* * *

Для полного описания возможностей VBA потребовалось бы несколько толстых книг. Поэтому я не буду описывать все команды VBA, рассказывать об их формате и требуемых параметрах - про это очень подробно рассказано в справке, да и средства подсказки тоже не дадут ошибиться. Если необходимо узнать, какая команда позволяет программно реализовать ту или иную возможность Word, то можно воспользоваться средством записи макросов, а потом почитать в справке о каждой записанной команде.

Однако в VBA есть некоторые вещи, о которых в справке упоминается не на первом плане, и человеку, незнакомому с программированием, трудно узнать о них. Поэтому в этой главе вы найдете описания лишь некоторых команд, которые нельзя записать средством записи макросов, а также некоторую информацию о неочевидных возможностях VBA.

Возможно, те, кто занимался программированием, сочтут многое из изложенного очевидным. В этом случае им можно пропустить данную главу.

2.1. События, методы, свойства

Когда-то давным-давно программы, написанные и работавшие тогда еще под Dos, получали от пользователя данные, обрабатывали их и выдавали затем результат. Почти любая программа предусматривала в своей работе период ввода данных, период обработки, период выдачи результата. Подобные действия были принципом работы Dos-овских программ. И программирование их называлось структурным - надо было строго и последовательно разрабатывать алгоритм, реализовывать заданный порядок действий программы, в который в процессе работы почти не мог вмешаться пользователь, разве только принудительно остановив программу.

С появлением операционной системы Windows стал широко известным другой принцип программирования и создания алгоритмических языков - принцип **объектно-ориентированного языка.**

Это значит, что основная направленность разработчика сместилась с *действия* на *объект* - на его реакции на действия пользователя, на его свойства и на их изменение. И VBA является типичным представителем объектно-ориентированных языков, как по своему синтаксису, так и по архитектуре программ. Основными понятиями объектно-ориентированного языка являются **объект, свойство, метод, событие.**

Объект - это все, над чем может совершаться какое-либо действие или то, что имеет определенные характеристики. К примеру, открытый документ Word - это объект, первая буква в нем - это тоже объект, тридцатое слово, десятое предложение, второй рисунок - это все объекты. Объектами также являются запущенная программа, какой-нибудь файл на диске, даже сам Word - это тоже объект.

Почти каждый объект внутри себя имеет подобъекты, которые, в свою очередь, являются полноценными объектами и могут иметь свои подобъекты. Например, у объекта "Документ" есть подобъект "Десятое предложение", у которого есть подобъект "Второе слово", у которого есть подобъект "Третья буква". С помощью объектно-ориентированного языка VBA можно обратиться к любому объекту, если знать его иерархию - то есть все те объекты, чьим подобъектом он является.

Свойство - это любая характеристика объекта. К примеру, у объекта - первой буквы документа есть свойства: выделение жирным, выделение цветом, подчеркивание, выделение курсивом, регистр и много еще других. У объекта - документа есть свойства: наличие автоматической расстановки переносов, наличие автоматической проверки грамматики и др.

Большинство свойств объектов VBA можно задавать программно, однако, есть свойства Read-Only - не допускающие изменений.

Многие свойства объектов Word также задаются через стандартные диалоговые окна Word - например, Файл-Параметры страницы или Сервис-Параметры, однако их всегда можно задать и в программе, а соответствующие диалоговые окна использовать при записи макроса для того, чтобы посмотреть синтаксис команды задания того или иного свойства.

Метод - это какое-либо действие над объектом. Например, печать текста или поиск текста в документе. У многих методов есть **параметры метода**, позволяющие задать параметры действия. Вот пример:

```
With Selection.Find
.Text = "Этот текст надо заменить"
.Replacement.Text = "Заменить на этот текст"
.Forward = True
.Wrap = wdFindContinue
.Format = False
.MatchCase = False
End With
Selection.Find.Execute Replace:=wdReplaceAll
```

В данном примере вначале объекту VBA "Find", являющемуся подобъектом объекта "Selection", задаются необходимые свойства ".Text", ".Forward", ".Format" и другие. Затем выполняется метод ".Execute" для объекта "Selection.Find" с параметром "Replace:=wdReplaceAll"(т.е. "Заменить все"). Команда With … End With позволяет не писать для каждого свойства или подобьекта полное название соответствующего обьекта, что дает возможность экономить место и делать программу лучше и быстрее работающей.

Свойства и методы по-разному отображаются в контекстной подсказке. Так, против названий свойств стоит серый символ указывающей руки, а против названий методов - зеленый значок летящей коробки (см. на рис.1.8 в предыдущей главе).

