_INFO_AREA_UNITS)</code>	Возвращает строку с установленной в программе единицей измерения площади.
<code>SessionInfo</code> <code>(SESSION_INFO_PAPER_UNITS)</code>	Возвращает строку с установленной в программе «бумажной» единицей измерения.

Если какие-то параметры в программе прямо не устанавливались, то данные функции будут возвращать значения актуальные в сеансе работы на текущий момент.

## Глава 2. Построение интерфейса

Чтобы программой могли пользоваться все, необходимо создать удобную и понятную среду общения пользователя с программой. Такой дружелюбный интерфейс, требующий минимального предварительного изучения, может стать залогом успеха программы в целом. В MapInfo взаимодействие с пользователем реализуется через меню, инструментальные панели, диалоговые окна.

Проектирование графического интерфейса пользователя один из важных этапов в создании программы. Может быть, для небольшой программы, которая управляется одной кнопкой в инструментальной панели это и не столь очевидно, но при проектировании геоинформационной системы этот вопрос будет одним из главных. Фрагмент программы, обеспечивающий графический интерфейс пользователя, часто занимает значительную часть ее кода и эта часть очень важна. Далее рассматриваются отдельные приемы построения интерфейса пользователя в MapBasic-программах.

### Тема 5. Создание меню

Для создания нового меню или переопределения уже существующего используется оператор `Create Menu`. Рассмотрим два небольших примера создания меню.

В первом примере создается новое меню, которое будет размещено в конце строки главного меню MapInfo. После выполнения процедуры `theEnd` программа закончит работу, а построенное меню будет выгружено.

#### Код: Создание нового меню

```
Include "mapbasic.def"
Declare Sub Main
Declare Sub TestPrint
Declare Sub theEnd

Sub Main
Create Menu "Тест меню" As
    "Печать" Calling TestPrint,
    "Выход" Calling theEnd
Alter Menu Bar Add "Тест меню"
end sub

Sub TestPrint
print "Тест меню"
end sub

Sub theEnd
End Program
end sub
```

Несколько изменим главную процедуру программы.

```
Sub Main
Create Menu "Тест меню" As
    "Печать" Calling TestPrint,
    "Выход" Calling theEnd
Alter Menu Bar Remove ID 6, ID 7
Alter Menu Bar Add "Тест меню", ID 6, ID 7
end sub
```

В этом случае наше меню разместится в строке меню MapInfo перед меню **Окно**.

Здесь вместо имен меню используются идентификаторы. Список идентификаторов для стандартных меню можно найти в приложении.

Разберем код второго примера.

### Код: Создание нового меню вместо существующего

```
Include "mapbasic.def"
Declare Sub Main
Declare Sub TestPrint
Declare Sub theEnd

Sub Main
Create Menu "Справка" As
    "Печать" Calling TestPrint,
    "Выход" Calling theEnd
end sub

Sub TestPrint
print "Тест меню"
end sub

Sub theEnd
End Program
end sub
```

В данном случае стандартное меню *Справка* заменит новое меню, состоящее из двух пунктов *Печать* и *Выход*. После окончания работы программы (процедура `theEnd`) будет восстановлено стандартное меню *Справка*.

Можно программно управлять состоянием элемента меню. Для этого используются специальные управляющие коды. Информацию по этим кодам можно найти в справочнике `MapBasic`. Приведем пример использования управляющих кодов.

```
Create Menu "Справка" As
    "Печать" + Chr$(9) + "Ctrl+д/W^L" Calling TestPrint,
    "(-",
    "Выход" Calling theEnd
```

Здесь в пункт меню *Печать* будет добавлен комментарий `Ctrl+д` и появится дополнительная возможность вызова обработчика `TestPrint` через нажатие на клавиатуре клавиш `Ctrl + д` (Д, I, L). Следующий пункт меню не имеет обработчика события и его роль чисто декоративная – он рисует линию разделяющую пункты меню.

### Тема 6. Редактирование существующего меню

Необходимость в редактировании меню возникает, если нужно

- ✓ добавить или удалить пользовательское меню как пункт существующего меню;
- ✓ добавить или удалить элементы строки меню `MapInfo`;
- ✓ изменить состояние элемента меню.

Рассмотрим каждый из этих случаев отдельно.

Допустим, мы хотим установить наше меню «Тест» как подменю меню «Программы».

### Код: Пользовательское меню как пункт существующего меню

```
Include "mapbasic.def"
Declare Sub Main
Declare Sub TestPrint
Declare Sub theEnd

Sub Main
Create Menu "Тест" As
    "Печать"
```

```

        ID 201
        HelpMsg "Тестируем меню"
        Calling TestPrint,
    "Выход"
        ID 202
        HelpMsg "Закрыть программу"
        Calling theEnd
Alter Menu "Программы" Add
    "(-",
    "Тест" As "Тест"
end sub

Sub TestPrint
print "Тест меню"
end sub

Sub theEnd
End Program
end sub

```

В код включены строки комментариев для обработчиков событий – при наведении курсора на пункт меню, в строке состояния (*Status bar*) будет выведен соответствующий комментарий.

Результатом работы программы будет появление меню **Тест** в конце списка пунктов меню **Программы**. Если теперь в окне **MapBasic** выполнить оператор `Alter Menu "Программы" Remove "Тест"`, то меню **Тест** будет удалено.

Добавить или удалить заголовок меню **Тест** в строке меню окна **MapInfo** можно с помощью следующих операторов:

```
Alter Menu Bar Add "Тест"
Alter Menu Bar Remove "Тест"
```

Рассмотрим вопрос о состоянии элемента меню на примере приведенного выше кода. Добавим несколько операторов в конец процедуры `Main`.

```
Alter Menu Item ID 201 Text "Вывод сообщения"
Alter Menu Item ID 202 Disable
Alter Menu Item ID 201 Calling theEnd
```

В результате, пункт меню **Печать** теперь будет именоваться **Вывод сообщения**, пункт меню **Выход** станет недоступным и после нажатия на **Вывод сообщения** программа будет закрыта.

### Тема 7. Контекстные меню

Контекстное меню является удобным средством интерфейса пользователя, заметно ускоряющим работу с приложением. В **MapInfo** различают контекстные меню окна карты, окна списка и т.д. Идентификаторы стандартных меню с *ID 16* по *ID 22* относятся к контекстным меню.

Такие меню управляются с помощью тех же программных средств, что и обычные меню (`Create Menu`, `Alter Menu`, `Alter Menu Item`). Отключить контекстное меню можно с помощью оператора типа `Create Menu "MapperShortcut" As "(-"`. В данном случае будет отключено контекстное меню окна карты. Восстановить доступность стандартного контекстного меню окна карты можно с помощью оператора `Create Menu "MapperShortcut" As Default`.

**Код: Построение контекстного меню окна карты**

```
Sub CreateContextualMenuMap
Create Menu "MapperShortcut" ID 17 As "(-"
```

```

Create Menu "MapperShortcut" ID 17 As
  "Управление слоями..." Calling 801,
  "(-",
  "Найти выборку" Calling 306,
  "(-",
  "Вы&резать\tCtrl+Ч/W^X/Mx/XCtrl+x" Calling 202,
  "&Копировать\tCtrl+C/W^C/Mc/XCtrl+c" Calling 203,
  "&Вставить\tCtrl+M/W^V/Mv/XCtrl+v" Calling 204,
  "У&далить\tDel" Calling 205,
  "(-",
  "Показать по-другому..." Calling 805,
  "Показать как было" Calling 806,
  "Показать слой полностью..." Calling 807,
  "(-",
  "Отменить выбор" Calling 304,
  "Обновить окно" Calling 610,
  "(-",
  "Удалить косметику" Calling 810,
  "!Выключить Автопрокрутку^Включить Автопрокрутку" Calling 815,
  "(-",
  "Объекты" as "Объекты",
  "(-",
  "Геоинформация..." Calling 207
end sub

```

Здесь вызываемые методы обозначены через числовые идентификаторы, так как это сделано в файле MapInfo.mnu. Для лучшей читабельности кода вместо чисел можно использовать именованные константы из файла menu.def, но предварительно его нужно подключить к программе с помощью оператора Include "menu.def". Тогда, например строка "Управление слоями..." Calling 801 будет выглядеть как "Управление слоями..." Calling M\_MAP\_LAYER\_CONTROL.

### Код: Построение контекстного меню окна списка

```

Sub CreateContextualMenuBrowse
Create Menu "BrowserShortcut" ID 18 As "(-"
Create Menu "BrowserShortcut" ID 18 As
  "Вы&резать\tCtrl+Ч/W^X/Mx/XCtrl+x" Calling 202,
  "&Копировать\tCtrl+C/W^C/Mc/XCtrl+c" Calling 203,
  "&Вставить\tCtrl+M/W^V/Mv/XCtrl+v" Calling 204,
  "У&далить\tDel" Calling 205,
  "(-",
  "Найти выборку" Calling 306,
  "Отменить выбор" Calling 304,
  "(-",
  "&Новая запись\tCtrl+П/W^G/Mn/XCtrl+n" Calling 702,
  "&Показывать поля..." Calling 704
end sub

```

Все та же схема построения, сначала меню полностью удаляется, а затем строится заново. Если нужно добавить к стандартному контекстному меню один или два новых пункта, то достаточно использовать оператор следующего вида Alter Menu ID 18 Add "Число записей" Calling GetRowCount.

### **Тема 8. Стандартные диалоги**

Диалоговые окна присутствуют почти в каждой программе. Их необходимость обусловлена хотя бы тем, что пользователю нужно:

- ✓ Настроить процесс решения задачи
- ✓ Получать сообщения и комментарии в ходе решения

- ✓ Управлять процессом решения задачи
- ✓ Обеспечить ввод/вывод данных

При программировании в MapBasic различают стандартные и пользовательские окна диалогов.

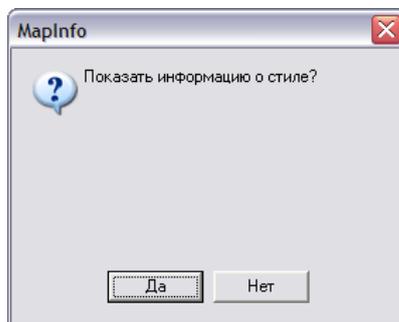
К стандартным диалогам относят окна диалогов сформированные следующими операторами:

Ask	Показывает диалоговое окно с сообщением или вопросом и предлагающее подтвердить или отменить предложение
Note	Показывает информационное сообщение
FileOpenDlg	Вызывает диалоговое окно «Открыть файл ...»
FileSaveAsDlg	Вызывает диалоговое окно «Сохранить файл как ...»
ProgressBar	Создает диалоговое окно-индикатор выполнения процесса
Set ProgressBars	Показывает или скрывает диалог-индикатор выполнения процесса
Alter	Делает недоступными, прячет или присваивает значение
MapInfoDialog	элементу стандартного диалогового окна в MapInfo

Диалог Ask имеет две кнопки и возвращает логическое значение в зависимости от нажатой кнопки<sup>5</sup> (Рис. 5).

**Код: Диалоговое окно с вопросом**

```
...
if Ask("Показать информацию о стиле?", "Да", "Нет")="Т" then
    print ss
else
    ss=""
end if
```

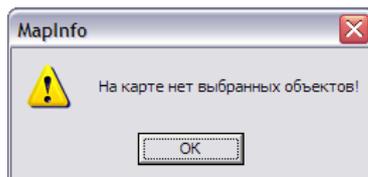


**Рис. 5**

Диалог Note скромнее до аскетизма и, тем не менее, используется весьма часто (Рис. 6).

**Код: Окно с информационным сообщением**

```
...
Note "На карте нет выбранных объектов!"
```



**Рис. 6**

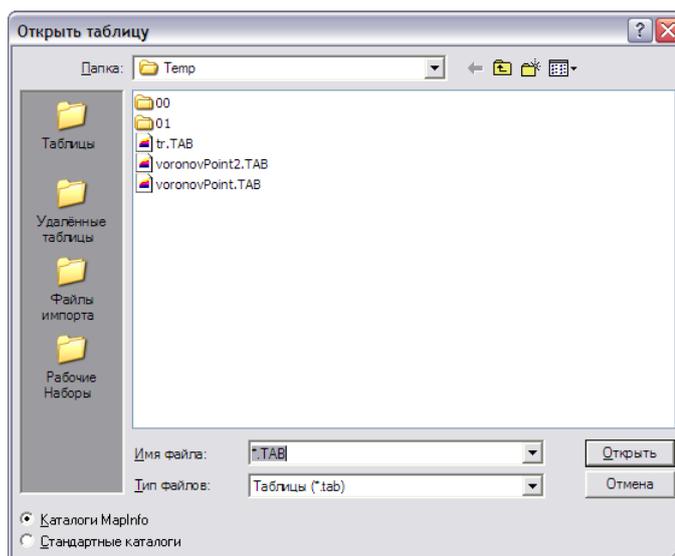
<sup>5</sup> Если нажать кнопку Enter на клавиатуре будет возвращено True, если Esc – False.

Функция `FileOpenDlg()` показывает диалоговое окно **MapInfo Открыть таблицу** (Рис. 7).

**Код: Диалоговое окно «Открыть файл ...»**

```

...
Dim fileName As String
fileName = FileOpenDlg("C:\Temp", "", "TAB", "Открыть таблицу")
if len(fileName)>0 then
    print fileName
else
    note "Таблица не выбрана!"
end if
    
```



**Рис. 7**

Функция `FileOpenDlg()` не открывает выбранный файл, а только возвращает полное имя файла. Если в диалоге нажата кнопка **Отмена** – возвращается пустая строка. Для того чтобы при старте диалога открывался текущий рабочий каталог поставьте вместо "C:\Temp" пустую строку ("").

Функция `FileSaveAsDlg()` работает подобно функции **MapInfo Файл /Создать копию**. Также как в предыдущем операторе возвращается полное имя файла.

Оператор `Alter MapInfoDialog` позволяет присваивать значения элементам стандартных диалогов **MapInfo**, делать недоступными или скрытыми их кнопки, флажки и т. п. Так как идентификаторы, используемые этим оператором, зависят от версии **MapInfo**, то я бы не рекомендовал его использовать.

В заключение приведем пример использования индикатора выполнения процесса `ProgressBar` для обработки объектов в таблице. Сформулируем задачу: есть таблица и нужно пройти по всем строкам этой таблицы и для каждой строки выполнить некоторую работу. Эту работу определим как запись в поле *ID* некоторого значения.

**Код: Диалоговое окно — индикатор выполнения процесса**

```

'Блок описаний
Include "MAPBASIC.DEF"
Declare sub Main
Declare Sub ProgressBarDemo
Declare Sub WorkingLine
dim ProgressStart,ProgressEnd as integer
dim dt,ProgressD as integer
dim tbName as string
'Главная процедура
sub Main
    
```

'Имя таблицы Points. Обязательно должно быть поле ID целого типа.

```
tbName="Points"
call ProgressBarDemo
end sub
```

```
Sub ProgressBarDemo
ProgressStart = 1
ProgressEnd = TableInfo(tbName, TAB_INFO_NROWS)
ProgressD=1
```

'Параметр гладкости процесса (регулируется по необходимости).

```
dt=ProgressEnd/100+1
ProgressBar "Обработка информации..."
    Calling WorkingLine
    Range ProgressEnd
If CommandInfo(CMD_INFO_STATUS) Then
    Note "Обработка информации закончена!"
Else
'Если процесс прерван принудительно.
    Note "Обработка информации прервана на строке "
        & Str$(ProgressStart)
End If
end sub
```

```
Sub WorkingLine
Do While ProgressStart<=ProgressEnd And
    ProgressStart<ProgressEnd/dt*ProgressD
    Fetch Rec ProgressStart From tbName
```

'Здесь выполняется полезная работа. В данном случае просто изменяем значение поля ID.

```
Update tbName Set
    ID=1000+ProgressStart
    Where RowID=ProgressStart
    ProgressStart=ProgressStart +1
Loop
ProgressD=ProgressD+1
If ProgressStart>ProgressEnd Then
    ProgressBar=-1
Else
    ProgressBar=ProgressStart
End If
end sub
```

## **Тема 9. Пользовательские диалоги**

Для формирования пользовательских диалоговых окон используют следующие операторы:

Dialog	Создает новое диалоговое окно.
Alter Control	Изменяет состояние элемента диалога.
TriggerControl()	Возвращает идентификатор элемента диалога, к которому пользователь обращался в последний раз.
ReadControlValue()	Возвращает текущее значение одного из элементов активного диалога.
Dialog Preserve	Приостанавливает закрытие пользовательского диалога <sup>6</sup> .
Dialog Remove	Обеспечивает возможность закрытия пользовательского диалога до нажатия кнопки Ok или Cancel..
CommandInfo()	Используется для определения следующих событий: нажатие кнопки OK, завершение процесса ProgressBar, двойное нажатие мыши в списках диалога.

Рассмотрим примеры.

---

<sup>6</sup> Обычно используется для организации подтверждения закрытия диалога.

Нужен диалог для программы вставки точек в карту по координатам из текстового файла. Сразу определим, какими параметрами мы хотим управлять из диалога:

- ✓ Определение полного имени файла с координатами, при этом считаем, что таблица будет размещена в той же папке и с тем же именем.
- ✓ Определение стиля точек.
- ✓ Определение формата записи (строки) в текстовом файле<sup>7</sup>.
- ✓ Определение разделителя полей в записи.
- ✓ Определение наличия заголовков в первой строке.

Для формирования диалога будем использовать программу **MapBasic Designer** (*mbd.exe*)<sup>8</sup>. На рис. 8 показан окончательный вид проекта диалога.

MapBasic Designer прост в использовании и это его основное преимущество. Программа заявлена как бета-версия. При работе возможны исключительные ситуации, поэтому рекомендуется чаще сохранять промежуточные результаты. По окончании работы получим файл MapBasic программы (\*.mb), который представляет собой некоторую заготовку диалога, отражающую геометрию размещения элементов. Саму программу можно найти в Интернете.



Рис. 8

Приводим доработанный вариант кода программы с диалогом.

**Код: Диалог для программы вставки точек в карту по координатам из текстового файла**

```
'Блок описаний
Include "MAPBASIC.DEF"
Declare sub Main
Declare sub diaInsertPointsFromFile
Declare sub ReadFileName
Declare sub forYes
Declare sub forCancel
dim txtFileName as string
dim StylePoint as Symbol
dim iFormatLine,iDelimiter as smallint
dim isHeadline as logical
'Главная процедура
```

<sup>7</sup> Одна запись – одна точка.

<sup>8</sup> Для облегчения формирования диалогов можно использовать и другие программы (DiaBuilder2000, Userform2Dialog, MBBuilder и др.). Их также можно найти в сети Интернет.

```

sub Main
'Стиль символа устанавливается равным текущему стилю символа
StylePoint= CurrentSymbol()
'Имя текстового файла
txtFileName=""
'Формат строки {Имя, X, Y, H}
iFormatLine=1
'Разделитель {табулятор}
iDelimiter=1
'Нет заголовков в первой строке
isHeadline="F"
'Открыть диалог
call diaInsertPointsFromFile
end sub
'Окно диалога
Sub diaInsertPointsFromFile
Dialog
    Title "Вставка точек из файла"
    Width 149
    Height 195
    Control OkButton
        Width 37
        Height 12
        Position 104, 144
    Control CancelButton
        Width 37
        Height 12
        Position 104, 160
    Control Button
        Title "Файл..."
        Width 37
        Height 12
        Position 104, 104
        ID 5001
        Calling ReadFileName
    Control SymbolPicker
        Width 20
        Height 20
        Position 120, 16
        ID 5002
        Value StylePoint
        Into StylePoint
    Control CheckBox
        Title "заголовки в первой строке"
        Width 104
        Height 12
        Position 10, 175
        ID 5003
        Value isHeadline
        Into isHeadline
    Control GroupBox
        Title "Формат записи"
        Position 8, 8
        Width 87
        Height 75
    Control RadioGroup
        Title "Имя X Y H;Имя X Y;X Y H;X Y"
        Width 49
        Height 10
        Position 16, 23
        ID 5004
        Value iFormatLine

```

```

        Into iFormatLine
Control GroupBox
    Title "Разделитель"
    Position 8, 96
    Width 87
    Height 74
Control RadioGroup
    Title "табулятор;пробел;запятая;точка с запятой"
    Width 68
    Height 10
    Position 17, 110
    ID 5005
    Value iDelimiter
    Into iDelimiter
If CommandInfo(CMD_INFO_DLG_OK) Then
'Если выбрана кнопка ОК
    call forYes
Else
    Call forCancel
End If
End sub
'Определение имени файла (например, с использованием FileOpenDlg)
sub ReadFileName
    note "Определяем имя файла (txtFileName)"
end sub
'Чтение координат из файла и последующая обработка информации
sub forYes
    note "Читаем файл координат." & chr$(10) &
        "Формируем новый слой и" &
        chr$(10) & "вставляем в него точки с" &
        chr$(10) & "выбранным стилем оформления."
end sub
'Отказ от чтения файла
sub forCancel
    note "Нажата кнопка Cancel"
end sub

```

Рассмотрим задачу построения диалога с использованием элементов `ListBox` и `PopupMenu`. Приведем код демонстрационной программы.

**Код: Диалог с использованием элементов `ListBox` и `PopupMenu`**

```

'Блок описаний
Include "MAPBASIC.DEF"
Declare sub Main
Declare Sub ListDemo
Declare Sub lbDb1Click
Declare Sub pmClick
Declare Sub CreateListLayerNames(byval id as smallint)
Declare Sub CreateListFieldsNames(ByVal Table as string)
Declare function ItsMap as integer
'Идентификатор окна карты
dim idWinMap as integer
dim i_lrType,i_lrName,i fldName as smallint
'Список слоев.
dim ListLayers() as string
'Список полей в выбранном слое.
dim ColList as string
'Главная процедура
sub Main

```

'Проверка: окно карты д.б. активно.

```
idWinMap=ItsMap()  
if idWinMap=0 then  
    exit sub  
end if  
call ListDemo  
end sub
```

'Диалог

```
sub ListDemo  
Dialog  
    Title "ПРИМЕР 2"  
    Width 144  
    Height 213
```

'Раскрывающийся список типов слоев.

```
Control PopupMenu  
    Width 126  
    Height 10  
    Position 9, 20  
    ID 1002  
    Title "Нормальный; Косметический; Растровый; Тематический; Поверхность"  
    Value 1  
    Into i_lrType
```

'Обработчик события (клик мышью на элементе списка)

```
Calling pmClick
```

'Список слоев выбранного типа

```
Control ListBox  
    Width 126  
    Height 54  
    Position 8, 52  
    ID 1001  
    Title "Не определен!"  
    Value 1  
    Into i_lrName
```

'Обработчик события (двойной клик мышью на элементе списка)

```
Calling lbDbClick
```

'Список полей для выбранного слоя

```
Control ListBox  
    Width 123  
    Height 70  
    Position 8, 128  
    ID 1003  
    Title "Не определен!"  
    Value 1  
    Into i_fldName
```

'Подпись к раскрывающемуся списку

```
Control StaticText  
    Title "Тип слоя"  
    Width 50  
    Height 8  
    Position 10, 7
```

'Подпись к списку

```
Control StaticText  
    Title "Список слоев"  
    Width 50  
    Height 8  
    Position 10, 40
```

'Подпись к списку

```
Control StaticText  
    Title "Список полей"  
    Width 50  
    Height 8
```

```

        Position 10, 117
    end sub
    'Обработчик события (клик мышью на элементе списка типов слоев)
    Sub pmClick
    dim ind as smallint
    'Определяем идентификатор типа слоя
    ind = ReadControlValue(1002)-1
    'Определяем список слоев заданного типа
    Call CreateListLayerNames(ind)
    if UBound(ListLayers)=0 Then
        Alter Control 1001 Title "Нет"
    Else
    'Заполняем список слоев
        Alter Control 1001 Title From Variable ListLayers
    end if
    end sub
    'Обработчик события (двойной клик мышью на элементе списка)
    Sub lbDbClick
    dim ind as smallint
    'Определяем тип слоя
    ind = ReadControlValue(1002)-1
    'Проверяем условие: двойной клик мышью и тип слоя нормальный
    If CommandInfo(CMD_INFO_DLG_DBL)=TRUE And ind=LAYER_INFO_TYPE_NORMAL Then
    'Определяем индекс для слоя
        ind = ReadControlValue(1001)
        if ubound(ListLayers)=0 then
            exit sub
        end if
    'Формируем список полей для выбранного слоя и записываем его в
    'соответствующий ListBox диалога
        call CreateListFieldsNames(ListLayers(ind))
        Alter Control 1003 Title ColList
    else
        Alter Control 1003 Title "Нет"
    End If
    end sub
    'Формирование списка слоев выбранного типа в массиве ListLayers.
    Sub CreateListLayerNames(byval id as smallint)
    Dim s As String
    Dim j, n, m As Integer
    Dim t As Integer
    ReDim ListLayers(0)
    t=0
    n =MapperInfo(idWinMap,MAPPER_INFO_LAYERS)
    For j = 0 To n
    s = LayerInfo(idWinMap , j,LAYER_INFO_NAME)
    m =LayerInfo(idWinMap ,j,LAYER_INFO_TYPE)
    If m = id Then
    t =t+1
    ReDim ListLayers(t)
    ListLayers(t) = s
    End If
    Next
    End Sub
    'Формирование списка полей для таблицы TableName. Список формируется
    'в виде строки с разделителем (;).
    Sub CreateListFieldsNames(ByVal TableName as string)
    Dim iColNum as SmallInt
    ColList = ColumnInfo(TableName,"col1",COL_INFO_NAME)
    For iColNum = 2 to NumCols(TableName)

```

```

ColList = ColList & ";" &
        ColumnInfo (TableName, "col" &
        Str$(iColNum), COL_INFO_NAME)
Next
end sub
'Выполняется проверка: активное окно это окно карты.
'Если ДА то возвращается идентификатор окна, иначе – 0.
function ItsMap() as integer
dim map_id as integer
ItsMap=0
map_id = FrontWindow()
If WindowInfo(map_id , WIN_INFO_TYPE) <> WIN_MAPPER Then
    Note "Окно карты должно быть активным!"
    Exit function
End If
ItsMap=map_id
end function

```

Как, наверное, уже видно из кода, наша программа строит диалог, состоящий из трех списков. Верхний, реализованный в виде `PopupMenu`, представляет собой фиксированный список типов слоев. Далее идет `ListBox` со списком слоев. Список заполняется после того как выбран тип слоя. Последним идет `ListBox` со списком полей выбранного слоя. Этот список заполняется после двойного клика мышью на имени слоя. Общий вид диалога показан на рис. 9.

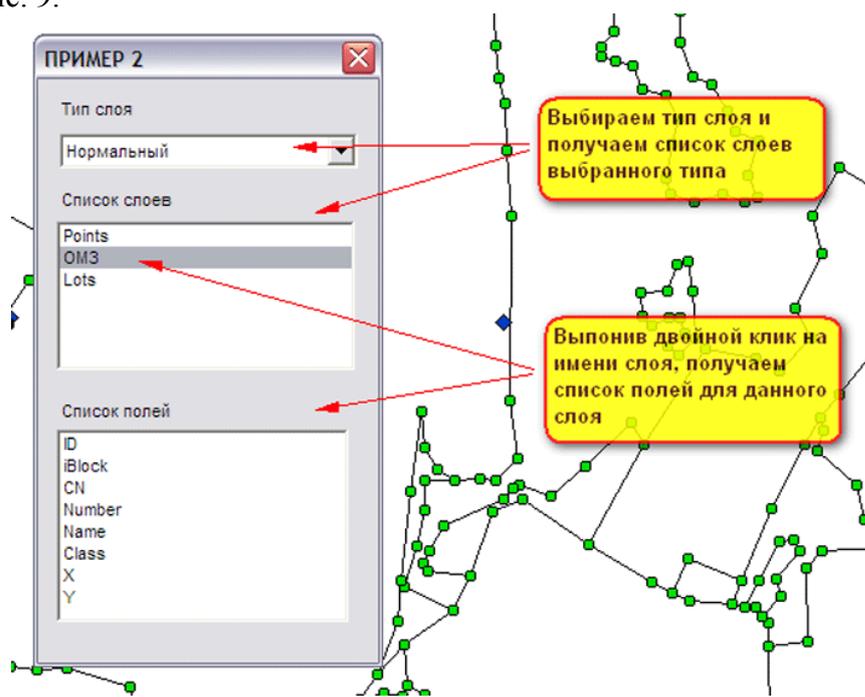


Рис. 9

### Тема 10. Создание окна карты

Покажем, как можно создать окно карты и выполнить с ним некоторые манипуляции. Считаем, что в системе загружены таблицы **Lots**, **Points**, **OMZ** и в программе определена общая переменная `idWinMap`.

**Код: Создание окна карты**

```

sub WinDemoMap
'Создаем новое окно карты из двух таблиц. После создания окно становится активным.
Map From Lots,Points

```

```

        Position (20,20) Units "mm"
        Width 100 Units "mm"
        Height 70 Units "mm"
'Определяем идентификатор окна
idWinMap=FrontWindow()
'В окно карты добавляем таблицу OMZ
Add Map Window idWinMap Layer OMZ
'Уточняем положение окна и добавляем новый заголовок окна
Set Window idWinMap
    Position(30,30) Units "mm"
    Title "Новая карта"
'Изменяем отображение карты в окне: устанавливаем масштаб 1:100000
Set Map Window idWinMap
    Scale 10 Units "mm" For 1000 Units "m"
'Открываем копию окна карты idWinMap
Run Command WindowInfo(idWinMap, WIN_INFO_CLONEWINDOW)
end sub

```

### **Тема 11. Создание окна списка**

Информация из таблиц может быть представлена в виде так называемого списка, отображаемого в окне списка. Список представляет собой набор записей (строк). Каждая запись состоит из нескольких полей (колонок). Для поддержки связи с графикой имеется скрытое поле *Obj*. Скрытое псевдополе *RowID* представляет собой номер записи в списке<sup>9</sup>.

Рассмотрим, как можно организовать работу с окном списка. Считаем, что в системе загружены таблицы **Lots**, **Points**, **OMZ** и в программе определена общая переменная *idWinList*.

#### Код: Создание окна списка

```

sub WinDemoList
'Создаем новое окно списка для таблицы Lots
Browse * From Lots
    Position (20,20) Units "mm"
    Width 100 Units "mm"
    Height 70 Units "mm"
'Определяем идентификатор этого окна
idWinList=FrontWindow()
'Для окна idWinList отключаем показ сетки
Set Browse Window idWinList Grid Off
'Уточняем положение окна и добавляем новый заголовок окна
Set Window idWinList
    Position(30,30) Units "mm"
    Title "Участки"
'Создаем новое окно списка для таблицы Points с тремя полями
Browse CN,X,Y From Points
    Position (50,20) Units "mm"
    Width 150 Units "mm"
    Height 170 Units "mm"
'Самой верхней в окне списка будет строка с номером 50
    Row 50
'Определяем идентификатор этого окна
idWinList=FrontWindow()

```

---

<sup>9</sup> После удаления записи нумерация не изменяется, пересчет выполняется только после упаковки таблицы.

```
'Добавляем новый заголовок окна
Set Window idWinList Title "Точки"
end sub
```

## Тема 12. Создание окна графика

В окне графика данные из таблицы (и результаты вычисления выражений) представляются в виде графика определенного типа. Простейший график, показанный на рис. 10, построен следующей процедурой MapBasic.

### Код: Создание графика

```
sub WinDemoGraph
Graph Название,Площ_проект,Площ_факт From Насел_пункты
  Position (20,20) Units "mm"
  Width 150 Units "mm"
  Height 120 Units "mm"
idWinGraph=FrontWindow()
Set Window idWinGraph Title "Населенные пункты"
Set Window Legend
  Position (100,50) Units "mm"
  Width 60 Units "mm" Height 30 Units "mm"
Open Window Legend
Set Graph
  Type Bar Stacked Off Overlapped On Droplines Off Rotated On Show3D Off
  Overlap 30 Gutter 10 Angle 0
  Title "Площадь населенных пунктов" Font ("Arial Cyr",1,18,0)
Set Graph Label Axis
  Major Tick Outside Major Grid Off Pen (1,2,0)
  Minor Tick None Minor Grid Off Pen (1,2,0)
  Labels At Axis Font ("Arial Cyr",0,8,0)
  Pen (1,2,0)
  Title "Название" Font ("Arial Cyr",1,10,0)
Set Graph Value Axis
  Major Tick Outside Major Grid On Pen (1,2,0)
  Minor Tick None Minor Grid Off Pen (1,2,0)
  Labels At Axis Font ("Arial Cyr",0,8,0)
  Pen (1,2,0)
  Title "Площадь (га)" Font ("Arial Cyr",1,10,0)
Set Graph Series 2
  Title "Площадь проектная"
Set Graph Series 3
  Title "Площадь фактическая"
Set Graph Legend
  Title "Площадь" Font ("Arial Cyr",0,10,0)
  Subtitle "населенных пунктов" Font ("Arial Cyr",0,8,0)
  Range Font ("Arial Cyr",0,8,0)
end sub
```

Здесь предполагается, что уже загружена таблица **Насел\_пункты**. В таблице имеются поля: *Название*, *Площ\_проект*, *Площ\_факт*.

Дадим некоторые полезные рекомендации облегчающие построение кода.

Построим график в MapInfo, так как нам нужно. Далее возможно:

1. Сохранить этот график как шаблон (*График /Сохранить как шаблон...*). Тогда в коде программы можно использовать оператор типа Graph Название, Площ\_проект, Площ\_факт From Насел\_пункты Using "C:\Templates\grTown.3tf" Series In Columns, где **grTown.3tf** сохраненный ранее шаблон. И дополнительно можно кое-что подправить, применяя оператор Set Graph.

- Сохранить *Рабочий набор*. Находим в *Рабочем наборе* код, относящийся к построению графика и легенды. В итоге, остается только адаптировать этот код к логике программы.

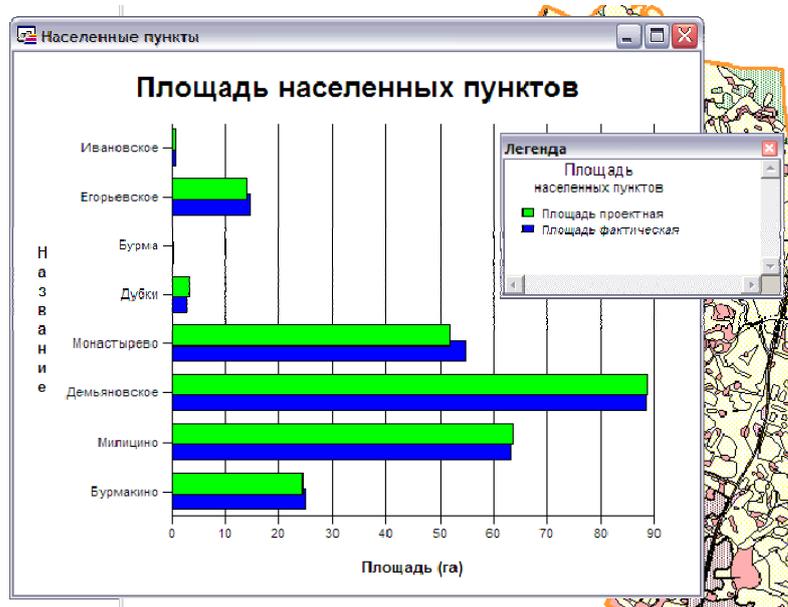


Рис. 10

### Тема 13. Создание окна отчета

Рассмотрим пример построения простого отчета. Предполагается, что открыто окно карты и для него определен идентификатор `idWinMap`. Среди открытых таблиц есть таблица **Насел\_пункты**.

Код: Формирование отчета

```
sub WinDemoLayout
'Устанавливаем единицы измерения для размеров и положения окон на экране
Set Paper Units "mm"
'Определяем координатную систему для отчета10
Set CoordSys Layout Units "mm"
'Создаем новое окно отчета
Layout Position (30, 30) Width 100 Height 170
'Определяем идентификатор окна
idWinLayout=FrontWindow()
'Настраиваем отображение содержимого окна отчета
Set Layout Window idWinLayout Extents To Fit Pagebreaks On
    Frame Contents Active
    Ruler On
    Zoom To Fit
'Вставляем текстовый объект в окно Отчета11
Create Text Into Window idWinLayout "Пример простого отчета"
    (70,20) (71,21) Font ("Arial", 513,14,BLACK,WHITE)
```

<sup>10</sup> Начало отсчета верхний левый угол листа, ось Y направлена вниз листа, ось X – вправо.

<sup>11</sup> Пары координат в скобках определяют точки: (верхний левый угол текстовой области) (противоположный по диагонали угол). В окне отчета вторая пара координат не учитывается. Достаточно чтобы эти значения были немного больше значений координат первой пары.

'В окне отчета (idWinLayout) создаем объект фрейм (рамка)

'в который впишем окно карты (idWinMap)

```
Create Frame Into Window idWinLayout (20,50) (170,200) Pen (1,2,0)
  Brush (2,16777215,16777215)
  From Window idWinMap
  FillFrame On
```

'Определим выборку из таблицы Насел\_пункты

```
Select Название, Площ_проект, Площ_факт
  from Насел_пункты order by Название into Selection
```

'Откроем выборку в окне списка

```
Browse * From Selection
```

'Определяем идентификатор окна списка

```
idWinList=FrontWindow()
```

'В окне отчета (idWinLayout) создаем объект фрейм (рамка)

'в который впишем окно списка (idWinList)

```
Create Frame Into Window idWinLayout (100,180) (200,250) Pen (1,1,0)
  Brush (2,16777215,16777215)
  From Window idWinList
  FillFrame On
```

'Устанавливаем новый заголовок окна отчета

```
Set Window idWinLayout Title "Новый отчет"
end sub
```

Отчет, созданный данным кодом, показан на рис. 11.

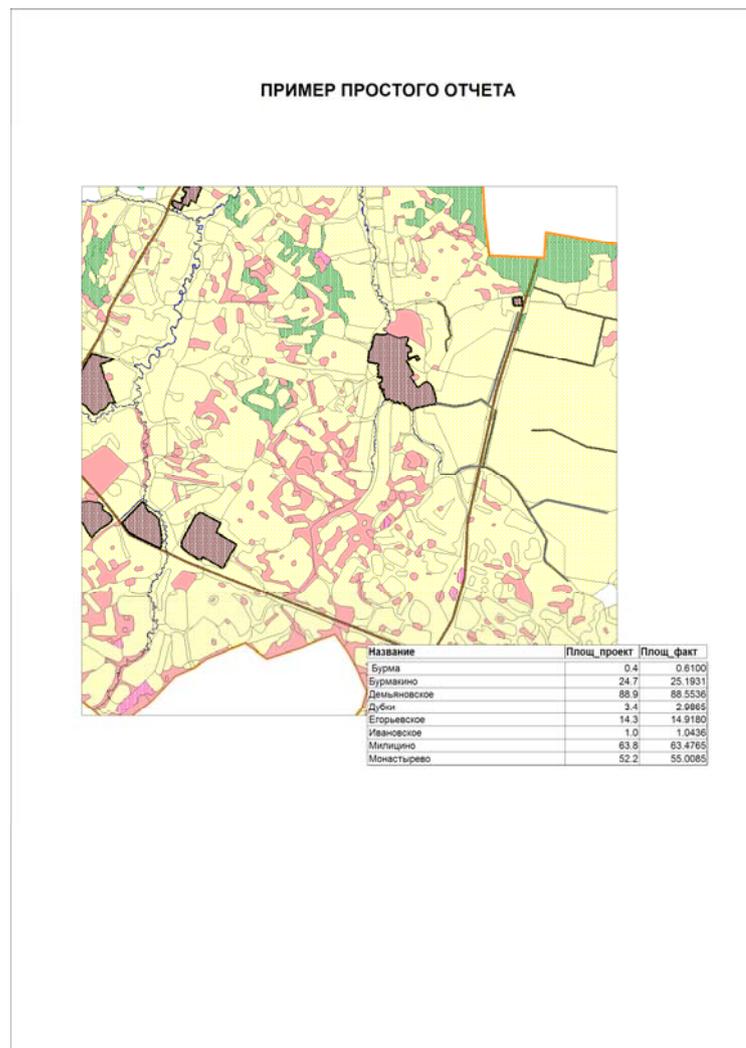


Рис. 11

Формирование отчета непосредственно в коде, что называется с чистого листа, занятие утомительное и неблагодарное. Значительно упростить эту задачу можно, если построить отчет в MapInfo и сохранить *Рабочий набор*. Весь код определяющий отчет будет записан в *Рабочем наборе*<sup>12</sup>. Далее можно скопировать весь этот код и оформить его как отдельную процедуру программы. Конечно, что-то придется добавить и изменить, но в любом случае это несравнимо с затратами на написание кода в полном объеме.

### Тема 14. Обращение к окну «Статистика»

Приведем пример обращения к окну **Статистика**. Результат выполнения процедуры представлен на рис. 12.

Код: Окно Статистика

```
sub WinDemoStatistics
'Установим единицы измерений
Set Paper Units "mm"
'Откроем окно Статистика
Open Window WIN_STATISTICS
'Уточним положение и размер окна
Set Window WIN_STATISTICS
    Position (30,30)
    Width 100
    Height 50
    Show
'Организуем выборку из таблицы Насел_пункты
Select * From Насел_пункты
end sub
```

Поле	Сумма	Среднее
Площ_проект	248.7	31.0875
Площ_факт	251.79	31.4737

Рис. 12

### Тема 15. Обращение к окну «Сообщения»

Для вывода информации в окно **Сообщения** используется оператор `Print`. Этот оператор часто используется при отладке MapBasic программ.

Код: Окно Сообщения

```
sub WinDemoMessage
dim i as smallint
'Очистка окна Сообщения
print Chr$(12)
'Цикл по числу открытых окон (Карта, Список и т.д.)
For i=1 to NumWindows()
'Выводим заголовок окна
    print WindowInfo(i,WIN_INFO_NAME)
Next
Set Paper Units "mm"
```

<sup>12</sup> Обратите внимание на установленные единицы измерений в Рабочем наборе.

'Уточним положение и размер окна

```
Set Window Message  
    Position (30,30)  
    Width 60  
    Height 50  
    Show
```

'Установим новый заголовок окна

```
    Title "Открытые окна"  
end sub
```

Иногда окно **Сообщения** оказывается скрыто от пользователя. Для того чтобы восстановить его введите в окно **MapBasic** операторы:

```
Open Window Message  
Set Window Message Position (2,2).
```

По умолчанию для «бумажных» единиц измерения установлены дюймы, то есть новое окно будет размещено на 2 дюйма левее и на 2 дюйма ниже верхнего левого угла экрана.

### **Тема 16. Обращение к окну «Информация»**

Окно **Информация**, в MapInfo, открывается, когда пользователь начинает использовать инструмент *Информация* в окне карты или списка. Запустить этот инструмент программно можно с помощью оператора `Run Menu Command M_TOOLS_PNT_QUERY`, где константа `M_TOOLS_PNT_QUERY` равна 1707.

Приведем пример программного обращения к окну **Информация**. Результат выполнения процедуры представлен на рис. 13.

#### **Код: Окно Информация**

```
sub WinDemoInfo  
Set Paper Units "mm"  
'Откроем окно Информация  
Open Window Info  
'Уточним положение и размер окна  
Set Window Info  
    Position (30,30)  
    Width 60  
    Height 50  
    Show  
'Добавим новый заголовок окна  
    Title "Населенный пункт"  
Set Window Info  
'Установим курсор: таблица Насел_пункты, запись 1.  
    Table Насел_пункты Rec 1  
'Запретим редактирование данных  
    ReadOnly  
'Установить окно поверх остальных окон  
    Front  
end sub
```

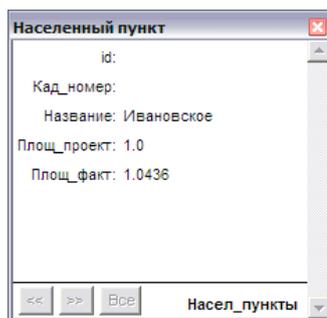


Рис. 13

### Тема 17. Обращение к окну «Легенда»

Работа с легендами имеет много деталей, как в части содержания, так и оформления. Для начала отметим, что легенды бывают:

- ✓ Картографические
- ✓ Тематические

Картографические легенды создаются для слоя (слоев) карты<sup>13</sup>. Для одной карты можно сформировать несколько картографических легенд. Приведем пример кода процедуры, которая строит легенду для карты. Результат выполнения процедуры показан на рис. 14.

#### Код: Картографическая легенда

```
sub WinDemoLegend
dim LrName as string
LrName="Угодья"
Set Paper Units "mm"
Create Cartographic Legend
    From Window idWinMap
    Position (30,30)
    Width 45 Height 80
    Portrait
    Style Size Large
    Frame From Layer LrName
    Border Pen (0,1,0)
    Title "Угодья" Font ("Arial Cyr",1,10,0)
    Subtitle "типы угодий" Font ("Arial Cyr",0,10,0)
    Style Font ("Arial Cyr",0,8,0) Norefresh
    Text "Пашня" Region Pen (1,2,0) Brush (2,14737632)
    Text "Пастбище" Region Pen (1,2,0) Brush (85,8421504,16774352)
    Text "Сенокос" Region Pen (1,2,0) Brush (2,13500288)
    Text "Кустарник" Region Pen (1,2,0) Brush (48,0,10551200)
    Text "Болото" Region Pen (1,2,0) Brush (138,32768,10551248)
    Text "Застройка" Region Pen (1,2,0) Brush (2,16756912)
    Text "Другие" Region Pen (1,2,0) Brush (5,0,13684944)
    Text "Залежь" Region Pen (1,2,0) Brush (66,14680288,16765160)
end sub
```

В данном случае легенда строится для одного слоя **Угодья**, если слоев будет больше, то соответственно увеличится число блоков `Frame`. Построенная легенда имеет фиксированные стили оформления и подписи, т.е. при обновлении она не будет изменяться.

Заменим, в данном коде, оператор создания легенды на следующий:

```
Create Cartographic Legend
    From Window idWinMap
```

<sup>13</sup> В том числе и для тематических слоев.

```

Window Title "Угодья"
Scrollbars Off
Portrait
Style Size Large
Default Frame Title "# "
    Font ("Arial Cyr",0,10,0)
Default Frame Style "%"
    Font ("Arial Cyr",0,8,0)
Frame From Layer LrName
    Using column object label "Тип "+Тип
    
```

В полученной таким образом легенде, стили оформления будут соответствовать стилям оформления объектов в слое, а подписи будут выбираться из указанного поля таблицы. При изменении оформления какого-либо типа объектов в слое достаточно обновить стили в легенде (Set Cartographic Legend Window idWinLegend Refresh). Какой вариант построения картографической легенды выбрать зависит от имеющихся данных и целей автора программы.

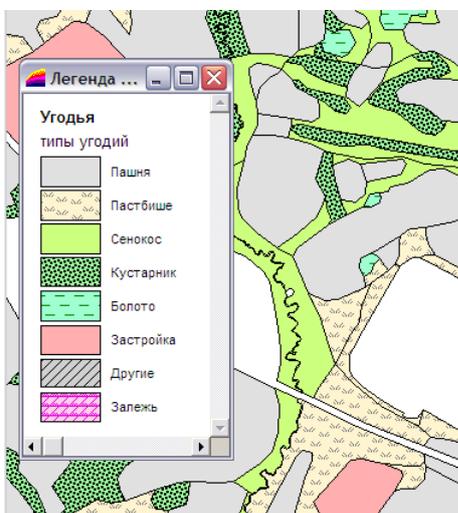


Рис. 14

Тематическая легенда может быть построена исключительно для тематического слоя карты. Приведем код реализующий построение тематического слоя карты и легенды для этого слоя.

### Код: Тематическая легенда

```

sub WinDemoLegend2
dim LrName as string
LrName="Угодья"
Set Paper Units "mm"
'Построение тематического слоя
shade Window idWinMap
    LrName with Балл ranges apply all use color Brush (2,15597520,16777215)
    1: 7 Brush (2,15597520,16777215) Pen (1,2,0) ,
    7: 12 Brush (2,11591840,16777215) Pen (1,2,0) ,
    12: 18 Brush (2,7389296,16777215) Pen (1,2,0) ,
    18: 24 Brush (2,4235312,16777215) Pen (1,2,0) ,
    24: 30 Brush (2,32768,16777215) Pen (1,2,0)
    default Brush (2,16777215,16777215) Pen (1,2,0)
    style replace off
'Открыть окно легенды и уточнить его положение и размеры
Open Window Legend
Set Window Legend
    
```

```

Position (30,30)
Width 40 Height 40
'Построить легенду
Set legend
layer 1
display on
shades on
symbols off
lines off
count on
title "Угодья в баллах" Font ("Arial Cyr",1,10,0)
subtitle "сентябрь 2010" Font ("Arial Cyr",0,8,0)
ascending off
ranges Font ("Arial Cyr",0,8,0)
auto display off ,
auto display on ,
auto display on ,
auto display on ,
auto display on ,
auto display on
end sub

```

Результат работы процедуры показан на рис. 15.

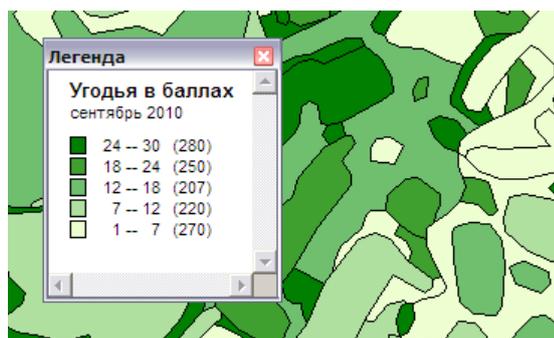


Рис. 15

## Тема 18. Обращение к окну «MapBasic»

Окно **MapBasic** в MapInfo<sup>14</sup> используется в следующих целях:

- ✓ **Интерактивное управление объектами MapInfo.** Непосредственно в окне можно вводить операторы языка MapBasic для управления окнами, таблицами, объектами и т.д. Круг используемых операторов достаточно широк, однако есть и исключения. В окне **MapBasic** нельзя использовать управляющие операторы, циклы и т.п., нельзя использовать константы, описываемые в файлах **MapBasic.DEF** и **Menu.DEF**. Длина оператора не более 256 символов. Например, введем в окно MapBasic оператор `browse Тип,Балл from Угодья` и нажмем клавишу **Enter**. В результате будет открыто окно **Списка** с выборкой двух полей *Тип* и *Балл* из таблицы **Угодья**.
- ✓ **Изучение языка программирования MapBasic.** При выполнении многих действий в MapInfo, в окне **MapBasic**, параллельно выводится протокол этих действий в терминах языка программирования MapBasic. Это свойство программы позволяет изучать и анализировать поведение системы при выполнении различных операторов MapBasic. Например, в диспетчере слоев для слоя **Угодья** настроим показ подписей: текст из колонки *Балл*, шрифт "Arial Cyr" красного цвета с белой каймой, жирный,

<sup>14</sup> Можно проследить некоторую аналогию с командной строкой в AutoCAD.

размер – 9. В окне **MapBasic** будут выведены строки соответствующие этому действию:

```
set map redraw off
Set Map Layer 2 Label Font ("Arial Cyr",261,9,16711680,16777215)
    With Балл Auto On Overlap On
set map redraw on
```

- ✓ **Отладка программ написанных на MapBasic.** Отладка основана на использовании оператора `stop` определяющего точку прерывания. Достигнув этой точки процесс прерывается. В окне **MapBasic** выводится комментарий следующего вида " \*\* Контрольная точка: C:\p10b.mb строка 48". Теперь можно определить следующую информацию доступную в данной контрольной точке:
1. Список локальных переменных (команда `?Dim`).
  2. Список всех глобальных переменных (команда `?Global`).
  3. Значение переменной в данной точке (команда `?ИмяПеременной`).
  4. Можно изменить значение переменной, доступной в данной точке (команда `?ИмяПеременной=Значение`).
- Затем продолжаем выполнение MapBasic программы (команда `Continue`). Вид окна **MapBasic**, с записью процедуры отладки, показан на рис. 16.

