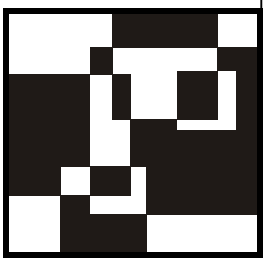
ГЛАВА 2



Знакомство с редактором Visual Basic

2.1. Общие сведения

Во многих ситуациях макрорекордер очень удобен, но в реальной работе одним им обойтись невозможно. Слишком много не умеет делать макрорекордер: он не умеет проверять значения, чтобы в зависимости от этого выполнять какое-либо действие, не работает с циклами, не умеет перехватывать и обрабатывать ошибки. Он использует только ограниченный и не лучший набор объектов (например, при вводе текста в Word использует чувствительный к действиям пользователя объект `Selection` вместо более надежного объекта `Range`). Макросы, которые созданы в макрорекордере, очень ограничены с функциональной точки зрения.

Полные возможности программирования в Office раскрываются при использовании редактора Visual Basic, и при серьезной работе без него не обойтись. В то же время, по моим наблюдениям, большинству обычных пользователей самостоятельно разобраться во всех способах его использования очень сложно.

В этой главе мы рассмотрим возможности редактора Visual Basic и познакомимся с тем, что можно сделать с помощью его многочисленных окон.

Внимание!

Для создания программ, которые используют возможности объектных моделей приложений Microsoft Office, использовать язык VBA и редактор Visual Basic совсем не обязательно. Вы вполне можете создавать такие программы на любом COM-совместимом языке, например, обычном Visual Basic, C++, Delphi, Java, VBScript и JavaScript, ActivePerl, C#, Visual Basic .NET и т. п. Однако язык VBA (и редактор Visual Basic) изначально создавались для автоматизации приложений Microsoft Office, и работать в них с объектными моделями приложений Microsoft Office удобнее всего.

Прежде чем начать работать с редактором Visual Basic, нужно его открыть. Во всех приложениях Office это делается одинаково:

- самый простой способ — в меню **Сервис | Макрос** выбрать **Редактор Visual Basic**;
- самый быстрый способ — нажать клавиши <Alt>+<F11>;
- можно также воспользоваться кнопкой на панели инструментов **Visual Basic** (предварительно сделав ее видимой);
- можно вызвать редактор при возникновении ошибки в макросе;
- можно открыть готовый макрос для редактирования в диалоговом окне **Макрос**.

В любом случае откроется окно, похожее на представленное на рис. 2.1.

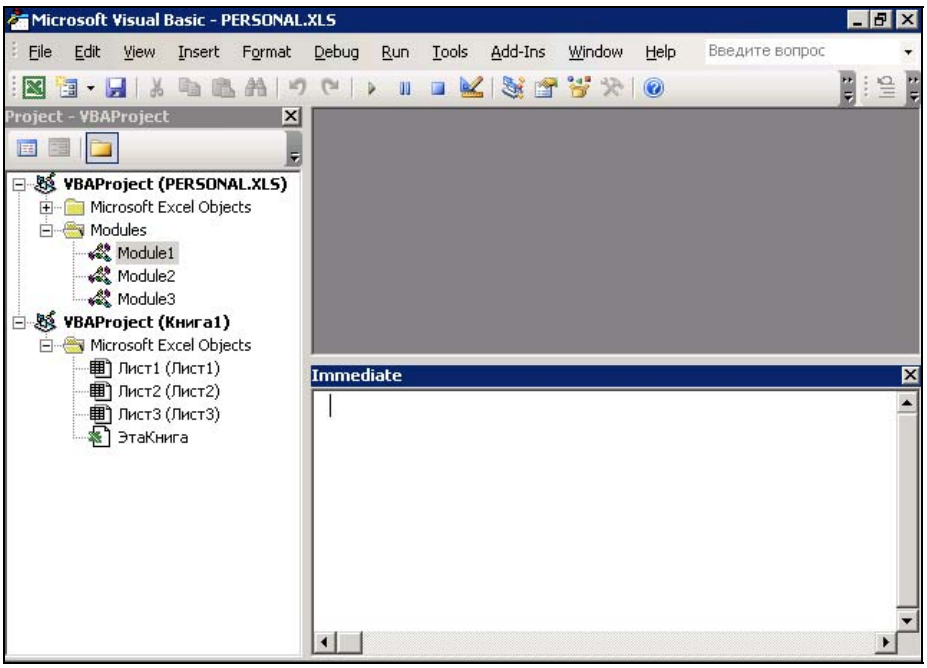


Рис. 2.1. Окно редактора Visual Basic в Excel

Внимание!

