Глава 56

Больше интерактивности: настраиваемые диалоговые окна

В этой главе...

- Использование Office для работы в Web
- Настройка окон сообщений и диалоговых окон
- Создание и программирование форм
- Редактирование и настройка элементов управления
- Использование новых технологий и элементов управления ActiveX

В других языках программирования, окна и диалоговые окна являются точками доступа, которые обеспечивают взаимодействие между пользователем и программой во время выполнения программы. В VBA это проявляется в меньшей степени; можно вставить свою VBA-программу в пользовательский интерфейс используемого приложения VBA. Но даже при таком положении дел возникает множество ситуаций, которые требуют применения настраиваемых диалоговых окон. В этой главе описываются навыки, необходимые для создания таких диалоговых окон.

Вначале речь пойдет о некоторых простых элементах VBA: окнах сообщений и окнах ввода данных. Если этого мало, то можно создать свои формы и диалоговые окна, сколь угодно сложные. После вступления, описывающего основы создания форм, все внимание будет сконцентрировано на элементах управления (элементы формы, которые щелчки мыши или нажатие клавиши заставляет выполнять какие-либо действия). И в завершение, будет затронута крайне важная тема написания такого кода, который “заставит” формы и элементы управления выполнять необходимые действия.

Простое взаимодействие с Word

Большой выбор инструментов для разработки форм VBA позволяет создавать настоящие диалоговые окна в стиле Windows и другие окна, необходимые пользователям. Если можно выполнить работу с минимальными физическими затратами, то лучше сделать именно так. Две функции VBA — `MsgBox` и `InputBox` — обеспечивают базовые средства с целью информировать пользователей и получать их отзывы. Основное назначение этих функций таково:

- `MsgBox` выводит сообщение, кроме этого, позволяет узнать на какой из двух или более кнопок щелкнул пользователь;
- `InputBox` выводит сообщение и текстовое окно, в котором пользователь может ввести свой ответ.

Вывод на экран окон сообщения

Формальный синтаксис функции MsgBox выглядит следующим образом:

```
MsgBox(строка[, кнопки][, заголовок][, файл справки, содержание])
```

Скобки указывают на то, что только параметр строка, определяющий, какое сообщение появится на экране, является обязательным.

Определение строки

В простейшем виде функция MsgBox действует как оператор. Все, что нужно сделать пользователю, — ввести этот оператор, добавив текст, который необходимо вывести в качестве единственного аргумента, например:

```
MsgBox "Это текст MsgBox."
```

Сообщение можно заключить в кавычки, но они не требуются, если функция применяется как оператор, т.е. когда пользователь не использует возвращаемое значение. Строка может быть литералом, переменной или любым выражением, как в следующем примере:

```
Sub WishCountDown()
    intWishCount = 3
    datWhen = Format(Now, "Short date")
    strInfo1 = "Из "
    strInfo2= " осталось желаний."
    MsgBox strInfo1 & datWhen & ", " & intWishCount _
        & strInfo2
End Sub
```

Чтобы вывести сообщение, состоящее из нескольких строк, нужно разделить строки, добавив несколько символов переноса (соответствует значению ASCII 13) в строку и используя для этого функцию Chr. Аналогично, можно выстроить текст в несколько столбцов при помощи символов табуляции (значение ASCII 9), вставленных при помощи функции Chr.

```
Sub LinesAndColumns()
    MsgBox "Вот первая строка." & Chr(9) & _
        "Второй столбец." & Chr(13) & _
        "Вот вторая строка." & Chr(9) & "И столбец тоже."
End Sub
```

Как сделать окно сообщения интереснее

Помимо вывода на экран текста, окно сообщения может содержать одну из нескольких пиктограмм и включать в себя кнопки различных типов. Для определения

используемых кнопок пользователь вводит числовое значение в качестве необязательного параметра кнопки. С помощью пиктограммы пользователь может украсить свое окно сообщения. На рис. 56.1 показан пример, в котором окно сообщения содержит пиктограмму сообщения об ошибке, что, несомненно, привлечет внимание пользователя. По умолчанию окно сообщения имеет только кнопку OK, но можно также добавить кнопки OK, Cancel, Yes, No, Abort, Retry и Ignore в различных сочетаниях.