Событие - это то, что "происходит с объектом помимо его воли", "результат действия какого-то другого метода". Это "все, что случается" с объектами по милости пользователя или программы. Нажатие кнопки, набор буквы, клик мыши, наступление какого-либо момента времени, завершение работы любой другой программы - все это события.

События - основа работы любой программы. И даже если программа должна работать автономно (например, планировщик заданий), то в качестве событий используется наступление того или иного времени на системных часах.

Так, когда пользователь нажимает кнопку на форме, происходит событие нажатия кнопки.

Для каждого события можно написать программу, которая будет срабатывать именно тогда, когда это событие произойдет. Особое значение понятие событий имеет при написании программы реакций формы на изменения ее компонентов, а также при описании новых классов.

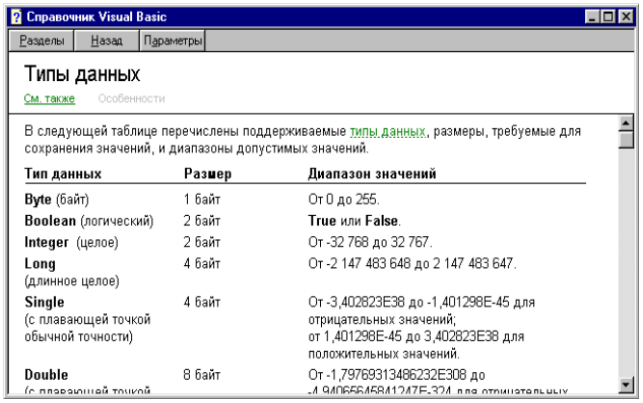
На каждое событие пишется своя подпрограмма, которая это событие "обрабатывает" - то есть, выполняет определенные действия, которые должны быть выполнены, когда это событие происходит.

2.2. Типы данных и объявление переменных

Все переменные в программе на VBA могут содержать какую-нибудь информацию: текст, число, утверждение истинности и др. В зависимости от вида содержимого они подразделяются на типы данных, - соответственно текстовые, числовые, булевы. В VBA определено большое количество различных типов данных, которые может содержать переменная. Вы можете получить информацию о возможных типах данных VBA из Справки, найдя в Предметном указателе раздел "Типы Данных" (рис.2.1).

Так, переменная типа Integer - это целое число -32 768 до 32 767, а переменная типа String - это строка текста длиной до двух миллионов символов. Все возможные в VBA типы данных подробно описаны в справке по этому языку.

Рис.2.1. Справка по типам данных VBA



В зависимости от типа данных с переменной можно производить те или иные действия и вычисления - с числовыми математические, с текстовыми - текстовые (выделение подстроки из строки, получение отдельных символов из строк и др.)

Для того чтобы переменная могла использоваться в программе, она должна быть объявлена - то есть, указана в соответствующем разделе модуля, который так, и называется - Описания и располагается в самой верхней его части, до начала первой программы (рис.2.2). Для этого нужно вставить туда описание переменной вида "Dim x(переменная) As (тип данных)": Dim a As Integer. Вместо слова Dim могут также использоваться слова Public и Private.