В окне редактора Visual Basic можно работать одновременно с работой в приложении, откуда этот редактор был вызван. Переход между окнами осуществляется через <Alt>+<Tab> (в редактор также можно "прыгнуть", повторно нажав <Alt>+<F11>).

Всего в редакторе Visual Basic предусмотрено 9 дополнительных окон:

- **Project Explorer** — окно проводника проекта. По умолчанию оно открыто и находится в левой части окна редактора Visual Basic. В нем можно просмотреть компоненты проекта и выполнить множество операций. Подробнее о работе в этом окне — в *разд. 2.2*;
- **UserForm** — окно формы. Появляется тогда, когда вы редактируете пользовательскую форму при помощи дизайнера форм. Подробнее про пользовательские формы и работу с ними — в *гл. 5*;
- **Toolbox** — панель инструментов управления. Из нее можно добавить элементы управления в форму или в сам документ. Появляется вместе с окном формы и будет рассматриваться тоже в *гл. 5*;
- **Properties** — одно из самых важных окон. Через него можно просмотреть свойства элемента управления или компонента проекта и изменить их;
- **Code** — окно программного кода. В этом окне выполняется основная работа по написанию кода макроса. При открытии программного модуля открывается автоматически. Приемы работы с этим окном будут рассмотрены в *разд. 2.3*;
- **Object Browser** — обозреватель объектов. Необходим для получения информации о классах, доступных программе. Подробная информация — в *гл. 4*, которая посвящена работе с объектами;
- **Watch** — окно контролируемых выражений. Используется во время отладки для отслеживания значений выбранных переменных программы и выражений. Работа в этом окне так же, как и работа с окнами **Locals** и **Immediate**, будет рассмотрена в *гл. 6*, в которой рассказывается про перехват ошибок и отладку;
- **Locals** — окно локальных переменных. Нужно для отслеживания во время отладки значений переменных текущей процедуры;
- **Immediate** — окно для немедленного выполнения команд в ходе отладки. Оно позволяет выполнить отдельные строки программного кода и немедленно получить результат.

Найти какое-либо окно можно очень просто: нужно выбрать в меню **View** одноименную команду, и если окно было скрыто, оно появится в редакторе.

И еще один момент, который многих пользователей может разочаровать. В русских версиях приложений Office для редактора Visual Basic предусмотрен англоязычный интерфейс. Справка по языку VBA и объектным моделям приложений Office — тоже только на английском. К сожалению, русифицированных вариантов не существует. Однако, по моим наблюдениям, знание

английского языка для того, чтобы писать программы в VBA, не критично (хотя и очень полезно): программы вполне можно создавать, не зная английского.

2.2. Окно проводника проекта (*Project Explorer*) и структура проекта VBA

Окно проводника проекта при первой активизации редактора Visual Basic обычно открыто. Если оно случайно было закрыто, то вызвать его можно тремя способами:

- нажать клавиши <Ctrl>+<R>;
- нажать кнопку **Project Explorer** на панели инструментов **Standard**;
- воспользоваться меню **View | Project Explorer**.

В окне **Project Explorer** представлено дерево компонентов вашего приложения VBA.

Самый верхний уровень — это *проект (Project)*, которому соответствует документ Word, рабочая книга Excel, презентация PowerPoint или другой файл, с которым работает данное приложение. Например, если вы открыли редактор Visual Basic из Word, то в **Project Explorer** будут представлены все открытые в настоящее время файлы Word и шаблон Normal.dot. Если редактор Visual Basic открыт из Excel, то в **Project Explorer** будут открытые книги Excel и специальная скрытая книга PERSONAL.XLS.