Рис. 56.1. Окно сообщения, обладающее пиктограммой и несколькими кнопками

Вычисление значения параметра кнопки

Пользователь вычисляет значение параметра кнопки, складывая вместе константы, представляющие различные комбинации пиктограмм и кнопок. Можно вычислить значение самому, однако проще создать выражение, которое использует именованные константы, определенные VBA для этой цели. В табл. 56.1 отображены все константы с их числовым значением и предназначением. Если основываться на таблице, то аргумент кнопки для окна сообщения, изображенного на рис. 56.1, должен быть 531. Вместо этого можно набрать следующий оператор:

```
int A = MsgBox("Нажмите кнопку", VbYesNoCancel + VbCritical +
VbDefaultButton3, "Библия Office XP")
```

В этом примере MsgBox используется как обычная функция, присваивая значение кнопки нажатой пользователем переменной intA (см. следующий раздел “Кто нажал кнопку”). Заметим, что VbDefaultButton3 — третья константа в выражении параметра кнопки — устанавливает третью кнопку в качестве выбора по умолчанию (считая слева направо). В данном случае третьей кнопкой является Cancel (Отмена). На рис. 56.1 фокус сосредоточен на кнопке Cancel, она выделена пунктиром, подчеркивающим, что кнопка может быть активизирована нажатием пробела или клавиши <Enter>.

Таблица 56.1. Константы VBA, описывающие внешний вид и поведение окон сообщений и ввода данных

Числовая константа	*Значение*	*Действие*
VbOKOnly	0	Отображает только кнопку OK
VbOKCancel	1	Отображает кнопки OK и Cancel
VbAbortRetryIgnore	2	Отображает кнопки Abort, Retry и Ignore
VbYesNoCancel	3	Отображает кнопки Yes, No и Cancel
VbYesNo	4	Отображает кнопки Yes и No

VbRetryCancel	5	Отображает кнопки Retry и Cancel
VbCritical	16	Отображает пиктограмму Ошибка
VbQuestion	32	Отображает пиктограмму Вопросительный знак
VbExclamation	48	Отображает пиктограмму Восклицательный знак
VbInformation	64	Отображает пиктограмму Информация
VbDefaultButton1	0	Определяет первую кнопку как используемую по умолчанию
VbDefaultButton2	256	Определяет вторую кнопку как используемую по умолчанию
VbDefaultButton3	512	Определяет третью кнопку как используемую по умолчанию
VbDefaultButton4	768	Определяет четвертую кнопку как используемую по умолчанию

Кто нажал кнопку

Задача кнопок в окне сообщения — предоставить пользователю выбор. Конечно, необходим способ определить, какая кнопка была выбрана. Это легко, так как функция MsgBox возвращает целое значение, отвечающее кнопке, которую нажал пользователь. Чтобы не напрягать память, можно сверять возвращаемые значения с предопределенными именованными константами, а не с числами. Ниже приведен список констант и их “действительные” значения:

Константа	*Значение*
vbOk	1
vbCancel	2
vbAbort	3
vbRetry	4
vbIgnore	5
vbYes	6
vbNo	7

Чтобы определить, какая кнопка была нажата, вполне подойдет оператор If...Then, если кнопок две, как в примере:

```
If MsgBox ("Продолжим?", VbYesNo) = VbYes Then
    DoSomething
Else
    DontDoAnything
End If
```

Если используются три кнопки, обратитесь к оператору `If...Then...Else If...`.

Заголовок

По умолчанию строка заголовка окна сообщения отображает имя используемого приложения VBA. Заголовок можно изменить на любую другую строку, вызвав функцию `MsgBox` с параметром `заголовок` (см. рис. 56.1).