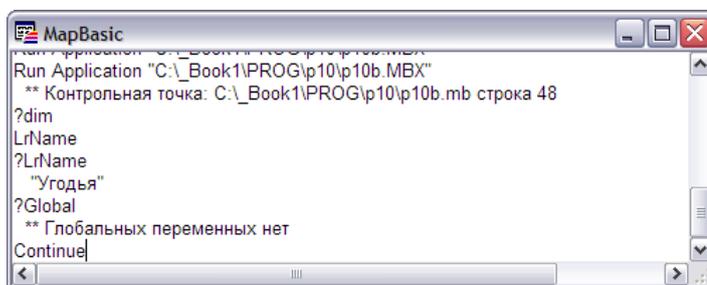


Рис. 16

На практике данная процедура отладки применяется редко вследствие значительных ограничений на использование оператора `Stop`:

1. Нельзя использовать внутри конструкции `Function...End Function`.
2. Нельзя использовать внутри конструкции `Dialog`.
3. Нельзя использовать в операторе `ProgressBar`.
4. Отлаживаемая программа должна быть единственной работающей MapBasic программой в системе MapInfo (включая и MapBasic программы находящиеся в режиме ожидания).

Непосредственное обращение к окну **MapBasic** из пользовательских программ встречается редко, да и необходимости в этом вообще то нет. Программно можно открыть/закрыть окно, установить его положение и размеры, настроить шрифт как это показано в следующем коде.

```
Sub Main
Set Paper Units "mm"
Open Window MapBasic
Set Window MapBasic Position(50,90)
    Width 100 Height 30
    Font MakeFont("Helvetica",1,10,BLACK,WHITE)
end sub
```

### Тема 19. Добавить кнопку к стандартной панели

Ниже представлена процедура, позволяющая добавить к инструментальной панели **Пенал** разделитель и новую кнопку типа `PushButton`.

Код: Добавить кнопку к панели Пенал

```
Sub AddButton
Alter ButtonPad ID 2 'панель Пенал
    Add Separator
    Add PushButton
        Icon MI_ICON_MB_11
        Calling Setka_kv
        HelpMsg "Построение прямоугольной сетки\nПостроение сетки"
    Show
End sub
```

Ссылка на иконку кнопки дана в виде именованной константы MI\_ICON\_MB\_11. Для возможности использовать в коде подобные константы необходимо добавить к модулю строку `Include "ICONS.DEF"` или использовать константу в явном виде `Icon 185`.

### **Тема 20. Создать новую инструментальную панель**

Принцип создания новой инструментальной панели демонстрирует следующий код.

#### **Код: Создать новую панель**

```
Sub CreateButtonPad
Create ButtonPad "LOTs" As
    PushButton
        Icon MI_ICON_MB_1
        Calling DialogTab
        HelpMsg "Создать таблицы\nСоздать таблицы"
        Enable
    PushButton
        Icon MI_ICON_MB_4
        Calling DialogPlan
        HelpMsg "Печать планов участков\nПечать планов"
        Enable
    PushButton
        Icon MI_ICON_MISC_7
        Calling DialogPoints
        HelpMsg "Печать каталогов для участков\nПечать каталогов"
        Enable
    Separator
    PushButton
        Icon MI_ICON_ARROW_17
        Calling exitMe
        HelpMsg "Выход из программы\nВыход"
        Enable
    Width 5
    Position (0,5) Units "cm"
    Show
    Float
End sub
```

Данный код формирует панель из четырех кнопок `PushButton` и разделителя `Separator`. Положение панели `Position (0,5)` устанавливается относительно экрана.

### **Тема 21. Использование кнопки переключателя**

Текущее состояние кнопки отслеживается программно с помощью общей логической переменной (`isShow`). В процедуре обработчике события `ShowHideMessage` ее состояние меняется на противоположное и в зависимости от ее значения выполняется то или иное действие.

#### **Код: Переключатель для окна Сообщения**

```

Include "ICONS.DEF"
Declare Sub Main
Declare Sub ShowHideMessage
dim isShow as logical

Sub Main
Open Window Message
Set Window Message Position (2,2) Hide
isShow="F"
Alter ButtonPad ID 3
    Add Separator
    Add ToggleButton
        Icon MI_ICON_MISC_7
        ID 102
        Calling ShowHideMessage
        HelpMsg "Показать/скрыть окно СООБЩЕНИЯ\нокно СООБЩЕНИЯ"
        Enable
        Uncheck
    Show
end sub

Sub ShowHideMessage
isShow=not isShow
if isShow="T" then
    Set Window Message Show
else
    Set Window Message Hide
end if
end sub

```

### **Тема 22. Использование кнопки инструмента**

Напишем программу для инструмента рисующего полилинии.

**Код: Кнопка инструмент**

```

Include "MAPBASIC.DEF"
Include "ICONS.DEF"
Declare Sub Main
Declare Sub drawPLine

Sub Main
Alter ButtonPad ID 2
    Add Separator
    Add ToolButton
        Icon MI_ICON_MISC_12
        ID 103
        Cursor MI_CURSOR_FINGER_LEFT
        DrawMode DM_CUSTOM_POLYLINE
        Calling drawPLine
        HelpMsg "Нарисовать ось дороги\нокно дороги"
        Enable
    Show
end sub

Sub drawPLine
dim objPLine as Object
dim LName as string
'Определяем имя самого верхнего (не косметического) слоя.
'Предполагается что слой векторный.
LName=LayerInfo(FrontWindow(), 1, LAYER_INFO_NAME)

```

```
'Определяем объект - полилиния
objPLine =CommandInfo(CMD_INFO_CUSTOM_OBJ)
'Вставляем объект в выбранный слой
Insert Into LName (Obj) Values (objPLine)
end sub
```

В данном случае рисовка полилинии выполняется текущим стилем на верхнем слое карты. Однако ничего не стоит указать требуемый слой и установить нужный стиль линии. Делать слой редактируемым не требуется.

### Тема 23. Использование процедуры ToolHandler

*ToolHandler* это специальная процедура обработки событий. Если в программе есть процедура *ToolHandler*, то в инструментальную панель **Операции** будет добавлена новая кнопка типа *ToolButton* с пиктограммой "перекрестие"<sup>15</sup>. При нажатии на эту кнопку курсор принимает вид перекрестия и как только пользователь кликнет мышью в окне карты, списка или отчета будет вызвана процедура *ToolHandler*. Эту кнопку еще называют инструмент MapBasic.

Обработка в *ToolHandler* может быть весьма замысловатой и включать в себя кроме определения активного окна также анализ нажатия клавиш клавиатуры **Shift** и **Ctrl** по отдельности или вместе.

#### Код: Пример использования инструмента MapBasic

```
Include "MAPBASIC.DEF"
Declare Sub Main
Declare Sub ToolHandler

Sub Main
Print chr$(12)
end sub

Sub ToolHandler
dim nmTable as string
dim rw,cl as integer
dim zz as string
dim aa as alias
If WindowInfo(FrontWindow(),WIN_INFO_TYPE)=WIN_MAPPER Then
'Если окно карты
    set CoordSys Window frontWindow()
    Print "КАРТА"
    Print "SHIFT=" & Str$(CommandInfo(CMD_INFO_SHIFT))
    Print "CTRL=" & Str$(CommandInfo(CMD_INFO_CTRL))
    Print "X=" & Str$(CommandInfo(CMD_INFO_X))
    Print "Y=" & Str$(CommandInfo(CMD_INFO_Y))
    Print "-----"
End If
If WindowInfo(FrontWindow(),WIN_INFO_TYPE)=WIN_BROWSER Then
'Если окно списка
    nmTable=WindowInfo(FrontWindow(),WIN_INFO_TABLE)
'Строка
    rw=CommandInfo(CMD_INFO_Y)
'Столбец
    cl=CommandInfo(CMD_INFO_X)
```

<sup>15</sup> Нужно иметь в виду, что если запущено несколько программ MapBasic одновременно и каждая из них содержит процедуру *ToolHandler*, то каждая из программ добавляет в панель "Операции" свою кнопку.

```

Print "СПИСОК"
Print "Имя таблицы=" & nmTable
Print "Номер строки=" & Str$(rw)
Print "Номер колонки=" & Str$(cl)
if cl=0 then cl=cl+1 end if
Fetch Rec rw From nmTable
aa=nmTable & ".Col" & str$(cl)
zz=aa
Print "Значение=" & Str$(zz)
Print "-----"
End If
End Sub

```

На рис. 17 показано как выглядит результат работы программы.

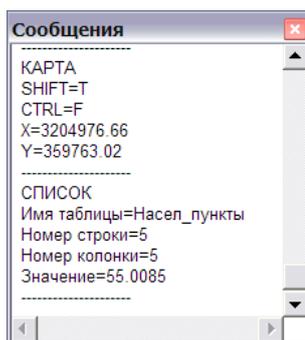


Рис. 17

### Тема 24. Использование функций Win32 API

Интерфейс прикладного программирования 32-разрядной операционной системы *Windows* (*Win32 API*), по определению, служит для связи языков программирования высокого уровня с низкоуровневыми компонентами операционной системы. Необходимость в использовании обычно возникает тогда когда штатных средств языка недостаточно для решения задачи.

Приведем пример небольшой программы демонстрирующей возможность использования функций *Win32 API*.

Блок описаний программы включает: подключение файла **mapbasic.def**, описание констант, описание структуры **RECT**, описание процедур, в том числе процедур *Win32 API*.

**Код: Демонстрация использования отдельных функций Win32 API**

```

'Блок описаний
Include "mapbasic.def"
Define HorzSize      4
Define VertSize      6
Define HorziRes      8
Define VertiRes      10
Define LogPixelsX    88
Define LogPixelsY    90
Define MAX_COMPNAME_LENGTH 31
Type RECT
    left as Integer
    top as Integer
    right as Integer
    bottom as Integer
End Type
Declare Function GetWindowRect
    Lib "User32" (ByVal hWnd as integer,lpRect As RECT) as integer
Declare Function GetDC Lib "User32"(ByVal hWnd As integer) As integer

```

```

Declare Function GetDeviceCaps
    Lib "GDI32" (ByVal hDC As integer, ByVal nIndex As SmallInt) As SmallInt
Declare Function SetCursorPos
    Lib "user32" (ByVal x As integer, ByVal y As integer) As integer
Declare Function GetComputerName Lib "kernel32"
    Alias "GetComputerNameA" (lpBuffer As String, nSize As SmallInt)
    As SmallInt
Declare Sub Main
Declare function GetCompName() as string
Declare Sub ScreenInfo
Declare sub GetWinInfo (bheight as SmallInt, bwidth as SmallInt,
    bleft as SmallInt, bright as SmallInt,
    btop as SmallInt, bbottom as SmallInt)

```

'Главная процедура

```

Sub Main
dim h,w,l,r,t,b as smallint
dim ret as integer
call GetWinInfo (h,w,l,r,t,b)

```

'Получили информацию по главному окну MapInfo

```

print chr$(12)
print h & ", " & w
print l & ", " & r
print t & ", " & b
print "-----"

```

'Устанавливаем курсор в левый верхний угол окна MapInfo

```
ret = SetCursorPos(l, t)
```

'Получаем информацию по экрану компьютера

```
call ScreenInfo
```

'Определяем имя компьютера

```
print GetCompName()
end sub
```

'Процедура GetWinInfo.

'Получаем информацию по ограничивающему прямоугольнику главного окна MapInfo<sup>16</sup>:

'высота, ширина, координаты левого, правого, верхнего, нижнего края окна (пиксели).

```

sub GetWinInfo (bheight as SmallInt, bwidth as SmallInt,
    bleft as SmallInt, bright as SmallInt, btop as SmallInt,
    bbottom as SmallInt)
dim hwnd,k as integer
Dim lpRect as RECT

```

'Определение манипулятора главного окна MapInfo

```
hwnd=SystemInfo (SYS_INFO_MAPINFOWND)
```

'Получаем информацию, при успехе (k>0)

'она будет записана в переменную lpRECT типа RECT

```

k=GetWindowRect (hwnd, lpRECT)
if k=0 then
    bheight=0
    bwidth=0
    bleft=0
    btop=0
    bright=0
    bbottom=0
else
    bheight = lpRect.bottom - lpRect.top
    bwidth = lpRect.right - lpRect.left
    bleft= lpRect.left

```

---

<sup>16</sup> Для всего окна включая рамку, заголовок, полосы прокрутки, меню и т.д.

```

        btop= lpRect.top
        bright= lpRect.right
        bbottom= lpRect.bottom
end if
end sub

```

### 'Процедура ScreenInfo

```

Sub ScreenInfo
Dim hwnd as integer
Dim ScreenhDc as integer
Dim HSize, VSize as SmallInt
hwnd=WindowInfo(frontwindow(),WIN_INFO_WND)
ScreenhDc = GetDc(hwnd)
'Горизонтальный и вертикальный размеры экрана в пикселах
HSize = GetDeviceCaps(ScreenhDc,HorziRes)
VSize = GetDeviceCaps(ScreenhDc,VertiRes)
Print "H,V of Screen px= " & Str$(HSize) & "X" & Str$(VSize)
'Горизонтальный и вертикальный размеры экрана в миллиметрах
HSize = GetDeviceCaps(ScreenhDc,HorzSize)
VSize = GetDeviceCaps(ScreenhDc,VertSize)
Print "H,V of Screen mm= " & Str$(HSize) & "X" & Str$(VSize)
'Горизонтальный и вертикальный размеры экрана в логич. пикселах
HSize = GetDeviceCaps(ScreenhDc,LogPixelsX)
VSize = GetDeviceCaps(ScreenhDc,LogPixelsY)
Print "H,V of Screen log px= " & Str$(HSize) & "X" & Str$(VSize)
End Sub

```

### 'Процедура GetCompName

```

function GetCompName() as string
Dim dwLen,k As SmallInt
Dim strString As String
dwLen=MAX_COMPNAME_LENGTH + 1 'длина буфера
strString = String$(dwLen,"X")
k=GetComputerName(strString,dwLen)
GetCompName = Left$(strString,dwLen)
end function

```

Приведем еще один небольшой пример. В MapBasic есть функция `Timer()` возвращающая время в целых секундах, а если нужен более точный результат, можно использовать функцию API — `GetTickCount()`, возвращающую время в миллисекундах.

#### Код: Определение времени в миллисекундах

```

Declare Function GetTickCount Lib "kernel32.dll" () As Integer
...
Sub Timing
Dim theStart,theEnd,i As Integer
theStart=GetTickCount()
'исследуемый процесс
for i=1 to 1000
    print chr$(12)
next
theEnd=GetTickCount()
print str$((theEnd-theStart)/1000) & " сек"
end sub

```

При работе в MapInfo несколько клавиш на клавиатуре можно использовать в качестве переключателей режимов окна карты.

Клавиша <sup>17</sup>	Режим (вкл./выкл.)	Комментарий в строке состояния
S	Точное позиционирование	УЗЛЫ
T	Автотрассировка	АВТОТРАССИРОВКА
N	Автоматическая (потоковая) трассировка	ПОТОК
C	Размер перекрестия (указателя) во все окно	—

Напишем программу, которая:

- ✓ определяет раскладку клавиатуры и если нужно переключает ее;
- ✓ имитирует нажатие клавиш на клавиатуре для изменения режима окна карты.

Определимся с ограничениями:

- ✓ в кольцевом списке две раскладки клавиатуры ("00000409" (ENG) и "00000419" (RUS));
- ✓ нажатие клавиш имитируется только для символов: S, T, N, C.

**Код: Переключатель режимов окна карты**

'Блок описаний

```

Include "mapbasic.def"
Define KEYEVENTF_KEYUP &H2
Define KL_NAMELENGTH 9
Define HKL_NEXT 1
Declare Sub Main
Declare Sub SetC
Declare Sub SetS
Declare Sub SetT
Declare Sub SetN
Declare Sub theEnd
Declare sub SendKey(byval c as string)
Declare Sub keybd_event Lib "user32.dll"
    (ByVal bVk As SmallInt, ByVal bScan As SmallInt,
    ByVal dwFlags As integer, ByVal dwExtraInfo As integer)
Declare function GetKeyboardLayoutName Lib "User32"
    Alias "GetKeyboardLayoutNameA"
    (pwszKLID as string * KL_NAMELENGTH) as integer
Declare Function ActivateKeyboardLayout Lib "user32"
    (ByVal HKL As integer, ByVal flags As integer) As integer
    
```

'Главная процедура

```

Sub Main
Create Menu "SendKey" As
    "C" Calling SetC,
    "S" Calling SetS,
    "T" Calling SetT,
    "N" Calling SetN,
    "(-",
    "Выход" Calling theEnd
Alter Menu "Программы" Add
    "(-",
    "SendKey" As "SendKey"
end sub

Sub SetC
call SendKey("C")
end sub
    
```

<sup>17</sup> Раскладка клавиатуры – английская.

```
Sub SetS
call SendKey("S")
end sub

Sub SetT
call SendKey("T")
end sub

Sub SetN
call SendKey("N")
end sub

sub SendKey(byval c as string)
dim vk,scan,dl as integer
dim dd as string * KL_NAMELENGTH
'Определяем раскладку клавиатуры
dl=GetKeyboardLayoutName(dd)
if RTrim$(LTrim$(dd))="00000419" then
'Выбрать следующий идентификатор языка в кольцевом списке
dl=ActivateKeyboardLayout(HKL_NEXT,0)
end if
'T.к нас интересует лишь четыре символа, подставим необходимые значения
'в требуемые параметры18:
'vk – код виртуальной клавиши для заданного символа
'scan – аппаратный скан-код символа
Do Case c
    Case "C"
        vk=67
        scan=46
    Case "S"
        vk=83
        scan=31
    Case "T"
        vk=84
        scan=20
    Case "N"
        vk=78
        scan=49
    Case Else
        exit sub
End Case
'Нажать клавишу
call keybd_event(vk,scan,0,0)
'Отпустить клавишу
call keybd_event(vk,scan,KEYEVENTF_KEYUP,0)
'Обновить окно
Run Menu Command 610
end sub

Sub theEnd
End Program
end sub
```

---

<sup>18</sup> При решении задачи в общем виде все эти коды должны определяться программно. Например, через функцию MapVirtualKey.

Конечно, в таком виде данная программа особого смысла не имеет. Ее цель лишь показать, как это можно сделать, чтобы при необходимости вы могли вставить эти функции в свой код.

В языке MapBasic нет функции диалога выбора папки. Напишем небольшую программу, которая строит такой диалог.

### Код: Диалог выбора папки

```

Include "mapbasic.def"
type BrowseInfo
    hwndOwner AS Integer
    iList as Integer
    sSelFolder AS String
    sCaption AS String
    iFlags AS Integer
    Null1 as Integer
    Nul2 as Integer
    Nul3 AS Integer
end type
Declare Sub Main
Declare Function SelectionFolder() as string
Declare Function SHBrowseForFolder LIB "Shell32.dll"
    Alias "SHBrowseForFolderW" (lpBrowseInfo AS BrowseInfo) AS Integer
Declare Sub CoTaskMemFree Lib "ole32.dll" (ByVal hMem As Integer)
Declare Function SHGetPathFromIDList Lib "shell32" (ByVal pidList As Integer,
    lpBuffer As String) As Integer

Sub Main
print chr$(12)
print SelectionFolder()
end sub

Function SelectionFolder() as string
Dim iRet as Integer
Dim sPath as SString
Dim lpBrowseInfo as BrowseInfo
lpBrowseInfo.iFlags=0
lpBrowseInfo.sSelFolder=""
lpBrowseInfo.iList = ""
lpBrowseInfo.sCaption = ""
lpBrowseInfo.hwndOwner=SystemInfo(SYS_INFO_APPLICATIONWND)
'Вызываем диалог
iRet=SHBrowseForFolder(lpBrowseInfo)
if iRet>0 then
'Получаем путь в виде строки
    sPath=string$(254," ")
    iRet=SHGetPathFromIDList(iRet,sPath)
'Освобождаем память
    Call CoTaskMemFree(iRet)
    SelectionFolder=sPath
else
    SelectionFolder="Cancel"
end if
end Function
    
```

Файлы инициализации удобное средство хранения настроек и свойств приложения. В следующей программе покажем, как можно работать с такими файлами в MapBasic.

### Код: Запись/чтение информации INI-файла

```

Declare Function WritePrivateProfileString Lib "kernel32" Alias
    
```

```

    WritePrivateProfileStringA" (ByVal lpSectionName As String,
    ByVal lpKeyName As String, ByVal lpString As String,
    ByVal lpIniFileName As String) As Integer
Declare Function GetPrivateProfileString Lib "kernel32" Alias
    "GetPrivateProfileStringA" (ByVal lpSectionName As String,
    ByVal lpKeyName As String, ByVal lpDefault As String,
    lpReturnedString As String, ByVal nSize As Integer,
    ByVal IniFileName As String) As Integer
Declare Sub Main
Declare Sub SetINI
Declare Sub GetINI

Sub Main
call SetINI
call GetINI
end sub

Sub SetINI
'Изменить/Добавить запись в INI-файл
dim length as integer
'Изменить запись в секции "Слой", ключ "L3" на "Реки и озера"
length=WritePrivateProfileString("Слой", "L3", "Реки и
озера", "C:\Temp\ini\myTest.ini")
'Добавить новую запись в секции "Файлы", ключ "New", значение "infoNew"
length=WritePrivateProfileString("Файлы", "New", "infoNew",
    "C:\Temp\ini\myTest.ini")
'Добавить новую секцию "Имена", ключ "Дороги", значение "RD1"
length=WritePrivateProfileString("Имена", "Дороги", "RD1",
    "C:\Temp\ini\myTest.ini")
end sub

Sub GetINI
'Чтение записей из INI-файла
dim length as integer
dim vv as string
'Прочитать запись из секции "Слой", ключ "L3"
length=512
vv=Space$(length)
length=GetPrivateProfileString("Слой", "L3", "No", vv, length, "C:\Temp\ini\myTest
.ini")
print "--> " & length & " == " & vv
'Прочитать запись из секции "Файлы", ключ "New"
length=512
vv=Space$(length)
length=GetPrivateProfileString("Файлы", "New", "No", vv, length, "C:\Temp\ini\myTe
st.ini")
print "--> " & length & " == " & vv
end sub

```

На рис. 18а показан файл **myTest.ini** до использования программы, а на рис. 18б – после использования программы.

Рассмотрим более подробно используемые функции.

`WritePrivateProfileString` – функция записывает строковое значение ассоциированное с заданным ключом в указанной секции файла инициализации. Возвращает ненулевое значение в случае успеха, ноль при неудаче.

Если указанный в параметрах функции ключ отсутствует в *INI-файле*, то он будет создан.

Если указанная в параметрах функции секция отсутствует в *INI-файле*, то она будет создана.

Если по указанному пути отсутствует заданный файл инициализации, то будет создан новый файл с указанным именем.

GetPrivateProfileString – функция читает строковое значение ассоциированное с заданным ключом в указанной секции файла инициализации. Возвращает ненулевое значение в случае успеха (количество байт скопированных в буфер), ноль при неудаче.

Параметры функций:

lpSectionName	Имя секции (регистр символов не учитывается).
lpKeyName	Имя ключа (регистр символов не учитывается).
lpString	Значение ключа.
lpIniFileName	Полное имя файла инициализации.
lpDefault	Строка по умолчанию, если заданный ключ не найден. Может быть пустой строкой.
lpReturnedString	Строковый буфер длиной не менее nSize символов
nSize	Максимальное количество символов записываемых в буфер.

[Файлы]  
 Ф1=info1  
 Ф2=info2  
 Ф3=info3  
 [Слой]  
 L1=Участки  
 L2=Угодья  
 L3=Реки  
 [Объекты]  
 obj1=r123  
 obj1=r456  
 obj1=r789

а

[Файлы]  
 Ф1=info1  
 Ф2=info2  
 Ф3=info3  
 New=infoNew  
 [Слой]  
 L1=Участки  
 L2=Угодья  
 L3=Реки и озера  
 [Объекты]  
 obj1=r123  
 obj1=r456  
 obj1=r789  
 [Имена]  
 Дороги=RD1

б

Рис. 18

## Глава 3. Работа с таблицами

Таблица является базовым понятием в системе MapInfo. Слой, список и т.д. это все лишь формы представления информации из таблицы. Именно поэтому важно уметь работать с таблицами. В данной главе рассмотрены некоторые приемы работы с таблицами от их создания и редактирования до вопросов геокодирования и формирования метаданных.

### Тема 25. Создание таблицы заданной структуры

Код: Создать новую таблицу (1)

```
Include "mapbasic.def"
Declare Sub Main
Declare Sub CreateTableMap(ByVal FullNameTable As String,sCols() as string,
    byval minX as float,byval maxX as float,
    byval minY as float,byval maxY as float,idmap as integer)
Declare function ItsMap() as integer
```

```
Sub Main
dim sColInfo(5) as string
dim fnmTab as string
dim idMapWin as integer
'Проверка наличия активного окна карты
idMapWin=ItsMap()
'Описание полей таблицы
sColInfo(1)="ID Integer"
sColInfo(2)="НомерУгодья Char(15) "
sColInfo(3)="ТипУгодья Integer"
sColInfo(4)="ПлощадьПроектГа Decimal(14,1) "
sColInfo(5)="Балл Integer"
'Полное имя новой таблицы
fnmTab="C:\Temp\01\Угодья2.tab"
'Устанавливаем систему координат для последующих
'операций с координатами в программе
Set CoordSys Nonearth Units "m" Bounds (3000000,0) (3500000,500000)
'Создаем таблицу
call CreateTableMap(fnmTab,sColInfo,0,500000,3000000,3500000,idMapWin)
'Добавляем новую запись в таблицу. Информация записывается в два поля:
'поле Obj – объект точка и поле Балл – значение 25.
Insert Into Угодья2 (obj,Балл) Values(CreatePoint(3100201.25,250252.52),25)
'Сохраняем таблицу
Commit Table Угодья2
end sub
```

'Создает новую таблицу и размещает ее в карте.

'Рабочая область для таблицы задается значениями координат minX, maxX, minY, maxY<sup>19</sup>.

'Параметр idmap определяет наличие окна Карты. Если idmap больше нуля

'то новая таблица будет добавлена к карте idmap, в противном случае будет

'создано новое окно карты для этой таблицы.

```
Sub CreateTableMap(ByVal FullNameTable As String,sCols() as string,
    byval minX as float,byval maxX as float,
    byval minY as float,byval maxY as float,idmap as integer)
dim i as smallint
```

<sup>19</sup> Здесь координаты задаются в системе координат принятой в геодезии (ось X направлена на север, а ось Y на восток).

```

dim str,nmTable as string
dim kk as string*1
kk=chr$(34)
'Имя файла без пути и расширения
nmTable=PathToTableName$(FullNameTable)
'Формируем оператор Create Table в виде строки
For i = 1 To Ubound(sCols)
    str=str & sCols(i) & ","
Next
str=Left$(str,Len(str)-1)
str="Create Table " & nmTable &
    " (" & str & ") File " & kk & FullNameTable &
    kk & " TYPE NATIVE CharSet " &
    kk & "WindowsCyrillic" & kk
'Создаем таблицу
Run Command str
'Разрешаем сопоставлять записям таблицы графические объекты.
'Если здесь исключить предложение CoordSys, то для таблицы будет
'использоваться система координат программы заданная оператором
'Set CoordSys в процедуре Main.
Create Map For nmTable CoordSys Nonearth Units "m" Bounds
    (minY,minX) (maxY,maxX)
if idmap>0 then
'Добавляем таблицу к карте
    Add Map Window idmap Auto Layer nmTable
Else
'Создаем новую карту
    Map From nmTable
    idmap=FrontWindow()
end if
End Sub

function ItsMap() as integer
dim map_id as integer
    ItsMap=0
    map_id = FrontWindow()
    If WindowInfo(map_id,WIN_INFO_TYPE) <> WIN_MAPPER Then
        Exit function
    End If
    ItsMap=map_id
end function

```

Другой вариант создания таблицы представляет процедура `CreateTableMapFromLayer`. Новая таблица создается со структурой и координатной системой таблицы-шаблона **fromLayer**<sup>20</sup>.

**Код: Создать новую таблицу (2)**

```

Declare Sub Main
Declare Sub CreateTableMapFromLayer(ByVal FullNameTable As String,
    byval fromLayer as string)

Sub Main
dim fnmTab as string
fnmTab="C:\Temp\01\Угодья3.tab"
call CreateTableMapFromLayer(fnmTab,"Угодья")

```

<sup>20</sup> Эта таблица должна быть открыта, она не может быть таблицей запроса или растровой таблицей.

```
Map From PathToTableName$(fnmTab)
end sub

Sub CreateTableMapFromLayer(ByVal FullNameTable As String,
    byval fromLayer as string)
dim nmTable as string
nmTable=PathToTableName$(FullNameTable)
Create Table nmTable Using fromLayer File FullNameTable
Create Map For nmTable CoordSys Table fromLayer
End Sub
```

## Тема 26. Изменение структуры таблицы

В процессе работы с таблицей может возникать необходимость в изменении ее структуры: какие-то поля нужно удалить, какие-то добавить, какие-то нужно проиндексировать и т.д. Основным оператором, изменяющим структуру открытой таблицы, является оператор `Alter Table`. Используя этот оператор, можно:

- ✓ Добавить новое поле.
- ✓ Удалить поле.
- ✓ Изменить имя и/или тип поля.
- ✓ Изменить порядок полей в записи.

Заметим, что изменяемая таблица не должна иметь не сохраненных записей. После выполнения оператора таблица будет удалена из карты, если до этого она там присутствовала.

Кроме `Alter Table` имеются и другие операторы, предназначенные для изменения структуры таблицы: `Create Index`, `Drop Index`, `Create Map`, `Drop Map`, `Add Column`.

В примере будет использоваться таблица **Угодья2** построенная в предыдущем параграфе. Напомню, что сформировали мы ее со следующей структурой:

ID	Integer
НомерУгодья	Char(15)
ТипУгодья	Integer
ПлощадьПроектГа	Decimal(14,1)
Балл	Integer

Кроме того добавлена новая таблица **Типы**, являющаяся по сути справочником по типам угодий. Таблица состоит из двух полей (*Тип*, *Имя*) и не имеет возможности подключать графические объекты.

### Код: Демонстрация использования операторов изменяющих структуру таблицы

```
Declare Sub Main
Declare Sub changeTab1
Declare Sub changeTab2

Sub Main
call changeTab1
call changeTab2
end sub

Sub changeTab1()
Commit Table Угодья2
Alter Table Угодья2(
'Добавить два поля
    Add ИмяТипа Char(20), Оценка Integer
'Удалить поле
    Drop Балл
'Переименовать два поля
    Rename ПлощадьПроектГа Площадь, НомерУгодья ИмяУгодья)
```

'Изменить порядок полей

```
Alter Table Угодья2 (Order ID,ИмяУгодья, ТипУгодья, ИмяТипа, Площадь, Оценка)
```

'Создать индексы для двух полей

```
Create Index On Угодья2 (ID)
Create Index On Угодья2 (ТипУгодья)
end sub
```

```
Sub changeTab2 ()
```

'В таблице Угодья2 поле ИмяТипа заполняется названиями типа из Типы.Имя

'Соответствие устанавливается по полям Угодья2.ТипУгодья и Типы.Тип

```
Add Column Угодья2 (ИмяТипа) From Типы
Set To Типы.Имя Where ТипУгодья=Тип
```

'В таблице Типы создается временное поле CountType в котором отображается

'сумма угодий, из таблицы Угодья2, определенного типа

```
Add Column Типы (CountType) From Угодья2
Set To Count (*) Where Тип=ТипУгодья
```

```
end sub
```

На рис. 19 показаны результаты работы программы.

Нужно отметить, что повторный запуск данной программы закончится генерированием ошибки, так как структура таблицы **Угодья2** изменилась и поле *ИмяТипа* и *Оценка* уже существуют, а поля *Балл* вообще нет и т.д. В отношении процедуры *changeTab2* можно сказать следующее: если обновляется постоянное поле (*ИмяТипа*), то никаких проблем не будет, а добавление нового временного поля с тем же именем (*CountType*) закончится ошибкой. Из всего этого следует, что подобные операторы нужно использовать аккуратно. Необходимо вводить в код проверки существования полей, а также проектировать обработку ошибок.

The image shows two side-by-side window screenshots. The left window, titled 'Угодья2 Список', displays a table with columns: ID, ИмяУгод, ТипУгодья, ИмяТипа, Площадь, and Оценка. The right window, titled 'Типы Список', displays a table with columns: Тип, Имя, and CountType.

ID	ИмяУгод	ТипУгодья	ИмяТипа	Площадь	Оценка
18		6	болото	1.4	19
19		3	сенокос	0.3	26
20		5	кустарник	0.3	7
21		5	кустарник	0.2	2
22		5	кустарник	1.8	3
23		1	пашня	0.7	25

Тип	Имя	CountType
0	не определен	1
1	пашня	333
2	пастбище	127
3	сенокос	300
4	лес	0
5	кустарник	293
6	болото	123

Рис. 19

## Тема 27. Получение информации о таблице

Для получения информации о таблице наиболее часто используется оператор *TableInfo*. Приведем пример использования этого оператора.

**Код: Использование оператора TableInfo**

```
Sub Main
Dim nmTab as string
'Принимаем, без проверки, что окно Карты активно.
set coordsys Window frontwindow ()
'Определяем границы рабочей области для таблицы Угодья
nmTab="Угодья"
print "minX=" & str$(GetCoordLimit (nmTab, "minX"))
print "maxX=" & str$(GetCoordLimit (nmTab, "maxX"))
print "minY=" & str$(GetCoordLimit (nmTab, "minY"))
print "maxY=" & str$(GetCoordLimit (nmTab, "maxY"))
end sub

Function GetCoordLimit (byval tbName as string,
```

```

        byval CoordLimits as string) as Float
'Sистема координат MapInfo
dim xy as float
do case CoordLimits
    case "minX"
        xy=TableInfo(tbName,TAB_INFO_COORDSYS_MINX)
    case "maxX"
        xy=TableInfo(tbName,TAB_INFO_COORDSYS_MAXX)
    case "minY"
        xy=TableInfo(tbName,TAB_INFO_COORDSYS_MINY)
    case "maxY"
        xy=TableInfo(tbName,TAB_INFO_COORDSYS_MAXY)
    case Else
        xy=0.99999
end Case
GetCoordLimit=xy
end function

```

Приведенная далее процедура `FieldsInfoForTable` для заданной таблицы (`tbName`) определяет имена (`sColNames`) и типы (`sColTypes`) всех полей. Размерность массивов указывает число полей в таблице.

**Код: Определение структуры таблицы**

```

...
Sub FieldsInfoForTable(byval tbName as string,
    sColNames() as string,sColTypes() as string)
dim i,n,tp as smallint
dim sType,sCol as string
n=NumCols(tbName)
redim sColNames(n)
redim sColTypes(n)
For i=1 to n
    sCol="COL" & Str$(i)
    sColNames(i)=ColumnInfo(tbName,sCol,COL_INFO_NAME)
    tp=ColumnInfo(tbName,sCol,COL_INFO_TYPE)
    Do case tp
        case COL_TYPE_CHAR
            sType="Символьный"
        case COL_TYPE_DECIMAL
            sType="Десятичный"
        case COL_TYPE_FLOAT
            sType="Вещественный"
        case COL_TYPE_INTEGER
            sType="Целочисленный"
        case COL_TYPE_SMALLINT
            sType="Короткое целое"
        case COL_TYPE_DATE
            sType="Дата"
        case COL_TYPE_LOGICAL
            sType="Логический"
        case Else
            sType="Не определен"
    end Case
    sColTypes(i)=sType
Next
end sub

```

## Тема 28. Последовательный перебор записей в таблице

Процедура `forAllRowsColValue` позволяет получить значения из заданного поля для всей таблицы. Поле задается порядковым номером (`col`) в списке полей таблицы (`tbName`). Результат выводится в окно **Сообщения**.

### Код: Перебор записей в таблице

```
...
sub forAllRowsColValue (byval tbName as string,
    byval col as integer)
dim n,row as smallint
n=NumCols (tbName)
if col>n then
    Note "Номер колонки больше числа колонок в таблице!"
    exit sub
end if
'Просмотреть построчно всю таблицу.
row=1
Fetch First From tbName
Do While Not EOT (tbName)
    print GetTableRowColValue (tbName, row, col)
    row=row+1
    Fetch Next From tbName
Loop
end sub
```

'Используемая в цикле функция `GetTableRowColValue` возвращает значение поля `col` строки `row` таблицы `tbName` в виде строки.

```
Function GetTableRowColValue (byval tbName as string,
    byval row as integer,
    byval col as integer) as string
dim sCol as string
dim aa as alias
sCol=".COL" & Str$(col)
aa=tbName & sCol
GetTableRowColValue=aa
end function
```

## Тема 29. Организация выборки записей

В качестве примера приведем код небольшой процедуры выбирающей в таблице `tbName` все полилинии заданного цвета. Выбранные объекты записываются в массив объектов.

### Код: Выборка полилиний заданного цвета

```
...
Sub SelectPolyLineColor (byval tbName as string,
    byval iRed as smallint, byval iGreen as smallint,
    byval iBlue as smallint, SelObjects() as Object)
dim nRows, row as integer
select * from tbName where Str$(obj)="Polyline" AND
    StyleAttr (ObjectInfo (Obj, 2), 4)=RGB (iRed, iGreen, iBlue)
into SelObj
nRows=SelectionInfo (SEL_INFO_NROWS)
redim SelObjects (nRows)
if nRows>0 then
    row=1
    Fetch First From SelObj
    Do While Not EOT (SelObj)
        SelObjects (row)=SelObj.obj
        row=row+1
    
```

```

Fetch Next From SelObj
Loop
end if
end sub

```

Цвет задается в координатах RGB модели цвета. Каждая цветовая координата это целое число в диапазоне от 0 до 256. Так красный цвет описывается следующим набором координат (255,0,0), черный – (0,0,0) и т.д.

В установках стиля объекта цвет может описываться в виде целочисленного кода. Этот код вычисляется следующим образом: (Red\*65536)+(Green\*256)+Blue. Код можно очень просто определить, если в окне «MapBasic» выполнить Print RGB(255,0,0). В окне «Сообщения» получим 16711680 – код красного цвета.

### Тема 30. Поиск

Для того чтобы наглядно показать механизм действия оператора Find сформируем две таблицы:

1. Таблица **НасПункты**. Каждая запись это площадной объект – населенный пункт (деревня). Имеется поле *Название* в котором записаны наименования населенных пунктов.
2. Таблица **Дома**. Каждая запись это площадной объект – дом. Имеется текстовое поле *Номер* в котором хранится номер дома. Поле Номер проиндексировано. Все дома накладываются на объекты таблицы **НасПункты** и нумеруются, начиная с единицы в каждом населенном пункте. То есть значения в поле *Номер* не уникальные. Информации о населенных пунктах в таблице **Дома** нет.

Приведем код процедуры, которая по номеру дома и имени населенного пункта найдет объект на карте.

#### Код: Поиск дома в населенном пункте

```

...
sub FindHouse(byval NumHouse as string,byval village as string,
byval MainTab as string,byval MainTabCol as string,
byval AddTab as string, byval AddTabCol as string)
dim x,y as float
dim id as integer
'Подготовка поиска
Find Using MainTab(MainTabCol)
Refine Using AddTab(AddTabCol)
Options Abbrs Off
ClosestAddr On
Find NumHouse,village
'Определить результат поиска
If CommandInfo(CMD_INFO_FIND_RC)>=1 Then
x=CommandInfo(CMD_INFO_X)
y=CommandInfo(CMD_INFO_Y)
id=CommandInfo(CMD_INFO_FIND_ROWID)
'Выбрать найденный объект
select * from MainTab where rowid=id
'Центрировать экран по найденному объекту, ширина картинка 100 метров.
Set Map Window idmap
Center (x,y) Zoom 100 Units "m"
Else
Note "Дом не найден."
End If
end sub

```

Так как в процедуре фигурируют координаты, то до вызова процедуры должна быть установлена система координат. Вызов процедуры может выглядеть следующим образом:  
`call FindHouse("3", "Дубки", "Дома", "Номер", "НасПункты", "Название").`

Следует обратить внимание на то, что у оператора `Find` нет видимых результатов работы таких как выделение объекта и т.п. Для получения результирующей информации нужно использовать функцию `CommandInfo` со следующими аргументами:

<code>CMD_INFO_FIND_RC</code>	Возвращает код соответствующий результату работы оператора. Список кодов можно найти в Справочнике MapBasic.
<code>CMD_INFO_X</code>	Возвращает координату X найденного объекта <sup>21</sup> .
<code>CMD_INFO_Y</code>	Возвращает координату Y найденного объекта.
<code>CMD_INFO_FIND_ROWID</code>	Возвращает номер строки, которой соответствует найденный объект.

Конечно, рассмотренный пример можно было бы решить и с помощью организации запросов с использованием оператора `Select`. Однако в некоторых случаях, особенно когда данные описываются в терминах улицы, сторона улицы, пересечение улиц, номер дома и т.д. использование `Find` предпочтительно.

**Код: Пример использования SearchPoint в связке с SearchInfo**

```

...
Sub Main
'Добавим инструментальную кнопку на панель Пенал
Alter ButtonPad ID 2
  Add Separator
  Add ToolButton
    Icon MI_ICON_EMERGENCY_17
    ID 201
    Cursor MI_CURSOR_CROSSHAIR
    DrawMode DM_CUSTOM_POINT
    Calling SearchPointDemo
    HelpMsg "Демонстрация оператора SearchPoint\nSearchPoint"
    Enable

  Show
end sub

sub SearchPointDemo
dim n,idMap,i,j as integer
Dim x,y As Float
dim ss as string
idMap=FrontWindow()
If WindowInfo(idMap,WIN_INFO_TYPE) <> WIN_MAPPER Then
  Note "Инструмент используется только в окне Карты."
  Exit Sub
End If
set CoordSys Window idMap
x=CommandInfo(CMD_INFO_X)
y=CommandInfo(CMD_INFO_Y)
n=SearchPoint(idMap,x,y)
print "--- Новый поиск ---"
print "X= " & str$(x)
print "Y= " & str$(y)
print "Число объектов= " & str$(n)
for i=1 to n
  ss=SearchInfo(i,SEARCH_INFO_TABLE)
  j=SearchInfo(i,SEARCH_INFO_ROW)
  print "Объект " & str$(i)

```

<sup>21</sup> В системе координат MapInfo.

```

print " Таблица: " & ss
print " Номер записи: " & str$(j)
next
end sub

```

В обработчике `SearchPointDemo` определяется позиция курсора и далее с помощью оператора `SearchPoint` информация обо всех объектах в этой точке. Извлекаем информацию с помощью оператора `SearchInfo`. Всю полученную информацию выводим в окно **Сообщения**. На рис. 20 показано как выглядит эта информация.

Естественно, что имея имя таблицы и номер записи можно без особых сложностей получить значение любого поля в этой записи, в том числе и сам графический объект. Аналогично работает и другой оператор поиска `SearchRect`, только область поиска в этом случае будет прямоугольник.

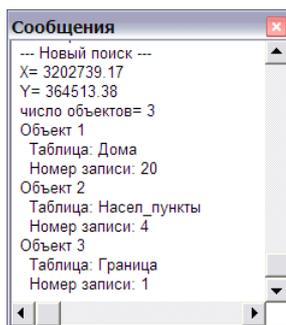


Рис. 20

### Тема 31. Геокодирование

*Геокодирование* – это процесс привязки ваших данных, к объектам на карте.

Привязка заключается в назначении каждой записи с информацией точечного объекта, который можно отобразить на карте `MapInfo`. Таким образом, данные получают географическую привязку, и работа с ними становится более эффективной.

Вначале на простом примере покажем, как выполняется геокодирование в `MapInfo`, а затем приведем код, выполняющий аналогичные действия.

Определимся с постановкой задачи.

Кодируемая таблица **LotInfo.tab**.

В этой таблице приведена некая информация по участкам. В данном случае эта информация описывается одним полем *Info*, но на самом деле таких полей может быть много и информация может быть весьма объемной.

Приведем описание таблицы в файле *TAB*:

```

!table
!version 300
!charset WindowsCyrillic

```

```

Definition Table
Type NATIVE Charset "WindowsCyrillic"
Fields 3
  ID Integer ;
  CN Char (40) ;
  Info Char (50) ;

```

К записям таблицы можно присоединять графические объекты, но таковых пока нет, то есть информацию можно лишь просматривать в окне списка. Вид таблицы в окне списка показан на рис. 21.

ID	CN	Info
1	99-99-0020314-1	Информация 1
2	99-99-0020314-2	Информация 2
3	99-99-0020314-3	Информация 3
4	99-99-0020314-4	Информация 4
5	99-99-0020314-5	Информация 5
6	99-99-0020314-6	Информация 6

Рис. 21

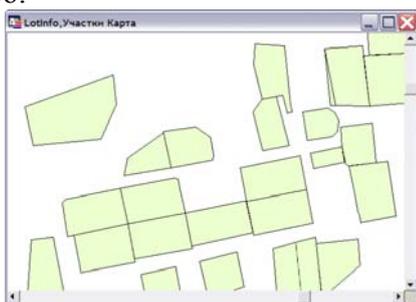
Таблица поиска **Участки.tab**.

Таблица с участками представленными на карте в виде областей (Рис. 22а). Структура файла *TAB* имеет вид:

```
!table
!version 300
!charset WindowsCyrillic
```

```
Definition Table
Type NATIVE Charset "WindowsCyrillic"
Fields 2
  ID Integer ;
  CN Char (40) Index 1 ;
```

Обратите внимание на то, что таблица имеет проиндексированное поле – это обязательное условие для выполнения геокодирования. Окно списка для таблицы **Участки** показано на рис. 22б.



а

ID	CN
0	99-99-0020314-1
0	99-99-0020314-2
0	99-99-0020314-3
0	99-99-0020314-4
0	99-99-0020314-5
0	99-99-0020314-6

б

Рис. 22

Записи в кодируемой таблице и таблице поиска будут сопоставляться по кадастровым номерам (поле *CN*). Кадастровый номер является уникальным для участка, и здесь мы имеем, по сути, однозначное соответствие между участком на карте и записью в таблице. Нередко геокодирование выполняется по адресным индексам или просто адресам и в этом случае некоторые записи могут остаться не геокодированными, и их обработку нужно будет выполнять в ручном режиме.

Теперь, когда мы определились с условиями задачи, приступим к ее решению. Выбираем меню **Таблица/Геокодирование...** В результате будет открыта форма ввода данных. Заполненная форма для нашего примера показана на рис. 23.

Геокодирование

Геокодировать таблицу: LotInfo

информация в колонке: CN

в границах: нет

Искать в таблице: Участки

объекты из колонки: CN

Режим:
   
 Автоматический
   
 Вручную

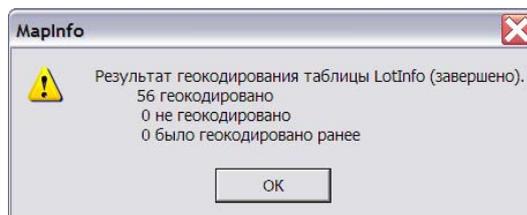
Символ: ●

Уточнить поиск:
   
Искать в другой таблице: нет
   
в границах, заданных в колонке: нет

OK Отмена Варианты... Справка

Рис. 23

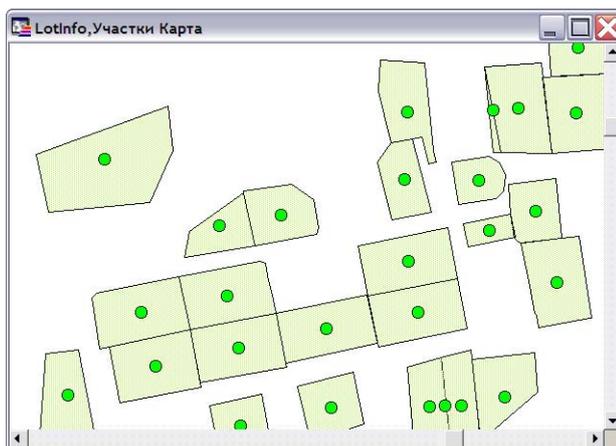
После нажатия кнопки *OK* стартует процесс геокодирования таблицы **LotInfo** в соответствии с таблицей поиска **Участки**. По окончании процесса будет выдано сообщение о результатах геокодирования (Рис. 24).



**Рис. 24**

Теперь окно карты приобретет вид показанный на рис. 25. Здесь зеленые кружки это точечные объекты из таблицы **LotInfo**. Координаты этих объектов соответствуют координатам центроидов соответствующих областей.

В данном случае вся таблица геокодирована полностью. Однако так бывает не всегда, и отдельные записи могут остаться не геокодированными, а разбираться с ними придется в ручном режиме. Выборку таких записей можно сделать с помощью запроса `select * from LotInfo where not Object into Selection` или `select * from LotInfo where Str$(obj)="" into Selection`.



**Рис. 25**

Перейдем к программной реализации приведенного процесса. Так как аналога этому процессу в виде отдельного оператора или функции в MapBasic нет, то построим код на использовании оператора `Find` и функции `CreatePoint`.

### Код: Геокодирование

```

Include "mapbasic.def"
Declare Sub Main
Declare Sub GeoCoding(tbCoding As String)

Sub Main
dim mapObj, infoObj as string
dim aa As Alias
'предполагается что таблицы уже загружены
mapObj = "Участки"
infoObj = "LotInfo"
aa="CN"
Set CoordSys table mapObj
Find Using mapObj(aa)
Call GeoCoding(infoObj)
End Sub

Sub GeoCoding(tbCoding As String)

```

```

Dim aRow, aCN As Alias
Dim i, iRtn As Integer
Dim x, y As Float
aRow=tbCoding + ".rowid"
aCN = tbCoding + ".CN"
Fetch First from tbCoding
Do While NOT EOT(tbCoding)
    i=aRow
    Find aCN
    iRtn=CommandInfo(CMD_INFO_FIND_RC)
    If iRtn < 0 Then
        Print "Строка " & i & ": " &
            aCN + " в карте не найден"
    Else
        x = CommandInfo(CMD_INFO_X)
        y = CommandInfo(CMD_INFO_Y)
        update tbCoding
            Set obj = CreatePoint(x,y)
            Where rowid = i
    End If
    Fetch Next from tbCoding
Loop
Commit table tbCoding
End Sub

```

Перед вызовом процедуры GeoCoding выполняются подготовительные действия:

- ✓ Определяются участвующие в процессе таблицы.
- ✓ Определяется поле в таблице поиска по которому и будет выполняться поиск.
- ✓ Так как предполагается работа с координатами устанавливается система координат.
- ✓ Выполняется настройка поиска. Указывается таблица и поле, используемые в поиске.

Сама процедура кодирования заключается в построчном просмотре кодируемой таблицы tbCoding с поиском соответствующего объекта в таблице mapObj. Поиск выполняется по кадастровому номеру (tbCoding.CN) при удачном поиске определяются координаты центра найденного объекта и текущей строке таблицы tbCoding присписывается точечный объект, построенный по найденным координатам. Стиль объекта определяется текущей настройкой стиля точечных объектов. После выхода из цикла таблица tbCoding сохраняется.