Кроме того, что обычно содержится в документах Office (текст, рисунки, формулы и т. п.), каждый проект (который и является документом) — это одновременно и контейнер для хранения стандартных модулей, модулей классов и пользовательских форм. Добавить в проект каждый из этих компонентов можно при помощи меню **Insert** или через контекстное меню в **Project Explorer**.

Стандартный модуль — это просто блок с текстовым представлением команд VBA. В модуле этого типа может быть только два раздела:

- раздел объявлений уровня модуля (объявление переменных и констант уровня модуля);
- раздел методов модуля (расположение процедур и функций).

При работе макрорекордера в Word в проекте Normal.dot или в текущем документе (в зависимости от места сохранения макроса) автоматически создается стандартный модуль **NewMacros** (в Excel — **Module1**), куда и записываются все создаваемые макрорекордером макросы.

В большинстве проектов VBA используется только один стандартный модуль, куда записывается весь код. Создавать новые стандартные модули имеет смысл только из следующих соображений:

- для удобства экспорта и импорта (из контекстного меню в **Project Explorer**). Так можно очень удобно обмениваться блоками кода между приложениями VBA (и обычного VB);
- для повышения производительности. При вызове любой процедуры модуля происходит компиляция всего модуля, поэтому иногда выгоднее разместить процедуры в разных модулях, чтобы компилировать только нужный в данный момент код;
- для улучшения читаемости. Если ваше приложение выполняет разные группы задач, то код, относящийся к каждой группе, лучше поместить в свой модуль.

Модули классов позволяют создавать свои собственные классы — чертежи, по которым можно создавать свои объекты. Обычно модули классов используются только в очень сложных приложениях. В обычных приложениях VBA они применяются редко, и в этой книге рассматриваться не будут.

Пользовательская форма является одновременно хранилищем элементов управления и программного кода, который относится к ним, к самой форме и происходящими с ними событиями. Подробнее про формы будет рассказано в *гл. 5*.

Еще один важный контейнер в **Project Explorer** — *контейнер References*, т. е. контейнер ссылок (в Excel его нет). В нем показывается, ссылки на какие другие проекты (документы Word) есть в нашем проекте, и, соответственно, какие "чужие" программные модули мы можем использовать. По умолчанию в каждый проект Word помещается ссылка на **Normal** (т. е. на шаблон Normal.dot), и вы в любом файле можете использовать макросы этого шаблона.

Обратите внимание, что в этом контейнере хранятся только ссылки на другие документы. Про добавление ссылок на другие объектные библиотеки мы поговорим в *гл. 4*.

Еще одна полезная возможность **Project Explorer** — *настройка свойства проекта*. Для этого нужно щелкнуть правой кнопкой мыши по узлу **Project** (**VBAProject** в Excel) и в контекстном меню выбрать **Project Properties** (окно свойств проекта можно открыть и через меню **Tools | Project Properties**). В этом окне можно:

- изменить имя проекта. Это может потребоваться, если у вас есть ссылки на проект с таким же именем;

- ввести описание проекта, информацию о файле справки и параметры, которые будут использоваться компилятором;
- защитить проект, введя пароль. Не зная этот пароль, проект нельзя будет просмотреть или отредактировать.

Тем не менее в окне **Project Explorer** обычно приходится выполнять следующие действия.

- Если вам нужно создать свой макрос вручную, а макросов в данном документе еще нет, то нужно щелкнуть правой кнопкой мыши по узлу проекта (строке, выделенной полужирным шрифтом) и в контекстном меню выбрать команду **Insert | Module**. В проекте будет создан новый модуль и сразу открыт в окне редактора кода. Что делать в этом окне — об этом в следующем *разд. 2.3*.
- Если вы уже создавали макросы в этом проекте (макрорекордером или вручную), то модуль будет уже создан. Его можно увидеть под контейнером **Modules**. Чтобы его открыть в окне редактора кода, достаточно щелкнуть по модулю два раза левой кнопкой мыши. Там можно будет найти макросы, созданные вами ранее средствами макрорекордера.

Внимание!