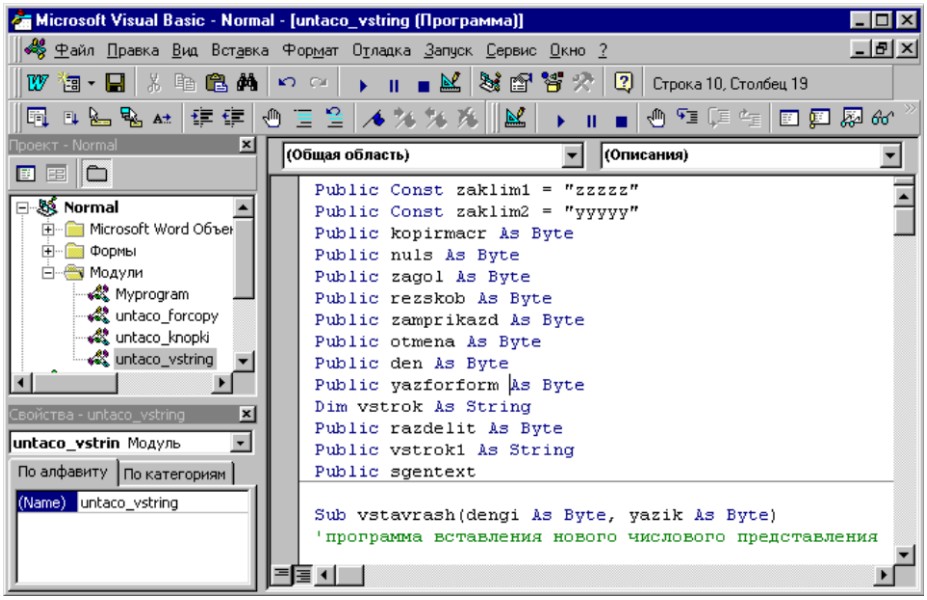


Рис.2.2. Область описаний

Переменная, обьявленная как Public, может использоваться и программами из других модулей и форм, при этом она своего значения не теряет, а переменные, обьявленные как Private или Dim, могут использоваться только в программах модуля, в котором они обьявлены. При наличии такой же переменной в программах другого модуля ее значение не будет передаваться в них при их вызове из программы модуля, где она обьявлена.

Обьявление типа переменных нужно программе, чтобы она заранее отвела под них именно столько оперативной памяти, сколько нужно для размещения именно такой переменной. Для логической - хватит и одного бита, для буквенной - нужен байт, а для строковой или числовой - более большие обьемы. В VBA можно не указывать вообще тип переменной, и тогда интерпретатор программы отведет для нее максимально возможную область данных, что, впрочем, не имеет особого значения из-за большого обьема физической памяти на современных компьютерах.

Как Public или Private переменная может быть обьявлена только в разделе описаний переменных. С помощью же инструкции (команды, не выполняющей реальных действий) Dim она может быть обьявлена и в тексте программы, но тогда ее другие модули уже использовать не смогут никак.

Можно вообще не обьявлять специально переменные, включая их в текст программы по мере надобности, так как тогда VBA присвоит им автоматически специальный для таких случаев тип данных Variant, который отличается еще тем, что переменные с таким типом автоматически могут преобразовываться в нужный другой тип, например, если переменная "x" имеет тип Variant и значение 2 (число), то после команды "x=Str(x)", которая преобразует число в строку, она уже будет иметь значение "2", то есть текстовая строка с цифрой "2". Такое преобразование было бы невозможно, если бы переменная "x" была обьявлена как Integer.

Может показаться, что определять все переменные как тип Variant или вообще не включать в модуль раздел описаний является очень хорошим решением, однако стоит помнить, что переменные типа Variant занимают очень много места в памяти, а необьявленные переменные не могут быть использованы в других модулях или формах. Можно запретить для себя использование необьявленных переменных, вставив в начало модуля, в верх раздела описаний инструкцию Option Explicit, и тогда программа остановится и перейдет в режим отладки, если вдруг наткнется на такую переменную.

Стоит быть внимательнее с типами данных, так как в некоторых случаях неправильное их использование может дать ошибку. Например, переменная с типом Integer не может превысить значение 32 767; если же надо использовать большие числа, то следует определить ее тип как Long, а если она может быть еще и дробной, то - Double.

2.3. Процедуры и функции

В принципе любая программа для каждого конкретного случая может быть написана в виде одного длинного блока текста. Однако такой код будет крайне нерациональным. Например, в программе необходимо несколько раз подсчитать сумму всех членов арифметической прогрессии для разных чисел. Можно, конечно, считать результат каждый раз по известной формуле, но это очень усложнит код. А если неизвестно, для какого количества чисел надо считать эту сумму (например, это определяет пользователь), то написание программы становится почти невозможным. Поэтому в VBA, как и в других современных языках программирования, есть понятие процедур и функций, иначе называемых **подпрограммами.**

Процедура - это программа, которая может быть вызвана из другой программы и при этом получить для дальнейшей обработки определенную информацию. Например, можно написать процедуру для вычисления суммы всех членов арифметической прогрессии и помещения результата в текст активного документа, и вызывать ее всякий раз, когда надо подсчитать такую сумму, передавая процедуре число, для которого такая сумма считается.

Функция же - это программа, которая не только получает из другой программы какую-нибудь информацию, но и возвращает той программе определенное значение, которое может быть ею в дальнейшем использовано.

Можно сказать, что программа, не использующая вызов процедур, состоит из одной большой процедуры.

В VBA любая процедура, получающая данные из другой программы, имеет следующий формат:

```
Sub "Название процедуры"("Получаемая переменная" As "Тип данных получаемой переменной")
"Текст процедуры"
End Sub
```

Получаемых переменных может быть несколько. Для каждой из них желательно указать ее тип, но это можно опустить.