### Тема 32. Редактирование таблиц

Для изменения записей таблицы используют следующие операторы:

Insert	Вставляет новую строку в открытую таблицу. Допустима вставка группы строк из другой таблицы.
Update	Изменяет одну или более строк в таблице.
Delete	Удаляет одну или более записей из открытой таблицы.

Далее рассмотрим вопросы использования этих операторов на коротких примерах кода.

В таблицу **Угодья** будет вставлена новая строка. В поля *Номер*, *Тип*, *Балл* которой будут записаны значения "rs423" из переменной number, "4" и целое число 25, все остальные поля примут значения по умолчанию. Этой записи не будет соответствовать графический объект.

```

number="rs423"
Insert Into Угодья (Номер, Тип, Балл)
    Values (number, "4", 25)

```

В следующем примере формируется новая запись, включая графический объект (треугольник). Измененная таблица сразу сохраняется на диске.

```
dim newLand as object
Create Region Into Variable newLand 1
    3
    (3249980, 379020)
    (3249995, 379005)
    (3249978, 379001)
Insert Into Угодья (Obj, Тип)
    Values (newLand, "4")
Commit Table Угодья
```

Не нужно забывать, что если программа работает с координатами, то до начала этой работы должна быть установлена система координат программы, например таким образом  
Set CoordSys Window idMap.

Оператор вставки можно также использовать для вставки группы записей, при этом имеются свои особенности. Так предложение `Select`, в данном случае, не может иметь расширения типа `Where`, `Into`, `Group By` и т.д. Результатом действия оператора `Insert Into Угодья (Obj) Select obj From tst02` будет вставка всех графических объектов из таблицы **tst02** в таблицу **Угодья**. То есть в таблицу **Угодья** будет добавлено столько записей, сколько было объектов в таблице **tst02**. Все поля, кроме *Obj*, будут заполнены значениями по умолчанию.

Оператор `Insert Into Угодья Select * From tst02` может привести к ошибке, если структуры таблиц (число полей, их тип) различаются. В тоже время предыдущий оператор и в этом случае даст корректный результат.

Случаи изменения значений в полях таблиц встречаются довольно часто. При этом различают следующие варианты:

1. Изменяется значение поля во всех строках таблицы.  
Update Угодья Set Name="r" & str\$(rowid), Тип=2
2. Изменяется значение поля в одной конкретной строке таблицы.  
Update Угодья Set Балл=25 Where RowID=j

Изменять можно как поля семантики, так и объектное поле *Obj*.

```
select * from Угодья where Str$(obj)="" into SelRows
if SelectionInfo(SEL_INFO_NROWS)>0 then
    Update SelRows Set obj=newLand Where RowID=1
    Commit Table Угодья
end if
```

В данном случае организована выборка из таблицы **Угодья** всех записей, не имеющих графических объектов. Далее для первой записи в выборке полю *Obj* присваиваем объектную переменную `newLand` (предполагается, что переменная не пустая). Не нужно забывать что, редактируя объект в выборке, мы тем самым редактируем этот объект в базовой таблице. Поэтому сохраняем измененную таблицу **Угодья**.

Удалить все записи в таблице *Угодья* можно оператором `Delete From Угодья`. При этом структура таблицы не меняется, таблица остается в карте, если до этого она была там, и в принципе ее можно продолжать использовать: добавлять записи, редактировать и т.д. Однако оператор `Delete` лишь отмечает записи как удаленные, но физически их не удаляет. Такое двойственное состояние при дальнейшей работе может привести к

ошибкам. Поэтому рекомендуется сразу выполнить сохранение таблицы и ее упаковку. В процессе упаковки записи, отмеченные как удаленные, будут удалены окончательно<sup>22</sup>.

Удалить одну конкретную запись, зная ее позицию в таблице, можно с помощью оператора следующего вида Delete From Угодья Where Rowid=2. Удалить только графический объект, оставляя всю семантику без изменений, можно используя оператор вида Delete Object From Угодья Where Rowid=3.

Далее покажем, как можно удалить часть записей в таблице

```
select * from Угодья where Тип="3" into SelRows
if SelectionInfo(SEL_INFO_NROWS)>0 then
    Delete From SelRows
    Commit Table Угодья
end if
```

### Тема 33. Работа с метаданными

Обычно метаданные определяют как данные о данных и используют для повышения качества поиска. В MapInfo под метаданными понимают некоторую вспомогательную информацию, записываемую в *ТАВ-файл* таблицы. Такой информацией могут быть сведения о таблице прямого доступа, информация о связанной таблице, информация о файле поверхности и т.д. Ниже приводится код *ТАВ-файла* поверхности с метаданными.

```
!table
!version 500
!charset WindowsCyrillic

Definition Table
File "лес2_хвойные.mig"
Type "RASTER"
(3205480.6725187418,360737.82118453976) (0,0) Label "Точка 1",
(3207736.4693940589,360737.82118453976) (135,0) Label "Точка 2",
(3207736.4693940589,359050.15092967299) (135,101) Label "Точка 3",
(3205480.6725187418,359050.15092967299) (0,101) Label "Точка 4"
CoordSys NonEarth Units "m"
Units "m"
RasterStyle 6 1
begin_metadata
"\Relief Shading" = ""
"\Relief Shading\Horizontal Angle" = "135"
"\Relief Shading\Vertical Angle" = "45"
"\Relief Shading\Vertical Scale Factor" = "50"
"\IsReadOnly" = "FALSE"
end_metadata
```

Такого рода метаданные формирует система MapInfo. Разработчики пользовательских программ также могут проектировать использование метаданных. Приведем примеры типов информации, которые могли бы использоваться в качестве метаданных.

- ✓ Информация о пространственно-временных характеристиках.
- ✓ Информация о координатной основе.
- ✓ Информация об объектовом составе.
- ✓ Информация о классификаторах.
- ✓ Информация об ограничениях.
- ✓ Идентификационная информация.

Приведем пример использования процедур записи и чтения метаданных.

<sup>22</sup> После упаковки таблица будет исключена из карты, если до этого была там, поэтому потребуется восстановить ее в карте (и восстановить глобальные стили оформления если необходимо).

### Код: Работа с метаданными

```

Declare Sub Main
Declare Sub SetMetadata(byval tbName as string)
Declare Sub RemoveMetadata(byval tbName as string)
Declare Sub PrintMetadata1(byval tbName as string)
Declare Sub PrintMetadata2(byval tbName as string)

Sub Main
dim tbName as string
tbName="Леса"
print chr$(12)
call SetMetadata(tbName)
call RemoveMetadata(tbName)
call PrintMetadata1(tbName)
call PrintMetadata2(tbName)
end sub

Sub SetMetadata(byval tbName as string)
'Сохраняется значение ключа23:
'если ключ уже существует, то его значение будет изменено;
'если ключ не существует, то ключ и его значение будут
'добавлены к метаданным таблицы.
Metadata Table tbName SetKey "\MY\Project" To "Петровский"
Metadata Table tbName SetKey "\MY\Автор" To "Иванов В.А."
Metadata Table tbName SetKey "\MY\Дата" To "20100809"
Metadata Table tbName SetKey "\ID" To "356412"
Metadata Table tbName SetKey "\BD\Name" To "dTown"
Metadata Table tbName SetKey "\BD\Tab" To "Population"
end sub

Sub RemoveMetadata(byval tbName as string)
'Удаляется значение ключа \MY\Дата
Metadata Table tbName DropKey "\MY\Дата"
'Удаляется вся иерархическая структура ключа \BD
Metadata Table tbName DropKey "\BD" Hierarchical
end sub

Sub PrintMetadata1(byval tbName as string)
'Вывод значения конкретного ключа
Print GetMetadata$(tbName,"\MY\Project")
Print GetMetadata$(tbName,"\MY\Автор")
Print GetMetadata$(tbName,"\MY\Дата")
Print GetMetadata$(tbName,"\ID")
Print "-----"
end sub

Sub PrintMetadata2(byval tbName as string)
'Вывод ключа и значения ключа, для случая когда структура ключей неизвестна
'Выводится вся иерархия ключей начиная с нулевого уровня ("")
'Глубина просматриваемой иерархии не более 10 уровней
dim tID as integer
dim tName,tValue as string
Metadata Table tbName SetTraverse "" Hierarchical Into ID tID
Metadata Traverse tID Next Into Key tName Into Value tValue
Do While len(tName)>0
    Print tName & " | " & tValue
    Metadata Traverse tID
    Next Into Key tName Into Value tValue
Loop

```

<sup>23</sup> Длина значения до 240 символов.

```
MetaData Traverse tID Destroy  
end sub
```

Результат работы программы показан на рис. 26. Файл *TAB* таблицы **Леса** примет вид:

```
!table  
!version 300  
!charset WindowsCyrillic  
  
Definition Table  
  Type NATIVE Charset "WindowsCyrillic"  
  Fields 3  
    id Char (10) ;  
    S Decimal (14, 4) ;  
    AreaHectare Decimal (14, 4) ;  
ReadOnly  
begin_metadata  
"\MY_" = ""  
"\MY\Project" = "Петровский"  
"\MY\Автор" = "Иванов В.А."  
"\MY\Дата" = ""  
"\IsReadOnly" = "FALSE"  
"\ID" = "356412"  
end_metadata
```

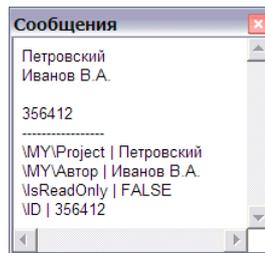


Рис. 26

## Глава 4. Работа с картой

Карту делают слои, а слои состоят из объектов. Работа с картой это работа со слоями и объектами.

### Тема 34. Слои

Если к таблице разрешено присоединять графические объекты, то эта таблица может быть представлена на карте в виде отдельного слоя. Слои обычно состоят из однотипных объектов связанных какой-то единой смысловой нагрузкой. Например, слой **Леса** представляет собой все леса на карте и тип всех объектов – *Region*. Хотя все конечно зависит от назначения системы. Так для ГИС лесного хозяйства одного слоя будет явно мало и придется формировать набор слоев связанных с лесами.

Для добавления слоя (слоев) к карте используют оператор `Add Map`, например `Add Map Window idWin Layer Угодья`. В результате выполнения этого оператора к карте *idWin* будет добавлен слой **Угодья**. Он станет самым верхним некосметическим слоем в окне карты *idWin*. Если указание на окно опущено, то слой будет добавлен в самое верхнее открытое окно карты. Нередко оператор используют в следующем виде `Add Map Auto Layer Угодья`. Ключевое слово *Auto* указывает на то, что будет выполняться ранжирование слоев по принципу: слои с точечными объектами помещаются наверх списка слоев, ниже размещаются слои с площадными объектами и в самом низу растровые слои.

Задача удаления слоя (слоев) из карты решается с помощью оператора `Remove Map`. Так для удаления слоя **Угодья** из карты *idWin* можно использовать оператор `Remove Map Window idWin Layer Угодья`. Отсутствие слоя в карте приведет к ошибке времени выполнения, а значит желательно, предварительно выполнить проверку есть ли слой в карте.

Приведем возможные формы записи оператора `Remove Map`.

```
Remove Map Layer Угодья
Remove Map Layer "Угодья"
```

```
nmLayer="Угодья"
Remove Map Layer nmLayer
```

```
nmLayer="Угодья(1)"
Remove Map Layer nmLayer
```

```
Remove Map Layer 1
```

Имя слоя записано в переменную `nmLayer`.

В переменную записан указатель на первый тематический слой, построенный на основе слоя **Угодья**. Следующий оператор удаляет этот тематический слой.

Удаляется самый верхний некосметический слой. Идентификация выполняется по номеру слоя в списке слоев карты.

Оператор `Remove Map` удаляет слой из окна карты. Соответствующая слою, таблица остается открытой и при желании может быть снова добавлена к карте. Для закрытия таблицы используется оператор `Close Table Угодья`. Для полного удаления таблицы (удаления с диска) используется оператор вида `Drop Table Угодья`. Отменить действие этого оператора невозможно.

### Тема 35. Тематические слои

В толковом словаре русского языка (Ожегов С.И.) прилагательное *тематический* определяется как «посвященный какой-нибудь одной теме». Таким образом, термин *тематический слой* можно трактовать как слой, посвященный одной теме, отражающий какое-то одно явление природное, социально-экономическое или какое-либо другое. Рассмотрим на простом примере построение кода для формирования тематического слоя. В нашем примере имеется базовая таблица **Лес** представляющая собой некоторое

множество участков леса. Для каждой записи таблицы (участка) определены следующие поля:

- ✓ Номер. Номер участка.
- ✓ Хвойные. Количество деревьев хвойных пород на участке в процентах.
- ✓ Лиственные. Количество деревьев лиственных пород на участке в процентах.

### Создание слоя диапазонов

Вид оператора `Shade` для формирования тематического слоя показывающего количество деревьев хвойных пород на участке (в процентах).

```
Shade Window idWinMap Лес with Хвойные Ranges Apply Color
0: 22 Brush (2,15597520,16777215),
22: 50 Brush (2,11591840,16777215),
50: 64 Brush (2,7389296,16777215),
64: 82 Brush (2,4235312,16777215),
82: 88 Brush (2,32768,16777215)
Style replace off
```

Слой строится в окне карты *idWinMap* для базового слоя **Лес**, используются данные из поля *Хвойные*. Служебное слово `Ranges` определяет тип построения – слой диапазонов, `Apply Color` означает, что для выделения тематической информации используется только цвет. Далее идет описание используемых диапазонов. Определение `Style replace` управляет прозрачностью штриховок в тематических слоях (диапазоны, отдельные значения). Значение *ON* (по умолчанию) - слои под тематическим слоем не прорисовываются, соответственно *OFF* - прорисовываются. Код формирования легенды для нового тематического слоя.

```
Set legend window idWinMap layer prev display on shades on symbols off lines
off count on title "Хвойные породы" Font ("Arial Cyr",1,9,0) subtitle "в
процентах" Font ("Arial Cyr",0,8,0) ascending off style size small ranges
Font ("Arial Cyr",0,8,0) auto display off ,auto display on ,auto display on
,auto display on ,auto display on ,auto display on
```

На рис. 27 показан сформированный тематический слой с легендой.

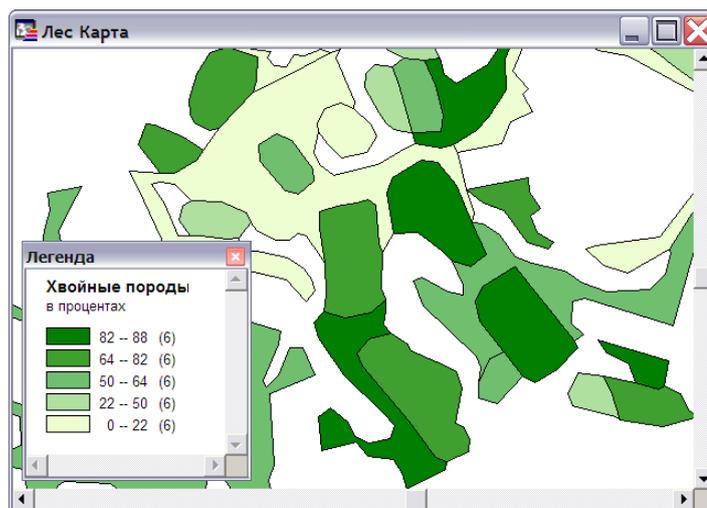


Рис. 27

### Создание слоя отдельных значений

Добавим к нашей карте слой **Контакты** на котором размещена информация о контактах с дикими животными. В таблице имеется поле *Животное*, в котором, в виде кода,

устанавливается животное, с которым имелась встреча и поле *Дата* указывающее дату этого события. Создадим, для слоя **Контакты** поле *Животное*, тематический слой отдельных значений.

```
Shade Window idWinMap Контакты with Животное Values 1 Symbol (61,16711680,12,"MapInfo Cartographic",0,0),2 Symbol (61,65280,12,"MapInfo Cartographic",0,0),3 Symbol (61,255,12,"MapInfo Cartographic",0,0),4 Symbol (61,16711935,12,"MapInfo Cartographic",0,0)
```

Здесь ключевое слово *values* указывает, что будет создан слой отдельных значений и далее следует описание этих отдельных значений. Результат действия оператора показан на рис. 28.

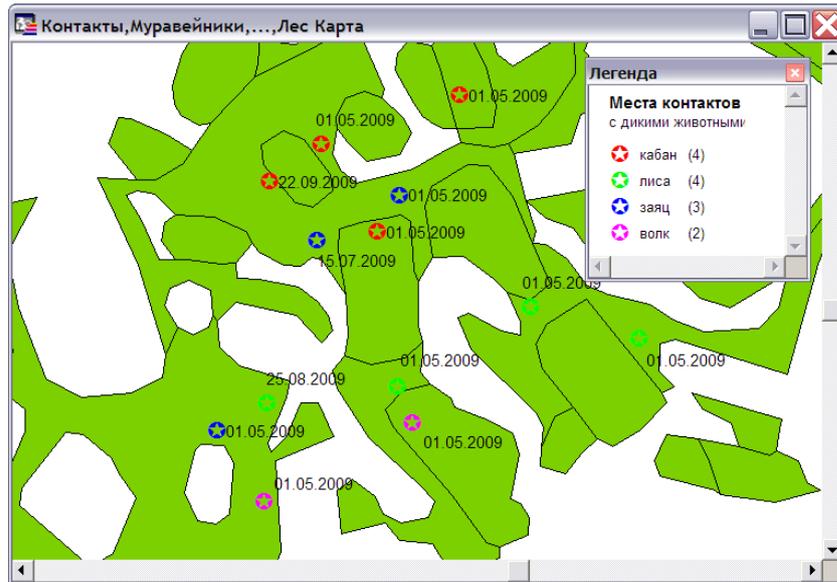


Рис. 28

Приведем еще один пример построения тематического слоя отдельных значений. Добавим к карте слой **Просеки** с полем *Ширина* (имеется в виду ширина просеки). Объектами этого слоя будут полилинии. Построим для поля *Ширина* тематический слой.

```
Shade Window idWinMap Просеки with Ширина Values apply color 5 Line (6,2,16711680),10 Line (6,2,65280),20 Line (6,2,255),25 Line (6,2,16711935),50 Line (6,2,16776960) style replace on
```

Результат показан на рис. 29.

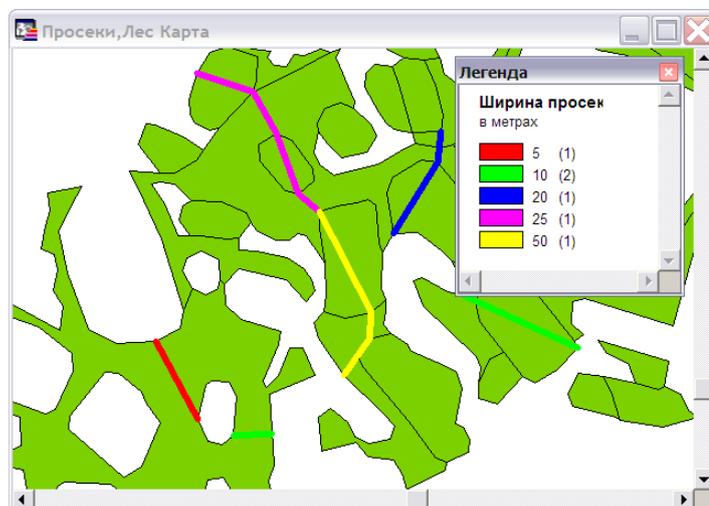


Рис. 29

### Создание слоя плотности точек

Сформируем тематический слой плотности точек для слоя **Лес** поле *Лиственные*.  
`Shade Window idWinMap Лес with Лиственные Density 1 square width 3 color 16711680`

На тип построения указывает ключевое слово `Density`. Одной точке соответствует одна единица из поля *Лиственные*, форма точки – квадрат, размер одной точки в пикселях 3, цвет – красный (Рис. 30).

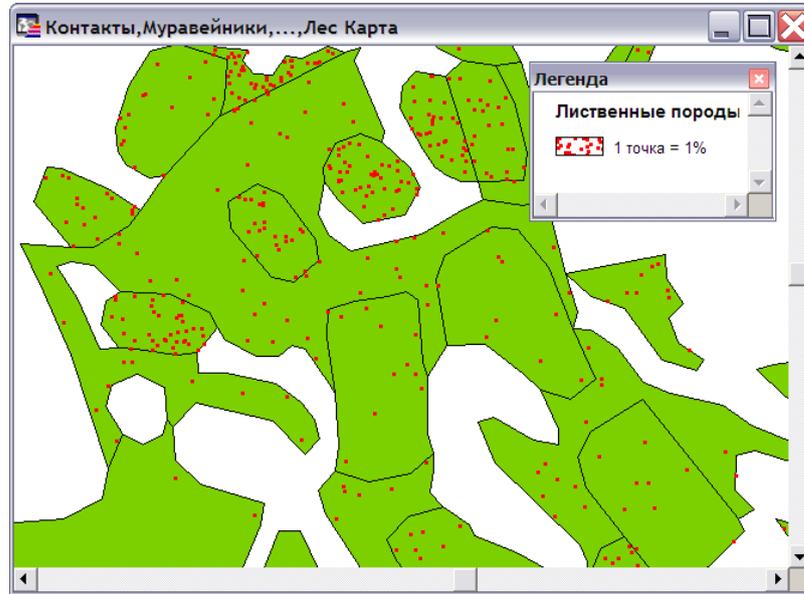


Рис. 30

### Создание слоя размерных символов

Для примера построения слоя размерных символов добавим к карте слой **Муравейники** представляющий точечные объекты. Поле *Диаметр* этой таблицы несет информацию о размерах муравейника (диаметр). Оператор `Shade` формирующий тематический слой имеет вид

`Shade Window idWinMap Муравейники with Диаметр Graduated 0.0:0 3:36 Symbol (34,16776960,36) vary size by "SQRT"`

Тип построения определяет ключевое слово `Graduated`. Далее идет описание диапазона размеров символов (мин\_диаметр:размер\_символа max\_диаметр:размер\_символа) и определение используемого символа. Часть оператора `vary size by "SQRT"` устанавливает, что размер символа определяется пропорционально квадратному корню из значения поля *Диаметр*<sup>24</sup>. Результат действия оператора показан на рис. 31.

<sup>24</sup> `vary size by "SQRT"` это значение по умолчанию и в данном случае его можно было и не включать в оператор. Другие допустимые значения: "LOG" – логарифмическая зависимость, "CONST" – линейная зависимость.

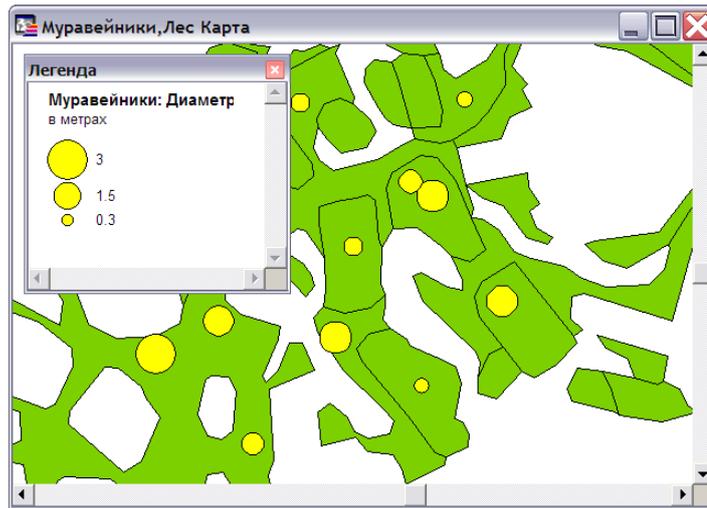


Рис. 31

### Создание слоя круговых диаграмм

Оператор формирования слоя круговых диаграмм имеет вид

```
Shade Window idWinMap Лес with Хвойные, Лиственные Pie Angle 180 Max Size 1
Units "cm" At Value 100 vary size by "SQRT" Border Pen (1,2,0) Position
center center style Brush (2,32768,16777215) ,Brush (2,65280,16777215)
```

Здесь ключевое слово `Pie` определяет тип построения, начальный угол отсчета параметров в диаграмме  $180^\circ$  ( $0^\circ$  соответствует направлению на восток), максимальный диаметр круга 1 см для максимального значения 100. `At Value` - является признаком диаграммы переменного размера. Назначение `vary size by "SQRT"` уже было описано выше. `Border Pen (1,2,0)` - определяет стиль граничных линий. `Position center center` - описывает положение диаграммы по отношению к центру объекта. В конце оператора идет описание стилей заливок диаграммы.

Результат действия оператора показан на рис. 32.

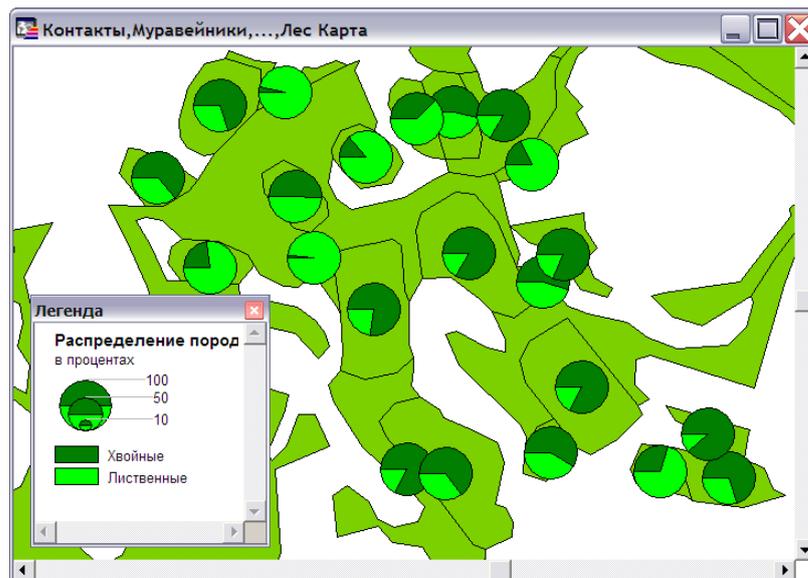


Рис. 32

### Создание слоя столбчатых графиков

Для иллюстрации формирования слоя столбчатых диаграмм приведем два оператора без комментариев.

```
Shade window idWinMap Лес with Хвойные,Лиственные Bar Max Size 1 Units "cm"
At Value 100 vary size by "CONST" border Pen (1,2,0) Width 1 Units "cm"
position center center style Brush (2,32768,16777215) ,Brush
(2,65280,16777215)
```

```
Shade Window idWinMap Лес with Хвойные,Лиственные Stacked bar Max Size 1
Units "cm" At Value 100 vary size by "CONST" border Pen (1,2,0) Width 1 Units
"cm" position center center style Brush (2,32768,16777215) ,Brush
(2,65280,16777215)
```

На рис. 33 показаны виды окна карты соответствующие этим операторам.

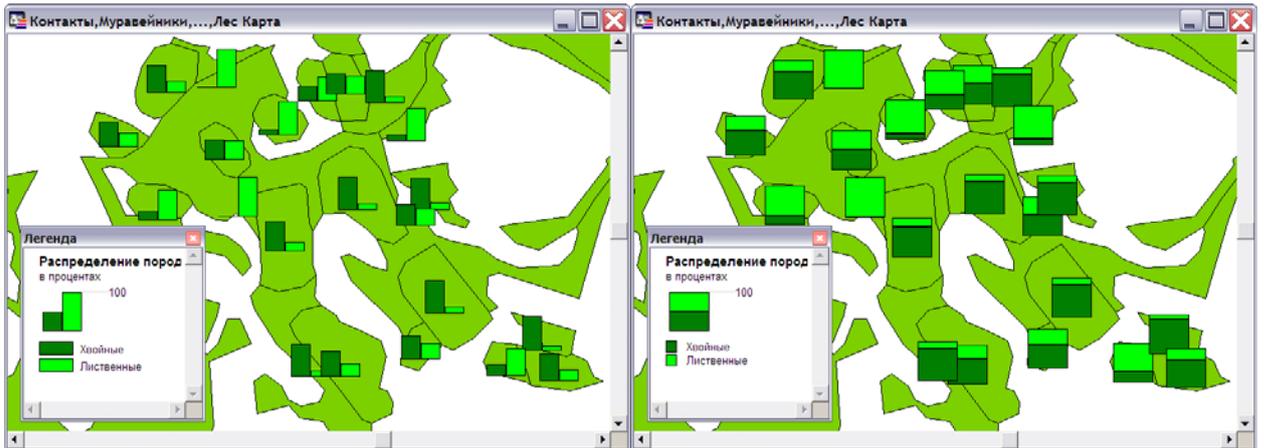


Рис. 33

### Создание тематической растровой поверхности

Для создания тематической растровой поверхности добавим к карте слой **Загрязнения** представляющий информацию по загрязнению почвы в отдельных точках. В качестве показателя загрязнения примем некоторый суммарный показатель загрязнения по шкале от 0 до 100 (поле Z). Также добавим слой **ГраницаПоверхности** хранящий область распространения поверхности. Оператор формирования тематической растровой поверхности будет иметь вид

```
Create Grid from Загрязнения with Z into "C:\tst\Загрязнения_Z.mig"
type "mig.ghl"
CoordSys NonEarth Units "m" Bounds (2000000, 0) (4000000, 2000000)
clipping table ГраницаПоверхности
inflect 5 by percent at
RGB(0, 0, 255) : 0
RGB(0, 255, 255) : 25
RGB(0, 255, 0) : 50
RGB(255, 255, 0) : 75
RGB(255, 0, 0) : 100
round 0.1
cell min 100
interpolate with "IDW" version "100"
using 5
"AGGREGATION METHOD": "0"
"BORDER": "0"
"CELL SIZE": "6.28"
"EXPONENT": "2"
"SEARCH RADIUS": "50"
```

Оператор `Create Grid` создаст таблицу тематической растровой поверхности. Эта таблица физически состоит из двух файлов с расширениями *TAB* и *MIG*. В *TAB*-файле находятся:

- ✓ Ссылка на файл растра (*MIG-файл*)
- ✓ Описание точек привязки растра (4 точки)
- ✓ Указание на систему координат и единицы измерений
- ✓ Настройки стиля отображения растра
- ✓ Блок метаданных связанных с построением поверхности

Далее таблицу нужно открыть и добавить к карте.

```
Open Table "C:\tst\Загрязнения_Z.TAB"
```

```
Add Map Layer Загрязнения_Z
```

Вид окна карты с построенной тематической поверхностью показан на рис. 34.

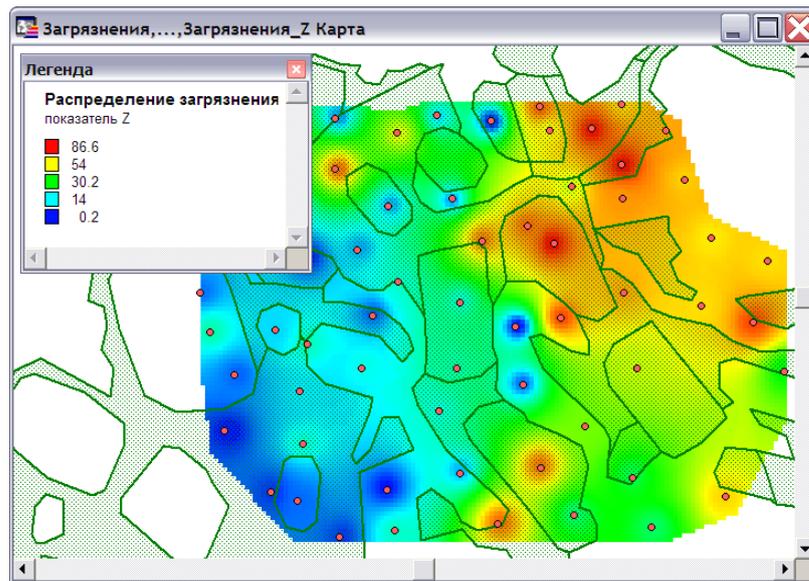


Рис. 34

Если к полученной поверхности применить команду MapInfo *Карта/Создать 3D карту...* то получим изображение поверхности в 3D виде показанном на рис. 35<sup>25</sup>.

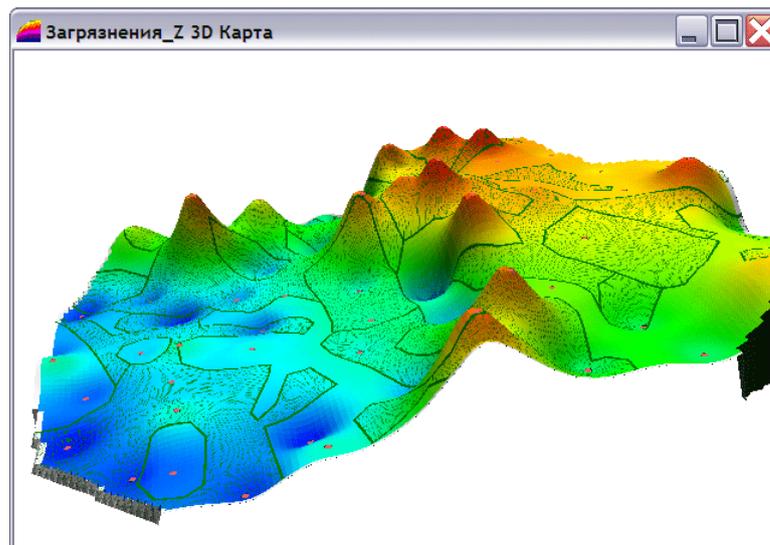


Рис. 35

<sup>25</sup> Этот процесс можно описать и в коде MapBasic. Протокол этой процедуры можно увидеть в окне MapBasic программы MapInfo после создания 3D вида.

### Тема 36. Подписи

Подписи, по сути, являются атрибутами объектов карты. Текст подписи извлекается из строки таблицы связанной с объектом. Вместо постоянной подписи, отражающей значения одного из полей записи, можно установить подпись являющуюся результатом вычисления выражения включающего несколько полей. На карте подпись привязывается к центру объекта. Все настройки подписей запоминаются в Рабочем наборе.

Не нужно путать подписи и текстовые объекты. Размер подписей не изменяется при изменении масштаба карты, что отличает их от текстовых объектов, размер которых изменяется с увеличением/уменьшением изображения в окне карты. Обычно текстовые объекты используются в виде заголовков и пояснений на карте или в отчете.

Управление подписями в MapBasic выполняется с помощью следующих операторов.

Set Map	Позволяет управлять множеством режимов представления данных в окне Карты. В частности позволяет установить, показать, скрыть, настроить подписи.
LayerInfo	Возвращает информацию о слое в окне Карты, в том числе весьма подробную информацию о настройках подписей.
LabelFindByID	Определяет наличие подписи в определенной строке заданного слоя. Возвращает логическое значение TRUE, если подпись существует или FALSE в противном случае.
LabelFindFirst	Опрашивает первую подпись на заданном слое. Возвращает логическое значение TRUE, если подпись существует или FALSE в противном случае.
LabelFindNext	Перемещает указатель к следующей подписи на заданном слое и опрашивает ее. Возвращает логическое значение TRUE, если подпись существует или FALSE в противном случае.
Labelinfo	Возвращает информацию о конкретной подписи определенной с помощью операторов LabelFind..
AutoLabel	Создает подписи в виде текстовых объектов на косметическом слое. Только для текущего вида в окне карты.

Приведем примеры использования оператора Set Map для управления подписями.

- ✓ Установить подписи: окно карты idMap, слой Лес, поле Хвойные, подписи могут накладываться друг на друга.  
Set Map Window idMap Layer Лес Label With Хвойные Auto On Overlap On
- ✓ Отключить показ подписей.  
Set Map Window idMap Layer Лес Label Auto Off
- ✓ Включить показ подписей.  
Set Map Window idMap Layer Лес Label Auto On
- ✓ Скрыть подписи.  
Set Map Window idMap Layer Лес Label Visibility Off
- ✓ Показать подписи.  
Set Map Window idMap Layer Лес Label Visibility On
- ✓ Настроить подписи: шрифт *Arial Cyr*, размер 10 пунктов, утолщенный, цвет зеленый с белой каймой, смещен относительно центра вверх на 15 пунктов (point)<sup>26</sup>.  
Set Map Window idMap Layer Лес Label Position Above Font ("Arial Cyr", 257, 10, 32768, 16777215) Offset 15
- ✓ Установить подписи в виде выражения включающего два поля (Хвойные и Лиственные). Подпись разместить в две строки.

<sup>26</sup> Один пункт (point) соответствует примерно 0.35 мм.

```
Set Map Window idMap Layer Лес Label With "Хвойные=" + Хвойные + " %"
+ Chr$(10) + "Лиственные=" + Лиственные + " %"
```

Поясним работу операторов `LabelFind...` на небольшом примере.

### Код: Конвертация подписей в текстовые объекты

```
dim idMap as integer
Sub Main
'Предполагается что окно карты активно
idMap=FrontWindow()
call SignatureToText ("Лес", "ТекстОбъекты")
end sub

sub SignatureToText(byval nmLayer as string,byval nmTxtLayer as string)
'Конвертирует подписи со слоя nmLayer в текстовые объекты на слое nmTxtLayer.
Dim isLabel As Logical
Dim iLayer as smallint
Dim textObj As Object
iLayer=FindNumForLayer(idMap,nmLayer)
isLabel=LabelFindFirst(idMap,iLayer,TRUE)
Do While isLabel
    textObj=LabelInfo(idMap,iLayer,LABEL_INFO_OBJECT)
    Alter Object textObj Info OBJ_INFO_TEXTFONT,
        MakeFont("Arial Cyr",0,12,BLACK,WHITE)
    insert Into nmTxtLayer (obj) Values (textObj)
    isLabel=LabelFindNext(idMap,iLayer)
Loop
end sub

function FindNumForLayer(idMapWindow as integer,
byval nmLayer as string) as smallint
'Вспомогательная функция: определяет номер слоя,
'в списке слоев окна карты, по имени слоя.
Dim i,n As SmallInt
dim nmLayer_i as string
n=MapperInfo(idMapWindow,MAPPER_INFO_LAYERS)
For i = 1 To n
    nmLayer_i=UCase$(LayerInfo(idMapWindow,i,LAYER_INFO_NAME))
    if nmLayer_i=UCase$(nmLayer) then
        FindNumForLayer=i
        exit function
    end if
next
FindNumForLayer=-1
end function
```

В данном примере подписи перебираются в цикле `Do While`, пока они не закончатся (пока оператор `LabelFindNext` не вернет *FALSE*). При каждом проходе тела цикла очередная подпись конвертируется в текстовый объект `textObj`. Далее для этого объекта выполняется настройка стиля оформления текста и он вставляется в таблицу `nmTxtLayer`.

### **Тема 37. Создание объектов**

`MapInfo` поддерживает следующие типы объектов: точка, прямая линия, полилиния, дуга, область, прямоугольник, скругленный прямоугольник, эллипс и рамка (прямоугольник в отчете с изображением из окна Карты, Списка и т.д.). Работа программ на `MapBasic` это, в основном, работа с объектами т.к. изображение карты это некоторая композиция элементарных объектов. Рассмотрим более внимательно, как строится эта работа. Для создания объектов используются операторы группы *Create*:

Create Arc	Создает новый объект дуга.
Create Frame	Создает новый объект рамка в окне Отчета.
Create Line	Создает новый объект линия.
Create PLine	Создает новый объект полилиния.
Create Point	Создает новый объект точка.
Create Rect	Создает новый объект прямоугольник.
Create Region	Создает новый объект область.
Create RoundRect	Создает новый объект прямоугольник со скругленными углами.
Create Text	Создает новый объект текст.
Create Ellipse	Создает новый объект эллипс (в том числе окружность).
Create Multipoint	Создает новый объект группа точек.
Create Collection	Создает новый объект коллекция. В коллекцию могут быть объединены точечные, линейные и площадные объекты.

Приведем примеры записи некоторых операторов:

- ✓ Создать дугу в прямоугольнике, построенном на точках (1) (2). Начало дуги 0°, конец 120° (0° - направление на восток; отсчет против часовой стрелки). Объект будет вставлен на редактируемый слой.

```
Create Arc Into Window idMap (3205630,360280) (3205730,360380) 0 120
Pen(1,4,0)
```

- ✓ Создать текстовый объект в прямоугольнике, построенном на точках (1) (2). Размер шрифта будут определять габариты прямоугольника, размер в определении Font будет проигнорирован. Объект будет вставлен на редактируемый слой.

```
Create Text "План" (3205730,360380) (3205931,360450)
Font("Helvetica",1,12,0)
```

- ✓ Создать группу из трех точек (1) (2) (3). Объект будет вставлен на редактируемый слой.

```
Create Multipoint Into Window idMap 3 (3205630,360280)
(3205810,360280) (3205980,360260) Symbol(74,0,12,"Wingdings",0,0)
```

Если нужно создать область с заранее неизвестным числом узлов, то обычно поступают следующим образом<sup>27</sup>. Создаем пустую область в переменной tObj (тип *Object*) Create Region Into Variable tObj 0 и далее изменяем форму объекта добавляя узлы к полигону.

```
for j=1 to k
    Alter Object tObj Node Add (x(j),y(j))
Next
```

Окончательно сформированный объект вставляем в таблицу tstObj.

```
Insert Into tstObj (Obj) Values (tObj)
```

В данном примере принято, что координаты узлов были, каким-то образом, получены заранее и хранятся в массивах x и y размерности k. Аналогичный подход используется и при создании полилиний.

Тот же прием используется при формировании группы точек в процедуре CreateMPoints.

### Код: Сформировать объект Группа точек (1)

```
Sub CreateMPoints(byval tbName as string,x() as float,y() as float)
Dim k,j as integer
Dim grPnt as object
set CoordSys Table tbName
k=Ubound(x)
Create Multipoint Into Variable grPnt 0 Symbol(32,RED,12)
for j=1 to k
```

<sup>27</sup> Для наглядности считаем, что область состоит из одного полигона. При наличии нескольких полигонов принцип решения не изменится.

```

        Alter Object grPnt Node Add (x(j),y(j))
next
Insert Into tbName (obj) Values (grPnt)
End sub

```

Другой прием при формировании группы точек показан в процедуре `CreateMPointsW`. Здесь задача сводится к построению оператора `Create Multipoint` в виде строки. Затем, используя `Run Command`, получаем группу точек.

**Код: Сформировать объект Группа точек (2)**

```

Sub CreateMPointsW(byval idMap as integer,byval tbName as string,x() as
float,y() as float)
Dim ss,sxy as string
Dim k,j as integer
set CoordSys Table tbName
set map window idMap Layer tbName Editable On
k=Ubound(x)
ss="Create Multipoint Into Window " & str$(idMap) & " " & str$(k)
sxy=""
for j=1 to k
    sxy=sxy & " (" & str$(x(j)) & "," & str$(y(j)) & ")"
next
sxy=sxy & " Symbol(32," & str$(RED) & ",12)"
ss=ss & sxy
Run Command ss
End sub

```

Нужно помнить, что если программа производит операции с координатами, то до этого момента необходимо определить систему координат в программе<sup>28</sup> с помощью оператора `Set CoordSys`. Нередко это упускается из виду и приводит к незапланированному поведению объектов.

Кроме операторов для создания объектов используются функции возвращающие объект.

CreateCircle	Возвращает графический объект - окружность заданного радиуса.
CreateLine	Возвращает графический объект - линия.
CreatePoint	Возвращает графический объект - точка.
CreateText	Возвращает графический объект - текст.
Buffer	Возвращает графический объект - область. Этот объект представляет собой сформированную буферную зону вокруг одного объекта. Метод расчета зависит от текущей координатной системы.
ConvexHull	Возвращает графический объект - область. Этот объект представляет собой выпуклую оболочку для объекта аргумента.

Функции используют текущие настройки стилей соответствующих объектов. Приведем пример.

```

Sub testFunction
dim oPoint,oCircle as object
Set CoordSys table Лес
Set Distance Units "m"
oPoint=CreatePoint(3205630,360280)
Insert Into tstObj (Obj) Values(Buffer(oPoint,10,200,"m"))
Insert Into tstObj (Obj) Values(CreateCircle(3205630,360280,150))
Insert Into tstObj (Obj) Values(oPoint)

```

<sup>28</sup> Система координат, установленная в окне карты `MapInfo`, не имеет отношения к координатам в вашей программе.

end sub

Особо следует остановиться на операторе `Create Object As`. Он позволяет решать более сложные задачи, такие как объединение, слияние объектов, построение буферных зон для группы объектов. Новые объекты типа *область* строятся на базе уже существующих объектов с использованием следующих операций<sup>29</sup>:

Buffer	Формирует буферную зону для объектов таблицы.
Union	Формирует объект типа область, определяемый как область объединения объектов.
Merge	Формирует объект типа область, определяемый как область слияния объектов.
ConvexHull	Строит выпуклую оболочку (объект типа область) для объектов таблицы.
Voronoi	Строит объекты типа область, представляющие собой полигоны Вороного, для исходных точек.

Данный оператор имеет ряд особенностей при использовании.

Рассмотрим, какие есть варианты при формировании буферной зоны из объектов таблицы. Во первых из базовой таблицы всегда можно выделить интересующую часть.

```
Set CoordSys table Лес
Set Distance Units "m"
Select * From Загрязнения Where Z>50 Into tmpTab
```

Здесь из таблицы **Загрязнения** выбраны все записи, имеющие в поле *Z* значения больше 50, и помещены во временную таблицу **tmpTab**.

Если далее используем оператор `Create Object As Buffer From tmpTab Into Table tstObj Width 55 Resolution 15`, то получим буферную зону в виде одного объекта типа *область*, возможно состоящего из нескольких полигонов (Рис. 36а).

Если использовать оператор `Create Object As Buffer From tmpTab Into Table tstObj Width 55 Resolution 15 Group By RowID`, получим несколько буферных зон, каждая из которых представляет объект типа область. Число таких объектов равно числу записей в таблице **tmpTab**.

И наконец, если использовать оператор `Create Object As Buffer From tmpTab Into Table tstObj Width 55 Resolution 15 Group By Z`, то все объекты из таблицы **tmpTab** имеющие в поле *Z* одинаковые значения образуют одну буферную зону, возможно в виде нескольких полигонов, а каждое уникальное значение - свою буферную зону (Рис. 36б).

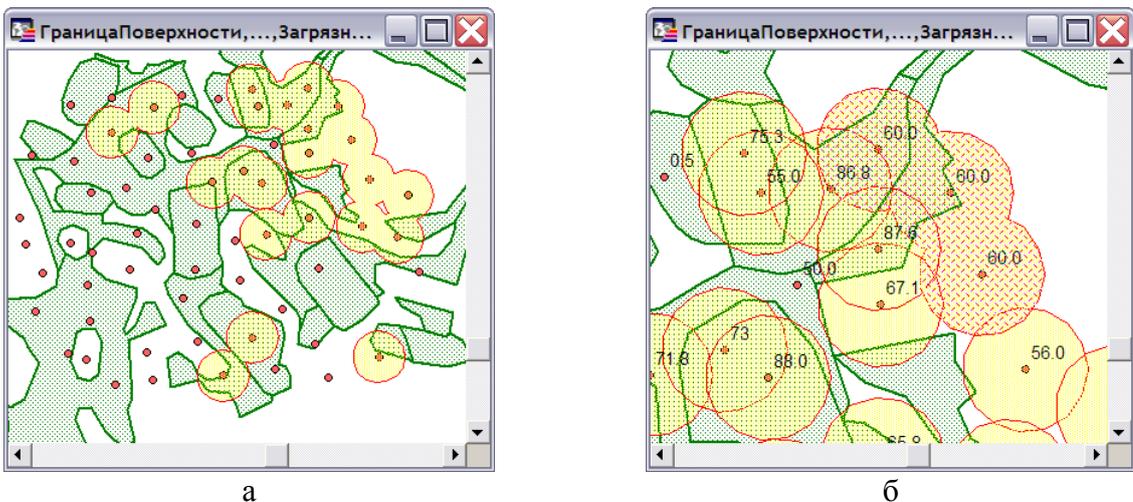


Рис. 36

<sup>29</sup> Следует обратить внимание на то, что существующие объекты не изменяются.

Для операции Voronoi оператор будет иметь вид Create Object As Voronoi From Загрязнения Into Table ЗонаБ. Использование предложения Group By приводит к ошибке. Результат работы оператора, диаграмма Вороного<sup>30</sup>, показан на рис. 37а.

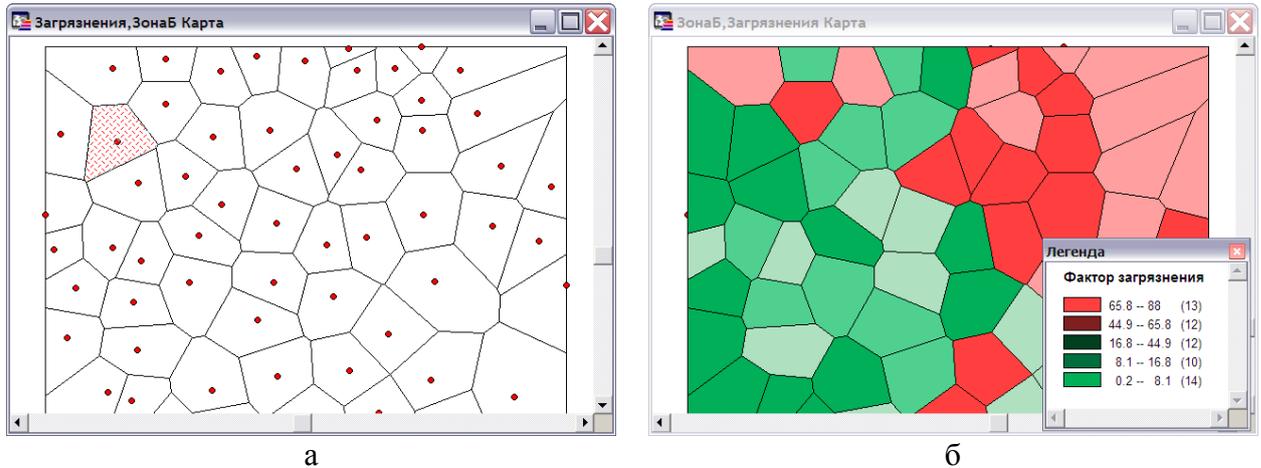


Рис. 37

На рис. 37б представлен результат работы следующей процедуры

```
Create Object As Voronoi From Загрязнения Into Table ЗонаБ Data Фактор=Z
Shade Window idMap ЗонаБ with Фактор Ranges apply color use color Symbol
(35,0,24) 0.2: 8.1 Brush (2,45144,16777215) Pen (1,2,0) Symbol (35,45144,24)
,8.1: 16.8 Brush (2,5296272,16777215) Pen (1,2,0) Symbol (35,28736,24) ,16.8:
44.9 Brush (2,11591872,16777215) Pen (1,2,0) Symbol (35,16416,24) ,44.9: 65.8
Brush (2,16752800,16777215) Pen (1,2,0) Symbol (35,8396832,24) ,65.8: 88
Brush (2,16728128,16777215) Pen (1,2,0) Symbol (35,16728128,24) default Brush
(2,16777215,16777215) Pen (1,2,0) style replace off
Set legend Window idMap layer prev display on shades on symbols off lines off
count on title "Фактор загрязнения" Font ("Arial CYR",1,9,0) subtitle auto
Font ("Arial CYR",0,8,0) ascending off style size small ranges Font ("Arial
CYR",0,8,0) auto display off ,auto display on ,auto display on ,auto display
on ,auto display on ,auto display on
```

Здесь, кроме построения диаграммы Вороного, из базовой таблицы **Загрязнения** в таблицу **ЗонаБ** перенесены значения загрязнения из поля *Z* и далее по этим значениям построен тематический слой и легенда к нему.