Обязательно подумайте, где вам будет нужен создаваемый код — только в одном документе или во всех документах данного приложения. Если он будет нужен только в одном документе, используйте стандартный программный модуль этого документа. Если во всех, то используйте программные модули проекта **Normal** (в Word) или PERSONAL.XLS (в Excel).

- Если вам нужно создать графическую форму с элементами управления (кнопками, текстовыми полями, раскрывающимися списками и т. п.), то нужно щелкнуть правой кнопкой мыши по узлу проекта и в контекстном меню выбрать **Insert | UserForm**. Новая форма будет создана и открыта в режиме дизайнера форм. Подробнее о работе с этим редактором — в *гл. 5*.

Теперь, когда программный модуль создан (или найден), можно приступать к работе с редактором кода VBA.

2.3. Работа с редактором кода (*Code Editor*)

2.3.1. Как открыть редактор кода и как он устроен

В редакторе кода выполняется основная часть работы по программированию, поэтому знать приемы работы с ним нужно очень хорошо. Открыть окно редактора кода можно несколькими способами:

- дважды щелкнуть по объекту модуля в **Project Explorer** (или выделить его и нажать клавишу <Enter>);

- выбрать нужный элемент (в **Project Explorer**, в дизайнере форм и т. п.) и в контекстном меню выбрать **View Code**;
- выделить нужный элемент и нажать клавишу <F7> (альтернатива — команда меню **View | Code**).

Редактор программного кода — это, по сути, обычный текстовый редактор, и в нем вы можете вырезать и вставлять код, перетаскивать фрагменты кода, скопировать путем перетаскивания с нажатой клавишей <Ctrl> — в вашем распоряжении почти все те же возможности, что и в редакторе Word. Однако он все-таки предназначен для специализированной задачи — создания кода программы. О его специальных возможностях рассказывается далее.

2.3.2. Список объектов и список событий

В верхней части окна редактора кода находятся два раскрывающихся списка. Слева находится *список объектов*. В нем вы можете выбрать объект, к которому будет относиться ваш код. Если вы открыли программный код модуля, то здесь будет только пункт **General**. Другое дело, если открыта форма, то в этом списке вы сможете выбрать саму форму или любой ее элемент управления и записать для него код.

Список справа — это *список процедур и событий*. В нем есть раздел **Declarations** — объявления уровня всего модуля и список всех процедур (макросов) для стандартного модуля или событий, если создается код для формы. При выборе нужного события будет автоматически создана нужная процедура, обрабатывающая это событие.

2.3.3. Закладки и разделение окна редактирования

Иногда в процессе написания программного кода в одном месте вам в голову приходит идея, относящаяся к другой части кода. Хочется перепрыгнуть в другое место, но разыскивать потом ту строку, где была прервана работа, очень не хочется. В этом случае опытные программисты используют закладки. *Закладка* (как и в случае с обычной книгой) — это метка, при помощи которой можно быстро найти нужное место. Работа с закладками производится либо с панели инструментов **Edit** (ее вначале нужно сделать видимой), либо через меню **Edit | Bookmark**. Для того чтобы включить или отключить закладку, нужно установить указатель ввода на нужную строку и воспользоваться командой **Toggle Bookmark**.

Часто бывает очень удобным разделение окна редактирования на две части — для просмотра разных частей модуля, для копирования и т. п. Делается это при помощи линии разбивки — маленького бегунка над вертикальной полосой прокрутки.

2.3.4. Как редактор помогает писать код

В редактор кода встроено множество средств, которые облегчают жизнь разработчику. Рассмотрим самые важные из них.

Самое полезное средство — это *получение списка свойств и методов*. В большинстве VBA-программ используются свойства и методы различных объектов (подробнее об этом — в *гл. 4*), при этом многие методы принимают параметры. Помнить точное название каждого свойства и метода, а также очередность передачи параметров невозможно, а разыскивать в очередной раз справку по этому объекту в документации — непроизводительная трата времени.