Вызов процедуры происходит так же, как и вызов любой команды VBA - путем указания ее названия и передаче ей соответствующих значений переменных:

```
Sub Main()
 Dim a As Integer
   a = 1
   uvelich a, 3
   MsgBox a
End Sub
Sub uvelich(b As Integer, c As Integer)
   b = b + c
End Sub
```

В этом примере из процедуры **Main** вызывается процедура **uvelich**, которой передаются два параметра - a (равное 1) и второй - число 3. Процедура **uvelich** увеличивает первую переданную переменную на значение второго переданного числа, а затем **Main** отображает результат.

Стоит помнить, что если в заглавии процедуры указываются типы данных переменных, то и в вызывающей процедуре передаваемые значения должны быть определены и иметь тот же тип, иначе VBA выдаст сообщение об ошибке[1].

Переменные могут передаваться в процедуру двумя способами - только для чтения или и для изменения. По умолчанию переменные могут в функции изменяться. Как, скажем, в вышеприведенном примере - переменная "a" была передана в процедуру (под именем "b", чтобы лучше проиллюстрировать этот момент) и там изменилась (к ней прибавили величину переменной "c"), и затем в исходной программе она тоже стала иметь новое измененное значение.

Если же не нужно, чтобы переменная в процедуре менялась (скажем, процедура использует переменную для каких-то своих нужд, не связанных с исходной программой), то перед именем этой переменной в заголовке процедуры следует поместить инструкцию **ByVal.** Тогда процедура просто использует переменную так, как в этой процедуре описано, возможно, изменив ее значение, а программа, вызвавшая процедуру, дальше будет работать с прежним значением переменной.

К примеру, если бы заголовок процедуры **uvelich** в вышеприведенном примере имел вид **Sub uvelich(ByVal b As Integer, c As Integer),** то никакого увеличения переменной "a" не произошло бы и программа отобразила бы в качестве результата число 1. Однако в самой процедуре **uvelich** соответству-

[1] Кроме того, при вызове процедуры или функции иногда требуется указывать команду Call (подробнее смотрите в справочной системе).

ющая переменная увеличилась бы на 3 и, если бы последняя ее команда была бы **MsgBox b**, то она отобразила бы значение 4.

Функция отличается от процедуры тем, что она возвращает определенное значение, которое может быть использовано в дальнейшей работе программы. В частности, вышеприведенный пример мог иметь такой вид:

```
Sub Main()
 Dim a As Integer
 Dim d As Integer
   a = 1
   d = uvelich(a, 3)
   MsgBox d
End Sub
Function uvelich(ByVal b As Integer, ByVal c As Integer) As Integer
   uvelich = b + c
End Function
```

Как нетрудно видеть, программа присваивает переменной "d" значение, вычисленное функцией.

При создании функций опять-таки следует помнить, что если в заголовке функции не указана инструкция **ByVal** перед описаниями переменных, то соответствующие переменные могут быть в функции изменены и после ее выполнения в исходной программе они будут иметь уже измененные значения. Так, если бы данный пример имел вид

```
Sub Main()
 Dim a As Integer
 Dim d As Integer
   a = 1
   d = uvelich(a, 3)
 MsgBox d
 MsgBox a
End Sub
Function uvelich(b As Integer, c As Integer) As Integer
   uvelich = b + c
   b=1555
End Function,
```

то программа бы в качестве значения переменной "a" отобразила бы число 1555, а не 1.

Указания типов переменных в заголовках функций и процедур необязательны - их можно опустить. В этом случае будет считаться, что все передаваемые переменные могут иметь любой тип (точнее, они просто преобразуются в тип Variant и такими уже используются в подпрограмме). Однако необходимо строго следить за соответствием типов переменных в заголовках подпрограмм и в вызывающей их программе. Так, если в заголовке функции указано, что первая переменная, передаваемая ей, имеет тип Integer, то та переменная, которая передается в функцию как первая (в последнем примере - "a"), должна быть определена именно как Integer до вызова функции (что мы и видим в этом примере).

Перед заголовком функции или процедуры можно поставить инструкции Public или Private ("Private Function uvelich(b As Integer, c As Integer) As Integer"). Функция или процедура, обьявленная как Public, может вызываться и из других модулей, в то время как функция или процедура, обьявленная как Private, доступна только из данного модуля.

По умолчанию все функции и процедуры считаются обьявленными как Public (а переменные - обьявленные как Private!).

Если в программе есть вложенные процедуры или функции (то есть процедура или функция вызывает другую процедуру или функцию, которая, в свою очередь, вызывает еще одну процедуру или функцию и.т.д.), то их взаимоотношения ("кто кого вызывает?") удобно при отладке отслеживать с помощью окна **Стек Вызова** (рис.2.3), в котором видны все произошедшие вызовы.