Рассмотрим несколько более сложный пример, построим для тех же данных триангуляцию Делоне.

**Код: Построение триангуляции Делоне**

```
Include "mapbasic.def"
Declare Sub Main
Declare Sub VoronoiToDelone
Определим вспомогательный тип данных
type vrn
    vObj as object
```

<sup>30</sup> Диаграмма Вороного (полигоны Тиссена, разбиение Дирихле) – одна из базовых структур вычислительной геометрии. Это такое разбиение плоскости, для конечного множества точек, при котором каждая область этого разбиения образует множество точек, более близких к одному из элементов множества, чем к любому другому элементу множества. Имеет тесную связь с триангуляцией Делоне и можно говорить об однозначном соответствии между диаграммой Вороного и триангуляциями Делоне. А именно, если соединить рёбрами точки, области Вороного которых граничат друг с другом, полученный граф будет являться триангуляцией Делоне.

```

        x as float
        y as float
        z as float
End Type

Sub Main
Set CoordSys table Лес
Set Distance Units "m"
call VoronoiToDelone
end sub

Sub VoronoiToDelone
dim vr() as vrn
dim n,i,j as integer
dim ob1,ob2 as object
Create Object As Voronoi From Загрязнения Into Table ЗонаБ Data Фактор=Z,
x=CentroidX(obj), y=CentroidY(obj)
n=TableInfo(ЗонаБ,TAB_INFO_NROWS)
redim vr(n)
for i=1 to n
    Fetch rec i From ЗонаБ
    vr(i).vObj=ЗонаБ.obj
    vr(i).x=ЗонаБ.x
    vr(i).y=ЗонаБ.y
    vr(i).z=ЗонаБ.Фактор
next
for i=1 to n
    ob1=vr(i).vObj
    for j=1 to n
        if i<>j then
            ob2=vr(j).vObj
            if ob1 Intersects ob2 then
                Insert Into ЗонаБ (obj)
                Values (createLine(vr(i).x,vr(i).y,vr(j).x,vr(j).y))
            end if
        end if
    next
next
select * from ЗонаБ where Str$(obj)="Line" into Selection
Objects Enclose Into Table ЗонаБ
end sub

```

В основной процедуре `VoronoiToDelone` используются три таблицы. В таблице **Загрязнения** хранится информация по точкам с зафиксированным уровнем загрязнения. В таблицах **ЗонаБ** и **ЗонаВ** данных нет.

Все решение можно разделить на четыре этапа:

1. Строим диаграмму Вороного для таблицы **Загрязнения** в таблице **ЗонаБ** и заполняем поля таблицы необходимой информацией.
2. Заполняем вспомогательный массив `vr` типа `vrn` информацией по каждой области Вороного.
3. Формируем стороны треугольников в таблице **ЗонаВ**.
4. Формируем покрытие из треугольников (тип *область*).

Таким образом, мы построили триангуляцию Делоне (Рис. 38). Возможно, для дальнейшего эффективного использования этой конструкции, еще придется позаботиться о формировании соответствующей структуры данных.

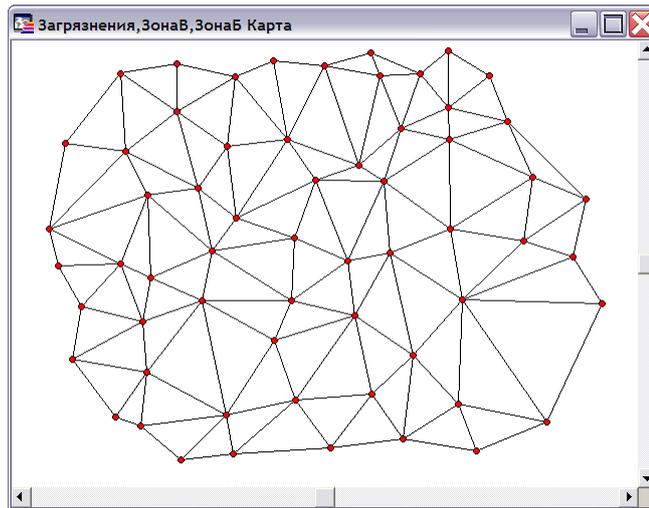


Рис. 38

Операции Union и Merge во многом схожи и чтобы, показать их отличия рассмотрим небольшой пример.

На слое **ЗонаА** имеется пять объектов (рис. 39а). Составим оператор

Create Object As Union From ЗонаА Into Table ЗонаБ

В результате работы этого оператора в таблицу **ЗонаБ** будет записан один объект *область*. Вид этого объекта показан на рис. 39б.

Если для тех же данных использовать оператор Create Object As Merge From ЗонаА Into Table ЗонаБ получим результат, показанный на рис. 39в. Здесь нужно обратить внимание на то, что в результате также получен один объект типа *область*, состоящий из четырех полигонов накладывающихся друг на друга. Вместо пятого полигона, полностью размещавшегося в пределах другого полигона, имеем внутренний полигон (дырку).

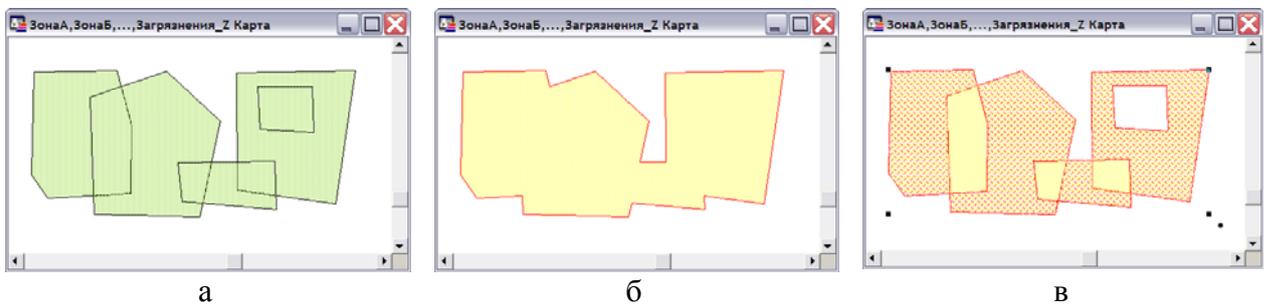


Рис. 39

Применение оператора Create Object As ConvexHull From ЗонаА Into Table ЗонаБ к тем же данным ведет к созданию одного объекта типа *область*, представляющего собой выпуклую оболочку для группы объектов слоя **ЗонаА** (Рис. 40).

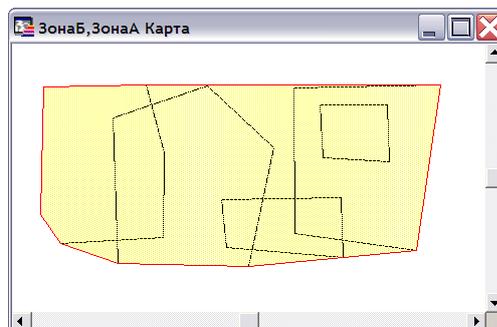


Рис. 40

**Тема 38. Изменение объектов**

Перечислим операторы, действие которых направлено на изменение существующих объектов.

Alter Object	Изменяет форму, положение или графический тип существующего объекта.
Create Cutter	Создает вспомогательный объект делитель. Обычно используется в группе с другими операторами редактирования.
Objects Clean	Коррекция объектов типа область в таблице выборки. Объекты проверяются на предмет топологической корректности.
Objects Check	Проверит указанную таблицу на наличие некорректных данных. Проверяются объекты типа область. Объекты, указатели ошибок, записываются или в исходную таблицу или в таблицу определяемую предложением Into Table.
Objects Combine	Объединяет выбранные объекты (аналог функции меню MapInfo Объекты/Объединить...). Объекты, участвующие в операции, должны быть или площадными или линейными. Также следует обратить внимание на то, в каких случаях эти объекты удаляются из таблицы, а в каких нет. Возможно использование предложения Data для формирования значений в полях новых записей.
Objects Disaggregate	Разъединяет составной объект (область, полилиния, группа точек, коллекция) на составляющие его компоненты.
Objects Enclose	Создает области из контуров, образованных полилиниями (соответствует команде меню MapInfo Объекты/Замкнуть).
Objects Erase	Удаляет часть изменяемого объекта (или весь объект), которая перекрывается другим объектом (или объектами). Соответствует команде меню MapInfo Объекты/Удалить часть...
Objects Intersect	Удаляет часть изменяемого объекта, которая остается свободной от перекрытия другим объектом (или объектами). Соответствует команде меню MapInfo Объекты/Удалить внешнюю часть...
Objects Move	Оператор сдвига выделенного объекта. Соответствует команде меню MapInfo Объекты/Сдвиг.../Переместить объекты.
Objects Offset	Оператор сдвига копии выделенного объекта. Соответствует команде меню MapInfo Объекты/Сдвиг.../Создать копию.
Objects Overlay	Добавляет узлы к изменяемому объекту в точках пересечения с выбранным объектом (или объектами). Тип выбранного объекта может быть любым кроме Text и Point. Соответствует команде меню MapInfo Объекты/Добавить узлы.
Objects Pline	Разрезает выделенную полилинию, состоящую из одной секции, на два полилинейных объекта.
Objects Snap	Выполняет коррекцию объектов таблицы и осуществляет различные топологические операции над ними, включая совмещение узлов разных объектов, прилегающих друг к другу и генерализацию/разреживание узлов. Все объекты в таблице должны быть или линейные или площадные.
Objects Split	Разделяет изменяемый площадной объект (объекты) на части, используя выбранный площадной объект (объекты). Соответствует команде меню MapInfo Объекты/Разрезать... Возможно использование предложения Data для формирования значений в полях новых записей.
Set Resolution	Устанавливает параметр графического разрешения для операций изменения типа объекта. Эта характеристика влияет на количество узлов в объекте, полученном преобразованием типа объекта. Влияет на результаты некоторых операторов и функций MapBasic, таких как ConvertToRegion, ConvertToPline, Objects Split, Combine и др. На действия Create Object As Buffer и Buffer не

Set Target

влияет.  
Назначает/Отменяет выбор изменяемого объекта или объектов.

Следует отметить, что в операциях с объектами, объекты потомки обычно наследуют стили оформления базового объекта.

Выше приводился пример использования оператора Alter Object с предложением Node для формирования области. Рассмотрим использование предложения Info для изменения стиля оформления объекта.

```
dim ob as object
Fetch First From Query1
ob=Query1.obj
Alter Object ob Info OBJ_INFO_PEN,MakePen(2,4,0)
Update Query1 Set obj=ob Where rowid=1
```

Кроме того с оператором Alter Object используется предложение Geography изменяющее расположение объекта. Применяется для всех типов объектов за исключением полилиний и областей, с которыми в этих целях используется предложение Node.

```
ob=CreatePoint(3205800,359600)
Alter Object ob Geography OBJ_GEO_POINTX,3205900
Alter Object ob Geography OBJ_GEO_POINTY,359500
Insert Into ЗонаБ (obj) Values (ob)
```

Далее рассмотрим на небольших примерах действие отдельных операторов.

### 1. Objects Combine

На карте выделены три площадных объекта (Рис. 41а). Использование оператора Objects Combine Data Фактор=avg(Фактор), Состояние=Состояние приведет к созданию нового площадного объекта, являющегося объединением этих выбранных объектов (Рис. 41б). Новый объект (область, состоящая из двух полигонов) будет создан в той же таблице, а исходные объекты будут удалены. Кроме того, для нового объекта, значение в поле *Фактор* будет установлено как среднее из значений этого поля для исходных объектов, значение в поле *Состояние* будет установлено равным значению этого поля для первого объекта в списке исходных объектов. Значения в остальных поля не устанавливаются, они будут равны значениям по умолчанию, как для новой записи. Для продолжения корректной работы с таблицей ее нужно сохранить, упаковать и если необходимо вернуть в карту с установкой необходимых стилей оформления.



Рис. 41

### 2. Objects Split, Create Cutter, Set Target

Имеется две таблицы **Участки** и **ЛЭП**. Один из объектов таблицы **Участки** выделен. Объект (полилиния) из первой строки таблицы **ЛЭП** пересекает выделенный объект. Нужно разделить выделенный участок по линии ЛЭП. Эту задачу решает приведенная ниже процедура. Предполагается, что проверка выделенного объекта уже выполнена.

```

Sub tstSplit
dim tmpTab as string
Set Map Layer Участки Editable On
Set Target On
Select * From ЛЭП where rowid=1
Create Cutter Into Target
tmpTab=SelectionInfo(SEL_INFO_SELNAME)
Objects Split Into Target
Delete From tmpTab
end sub

```

Приведенные ниже рисунки иллюстрируют процесс решения. На рис. 42а показана исходная ситуация. На рис. 42б представлены, изменяемый объект – участок и вспомогательный объект сформированный оператором `Create Cutter Into Target`. Окончательное решение на рис. 42в. В таблице **Участки** появятся две, помеченные удаленными записи, это исходный участок и вспомогательный объект и две новые записи – два новых участка.

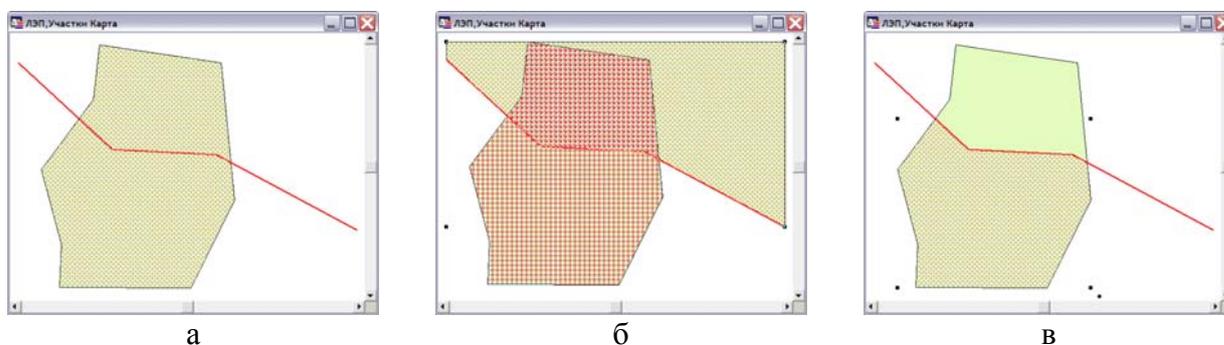


Рис. 42

### 3. Objects Disaggregate

В таблице **Участки** имеется составной объект, состоящий из двух внешних полигонов и одного внутреннего (дырка). Разъединим этот объект на составляющие компоненты. Для этого выделим объект (Рис. 43а) и используем оператор `Objects Disaggregate`. В результате исходный объект будет удален и в таблицу **Участки** будут добавлены два новых объекта эквивалентные компонентам базового объекта (Рис. 43б). Если использовать оператор в виде `Objects Disaggregate Into Table ЗонаВ`, то исходный объект в таблице **Участки** остается, а в таблицу **ЗонаВ** добавляется два новых объекта – компоненты базового объекта. Обратите внимание, что в обоих случаях дырка остается (Рис. 43б). Если использовать оператор в виде `Objects Disaggregate Into Table ЗонаВ All`, то исходный объект в таблице **Участки** остается, а в таблицу **ЗонаВ** добавляется три новых объекта. Два представляют внешние полигоны (без дырок) и один представляет внутренний полигон базового объекта (Рис. 43в).

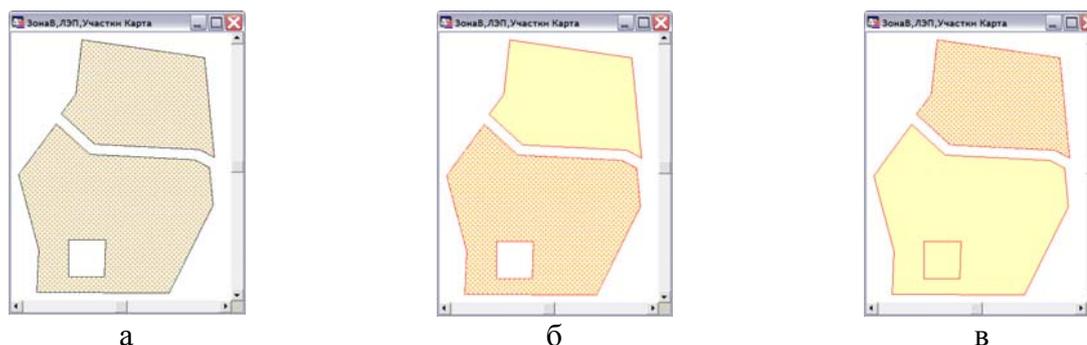


Рис. 43

В операторе может использоваться дополнительное предложение Data определяющее значения в полях новых записей.

#### 4. Objects Erase, Objects Intersect, Objects Move

В таблице **Участки** выделен участок (Рис. 44а). Имеется таблица **Ограничения** с площадными объектами. Требуется изменить участок в соответствии с имеющимися ограничениями.

Для решения задачи используем следующую процедуру.

```
Sub tstErase
Set Map Layer Участки Editable On
Set Target On
Select * From Ограничения
Objects Erase Into Target
Objects Move angle 240 distance 30 units "m" type Cartesian
end sub
```

Результат работы процедуры показан на рис. 44б. Оператор `Objects Move` включен в процедуру исключительно с целью, чтобы повысить наглядность результата. Если в данной процедуре заменить оператор `Objects Erase Into Target` на оператор `Objects Intersect Into Target` получим результат показанный на рис. 44в.

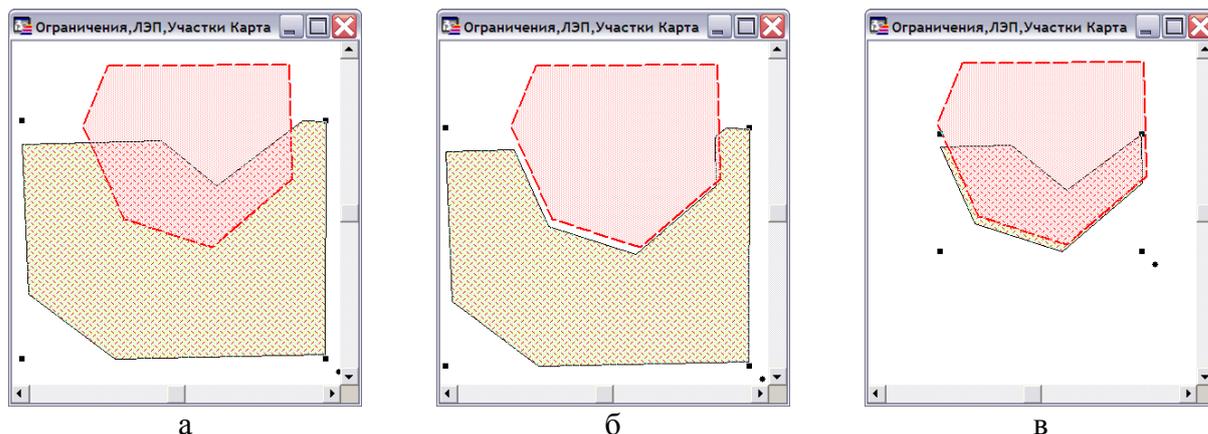


Рис. 44

В обоих случаях базовый объект будет удален из таблицы **Участки** и добавлен в нее новый измененный объект.

В операторах `Objects Erase` и `Objects Intersect` может использоваться дополнительное предложение `Data` определяющее значения в полях новых записей.

#### 5. Objects Overlay

Есть таблица **Участки** и таблица **Ограничения**. Требуется для выделенного участка в местах его пересечения с границами ограничений добавить новые узлы. Задачу решает следующая процедура

```
Sub tstOverlay
Set Map Layer Участки Editable On
Set Target On
Select * From Ограничения
Objects Overlay Into Target
end sub
```

#### 6. Objects Snap

```
Objects Snap From Selection
```

Для выделенного объекта будут удалены все дублирующиеся узлы.

#### 7. Objects Pline

В таблице ЛЭП выделена полилиния. Требуется разделить эту полилинию в узле номер 4 на две полилинии<sup>31</sup>.

Для решения этой задачи можно использовать оператор `Objects Pline` в следующем виде `Objects Pline Split At Node 4`. В этом случае базовый объект будет удален из таблицы ЛЭП и в нее будут записаны две новые полилинии: полилиния с узла 1 базового объекта по узел 4 и полилиния с узла 4 по последний узел базового объекта.

Если нужно сохранить базовый объект, то следует использовать предложение `Into Table: Objects Pline Split At Node 4 Into Table ЗонаВ`. Здесь новые объекты будут сформированы в таблице **ЗонаВ**, а базовый объект в таблице ЛЭП не изменится.

В операторе `Objects Pline` может использоваться дополнительное предложение `Data` определяющее значения в полях новых записей.

Кроме операторов, для изменения объектов, можно использовать следующие функции.

<code>Combine</code>	Возвращает объект, являющийся результатом объединения двух объектов аргументов. В качестве аргументов могут выступать линейные, площадные, точечные объекты, группы точек, коллекции. Текстовые объекты объединять нельзя.
<code>ConvertToPline</code>	Возвращает полилинию, в которую конвертируется объект аргумент. Тип объекта аргумента любой кроме текстового и точечного. Функция создает новый объект, не меняя объект аргумент.
<code>ConvertToRegion</code>	Возвращает область, в которую конвертируется объект аргумент. Тип объекта аргумента любой кроме текстового и точечного. Функция создает новый объект, не меняя объект аргумент.
<code>Erase</code>	Возвращает объект, полученный в результате удаления части существующего объекта (аргумент 1) перекрываемого другим объектом (аргумент 2).
<code>OverlayNodes</code>	Возвращает объект, созданный на основе существующего (аргумент 1), добавлением узлов в точках пересечения со вторым объектом (аргумент 2).
<code>Overlap</code>	Возвращает объект, полученный в результате пересечения двух объектов аргументов. Объекты могут быть площадного или линейного типа.
<code>Rotate</code>	Возвращает объект, повернутый на заданное число градусов вокруг точки центра. Положительное вращение против часовой стрелки. Исходный объект не изменяется. Для всех типов объектов кроме текстовых <sup>32</sup> .
<code>RotateAtPoint</code>	Возвращает объект, повернутый на заданное число градусов вокруг заданной точки. Исходный объект не изменяется. Для всех типов объектов кроме текстовых.

Рассмотрим примеры использования некоторых функций.

В процедуре `tstConvertToRegion` формируются две непересекающиеся окружности (объекты `ob1` и `ob2`). И далее в таблицу **ЗонаВ** вставляются две области, результат конвертирования окружности при разных установках оператора `Set Resolution`. Результат показан на рис. 45.

```
Sub tstConvertToRegion
dim ob1,ob2 as object
ob1=CreateCircle(3205630,360280,100)
ob2=CreateCircle(3205850,360280,100)
Set Resolution 7
```

<sup>31</sup> Нумерация узлов в полилинии начинается с 1.

<sup>32</sup> Для поворота текстовых объектов используется оператор `Alter Object ... Geography` с кодом `OBJ_GEO_TEXTANGLE`.

```
insert Into ЗонаВ (obj) Values (ConvertToRegion(ob1))
Set Resolution 100
insert Into ЗонаВ (obj) Values (ConvertToRegion(ob2))
end sub
```

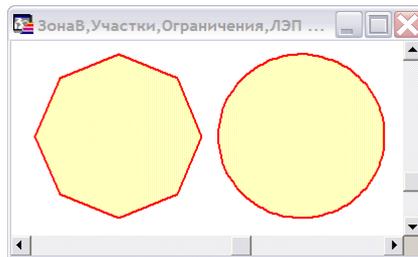


Рис. 45

Рассмотрим технику использования функций Erase, OverlayNodes, RotateAtPoint, Rotate на некотором искусственном примере.

**Код: Использование функций Erase, OverlayNodes, RotateAtPoint, Rotate**

```
Sub tstFunction
dim ob1,ob2,ob3,p1 as object
dim tmpTab as string
dim aa as alias
'Определяем выделенный объект на слое Участки (ob1)
tmpTab=SelectionInfo(SEL_INFO_SELNAME)
aa=tmpTab & ".obj"
ob1=aa
'Определяем объект на слое Ограничения (ob2)
select * from Ограничения Where rowid=3
tmpTab=SelectionInfo(SEL_INFO_SELNAME)
aa=tmpTab & ".obj"
ob2=aa
'Точка вращения для ob1.
p1=CreatePoint(3206260,360100)
'Поворачиваем ob1 на 26.5° против часовой стрелки вокруг p1.
ob3=RotateAtPoint(ob1,26.5,p1)
'Удаляем часть ob3 перекрываемую ob2.
ob3=Erase(ob3,ob2)
'Поворачиваем ob3 вокруг центроида на 10°.
ob3=Rotate(ob3,10)
'К ob3 добавляем узлы в местах пересечения с ob2. Результат присваиваем ob1.
ob1=OverlayNodes(ob3,ob2)
'Добавляем ob1 к таблице ЗонаВ.
insert Into ЗонаВ (obj) Values (ob1)
end sub
```

Результат работы процедуры показан на рис. 46.

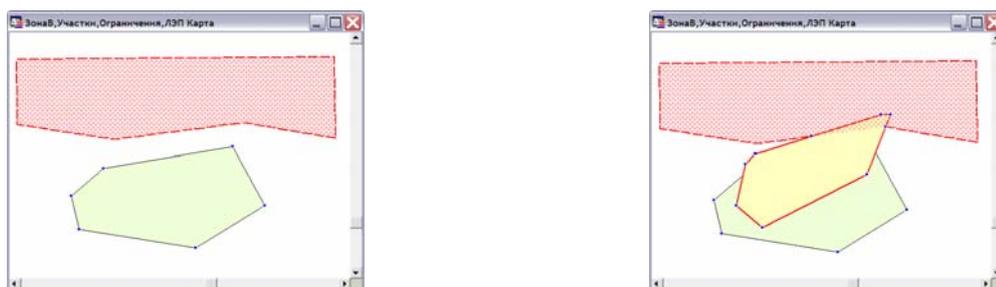


Рис. 46

### Тема 39. Извлечение информации об объектах

Операторы и функции, предназначенные для извлечения информации об объектах, довольно многочисленны. Рассматривать их вне контекста реальной задачи нет особого смысла. Для понимания их работы достаточно ознакомиться со справочной документацией. Поэтому здесь дается лишь их перечисление с краткими пояснениями.

Area	Возвращает площадь объекта в указанных единицах измерения.
AreaOverlap	Возвращает площадь пересечения двух площадных объектов аргументов.
Distance	Возвращает расстояние между двумя точками заданными своими координатами в указанных единицах измерения.
Perimeter	Возвращает периметр площадного объекта в указанных единицах измерения.
ObjectLen	Возвращает длину объекта аргумента (линия, полилиния) в указанных единицах измерения.
CartesianObjectLen	Возвращает длину (на плоскости) объекта аргумента (линия, полилиния) в указанных единицах измерения. Для некорректных данных возвращает -1.
SphericalObjectLen	Возвращает длину (на сфере) объекта аргумента (линия, полилиния) в указанных единицах измерения. Для некорректных данных возвращает -1.
Centroid	Возвращает точечный объект, имеющий координаты центроида объекта.
CentroidX	Возвращает координату X центроида объекта. Ось X направлена на восток.
CentroidY	Возвращает координату Y центроида объекта. Ось Y направлена на север.
ConnectObjects	Возвращает объект, полилинию из одного сегмента и двух точек, представляющий минимальное или максимальное расстояние между объектами аргументами.
CartesianConnectObjects	Аналог ConnectObjects. Все вычисления на плоскости. Если вычисления не могут быть сделаны, то функция выдаст сообщение об ошибке.
SphericalConnectObjects	Аналог ConnectObjects. Все вычисления на сфере. Если вычисления не могут быть сделаны, то функция выдаст сообщение об ошибке.
ObjectDistance	Возвращает расстояние между двумя объектами аргументами.
CartesianObjectDistance	Аналог ObjectDistance. Все вычисления на плоскости. Если вычисления не могут быть сделаны, то функция выдаст сообщение об ошибке.
SphericalObjectDistance	Аналог ObjectDistance. Все вычисления на сфере. Если вычисления не могут быть сделаны, то функция выдаст сообщение об ошибке.
ExtractNodes	Возвращает объект (полилиния или область) из подмножества узлов существующего объекта. Нередко используется для копирования полигона из области.
IntersectNodes	Возвращает объект (полилиния) образованный из подмножества узлов образованных пересечением объектов аргументов. Конкретный вид результата определяется кодом.
MBR	Возвращает объект (прямоугольник), представляющий минимальное прямоугольное покрытие объекта аргумента.
ObjectGeography	Возвращает информацию о графическом объекте аргументе в зависимости от кода, определяющего результат. В целом это информация о координатах, углах, центроиде объекта, о параметрах минимального

ObjectInfo	прямоугольного покрытия. Возвращает информацию о графическом объекте аргументе в зависимости от кода, определяющего результат. Возвращается информация о типе объекта, стиле оформления, числе полигонов (сегментов), числе узлов в полигоне (сегменте) и др.
ObjectNodeX	Возвращает значение координаты X определенного узла для указанного полигона (сегмента) объекта аргумента.
ObjectNodeY	Возвращает значение координаты Y определенного узла для указанного полигона (сегмента) объекта аргумента.
ProportionOverlap	Возвращает процент перекрытия одного объекта другим по отношению к площади первого, в списке аргументов, объекта. Объекты не могут быть точечными или текстовыми.

Если, в программе, используется координатная система не *план-схема*, то расстояние, длина, периметр, площадь вычисляются на сфере в противном случае на плоскости.

Если в списке аргументов функции присутствуют единицы измерения, то результат будет получен в этих единицах, если нет то в соответствии с установками операторов `Set Distance Units`, `Set Area Units` или по умолчанию.

Отдельные функции все же требуют некоторого пояснения. Приведем пример кода для функции `ConnectObjects`.

'Минимальное расстояние

```
ob1=ConnectObjects (ob3,ob4, "Т")
insert into ЗонаВ (obj) Values (ob1)
```

'Максимальное расстояние

```
ob1=ConnectObjects (ob3,ob4, "Ф")
insert into ЗонаВ (obj) Values (ob1)
```

На рис. 47 показан результат работы этого кода. `ob3` и `ob4` объекты типа область, причем `ob4` состоит из двух полигонов.

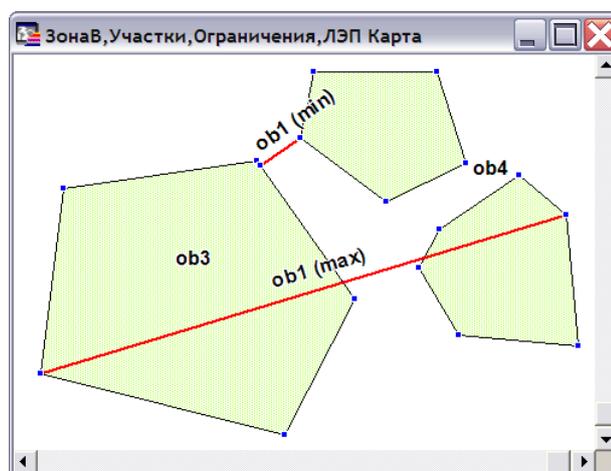


Рис. 47

Протестируем работу функции `ExtractNodes`.

```
ob1=ExtractNodes (ob3,1,1,4,0)
insert into ЗонаВ (obj) Values (ob1)
```

```
ob1=ExtractNodes (ob3,1,1,4,1)
insert into ЗонаВ (obj) Values (ob1)
```

Результат работы приведенного кода показан на рис. 48. В первом случае получена полилиния, во втором область.

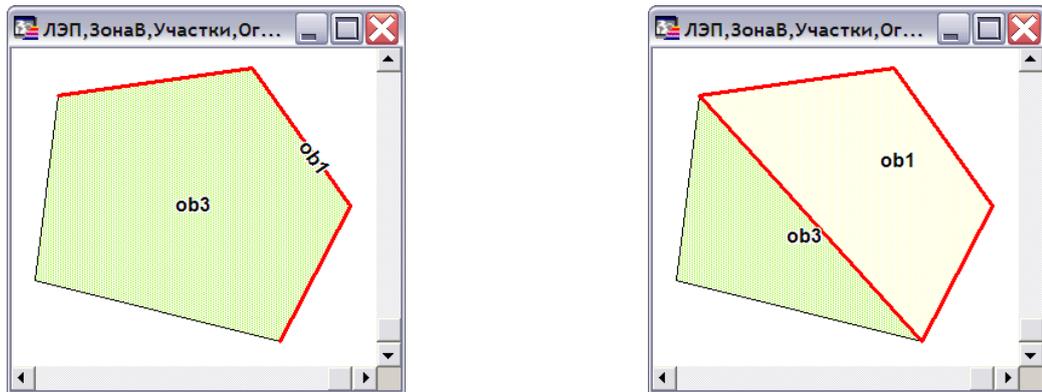


Рис. 48

Для получения координат всех узлов объекта (область, полилиния) можно использовать следующий код<sup>33</sup>.

**Код: Получить координаты всех узлов объекта**

```
dim ob3 as object
dim k,n,i,j as integer
ob3=selection.obj
'Число полигонов в области
k=ObjectInfo(ob3,OBJ_INFO_NPOLYGONS)
for i=1 to k
'Число узлов в i-ом полигоне
n=ObjectInfo(ob3,OBJ_INFO_NPOLYGONS+i)
print "Полигон " & str$(i)
for j=1 to n
print "узел " & str$(j)
print Format$(ObjectNodeX(ob3,i,j),"#. #0") &
", " & Format$(ObjectNodeY(ob3,i,j),"#. #0")
next
next
```

Приведем другие примеры использования функций.

```
Print ProportionOverlap(ob3,ob4) или эквивалентный код Print
Area(Overlap(ob3,ob4),"sq m")/Area(ob3,"sq m"). В обоих случаях получено одно и
то же значение 0.400223, т.е. объект ob4 перекрывает 40.02% площади объекта ob3.
Print ObjectDistance(ob3, ob4, "m"). Если объекты пересекаются, получим 0, если
нет – реальное минимальное расстояние между объектами.
Set Area Units "sq m"
Print AreaOverlap(ob3,ob4).
```

Получим площадь перекрытия объекта ob3 объектом ob4 в квадратных метрах.

Кроме функций для извлечения информации можно использовать следующие операторы.

1. Farthest. Служит для определения объектов максимально удаленных от заданного. Результат работы – полилиния (полилинии) записываемая в указанную таблицу. Далее показаны простейшие формы этого оператора.

<sup>33</sup> Для области первый узел повторяется в конце списка.

Farthest From Variable ob3 To Загрязнения Into ЗонаБ. В таблице **Загрязнения** ищется объект максимально удаленный от объекта ob3. Результат (полилиния) записывается в таблицу **ЗонаБ**. Вид окна карты на рис. 49а.

Farthest 3 From Variable ob3 To Загрязнения Into ЗонаБ. В таблице **Загрязнения** ищется три объекта максимально удаленных от объекта ob3. Результат (три полилинии) записывается в таблицу **ЗонаБ**. Вид окна карты на рис. 49б.

Farthest All From Variable ob3 To Загрязнения Into ЗонаБ. Определяется удаление объекта ob3 от каждого объекта таблицы **Загрязнения**. Результат (полилинии, по числу объектов в таблице **Загрязнения**) записывается в таблицу **ЗонаБ**. Вид окна карты на рис. 49в.

Farthest From Table ЗонаВ To Загрязнения Into ЗонаБ. В таблице **Загрязнения** определяются объекты максимально удаленные от объектов таблицы **ЗонаВ**. Результат (полилинии, по числу объектов в таблице **ЗонаВ**) записывается в таблицу **ЗонаБ**. Вид окна карты на рис. 49г.

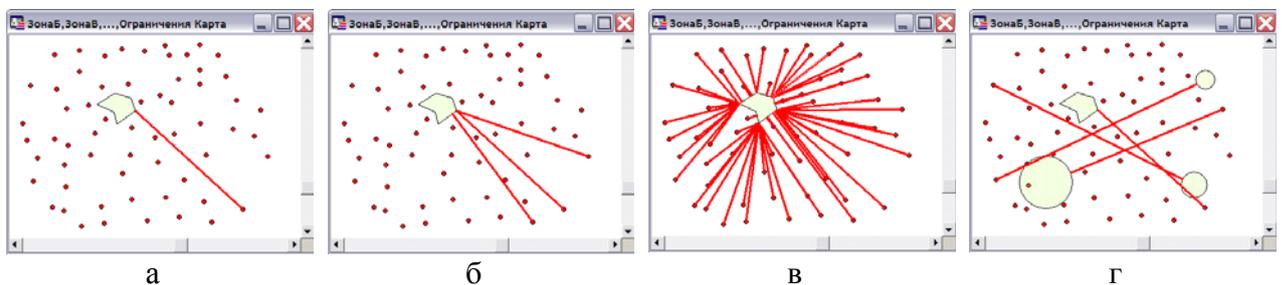


Рис. 49

2. Nearest. Служит для определения объектов минимально удаленных от заданного. Результат работы – полилиния (полилинии) записываемая в указанную таблицу. По синтаксису оператор полностью аналогичен предыдущему, по результату – его антипод.

## Тема 40. Управление стилями объектов

В понятие управление стилями входит как изменение стилей, так и определение их текущего состояния.

Рассмотрим программу позволяющую определить стиль выделенного в карте объекта. Итак, на входе наша программа должна иметь выделенный на карте объект, результат – информация по данному объекту будет выводиться в окно **Сообщения**. Ограничений в данном случае немного — должен быть выбран только один объект. Вся программа будет состоять из одного модуля.

В блок описаний входит строка подключения файла **mapbasic.def** (описание констант MapBasic) и декларирование трех процедур.

Главная процедура **Main** включает только вызов подпрограммы **GetStyle**.

**Код: Определение стиля выделенного на карте объекта**

```
Include "mapbasic.def"
Declare Sub Main
Declare Sub GetStyle
Declare sub SelectionInfo(iSelRows as integer,
                        sBaseTab as string,sTmpTab as string)

Sub Main
Call GetStyle
```

```

end sub

Sub GetStyle
'Описание локальных переменных для процедуры
dim iRows as integer
dim sObjTab as string
dim sSelTab as string
dim ss,st as string
dim selObj as object
'Получаем информацию по выборке
call SelectionInfo(iRows,sObjTab,sSelTab)
'Проверка «объекты не выбраны»
if iRows=0 then
    Note "Объект не выбран!"
    exit sub
end if
'Проверка «объектов больше одного»
if iRows>1 then
    Note "Выбрано объектов больше одного!"
    exit sub
end if
'Устанавливаем курсор на первую строку таблицы с выборкой
Fetch First From sSelTab
dim aa as alias
'Определяем выбранный объект
aa= sSelTab & ".obj"
selObj=aa
'Формируем строку с информацией по объекту
ss="Выбран 1 объект" & chr$(10) & "на слое " & sObjTab &
chr$(10) & "Имя выборки: " & sSelTab &
chr$(10) & "Тип объекта: " & str$(selObj)
'Определяем стиль объекта в зависимости от его типа
'Здесь определяется реальный стиль объекта
'(глобальные настройки для слоя к этому отношения не имеют)
Do Case ObjectInfo(selObj, OBJ_INFO_TYPE)
    Case OBJ_TYPE_ARC,OBJ_TYPE_LINE,OBJ_TYPE_PLINE
        st=ObjectInfo(selObj, OBJ_INFO_PEN)
    Case OBJ_TYPE_ELLIPSE,OBJ_TYPE_FRAME,OBJ_TYPE_REGION,
        OBJ_TYPE_RECT,OBJ_TYPE_ROUNDRECT
        st=ObjectInfo(selObj, OBJ_INFO_PEN) & ", " &
        ObjectInfo(selObj, OBJ_INFO_BRUSH)
    Case OBJ_TYPE_POINT
        st=ObjectInfo(selObj, OBJ_INFO_SYMBOL)
    Case OBJ_TYPE_TEXT
        st=ObjectInfo(selObj, OBJ_INFO_TEXTFONT)
End Case
'Дописываем стиль объекта в информационную строку
'и выводим ее в окно «Сообщения»
'При необходимости можно предварительно очистить это окно
'с помощью оператора Print Chr$(12)
ss=ss & chr$(10) & "Стиль объекта: " & st
print ss
end sub

sub SelectionInfo(iSelRows as integer,sBaseTab as string,
                 sTmpTab as string)
    sBaseTab =""
    sTmpTab =""
'Определяем число объектов в выборке
iSelRows =SelectionInfo(SEL_INFO_NROWS)

```

'Если выборка не пуста, определяем имена таблиц:

'таблицы из которой выбирались объекты и временной таблицы с выборкой

```

if iSelRows>0 then
    sBaseTab =SelectionInfo (SEL_INFO_TABLENAME)
    sTmpTab =SelectionInfo (SEL_INFO_SELNAME)
end if
end sub

```

Результат работы программы представлен на рис. 50.

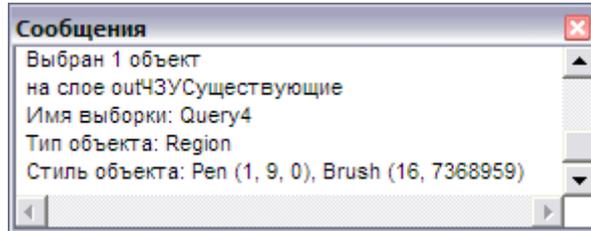


Рис. 50

Вид карты определяется как стилевым оформлением слоев в целом, так и настройкой стилей для отдельных объектов. Для контроля используемых стилей, формирования и установки новых стилей в MapBasic используются специальные функции и операторы. Функции, возвращающие информацию о стилях:

CurrentBrush	Возвращает текущий стиль заливки.
CurrentPen	Возвращает текущий стиль линии (общий).
CurrentLinePen	Возвращает текущий стиль линии (полилинии).
CurrentBorderPen	Возвращает текущий стиль линии границы области.
CurrentFont	Возвращает текущий стиль для текста.
CurrentSymbol	Возвращает текущий стиль для символа точечного объекта.
TextSize	Возвращает размер шрифта (в пунктах) для текстового объекта в текущем окне (карта, отчет).
StyleAttr	Возвращает значение одной из компонент стиля оформления объекта.
PenPattern	Возвращает уточненный тип линии с учетом стиля пересечения линий.
PenWidthToPoints	Возвращает толщину линии в пунктах (points).
PointsToPenWidth	Возвращает толщину линии в условных единицах. Функция обратная по отношению к PenWidthToPoints.
IsPenWidthPixels	Определяет, в каких единицах задана толщина линии в пикселах или пунктах. Если возвращает TRUE то в пикселах, если FALSE то в пунктах.
RGB	Возвращает целочисленное значение цвета в системе RGB.

Для определения стилей оформления объекта также используется описанная выше функция ObjectInfo.

Далее приведу комментарии к работе отдельных функций.

1. TextSize.

Введем в окне MapBasic оператор `print TextSize(FrontWindow(), selection.obj)`.

Получили результат 10 пунктов. Увеличим изображение в окне карты, и тот же оператор дает результат 20 пунктов. Это иллюстрация того, что размер текстового объекта в карте величина переменная и зависит от масштаба изображения. Если проделать тот же опыт для окна отчета увидим что размер текста во всех случаях величина постоянная.

2. PenPattern.

Функция определяет тип линии с учетом стиля пересечения линий. Имеет смысл для сложных двойных линий<sup>34</sup>. Рассмотрим два оператора: `print PenPattern(63, "F")` и `print PenPattern(63, "T")`. В данном случае 63 это нормальный код линии и результат работы первого оператора будет значение 63. Ситуация с пересечением таких линий показана на рис. 51а. Результатом работы второго оператора будет значение 191 и пересечение подобных линий показано на рис. 51б.

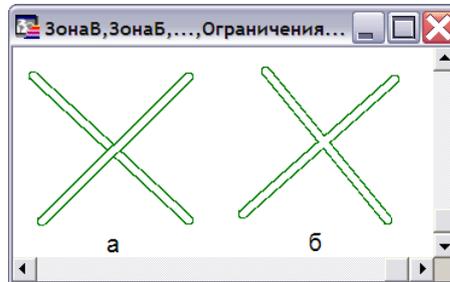


Рис. 51

3. `PointsToPenWidth`, `PenWidthToPoints`, `StyleAttr`, `Set Style`.

В `MapInfo` толщина линии может назначаться как в пикселях, так и в пунктах (*points*). При установке стиля линии с помощью оператора `Set Style` параметр, отвечающий за толщину линии, определяется следующим образом:

- ✓ Значение параметра 1, 2, ..., 8, 9, 10 соответствует толщине линии в пикселях 1, 2, ..., 8, 9, 1.
- ✓ Значение параметра 11, 12, ..., 70 соответствует толщине линии в пунктах 0.1, 0.2, ..., 6.0.

Так, например, оператор `Set Style Pen MakePen(4, 2, 32768)` установит для текущего стиля линии толщину 4 пикселя, а `Set Style Pen MakePen(15, 2, 32768)` – 0.5 пункта. Функция `PenWidthToPoints` позволяет получить значение толщины линии в пунктах. Аргументом функции является значение рассмотренного выше параметра. Так оператор `print PenWidthToPoints(15)` даст результат 0.5, а `print PenWidthToPoints(4)` даст сообщение об ошибке.

Функция `PointsToPenWidth` позволяет получить значение параметра толщины по значению в пунктах. Оператор `print PointsToPenWidth(0.5)` даст результат 15. Оператор, в окне `MapBasic`, `print StyleAttr(CurrentPen(), 1)` позволяет получить толщину линии для текущего стиля. Если толщина линии составляет 4 пикселя, функция вернет значение 4, если же толщина была установлена как 0.5 пункта, то получим 15. В программе, чтобы правильно определиться, в каких единицах задана толщина линии, обычно используют функцию `IsPenWidthPixels`.

Для установки стиля кроме указанного выше оператора `SetStyle` используются:

<code>MakeBrush</code> <code>MakePen</code> <code>MakeFont</code> <code>MakeSymbol</code>	Возвращает величину типа <code>Brush</code> , определяющую стиль заливки. Возвращает величину типа <code>Pen</code> , определяющую стиль линии. Возвращает величину типа <code>Font</code> , определяющую стиль текста. Возвращает величину типа <code>Symbol</code> , определяющую стиль отображения точечного объекта. Для символов формата <code>MapInfo</code> версии 3.
<code>MakeFontSymbol</code>	Возвращает величину типа <code>Symbol</code> , определяющую стиль отображения точечного объекта. Для определения символов используются шрифты TrueType ( <code>ttf</code> ).
<code>MakeCustomSymbol</code>	Возвращает величину типа <code>Symbol</code> , определяющую стиль

<sup>34</sup> Для простых сплошных линий понятие стиль пересечения не определено.

Reload Symbols	отображения точечного объекта. Определение символа основано на растровом файле. Открывает и перезагружает файл символов MapInfo (для версии 3) или устанавливает папку для растровых символов.
----------------	---

Для изменения стилей оформления объекта также используется оператор `Alter Object`. При установке стилей конкретная настройка стиля определяется его описанием `Brush`, `Pen`, `Font`, `Symbol`. Отдельные параметры этих описаний приведены в приложении. Рассмотрим примеры установки текущего стиля линии.

Установить стиль для линейных объектов (линии, полилинии).

```
Set Style LinePen MakePen(3, 3, 5623)
```

Установить стиль границы для объектов типа область.

```
Set Style BorderPen MakePen(1, 2, 5623)
```

Установить один и тот же стиль линий для линейных и площадных объектов.

```
Set Style Pen MakePen(5, 2, 5623)
```

### Тема 41. Получение информации о карте

Информацию о карте в целом дают такие функции как `MapperInfo` и `LayerInfo`. Эти функции уже встречались ранее в коде примеров, здесь же они будут рассмотрены более подробно.

#### 1. `MapperInfo`.

Функция возвращает в основном метрическую информацию об окне карты такую как координаты центральной точки окна, минимальные и максимальные значения координат для участка, отображаемого в окне, система координат, текущий масштаб карты и т.д. Тип возвращаемого значения зависит от управляющего кода в списке аргументов. Приведем несколько примеров записи функции с описанием возвращаемого значения:

<code>MapperInfo (idMap, MAPPER_INFO_AREAUNITS)</code>	Текущая единица измерения площади в виде строки (sq m).
<code>MapperInfo (idMap, MAPPER_INFO_DISTUNITS)</code>	Текущая единица измерения расстояния в виде строки (m).
<code>MapperInfo (idMap, MAPPER_INFO_CENTERX)</code>	Координата X центра окна (3208251.12).
<code>MapperInfo (idMap, MAPPER_INFO_CENTERY)</code>	Координата Y центра окна (360173.64).
<code>MapperInfo (idMap, MAPPER_INFO_MAXX)</code>	Максимальная координата X части карты, показанной в окне (3208600.44).
<code>MapperInfo (idMap, MAPPER_INFO_MINY)</code>	Минимальная координата Y части карты, показанной в окне (359940.76).
<code>MapperInfo (idMap, MAPPER_INFO_COORDSYS_CLAUSE)</code>	Строка, соответствующая предложению <code>CoordSys (CoordSys NonEarth Units "m")</code> для окна <code>idMap</code> .
<code>MapperInfo (idMap, MAPPER_INFO_DISPLAY)</code>	Код, определяющий тип информации показываемой в строке сообщений (2 – положение курсора)
<code>MapperInfo (idMap, MAPPER_INFO_EDIT_LAYER)</code>	Номер редактируемого слоя (1).
<code>MapperInfo (idMap, MAPPER_INFO_SCALE)</code>	Текущий масштаб: количество единиц измерения расстояния ( <code>Set Distance Units</code> ), помещающееся в одной "бумажной" единице. "Бумажные" единицы описываются оператором <code>Set Paper Units</code> . По умолчанию используются дюймы.
<code>MapperInfo (idMap, MAPPER_INFO_ZOOM)</code>	Размер участка карты

MapperInfo(idMap, MAPPER\_INFO\_NUM\_THEMATIC)

показанного в окне (по ширине окна) в текущих единицах измерения расстояния. Число тематических слоев в карте.

2. LayerInfo.

Функция возвращает информацию о конкретном слое в окне карты. Слой определяется своим номером в списке слоев окна. Возвращаемое значение зависит от целочисленного кода в списке аргументов функции. Большинство кодов применимо только для "нормальных" слоев карты, то есть использование этой функции для специальных слоев (косметическому, тематическому, растровому) ограничено.

В списке аргументов функции присутствует номер слоя и сразу встает вопрос, как его получить. Приводимая ниже функция позволяет по имени слоя nameLr в окне карты idWin получить его номер.

**Код: Определение номера слоя**

```
Function GetNumLayer(byval idWin as integer,byval nameLr as string) as
smallint
dim n,i as smallint
GetNumLayer=-1
n=MapperInfo(idWin,MAPPER_INFO_LAYERS)
for i=1 to n
    if LayerInfo(idWin,i,LAYER_INFO_NAME)=nameLr then
        GetNumLayer=i
        exit function
    end if
next
end function
```

Далее, получив номер слоя, необходимо убедиться, что слой имеет требуемый тип. Например, проверку на нормальный тип слоя можно выполнить с помощью следующей функции.

**Код: Определение нормального слоя**

```
Function IsNormalLayer(byval idWin as integer,
byval numLr as smallint) as logical
dim tp as smallint
IsNormalLayer="F"
tp=LayerInfo(idWin,numLr,LAYER_INFO_TYPE)
if tp=LAYER_INFO_TYPE_NORMAL then
    IsNormalLayer="T"
end if
end function
```

Конечно, этот вопрос можно решить с помощью одного оператора вида If LayerInfo(idWin,numLr,LAYER\_INFO\_TYPE)=LAYER\_INFO\_TYPE\_NORMAL then..., но если эта задача встает достаточно часто то лучше оформить ее в виде отдельной процедуры.