Показ списка свойств и методов в редакторе Visual Basic включен по умолчанию. Пользоваться этой возможностью очень просто: достаточно напечатать имя переменной, представляющей объект, и поставить после него точку. Автоматически откроется список всех свойств и методов этого объекта. В этом списке можно выбрать нужное свойство или метод (клавишами со стрелками или мышью, а если список большой, то можно набрать первые буквы имени свойства или метода), а затем нажать клавишу <Tab>. Если вы случайно закрыли список, то открыть его заново можно при помощи меню **Edit | List Properties/Methods** или клавиш <Ctrl>+<J>.

Если показ списка свойств и методов у вас отключен, то включить его можно при помощи меню **Tools | Options** (флажок **Auto List Members** на вкладке **Editor** окна **Options**).

Редактор Visual Basic готов показать вам не только перечень всех свойств и методов, но и все параметры, которые принимает данный метод. Это свойство также работает автоматически: достаточно после имени метода напечатать пробел. Для того чтобы явно вызвать список всех параметров, можно воспользоваться меню **Edit | Parameter Info** или клавишами <Ctrl>+<Shift>+<I>. Включить или отключить автоматический показ информации о параметрах можно при помощи флажка **Auto Quick Info** на той же вкладке **Editor** окна **Options**.

Список констант (допустимых значений для данного свойства) также появляется автоматически после того, как вы напечатаете знак равенства '='. Можно воспользоваться также меню **Edit | List Constants** или комбинацией клавиш <Ctrl>+<Shift>+<J>. Про сами константы будет рассказано в *гл. 3*.

Ключевые слова VBA и имена доступных в данный момент классов очень удобно вводить при помощи автоматического дополнения слов. Для этого достаточно выбрать меню **Edit | Complete Word** или нажать клавиши <Ctrl>+<Пробел>. Можно предварительно ничего не печатать, а можно набрать одну-две буквы.

Приведем еще несколько моментов, связанных с редактором кода:

- если вы напечатаете одну строку кода с отступом, то такой же отступ будет установлен для следующих строк. Изменить поведение можно при помощи параметра **Auto Indent** в диалоговом окне **Options**;
- если редактор кода распознает ключевое слово, он автоматически делает его первую букву заглавной и выделяет все слово синим цветом;
- часто бывает необходимо закомментировать или раскомментировать несколько строк сразу. Для этой цели можно включить отображение панели инструментов **Edit** и воспользоваться кнопками **Comment Block** и **Uncomment Block**;
- если при создании процедуры вы пишете ключевое слово `Sub` или `Function`, то редактор автоматически дописывает оператор `End Sub` или `End Function`. Между процедурами вставляется строка-разделитель;
- если при переходе на новую строку редактор кода обнаружит синтаксическую ошибку, то вам будет выдано предупреждение. Меня, например, такое поведение обычно сильно раздражает. Отменить протесты редактора можно, сняв флажок **Auto Syntax Check** в диалоговом окне **Options**. Работе это сильно не повредит, потому что синтаксически неверные строки в любом случае будут автоматически выделяться красным цветом;
- в редакторе кода вполне допускается работа сразу с несколькими окнами редактирования кода. Переход между ними осуществляется по клавишам <Ctrl>+<Tab> или <Ctrl>+<F6>;
- по умолчанию редактор кода работает в режиме **Full Module View** — показ всего содержимого модуля. Если вы хотите просматривать процедуры по отдельности, переключитесь в режим **Procedure View**. Кнопки для переключения находятся в левом нижнем углу окна редактора кода.

2.4. Работа со справкой

Работа со справкой по программированию в Office не так очевидна, как может показаться на первый взгляд.

Вызов окна справки производится из редактора Visual Basic по нажатию клавиши <F1>. Второй вариант — воспользоваться кнопкой **Справка** на панели инструментов **Standard**. В результате откроется дополнительное окно, аналогичное представленному на рис. 2.2.

Еще одна возможность вызвать справку — установить указатель мыши в нужное место в окне редактора кода (например, на имя вызываемого метода или используемого свойства) и нажать клавишу <F1>. Преимуществом такого

подхода является то, что при наличии нескольких вариантов (например, объект `Range` и свойство `Range`) вам автоматически откроется нужная страница.