Получение информации от пользователя

Если необходимо узнать от пользователя более трех вариантов ответа, можно воспользоваться функцией `InputBox`. Ниже приведен ее формальный синтаксис, не включены лишь редко встречающихся необязательные аргументы:

```
InputBox(строка[, заголовок][, значение по умолчанию])
```

Как видно из рис. 56.2, диалоговое окно, выводимое этой функцией, содержит текстовое поле, где пользователь может ввести какую-либо предположительно важную информацию. Чтобы ввести эту информацию в программу, нужно просто присвоить значение, возвращаемое функцией `InputBox` строковой переменной:

```
strB = InputBox ("Предпочтительное место?", "Авиалинии России", "у прохода")
```

Рис. 56.2. Окно ввода данных применяется для получения информации от пользователя с целью дальнейшего использования этой информации в программе

Хотя функция `InputBox` может получать намного больший объем информации, чем `MsgBox`, ее основные действия проще — тут нет ни пиктограмм, ни кнопок. Параметры `строка` и `заголовок` работают абсолютно так же, как и в функции `MsgBox`. Можно дополнительно облегчить жизнь пользователю, предусмотрев ответ по умолчанию. Если он подходит, достаточно нажать клавишу <Enter>.

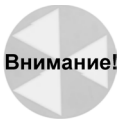
Проектирование форм

Когда окна сообщений и ввода данных уже не обеспечивают необходимую функциональность, можно создать пользовательскую форму. По сравнению с созданием кода или даже обыкновенных предложений, создать форму VBA может даже ребенок. Хотя, не зная некоторых тонкостей, можно столкнуться с неприятностями.

Запуск форм

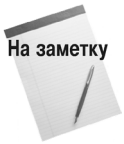
После создания формы ее можно “запустить” (нажав <F5> или щелкнув на кнопке Run (Выполнить) панели управления). После запуска форма возникает на экране поверх используемого приложения VBA (редактор Visual Basic на это время исчезает).

Форму можно оттестировать, пощелкав на ее элементах управления и посмотрев на результаты. Помните только, что если форма является частью большей программы, возможно, она не будет работать, как следует.



Перед запуском полностью выделите форму. В редакторе Visual Basic можно выделить форму частично, щелкнув в любом ее месте, выделение возникнет вокруг всей формы, но она не запустится по нажатию клавиши <F5>. Если внезапно появится диалоговое окно Macros (Макросы), закройте его и щелкните в любом месте формы перед попыткой запустить ее вновь.

Чтобы остановить запущенную форму, в которой отсутствует кнопка Cancel, щелкните на кнопке закрытия в правой верхней части окна, а чтобы вернуться в редактор Visual Basic, нажмите комбинацию клавиш <Alt+Tab> и щелкните на кнопке Reset (Сброс) на панели управления.



Формы и элементы управления — полноправные объекты VBA. Что касается свойств форм и элементов управления, их, как выяснится из этой главы, не нужно устанавливать в коде.

Помните: необходимо написать код для каждого добавляемого в программу элемента управления, за исключением некоторых надписей и элементов управления фреймов. Формы также требуют наличия кода. Учтите этот факт, продумывая свой проект.

Планирование форм для программы

Создавать формы в VBA легко и просто, но их разработка требует тщательного обдумывания. Нужно помнить, что формы являются частью большей программы, перед которой стоят свои задачи.

Поэтому перед работой с формами определите задачу программы в целом и роль форм для выполнения этой задачи. Обдумайте, как сгруппировать эти задачи в различных формах.

Распечатка форм в процессе создания

В работе по созданию форм иногда полезно иметь с собой распечатанные копии, чтобы обдумывать их на досуге. Распечатанные копии великолепно подходят для обдумывания новых дизайнерских решений, на них можно делать пометки, их можно показывать программистам и пользователям, чтобы узнать их мнение.

Чтобы распечатать формы проекта в редакторе Visual Basic, выполните следующие действия.

1. Выберите нужную форму (или формы). Если нужно распечатать одну форму, выберите ее в окне Project Explorer. Чтобы распечатать все формы проекта, выберите любую форму, модуль или другой компонент проекта.
2. Выберите команду File⇨Print (Файл⇨Печать) или нажмите комбинацию клавиш <Ctrl+P>. В результате появится диалоговое окно.
3. Установите флажок Form Image (Изображение формы); снимите флажок Code (Код) (если конечно не хотите заодно распечатать и код).
4. Выберите Current Module (Текущий модуль), чтобы распечатать текущую форму, или Current Project (Текущий проект), чтобы распечатать все формы проекта.
5. Щелкните на кнопке OK.