Рассмотрим, какого вида информацию можно получить через функцию LayerInfo. Приведем наиболее часто используемые варианты записи функции с описанием возвращаемого значения.

LayerInfo(idWin,numLr,LAYER\_INFO\_EDITABLE)

LayerInfo(idWin,numLr,LAYER\_INFO\_SELECTABLE)

LayerInfo(idWin,numLr,LAYER\_INFO\_PATH)

Значение типа Logical: TRUE если слой редактируемый.  
Значение типа Logical: TRUE если слой доступен.  
Строка, содержащая полный путь

```

LayerInfo (idWin, numLr, LAYER_INFO_DISPLAY)
LayerInfo (idWin, numLr, LAYER_INFO_OVR_LINE)

LayerInfo (idWin, numLr, LAYER_INFO_OVR_PEN)

LayerInfo (idWin, numLr, LAYER_INFO_OVR_BRUSH)

LayerInfo (idWin, numLr, LAYER_INFO_OVR_SYMBOL)

LayerInfo (idWin, numLr, LAYER_INFO_OVR_FONT)

LayerInfo (idWin, numLr, LAYER_INFO_LBL_FONT)

LayerInfo (idWin, numLr, LAYER_INFO_LBL_EXPR)

LayerInfo (idWin, numLr, LAYER_INFO_NODES)

```

для файла таблицы, данные которой представлены на слое. Целочисленный код, указывающий на режим отображения слоя. Строка, содержащая описание стиля Pen для отображения линейных объектов (для режима LAYER\_INFO\_DISPLAY\_GLOBAL<sup>35</sup>). Строка, содержащая описание стиля Pen для отображения площадных объектов (для режима LAYER\_INFO\_DISPLAY\_GLOBAL). Строка, содержащая описание стиля Brush для отображения площадных объектов (для режима LAYER\_INFO\_DISPLAY\_GLOBAL). Строка, содержащая описание стиля Symbol для отображения точечных объектов (для режима LAYER\_INFO\_DISPLAY\_GLOBAL). Строка, содержащая описание стиля Font для отображения текстовых объектов (для режима LAYER\_INFO\_DISPLAY\_GLOBAL). Строка, содержащая описание стиля Font для отображения подписей. Строка, содержащая выражение формирующее подпись, в простейшем случае имя поля<sup>36</sup>. Значение типа Logical: TRUE если показываются узлы объектов.

Кроме того с помощью данной функции можно получить исчерпывающую информацию о подписях (положение, поворот, указка, видимость, влияние масштабного эффекта и т.д.).

### **Тема 42. Изменение изображения в окне карты**

Открыть новое окно карты можно с помощью оператора Map from, например

```

Set paper Units "mm"
Map from ЗонаВ,Участки Position(60,70) Width 100 Height 50

```

Оператор Set Map, в справочном руководстве MapBasic, определяется как оператор управляющий настройкой отображения объектов в окне карты. Однако функции этого оператора столь многочисленны и многообразны, что его скорее можно определить не как оператор, а как своего рода макроязык. Для изучения работы оператора полезно использовать окно MapBasic или файл Рабочего набора<sup>37</sup>.

Используемое в операторе предложение Window имеет смысл, только если число открытых окон карты больше одного. Если это предложение не используется, то все действия будут

<sup>35</sup> Включен режим или нет, значения не имеет.

<sup>36</sup> Включена подпись или нет, значения не имеет.

<sup>37</sup> Код Рабочего набора можно также получить с помощью оператора print WindowInfo (FrontWindow(), WIN\_INFO\_WORKSPACE) здесь константа WIN\_INFO\_WORKSPACE равна 14.

относиться к самому верхнему окну карты. Далее это предложение везде, где возможно, опускается, чтобы сократить запись оператора.

В последующих нескольких темах будут рассмотрены основные направления использования оператора `Set Map`.

Приводимая ниже процедура выполняет сдвиг изображения в окне карты на величины `dx` и `dy` по соответствующим осям. Основное действие выполняется оператором `Set Map Window idMap Center(x,y)`, который устанавливает точку `{x,y}` на центр окна карты.

### Код: Сдвиг карты

```
sub MoveMap(byval idMap as integer,byval dx as float,byval dy as float)
dim x,y as float
x=MapperInfo(idMap,MAPPER_INFO_CENTERX)+dx
y=MapperInfo(idMap,MAPPER_INFO_CENTERY)+dy
Set Map Window idMap Center(x,y)
end sub
```

Для сдвига окна просмотра по отношению к карте на север на 155 метров и на восток на 204 метра можно использовать следующий оператор: `set map Pan 155 Units "m" North Pan 204 Units "m" East`. Если ранее в тексте программы вы использовали оператор типа `set distance units "m"`, то уточнение `Units "m"` можно не использовать.

Ширина фрагмента карты, показанного в окне, устанавливается оператором `Set Map Zoom 200 Units "m"`. Чтобы весь слой отобразить на окно карты нужно использовать оператор типа `Set Map Zoom Entire Layer Лес`, чтобы отобразить все слои - `Set Map Zoom Entire`.

Установить масштаб для изображения в окне карты можно с помощью оператора вида `Set Map Scale 1 Units "cm" For 100 Units "m"`. В данном случае карта будет показана в масштабе 1:10000.

### Тема 43. Управление поведением карты в окне

Если размеры окна карты изменяются, то возможны два варианта формирования изображения в окне:

1. Площадь участка карты, отображаемая в окне, изменяется, масштаб изображения сохраняется. Этот режим устанавливается с помощью оператора `Set Map Preserve Scale`.
2. Участок карты остается тот же самый, а изменяется масштаб отображения. Этот режим устанавливается с помощью оператора `Set Map Preserve Zoom`.

Для программы на MapBasic единицы площади устанавливаются с помощью оператора вида `Set Area Units "sq m"`. Но в тоже время можно установить единицы измерения площади с помощью оператора вида `Set Map Area Units "hectare"`. Разницу в использовании этих операторов показывает приводимая ниже процедура.

```
sub testArea
dim ob1,ob2 as object
Set Area Units "sq m"
Set Map Area Units "hectare"
Fetch First From Query1
ob1=Query1.obj
Fetch Next From Query1
ob2=Query1.obj
print AreaOverlap(ob1,ob2)
select * from Query1 where rowid=1
run menu Command M_EDIT_GETINFO
end sub
```

В выборке Query1 имеются два пересекающихся объекта. Результат функции AreaOverlap будет получен в кв. метрах, а вот площадь участка в окне **Геоинформация** будет получена в гектарах.

Оператор Set Map Distance Units "km" определяет единицы измерения в окне **Линейка** и в информационной панели («Ширина окна») в нижнем левом углу окна MapInfo.

Оператор Set Map XY Units "km" задает единицы измерения для координат в информационной панели («Положение курсора») в нижнем левом углу окна MapInfo.

Предложение Clipping Object задает так называемый фрагмент-врезку, когда граница видимого участка карты определяется площадным объектом шаблоном. Установка врезки выполняется с помощью оператора вида Set Map Clipping Object ob1 Using Display All. В данном случае объект шаблон (ob1) это круг (Рис. 52). Операторы Set Map Clipping Off и Set Map Clipping On управляют показом врезки.

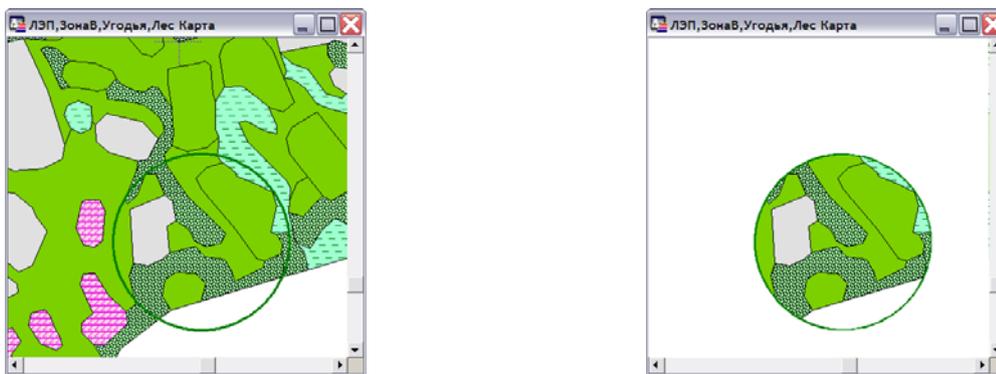


Рис. 52

Управление информационной панелью в нижнем левом углу окна MapInfo выполняется с помощью оператора Set Map Display:

Set Map Display Zoom		Активируется строка «Ширина окна».
Set Map Display Scale		Активируется строка «Масштаб карты».
Set Map Display Position		Активируется строка «Положение курсора».

Автоматической перерисовкой окна карты управляет оператор Set Map Redraw:

Set Map Redraw Off		Отменить режим автоматического обновления изображения в окне карты.
Set Map Redraw On		Восстановить режим автоматической перерисовки окна.

Для изменения порядка слоев в карте используется оператор вида Set Map Order 3, 2, 1, 4 (или Set Map Order Леса, Дороги, Угодья, Вода). В данном случае поменялись местами первый и третий слой.

#### **Тема 44. Изменение поведения отдельного слоя**

Предложение Editable используется для установки признака редактируемости слоя:

```
Set Map Layer 1 Editable On
Set Map Layer Лес Editable Off
```

Редактируемым может быть только один слой в карте. Поэтому если какой либо слой становится редактируемым, то все остальные слои будут нередатируемыми.

Предложение Selectable отвечает за установку режима доступности слоя. Оператор Set Map Layer 1 Selectable On Layer Лес Selectable Off делает слой 1 доступным, а слой **Лес** недоступным.

Если до этой операции слой **Лес** был объявлен редактируемым, то после выбора объекта на этом слое он автоматически становится доступным.

Если недоступный слой объявить редактируемым, то он становится и доступным.

Предложение `Zoom` задает пределы масштабного эффекта для слоя:

```
Set Map Layer Угодья Zoom (0,
2) Units "km" On
```

Масштабный эффект для слоя Угодья включен. Слой будет виден только для ширины окна (информационная панель в нижнем левом углу окна MapInfo) от 0 до 2 км.

```
Set Map Layer Угодья Zoom (0,
2) Units "km" Off
```

Масштабный эффект для слоя Угодья отключен.

## Тема 45. Изменение представления отдельного слоя

Операторы вида:

```
Set Map Layer 1 Nodes On Arrows On Centroids On
Set Map Layer 2 Nodes Off Arrows Off
```

управляют показом узлов, направлений линий и центроидов.

По признаку *видимость* слой может находиться в трех состояниях:

1. Быть невидимым. Установка выполняется оператором вида `Set Map Layer Лес Display Off`.
2. Слой виден, настройки для стилей оформления объектов локальные, ранее установленные<sup>38</sup>. Для переключения в этот режим используется оператор вида `Set Map Layer Лес Display Graphic`.
3. Слой виден, настройки для стилей оформления объектов глобальные, ранее установленные. Для переключения в этот режим используется оператор вида `Set Map Layer Лес Display Global`. Если, одновременно с переходом к глобальным настройкам стилей, требуется изменение этих стилей, используется оператор вида `Set Map Layer Лес Display Global Global Line(1,2,10506240) Global Pen(1,5,10506240) Global Brush(16,4523)`. Для растровых слоев — `Set Map Layer 4 Display Global contrast 51 brightness 35 alpha 204 transparency on color 8489728 grayscale off`. Здесь `contrast` и `brightness` определяются в процентах (0-100), `alpha` от 0 до 255 (от прозрачного растра до непрозрачного). Предложение `transparency on color 8489728` означает, что нужно установить цвет 8489728 прозрачным, а `grayscale off` — показать растр в цвете.

И несколько слов о настройке слоев представляющих поверхность. Хотя по сути это тот же растровый слой, но имеется возможность изменять его внешний вид. На рис. 53 показаны два вида слоя поверхность до изменения и после изменения с помощью оператора

```
set map layer 3 inflect 4 at
RGB(0,64,0):0
RGB(0,255,0):29
RGB(255,255,0):59
RGB(255,128,128):88
round 1
contrast 42 brightness 67 alpha 255
grayscale off
relief on
```

<sup>38</sup> Для растровых слоев локальные настройки определяются меню MapInfo Таблица/Растр/Подстройка изображения...

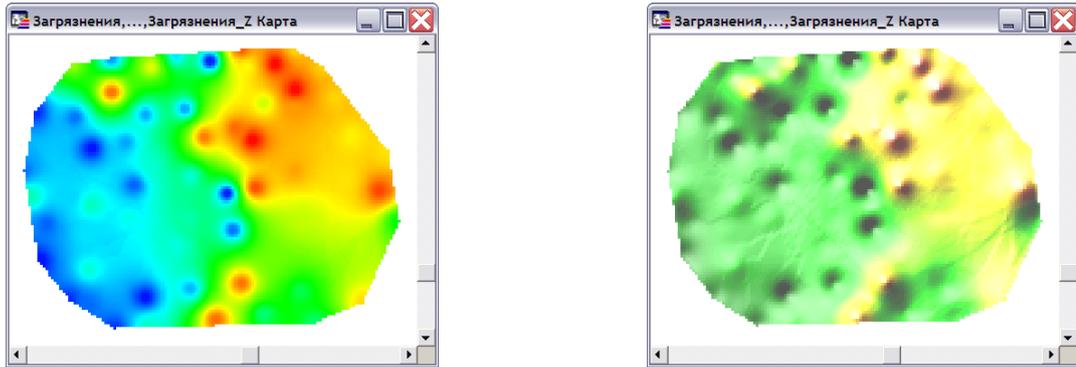


Рис. 53

### Тема 46. Изменение режима подписывания отдельного слоя

Выражения для подписи определяется предложением `With`, например `Set Map Layer 2 Label With Int(Фактор) & "%"`.

Положение подписи относительно центра объекта определяется следующими служебными словами: `Center`, `Above`, `Below`, `Left`, `Right`. Приведем примеры операторов.

<code>Set Map Layer 2 Label Position Above Right</code>	Положение подписи вверху справа.
<code>Set Map Layer 2 Label Position Below</code>	Положение подписи внизу.
<code>Set Map Layer 2 Label Position Below Left</code>	Положение подписи внизу слева.
<code>Set Map Layer 2 Label Position Center</code>	Положение подписи по центру.

Величина сдвига, в пунктах, устанавливается предложением `Offset`, например `Set Map Layer 2 Label Offset 10`.

Оператор `Set Map` с предложением `Line` устанавливает, что подпись и точка привязки подписи будут соединены прямой линией (стрелкой). Линия появляется только после сдвига подписи.

<code>Set Map Layer 2 Label Line Simple</code>	Прямая линия.
<code>Set Map Layer 2 Label Line Arrow</code>	Стрелка.
<code>Set Map Layer 2 Label Line None</code>	Без линии.
<code>Set Map Layer 2 Label Line Arrow Pen(2,3,RED)</code>	Стрелка, пунктир красного цвета толщиной 2 пикселя.

Шрифт определяет предложение `Font`, например оператор `Set Map Layer 2 Label Font MakeFont("Helvetica", 4, 12, 0, 16777215)` устанавливает для подписи шрифт *Helvetica* черного цвета на белом фоне с подчеркиванием и размером 12 пунктов.

Оператор `Set Map Layer 2 Label Auto Off` отключает показ автоматически созданных подписей, видимыми остаются только подписи, измененные вручную, например перемещенные с помощью мыши. Оператор `Set Map Layer 2 Label Auto On` вновь сделает эти подписи видимыми.

Оператор `Set Map Layer 2 Label Default` переведет все подписи в состояние автоматически созданных. Отменить действие оператора нельзя.

Оператор вида `Set Map Layer 2 Label PartialSegments On` управляет подписыванием объектов, когда центр объекта находится вне видимой области окна карты. Если установлено `On`, будет подписываться видимая часть объекта. Если `off`, то объекты будут подписываться только в том случае, если центр объекта попадает в видимую область окна карты.

Еще некоторые возможности в управлении подписями.

<code>Set Map Layer 2 Label Parallel On</code>	Строка подписи линейного объекта будет параллельна подписываемой линии.
<code>Set Map Layer 2 Label Duplicates On</code>	Разрешает дублирование подписей.

---

Set Map Layer 2 Label Overlap On	Подписи могут пересекаться.
Set Map Layer 2 Label Max 10	Устанавливает максимальное число подписей, которое MapInfo может показать на этом слое. По умолчанию ограничения на число подписей нет.
Set Map Layer 2 Label Max	Отмена ограничения на число подписей.

Для отмены соответствующего режима требуется служебное слово `off`.

## Глава 5. Работа с файлами

Из приложений на MapBasic часто приходится обращаться к файлам, которые не являются таблицами MapInfo. При этом можно говорить о двух вариантах работы с такими файлами.

1. Файлы конвертируются в таблицы MapInfo, например Register Table "C:\p22\tst1.dbf" TYPE DBF Charset "WindowsCyrillic" Into "C:\p22\tst1.TAB" или Register Table "C:\p22\tst2.txt" TYPE ASCII Delimiter 9 Charset "WindowsCyrillic" Into "C:\p22\tst2.TAB". В первом случае таблица MapInfo будет получена из таблицы *dBASE*, а во втором из текстового файла с разделителем Chr\$(9) (символ табуляции). Далее полученный *tab-файл* открывается в обычном порядке: Open Table "C:\p22\tst2.TAB" Browse \* From tst2. Можно также сохранить данные таблиц MapInfo в других форматах, например Export tstData Into "C:\p22\tstDataNew.txt" Type "ASCII" CharSet "WindowsCyrillic" Titles. Данные таблицы tstData будут сохранены в текстовом файле: разделитель полей — Chr\$(9), имена полей — в первой записи файла.
2. Файлы данных не связанные с таблицами. Для работы с такими файлами используется отдельная группа операторов ввода/вывода данных. Цикл работы с файлами включает следующие этапы: открытие файла; выполнение операций с данными файла; закрытие файла. Далее речь пойдет именно о таких файлах и работе с ними.

### Тема 47. Типы доступа к файлам

Все файлы условно можно разделить на две группы текстовые и двоичные. Существует три режима доступа к файлам: последовательный, произвольный и двоичный. Первые два применяются к текстовым файлам, третий – к двоичным. Режим доступа к файлу определяет способ перемещения по файлу и варианты чтения/записи данных. После того как файл открыт режим доступа изменен быть не может.

**Последовательный доступ.** Файл рассматривается как последовательность строк произвольной длины, разделенных специальными символами (перевод строки, возврат каретки). Чтение из файла и запись в файл производятся построчно, от первой строки и до конца файла. Данный тип доступа к файлу используется наиболее часто.

**Произвольный доступ.** В данном случае предполагается, что файл состоит из записей одинаковой фиксированной длины (включая спецсимволы). Благодаря этому становится возможным позиционирование курсора на произвольную запись по ее номеру.

**Двоичный доступ.** Можно рассматривать как частный случай произвольного доступа, где размер записи равен одному байту, и можно позиционировать курсор на произвольный байт по его номеру в файле. Количество считанных/записанных байт определяется типом переменной указанной в операторах Get/Put. Обычно представляет интерес как способ хранения большого объема числовых данных, так как двоичное представление является для них наиболее компактным.

### Тема 48. Работа с файлами последовательного доступа

Разберем работу с файлами в режиме последовательного доступа.

Построим процедуру записи информации в файл.

#### Код: Запись информации в файл

```
Sub tstSerialAccessWrite(x() as float,y() as float)
dim pp as string
dim i as integer
```

```
pp=FileSaveAsDlg("C:\", "", "TXT", "Сохранить файл данных")
if Len(pp)=0 then Exit sub end if
OnError GoTo theError
Open File pp For Output Access Write as #1 CharSet "WindowsCyrillic"
For i=1 to ubound(x)
    Write #1, x(i),y(i)
Next
theError:
Close File #1
end sub
```

Определим, тем или иным способом, полное имя записываемого файла. В данном случае имя файла определим через диалог (функция FileSaveAsDlg). И далее, если имя корректно, откроем (создадим) файл для записи. Используя оператор Write, перепишем массивы x и y в файл. Закроем сформированный файл. На рис. 54а показан вид фрагмента этого файла открытого в Блокноте (notepad.exe).

Если заменить, в данной процедуре, строку Write #1, x(i),y(i) на Write #1, Format\$(x(i), "#.#0"), Format\$(y(i), "#.#0"), то есть перейти к записи строк, то результат будет иметь вид, показанный на рис. 54б. Если строку Write #1, x(i),y(i) заменить на Print #1, Format\$(x(i), "#.#0") & Chr\$(9) & Format\$(y(i), "#.#0") то результат будет иметь вид, показанный на рис. 54в.

3207526.94,360069.3	"3207526.94","360069.30"	3207526.94	360069.30
3207518.87,360184.36	"3207518.87","360184.36"	3207518.87	360184.36
3207684.4,360190.42	"3207684.40","360190.42"	3207684.40	360190.42
3207728.81,360103.62	"3207728.81","360103.62"	3207728.81	360103.62
3207882.22,360099.58	"3207882.22","360099.58"	3207882.22	360099.58
3207900.39,360192.44	"3207900.39","360192.44"	3207900.39	360192.44
3208082.06,360196.48	"3208082.06","360196.48"	3208082.06	360196.48
a	б	в	

Рис. 54

Перейдем к чтению информации из файла. Общая схема работы остается той же.

Определяем имя файла, открываем файл, выполняем некоторые действия по считыванию информации, закрываем файл.

В представленной ниже процедуре tstSerialAccessLineInput выполняется чтение файла с помощью оператора Line Input, то есть каждая запись считывается в виде одной строки.

#### Код: Чтение информации из файла (1)

```
Sub tstSerialAccessLineInput
dim pp,str as string
dim i as integer
pp=FileOpenDlg("C:\", "", "TXT", "Открыть файл данных")
if Len(pp)=0 then Exit sub end if
OnError GoTo theError
Open File pp For Input Access Read as #1 CharSet "WindowsCyrillic"
i=1
Do
    Line Input #1,str
    if EOF(1) then exit do end if
    print str$(i) & " | " & str
    i=i+1
Loop
theError:
Close File #1
end sub
```

На рис. 55 показано как будет выглядеть результат чтения для ранее сформированных файлов.

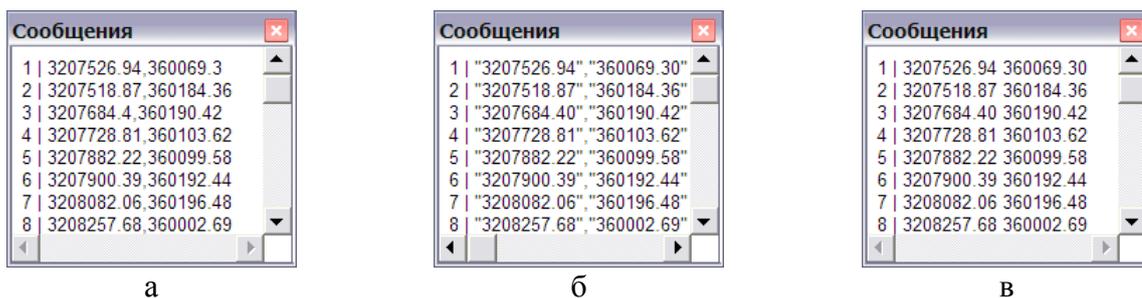


Рис. 55

Если для чтения используется оператор `Input`, как в процедуре `tstSerialAccessInput`, то результат чтения будет иметь вид, показанный на рис. 56а. Обратите внимание что в случае *б* и *в* получаемый результат некорректен (в одном случае все нули, в другом из файла считывается только значение `x`).

**Код: Чтение информации из файла (2)**

```
Sub tstSerialAccessInput
dim pp as string
dim xx,yy as float
dim i as integer
pp=FileOpenDlg("C:\", "", "TXT", "Открыть файл данных")
if Len(pp)=0 then Exit sub end if
OnError GoTo theError
Open File pp For Input Access Read as #1 CharSet "WindowsCyrillic"
i=1
Do
    Input #1, xx,yy
    if EOF(1) then exit do end if
    print str$(i) & " | " & str$(xx) & " | " & str$(yy)
    i=i+1
Loop
theError:
Close File #1
end sub
```

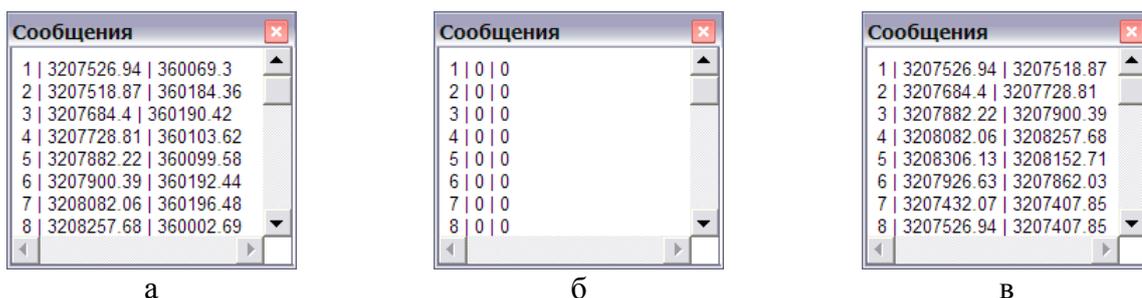


Рис. 56

**Тема 49. Работа с файлами произвольного доступа**

Рассмотрим коды процедур записи (`tstRundomAccessPutFloat`) и чтения (`tstRundomAccessGetFloat`) числовых данных в режиме произвольного доступа. Так как предполагается работа с числами типа *Float*, длина записи в операторах открытия файла установлена равной 8 байт. В операторах `Put` и `Get` номер записи не указывается, потому что выполняется последовательная запись и считывание всего объема информации.

**Код: Запись информации в файл (Float)**

```

Sub tstRundomAccessPutFloat(x() as float,y() as float)
dim pp as string
dim i as integer
pp=FileSaveAsDlg("C:\", "", "TXT", "Сохранить файл данных")
if Len(pp)=0 then Exit sub end if
OnError GoTo theError
Open File pp For Random Access Write as #1 Len=8
For i=1 to ubound(x)
    Put #1,,x(i)
    Put #1,,y(i)
Next
theError:
Close File #1
end sub

```

**Код: Чтение информации из файла (Float)**

```

Sub tstRundomAccessGetFloat
dim pp as string
dim xx,yy as float
dim i as integer
pp=FileOpenDlg("C:\", "", "TXT", "Открыть файл данных")
if Len(pp)=0 then Exit sub end if
OnError GoTo theError
Open File pp For Random Access Read as #1 Len=8
i=1
Do
    Get #1,,xx
    if EOF(1) then exit do end if
    Get #1,,yy
    print str$(i) & " | " & str$(xx) & " | " & str$(yy)
    i=i+1
Loop
theError:
Close File #1
end sub

```

На рис. 57а показано содержимое файла сформированного процедурой `tstRundomAccessPutFloat`. Из картинки видно, что полученный файл двоичный. Результат чтения этого файла с помощью процедуры `tstRundomAccessGetFloat` показан на рис. 57б.

Однако основное отличительное свойство режима произвольного доступа это возможность считывать записи в произвольном порядке. В основном этот тип доступа к файлу следует использовать в случаях, когда имеется большой файл данных, а для конкретной процедуры из этого файла нужны лишь несколько записей. Для считывания отдельной записи из открытого файла можно использовать функцию `GetFloatValue`. Данная функция возвращает значение типа *Float* (одна из записанных координат) по аргументам номер записи (*jj*) и номер файла (*nf*).

**Код: Чтение отдельной записи (Float)**

```

Function GetFloatValue(byval jj as integer,byval nf as smallint) as float
dim vv as float
if jj>0 then
    Get #nf,jj,vv
    if EOF(nf) then
        GetFloatValue=-999999
    else
        GetFloatValue=vv
    end if
else

```

```

        GetFloatValue=-999999
end if
end function

```

Обращение к функции GetFloatValue может иметь вид

```

print GetFloatValue (2,1)
print GetFloatValue (29,1)
print GetFloatValue (30,1)
print GetFloatValue (31,1)
print GetFloatValue (32,1)
print GetFloatValue (8,1)
print GetFloatValue (-5,1)

```

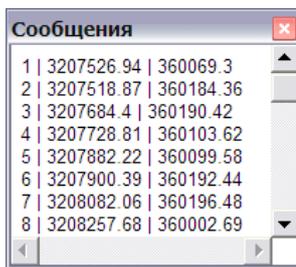
Результаты работы этого кода показаны на рис. 57в.

```

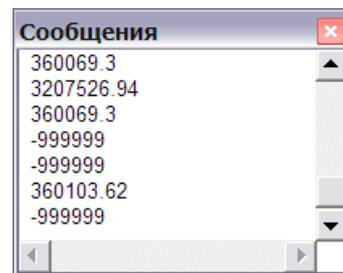
ЦВххixНА...лQ8тЪПА'н|oixНАЦВu
быПAE
°2пуНАsuПщыПАпп+gпуНА<lз(ъъ
ПAGx(пeyНА"ппVГъПАЗы©1пуНА
)\ЦВПЪПАkпПЙуНА!°риПЪПАq=ПЧ
zНАПГ•ГпщПАНьД
П9zНАЭ$ПГЛЪхПАeP"[myНА°rhП=y
ПАП
ЧJP{yНАль™™™™OfПA~ПZП[yНА
L7%oАмцПА+±П
,xНАпп$ЖнцПА~СmwxНАуЕПйуцщП
АЦВххixНА...лQ8тЪПА

```

а



б



в

Рис. 57

Работа со строками в режиме произвольного доступа имеет некоторые отличия.

Рассмотрим процедуру `tstRundomAccessPutString` записи в файл текстовой информации. В качестве тестовой информации используется массив `tstStrings` содержащий 10 строк, каждая строка состоит из 19 случайным образом выбранных символов ASCII<sup>39</sup> (коды 65-90) плюс номер строки.

Обратите внимание, что длина записи в операторе `Open File` равна 21 байт, та же длина установлена для строковой переменной `ss`. Это связано с тем, что кроме самой строки (20 байт) нужно учитывать символ конца строки.

**Код: Запись информации в файл (String)**

```

Sub tstRundomAccessPutString(tstStrings() as string)
dim pp as string
dim ss as string*21
dim i as integer
pp=FileSaveAsDlg("C:\", "", "TXT", "Сохранить файл данных")
if Len(pp)=0 then Exit sub end if
OnError GoTo theError
Open File pp For Random Access Write as #1 Len=21 CharSet "WindowsCyrillic"
For i=1 to ubound(tstStrings)
    ss=tstStrings(i)
    Put #1,,ss
Next
theError:
Close File #1
end sub

```

Содержимое файла сформированного этой процедурой показано на рис. 58а. Как видно из рисунка мы имеем текстовый файл. Этот файл можно полностью прочитать с помощью процедуры `tstRundomAccessGetString`. Результат этого чтения показан на рис. 58б.

<sup>39</sup> ASCII - American Standard Code for Information Interchange

**Код: Чтение информации из файла (String)**

```

Sub tstRandomAccessGetString
dim pp as string
dim ss as string*21
dim i as integer
pp=FileOpenDlg("C:\","",".TXT","Открыть файл данных")
if Len(pp)=0 then Exit sub end if
OnError GoTo theError
Open File pp For Random Access Read as #1 Len=21 CharSet "WindowsCyrillic"
i=1
Do
    Get #1,,ss
    if EOF(1) then exit do end if
    print str$(i) & " | " & ss
    i=i+1
Loop
theError:
Close File #1
end sub

```

Для чтения отдельной записи можно использовать функцию GetStringValue.

**Код: Чтение отдельной записи (String)**

```

Function GetStringValue(byval jj as integer,byval nf as smallint) as string
dim ss as string*21
if jj>0 then
    Get #nf,jj,ss
    if EOF(nf) then
        GetStringValue="Error"
    else
        GetStringValue=ss
    end if
else
    GetStringValue="Error"
end if
end function

```

Обращение к функции GetStringValue может иметь вид

```

print GetStringValue(2,1)
print GetStringValue(4,1)
print GetStringValue(30,1)
print GetStringValue(8,1)
print GetStringValue(-5,1)

```

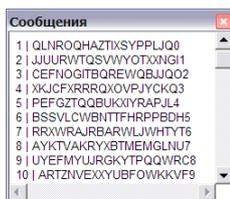
результат этих обращений показан на рис. 58в.

```

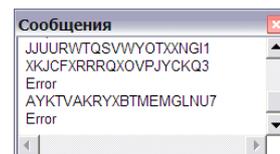
QLNQOHAZTIXSYPLJQ0
JJUURWTQSVWYOTXXNG1
CEFNQGITBQREWQBJJQ02
XKJCFXRRRQXOVPIJYCKQ3
PEFGZTQQBUKXIYRAPJL4
BSSVLCWBNTTFHRPPBDH5
RRXWRAJRBARWLJWHTYT6
AYKTVAKRYXBTMEMGLNU7
UYEFMYUJRGKYTPQQWRC8
ARTZNVEXXYUBFOWKKVF9

```

а



б



в

Рис. 58

**Тема 50. Работа с файлами в режиме двоичного доступа**

Если работа выполняется с записями равной длины, то работа в режиме двоичного доступа мало чем отличается от режима произвольного доступа. И для этой работы можно

использовать те же процедуры `tstRundomAccessPutFloat`, `tstRundomAccessGetFloat`, `tstRundomAccessPutString`, `tstRundomAccessGetString` лишь отредактировав, в них оператор `Open File`, заменив в нем тип доступа `Random` на `Binary` и убрав лишние в данном случае предложения `Len` и `CharSet`.

Принципиальное отличие режима двоичного доступа в том, что он позволяет работать с записями различной длины. Рассмотрим на примере как можно организовать эту работу.

Пусть имеется массив значений со структурой следующего вида

```
Type pntInfo
    x as float
    y as float
    num as integer
    comment as string*15
End Type
```

Тогда код для записи информации из этого массива в файл может иметь следующий вид.

### Код: Запись информации в файл

```
Sub tstBinaryAccessPut(pnt() as pntInfo)
    dim pp as string
    dim i as integer
    pp=FileSaveAsDlg("C:\", "", "DAT", "Сохранить файл данных")
    if Len(pp)=0 then Exit sub end if
    OnError GoTo theError
    Open File pp For Binary Access Write as #1
    For i=1 to ubound(Points)
        Put #1,,pnt(i).x
        Put #1,,pnt(i).y
        Put #1,,pnt(i).num
        Put #1,,pnt(i).comment
    Next
theError:
    Close File #1
end sub
```

Содержимое двоичного файла сформированного этой процедурой показано на рис. 59а. для чтения этого файла целиком можно использовать процедуру `tstBinaryAccessGet`. Результат работы этой процедуры представлен на рис. 59б.

### Код: Чтение информации из файла

```
Sub tstBinaryAccessGet
    dim pp as string
    dim x,y as float
    dim ss as string*15
    dim i,k as integer
    pp=FileOpenDlg("C:\", "", "DAT", "Открыть файл данных")
    if Len(pp)=0 then Exit sub end if
    OnError GoTo theError
    Open File pp For Binary Access Read as #1
    i=1
    Do
        Get #1,,x
        if EOF(1) then exit do end if
        Get #1,,y
        Get #1,,k
        Get #1,,ss
        print str$(i) & " | " & str$(x) & " | " & str$(y) & " | " &
str$(k) & " | " & ss
    Loop
```

```

        i=i+1
Loop
theError:
Close File #1
end sub

```

Для чтения отдельной записи предназначена функция `GetBinaryValue`. Здесь под записью понимается отдельный элемент исходного массива `pnt` со структурой `pntInfo`. Функция имеет аргументы: `jj` – номер записи, `nf` – номер файла. Возвращаемое значение это строка, которая включает номер записи, значения координат ( $x$  и  $y$ ), номер, комментарий.

**Код: Чтение отдельной записи**

```

Function GetBinaryValue(byval jj as integer,byval nf as smallint) as string
dim ss as string*15
dim x,y as float
dim i,k as integer
if jj>0 then
    i=(8+8+4+15)*(jj-1)+1
    Get #nf,i,x
    if EOF(nf) then
        GetBinaryValue="Error"
    else
        Get #nf,,y
        Get #nf,,k
        Get #nf,,ss
        GetBinaryValue=str$(jj) & " | " & str$(x) & " | " & str$(y) &
            " | " & str$(k) & " | " & ss
    end if
else
    GetBinaryValue="Error"
end if
end function

```

В этом коде следует обратить внимание на вычисление значения `i`, номера первого байта записи `jj`. В этом выражении специально подробно расписано вычисление общей длины записи.

К функции можно обращаться следующим образом

```

print GetBinaryValue(1,1)
print GetBinaryValue(2,1)
print GetBinaryValue(10,1)
print GetBinaryValue(15,1)
print GetBinaryValue(-5,1)

```

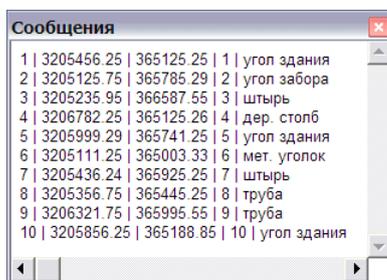
Результат этих обращений представлен на рис. 59в.

```

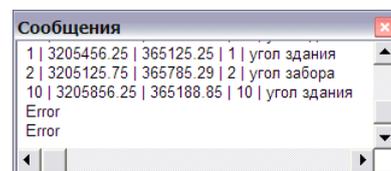
    ЁтНА  ппАп  угол здания
аптНАЦВх(еSтАп  угол забора
ль™™щ9тНА3333о_пАп  штырь
?wНАрр=пппАп  дер. столб  RëпГ·иНА
µRтАп  угол здания
ыsНАп...лQ-GтАп  мет. уголок
мQëптНА  *УтАп  штырь  `vtНА
пNпАп  труба  аXvНА3333@VтА
труба  puНАffffпАп  угол
здания

```

а



б



в

Рис. 59

### Тема 51. Операции с файлами и каталогами

При работе с файлами, кроме использованных выше операторов, также используются:

FileAttr	Функция возвращает информацию об открытом файле. Тип информации определяется кодом в списке аргументов.
FileExists	Функция возвращает логическое значение (TRUE/FALSE), в зависимости от того, существует файл или нет. Аргумент полное имя файла.
Kill	Оператор физически удаляет файл. Отменить нельзя. Файл не должен быть открыт для доступа.
LOF	Функция возвращает длину открытого файла в байтах. Аргумент номер файла.
Rename File	Оператор переименовывает файл. Файл не должен быть открыт для доступа. Обратите внимание, что переименовывается полное имя файла, и если путь к файлу изменился, то файл будет перенесен по новому адресу.
Save File	Оператор копирует файл с новым именем. Файл не должен быть открыт для доступа. При использовании ключевого слова Append происходит добавление одного файла к другому.
Seek	Оператор устанавливает доступ к содержимому открытого файла, начиная с позиции, номер которой указан как параметр оператора. Для режима RANDOM это номер записи, для других режимов - номер байта.
Seek	Функция возвращает текущую позицию в открытом файле для операций ввода/вывода. Если файл был открыт в режиме RANDOM, функция вернет номер записи, которая будет прочитана или записана. Для других режимов, функция вернет номер байта, с которого начнется чтение или запись следующей операцией ввода/вывода.

Файлы организованы в каталоги (папки, директории) формирующие систему хранения файлов на запоминающем устройстве. Для работы с ними в MapBasic используются следующие функции:

ApplicationDirectory	Возвращает полный путь, для файла выполняющегося приложения (приложения в котором выполнено обращение к данной функции).
ProgramDirectory	Возвращает путь, по которому была установлена рабочая версия MapInfo.
PathToDirectory	Возвращает путь к файлу извлеченный из полного имени файла. Функция PathToDirectory\$("C:\Temp\pr1.txt") вернет строку "C:\Temp\". Функция не проверяет наличие файла на диске.
PathToFileName	Извлекает из полного имени файла только имя файла с расширением. Функция PathToFileName\$("C:\Temp\pr1.txt") вернет строку "pr1.txt". Функция не проверяет наличие файла на диске.
PathToTableName	Извлекает из полного имени таблицы только имя таблицы <sup>40</sup> . Функция PathToTableName\$("C:\Temp\Points.TAB") вернет строку "Points".
TempFileName	Возвращает полное уникальное имя, которое можно использовать при создании временного файла. Функция TempFileName\$("C:\Temp") вернет строку следующего вида "C:\Temp\~MAP0004.TMP".
LocateFile	Возвращает путь к одному из служебных файлов MapInfo. Конкретный результат определяется кодом аргументом. Например, функция LocateFile\$(LOCATE_PRJ_FILE) вернет строку следующего вида "C:\Program Files\MapInfo\Professional\MAPINFOW.PRJ" (полное имя файла проекций MapInfo). Список соответствующих кодов можно найти в файле MAPBASIC.DEF.

---

<sup>40</sup> Работает и для произвольного расширения файла.

GetFolderPath

Возвращает путь к специальным папкам MapInfo и Windows. Конкретный результат определяется кодом аргументом. Например, функция GetFolderPath\$(FOLDER\_MYDOCS) вернет путь к системной папке "Мои документы". Список соответствующих кодов можно найти в файле MAPBASIC.DEF.

## Глава 6. Программы

### Тема 52. Формирование баланса территорий

#### Постановка задачи

При проектировании генпланов различных объектов, в землеустройстве, лесном хозяйстве и т.п. требуется определение баланса территорий. Баланс территорий отражает существующее или проектируемое деление территории по различным, в зависимости от назначения баланса, критериям.

Проектируемая программа должна упростить процедуру формирования баланса территорий, а также представлять необходимые отчетные материалы. Так как у нас нет прототипа отчета, будем считать, что результатом будут материалы для отчета. Это также позволит уменьшить размер программного кода.

В качестве исходных данных примем, построенные в MapInfo на некотором слое, множество непересекающихся объектов типа область. Например, земельные участки с различными типами угодий (пашня, пастбище, сенокос и т.д.) или участки леса с установленным классом пожарной опасности и т.п. Кроме баланса территорий представленного в единицах площади должна быть возможность сравнивать балансовые группы по некоторым оценочным показателям производным от площади и учитывающим индивидуальность каждого участка.

Материалы, которые мы хотим иметь в качестве результата:

- ✓ План территорий
- ✓ Тематический план территорий.
- ✓ Диаграммы, отображающие баланс территорий в единицах площади, в оценочных показателях, в штуках.
- ✓ Текстовый файл баланса территорий для использования при подготовке окончательного отчета.

#### Проект

Сразу определимся, что пользовательский интерфейс программы будет представлять собой инструментальную панель с набором кнопок. За каждой кнопкой будет закреплен некоторый объем полезной работы.

Определим, какие функциональные блоки необходимы:

- ✓ Настройка параметров программы.
- ✓ Присвоение кодов группам участков.
- ✓ Построение баланса.

В качестве дополнительных блоков можно назвать:

- ✓ Формирование новой таблицы с заданной структурой.
- ✓ Управление закрытием программы.

Таким образом, мы разделили задачу на отдельные блоки. С каждым блоком будет связана отдельная кнопка на инструментальной панели. Определим, как установленные блоки должны функционировать.

#### Настройка параметров программы

Определим, какие подзадачи нужно решить до перехода к основному вопросу определения баланса.

- ✓ Установить наличие окна карты.
- ✓ Определить слой, с которым нужно работать.
- ✓ Получить информацию о кодах для разных групп участков (классификатор).
- ✓ Определить рабочую единицу площади (кв. км, гектар, кв. м).

- ✓ Установить состав отчетных материалов (построить диаграммы, сформировать тематический слой, подготовить файл баланса).

Классификатор будет оформлен в виде текстового файла состоящего из строк следующего вида: код, наименование. Такой файл легко создать и при необходимости можно иметь набор таких файлов.

### **Присвоение кодов группам участков**

Реализуемое в данном блоке действие можно определить следующим образом: выбранной группе участков присваивается выбранный из списка код.

### **Построение баланса**

Это основной блок программы. Заполняем базовую таблицу результатами вычисления площади и значением оценки. Для вычисления площади используем функцию MapBasic Area<sup>41</sup>. Далее формируется таблица баланса.

В соответствии с настройками программы построить тематический слой, диаграммы и сохранить файл баланса.

### **Формирование новой таблицы с заданной структурой**

Сформировать новую таблицу со следующей структурой: *Cod {SmallInt}, Name {Char(20)}, Area {Float}, Coeff {Float}, Value {Float}*. Границы рабочей области и стили оформления задать через диалог. Открыть новую таблицу в окне карты.

### **Код**

Так как разрабатываемая программа небольшая по объему и по логике, скорее простая чем сложная, то весь код будет организован в виде одного модуля.

### **Блок описаний**

Определим подключаемые файлы заголовков.

```
Include "MAPBASIC.DEF"  
Include "MENU.DEF"  
Include "ICONS.DEF"
```

Опишем пользовательский тип данных. Он соответствует строке классификатора.

```
Type DatInfo  
    cod as smallint  
    name as string  
End Type
```

Затем следует декларирование процедур используемых в программе.

```
Declare Sub Main  
Declare Sub zb_NewTable  
Declare Sub zb_Settings  
Declare Sub zb_SetCode  
Declare Sub zb_CreateBalance  
Declare Sub zb_Exit  
Declare function ItsMap() as integer  
Declare Sub CreateListLayerNames(byval id as smallint)
```

---

<sup>41</sup> Для получения более точных результатов лучше организовать вычисление площади через координаты узлов полигонов.

```

Declare Sub LoadDat
Declare Sub SetTableName
Declare Sub CreateTableMap(ByVal FullNameTable As String,sCols() as string,
    byval minX as float,byval maxX as float,
    byval minY as float,byval maxY as float,idmap as integer)
Declare sub CopyArray(sender() as DatInfo,recipient() as DatInfo)
Declare function IsSelectionObjectInLayer(byval nameLayer as string)
as integer

```

Здесь

Main	Главная процедура.
zb_NewTable	Процедура для кнопки «Создать таблицу»
zb_Settings	Процедура для кнопки «Настройки»
zb_SetCode	Процедура для кнопки «Присвоить коды»
zb_CreateBalance	Процедура для кнопки «Определить баланс»
zb_Exit	Процедура для кнопки «Выход»
ItsMap	Функция проверяет наличие активного окна карты и возвращает идентификатор этого окна или ноль.
CreateListLayerNames	Формирует список слоев для окна карты idMap. Тип отбираемых в список слоев определяется параметром процедуры.
LoadDat	Загружает данные из классификатора
SetTableName	Задаёт полное имя для новой таблицы.
CreateTableMap	Формирует новую таблицу и открывает ее в окне карты idMap или если idMap=0 в новом окне.
CopyArray	Делает копию массива со структурой DatInfo.
IsSelectionObjectInLayer	Функция проверяет наличие выбранных объектов на заданном слое. Возвращает число выбранных объектов.

Далее следует описание общих переменных уровня модуля.

```

dim idMap as integer
dim iLayer,iAreaUnits as smallint
dim isDiagram,isThematic,isFile as Logical
dim listLayers() as string
dim Cods() as DatInfo
dim nmLayer,SelName,NewTableName,UnitNm as string

```

### Главная процедура

Главная процедура начинается с инициализации переменных для диалога **Настройки** и установки «бумажных» единиц. Далее следует код, отвечающий за построение инструментальной панели. Вид сформированной панели показан на рис. 60.

```

Sub Main
iLayer=1
isDiagram="F"
isThematic="F"
isFile="T"
iAreaUnits=2
Set paper Units "mm"
Create ButtonPad "Баланс" as
    PushButton
        Icon MI_ICON_NEW_DOC
        Calling zb_NewTable
        ID 2001
        HelpMsg "Создать новую таблицу территорий\nСоздать таблицу"
        Enable
    Separator
    PushButton

```

```

        Icon MI_ICON_LEGEND
        Calling zb_Settings
        ID 2001
        HelpMsg "Начальные установки для программы\nНастройки"
        Enable
    PushButton
        Icon MI_ICON_ARROW_4
        Calling zb_SetCode
        ID 2002
        HelpMsg
        "Для выделенных областей установить код территории \nПрисвоить коды"
        Enable
    PushButton
        Icon MI_ICON_DISTRICT_MANY
        Calling zb_CreateBalance
        ID 2003
        HelpMsg "Получить баланс территорий\nОпределить баланс"
        Enable
    Separator
    PushButton
        Icon MI_ICON_ARROW_17
        Calling zb_Exit
        ID 2004
        HelpMsg "Закрыть программу\nВыход"
        Enable
    Width 7
    Position (0,40)
    Show
    Float
end sub

```



Рис. 60

### **Процедура настройки параметров программы**

В процедуре выполняется следующая последовательность операций:

- ✓ Проверка наличия активного окна карты (функция *ItsMap*).
- ✓ Создание списка нормальных слоев для активной карты (процедура *CreateListLayerNames*).
- ✓ Создание временной копии массива *Cods* (процедура *CopyArray*). Для обеспечения возможности восстановления оригинального классификатора в случае отката, если пользователь нажал кнопку **Cancel**.
- ✓ Создание пользовательского диалога **Настройки**.

В диалоге должны быть определены следующие параметры:

- ✓ Имя рабочего слоя (непосредственно определяется индекс *iLayer* для списка слоев *listLayers*).
- ✓ Определяется классификатор и его данные загружаются в массив *Cods* (процедура *LoadDat*).
- ✓ Локальная переменная *isClear* (если значение равно *TRUE*, то все поля таблицы рабочего слоя очищаются).
- ✓ Общая переменная уровня модуля *isDiagram* (если значение равно *TRUE*, то в процессе решения будут созданы диаграммы баланса территорий).
- ✓ Общая переменная уровня модуля *isThematic* (если значение равно *TRUE*, то в процессе решения будет создан тематический слой для рабочей таблицы).
- ✓ Общая переменная уровня модуля *isFile* (если значение равно *TRUE*, то в процессе решения будет создан текстовый файл баланса).

- ✓ Определяется единица измерения площади (общая переменная уровня модуля UnitNm определяемая через индекс iAreaUnits для списка единиц площади).