Справка по программированию в приложении Microsoft Office обычно состоит из трех частей:

- первая часть (**Microsoft Excel Visual Basic Reference**, **Microsoft Word Visual Basic Reference** и т. п.) — это справка по объектной модели самого приложения Office;
- вторая часть (**Microsoft Visual Basic Documentation**, она одинакова во всех приложениях Office) — это справка по синтаксису и встроенным функциям самого языка VBA;
- третья часть (**Microsoft Office Visual Basic Reference**, она также одинакова во всех приложениях Office) — это справка по общим возможностям приложений Office: программная работа с панелями инструментов и меню, работа с помощником, организация взаимодействия с Windows SharePoint Services и т. п.

В некоторых приложениях (например, в Microsoft Access) в справку добавлены дополнительные разделы (рис. 2.3) — по объектной модели ADO, по языку SQL и т. п.

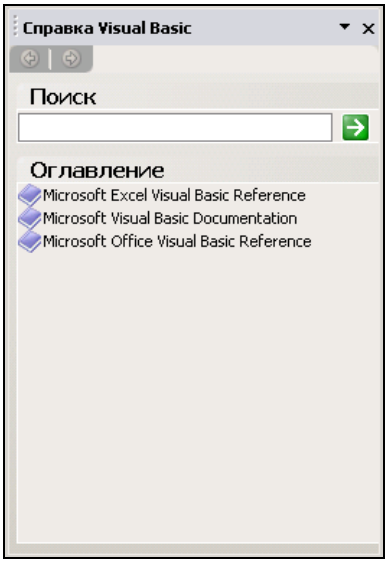


Рис. 2.2. Справка VBA в Excel

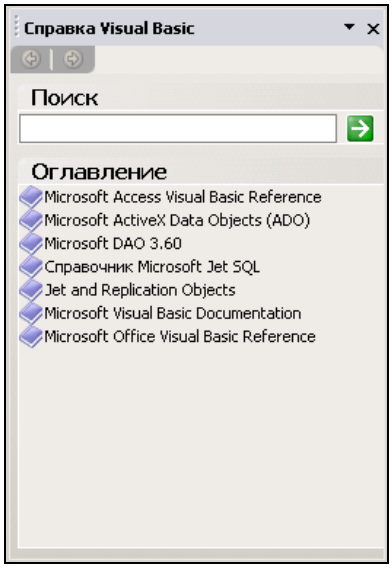


Рис. 2.3. Справка VBA в Access

Обычно самый важный раздел справки — это раздел, который посвящен возможностям конкретного приложения Office. Этот раздел условно можно разделить на две главные части (рис. 2.4):

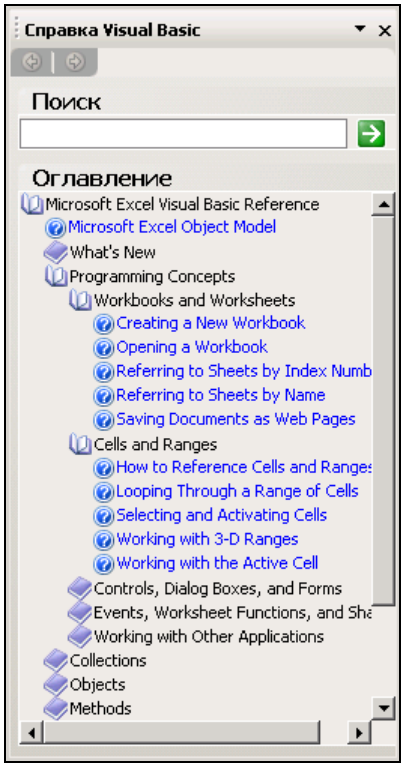


Рис. 2.4. Справка по компонентам объектной модели Excel

- *Programming Concepts (Концепции программирования)* — в этой части рассказывается, как программным образом выполнять самые распространенные операции. Например, для Excel это возможность создать или открыть рабочую книгу, найти нужный лист, получить или записать информацию в ячейку и т. п.;
- *справка по компонентам объектной модели приложения Office*: коллекциям (**Collections**), объектам (**Objects**), методам (**Methods**), свойствам (**Properties**) и т. п. При этом самые важные моменты, которые относятся скорее к области концепций (какими способами, например, можно создать объект `Range` в Excel), приводятся в справке по соответствующему объекту. Представление о всех функциональных возможностях данного объекта можно получить, только просмотрев подряд все его свойства и методы.