В выпадающем меню в правом верхнем углу окна программы (см. на рис.2.2) перечислены все процедуры и функции открытого модуля. Это меню можно использовать для быстрого перехода к необходимому месту модуля, а также в раздел описаний переменных, указание на который стоит в этом меню первым.

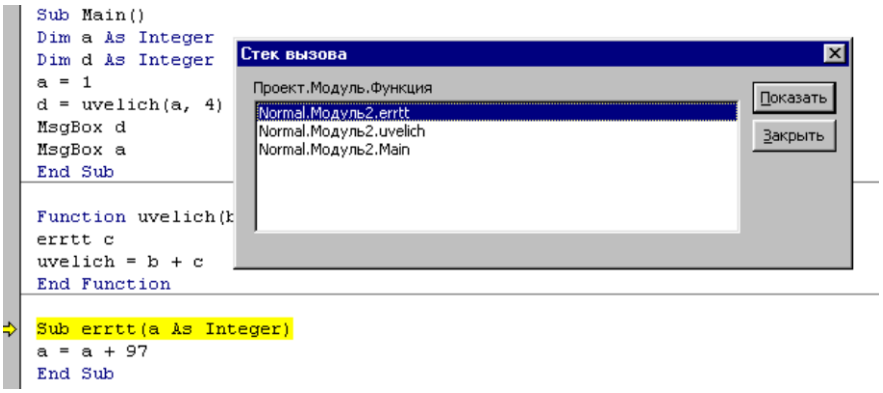


Рис.2.3. Окно Стек вызова.

Главная процедура - собственно программа - не может иметь параметров (в ее заголовке должны стоять пустые скобки). Только таким процедурам можно назначать кнопки и сочетания клавиш для их вызова на выполнение.

Во время пошагового исполнения при отладке программы с процедурами или функциями имеется возможность отказаться от прохода по всем командам той или иной процедуры или функции, вызвав команду "Шаг с обходом" из меню Отладка или кнопкой на одноименной панели. Тогда вся процедура или функция будет выполнена безостановочно, как при обычном исполнении.

Команда "Шаг с выходом" доступна в любой процедуре или функции, включая главную, и выполняет в обычном режиме все команды до конца процедуры или функции, переходя затем в режим пошагового исполнения в вызвавшей программе.

Разбивая программу на отдельные процедуры и функции, вы получаете возможность легко использовать повторно одни и те же фрагменты кода, обращаясь к ним из программы по мере надобности. К сожалению, увеличить быстродействие программы на VBA при этом не удастся, так как при запуске в шаблоне хотя бы одного макроса он компилируется полностью (в то время как в программах, написанных на многих других языках программирования, загрузка в память функций и процедур происходит лишь по мере обращения к ним).

2.4. Работа с формами

Формы - это окна интерфейса программы. С их помощью можно сообщать пользователю необходимую информацию или получать ее от него. Для создания форм используются средства редактора VBA. Если хотите наглядно посмотреть, что такое форма - откройте окно установки параметров шрифта (меню "Формат").

Создав форму (из меню правой кнопки мыши в Менеджере проектов выберите Вставить-UserForm) или дважды щелкнув на имени существующей формы, можно попасть в окно Дизайна форм. На появившейся панели инструментов "Панель управления" (если ее нет, то ее можно вызвать из меню Вид-Панель элементов) представлены все возможные элементы формы (рис.2.4):

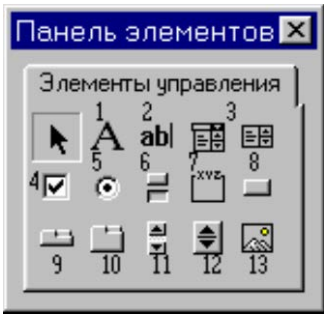


Рис.2.4. Панель инструментов с компонентами форм

1. Поле отображения текста - отображает текстовую информацию. Просто текст на форме - совет пользователю, к примеру.

2. Поле ввода текста - дает пользователю возможность ввести текстовую информацию. Потом введенные данные можно обрабатывать в программе.

3. Несколько видов списков - средства выбора одного варианта из нескольких представленных, вроде выпадающего списка шрифтов на панели инструментов "Форматирование" в Word.

Значения элементов списков задаются только в программе - при разработке формы с помощью окна "Дизайн форм" их задать нельзя.

4. Флажок - элемент, который независимо от других может находиться в трех состояниях: включенном, выключенном (могут быть определены как пользователем, так и программистом) и неактивном (определяется программой).