Вид формы диалога показан на рис. 61а.

```

Sub zb_Settings
'Настройки
dim _Cods() as DatInfo
dim isClear as Logical
idMap=ItsMap()
if idMap=0 then
    exit sub
end if
call CreateListLayerNames(LAYER_INFO_TYPE_NORMAL)
call CopyArray(Cods,_Cods)
Dialog
    Title "Настройки"
    Width 197
    Height 155
    Control ListBox
        Title "ListBox"
        Width 87
        Height 105
        Position 7, 14
        Title From Variable listLayers
        Value iLayer
        Into iLayer
    Control StaticText
        Title "Имя слоя"
        Width 39
        Height 10
        Position 8, 1
    Control Button
        Title "Данные"
        Width 43
        Height 17
        Position 7, 134
        Calling LoadDat
    Control CheckBox
        Title "удалить семантику"
        Width 122
        Height 12
        Position 102, 77
        Value isClear
        Into isClear
    Control CheckBox
        Title "показать диаграмму"
        Width 104
        Height 12
        Position 102, 90
        Value isDiagram
        Into isDiagram
    Control CheckBox
        Title "тематический слой"
        Width 96
        Height 12
        Position 102, 103
        Value isThematic
        Into isThematic
    Control CheckBox
        Title "файл баланса"
        Width 96
        Height 12

```

```

        Position 102, 115
        Value isFile
        Into isFile
    Control OkButton
        Width 37
        Height 17
        Position 104, 134
    Control CancelButton
        Width 37
        Height 17
        Position 153, 134
    Control GroupBox
        Title "Ед. площади"
        Position 102, 12
        Width 86
        Height 62
    Control RadioGroup
        Title "кв. километр; гектар; кв. метр"
        Width 71
        Height 10
        Position 108, 30
        Value iAreaUnits
        Into iAreaUnits
If CommandInfo(CMD_INFO_DLG_OK) Then
    nmLayer=listLayers(iLayer)
    Do Case iAreaUnits
        Case 1
            Set Area Units "sq km"
            UnitNm="sq km"
        Case 2
            Set Area Units "hectare"
            UnitNm="hectare"
        case 3
            Set Area Units "sq m"
            UnitNm="sq m"
        Case Else
            Set Area Units "sq m"
            UnitNm="sq m"
    End Case
    if isClear then
        Update nmLayer Set Cod=0,Name="",Area=0,Coeff=1,Value=0
    end if
else
    call CopyArray(_Cods,Cods)
End If
end sub

```

**Функция ItsMap** возвращает идентификатор активного окна карты или ноль, если нет активной карты.

```

function ItsMap() as integer
dim map_id as integer
    ItsMap=0
    map_id = FrontWindow()
    If WindowInfo(map_id , WIN_INFO_TYPE) <> WIN_MAPPER Then
        Note "Окно карты должно быть активным!"
        Exit function
    End If
    ItsMap=map_id
end function

```

**Процедура CreateListLayerNames** создает список слоев заданного типа определяемого формальным параметром id.

```

Sub CreateListLayerNames (byval id as smallint)
Dim s As String
Dim j, n, m As Integer
Dim t As Integer
ReDim ListLayers(0)
t=0
n=MapperInfo(idMap,MAPPER_INFO_LAYERS)
  For j=0 To n
    s = LayerInfo(idMap,j,LAYER_INFO_NAME)
    m =LayerInfo(idMap,j,LAYER_INFO_TYPE)
    If m=id Then
      t =t+1
      ReDim ListLayers(t)
      ListLayers(t) = s
    End If
  Next
End Sub

```

Процедура CopyArray создает копию массива sender в переменной recipient. Оба массива имеют структуру DatInfo.

```

sub CopyArray(sender() as DatInfo,recipient() as DatInfo)
dim n,i as integer
n=Ubound(sender)
if n=0 then exit Sub end if
redim recipient(n)
for i=1 to n
  recipient(i).cod=sender(i).cod
  recipient(i).name=sender(i).name
next
end sub

```

Процедура чтения данных из файла классификатора. Файл имеет расширение *DAT*. Пример содержимого файла показан на рис. 61б.

```

Sub LoadDat
dim pp,name as string
dim i,jj as smallint
pp=FileOpenDlg("C:\","",".DAT","Открыть файл данных")
if Len(pp)=0 then Exit sub end if
OnError GoTo theError
Open File pp For Input Access Read as #1 CharSet "WindowsCyrillic"
i=1
Do
  Input #1, jj,name
  if EOF(1) then exit do end if
  i=i+1
Loop
redim Cods(i-1)
i=1
Seek #1,1
Do
  Input #1, jj,name
  if EOF(1) then exit do end if
  Cods(i).cod=jj
  Cods(i).name=name
  i=i+1
Loop
theError:
Close File #1
End Sub

```

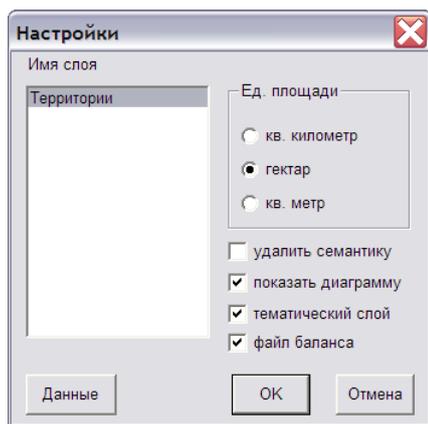
Данный код вполне можно заменить на следующий

```

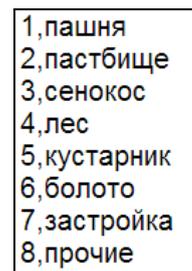
i=1
Do
  Input #1, jj,name
  if EOF(1) then exit do end if
  redim Cods(i)
  Cods(i).cod=jj
  Cods(i).name=name
  i=i+1

```

Просто в используемом варианте показано, как можно заново прочитать тот же файл при последовательном доступе.



а



б

Рис. 61

### Процедура присвоения кодов

На входе в процедуру выполняется блок проверок:

- ✓ Есть ли в рабочей таблице объекты или другими словами не пустая ли она?
- ✓ Есть ли в рабочей таблице выделенные объекты?
- ✓ Загружен ли классификатор?

Далее формируется диалог **Присвоить коды**. Общий вид диалога показан на рис. 62. В диалоге, из списка, выбирается тип территории который, при выходе из диалога по кнопке **Ok**, будет присвоен всем выделенным объектам. Если при этом отмечен переключатель «Поле Coeff по умолчанию», то в поле *Coeff* для выделенных объектов будет вставлена единица.

Здесь хотелось бы несколько слов сказать о назначении поля *Coeff*. Значения в этом поле позволяют сформировать баланс территории не только в гектарах или кв. метрах, но и в других показателях отражающих некоторые полезные характеристики рассматриваемой группы объектов. Например, для случая с угодьями в поле *Coeff* можно внести стоимость гектара для каждого участка. Тогда в поле Value будет величина, отражающая стоимость участка и в итоге получим баланс территорий по стоимости.

```
Sub zb_SetCode
'Присвоить коды
dim i,n,m,iType as integer
dim NamesType() as string
dim isCoeff as Logical
onerror goto theErrrSetCode
n=TableInfo(nmLayer,TAB_INFO_NROWS)
if n=0 then
    note "В таблице " & nmLayer & " нет объектов!"
    exit sub
end if
n=IsSelectionObjectInLayer(nmLayer)
if n=0 then
    note "В таблице " & nmLayer & " нет выделенных объектов!"
    exit sub
end if
SelName=SelectionInfo(SEL_INFO_SELNAME)
iType=1
isCoeff="T"
m=Ubound(Cods)
if m=0 then
    note "Не определены типы территорий!"
    exit sub
end if
redim NamesType(m)
```

```

for i=1 to m
    NamesType(i)=Cods(i).name
next
Dialog
    Title "Присвоить коды"
    Width 133
    Height 175
    Control StaticText
        Title "Тип"
        Width 15
        Height 10
        Position 7, 6
    Control ListBox
        Title "ListBox"
        Width 117
        Height 107
        Position 7, 19
        Title From Variable NamesType
        Value iType
        Into iType
    Control CheckBox
        Title "Поле Coeff по умолчанию"
        Width 121
        Height 12
        Position 7, 133
        Value isCoeff
        Into isCoeff
    Control OkButton
        Width 37
        Height 16
        Position 45, 154
    Control CancelButton
        Width 37
        Height 16
        Position 89, 154
If CommandInfo(CMD_INFO_DLG_OK) Then
    if isCoeff then
        Update SelName Set
            Cod=Cods(iType).cod,Name=Cods(iType).name,Coeff=1
    else
        Update SelName Set Cod=Cods(iType).cod,Name=Cods(iType).name
    end if
End If
Exit sub
theErrSetCode:
note Error$()
end sub

```

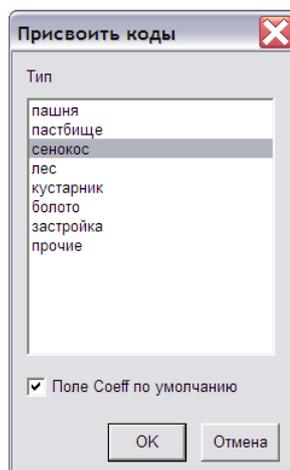


Рис. 62

Функция `IsSelectionObjectInLayer` возвращает число выделенных объектов на заданном слое.

```
function IsSelectionObjectInLayer(byval nameLayer as string) as integer
dim n as integer
n=SelectionInfo(SEL_INFO_NROWS)
IsSelectionObjectInLayer=n
if SelectionInfo(SEL_INFO_TABLENAME) <> nameLayer then
    IsSelectionObjectInLayer=0
end if
End function
```

### **Процедура построения баланса территорий**

Процедура стартует с блока анализа текущей ситуации. Выполняется проверка следующих утверждений:

- ✓ Рабочая таблица не пуста.
- ✓ Классификатор загружен.
- ✓ Окно карты активно.

Далее происходит заполнение полей *Area* и *Value* рабочей таблицы (Рис. 63а). Запрос к рабочей таблице дает необходимую для построения баланса информацию. Результат запроса помещается во временную таблицу **Balance** (Рис. 63б).

Затем, в зависимости от состояния переменных `isThematic`, `isDiagram`, `isFile`, выполняются, или не выполняются, следующие действия.

1. Построение тематического слоя на основе рабочей таблицы. Оператор `Shade` формируется в виде строки. Цвета тематического слоя определяются с помощью функции `RGB` с некоторым элементом случайности. Палитру цветов можно изменить, несколько поэкспериментировав с кодом определения цветов.
2. Построение, на базе таблицы **Balance**, трех круговых диаграмм следующего содержания:
  - ✓ Баланс территорий по площадям.
  - ✓ Баланс территорий по значениям из поля *Value*.
  - ✓ Баланс территорий по числу участков в каждой классифицированной группе.
3. Сохранение таблицы **Balance** в текстовом файле (разделитель символ табуляции) для дальнейшего использования при формировании отчетных документов. На рис. 64а показан вид этого файла открытый в *MS Excel*, а на рис. 64б тот же файл и там же, но в слегка отредактированном виде.

На рис. 65 показан фрагмент окна `MapInfo` с результатом работы процедуры.

```
Sub zb_CreateBalance
'Определить баланс
dim n,m,i,r,g,b,clr as integer
dim pp,ss,nm,tmpGr,crDate as string
dim kk as string*1
kk=chr$(34)
onerror goto theErr
n=TableInfo(nmLayer,TAB_INFO_NROWS)
if n=0 then
    note "В таблице " & nmLayer & " нет объектов!"
    exit sub
end if
m=Ubound(Cods)
if m=0 then
    note "Не определены типы территорий!"
    exit sub
end if
```

```

if idMap=0 then
    note "Не определено окно Карты!"
    exit sub
end if
Update nmLayer Set Area=Area(obj,UnitNm)
Update nmLayer Set Value=Area*Coeff
Select Cod,Name,Sum(Area),Sum(Value),Count(*) from nmLayer group by Cod
    into Balance
Browse * From Balance
if isThematic then
    set map window idMap redraw off
    ss="Shade window " & str$(idMap) & " " & nmLayer & " with Name ignore "
        & kk & kk & " values "
    m=TableInfo(Balance,TAB_INFO_NROWS)
    Randomize
    g=114+(140*Rnd(1))
    nm=""
    for i=1 to m
        r=200+55*Rnd(1)
        g=114+140*Rnd(1)
        b=255*Rnd(1)
        clr=RGB(r,g,b)
        Fetch Rec i From Balance
        nm=nm & kk & Balance.Name & kk & " Brush(2," &
            str$(clr) & ",16777215),"
    next
    nm=Left$(nm,len(nm)-1) &
        " default Brush(2,0,16777215) style replace off"
    ss=ss & nm
    Run Command ss
    set legend window idMap layer prev display on shades on symbols off
lines off count on title auto Font ("Arial CYR",1,10,0)
subtitle "распределение по типам" Font ("Arial CYR",0,8,0)
ascending on style size small ranges Font ("Arial CYR",0,8,0) auto
display off,auto display on,auto display on,auto display on,auto display on,
auto display on,auto display on ,auto display on
    Open Window Legend
    set map window idMap redraw on
end if
if isDiagram then
    ss=""
    crDate=FormatDate$(CurDate())
    tmpGr=ApplicationDirectory$() & "typesArea.3tf"
    Graph Name, COL3 From Balance Using tmpGr
    Set Graph Title "Баланс территорий" SubTitle "Площадь участков"
    ss="Set Graph Footnote " & kk & crDate & kk
    Run Command ss
    tmpGr=ApplicationDirectory$() & "typesValue.3tf"
    Graph Name, COL4 From Balance Using tmpGr
    Set Graph Title "Баланс территорий" SubTitle "Оценка участков"
    Run Command ss
    tmpGr=ApplicationDirectory$() & "typesCount.3tf"
    Graph Name, COL5 From Balance Using tmpGr
    Set Graph Title "Баланс территорий" SubTitle "Число участков"
    Run Command ss
end if
if isFile then
    pp=FileSaveAsDlg("", "", "TXT", "Сохранить файл баланса")
    if Len(pp)=0 then Exit sub end if
    Export Balance Into pp Type "ASCII" CharSet "WindowsCyrillic"
    Titles Overwrite
end if
exit sub

```

```
theErr:
note Error$()
end sub
```

Cod	Name	Area	Coeff	Value
4	лес	12.7455	20.2	257.458
5	кустарник	28.4609	12.6	358.608
6	болото	23.7281	12.4	294.228
7	застройка	12.8406	95.6	1 227.56
5	кустарник	14.3414	10.6	152.019
6	болото	14.5429	17.2	250.139
1	пашня	28.2893	44.1	1 247.56
3	сенокос	23.2224	14.7	341.369
4	лес	16.3534	26.4	431.731
1	пашня	37.4976	51.4	1 927.38
5	кустарник	19.7479	11.3	223.151
4	лес	9.60497	27.3	262.216
5	кустарник	11.5665	12.4	143.425

а

Cod	Name	Sum(Area)	Sum(Value)	Count
4	лес	158.17	4 178.18	7
5	кустарник	84.353	997.99	5
6	болото	93.5997	1 370.23	4
7	застройка	25.5813	2 389.52	2
1	пашня	195.538	8 982.59	6
3	сенокос	42.82	639.253	2
2	пастбище	11.0591	123.862	1

б

Рис. 63

	A	B	C	D	E
1	Cod	Name	_COL3	_COL4	_COL5
2	4	лес	158.17	158.17	7
3	5	кустарник	84.353	84.353	5
4	6	болото	93.5997	93.5997	4
5	7	застройка	25.5813	25.5813	2
6	1	пашня	195.538	195.538	6
7	3	сенокос	42.82	42.82	2
8	2	пастбище	11.0591	11.0591	1

а

Код	Наименование	Площадь	Оценка	Кол-во
1	пашня	195.5	8983	6
2	пастбище	11.1	124	1
3	сенокос	42.8	639	2
4	лес	158.2	4178	7
5	кустарник	84.4	998	5
6	болото	93.6	1370	4
7	застройка	25.6	2390	2
Итого:		<b>611.1</b>	<b>18682</b>	<b>27</b>

б

Рис. 64

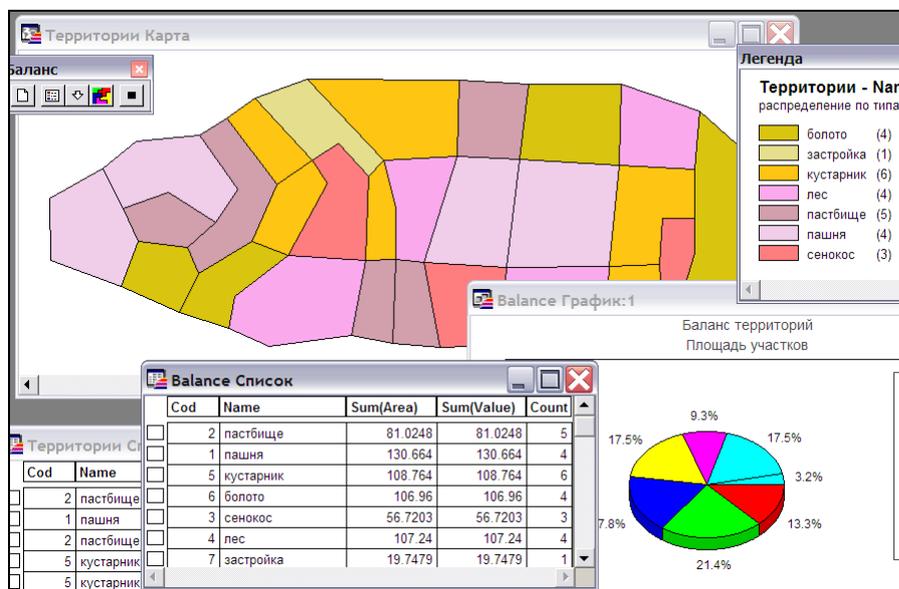


Рис. 65

### Процедура формирования новой таблицы

Для работы программы требуется рабочая таблица с фиксированной структурой. Упростить для пользователя ее создание позволит процедура `zb_NewTable`.

Структура полей таблицы записывается в массив `sColumnInfo` в виде строк вида: *Имя\_поля Пробел Тип\_поля*. Далее в диалоге настраиваются параметры таблицы:

- ✓ Рабочая область.
- ✓ Полное имя.
- ✓ Стилль заливки и границы области.

После выхода из диалога по кнопке **Ок** требуемая таблица будет создана и открыта в окне карты. Вид формы диалога показан на рис. 66.

```
Sub zb_NewTable
'Создать таблицу
dim sColumnInfo(5) as string
dim stBru as Brush
dim stPen as Pen
dim sminx,sminy,smaxx,smaxy as string
stBru=CurrentBrush()
stPen=CurrentBorderPen()
sColumnInfo(1)="Cod SmallInt"
sColumnInfo(2)="Name Char(20)"
sColumnInfo(3)="Area Float"
sColumnInfo(4)="Coeff Float"
sColumnInfo(5)="Value Float"
Dialog
    Title "Новая таблица"
    Width 273
    Height 82
    Control StaticText
        Title "Рабочая область:"
        Width 74
        Height 10
        Position 7, 4
    Control StaticText
        Title "min X"
        Width 21
        Height 10
        Position 9, 19
    Control StaticText
        Title "max X"
        Width 23
        Height 10
        Position 9, 36
    Control EditText
        Value ""
        Width 55
        Height 13
        Position 38, 18
        ID 101
        Value sminx
        Into sminx
    Control EditText
        Value ""
        Width 55
        Height 13
        Position 38, 36
        ID 102
        Value smaxx
        Into smaxx
    Control StaticText
        Title "min Y"
        Width 21
        Height 10
        Position 108, 19
    Control StaticText
```

```

        Title "max Y"
        Width 23
        Height 10
        Position 108, 36
Control EditText
        Value ""
        Width 55
        Height 13
        Position 142, 18
        ID 103
        Value sminy
        Into sminy
Control EditText
        Value ""
        Width 55
        Height 13
        Position 142, 36
        ID 104
        Value smaxy
        Into smaxy
Control Button
        Title "Таблица"
        Width 51
        Height 21
        Position 211, 17
        Calling SetTableName
Control OkButton
        Width 37
        Height 17
        Position 185, 60
Control CancelButton
        Width 37
        Height 17
        Position 229, 60
Control StaticText
        Title "Направление осей координат"
        Width 100
        Height 10
        Position 7, 60
        Control StaticText
            Title "X - на Север, Y - на Восток"
            Width 100
            Height 10
            Position 7, 70
Control BrushPicker
        Position 115,58
        Value stBru
        Into stBru
Control PenPicker
        Position 140,58
        Value stPen
        Into stPen
If CommandInfo(CMD_INFO_DLG_OK) Then
    if Len(NewTableName)=0 then exit sub end if
idMap=ItsMap()
    call CreateTableMap(NewTableName, sColInfo, val(sminx), val(smaxx),
val(sminy), val(smaxy), idMap)
    Set Map Window idMap Layer PathToTableName$(NewTableName)
    Display Global Global Pen stPen Global Brush stBru
else
    NewTableName=""
End If
end sub

```

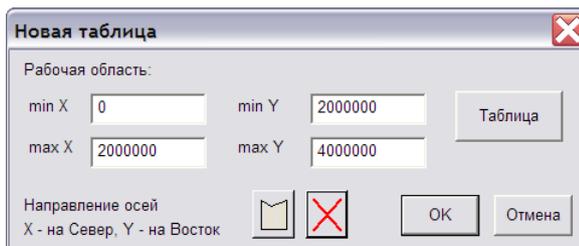


Рис. 66

Процедура установки полного имени таблицы.

```
Sub SetTableName
NewTableName=FileSaveAsDlg("C:\", "", "ТАВ", "Создать таблицу")
if Len(NewTableName)=0 then
    note "Имя для таблицы не установлено!"
    Exit sub
end if
end sub
```

Процедура CreateTableMap непосредственно формирует таблицу и открывает ее в окне карты.

```
Sub CreateTableMap(ByVal FullNameTable As String,sCols() as string,
    byval minX as float,byval maxX as float,
    byval minY as float,byval maxY as float,idmap as integer)
dim i as smallint
dim str,nmTable as string
dim kk as string*1
kk=chr$(34)
nmTable=PathToTableName$(FullNameTable)
For i = 1 To Ubound(sCols)
    str=str & sCols(i) & ", "
Next
str=Left$(str,Len(str)-1)
str="Create Table " & nmTable &
    " (" & str & ") File " & kk & FullNameTable &
    kk & " TYPE NATIVE CharSet " &
    kk & "WindowsCyrillic" & kk
Run Command str
Create Map For nmTable CoordSys Nonearth Units "m" Bounds
(minY,minX)(maxY,maxX)
if idmap>0 then
    Add Map Window idmap Auto Layer nmTable
Else
    Map From nmTable
    idmap=FrontWindow()
end if
end sub
```

### **Процедура окончания работы с программой**

Процедура не выполняет никакой дополнительной работы по выгрузке программы, поэтому имеет тривиальный вид, сводящийся к вызову оператора End Program.

```
Sub zb_Exit
'Выход
End Program
end sub
```

## Тема 53. Управление растровыми слоями

### Постановка задачи

Нередко в проектах MapInfo участвует много растровых слоев, настолько много, что управление ими становится затруднительным. Для этого случая предлагается все растровые слои разбить на группы. Признаком группы установить уникальный префикс (постфикс)<sup>42</sup> в имени таблицы.

Проектируемая программа должна обеспечить простой способ включения/отключения видимости групп растровых слоев<sup>43</sup>.

### Проект

Пользовательский интерфейс программы будет представлен в виде инструментальной панели с переменным числом кнопок. Число кнопок равно числу префиксов/постфиксов (плюс кнопка **Выход из программы**). Состояние кнопок: кнопка нажата – группа растровых слоев не видна, кнопка отжата – видна.

Имена префиксов (постфиксов) будут храниться во внешнем текстовом файле.

### Код

Весь код программы будет организован в виде одного модуля.

#### Блок описаний

```

Include "mapbasic.def"
Include "Menu.def"
Include "icons.def"
Declare Sub Main
Declare Sub AlterButtonToBPad
Declare sub readPref
Declare Sub ShowRastr
Declare Sub InverseState (state as logical,pstf as string)
Declare function isMap() as integer
Declare sub mdEnd
Type prefRastr
    pref as string
    state as logical
End Type
dim idWinMap as integer
dim isPref as logical
dim pref() as prefRastr
dim tt as integer
    
```

#### Главная процедура

```

Sub Main
dim i as integer
'Построение базовой панели с одной кнопкой
Create ButtonPad "Показать/Скрыть" As
    PushButton
        Icon MI_ICON_SIGNS_1
        Calling mdEnd
    
```

<sup>42</sup> Префикс - часть слова, стоящая перед корнем и изменяющая его лексическое или грамматическое значение; постфикс - словообразовательная морфема, следующая за окончанием.

<sup>43</sup> Аналогичный подход можно использовать и для векторных слоев.

```

        ID 2562
        HelpMsg "Выход из программы\nВыход из программы"
        Enable
    Width 15
    Show
    ToolbarPosition (1,0)
    Fixed
'Читаем файл префиксов
call readPref
'Включаем дополнительные кнопки в базовую панель
for i=1 to ubound(pref)
    tt=i+1000
    call AlterButtonToVPad
next
'Обеспечиваем соответствие карты состоянию кнопок
if isMap()=1 then
    for i=1 to ubound(pref)
        pref(i).state="T"
    next
    exit sub
end if
for i=1 to ubound(pref)
    call InverseState(pref(i).state,pref(i).pref)
next
end sub

```

Текстовый файл имеет фиксированное имя **pref.txt** и размещается в одной папке с программой. Пример такого файла показан ниже.

_25_1	Префикс для первой группы растровых таблиц <sup>44</sup>
_24_2	Префикс для второй группы растровых таблиц
_25_2	Префикс для третьей группы растровых таблиц
t	Последняя строка файла. Признак того, что в файле указаны префиксы (t или T). Если будет f или F значит в файле постфиксы.

Вид построенной инструментальной панели показан на рис. 67.

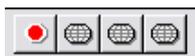


Рис. 67

Кнопки имеют всплывающие подсказки. Так для первой дополнительной кнопки она будет иметь вид «Растр: \_25\_1\*», аналогично и для других кнопок. В случае постфикса подсказка будет иметь вид «Растр: \*AA» (постфикс равен значению «AA»).

### Процедура чтения файла префиксов

```

sub readPref
Dim sAppPath As String
dim i as integer
dim s as string
i=0
sAppPath = ApplicationDirectory$() & "pref.txt"
If FileExists(sAppPath) Then
    Open File sAppPath For Input as #1
    Do
        Input #1, s
        i=i+1
        redim pref(i)
    
```

<sup>44</sup> В данном случае имена таблиц начинаются с цифры, а так как MapInfo к таким именам автоматически добавляет подчеркивание то и в файле префиксы указаны с подчеркиванием.

```

        pref(i).pref=s
        pref(i).state=false
Loop While Not EOF(1)
redim pref(i-1)
Close File #1
s=pref(i-1).pref
if s<>"T" and s<>"t" and s<>"F" and s<>"f" then
    note "Неверный параметр в последней строке файла pref.txt." &
        chr$(10) & "Исправьте файл и загрузите программу."
    End Program
end if
isPref =pref(i-1).pref
redim pref(i-2)
if isPref<>true and isPref<>false then
    note "Неверный параметр в последней строке файла pref.txt." &
        chr$(10) & "Исправьте файл и загрузите программу."
    End Program
end if
end if
end sub

```

### **Процедура включения дополнительных кнопок в базовую панель**

```

Sub AlterButtonToBPad
dim s as string
s="Пастр: "
if isPref=false then
    s=s & "*"
    s=s & pref(tt-1000).pref & "\n" & s & pref(tt-1000).pref
else
    s=s & pref(tt-1000).pref & "*" & "\n" & s & pref(tt-1000).pref & "*"
end if
Alter ButtonPad "Показать/Скрыть"
Add ToggleButton
    Icon MI_ICON_MB_8
    Calling ShowRastr
    HelpMsg s
    ID tt
    Enable Uncheck
end sub

```

### **Процедуры визуализации слоев**

```

Sub ShowRastr
dim r as integer
r=CommandInfo(CMD_INFO_TOOLBTN )-1000
if isMap()=1 then
    pref(r).state=not pref(r).state
    exit sub
end if
call InverseState (pref(r).state,pref(r).pref)
end sub

Sub InverseState (state as logical,pstf as string)
dim j,n,k as smallint
dim s,ff as string
k=len(pstf)
state=not state
n=MapperInfo(idWinMap,MAPPER_INFO_LAYERS)
for j=1 to n
'Здесь для большей определенности можно добавить
'проверку типа слоя (LAYER_INFO_TYPE_IMAGE)
    s=LayerInfo(idWinMap,j,LAYER_INFO_NAME)

```

```

        if isPref then
            ff=left$(s,k)
        else
            ff=right$(s,k)
        end if
        if ff=pstf then
            if state=true then
'Слой виден
                Set Map Window idWinMap Layer j Display Global
            Else
'Слой не виден
                set map Window idWinMap layer j Display Off
            end if
        end if
    next
end sub

```

### **Прочие процедуры**

```

function isMap() as integer
'Проверка типа окна
isMap=0
idWinMap= FrontWindow()
If WindowInfo(idWinMap, WIN_INFO_TYPE) <> WIN_MAPPER Then
    Note "Инструмент используется только в окне Карты."
    isMap=1
End If
end function

Sub mdEnd
'Выход из программы
End Program
end sub

```

## **Тема 54. Динамическая загрузка растровых таблиц**

### **Постановка задачи**

Имеется папка, в которой хранится множество растровых таблиц. Для работы с конкретным проектом пользователю нужны из этого множества одна, пять, десять таблиц, причем, как вы понимаете найти их в этой куче не просто. Проектируемая программа должна максимально упростить решение этой задачи.

### **Проект**

Пользовательский интерфейс программы будет представлен в виде инструментальной панели с тремя кнопками. Процессы связанные с кнопками:

1. Определение папки с растровыми таблицами.
2. Загрузка таблиц имеющих пересечение с выбранным на карте объектом.
3. Выгрузка всех растровых таблиц.

### **Код**

Весь код программы будет организован в виде одного модуля.

### **Блок описаний**

```

Include "mapbasic.def"
define MAX_PATH 260
define INVALID_HANDLE_VALUE -1

```

```

Type FILETIME
    dwLowDateTime as integer
    dwHighDateTime as integer
End Type
Type WIN32_FIND_DATA
    dwFileAttributes as integer
    ftCreationTime as FILETIME
    ftLastAccessTime as FILETIME
    ftLastWriteTime as FILETIME
    nFileSizeHigh as integer
    nFileSizeLow as integer
    dwReserved0 as integer
    dwReserved1 as integer
    cFileName as string * MAX_PATH
    cAlternate as string *14
End Type
Type bmpCoord
    bmpName as string
    bmpX() as float
    bmpY() as float
End Type
Declare function FindFirstFile lib "kernel32" alias "FindFirstFileA"(byval
    lpFileName as string,lpFindFileData as WIN32_FIND_DATA) as integer
Declare function FindNextFile lib "kernel32" alias "FindNextFileA"(byval
    hFindFile as integer,lpFindFileData as WIN32_FIND_DATA) as integer
Declare function FindClose lib "kernel32"(byval hFindFile as integer) as
    integer
Declare Sub Main
Declare sub SetFOLDER
Declare sub SetBMP
Declare sub UnloadBMP
Declare Sub CreateListBmpLayerNames()
Declare sub ExamineFile(byval cFile as string,byval t as integer)
Declare sub SetBmpCoord(byval strCrd as string,byval t as integer)
Declare Sub Extract(ByVal index As Integer, ByVal StrToParse As String,
    retStr As String,byval delim as string)
Declare function IsSelectionObject() as integer
Declare Function IsLayerInMap(nmTab as string) as logical
Declare function IsTabInMI(byval nameTab as string) as logical
Declare Function count_items(ItemTitleList as String,
    byval delim as string) As SmallInt
Declare function isMap() as integer dim idWinMap as integer
dim idWinMap as integer
dim curPtch as string
dim bmpInfo() as bmpCoord
dim curFiles() as string
dim curCoordFiles() as string
dim ListLayers() as string

```

### **Главная процедура**

```

Sub Main
curPtch = ""
Create ButtonPad "bmpOptima" as
    PushButton
        HelpMsg
        "Выбор активной папки с растровыми таблицами\nАктивная папка"
        Calling SetFOLDER
        ID 2001
        Icon 100 File "bmpOptima.dll"
        Enable
    PushButton
        HelpMsg "Установка растров\nУстановить растры"

```

```

Calling SetBMP
    ID 2002
    Icon 102 File "bmpOptima.dll"
    Enable
PushButton
    HelpMsg "Выгрузка всех растров\nВыгрузить растры"
    Calling UnloadBMP
    ID 2003
    Icon 104 File "bmpOptima.dll"
    Enable
ToolbarPosition (1,0)
Show
Fixed
end sub

```

Обратите внимание на выбор иконок для кнопок. В данном случае используются иконки из файла **bmpOptima.dll**. Их можно заменить на любые доступные в MapInfo предварительно подключив файл **Icons.def**.

### **Процедура выбора папки с растровыми таблицами**

```

sub SetFOLDER
dim k,i,n,j as integer
dim hSearch as integer
dim WFD as WIN32_FIND_DATA
dim cont as integer
dim SearchStr as string
dim curFile as string
'Здесь для определения папки используется диалог выбора файла45 и
'от пользователя потребуется дополнительное действие:
'выбрать в папке файл (любой)
curPtch=FileOpenDlg(curPtch,"" , "ТАБ", "Выбор папки")
if curPtch ="" then
    note "Папка не определена!"
    exit sub
end if
curPtch=PathToDirectory$(curPtch)
SearchStr="*.tab"
onerror goto nextErr
cont=true
k=0
'Поиск первого файла *.tab в папке curPtch:
'Манипулятор hSearch используется для продолжения поиска.
'Переменная WFD со структурой WIN32_FIND_DATA заполняется информацией
'о найденном файле и используется при продолжении поиска.
hSearch =FindFirstFile(curPtch & SearchStr ,WFD)
if hSearch <> INVALID_HANDLE_VALUE Then
    do while cont
        k=k+1
        redim curFiles(k)
        curFiles(k)=WFD.cFileName
'Функция ищет следующий файл.
'В случае удачи возвращает TRUE (<>0) или FALSE (=0) при неудаче.
        cont=FindNextFile(hSearch,WFD)
    loop
'Закрывает манипулятор.
    cont=FindClose(hSearch)

```

<sup>45</sup> Нет проблем и с использованием диалога выбора папки см. Тема 24.

```

else
    note "Файлы *.tab в папке " & curPtch & " не найдены!"
    exit sub
end if
onerror goto 0
redim curCoordFiles(k)
for i=1 to k
    curFile=curPtch & curFiles(i)
'Считывание координат из растровых таблиц
    call ExamineFile(curFile,i)
next
'Уточняется число растровых таблиц
n=0
for i=1 to k
    if len(curFiles(i))>0 then
        n=n+1
    end if
next
if n=0 then
    note "В папке " & curPtch & " нет таблиц растров!"
    exit sub
end if
'Заполнение bmpInfo со структурой bmpCoord
redim bmpInfo(n)
j=0
for i=1 to k
    if len(curFiles(i))>0 then
        j=j+1
        bmpInfo(j).bmpName=curFiles(i)
        call SetBmpCoord(curCoordFiles(i),j)
    end if
next
redim curFiles(0)
redim curCoordFiles(0)
exit sub
nextErr:
note "Ошибка при определении файлов в папке " & curPtch & "!"
end sub

```

### ***Процедура загрузки растровых таблиц***

```

sub SetBMP
dim tt,n,m,i,j as integer
dim objRegion,objMBR,objSel as object
dim tab,tabSel as string
idWinMap=isMap()
if idWinMap=0 then
    exit sub
end if
'Проверка: выбран один объект
tt=IsSelectionObject()
if tt=0 then
    exit sub
end if
if tt>1 then
    note "Только один объект может быть выбран!"
    exit sub
end if
tabSel =SelectionInfo(SEL_INFO_TABLENAME)
Set CoordSys Table tabSel
objSel=Selection.obj
'Создаем список имен всех растровых слоев (ListLayers) в окне idWinMap

```

```

call CreateListBmpLayerNames
n=UBound(bmpInfo)
Set Map Window idWinMap Redraw Off
for i=1 to n
'Создаем область objRegion для таблицы bmpInfo(i)
  Create Region Into Variable objRegion 0
  m=UBound(bmpInfo(i).bmpX)
  for j=1 to m
    Alter Object objRegion Node Add
      (bmpInfo(i).bmpY(j), bmpInfo(i).bmpX(j) )
  Next
'Строим MBR покрытие объекта objRegion.
  objMBR=mbr(objRegion)
'Определяем пересечение с выбранным на карте объектом
'и если необходимо добавляем растровую таблицу к карте idWinMap
  if (objMBR Intersects objSel)=TRUE then
    tab=PathToTableName$(bmpInfo(i).bmpName)
    if IsTabInMI(tab)=FALSE then
      Open Table bmpInfo(i).bmpName
      Add Map Window idWinMap Auto Layer tab
    else
      if IsLayerInMap(tab)=true then
        Set Map Window idWinMap Layer tab Display Graphic
      else
        Add Map Window idWinMap Auto Layer tab
        Set Map Window idWinMap Layer tab Display Graphic
      end if
    end if
  end if
next
Set Map Window idWinMap Redraw On
end sub

```

### ***Процедура выгрузки растровых таблиц***

```

sub UnloadBMP
dim n,i as integer
dim tp as smallint
dim nm as string
idWinMap=isMap()
if idWinMap=0 then
  exit sub
end if
Set Map Window idWinMap Redraw Off
'Создаем список имен всех растровых слоев (ListLayers) в окне idWinMap
'и закрываем эти таблицы
call CreateListBmpLayerNames
n=ubound(ListLayers)
for i=1 to n
  Close Table ListLayers(i)
Next
'Число всех открытых на данный момент таблиц
n=NumTables()
if n<1 then exit sub end if
'Закрываем остающиеся в системе растровые таблицы
for i=1 to n
  tp=TableInfo(i,TAB_INFO_TYPE)
  if tp=TAB_TYPE_IMAGE then
    nm=TableInfo(i,TAB_INFO_NAME)
    Close Table nm
  end if
next

```

```
Set Map Window idWinMap Redraw On
end sub
```

### ***Процедура считывания координат из растровых таблиц***

```
sub ExamineFile(byval cFile as string,byval t as integer)
dim isBmp as logical
Dim str,ptStr1,coordStr As String
'Открыть файл cFile для чтения
Open File cFile For Input As #1 CharSet "WindowsCyrillic"
isBmp =false
ptStr1="%Type%RASTER%"
Do While Not EOF(1)
    Line Input #1, str
'Проверка: таблица растровая
    if Like(str,ptStr1,"") then
        isBmp =true
'Sформировать строку с координатами coordStr (разделитель @)
        Do While Not EOF(1)
            Line Input #1, str
            ptStr1="%(%,%) (%,%)%"
            if Like(str,ptStr1,"") then
                coordStr =coordStr & str & "@"
            end if
        loop
    end if
Loop
Close File #1
'Заполнить элементы t массивов curFiles и curCoordFiles
if isBmp then
    curFiles(t)=cFile
    curCoordFiles(t)=coordStr
else
    curFiles(t)=""
    curCoordFiles(t)=""
end if
end sub
```

### ***Процедура извлечения координат из строки***

```
sub SetBmpCoord(byval strCrd as string,byval t as integer)
dim m,i,is1,is2 as integer
dim str,s,ss as string
'Число элементов в строке
m=count_items(strCrd,"@")
redim bmpInfo(t).bmpX(m)
redim bmpInfo(t).bmpY(m)
for i=1 to m
'Извлечь элемент i из строки strCrd
    call Extract(i,strCrd,str,"@")
    is1=instr(1,str,"(")
    is2=instr(is1,str,")")
    s=mid$(str,is1+1,is2-is1-1) & ","
'Определить значение X
    call Extract(2,s,ss,",")
    bmpInfo(t).bmpX(i)=val(ss)
'Определить значение Y
    call Extract(1,s,ss,",")
    bmpInfo(t).bmpY(i)=val(ss)
next
end sub
```

### **Прочие процедуры**

```
function isMap() as integer
```

**'Проверка: окно карты активно.**

**'Возвращает идентификатор окна или ноль.**

```
dim map_id as integer
IsMap=0
map_id = FrontWindow()
If WindowInfo(map_id , WIN_INFO_TYPE) <> WIN_MAPPER Then
    Note "Инструмент используется только в окне Карты!"
    Exit function
End If
IsMap=map_id
end function
```

```
function IsSelectionObject() as integer
```

**'Проверка: есть выбранные объекты.**

**'Возвращает число выбранных объектов.**

```
dim n as integer
n=SelectionInfo(SEL_INFO_NROWS)
if n=0 then
    note "Не выбраны объекты!"
end if
IsSelectionObject=n
End function
```

```
Sub CreateListBmpLayerNames()
```

**'Создает список имен растровых слоев (ListLayers) окна idWinMap**

```
Dim s As String
Dim j,n,m,t As Integer
t=0
n=MapperInfo(idWinMap,MAPPER_INFO_LAYERS)
For j=1 To n
    s=LayerInfo(idWinMap,j,LAYER_INFO_NAME)
    m=LayerInfo(idWinMap,j,LAYER_INFO_TYPE)
    If m=LAYER_INFO_TYPE_IMAGE Then
        t=t+1
        ReDim ListLayers(t)
        ListLayers(t)=s
    End If
Next
End Sub
```

```
function IsTabInMI (byval nameTab as string) as logical
```

**'Проверка: открыта ли таблица nameTab.**

**'Возвращает TRUE если таблица открыта.**

```
dim num,i as smallint
dim s as string
IsTabInMI =false
num=NumTables()
if num<1 then exit function end if
for i=1 to num
    s=TableInfo(i,TAB_INFO_NAME)
    if s=nameTab then
        IsTabInMI=true
        exit for
    end if
next
end function
```

```
Function IsLayerInMap(nmTab as string) as logical
```

'Проверка: слой (таблица nmTab) присутствует в карте idWinMap.

'Фактически проверяется список ListLayers соответствующий карте idWinMap.

'Возвращает TRUE если такой слой есть в карте.

```
Dim s As String
Dim j,n As Integer
IsLayerInMap=false
n=UBound(ListLayers)
For j = 1 To n
    if nmTab=ListLayers(j) then
        IsLayerInMap=true
        exit for
    end if
Next
End Function
```

```
Function count_items(ItemTitleList as String,
    byval delim as string) as SmallInt
```

'Возвращает число элементов в строке ItemTitleList.

'Разделитель элементов delim.

```
Dim i,counter as SmallInt
    For i = 1 to Len(itemTitleList)
        If Mid$(itemTitleList, i, 1)=delim Then
            counter = counter + 1
        End If
    Next
    count_items = counter
End Function
```

```
Sub Extract(ByVal StrIndex As Integer,ByVal StrToParse As String,
    retStr As String,byval delim as string )
```

'Возвращает элемент строки StrToParse (строка retStr) по индексу

'элемента StrIndex. Разделитель элементов строки delim.

```
Dim iBeg, iEnd, r As Integer
iBeg = 1
iEnd = InStr(iBeg,StrToParse,delim)
For r = 2 To StrIndex
    iBeg = iEnd + 1
    iEnd = InStr(iBeg,StrToParse,delim)
Next
If iEnd = 0 Then
    iEnd = 99
End If
retStr = Mid$(StrToParse,iBeg,iEnd - iBeg)
End Sub
```

## Глава 7. Интегрированная картография

Термин интегрированная картография порожден эффектом встраивания (интеграции) информационных окон MapInfo в разрабатываемое приложение. Это приложение должно осуществлять управление программой MapInfo посредством механизма управления объектами *OLE-Automation*. *OLE<sup>46</sup>-Automation* это технология на основе *COM<sup>47</sup>*, позволяющая прикладной программе-серверу предоставлять свои сервисы в распоряжение другой программы-клиента через соответствующие интерфейсы.

Программа MapInfo может быть запущена внешним приложением в качестве *OLE-Automation* сервера с предоставлением доступа ко всем своим объектам и методам, с помощью которых можно выполнить последовательность операторов или вычислить выражение, заданное на языке MapBasic. Внешнее приложение в этом случае выступает в роли клиента. Кроме того для управления MapInfo можно использовать технологию динамического обмена данными (*DDE<sup>48</sup>*). И хотя в отдельных случаях ее использование может быть полезным, но здесь она не рассматривается.

Общая схема создания приложения с использованием интегрированной картографии такова: пишем программу клиент на некотором языке (не MapBasic) и из этой программы организуем обращения к MapInfo с помощью *OLE-Automation* для решения каких либо задач связанных с обработкой географической информации. В качестве языка программирования будем использовать *VB.NET*. В программе клиенте объявляется объектная переменная, которая будет представлять MapInfo. При присвоении объекта переменной выполняется процесс связывания. Различают раннее и позднее связывание.

- ✓ Для раннего связывания необходимо наличие библиотеки типов *OLE-сервера*. При этом компилятор может проверить синтаксис и соответствие типов, правильность указания числа и типа параметров, передаваемых методу или свойству, и тип возвращаемого значения. Поскольку при выполнении приложения раннее связывание требует меньше времени для вызова свойств или методов, оно будет работать быстрее, чем позднее. Однако разница в производительности, как правило, незначительна. Основной недостаток раннего связывания заключается в проблеме совместимости версий программы сервера. Подключенная во время программирования библиотека типов может не соответствовать библиотеке типов на другом компьютере при выполнении программы.
- ✓ В отличие от раннего связывания, позднее связывание сопоставляет вызовы свойств и методов соответствующим объектам лишь в процессе выполнения программы. При компиляции определяется только некоторый объект общего вида (тип *Object*), и выполнить проверку соответствия типов невозможно. Но в тоже время это дает возможность избежать некоторой зависимости от версии,

<sup>46</sup> Аббревиатура OLE расшифровывается как Object Linking Embedding (встраивание и связывание объектов).

<sup>47</sup> Аббревиатура COM расшифровывается как Component Object Model (модель компонентных объектов).

<sup>48</sup> DDE (Dynamic Data Exchange) — механизм взаимодействия приложений в операционной системе Microsoft Windows. Хотя этот механизм до сих пор поддерживается, но в последнее время он все чаще заменяется на более мощные технологии COM.

присущей раннему связыванию. При позднем связывании не могут использоваться возможности IntelliSense<sup>49</sup>, предоставляющие сведения для правильного использования методов и свойств.

Выбор типа связывания влияет на такие характеристики приложения, как производительность, гибкость и удобство сопровождения. Варианты кода при раннем и позднем связывании будут рассмотрены далее при разборке примера программы. Следует обратить внимание на то, что в любом случае, для реализации технологии интегрированной картографии, на компьютере должна быть установлена программа MapInfo или *Runtime-модуль* MapInfo.

## Тема 55. Объектная модель MapInfo OLE Automation Type Library<sup>50</sup>

### Объект Application

Представляет работающий экземпляр MapInfo.

#### Свойства:

ReadOnly Property Application As Object	Возвращает объект Application (MapInfo).
ReadOnly Property FullName As String	Возвращает полный путь к Mapinfow.exe.
Property LastErrorCode As Integer	Указывает код последней MapBasic-ошибки, полученной в процессе вызова метода Do, Eval или RunCommand. Этот код автоматически заменяется кодом следующей ошибки и т.д. Автоматически в ноль никогда не сбрасывается. Коды ошибок, возвращаемые этим свойством, превышают на 1000 соответствующие коды ошибок в среде MapBasic.
ReadOnly Property LastErrorMessage As String	Сообщение об ошибке, соответствующее коду LastErrorCode.
ReadOnly Property MBAApplications As Object	Возвращает коллекцию MapBasic программ, работающих в рамках приложения Application.
ReadOnly Property MIMapGen As Object	Объект изначально использовался в MapInfo ProServer (картографический Internet-сервер MapInfo) и рассчитан на работу с одним окном карты.
ReadOnly Default Property Name As String	Возвращает имя приложения (MapInfo Professional)
ReadOnly Property Parent As Object	Аналогично Application.
ReadOnly Property ProductLevel As Integer	Возвращает целое значение, обозначающее «уровень» программы MapInfo. Для MapInfo Professional это значение равно 200.
ReadOnly Property Version As String	Номер текущей версии, умноженный на 100 (для MapInfo 8.0.0 вернет строку «800»).
Property Visible As Boolean	Определяет или устанавливает видимость окна MapInfo. Не имеет

<sup>49</sup> IntelliSense — технология автодополнения Microsoft. Дописывает название функции при вводе начальных букв. Позволяет увидеть описание функции, в том числе список аргументов.

<sup>50</sup> В данном случае MapInfo 8.0 OLE Automation Type Library.

**Методы:**

```
Function DataObject (ByVal windowID As Integer) As Object
Sub [Do] (ByVal command As String)

Function Eval (ByVal expression As String) As String

Sub RunCommand (ByVal command As String)

Sub RunMenuCommand (ByVal id As Integer)

Sub SetCallback (ByVal callbackobject As Object)
```

отношения к видимости встроенного в приложение окна карты, списка и т.д.

Возвращает интерфейс IUnknown, представляющий окно windowID. Выполняет оператор MapBasic записанный в виде строки. Вычисляет выражение MapBasic, записанное в виде строки, и возвращает полученное значение как строку. Выполняет оператор MapBasic, заданный строкой (то же что метод Do). Выполняет команду меню, заданную кодом id. Регистрирует объект OLE Automation как получатель уведомлений, генерируемых программой MapInfo.

**Коллекция MVAApplications**

Представляет программы MapBasic, работающие в рамках приложения *Application*.

**Свойства:**

```
ReadOnly Property Application As Object
ReadOnly Property Count As Integer
ReadOnly Property Parent As Object
```

Возвращает объект Application (MapInfo).  
Возвращает число приложений (MBX) в коллекции.  
Аналогично Application.

**Методы:**

```
Function Item (ByVal index As Object) As Object
```

Возвращает объект MapBasicApplication. Аргумент может быть целочисленным индексом (1, ..., Count) или строкой (имя программы MBX).

**Объект MapBasicApplication**

Представляет программу MapBasic из коллекции *MVAApplications*.

**Свойства:**

```
ReadOnly Property Application As Object
ReadOnly Property FullName As String

ReadOnly Property MBGlobals As Object

ReadOnly Default Property Name As String
ReadOnly Property Parent As Object
```

Возвращает объект Application (MapInfo).  
Возвращает полное имя MBX-программы, например C:\PROG\ZoneBalance.MBX  
Возвращает коллекцию глобальных переменных для объекта MapBasicApplication.  
Возвращает имя MBX-программы, например ZoneBalance.MBX  
Аналогично Application.

**Методы:**

```
Sub [Do] (ByVal command As String)

Function Eval (ByVal expression As String) As String
```

Оператор MapBasic в виде строки (command), пересылается в процедуру RemoteMsgHandler MapBasic приложения (объект MapBasicApplication) и выполняется. Строка expression передается как аргумент в функцию RemoteQueryHandler() MapBasic приложения (объект MapBasicApplication). Возвращается значение этой функции.

## Коллекция MBGlobals

Представляет коллекцию глобальных переменных для объекта *MapBasicApplication*.

### Свойства:

ReadOnly Property Application As Object	Возвращает объект Application (MapInfo).
ReadOnly Property Count As Integer	Число объектов в коллекции.
ReadOnly Property Parent As Object	Аналогично Application.

### Методы:

Function Item(ByVal index As Object) As Object	Возвращает объект MBGlobal. Аргумент может быть целочисленным индексом (1, ..., Count) или строкой (имя глобальной переменной).
--	---

## Объект MBGlobal

Представляет глобальную переменную из коллекции *MBGlobals*.

### Свойства:

ReadOnly Property Application As Object	Возвращает объект Application (MapInfo).
ReadOnly Property Name As String	Возвращает имя глобальной переменной.
ReadOnly Property Parent As Object	Аналогично Application.
ReadOnly Property Type As String	Возвращает тип глобальной переменной.
Default Property Value As String	Возвращает значение глобальной переменной.

## Тема 56. Простое приложение с интегрированной картой

Зададимся целью создать простое приложение на VB.NET, которое будет иметь встроенное окно карты и реализует некоторые основные инструменты MapInfo.

Сформируем в *Visual Studio* новый проект *Windows Forms Application*. Уже имеющуюся в проекте форму назовем **Main**. Создадим и добавим к проекту:

1. Модуль **Common.vb**. Объявления общих переменных и констант.
2. Модуль **mbDEF.vb**. Константы MapBasic. Информация из файла **Mapbasic.def** приведенная к виду объявления констант в VB.NET.
3. Класс **cMI80.vb**. Методы и свойства использующие MapInfo.
4. COM класс **comCallBk.vb**. Получение текста из строки сообщений MapInfo.