Найти направление, т. е. объект и его свойства и методы, которые нужно использовать в вашей ситуации, можно тремя способами:

- просмотреть раздел **Programming Concepts** в справке — не описана ли там ваша ситуация;

- просматривать все подряд объекты, свойства и методы в справке, пытаясь догадаться, что вам может помочь. Это самый неэффективный способ, поскольку объектов в любом приложении Office сотни (часто используемых — намного меньше, и все они рассмотрены в этой книге). Однако если вам предстоит в течение долгого времени заниматься программированием в каком-либо приложении Office, то имеет смысл потратить несколько дней, чтобы подряд прочитать справку по всем объектам, конспектируя самые важные моменты. Я могу гарантировать, что вы узнаете множество таких возможностей, о которых раньше и не подозревали;
- наиболее разумный способ — выполнить нужные вам операции в макрорекордере, а потом проанализировать созданный им код. Однако гарантировать то, что макрорекордер покажет вам самый эффективный путь, невозможно.

И напомним еще один момент, про которые мы уже говорили: справки по VBA на русском языке, к сожалению, не существует. Возможно, в какой-то степени ее сможет заменить эта книга, в которой рассмотрено множество свойств и методов самых важных объектов приложений Office.

Задание для самостоятельной работы 2: Редактирование макроса

ЗАДАНИЕ:

Измените созданный вами в задании для самостоятельной работы к *гл. 1* макрос таким образом, чтобы он запрашивал фамилию ответственного исполнителя. Для этого:

- добавьте вначале кода макроса (над первой строкой `Selection.TypeText`) следующие строки:

```
Dim sInput As String
sInPut = InputBox("Введите фамилию ответственного исполнителя",
"Запрос данных")
```

- замените строку:

```
Selection.TypeText Text:=("Отв. исполнитель Петрова М. М.")
```

(фамилия должна быть ваша) на строку:

```
Selection.TypeText Text:=("Отв. исполнитель " & sInPut)
```

Сохраните измененный макрос, закройте окно редактора кода и убедитесь, что макрос теперь работает по-новому.

Примечание

Не волнуйтесь, что вводимый вами код может показаться непонятным. Почему он именно такой, станет ясно после рассмотрения встроенных функций Visual Basic (функция `InputBox()`) и объектной модели Word (метод `TypeText()` объекта `Selection`) в соответствующих главах. Задача этой самостоятельной работы — позволить вам освоиться в окне редактора кода.

Ответ к заданию 2

1. Откройте Word и нажмите клавиши <Alt>+<F11>. В открывшемся окне Microsoft Visual Basic найдите окно **Project Explorer**, раскройте в нем узел **Normal | Modules | NewMacros**, выделите **NewMacros** и нажмите клавишу <F7>. Откроется окно с кодом **NewMacros**.
2. Найдите вашу процедуру `Sub Подпись()` и внесите необходимые изменения. Общий текст процедуры может быть, например, таким (обратите внимание: код вполне может не совпадать с вашим текстом — все зависит о того, какие действия вы выполняли в макрорекордере):

```
Sub Подпись()
'
' Подпись Макрос
' Макрос записан 02.05.2004 R
'
    Dim sInPut As String
    sInPut = InputBox("Введите фамилию ответственного исполнителя", _
                      "Запрос данных")
    Selection.TypeText Text:="Генеральный директор:" & vbTab & vbTab _
                      & vbTab & "Иванов А. А."
    Selection.ParagraphFormat.Alignment = wdAlignParagraphCenter
    Selection.TypeParagraph
    Selection.ParagraphFormat.Alignment = wdAlignParagraphLeft
    Selection.TypeText Text:=("Отв. исполнитель " & sInPut)
    Selection.TypeParagraph
    Selection.TypeText Text:="т. 55-55"
End Sub
```

3. Нажмите клавиши <Ctrl>+<S>, чтобы сохранить изменения, и <Alt>+<Q>, чтобы вернуться в Word. Выполните макрос, чтобы убедиться, что он работает в соответствии с заданием.