Разработка форм

Новая форма (рис. 56.3) — пустое полотно для разработки пользовательского интерфейса. Можно изменить его размер, положение на экране и цвет, заполнить элементами управления и заставить их выполнять определенные задачи.

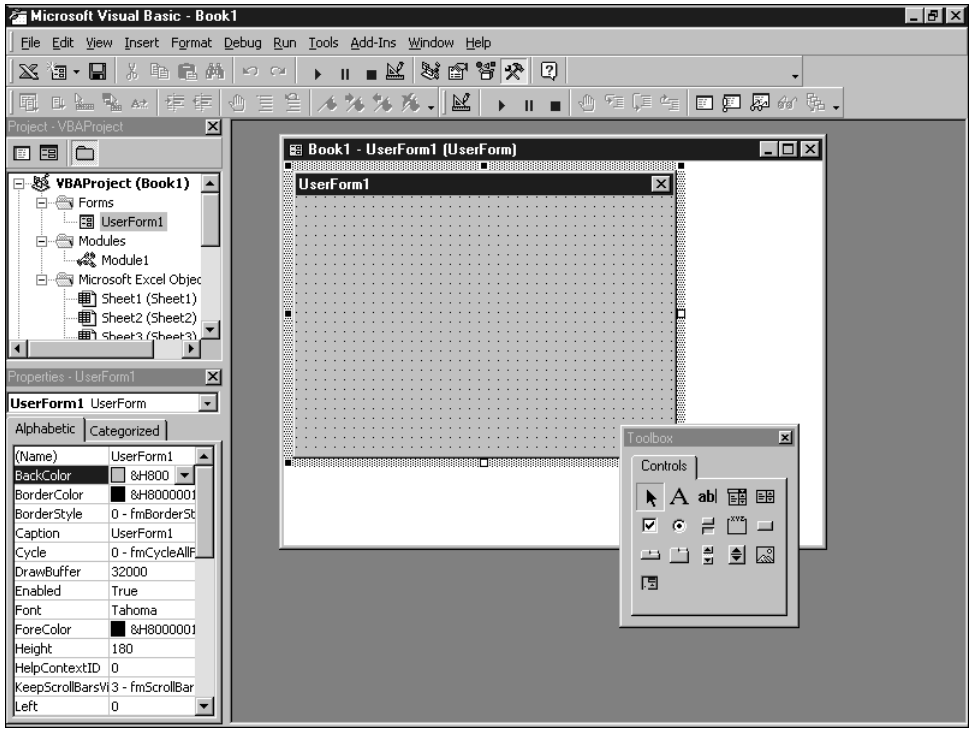


Рис. 56.3. Новая пользовательская форма. Обратите внимание на меню Toolbox и элементы управления

Перед созданием новой формы убедитесь, что для нее выбран нужный проект (в окне Project Explorer).

Создание формы

Чтобы создать новую форму, выберите команду Insert⇨UserForm (Вставить⇨Пользовательскую форму) из меню редактора Visual Basic или из

контекстного меню Project Explorer. Новая пользовательская форма появляется в новом окне. Что касается изменения свойств пользовательских форм, VBA использует обычные диалоговые окна. Для изменения заданного по умолчанию имени и заголовка пользовательской формы используйте окно Properties (Свойства).

Добавление в форму элементов управления

На пустое полотно новой пользовательской формы можно добавить *элементы управления* — элементы, которые применяются для взаимодействия между пользователем и программой. Эти элементы вы найдете в меню Toolbox (Элементы управления), которое появляется автоматически при создании новой формы (рис. 56.3). При выборе любого другого окна меню исчезает с экрана и возникает вновь при выборе окна пользовательской формы.

Если меню Toolbox отсутствует на экране, выберите команду View⇨Toolbox (Вид⇨Элементы управления).