5. Переключатель - элемент, который также может находиться во включенном, выключенном и неактивном состояниях. В отличие от флажка, переключатели должны быть обьединены в группы с помощью элемента №7 - рамки - и, если один из переключателей в группе включен, то остальные включены быть не могут.

В Microsoft Word имеются диалоговые окна, содержащие как флажки, так и переключатели. Так, почти все вкладки меню Сервис-Параметры состоят из флажков, а меню Вставка-Сноска - из переключателей.

6. Выключатель. Это кнопка, которая может находиться в нажатом или отжатом состоянии. Например, в Word так себя ведут кнопки показа скрытых символов или форматирования текста курсивом, жирным шрифтом, подчеркиванием.

7. Рамка. Отображает прямоугольник с заголовком. В основном используется для обьединения групп переключателей (тех, что под номером 5 в нашем списке). В одной форме может быть несколько рамок и, как следствие, несколько групп переключателей, могущих действовать независимо друг от друга.

8. Командная кнопка. Обычная командная кнопка вроде кнопок "Ok" или "Отмена" в любом диалоговом окне.

9. Набор вкладок и 10. Набор страниц. Представляют из себя набор страничек-вкладок вроде вкладок "Общие", "Сохранение", "Печать" и др. в диалоговом окне Word Сервис-Параметры. Отличие между этими двумя элементами состоит в том, что 9-й элемент при переключении на другую страничку совершенно не затрагивает другие элементы формы, даже и находящиеся на нем, и изменения в состоянии других элементов можно задать только программно. 10-й же элемент при своем переключении принудительно (т.е. вне зависимости от программы) скрывает элементы на одной своей странице и показывает элементы на другой.

Если привести пример, то 9-й элемент может использоваться в программе - базе данных, отображая вкладки с именами сотрудников. На каждого сотрудника имеется стандартная информация в нескольких полях отображения текста, и количество этих полей для всех одинаково. В этом случае разумнее использовать именно 9-й элемент, соответственно изменяя содержимое полей отображения текста при переключении вкладок, чем создавать с помощью 10-го элемента набор страничек с отдельными полями отображения текста на каждой, что приведет к резкому увеличению и усложнению программы.

К сожалению, в настоящее время не сложилась твердая номенклатура этих двух элементов форм VBA, поэтому в справочной системе, в различной литературе можно встретить разные названия для каждого из них.

11. Полоса прокрутки. Такая же, как и в окне Word. Может передавать в программу число, равное расстоянию в пунктах от ее начала. Верхний предел расстояния неограничен.

12. Счетчик. Две нажимающиеся кнопки со стрелками. Может передавать в программу свое значение от 1 до 100.

13. Рисунок. В этот элемент можно вставить из файла рисунок, который будет храниться в форме (и шаблоне или документе с ней).

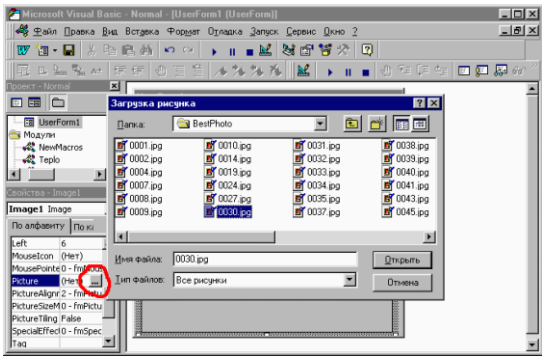


Рис.2.5. Вставка рисунка.на форму

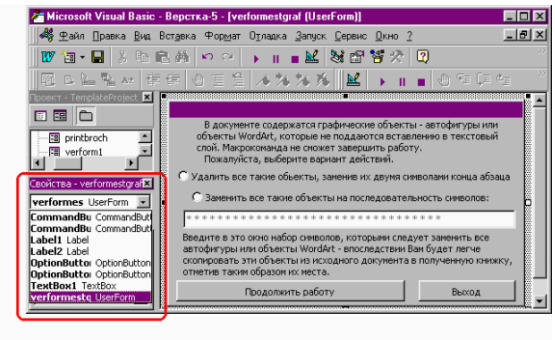


Рис.2.6. Выпадающее меню Окна свойств - навигатор по элементам формы

(Для вставки самого рисунка нужно после помещения на форму элемента управления Рисунок в Окне свойств выбрать пункт "Picture", нажать на кнопку слева от него и в появившемся окне открытия файла выбрать нужную картинку - рис.2.5)

У каждого элемента имеется свое уникальное имя, а также набор свойств. Их можно посмотреть в Окне свойств, выделив элемент и выбрав из контекстного меню правой кнопки мыши пункт "Свойства".