## Модуль Common

В модуле **Common** разместим три строки объявлений.

```
Public mInfo As cMI80
Public miCB As New comCallBk
Public Const kk As Char = Chr(34)
```

В первой строке объявляется объект **mInfo** класса **cMI80**, свойства и методы которого, используют MapInfo. По сути, этот объект и будет представлять для нас MapInfo в программе. К нему мы будем обращаться, если нужно выполнить настройку карты, создать объект, сохранить таблицу и т.д.

## Модуль mbDEF.vb

Здесь объявляются константы MapBasic. Конечно, без этого можно обойтись и везде в коде вместо констант использовать их значения, но с ними все-таки лучше и код становится более информативным. Чтобы показать, как трансформируется информация из файла **Mapbasic.def** в код VB.NET, приведу два фрагмента кода один из файла **Mapbasic.def** и другой соответствующий ему на VB.

### Mapbasic.def:

```
Define COL_TYPE_CHAR 1
Define COL_TYPE_DECIMAL 2
Define COL_TYPE_INTEGER 3
Define COL_TYPE_SMALLINT 4
```

### mbDEF.vb:

```
Public Const COL_TYPE_CHAR As Int16 = 1
Public Const COL_TYPE_DECIMAL As Int16 = 2
Public Const COL_TYPE_INTEGER As Int16 = 3
Public Const COL_TYPE_SMALLINT As Int16 = 4
```

### Класс cMI80

Создание класса начинается с выбора меню *Project/Add New Item.../Class* и далее вводим код реализующий свойства и методы класса.

Приведу начальный фрагмент кода класса **cMI80**.

```
Public Class cMI80
    Private mi As Object
    Public idmap As Integer
    Friend Declare Function MoveWindow Lib "user32.dll" _
        (ByVal hwnd As Int32, ByVal x As Int32, ByVal y As Int32, _
        ByVal nWidth As Int32, ByVal nHeight As Int32, _
        ByVal bRepaint As Int32) As Boolean
    Public Sub New(ByVal mapHWND As Long)
        'mapHWND - Handle контейнера под карту
        MyBase.New()
        Try
            If mi Is Nothing Then
                mi = CreateObject("MapInfo.application")
                mi.SetCallBack(miClassCB)
            Else
                Exit Sub
            End If
            mi_CreateNewWindowMapInfo(mapHWND)
            mi.do("Set Distance Units " & kk & "m" & kk)
            mi.do("Set Area Units " & kk & "sq m" & kk)
            mi.do("Set Paper Units " & kk & "mm" & kk)
            mnuTabOpen()
            CreateMapMapperShortcut()
            idmap = mi.Eval("FrontWindow()")
        Catch ex As Exception
            MsgBox(ex.ToString, MsgBoxStyle.OkOnly, "Ошибка")
        End Try
    End Sub
...

```

Здесь переменная *mi* представляет *COM* объект *Application* из объектной модели MapInfo. В данном фрагменте кода реализуется процесс позднего связывания. На это указывают следующие признаки:

- ✓ Переменная *mi* объявлена с типом *Object*.
- ✓ Для создания объекта использована функция *CreateObject*.

Вообще для доступа к объектам *COM* сервера можно использовать функции *CreateObject* или *GetObject*. При использовании первой функции запускается новый экземпляр приложения-сервера, при применении второй используется уже работающий экземпляр приложения.

В нашем случае всегда запускается новый экземпляр сервера, но иногда возникает необходимость подключиться к работающему приложению, а если такого нет, то тогда

запустить новый экземпляр приложения. Для реализации такого поведения можно использовать следующий код.

```
Try
    mi = GetObject(, "MapInfo.application")
Catch ex As System.Exception
    mi = CreateObject("MapInfo.application")
End Try
```

В случае с MapInfo такая необходимость возникает редко, а вот при использовании Word или Excel достаточно часто.

Теперь посмотрим, как изменится рассматриваемый фрагмент кода, если реализуется раннее связывание. Во-первых, необходимо подключить соответствующую библиотеку типов, в данном случае это *MapInfo 8.0 OLE-Automation Type Library*. Для этого воспользуемся меню Visual Studio **Project/Add Reference.../COM**. Отредактированный код имеет следующий вид.

```
Public Class cMI80
    Private mi As MapInfo.MapInfoApplication
    Public idmap As Integer
    Friend Declare Function MoveWindow Lib "user32.dll" (ByVal hwnd As Int32,
        ByVal x As Int32, ByVal y As Int32, ByVal nWidth As Int32, ByVal nHeight As
        Int32, ByVal bRepaint As Int32) As Boolean
    Public Sub New(ByVal mapHWND As Long)
        MyBase.New()
        Try
            If mi Is Nothing Then
                mi = New MapInfoApplication
                mi.SetCallback(miCB)
            Else
                Exit Sub
            End If
            mi_CreateNewWindowMapInfo(mapHWND)
            mi.Do("Set Distance Units " & kk & "m" & kk)
            mi.Do("Set Area Units " & kk & "sq m" & kk)
            mi.Do("Set Paper Units " & kk & "mm" & kk)
            mnuTabOpen()
            CreateMenuMapperShortcut()
            idmap = mi.Eval("FrontWindow()")
        Catch ex As Exception
            MsgBox(ex.ToString, MsgBoxStyle.OkOnly, "Ошибка")
        End Try
    End Sub
End Class
```

Здесь следует обратить внимание на формирование объекта *mi*. Таким образом, реализовано раннее связывание и других изменений в программе не потребуется.

Остановимся на других элементах рассматриваемого кода.

Объявленная функция *Win32 API MoveWindow* будет использована в дальнейшем при обработке изменения размеров окна карты.

Процедура *mi\_CreateNewWindowMapInfo* создает новое окно карты. Формальный параметр *mapHWND* это манипулятор (*Handle*) контейнера под карту.

```
Public Sub mi_CreateNewWindowMapInfo(ByVal mapHWND As Long)
    'создать новое окно для карты (Style 1)
    Try
        Dim msg As String = "Set Application Window " & mapHWND.ToString
        mi.Do(msg)
        msg = "Set Next Document Parent " & mapHWND.ToString & " Style 1"
        mi.Do(msg)
    End Try
End Sub
```

```

Catch ex As Exception
    MsgBox(ex.ToString, MsgBoxStyle.OkOnly, "Ошибка")
End Try
End Sub

```

Процедура `mnuTabOpen` просто использует функцию меню `MapInfo` *Открыть таблицу*.

```

Public Sub mnuTabOpen()
    mi.RunMenuCommand(M_FILE_OPEN)
End Sub

```

Процедура `CreateMenuMapperShortcut` формирует «быстрое» (контекстное) меню окна карты, которое активируется после нажатия правой кнопки мыши.

```

Public Sub CreateMenuMapperShortcut()
    Dim str As String = kk & "==> Указатель" & kk & " Calling " _
        & M_TOOLS_SELECTOR.ToString & ", "
    str += kk & "==> Сдвиг" & kk & " Calling " _
        & M_TOOLS_RECENTER.ToString & ", "
    str += kk & "==> Увеличить" & kk & " Calling " _
        & M_TOOLS_EXPAND.ToString & ", "
    str += kk & "==> Уменьшить" & kk & " Calling " _
        & M_TOOLS_SHRINK.ToString & ", "
    str += kk & "(-" & kk & ", "
    str += kk & "Управление слоями..." & kk & " Calling " _
        & M_MAP_LAYER_CONTROL.ToString & ", "
    str += kk & "(-" & kk & ", "
    str += kk & "Найти выборку" & kk & " Calling " _
        & M_ANALYZE_FIND_SELECTION.ToString & ", "
    str += kk & "(-" & kk & ", "
    str += kk & "Показать по-другому..." & kk & " Calling " _
        & M_MAP_CHANGE_VIEW.ToString & ", "
    str += kk & "Показать как было" & kk & " Calling " _
        & M_MAP_PREVIOUS.ToString & ", "
    str += kk & "Показать слой полностью..." & kk & " Calling " _
        & M_MAP_ENTIRE_LAYER.ToString & ", "
    str += kk & "(-" & kk & ", "
    str += kk & "Отменить выбор" & kk & " Calling " _
        & M_ANALYZE_UNSELECT.ToString & ", "
    str += kk & "Обновить окно" & kk & " Calling " _
        & M_WINDOW_REDRAW.ToString & ", "
    str += kk & "(-" & kk & ", "
    str += kk & "Удалить косметику" & kk & " Calling " _
        & M_MAP_CLEAR_COSMETIC.ToString & ", "
    str += kk & "!Выключить Автопрокрутку^Включить Автопрокрутку" & kk _
        & " Calling " & M_MAP_AUTOSCROLL_ONOFF.ToString & ", "
    str += kk & "(-" & kk & ", "
    str += kk & "Геоинформация..." & kk & " Calling " _
        & M_EDIT_GETINFO.ToString
    mi.do("Create Menu " & kk & "MapperShortcut" & kk _
        & " ID 17 As " & kk & "(-" & kk & " ")
    mi.do("Create Menu " & kk & "MapperShortcut" & kk _
        & " ID 17 As " & str)
End Sub

```

Меню получилось несколько длинноватым в силу того что хотелось показать возможность вызова различных функций `MapInfo`. Сюда же включены вызовы некоторых инструментов (указатель, сдвиг, масштаб) которые обычно привязываются к кнопкам. Контекстные меню для других окон (списка, отчета и др.) формируются аналогично. Далее рассмотрим другие методы этого класса.

```

Public Sub mi_ResizeMapWin(ByVal panWidth As Integer, ByVal panHeight As Integer)
    Dim miWinHwnd As Int32
    Dim jj As Boolean
    miWinHwnd = Val(mi.Eval("WindowInfo(FrontWindow(), " _
        & WIN_INFO_WND.ToString & ")"))
    jj = MoveWindow(miWinHwnd, 0, 0, panWidth, panHeight, 0)
    mWinUpdate()
End Sub

```

Данная процедура обрабатывает изменения размеров окна MapInfo. Формальные параметры `panWidth` и `panHeight` отражают текущие размеры окна. Непосредственно изменение размеров реализуется функцией *Win32 API* `MoveWindow`.

Процедура `mWinUpdate` обновляет окно карты и имеет следующий вид.

```

Public Sub mWinUpdate()
    mi.RunMenuCommand(M_WINDOW_REDRAW)
End Sub

```

Весьма полезной будет процедура, которая предлагает сохранение таблиц, но не всех, а только тех, которые имеют несохраненные изменения. И хотя в данном приложении и не предполагается изменение таблиц, я посчитал, что в ваших проектах эта процедура может быть полезной.

```

Public Sub mi_WriteAllTabs_OnlyChange()
    Dim listNormal() As String
    Dim i As Integer
    Try
        ReDim listNormal(0)
        mi_CreateList_NormalLayerNames(listNormal)
        For i = 0 To UBound(listNormal)
            mi_WriteTabOnlyChange(listNormal(i))
        Next
    Catch ex As Exception
        MsgBox(ex.ToString, MsgBoxStyle.OkOnly, "Ошибка")
    End Try
End Sub

```

Процедура `mi_CreateList_NormalLayerNames` формирует список имен всех нормальных слоев карты и помещает его в массив `ListLayers`.

```

Public Sub mi_CreateList_NormalLayerNames(ByRef ListLayers() As String)
    Dim s As String
    Dim j, n, m As Integer
    Dim str As String
    Dim t As Integer = 0
    Try
        str = "MapperInfo(" & idmap.ToString & ", " _
            & MAPPER_INFO_LAYERS.ToString & ")"
        n = mi.Eval(str)
        For j = 1 To n
            str = "LayerInfo(" & idmap.ToString & ", " & j.ToString _
                & ", " & LAYER_INFO_NAME.ToString & ")"
            s = mi.Eval(str)
            str = "LayerInfo(" & idmap.ToString & ", " & j.ToString _
                & ", " & LAYER_INFO_TYPE.ToString & ")"
            m = mi.Eval(str)
            If m = LAYER_INFO_TYPE_NORMAL Then
                ReDim Preserve ListLayers(t)
                ListLayers(t) = s
                t += 1
            End If
        Next
    End Try
End Sub

```

```
        End If
    Next
    Catch ex As Exception
        MsgBox(ex.ToString, MsgBoxStyle.OkOnly, "Ошибка")
    End Try
End Sub
```

Процедура `mi_WriteTabOnlyChange` проверяет наличие изменений в отдельной таблице и если изменения есть сохраняет ее.

```
Public Sub mi_WriteTabOnlyChange(ByVal NameTab As String)
    Dim str As String
    Dim dd As String
    If NameTab Is Nothing Then Exit Sub
    Try
        str = "TableInfo(" & NameTab & ", " &
            & TAB_INFO_EDITED.ToString & ")"
        dd = mi.eval(str)
        If dd = "T" Then
            mi_WriteTab(NameTab)
        End If
    Catch ex As Exception
        MsgBox(ex.ToString, MsgBoxStyle.OkOnly, "Ошибка")
    End Try
End Sub
```

Непосредственно сохранение таблицы выполняется в процедуре `mi_WriteTab`.

```
Public Sub mi_WriteTab(ByVal NameTabl As String)
    Dim str As String
    Try
        str = "Commit Table " & NameTabl & " Automatic ApplyUpdates"
        mi.do(str)
    Catch ex As Exception
        MsgBox(ex.ToString, MsgBoxStyle.OkOnly, "Ошибка")
    End Try
End Sub
```

Также понадобится процедура закрытия всех таблиц

```
Public Sub mTabsClose()
    mi.RunMenuCommand(M_FILE_CLOSE_ALL)
End Sub
```

и процедура позволяющая выполнить оператор `MapInfo` заданный в виде строки

```
Public Sub mi_Do(ByVal s As String)
    Try
        mi.do(s)
    Catch ex As Exception
        MsgBox(ex.ToString, MsgBoxStyle.OkOnly, "Ошибка")
    End Try
End Sub
```

### Класс `comCallBk`

Создание *COM* класса начинается с выбора меню **Project/Add New Item.../COM Class**. Далее введем код, который будет состоять из двух небольших процедур.

```
Public Sub New()
    MyBase.New()
```

```

End Sub

Public Sub SetStatusText(ByVal cb As String)
    Dim Trash As String
    Trash = cb
    If Trash.Length > 0 Then
        Dim s As String = Chr(9)
        Do While Trash.LastIndexOf(s) = Trash.Length - 1
            Trash = Trash.Remove(Trash.Length - 1, 1)
        Loop
        Trash = Trash.Replace(s, " | ")
    End If
    Main.mn_StatusBar = Trash
End Sub

```

Класс **comCallBk** позволит нам реализовать функцию обратного вызова (*Callback*), а конкретнее наша программа клиент будет получать текст из строки сообщений MapInfo всякий раз, когда этот текст будет изменяться.

Клиент может получать информацию от MapInfo (через уведомление) в следующих случаях:

- Пользователь применяет инструмент в окне.
- Пользователь выбирает команду меню.
- Окно Карты изменяется.
- Изменяется текст в строке сообщений MapInfo.

### Класс Main

Класс **Main** представляет основную (и единственную) форму нашей программы. На рис. 68 показана проектируемая форма и элементы управления размещенные на ней.

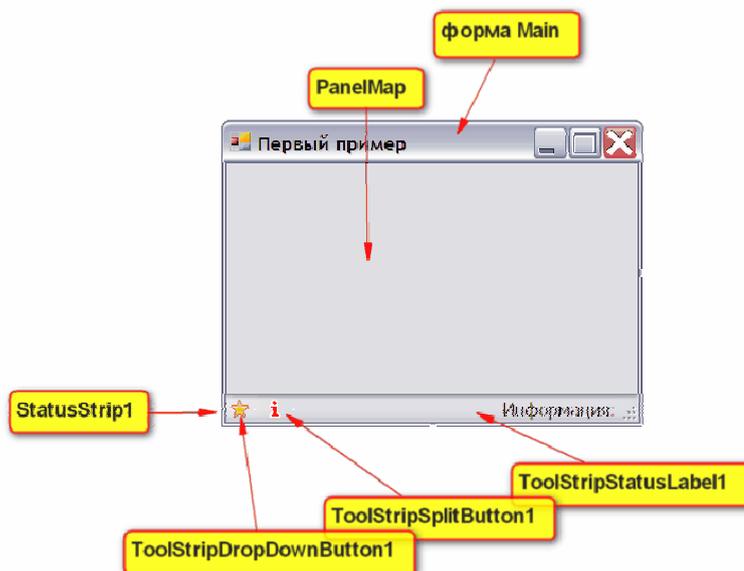


Рис. 68

Элемент управления **PanelMap** занимающий почти всю площадь формы является контейнером для окна карты MapInfo. На форме также размещен элемент управления **StatusStrip1** (строка состояния) с установленными на нем элементами **ToolStripDropDownButton1**, **ToolStripSplitButton1** и **ToolStripStatusLabel1**. В элемент **ToolStripStatusLabel1** будет выводиться текст из строки сообщений MapInfo. Здесь я хочу предложить рассмотреть два варианта получения текста сообщения в элементе **ToolStripStatusLabel1**.

### Вариант 1 (с использованием таймера)

Добавим к форме элемент управления *Timer1*.

Рассмотрим начальный фрагмент кода класса формы.

```
Public Class Main
    Public Shared mn_StatusBar As String
    Private Sub Main_Load(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles MyBase.Load
        Me.Height = 500
        Me.Width = 600
        Me.ControlBox = False
        ToolStripDropDownButton1.DropDownDirection _
            = ToolStripDropDownDirection.BelowRight
        mInfo = New cMI80(PanelMap.Handle)
        mInfo.mi_Do("Set Map Window " & mInfo.idmap.ToString _
            & " Display Zoom")
        Timer1.Interval = 100
        Timer1.Enabled = True
    End Sub

    Private Sub Timer1_Tick(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles Timer1.Tick
        ToolStripStatusLabel1.Text = mn_StatusBar
    End Sub
...

```

Обратите внимание на то, что при объявлении переменной `mn_StatusBar` использован модификатор `Shared`, а также вспомните, что в процедуре `SetStatusText` класса `comCallBk` также фигурирует эта переменная. Таким образом, строка статуса нашей программы будет, с помощью таймера, постоянно обновляться с интервалом 100 миллисекунд. Это, пожалуй, самый простой способ решения этой задачи.

### Вариант 2 (с использованием делегата)

Во первых таймер здесь не понадобится и его можно удалить. Далее добавим пару строк в модуль `Common`.

```
Public Delegate Sub SetStatus(ByVal s As String)
Public setSCB As SetStatus

```

Потребуется редакции и класс `comCallBk`. В процедуре `SetStatusText` выражение `Main.mn_StatusBar=Trash` необходимо заменить на следующее `If setSCB IsNot Nothing Then setSCB.Invoke(Trash)`.

Начальный фрагмент кода класса формы будет иметь следующий вид.

```
Public Class Main
    Private Sub Main_Load(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles MyBase.Load
        Me.Height = 500
        Me.Width = 600
        Me.ControlBox = False
        ToolStripDropDownButton1.DropDownDirection _
            = ToolStripDropDownDirection.BelowRight
        mInfo = New cMI80(PanelMap.Handle)
        mInfo.mi_Do("Set Map Window " & mInfo.idmap.ToString _
            & " Display Zoom")
        setSCB = New SetStatus(AddressOf SetStatusValue)
    End Sub

    Public Sub SetStatusValue(ByVal msgS As String)

```

```

    If msgS.Length = 0 Then Exit Sub
    ToolStripStatusLabel1.Text = msgS
End Sub

```

...

В данном случае строка статуса клиента будет обновляться в момент изменения строки сообщений на сервере.

### Процедуры обработки событий

Обработка событий для пунктов меню элемента *ToolStripDropDownButton1*.

```

Private Sub ToolStripMenuItem1_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles ToolStripMenuItem1.Click
    'открыть таблицу
    mInfo.mnuTabOpen()
End Sub

Private Sub ToolStripMenuItem2_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles ToolStripMenuItem2.Click
    'закреть программу
End
End Sub

```

Обработка событий для пунктов меню элемента *ToolStripSplitButton1*.

```

Private Sub ToolStripMenuItem3_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles ToolStripMenuItem3.Click
    'статус: ширина
    If mInfo IsNot Nothing Then
        mInfo.mi_Do("Set Map Window " & mInfo.idmap.ToString _
            & " Display Zoom")
    End If
End Sub

Private Sub ToolStripMenuItem4_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles ToolStripMenuItem4.Click
    'статус: масштаб
    If mInfo IsNot Nothing Then
        mInfo.mi_Do("Set Map Window " & mInfo.idmap.ToString _
            & " Display Scale")
    End If
End Sub

Private Sub ToolStripMenuItem5_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles ToolStripMenuItem5.Click
    'статус: координаты
    If mInfo IsNot Nothing Then
        mInfo.mi_Do("Set Map Window " & mInfo.idmap.ToString _
            & " Display Position")
    End If
End Sub

```

Обработка события изменение размеров элемента управления *PanelMap*.

```

Private Sub PanelMap_Resize(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles PanelMap.Resize
    Try
        Dim w As Integer = PanelMap.DisplayRectangle.Width
        Dim h As Integer = PanelMap.DisplayRectangle.Height
        If mInfo IsNot Nothing Then
            mInfo.mi_Do("Set Window " & mInfo.idmap.ToString _

```

```

        & " Front")
        mInfo.mi_ResizeMapWin(w, h)
    End If
Catch ex As Exception
    MsgBox(ex.ToString, MsgBoxStyle.OkOnly, "Ошибка")
End Try
End Sub

```

Обработка события закрытие формы.

```

Private Sub fMain_FormClosing(ByVal sender As Object, _
    ByVal e As System.Windows.Forms.FormClosingEventArgs) _
    Handles Me.FormClosing
    mInfo.mi_WriteAllTabs_OnlyChange()
    mInfo.mTabsClose()
    mInfo = Nothing
End Sub

```

Таким образом, мы закончили построение приложения использующего механизм интегрированной картографии. На рис. 69 показан вид работающего приложения с раскрытым контекстным меню.

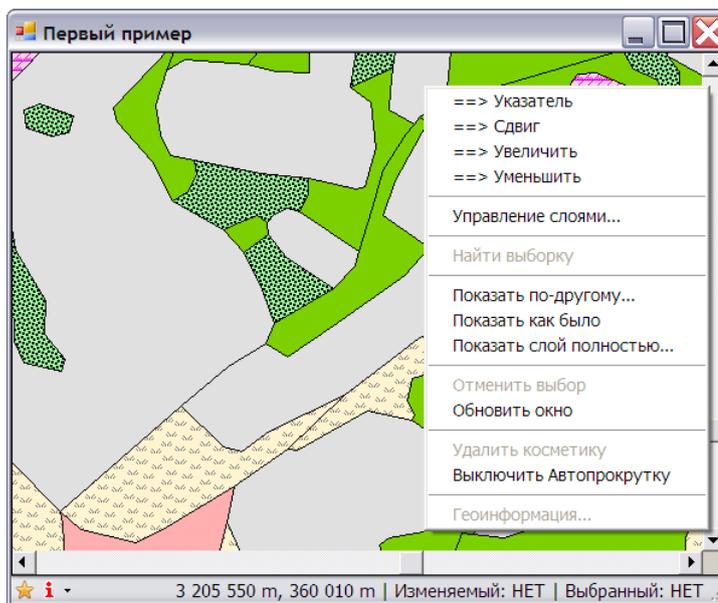


Рис. 69

### Тема 57. Использование MapBasic программ в приложениях с интегрированной картографией

При создании приложения с интегрированной картографией можно, а иногда и необходимо, использовать MapBasic программы. При этом можно выделить два случая.

1. Приложение клиент запускает *tbx-программу* с целью одноразового выполнения некоторой объемной процедуры. Все параметры программы определяются заранее при ее разработке.
2. Приложение клиент запускает *tbx-программу* и взаимодействует с ней через объекты *MapBasicApplication* и *MBGlobal*.

MapBasic программы, предназначенные для использования в рамках комплекса интегрированной картографии, обычно не имеют таких элементов управления как меню или инструментальные панели. Есть и другие особенности.

Программы, написанные на MapBasic, для первого случая можно схематично представить в следующем виде.

```
'Блок описаний
...
Sub Main
call BigProcedure
end sub

Sub BigProcedure
...
end sub

'Вспомогательные процедуры и функции
...
```

Второй случай разберем более подробно и для этого используем построенное ранее приложение с интегрированной картой (Тема 56). Рассматриваемый пример не имеет какой-либо практической ценности, его основная задача показать некоторые приемы использования MapBasic программ в комплексе интегрированной картографии. Начнем с MapBasic программы (**p25.mb**), ее код представлен ниже.

```
Include "MAPBASIC.DEF"
Declare Sub Main
Declare Sub tst1
Declare Sub tst2
Declare Function RemoteQueryHandler() As String
Declare Sub RemoteMsgHandler
Global glTest as string
dim rr as string

Sub Main
glTest=""
end sub

Sub tst1
dim ar as float
OnError Goto theErr
ar=Area(selection.obj,"sq m")
rr=str$(ar)
exit sub
theErr:
rr="не определено"
end sub

Sub tst2
dim ar as float
OnError Goto theErr
ar=Perimeter(selection.obj,"m")
rr=str$(ar)
exit sub
theErr:
rr="не определено"
end sub

Sub RemoteMsgHandler
Dim ss As String
ss=CommandInfo(CMD_INFO_MSG)
Run Command ss
End Sub

Function RemoteQueryHandler() As String
Dim ss As String
ss=CommandInfo(CMD_INFO_MSG)
```

```

If ss="T" Then
    call tst1
    glTest="Вариант ПЛОЩАДЬ"
Else
    call tst2
    glTest="Вариант ПЕРИМЕТР"
End If
RemoteQueryHandler=rr
End Function

```

Как видно из кода процедура `Main` не выполняет никакой значимой работы. После запуска программы отработает процедура `Main`, и программа перейдет в режим ожидания вызова от приложения клиента. Эти вызовы обрабатываются обработчиками событий `RemoteMsgHandler` и `RemoteQueryHandler`.

Рассмотрим `RemoteMsgHandler`. В локальную переменную `ss` передается от клиента строка, представляющая собой оператор `MapBasic` который затем выполняется. Вероятно, здесь имеет смысл включить в процедуру механизм обработки ошибок, так как передаваемый оператор может быть некорректным.

Функция `RemoteQueryHandler` предлагает большее многообразие вариантов использования. В переменную `ss` от клиента передается строка, которая может быть критерием выбора, как в данном случае, или параметром процедуры, или чем-либо другим оказывающим влияние на процесс обработки. Функция возвращает строку `rr` которая является общей для модуля. Кроме того дополнительная информация о процессе будет передаваться через глобальную переменную `glTest`.

Теперь перейдем к приложению клиенту. Здесь потребуется внести некоторые изменения в модуль **Common** и классы **Main** и **сМ180**.

В модуль **Common** добавим объявление `Public mbx As Object`. Этот объект будет представлять программу на `MapBasic`.

К коду класса **сМ180** добавим следующие методы.

```

Public Sub setMBX(ByVal name As String)
    mbx = mi.MBApplications.item(name)
End Sub

Public Sub tstZoomMap(ByVal id As Integer, ByVal width As Integer)
    mbx.do("Set Map Window " & id.ToString & " Zoom " & _
width.ToString & " Units " & kk & "m" & kk)
End Sub

Public Sub tstScaleMap(ByVal id As Integer, ByVal T As Integer)
    'T - знаменатель численного масштаба
    mbx.do("Set Map Window " & id.ToString & " Scale 1 " & " Units " & _
kk & "cm" & kk & " For " & T.ToString & " Units " & kk & _
"cm" & kk)
End Sub

Public Sub tstCenterMap(ByVal id As Integer, ByVal x As Double, _
ByVal y As Double)
    'координаты геодезические (x - на север, y - на восток)
    mbx.do("Set Map Window " & id.ToString & " Center(" & _
y.ToString & "," & x.ToString & ")")
End Sub

Public Sub tstAreaPerimeter(ByVal TF As String)
    Dim rr As String = mbx.eval(TF)
    Dim vGlob As Object = mbx.MBGlobals.Item("glTest")
    rr = vGlob.Value & " - значение=" & rr
Debug.Print(rr)
End Sub

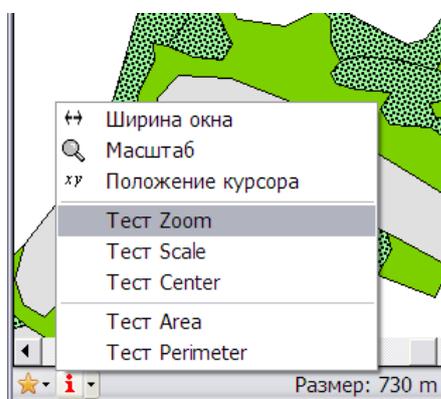
```

```
Public Sub tstSetCoordSys()
    mbx.do("Set CoordSys Nonearth Units " & kk & "m" & kk & _
" Bounds(0,0) (1000000,5000000)")
End Sub
```

Перейдем к редактированию класса **Main**. К методу `Main_Load` добавим три строки кода.

```
mInfo.mi_Do("Run Application " & kk & "C:\PROG\p25.MBX" & kk)
mInfo.setMBX("p25")
mInfo.tstSetCoordSys()
```

Таким образом, при загрузке основной формы будет запущена программа **p25.mbx**, а объект `mbx` будет представлять объект *MapBasicApplication* или конкретнее **p25.mbx**. Далее внесем изменения в саму форму. В коллекцию **DropDownItems** элемента управления *ToolStripSplitButton1* добавим пять новых элемента типа *ToolStripMenuItem*. Новый вид списка показан на рис. 70.



**Рис. 70**

Приведем код обработчиков события *Click* для этих элементов.

```
Private Sub ToolStripMenuItem6_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles ToolStripMenuItem6.Click
    'Тест Zoom
    mInfo.tstZoomMap(mInfo.idmap, 1000)
End Sub

Private Sub ToolStripMenuItem7_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles ToolStripMenuItem7.Click
    'Тест Scale
    mInfo.tstScaleMap(mInfo.idmap, 10000)
End Sub

Private Sub ToolStripMenuItem9_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles ToolStripMenuItem9.Click
    'Тест Center
    mInfo.tstCenterMap(mInfo.idmap, 362800, 3204600)
End Sub

Private Sub ToolStripMenuItem10_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles ToolStripMenuItem10.Click
    'Тест Area
    mInfo.tstAreaPerimeter("T")
End Sub

Private Sub ToolStripMenuItem11_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles ToolStripMenuItem11.Click
```

```
'Тест Perimeter
mInfo.tstAreaPerimeter("F")
End Sub
```

Примеры вывода результатов для методов `ToolStripMenuItem10_Click` и `ToolStripMenuItem11_Click` показаны на рис. 71.

```
Вариант ПЛОЩАДЬ - значение=не определено
Вариант ПЛОЩАДЬ - значение=25619.52
Вариант ПЕРИМЕТР - значение=376.947
```

Рис. 71

### Тема 58. Формирование отчетов, включающих карты

Построение системы формирования и вывода документов требуют значительных затрат времени и сил разработчиков. Особенно если такое построение выполняется что называется с чистого листа. Обычно дело сводится к использованию внешних библиотек с более или менее укрупненными функциями формирования документа требуемого формата.

В последнее время для этих целей часто применяется технология *COM* для использования объектных моделей приложений MS Office. И в данном примере будет использоваться приложение MS Word в качестве *COM-сервера*.

Так как только описание объектной модели MS Word может занять не один десяток страниц, мы рассмотрим лишь наиболее общие подходы использования объектов Word на конкретном примере. Наш пример на формирование документа с картой будет базироваться на коде, размещенном в Теме 56.

Итак, имеется проект приложения с интегрированной картой открытый в *Visual Studio*. Для объекта MS Word будем использовать процедуру раннего связывания<sup>51</sup>, то есть к проекту необходимо сразу подключить соответствующую библиотеку типов (*Microsoft Word 11.0 Object Library*<sup>52</sup>). Подключение выполняем через меню *Visual Studio Project/Add Reference.../COM*.

Добавим к проекту новый модуль **forWord** в котором разместим все процедуры связанные с объектом MS Word.

#### Модуль forWord

```
Imports Microsoft.Office.Interop.Word
Module forWord
    Public W As Application
    Public D As Document
    Public Wind As Window
    Private wdAlert As Integer

    Public Function WrdStart() As Boolean
'Начало работы с Word
        Try
            W = GetObject(, "Word.Application")
        Catch ex As Exception
            W = New Application
        End Try
        If Not isWrdVersion11() Then
```

<sup>51</sup> Можно использовать и позднее связывание. В этом случае потребуется создать соответствующий класс также как это было в случае с подключением `MapInfo`. Вид сформированного документа Word не зависит от типа связывания.

<sup>52</sup> В данном случае речь идет о MS Word 2003.

```

        W = Nothing
        Return False
    End If
    D = New Document
    wdAlert = W.DisplayAlerts
    WSetBaseSetting(False)
    Return True
End Function

Public Function isWrdVersion11() As Boolean
'Проверка версии Word (если 11 [Word 2003] то TRUE)
    Dim bb As Boolean
    If CInt(W.Version) = 11 Then
        bb = True
    Else
        bb = False
    End If
    Return bb
End Function

Private Sub WSetBaseSetting(ByVal State As Boolean)
'Установки для объекта Application
    If State Then
        W.DisplayAlerts = wdAlert
    Else
        W.DisplayAlerts = WdAlertLevel.wdAlertsNone
    End If
    W.Options.CheckSpellingAsYouType = State
    W.Options.CheckGrammarAsYouType = State
    W.Options.CheckGrammarWithSpelling = State
    W.Visible = State
End Sub

Public Function WCreateNewDocFromDot(ByVal wdot As String) As Boolean
'Создать документ по шаблону wdot
    If wdot.Length = 0 Then Return False
    If Not IO.File.Exists(wdot) Then
        Return False
    End If
    D = W.Documents.Add(wdot, False, _
        dNewDocumentType.wdNewBlankDocument, Visible:=True)
    Wind = W.ActiveWindow
    Return True
End Function

Public Sub WFindReplace(ByVal ss As String, ByVal aText As String)
'Замена строки на другую строку (производится во всем документе)
'ss - что менять; aText - на что менять;
'есть ограничение на длину строки (255)
    If aText.Length > 255 Then Exit Sub
    With D.Content.Find
        .ClearFormatting()
        .Text = ss
        With .Replacement
            .ClearFormatting()
            .Text = aText
        End With
        .Execute(Replace:=WdReplace.wdReplaceAll)
    End With
End Sub

Public Sub GetWHcell(ByRef cWmm As Double, ByRef cHmm As Double, _

```

```

        ByVal bmk As String)
'Позволяет получить размеры контейнера bmk
    Dim rngW As Object
    rngW = D.Bookmarks.Item(bmk).Range
    rngW.cells.PreferredWidthType = _
        WdPreferredWidthType.wdPreferredWidthPoints
    cWmm = rngW.cells.PreferredWidth
    cWmm = W.PointsToMillimeters(cWmm)
    rngW.cells.HeightRule = WdRowHeightRule.wdRowHeightExactly
    cHmm = rngW.cells.Height
    cHmm = W.PointsToMillimeters(cHmm)
End Sub

    Public Sub InsertDrawing_MapScale(ByVal bmk As String, _
        ByVal flgScale As Boolean, ByVal resolution As Integer)
'Экспорт окна карты. Вставка ее в закладку bmk.
'flgScale=TRUE - вставлять с масштабом окна карты
'flgScale=FALSE - вставлять в контейнера bmk с сохранением пропорций
    Dim idMWin As Integer = 0
'Ширина, высота окна карты в мм
    Dim Wscr, Hscr As Double
'Масштаб (число метров в 1 мм), ширина окна в м
    Dim SCscr, WMscr As Double
    Dim rrErr As Boolean
    Dim fMap As String = ""
    'определить окно карты
    idMWin = mInfo.idmap
    If idMWin <= 0 Then
        MsgBox("Нет карты для вставки в закладку - " & bmk)
        Exit Sub
    End If
'Прочитать информацию по окну карты
    mInfo.pWindowInfo(idMWin, Wscr, Hscr, SCscr, WMscr)
'Сохранить копию окна карты
    Dim pthDATA As String = "c:\Temp\"
    mInfo.SaveTmpRastr(idMWin, fMap, "png", CInt(Math.Round(Wscr, 0)), _
        CInt(Math.Round(Hscr, 0)), resolution, 0, rrErr, pthDATA)
    If rrErr Then Exit Sub
    Dim cWmm As Double
    Dim cHmm As Double
'Размеры контейнера (мм) для закладки bmk
    GetWHcell(cWmm, cHmm, bmk)
    Dim rngW As Object = D.Bookmarks.Item(bmk).Range
    Dim inLineSh As Object
    Dim wp As Double = W.MillimetersToPoints(cWmm)
    Dim hp As Double = W.MillimetersToPoints(cHmm)
    inLineSh = rngW.InlineShapes.AddPicture(FileName:=fMap, _
        LinkToFile:=False, SaveWithDocument:=True)
    inLineSh.ScaleWidth = 100
    inLineSh.ScaleHeight = 100
    inLineSh.LockAspectRatio = -1 'msoTrue
    Dim ww As Single = inLineSh.Width
    Dim hh As Single = inLineSh.Height
    Dim kR As Single = wp / hp
    Dim k As Single = ww / hh
    If Not flgScale Then
'Картинка будет вписана в область bmk с сохранением пропорций
        If kR < k Then
            inLineSh.Width = wp
            inLineSh.ScaleHeight = inLineSh.ScaleWidth
        Else

```

```

        inLineSh.Height = hp
        inLineSh.ScaleWidth = inLineSh.ScaleHeight
    End If
    System.Windows.Forms.Application.DoEvents()
End If
If flgScale Then
'Картинка будет вписана в область bmk с сохранением масштаба
    Dim mScInt As Integer
    Dim mSc0 As Double = SCscr * 1000
    mScInt = CInt(Math.Round(mSc0, 0))
    inLineSh.ScaleHeight = 100
    inLineSh.ScaleWidth = 100
    If bmk = "Карта" Then
        WFindReplace("ЁМасштабЁ", "Масштаб 1:" & mScInt.ToString)
        Dim dd As Double = CDbl(mScInt) / 100
        WFindReplace("ЁМасштаб_в_смЁ", "B 1 см " & _
            & Format(dd, "#0.##") & " метров")
    End If
End If
D.Bookmarks("Карта").Delete()
W.ActiveWindow.ActivePane.View.Zoom.PageFit = _
    WdPageFit.wdPageFitFullPage
IO.File.Delete(fMap)
End Sub

```

## Класс cMI80

Добавим в класс cMI80 две процедуры.

```

    Public Sub pWindowInfo(ByVal idmapWin As Integer, ByRef mWmm As Double, _
        ByRef mHmm As Double, ByRef mScale As Double, ByRef mWm As Double)
'Возвращает параметры окна карты с заданным идентификатором
        Dim str As String
        str = "windowinfo(" & idmapWin.ToString & "," & _
            & WIN_INFO_HEIGHT.ToString & ")"
        mHmm = mi.eval(str)
        str = "windowinfo(" & idmapWin.ToString & "," & _
            & WIN_INFO_WIDTH.ToString & ")"
        mWmm = mi.eval(str)
        str = "Mapperinfo(" & idmapWin.ToString & "," & _
            & MAPPER_INFO_SCALE.ToString & ")"
        mScale = mi.eval(str)
        str = "Mapperinfo(" & idmapWin.ToString & "," & _
            & MAPPER_INFO_ZOOM.ToString & ")"
        mWm = mi.eval(str)
    End Sub

    Public Sub SaveTmpRastr(ByVal idmapWin As Integer, _
        ByRef fileR As String, ByVal ext As String, ByVal w As Integer, _
        ByVal h As Integer, ByVal Res As Integer, ByVal Brd As Integer, _
        ByRef rErr As Boolean, ByVal pthDATA As String)
'Запись карты (растр) на диск во временный файл
        Dim str, s As String
        rErr = False
        fileR = pthDATA & "tmpImage" & "." & ext
        If Brd = 0 Then
            s = "Off"
        Else
            s = "On"
        End If
        str = "Set Window " & idmapWin.ToString & " Export Border " & s & _
            " TrueColor On Dither Halftone Transparency Raster ROP Vector ROP"
    End Sub

```

```

str = "Save Window " & idmapWin.ToString & " As " & kk & fileR & kk _
    & " Type " & kk & ext & kk & " Resolution " & Res.ToString
If w > 0 And h > 0 Then
    str += " Width " & w & " Units " & kk & "mm" & kk
    str += " Height " & h & " Units " & kk & "mm" & kk
End If
Try
    mi.do(str)
Catch ex As Exception
    rErr = True
    MsgBox(ex.Message, , "ОШИБКА")
End Try
End Sub

```

## Класс Main

В коллекцию **DropDownItems** объекта `ToolStripDropDownButton1` добавим новый элемент `ToolStripMenuItem6` с текстом меню «Подготовить ПЛАН» (Рис. 72).



Рис. 72

Обработчик события `Click` для этого пункта меню имеет вид

```

Private Sub ToolStripMenuItem6_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles ToolStripMenuItem6.Click
If Not WrdStart() Then
    MsgBox("Ошибка подключения Word!", , "ОШИБКА")
    Exit Sub
End If
If Not WCreateNewDocFromDot("C:\pattern\ПЛАН_A4P.dot") Then
    MsgBox("Не найден файл шаблона!", , "ОШИБКА")
    Exit Sub
End If
WFindReplace("ЁСегодняЁ", Format(Date.Now, "Long Date"))
InsertDrawing_MapScale("Карта", True, 300)
WrdEnd()
End Sub

```

Это все изменения в программе необходимые для формирования документа Word на основе шаблона **ПЛАН\_A4P.dot**. Шаблон и документ, построенный на его основе, показаны на рис. 73. Документ отображает карту в тех же границах и в том же масштабе как она представлена в окне приложения. В случае если габариты изображения не размещаются в документе, часть изображения будет скрыта, масштаб не изменяется. В используемом шаблоне область карты определяется закладкой с именем **Карта**. Это, пожалуй, простейший вариант. Более правильным будет решение использовать для этих целей текстовое поле,<sup>53</sup> в котором можно хранить коды необходимых операций с изображением карты (масштабирование, размеры, включение рамки и т.д.). Использование приложений MS Office в качестве *СОМ-сервера* это отдельная большая тема, которой можно посвятить целую книгу. Здесь же лишь кратко показано как это вообще может выглядеть. Созданный документ достаточно прост, но в реальных проектах документы могут быть весьма сложными. Для формирования объемного и сложного

<sup>53</sup> Вставить текстовое поле можно через панель инструментов «Формы» (меню Вид/Панели инструментов/Формы) кнопка «Текстовое поле».

документа с множеством таблиц и рисунков обычно используется не один, а группа шаблонов<sup>54</sup>. Окончательный вид такого документа определяется, как результат слияния нескольких частей, каждая из которых построена на своем шаблоне.

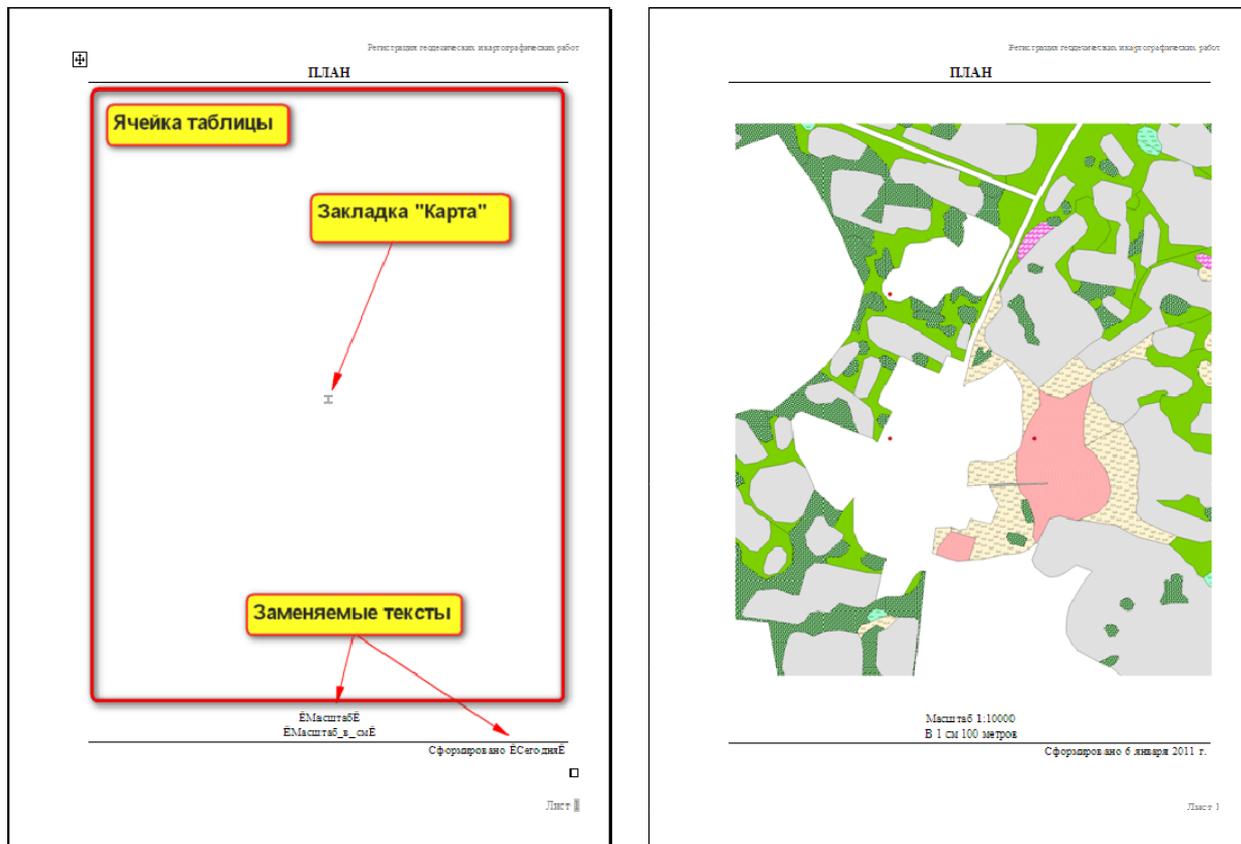


Рис. 73

### Тема 59. Использование процедур NET в MapBasic программах

Возможность использования процедур NET появилась относительно недавно с версии 9.5 MapBasic. Описание оператора выглядит следующим образом<sup>55</sup>

```
Declare Method fname Class "class_name" Lib "assembly_name"
[ Alias function_alias ]
( [ [ ByVal ] parameter As var_type ]
[, [ ByVal ] parameter As var_type... ] ) [ As return_type ]
```

Практическое использование данного средства включает два этапа:

1. Формирование библиотеки процедур в *Visual Studio*.
2. Использование процедур библиотеки в коде MapBasic.

#### Формирование библиотеки

Сформируем в *Visual Studio* новый проект *Class Library*. В классе разместим три процедуры.

```
mbMessageBox | Эмулирует работу функции MessageBox.
mbSortArray  | Демонстрация сортировки массива строк.
```

<sup>54</sup> Для управления шаблонами может понадобиться справочник шаблонов.

<sup>55</sup> MapBasic Version 9.5 Supplement

mbSetForm | Демонстрация использования формы.

Следует обратить внимание, что методы формируются как статические (Shared). Такой метод можно вызывать без предварительного создания объекта данного класса. Имеются и ограничения, код внутри статического метода может получать доступ к другим статическим членам, но для него недоступны поля, свойства и методы экземпляров.

```
Namespace forMbDLL
    Public Class forMB

        Public Shared Function mbMessageBox(ByVal text As String, _
            ByVal caption As String, ByVal buttons As Integer, _
            ByVal icon As Integer, ByVal defaultButton As Integer) As Integer
            Dim Ebuttons As MessageBoxButtons = CType(buttons, _
                MessageBoxButtons)
            Dim Eicon As MessageBoxIcon = CType(icon, MessageBoxIcon)
            Dim EdefaultButton As MessageBoxDefaultButton = _
                CType(defaultButton, MessageBoxDefaultButton)
            Dim dr As DialogResult = _
                System.Windows.Forms.MessageBox.Show(text, _
                    caption, Ebuttons, Eicon, EdefaultButton)
            Return CType(dr, Integer)
        End Function

        Public Shared Function mbSortArray(ByVal ss() As String) As Boolean
            Array.Sort(ss)
            Return True
        End Function

        Public Shared Sub mbSetForm(ByVal s As String)
            Dim myForm As New theForm
            myForm.Width = 407
            myForm.Height = 225
            myForm.Text = "myTest"
            myForm.MinimizeBox = False
            myForm.MaximizeBox = False
            myForm.FormBorderStyle = _
                Windows.Forms.FormBorderStyle.FixedSingle
            myForm.WindowState = FormWindowState.Normal
            myForm.StartPosition = FormStartPosition.CenterScreen
            myForm.BackgroundImage = My.Resources.f2
            myForm.BackgroundImageLayout = ImageLayout.Stretch
            myForm.Icon = My.Resources.DF
            Dim Label1 As Label = Nothing
            Label1 = New Windows.Forms.Label
            Label1.AutoSize = False
            Label1.BackColor = Drawing.Color.Transparent
            Label1.Location = New Drawing.Point(0, 0)
            Label1.Dock = DockStyle.Left
            Label1.Font = New Drawing.Font("Mistral", 72)
            Label1.ForeColor = Drawing.Color.Navy
            Label1.Image = My.Resources.icon_250
            Label1.ImageAlign = Drawing.ContentAlignment.MiddleCenter
            Label1.Size = New Drawing.Size(187, 187)
            Label1.Text = "Test"
            Label1.TextAlign = Drawing.ContentAlignment.MiddleCenter
            myForm.Controls.Add(Label1)
            Dim Label2 As Label = Nothing
            Label2 = New Windows.Forms.Label
            Label2.AutoSize = True
            Label2.BackColor = Drawing.Color.Transparent
            Label2.Location = New Drawing.Point(210, 53)
```

```

        Label2.Text = s
        myForm.Controls.Add(Label2)
        myForm.ShowDialog()
    End Sub
End Class
Public Class theForm
    Inherits Form
End Class
End Namespace

```

Не забудьте подключить дополнительно библиотеки *System.Windows.Forms* и *System.Drawing*. В свойствах проекта определено *Assembly name* как *forMapBasic* и в ресурсы добавлено два растра и иконка. После компиляции и компоновки проекта (**Build/Build Solution**) получим файл библиотеки **forMapBasic.dll**.

### Использование процедур библиотеки

Код тестовой MapBasic программы демонстрирующей использование процедур библиотеки **forMapBasic.dll** приводится ниже.