Чтобы добавить в форму элемент управления, выберите его пиктограмму в меню Toolbox и перетащите ее в ту часть формы, где должен находиться выбранный элемент управления (рис. 56.4). после отпускания кнопки мыши элемент управления появляется в форме.

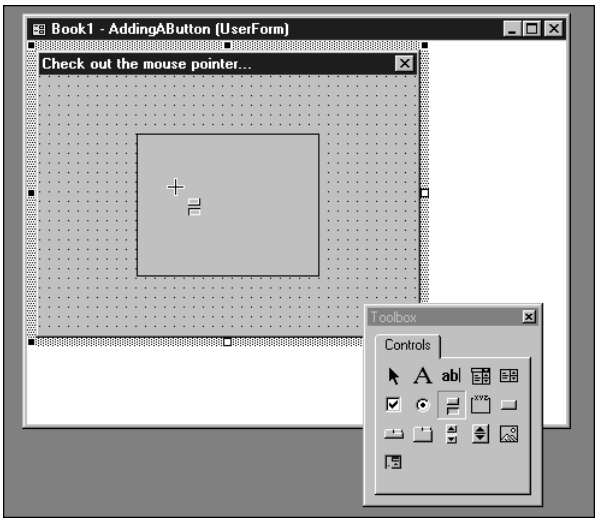


Рис. 56.4. Добавление элемента управления в пользовательскую форму

Обычно VBA возвращает курсор в привычный вид сразу после добавления нового элемента управления. Если нужно добавить в форму несколько элементов управления, дважды щелкните на пиктограмме этого элемента в меню Toolbox. Когда все нужные элементы добавлены, щелкните на пиктограмме со стрелкой, чтобы вернуться в обычный режим.

Изменение свойств формы и элементов управления

Так как формы и элементы управления VBA являются объектами, у них есть свойства, определяющие их вид и поведение. Как и в случае с другими объектами, можно проверять и изменять в коде свойства форм и элементов управления.

Однако для этих объектов не нужно писать код — окно Properties редактора Visual Basic позволяет легко контролировать множество важных характеристик форм и элементов управления, не программируя их. По мере совершенствования программ можно добавлять изменения свойств в код. И в этом случае окно Properties (рис. 56.5) предоставит более простой путь для изменения первичных свойств форм и элементов управления.

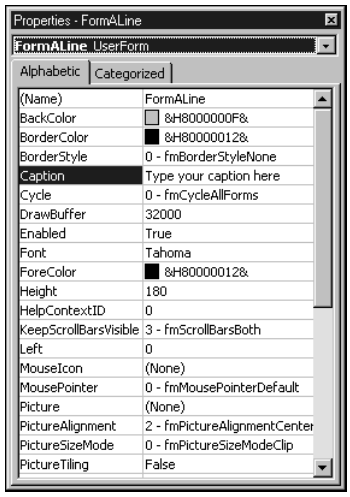


Рис. 56.5. Окно Properties формы VBA

Если окно Properties отсутствует на экране, его можно вызвать, выбрав команду меню View⇨Properties (Вид⇨Свойства) или нажав <F4>.

Знакомство с окном Properties

Окно Properties обладает такими свойствами:

- оно автоматически показывает свойства любого объекта, выбранного в окне UserForm (не важно, элемента управления или самой формы);
- позволяет работать со свойствами любого объекта — выберите лишь элемент из раскрывающегося списка в верхней части окна Properties (рис. 56.6).

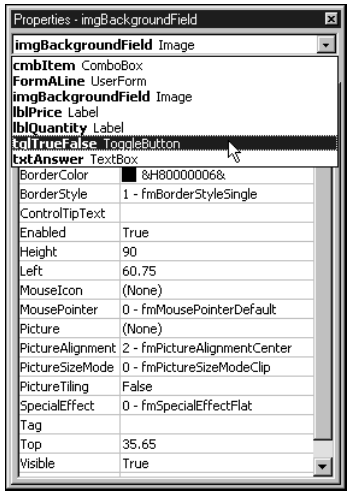


Рис. 56.6. Раскрывающийся список в верхней части окна Properties *предоставляет возможность мгновенно получить доступ к свойствам любого элемента формы (или самой формы)*

Заметим, что в верхней части окна всегда находятся имя и тип элемента, со свойствами которого идет работа в настоящий момент. Окно Properties имеет две вкладки, обе отображают один и тот же набор свойств, только вкладка Alphabetic (По алфавиту) отображает их в алфавитном порядке, а вкладка Categorized (По категориям) делит на группы. Независимо от выбранной вкладки, окно работает одинаково. Справа от имени свойства располагается поле, в котором можно изменить его значение.