Стоит внимательно изучить все возможные свойства для каждого элемента, по возможности осознав их предназначение. При необходимости можно вызвать справку по каждому свойству, поставив на него курсор и нажав F1.

Большой набор свойств также имеет сама форма. С помощью выпадающего меню в Окне свойств можно быстро перейти к свойствам необходимого элемента (рис.2.6).

Среди наиболее распространенных свойств, имеющихся почти у всех элементов - **Caption** (т.е. надпись на поверхности или в заголовке, если он есть), **Top** и **Left** - координаты верхнего левого угла элемента, **Height** и **Width** - высота и ширина, **Enabled** - доступность для изменений пользователем, **TabIndex** - число, показывающее, в какую очередь на данный элемент перейдет фокус (т.е. возможность изменения пользователем содержания или состояния) при переходах между элементами с помощью клавиши табуляции (можно запретить такой переход на какой-либо элемент, указав его свойство **TabStop** как False). Обратите внимание на свойства **Picture, PictureAlignment, PictureSizeMode, PictureTiling** собственно формы - с их помощью на поверхность формы можно поместить какой-нибудь фоновый рисунок!

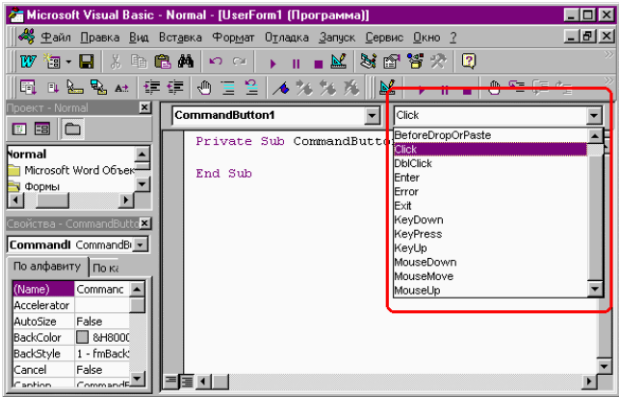
Необходимо помнить, что создание элемента - только начало работы над формой. Сам по себе элемент никаких действий не выполняет, за исключением тех, которые его определяют. Кнопка - нажимается под мышкой или Enter'ом, список показывает значения, во флажке можно поставить и убрать галку, но не более того! Для того, чтобы элемент выполнял какое-нибудь действие при действии с ним, нужно написать для него программу **реакции на события.**

Событий, могущих произойти с элементом, много. Это и двойной щелчок мыши на нем, это и проводка мыши над ним, это и ввод текста в поле ввода, это и нажатие на вкладке, это и переход к другому элементу… Для того, чтобы написать программу обработки реакции на событие, нужно два раза щелкнуть на элементе. Откроется окно написания программ реакций на события, и автоматически написанная заготовка будет иметь вид:

```
Private Sub CommandButton1_Click()
End Sub
```

Теперь в текст этой процедуры можно вводить команды, которые выполнятся, если событие в заголовке - Click мыши - произойдет. Это событие считается стандартным для кнопок, флажков, рисунков, рамок, переключателей и окон отображения текста. Для остальных элементов - полей ввода текста, полос прокрутки, счетчиков и др. - стандартным считается событие Change, то есть их изменение.

Рис.2.7. Выпадающий список событий



Список возможных событий для элемента можно получить, поставив курсор внутрь процедуры обработки стандартного события и выбрав нужное из выпадающего меню справа вверху окна написания программы. В выпадающем меню слева вверху этого окна указаны все процедуры и функции программы реакции формы на события (рис.2.7).

В программе реакции формы на события могут быть и другие процедуры и функции, не связанные с элементами формы.

У некоторых событий есть параметры, с которыми информация об его происшествии передается программе. Так, процедура обработки события KeyDown (то есть нажатия какой-либо клавиши тогда, когда активен данный элемент) имеет вид

```
    Private Sub CommandButton1_KeyDown(ByVal KeyCode As
MSForms.ReturnInteger, ByVal Shift As Integer)
    End Sub
```

Переменная KeyCode будет после срабатывания данной процедуры будет иметь значение, равное коду нажатой клавиши, а переменная Shift - 1, если соответствующая клавиша была нажата, и 0, если нет. Эти переменные можно использовать в процедуре.