#### Блок описаний

```

Include "MAPBASIC.DEF"
Include "ICONS.DEF"
'Константы для mbMessageBox
define DialogResult_Abort           3
define DialogResult_Cancel         2
define DialogResult_Ignore        5
define DialogResult_No            7
define DialogResult_None          0
define DialogResult_Ok            1
define DialogResult_Retry         4
define DialogResult_Yes           6
define MessageBoxButtons_Ok       0
define MessageBoxButtons_OkCancel 1
define MessageBoxButtons_RetryCancel 5
define MessageBoxButtons_YesNo    4
define MessageBoxButtons_YesNoCancel 3
define MessageBoxButtons_AbortRetryIgnore 2
define MessageBoxIcon_Error       16
define MessageBoxIcon_Information  64
define MessageBoxIcon_None        0
define MessageBoxIcon_Question    32
define MessageBoxIcon_Warning     48
define MessageBoxDefaultButton_Button1 0
define MessageBoxDefaultButton_Button2 256
define MessageBoxDefaultButton_Button3 512
'Описание процедур
Declare Sub Main
Declare Sub RuntestNET1
Declare Sub RuntestNET2
Declare Sub RuntestNET3
Declare function GetCurrentDir() as string
Declare Sub SetCurrentDir(byval PathStr as string)
Declare Sub exitMe
'Описание Net-метода включает имя метода, полное имя класса,
'включая пространство имен (namespace) и имя библиотеки. В данном случае
'библиотека размещается в одной папке с mbx-программой.
Declare Method mbMessageBox
    Class "forMbDLL.forMB" Lib "forMapBasic.dll"
        (ByVal text As String, ByVal caption As String,

```

```
    ByVal buttons As Integer, ByVal icon As Integer,  
    ByVal defaultButton As Integer) as Integer  
Declare Method mbSortArray  
    Class "forMbdLL.forMB" Lib "forMapBasic.dll"  
        (ss() As String) as logical  
Declare Method mbSetForm  
    Class "forMbdLL.forMB" Lib "forMapBasic.dll"  
        (ByVal s As String)
```

### **Главная процедура**

```
Sub Main  
Create ButtonPad "Тест" As  
    PushButton  
        Icon MI_ICON_NUMBERS_2  
        Calling RuntestNET1  
        HelpMsg "Тест1\nТест1"  
        Enable  
    PushButton  
        Icon MI_ICON_NUMBERS_3  
        Calling RuntestNET2  
        HelpMsg "Тест2\nТест2"  
        Enable  
    PushButton  
        Icon MI_ICON_NUMBERS_4  
        Calling RuntestNET3  
        HelpMsg "Тест3\nТест3"  
        Enable  
    Separator  
    PushButton  
        Icon MI_ICON_ARROW_17  
        Calling exitMe  
        HelpMsg "Выход из программы\nВыход"  
        Enable  
    Width 5  
    Position (0,5) Units "cm"  
    Show  
    Float  
End Sub
```

### **Прочие процедуры**

```
Sub RuntestNET1  
'Тест MessageBox  
dim tt,cc As String  
dim bt,ic,dbt As Integer  
dim r as Integer  
dim rr as string  
tt="Тест!"  
cc="ВНИМАНИЕ"  
bt=MessageBoxButtons_YesNoCancel  
ic=MessageBoxIcon_Question  
dbt=MessageBoxDefaultButton_Button2  
r=mbMessageBox(tt,cc,bt,ic,dbt)  
Do Case r  
    Case DialogResult_Abort  
        rr="DialogResult_Abort"  
    Case DialogResult_Cancel  
        rr="DialogResult_Cancel"  
    Case DialogResult_Ignore  
        rr="DialogResult_Ignore"  
    Case DialogResult_No  
        rr="DialogResult_No"
```

```

Case DialogResult_None
    rr="DialogResult_None"
Case DialogResult_Ok
    rr="DialogResult_Ok"
Case DialogResult_Ok
    rr="DialogResult_Ok"
Case DialogResult_Yes
    rr="DialogResult_Yes"
Case Else
    rr="Результат не определен!"
End Case
print rr
End Sub

Sub RuntestNET2
'Тест сортировка массива
dim ss(5) as string
ss(1)="frwe"
ss(2)="njje"
ss(3)="aewe"
ss(4)="qwee"
ss(5)="aswe"
onError Goto www
print("BEFORE= " & ss(1) & ", " & ss(2) & ", " & ss(3) & ", "
    & ss(4) & ", " & ss(5))
dim bb as logical
bb=mbSortArray(ss)
print ("SORT= " & ss(1) & ", " & ss(2) & ", " & ss(3) & ", "
    & ss(4) & ", " & ss(5))
note "ok"
exit sub
www:
print Error$()
End Sub

Sub RuntestNET3
'Тест форма
call mbSetForm("Новая форма")
End Sub

Sub exitMe
'Выход из программы
End Program
end sub

```

На рис. 74 показаны результаты работы тестовых процедур.

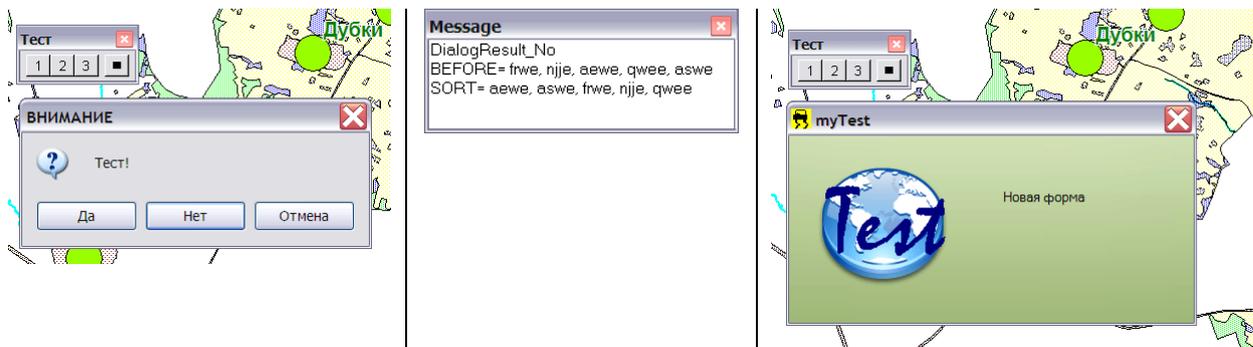


Рис. 74

Отметим некоторые особенности в использовании процедур *NET* в MapBasic.

Во-первых, версия MapInfo должна соответствовать версии MapBasic, во-вторых, версия *Visual Studio* также имеет значение, так для MapBasic версии 10 требуется *Visual Studio* версии не выше *VS2008* (сборка, подготовленная в *VS2010*, работать с MapBasic 10 не будет). В данном проекте используются MapInfo и MapBasic версии 10, и *Visual Studio 2008*.

Если сборка зарегистрирована в глобальном кэше сборок (GAC<sup>56</sup>), MapInfo будет загружать ее из GAC. В противном случае сборка будет загружаться из той же директории, в которой находится вызывающий ее *mbx*-файл. В этой связи нужно сказать, что можно использовать не только пользовательские процедуры, но и процедуры из библиотеки классов NET Framework. Условие остается одно – процедуры должны быть статические (Shared)<sup>57</sup>. Приведем пример из документации MapBasic демонстрирующий описание таких процедур.

```
' Метод из библиотеки System.Windows.Forms.dll
Declare Method Show
Class "System.Windows.Forms.MessageBox"
Lib "System.Windows.Forms, Version=2.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089"
(ByVal str As String, ByVal caption As String)
' Метод из библиотеки mscorlib.dll
Declare Method Move
Class "System.IO.File"
Lib "mscorlib, Version=2.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089"
(ByVal sourceFileName As String, ByVal destFileName As String)
```

Строгое имя сборки можно получить, если воспользоваться утилитой **gacutil.exe** входящей в состав *Visual Studio*. На рис. 75 показан этот процесс для **mscorlib.dll**.

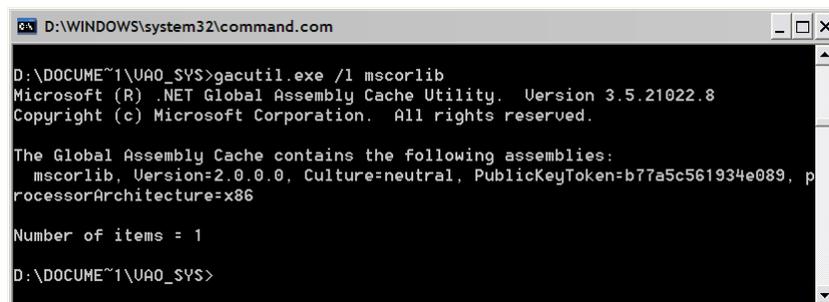


Рис. 75

## Тема 60. Примеры компоновки приложения

Приложения с интегрированной картографией могут иметь различную компоновку, в смысле размещения окон MapInfo. Ранее мы рассмотрели простое приложение, имеющее одно окно карты встроенное в главную форму. Но такая компоновка далеко не единственный вариант и далее будет кратко рассмотрено несколько иных решений.

### Пример 1

Приложение позволяет читать информацию из *XML* файла, представляющего собой кадастровый план территории (*КИПТ*) и формировать на базе этой информации таблицы MapInfo и текстовый отчет.

<sup>56</sup> Global Assembly Cache (размещение \WINDOWS\assembly)

<sup>57</sup> Таких процедур достаточно много, например это все процедуры класса Math.

Загрузка и обработка *XML* файла производится в соответствии с имеющейся *XML-схемой* документа. Если быть более конкретным, то на базе *XML-схемы* создается класс. Объект этого класса, после процесса десериализации *XML* файла, будет содержать всю информацию обрабатываемого *КПТ*. Далее этот объект используется для построения таблиц *MapInfo* (**Участки**, **Точки**, **ОМС** (опорно-межевая сеть)) и текстового отчета. На рис. 76 показана главная форма программы на этапе проектирования. Элемент управления *PanelMap* служит контейнером для окна карты, а в элементе *PanelList* в зависимости от нажатой кнопки будут отображаться или список для таблицы **Участки**, или список **Точки** или **ОМС**. Текстовый отчет будет формироваться в элементе управления *RichTextBox1*. Элементы управления картой упрятаны в контекстное меню.

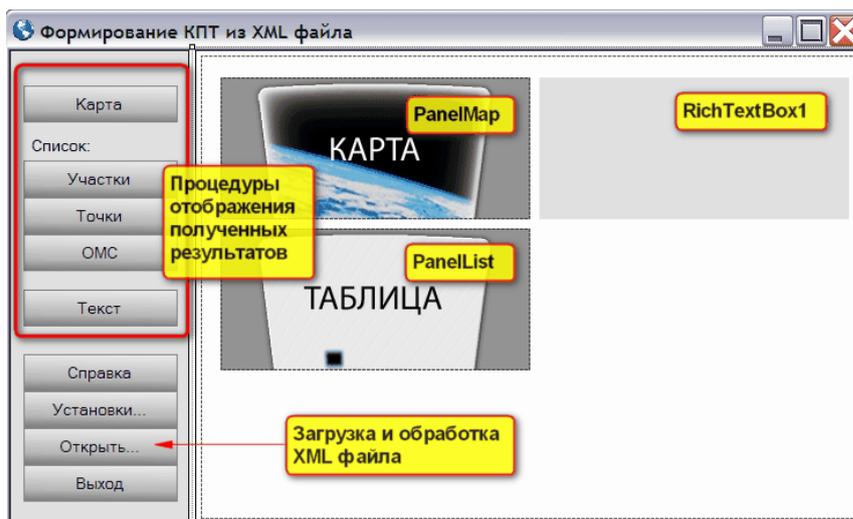


Рис. 76

На рис. 77 показан вид работающей программы с открытым окном карты.

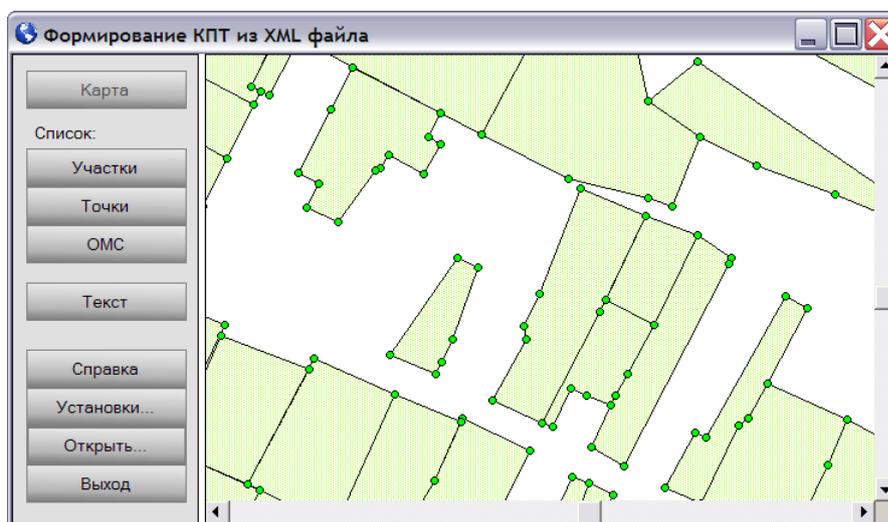


Рис. 77

## Пример 2

На главной форме приложения имеется элемент *TabControl* состоящий из трех объектов *TabPage*, представляющих отдельные вкладки в составе элемента управления (Рис. 78). Каждая вкладка представляет одно окно карты и связана с одной темой. Под темой понимается набор слоев и связанные с ними стили оформления. Такая тема может быть легко создана и сохранена. Все доступные темы представлены в списке на левой части формы и легко могут быть загружены в активное окно.

Такая конструкция весьма удобна в следующих случаях:

- ✓ Представление одного и того же участка местности в разных масштабах или с разным набором слоев.
- ✓ Визуализация протяженного участка местности в виде последовательных кадров в разных окнах.
- ✓ Представление удаленных друг от друга участков.

Так как используется три окна карты, то переход от одной вкладки к другой происходит очень быстро.

Дополнительные окна (**Список**, **График** и т.д.) могут быть представлены в виде вспомогательных форм (Рис. 79).

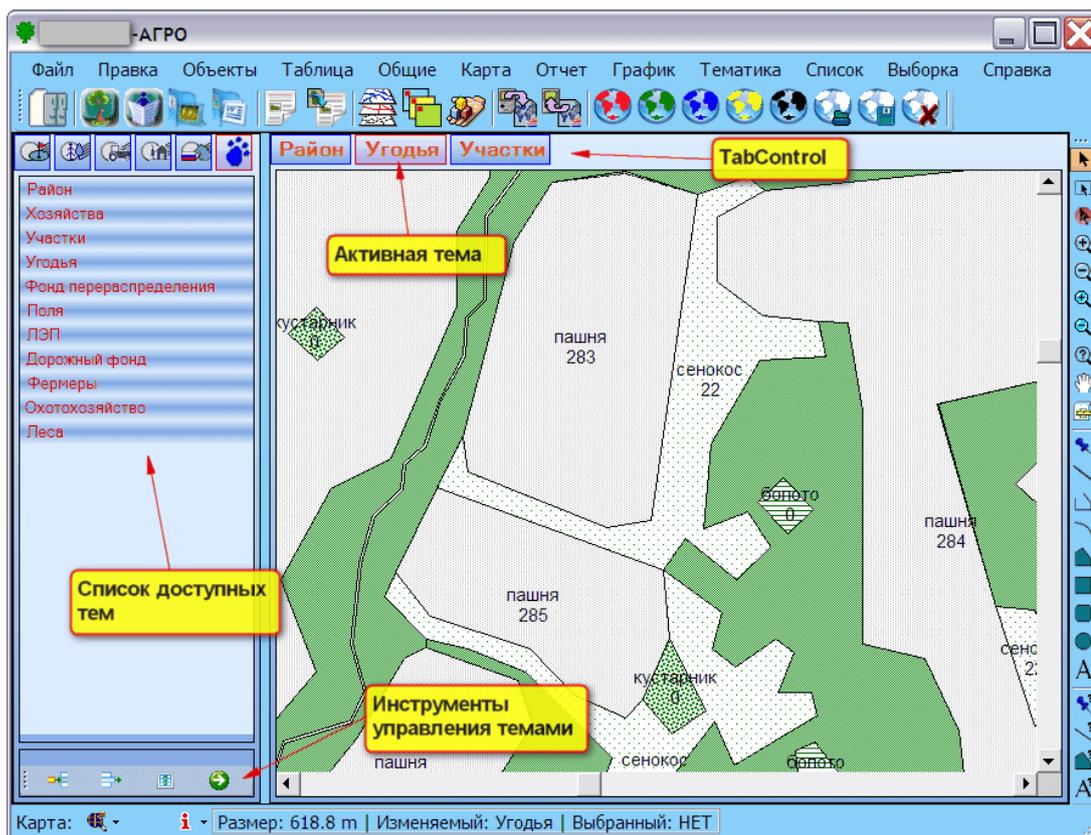


Рис. 78

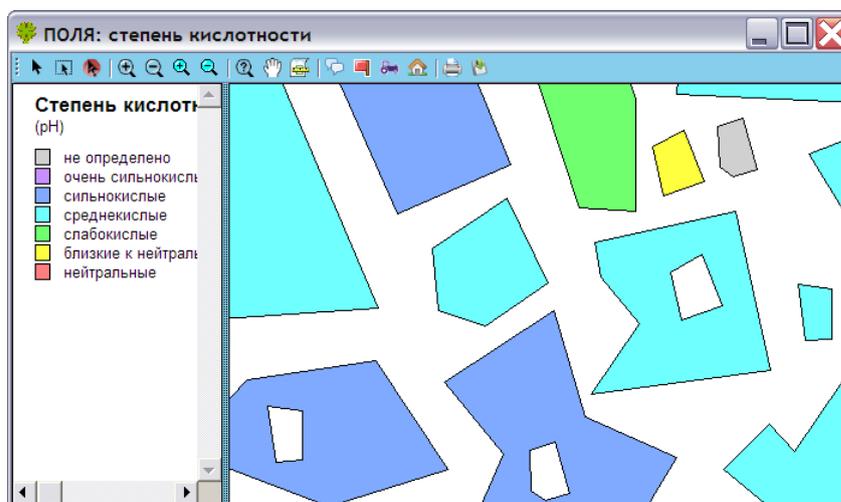


Рис. 79

### Пример 3

В качестве данного примера можно рассмотреть саму программу MapInfo. Основой приложения является родительская *MDI*<sup>58</sup>-форма. Эта форма содержит дочерние *MDI*-окна (Карта, Список и т.д.). Может быть, реализовывать MapInfo в полном объеме и нет особого смысла, но в каких-то задачах такая конструкция может быть очень эффективной. Просто не стоит о ней забывать.

### Пример 4

Комбинированный вариант, когда и приложение клиент и сервер работают в своих окнах (Рис. 80).

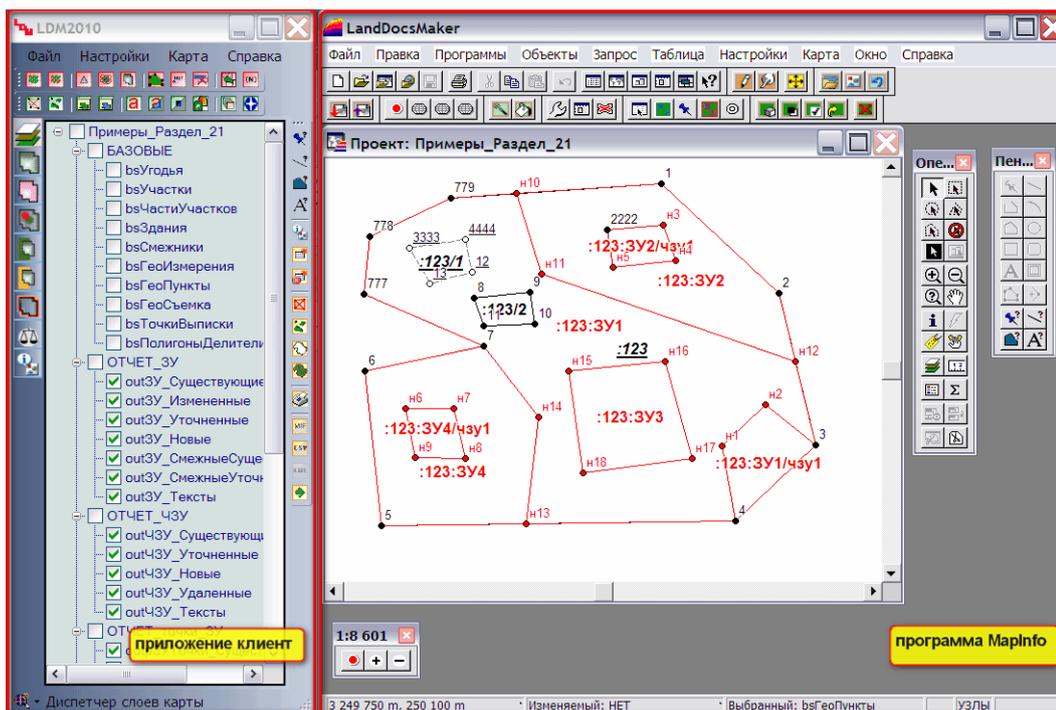


Рис. 80

При этом могут использоваться все возможности MapInfo. К тому же сокращается время на освоение новой программы, так как предполагается, что с MapInfo все сотрудники работать умеют.

При загрузке управляющего приложения запускаются необходимые MapBasic программы, в том числе и имеющие пользовательский интерфейс в виде инструментальных панелей или разделов меню.

Далее приводится код, показывающий как можно реализовать такую комбинированную конструкцию.

```
Public Class Form1
    Public mInfo As cMI80
    Private Sub Button1_Click(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles Button1.Click
        Me.Left = 0
        mInfo = New cMI80(Me.Width, My.Computer.Screen.WorkingArea.Width, _
            My.Computer.Screen.WorkingArea.Height)
        Me.Refresh()
    End Sub
End Class
```

<sup>58</sup> Multiple Document Interface - мультидокументный интерфейс.

End Class

```
Public Class cMI80
    Friend Declare Function SetWindowText& Lib "user32.dll" _
        Alias "SetWindowTextA" (ByVal hwnd As Int32, _
            ByVal lpString As String)
    Friend Declare Function OpenIcon& Lib "user32.dll" (ByVal hwnd As Int32)
    Friend Declare Function MoveWindow Lib "user32.dll" _
        (ByVal hwnd As Int32, ByVal x As Int32, ByVal y As Int32, _
            ByVal nWidth As Int32, ByVal nHeight As Int32, _
            ByVal bRepaint As Int32) As Boolean
    Public Const SYS_INFO_MAPINFOWND As Int16 = 9
    Dim mi As Object
    Public miHWND As Int32

    Public Sub New(ByVal FormMainWidth As Integer, _
        ByVal scrWidth As Integer, ByVal scrHeight As Integer)
        MyBase.New()
        mi = CreateObject("MapInfo.application")
        miHWND = mi.eval("SystemInfo(" & SYS_INFO_MAPINFOWND.ToString & ")")
        SetMIwindow(FormMainWidth, scrWidth, scrHeight)
    End Sub

    Public Sub SetMIwindow(ByVal FormMainWidth As Integer, _
        ByVal scrWidth As Integer, ByVal scrHeight As Integer)
        ' Настройки окна MapInfo
        Dim OpIcon, jj As Boolean
        Dim miTop, miLeft, miHeight, miWidth As Int32
        Dim str As String = "TEST"
        miTop = 2
        miLeft = FormMainWidth + 3
        miHeight = scrHeight
        miWidth = scrWidth - FormMainWidth - 4
        OpIcon = OpenIcon(miHWND)
        jj = MoveWindow(miHWND, miLeft, miTop, miWidth, miHeight, 0)
        SetWindowText(miHWND, str)
    End Sub
End Class
```

## Приложения

### . Файл *MapInfow.mnu*

Практически весь графический интерфейс MapInfo определяется файлом **MapInfow.mnu**. Этот текстовый файл представляет собой кусок кода MapBasic. Код можно редактировать, открыв файл в любом текстовом редакторе, например **Notepad.exe**<sup>59</sup>. Таким образом, можно управлять графическим интерфейсом MapInfo даже не прибегая к программированию.

Кроме того этот файл является хорошим примером использования кода управляющего графическим интерфейсом. При программировании можно вызвать любую процедуру из указанных в **MapInfow.mnu**, используя оператор `Run Menu Command ID_Command`. Здесь `ID_Command` – код процедуры. Например, закрыть все открытые таблицы можно выполнив, оператор `Run Menu Command 104`. Более наглядно будет `Run Menu Command M_FILE_CLOSE_ALL`. Использование таких осмысленных констант будет доступно, если подключить к программе файл **Menu.def** (`Include "Menu.def"`).

При старте MapInfo ищет файл **MapInfow.mnu** по следующей схеме:

*AppDirMI* → *LocalAppDirMI* → *AppDir* → *ProgrDir*.

Здесь

<i>AppDirMI</i>	C:\Documents and Settings\ <i>&lt;имя учетной записи&gt;</i> \Application Data\MapInfo\MapInfo\Professional\800 <sup>60</sup>
<i>LocalAppDirMI</i>	C:\Documents and Settings\ <i>&lt;имя учетной записи &gt;</i> Local Settings\Application Data\MapInfo\MapInfo\Professional\800
<i>AppDir</i>	C:\Documents and Settings\ <i>&lt;имя учетной записи &gt;</i> \Application Data\MapInfo\MapInfo
<i>ProgrDir</i>	папка, где находится файл <i>mapinfow.exe</i>

По умолчанию **MapInfow.mnu** размещается в *ProgrDir* однако можно отредактированный файл **MapInfow.mnu** установить в *AppDirMI* и тогда именно он будет загружаться при запуске MapInfo. Этот факт можно использовать при разработке ГИС, когда ваша программа (не MapBasic), перед запуском MapInfo, копирует свой файл меню в папку *AppDirMI*, а по окончании работы удаляет его<sup>61</sup>.

<sup>59</sup> Рекомендуется предварительно сохранить копию редактируемого файла.

<sup>60</sup> Имеется ввиду MapInfo v.8.0

<sup>61</sup> Или в качестве шутки для вашего коллеги.

## 2. Точность представления координат в MapInfo

Назначение рабочей области является очень важным моментом при формировании проекта в MapInfo<sup>62</sup>. Границы рабочей области необходимо устанавливать исходя из задачи, возможно с некоторым запасом. Установка очень большой рабочей области, так сказать на все случаи жизни, отразится на точности снимаемых с карты координат. Точность представления координат в MapInfo можно оценить исходя из следующего. MapInfo хранит координаты как 32-битные целые числа с одинарной точностью от 0 до +2000000000. Отобразив рабочую область карты на этот диапазон, получим оценку для точности представления координат

$$dX = (X_{\max} - X_{\min}) / 2000000000$$

$$dY = (Y_{\max} - Y_{\min}) / 2000000000$$

Сделаем некоторые расчеты и построения:

Вариант	Границы рабочей области по каждой оси (м)	d (м)
A	-70 000 000; +70 000 000	0.070
B	-10 000 000; +10 000 000	0.010
C	0; +500 000	0.00025

Увеличьте изображение в окне карты таким образом, чтобы оно было соизмеримо со значением  $d$ . Вставьте точку. Следующую точку вставьте слева от предыдущей на минимально возможном расстоянии ( $d$ ). Аналогично вправо, вниз, вверх. Получится сетка из точек, для которой невозможно вставить другие точки кроме узлов сетки. При этом шаг сетки будет равен значению  $d$ . Таким образом:

- ✓ Поле координат в карте MapInfo является дискретным.
- ✓ С ростом размеров рабочей области точность представления координат в MapInfo снижается.
- ✓ Все таблицы проекта должны иметь одну и ту же рабочую область.

Рассмотрим вопрос из практики связанный с рассматриваемой темой. Нередко пользователи возмущаются по поводу того, что после округления координат до 0.01 метра (и изменения положения точки) в значениях координат все равно сохраняется некоторый хвост из цифр, хотя должны быть нули. Это является следствием дискретности представления координат в MapInfo.

Если установим дискретность сетки в 1 см, то все определяемые по карте координаты после сантиметров будут иметь нули. Дискретности сетки 1 см соответствует рабочая область 20000000 на 20000000 метров. Нужно понимать, что при этом мы будем закруглять результаты. Практически, все промежуточные операции нужно выполнять с координатами до 0.001 м, а окончательный результат округлять до 0.01 м. Чтобы это выполнить дискретность сетки должна быть 1 мм, а рабочая область 2000000 на 2000000 метров. Конкретно, если работать предполагается с координатами  $3000000 < X < 3500000$  и  $0 < Y < 500000$ <sup>63</sup>, то для того чтобы обеспечить дискретность сетки 1 мм и получать ожидаемые результаты округления координат нужно установить следующую рабочую область:  $2000000 < X < 4000000$  и  $0 < Y < 2000000$ .

<sup>62</sup> Здесь рассматривается План-схема (метры).

<sup>63</sup> Здесь X и Y даны в системе координат MapInfo.

### 3. Формирование запросов

Назначение запроса это выборка и фильтрация данных в соответствии с некоторым условием, а также с возможностью их сортировки, группировки, вычисления итоговых сумм и т.п. Формировать запросы приходится достаточно часто. Ниже приводятся примеры организации оператора `Select` и условий для формирования выборки.

Оператор	Описание запроса
<code>Select * From bsУгодья Order by ТипУгодья Into Selection</code>	Выбрать всю таблицу <code>bsУгодья</code> и упорядочить записи по полю <code>ТипУгодья</code> .
<code>Select * From bsУгодья Where ТипУгодья=2 Into Selection</code>	Выбрать все записи из таблицы <code>bsУгодья</code> , где поле <code>ТипУгодья</code> имеет значение 2.
<code>Select * From bsУгодья Where ТипУгодья = 2 And Балл &gt; 10 Into Selection</code>	Выбрать записи из таблицы <code>bsУгодья</code> , где поле <code>ТипУгодья</code> имеет значение 2 и у которых значение поля <code>Балл</code> больше 10.
<code>Select * From bsСмежники Where Name Like "000%" Into Selection</code>	Выбрать все записи из таблицы <code>bsСмежники</code> , для которых значение поля <code>Name</code> начинается с символов «000».
<code>Select * From bsСмежники Where Pravo Like "_._.Б%" Into Selection</code>	Выбрать все записи из таблицы <code>bsСмежники</code> , для которых значение поля <code>Pravo</code> имеет следующую структуру: буква, точка, буква, точка, буква «Б», любой текст.
<code>Select id, Sum(Area) From УчасткиСад Where Area &gt; 0 Into Selection</code>	Определить сумму площадей всех участков в таблице <code>УчасткиСад</code> <sup>64</sup> .
<code>Select id, Sum(Area), Count(*), Avg(Area) From УчасткиСад Where Area &gt; 5000 Into Selection</code>	Выбрать все записи имеющие значение в поле <code>Area</code> больше 5000. Для них определить сумму площадей, число участков, среднее значение площади из отобранных записей <sup>65</sup> .
<code>Select Sum(Area), Year(Дата) "Год" From УчасткиСад Where Area &gt; 5000 Group by Right\$(Дата,4) Into Selection</code>	Выбрать все записи из таблицы <code>УчасткиСад</code> , где поле <code>Area</code> больше 5000. В результирующей выборке информацию <code>Sum(Area)</code> сгруппировать по годам. Для колонки <code>Year(Дата)</code> установить псевдоним <code>Год</code> .
<code>Select Угодья.Тип, Sum(Угодья.Площадь) "ПлощадьУгодий" From УчасткиСад, Угодья Where УчасткиСад.num="2" And УчасткиСад.obj Contains Угодья.obj Group by Угодья.Тип Into Selection</code>	Кроме таблицы <code>УчасткиСад</code> имеется таблица <code>Угодья</code> , объекты которой покрывают объекты таблицы <code>УчасткиСад</code> и имеют с ними общие границы. Определить суммарную площадь для каждого типа угодий на участке с номером 2.

Как видно из приведенного выше, основное при формировании запроса это правильно организовать условие отбора записей. Далее приведены примеры различного вида условий отбора записей из базовой таблицы. Здесь предполагается следующее соответствие значений и типов полей: *A, B* – числовое; *C, D* – символьное; *E* – дата; *H* – логическое.

Условие	Описание
<code>A&gt;=1000 And A&lt;=2000</code> или <code>A Between 1000 And 2000</code>	Все записи из базовой таблицы, для которых значение поля <code>A</code> находится в диапазоне от 1000 до 2000 (включая граничные значения).

<sup>64</sup> В таблице `УчасткиСад` имеются поля: `id`, `Area`, `Дата`.

<sup>65</sup> Поле `id` будет равно значению для первой отобранной записи.

## ПРИЛОЖЕНИЯ

<code>D=Any ("Иванов", "Петров", "Сидоров")</code>	Все записи из базовой таблицы, для которых значение поля D равно или "Иванов", или "Петров", или "Сидоров".
<code>D=""</code>	Все записи из базовой таблицы, для которых не заполнено поле D.
<code>D=Not (D="")</code>	Все записи из базовой таблицы, для которых заполнено поле D.
<code>E&gt;"2.12.2005"</code>	Все записи из базовой таблицы, для которых дата в поле E позднее 2.12.2005.
<code>Year (E) &gt;=2004</code>	Все записи из базовой таблицы, для которых значение года в поле E больше 2003.
<code>H="T"</code>	Все записи из базовой таблицы, для которых значение поля H равно True.
<code>Str\$(Obj) = "Point"</code>	Все записи из базовой таблицы, для которых тип объекта – Point. Аналогичное условие получим и для других типов объектов, подставляя вместо Point соответствующий тип объекта (Line, Polyline, Region, Text).
<code>Str\$(Obj)="Polyline" And StyleAttr (ObjectInfo (Obj, 2), 4)=16711680</code>	Все полилинии красного цвета из базовой таблицы. Здесь 16711680 – RGB-код для красного цвета.
<code>Str\$(Obj)="Point" And (CentroidX(Obj) Between 500 And 3000) And (CentroidY(Obj) Between 1500 And 5000)</code>	Все объекты типа Point из базовой таблицы, для которых значение координаты X находится в диапазоне от 500 до 3000 и значение координаты Y находится в диапазоне от 1500 до 5000.

Определить значения кодов для стилей оформления объектов, например для линейных объектов, можно следующим образом.

Установим стиль для линий (**Настройка /Стиль линий**) – простая линия, толщина 1 пиксел, цвет красный. Этот стиль становится текущим для линий. Затем, в окне **MapBasic**, введем оператор `print Str$(CurrentPen())`. После нажатия клавиши **Enter**, в окне **Сообщения**, получим следующий текст `Pen(1,2,16711680)`. Здесь: 1 – толщина линии; 2 – тип линии; 16711680 – код цвета.

#### 4. Список идентификаторов для стандартных меню

<i>Файл</i>	ID 1	
<i>Правка</i>	ID 2	
<i>Объекты</i>	ID 14	
<i>Запрос</i>	ID 3	
<i>Таблица</i>	ID 15	
<i>Настройки</i>	ID 5	
<i>Окно</i>	ID 6	
<i>Справка</i>	ID 7	
<i>Список</i>	ID 8	Используется, при активном окне Списка
<i>Карта</i>	ID 9	Используется, при активном окне Карты
<i>График</i>	ID 11	Используется, при активном окне Графика
<i>Отчет</i>	ID 10	Используется, при активном окне Отчета
<i>Геогруппы</i>	ID 13	Используется, при активном окне Геогрупп
<i>MapBasic</i>	ID 12	Используется, при активном окне MapBasic
<i>Программы</i>	ID 4	
<i>WinSpecific</i>		Обозначает меню, соответствующее открытому окну: "Карта", "График", "Список", "Отчет", "MapBasic" или "Справка"
<i>DefaultShortcut</i>	ID 16	Контекстное меню для окон, не имеющих своего персонального контекстного меню
<i>MapperShortcut</i>	ID 17	Контекстное меню окна Карты
<i>BrowserShortcut</i>	ID 18	Контекстное меню окна Списка
<i>LayoutShortcut</i>	ID 19	Контекстное меню окна Отчета
<i>GrapherShortcut</i>	ID 20	Контекстное меню окна Графика
<i>CmdShortcut</i>	ID 21	Контекстное меню окна MapBasic
<i>RedistrictShortcut</i>	ID 22	Контекстное меню окна Геогрупп

## 5. Вспомогательные окна

Для открытия вспомогательных окон используется оператор `Open Window window_name` или `Open Window window_cod`. Соответственно закрыть окно можно оператором `Close Window window_name` или `Close Window window_cod`.

Для настройки отображения вспомогательного окна применяют оператор `Set Window window_name...` или `Set Window window_cod...` Однако не все параметры этого оператора доступны для вспомогательных окон.

<b>Имя окна</b>	<b>Код<sup>66</sup></b>	<b>Имя окна (рус.)</b>
MapBasic	WIN_MAPBASIC	MapBasic
Statistics	WIN_STATISTICS	Статистика
Legend	WIN_LEGEND	Легенда
Info	WIN_INFO	Информация
Ruler	WIN_RULER	Линейка
Help	WIN_HELP	Справка
Message	WIN_MESSAGE	Сообщения

---

<sup>66</sup> Код по файлу MAPBASIC.DEF.

## 6. Единицы измерений

### Единицы измерения площадей

Список доступных единиц измерения площади. По умолчанию квадратные мили.

"acre"	акр
"hectare"	гектар
"sq cm"	квадратный сантиметр
"sq ft"	квадратный фут
"sq in"	квадратный дюйм
"sq km"	квадратный километр
"sq m"	квадратный метр
"sq mi"	квадратная миля
"sq mm"	квадратный миллиметр
"sq survey ft"	квадратный топографический фут (США)
"sq yd"	квадратный ярд

Пример: Set Area Units "sq m"

### Единицы измерения расстояний

Список доступных единиц измерения расстояний. По умолчанию мили.

"cm"	сантиметр
"ft"	фут (один международный фут примерно равен 30.48 сантиметрам)
"in"	дюйм
"km"	километр
"m"	метр
"mi"	миля
"mm"	миллиметр
"nmi"	морские мили (одна морская миля равна 1852 метрам)
"survey ft"	топографический фут (использовался при обмере территории США в 1927; один топографический фут примерно равен 30.48006 сантиметрам)
"yd"	ярд

Пример: Set Distance Units "km"

### "Бумажные" единицы измерения

Список доступных единиц измерения для размеров и положения окон на экране или объектов на печатном листе. По умолчанию дюйм.

"cm"	сантиметр
"in"	дюйм
"mm"	миллиметр
"pt"	пункт (точка)
"pica"	пика

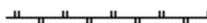
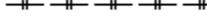
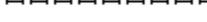
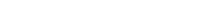
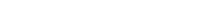
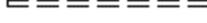
Пример: Set Paper Units "mm"

## 7. Стили оформления

### Типы линий

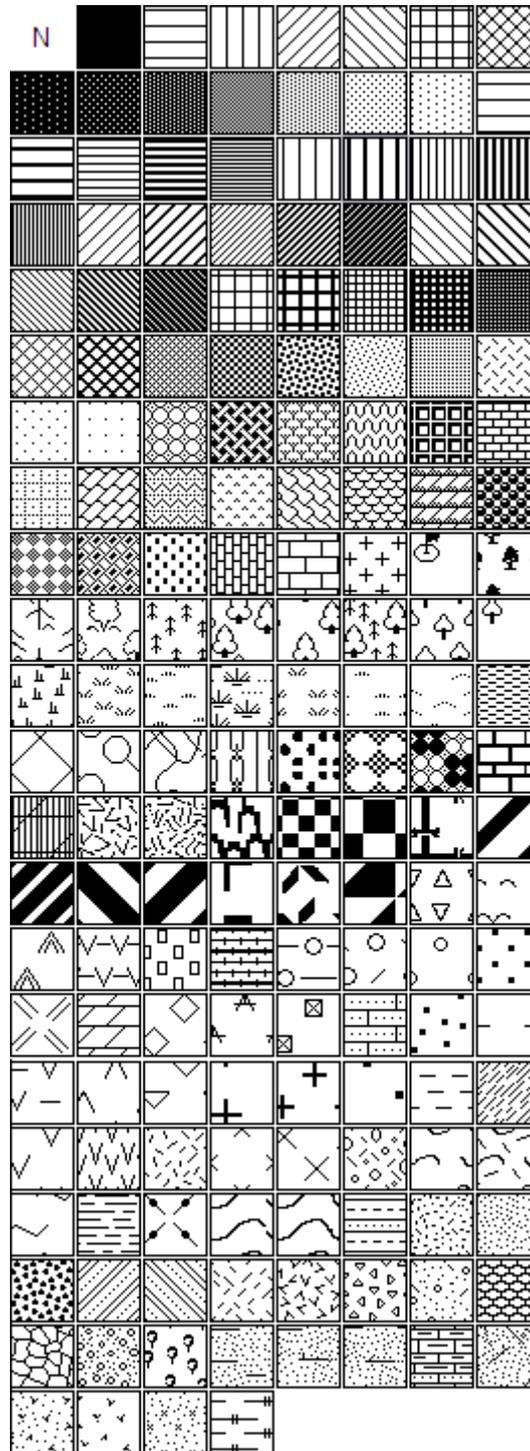
Тип линии определяется файлом **MAPINFOW.PEN**.

В таблице показаны образцы линий для стандартного файла **MAPINFOW.PEN** и соответствующие им коды:

01		28		55	
02		29		56	
03		30		57	
04		31		58	
05		32		59	
06		33		60	
07		34		61	
08		35		62	
09		36		63	
10		37		64	
11		38		65	
12		39		66	
13		40		67	
14		41		68	
15		42		69	
16		43		70	
17		44		71	
18		45		72	
19		46		73	
20		47		74	
21		48		75	
22		49		76	
23		50		77	
24		51			
25		52			
26		53			
27		54			

## Тип заливки

Доступные шаблоны заливки в MapInfo v.8



Конкретный тип заливки можно определить следующим образом:

- ✓ установить требуемый стиль текущим;
- ✓ ввести в окно MapBasic оператор `Print Str$(CurrentBrush());`
- ✓ получить описание этого стиля в окне **Сообщения**.

## Стиль шрифта

Значение Style	Стиль написания
0	Нормальный
1	Жирный
2	Курсив
4	Подчеркнутый
16	Контурный (только для Macintosh)
32	Оттененный
256	Кайма
512	Полная капитализация
1024	Разреженный

Для задания сложного стиля значения *Style* складываются. Например: жирный с каймой – 257.

## Стиль символа

Коды стандартного набора символов MapInfo 3.0

31		41	☆	51	✱	61	🛡
32	■	42	△	52	✈	62	🛡
33	◆	43	▽	53	🚧	63	🚧
34	●	44	▣	54	↑	64	✕
35	☆	45	▲	55	🏠	65	📍
36	▲	46	●	56	+	66	📍
37	▽	47	➔	57	⚓	67	📍
38	□	48	↙	58	⚓		
39	◇	49	+	59	⊙		
40	○	50	✕	60	🏠		

Для символов из шрифта *TrueType* значения параметра *Fontstyle* управляющего написанием символа:

Значение Fontstyle	Стиль написания
0	Нормальный
1	Жирный
16	Черная кайма
32	Оттененный
256	Белая кайма

Для задания сложного стиля значения *Fontstyle* складываются. Например: жирный с белой каймой – 257.

**. Параметры предложения CharSet**

<b>Параметр</b>	<b>Комментарий</b>
"Neutral"	нет преобразования
"ISO8859_1"	ISO 8859-1 (UNIX)
"ISO8859_2"	ISO 8859-2 (UNIX)
"ISO8859_3"	ISO 8859-3 (UNIX)
"ISO8859_4"	ISO 8859-4 (UNIX)
"ISO8859_5"	ISO 8859-5 (UNIX)
"ISO8859_6"	ISO 8859-6 (UNIX)
"ISO8859_7"	ISO 8859-7 (UNIX)
"ISO8859_8"	ISO 8859-8 (UNIX)
"ISO8859_9"	ISO 8859-9 (UNIX)
"PackedEUCJapanese"	UNIX, японский стандарт
"WindowsLatin1"	Windows, стандарт США и Западной Европы
"WindowsLatin2"	Windows, стандарт Восточной Европы
"WindowsArabic"	
"WindowsCyrillic"	
"WindowsGreek"	
"WindowsHebrew"	
"WindowsTurkish"	
"WindowsTradChinese"	Windows Traditional Chinese
"WindowsSimpChinese"	Windows Simplified Chinese
"WindowsJapanese"	
"WindowsKorean"	
"CodePage437"	DOS Code Page 437 = IBM Extended ASCII
"CodePage850"	DOS Code Page 850 = Multilingual
"CodePage852"	DOS Code Page 852 = Eastern Europe
"CodePage855"	DOS Code Page 855 = Cyrillic
"CodePage857"	
"CodePage860"	DOS Code Page 860 = Portugese
"CodePage861"	DOS Code Page 861 = Icelandic
"CodePage863"	DOS Code Page 863 = French Canadian
"CodePage864"	DOS Code Page 864 = Arabic
"CodePage865"	DOS Code Page 865 = Nordic
"CodePage869"	DOS Code Page 869 = Modern Greek
"LICS"	Lotus worksheet release 1,2 character set
"LMBCS"	Lotus worksheet release 3,4 character set

## . Описание формата MIF/MID

MIF/MID универсальный формат обмена данными для MapInfo. Одна пара файлов MIF/MID представляет одну таблицу MapInfo.

Информация хранится в двух текстовых (ASCII) файлах:

- ✓ Файл с расширением MIF. Содержит всю графическую информацию.
- ✓ Файл с расширением MID. Содержит всю семантическую информацию.

MIF-файл состоит из двух частей: заголовка и секции данных.

Заголовок файла содержит информацию о структуре таблицы, ее свойствах и имеет следующий вид<sup>67</sup>.

```
VERSION n
NAME
Charset "characterSetName"
[ DELIMITER "<c>" ]
[ UNIQUE n,n.. ]
[ INDEX n,n.. ]
[ COORDSYS... ]
[ TRANSFORM... ]
COLUMNS n
    <name> <type>
    <name> <type>
```

...

Секция данных следует после заголовка и начинается со служебного слова *DATA* на отдельной строке. В секцию данных записывается информация о графических объектах. Могут использоваться следующие графические объекты:

### 1. Точка.

```
POINT x y
    [ SYMBOL (образец, цвет, размер) ]
```

Если стиль символа не задан, то будет использоваться текущий символ.

### 2. Линия.

```
LINE x1 y1 x2 y2
    [ PEN (ширина, тип_линии, цвет) ]
```

Если стиль линии не задан, то будет использоваться текущий стиль линии.

### 3. Полилиния.

```
PLINE [ MULTIPLE numsections ]
    numpts1
    x1 y1
    x2 y2
...
[numpts2
    x1 y1
    x2 y2
... ]
    [ PEN (ширина, тип_линии, цвет) ]
    [ SMOOTH ]
```

Если используется *SMOOTH*, то полилиния будет сглажена. Обозначения: *numsections* – число секций, *numpts1* – число узлов в первой секции и т.д.

### 4. Область.

```
REGION numpolygons
    numpts1
    x1 y1
    x2 y2
    ...
```

<sup>67</sup> Квадратные скобки определяют необязательный элемент.

```
[numpts2
  x1 y1
  x2 y2
  ... ]
[ PEN (ширина, тип_линии, цвет) ]
[ BRUSH (шаблон, основной_цвет, цвет_фона) ]
[ CENTER x y ]
```

Ключевое слово *CENTER* задает координаты центра области. Центр должен быть внутри объекта. Обозначения: *numpolygons* – число секций, *numpts1* – число узлов в первой секции и т.д.

### 5. Дуга.

```
ARC x1 y1 x2 y2
  a b
  [ PEN (ширина, тип_линии, цвет) ]
```

Обозначения: *x1 y1 x2 y2* - противоположные по диагонали углы описанного прямоугольника, *a* – начальный угол дуги в градусах, *b* - конечный угол дуги в градусах<sup>68</sup>.

### 6. Текст.

```
ТЕХТ "textstring"
  x1 y1 x2 y2
  [ FONT... ]
[ Spacing {1.0 | 1.5 | 2.0} ]
[ Justify {Left | Center | Right} ]
[ Angle text_angle ]
[ Label Line {simple | arrow} x y ]
```

Длина *textstring* не должна превышать 255 символов. Обозначения: *x1, y1, x2, y2* - противоположные по диагонали углы описанного прямоугольника, *Spacing* - интервал между строками, *Justify* – выравнивание, *Angle* – поворот текста, *Label Line* – указатель для строки (простая линия или стрелка).

### 7. Прямоугольник.

```
РЕСТ x1 y1 x2 y2
  [ PEN (ширина, тип_линии, цвет) ]
  [ BRUSH (шаблон, основной_цвет, цвет_фона) ]
```

Обозначения: *x1, y1, x2, y2* - противоположные по диагонали углы прямоугольника.

### 8. Скругленный прямоугольник.

```
ROUNDRECT x1 y1 x2 y2
  a
  [ PEN (ширина, тип_линии, цвет) ]
  [ BRUSH (шаблон, основной_цвет, цвет_фона) ]
```

Обозначения: *x1, y1, x2, y2* - противоположные по диагонали углы прямоугольника, *a* - степень сглаживания.

### 9. Эллипс.

```
ELLIPSE x1 y1 x2 y2
  [ PEN (ширина, тип_линии, цвет) ]
  [ BRUSH (шаблон, основной_цвет, цвет_фона) ]
```

Обозначения: *x1, y1, x2, y2* - противоположные по диагонали углы описанного прямоугольника.

### 10. Группа точек.

```
MULTIPOINT num_points
  x1 y1 x2 y2 x3 y3 ...
  [ SYMBOL (образец, цвет, размер) ]
```

Обозначения: *num\_points* – число точек в группе.

### 11. Коллекция.

```
COLLECTION num_parts
Region
Pline
```

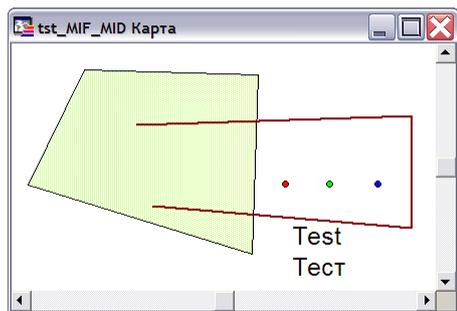
<sup>68</sup> Начало отсчета направление на восток, отсчет против часовой стрелки.

Multipoint

Индивидуальные параметры для *Region*, *Pline* и *Multipoint*, входящих в коллекцию, такие же, как и в соответствующих одиночных типах объектов. Обозначения: *num\_parts* - число частей коллекции.

Между объектами в секции данных и строками файла *MID* имеется соответствие. Первому объекту в секции данных соответствует первая строка файла *MID* и так далее. Если со строкой *MID-файла* не связан графический объект, то в *MIF-файле* в соответствующей позиции должно быть *NONE* («пустой» объект).

Покажем на примере, как может выглядеть таблица в формате *MIF/MID*. Имеется таблица MapInfo. Ее вид в окне карты и в виде списка показан на рисунке.



ID	typeObj
<input type="checkbox"/>	2 Region
<input type="checkbox"/>	3 Polyline
<input type="checkbox"/>	4 Point
<input type="checkbox"/>	5 Point
<input type="checkbox"/>	6 Point
<input type="checkbox"/>	10 Text

### Содержание соответствующего файла MIF.

```
Version 300
Charset "WindowsCyrillic"
Delimiter ","
CoordSys NonEarth Units "m" Bounds (3000000, 6000000) (3500000, 6500000)
Columns 2
  ID Integer
  typeObj Char(15)
Data

Region 1
  5
  3148699.822 6343190.176
  3148940.636 6343683.861
  3149687.431 6343661.905
  3149659.946 6342893.278
  3148699.822 6343190.176
  Pen (1,2,0)
  Brush (16,11796288)
  Center 3149042.212 6343139.999

Pline 4
  3149165.764 6343447.784
  3150343.913 6343487.102
  3150343.895 6343005.152
  3149231.644 6343101.902
  Pen (2,2,8388608)

Point 3149802.728 6343195.239
  Symbol (34,16711680,6)
Point 3149989.427 6343195.24
  Symbol (34,65280,6)
Point 3150198.09 6343195.242
  Symbol (34,255,6)

Text
  "Test\nТест"
  3149830.172 6342756.025 3150120.105 6343041.515
  Font ("Arial",256,0,0,16777215)
```

### Содержание соответствующего файла MID.

```
2, "Region"
```

3, "Polyline"  
4, "Point"  
5, "Point"  
6, "Point"  
10, "Text"