Получение справки по свойствам

Если вам не известно, для чего предназначено то или иное свойство (или как оно работает), вызовите справку. Щелкните на поле свойства и нажмите <F1>, чтобы появилась соответствующая тема файла справки VBA по формам.

Изменение значений свойств

В зависимости от свойства, можно изменить значение в его поле тремя способами:

- введя новое значение;
- выбрав новое значение из раскрывающегося списка;
- выбрав новое значение в диалоговом окне.

Некоторые свойства позволяют пользователю и вводить новые значения, и выбирать их из списка.

Нельзя определить, имеет свойство раскрывающийся список или диалоговое окно, не щелкнув в поле свойства. По щелчку в поле появится кнопка со стрелкой “вниз” (если поле имеет раскрывающийся список) или троеточие (если в нем есть диалоговые окна.)

Не забудьте выбрать нужный элемент

Перед добавлением в форму элементов управления форма является единственным элементом, отображаемым в окне Properties. Во время добавления в форму элементов управления необходимо уделить внимание тому, какой выбран элемент (форма или один из элементов управления) — имя каждого элемента указано на нем в прямоугольной рамке.

Проще всего выбрать элемент, щелкнув на нем. Можно также выбрать элемент из раскрывающегося списка в верхней части окна Properties. Когда в окне Properties выбран объект, по его периметру появится пунктир выделения. Даже предпочитая работать мышью, поглядывайте в окно Properties, чтобы убедиться, что выбран нужный объект.

Основные свойства формы и элементов управления

Многие свойства форм и элементов управления работают аналогично с разными объектами и свойствами. Например, пользователь применяет подобные подходы для изменения размеров формы и управления, и все свойства, связанные с цветом, также идентичны, независимо от того, изменяется цвет формы, кнопки или текста. В этом разделе описаны наиболее важные свойства, общие для форм и элементов управления.

Имя — не значит заголовок

Каждый объект VBA имеет имя и формы. Элементы управления в этом плане не исключение. Но имя объекта в форме отсутствует. Оно используется для ссылок на объект в коде, если, например, программе нужно активизировать один из методов объекта или изменить одно из его свойств.

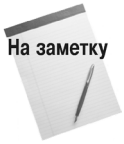
Конечно, каждый объект имеет свойство `Name`. Как и имя, присваиваемое каждому объекту, свойство `Name` содержит то имя, которое используется в коде. Форма или элемент управления должны иметь подходящее для VBA имя, т.е. такое, которое не содержало бы пробелов и знаков препинания (см. главу 53).

Свойство `Caption` является несколько иным. Заголовок формы определяет текст, появляющийся в строке заголовка; а заголовок элемента управления, если таковой имеется, определяет текст на элементе управления. Изменить свойство `Caption` можно, введя новую строку в соответствующее поле в окне Properties. В случае с элементами управления щелкните один раз на объекте, чтобы выбрать его, и затем еще раз, чтобы отредактировать заголовок прямо в элементе управления.

Изменения размеров, а также расположения форм и элементов управления

Перед тем, как начинать серьезную работу по разработке проекта, убедитесь, что в редакторе Visual Basic включена *разметка*. Разметка определяет условные “магнитные” линии: и вертикальные, и горизонтальные. При передвижении элемента или изменении его размера края автоматически подстраиваются к ближайшей линии разметки. (Подробнее см. “Работа с разметкой” далее в этой главе.)