Все свойства, задаваемые в Окне свойств, можно изменять и из программы. Например, команда **TextBox1.Enabled=True** даст возможность пользователю ввести текст в поле ввода TextBox1, а команда **CheckBox1.Value=False** уберет отметку из флажка с именем CheckBox1. Эта возможность делает формы VBA динамическими - то есть содержимое форм может меняться немедленно в ответ на действия пользователя без выгрузки и повторной загрузки формы.

Форма может работать как обычное диалоговое окно программы для Windows - например, выбор пользователем значения из списка может каждый раз выводить в форму какой-либо текст, разный для разных значений.

Наиболее часто возможность создания динамических форм используется для инактивации элементов, не могущих быть использованными при каких-либо условиях. К примеру, в форме, состоящей из поля ввода текста и кнопки, вызывающей процедуру вычисления квадратного корня, кнопка может становиться неактивной и не реагирующей на нажатия, если число, введенное в поле ввода, отрицательное.

Для просмотра формы в режиме выполнения (то есть как она будет выглядеть в готовой программе) надо нажать клавишу F5 - ту же, что и для запуска на выполнение обычной программы. Все средства отладки, описанные выше, работают и при отладке форм.

Стоит иметь в виду, что после выгрузки формы (методом **Unload Me**) все переменные, описанные в программе реакции формы на события, обнуляются. Поэтому после выгрузки формы получить от нее какие-либо данные уже нельзя. Если в результате работы формы в основную программу должны передаваться какие-либо данные, то нужно в основной программе обьявить несколько переменных с помощью инструкции Public, а затем в программе реакции формы на события записать в них всю необходимую информацию, полученную от пользователя, и использовать значения этих переменных в основной программе.

Программа реакции формы на события представляет собой полноценный модуль проекта за одним исключением: процедуры из него могут быть вызваны только из другой программы. Вызов формы на исполнение с помощью кнопки, строки меню или сочетания клавиш невозможен. Поэтому для вызова формы необходимо в основной программе (если все необходимые функции реализованы в процедурах программы реакции формы на события, то основную программу все равно необходимо создать, хотя бы только для вызова этой формы) надо указать команду **ИмяФормы.Show**, а кнопки и сочетания клавиш назначать именно этой основной программе.

Формы, так же как и модули, могут быть сохранены в отдельных файлах путем экспортирования функцией "Экспорт файла" своего контекстного меню правой кнопки мыши в Менеджере проектов или путем перетаскивания ее названия из Менеджера проектов в Explorer. Но, в отличие от модулей, при сохранении форм получается два файла, в одном из которых содержится текст программы реакции формы на события, а в другом - информация о расположении элементов на форме, их свойствах, а также рисунки и фон формы. Поэтому при импортировании формы необходимо, чтобы оба этих файла были в одной папке.

2.5. Операторы цикла и перехода

Если необходимо повторить ту или иную группу операций несколько раз, то используются операторы цикла: Do...Loop, For...Next, For Each...Next и While…Wend.

Если необходимо совершить те или иные действия в зависимости от наличия или отсутствия выполнения какого-либо условия или в зависимости от значения той или иной переменной, используются операторы условия: If...Then...Else и Select Case.

Если необходимо совершить переход к другой части той же программы, то используется оператор GoTo "имя метки". Там, куда должен быть совершен переход, на отдельной строке должна стоять метка с соответствующим именем, а после нее - двоеточие.

Подробное описание функций, особенностей и синтаксиса всех этих операторов можно получить в справочной системе редактора VBA, поэтому не стоит здесь на них останавливаться. Просто наберите интересующий вас оператор в окне Редактора VBA, установите на него курсор и нажмите кнопку F1.

* * *

Данный текст - глава из книги "Тайны и секреты компьютера", недавно вышедшей в издательстве "Радио и связь". Эта книга предназначена для тех, кто самостоятельно осваивает мир информационных технологий. Программирование в среде Microsoft Office, создание сайтов, устройство сети Интернет, структура системного реестра Windows и файловой системы, сеть Fidonet, строение жидкокристаллических дисплеев и проблема наличия различных кодировок русского языка, - про все это рассказывается в ней. Многообразие тем и легкий стиль изложения сделают ее вашим спутником на долгое время, и вы всегда сможете найти в ней нужную именно в данный момент информацию. Если Вы интересуетесь компьютерными технологиями, желали бы расширить свои знания и умения в этой области, то она Вам наверняка понравится. На сайте http://comptain.chat.ru, посвященном этой книге, вы можете ознакомиться с ее оглавлением и аннотацией, прочитать некоторые главы, купить в Интернет-магазине.