Размер объекта — свойство, которое изменяется легче всего. Хотя можно изменить значение высоты и ширины (свойства `Height` и `Width`) в окне Properties, проще перетащить границу объекта. Перетаскивание нижней или правой границы изменяет один параметр объекта, если же перетащить правый нижний угол объекта, одновременно изменятся его высота и ширина.



Для изменения размеров используются только белые метки, а большинство меток формы — черные. Задача черных меток — помочь в расположении формы в нужном месте. Расположение формы всегда определяется положением верхнего левого угла их окна.

Изменение положения формы на экране

Может изменить место первоначального появления формы на экране при запуске программы (или только формы во время проверки) с помощью свойства `StartUpPosition`.

Значение этого свойства по умолчанию `1 - CenterOwner`. Это значит, что форма появится в центре окна приложения VBA, независимо от его размера и положения на экране. Если нужно, чтобы форма появилась по центру экрана, независимо от того, какое приложение VBA запущено, установите значение свойства `StartUpPosition` равным `2 - CenterScreen`. Или определите значение самостоятельно, выбрав `0 - Manual` и затем введя значения в полях свойств `Left` и `Top`.

Важные свойства, связанные с видом объектов

Среди свойств, заслуживающих внимания, следует упомянуть следующие.

- `Left` и `Top`. При изменении положения элемента управления в форме (обычно это происходит в процессе перетаскивания элемента управления в другое место) эти параметры изменяются, и наоборот, при изменении этих параметров меняется позиция элемента управления. Если необходима высокая точность, можно вводить числовые значения в поля свойств `Left` и `Top`.
- `ForeColor` и `BackColor`. Эти свойства отвечают за цвет формы и элементы управления.
- `Font`. Данное свойство позволяет выбрать шрифт из тех, что установлены на компьютере. Чтобы установить шрифт по умолчанию для всех элементов управления формы, установите свойство `Font` перед тем, как добавлять в форму элементы управления. Таким образом устраняется необходимость изменять свойство `Font` для каждого элемента управления отдельно.
- `SpecialEffect`. Это свойство создает различными способами слабое, но четкое ощущение объема объекта.
- `ToolTip`. Указанное свойство определяет, какой текст возникнет при наведении указателя мыши на элемент управления.
- `Picture` — позволяет добавить в элемент управления рисунок (рис.56.7). После размещения рисунка можно с помощью свойств `PictureAlignment`, `PictureSizeMode` и `PictureTiling` (если, конечно, они доступны при работе с данным объектом) настроить рисунок по вашему усмотрению.

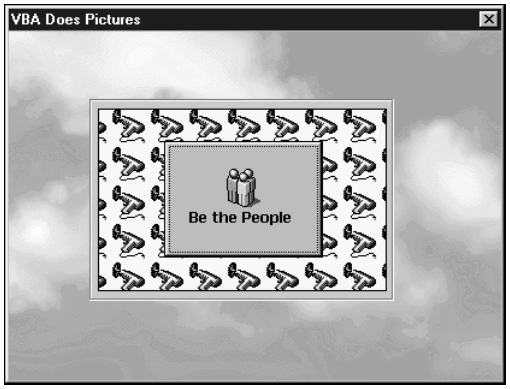


Рис. 56.7. Рисунки, добавленные в форму VBA, и некоторые из элементов управления

Использование свойств Enabled и Locked

Два свойства, `Enabled` и `Locked`, управляют доступом пользователя к форме или элементу управления. Естественно, элементы управления созданы для того, чтобы люди ими пользовались. Однако бывают ситуации, когда необходимо, чтобы элемент управления был виден (т.е. чтобы пользователь знал о его существовании), но в то же время являлся "затененным" (иначе говоря, приобрел серый цвет, который показывает, что элемент управления в данный момент не доступен).

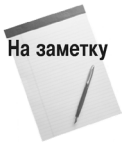
Свойство `Enabled` определяет, может ли *фокус* (имеется в виду способность элемента управления реагировать на события мыши и клавиатуры) перейти к элементу управления. В Windows в фокусе может находиться только один объект. Чтобы определить, на каком именно объекте сосредоточен фокус в данный момент, Windows размещает по периметру объекта пунктирную линию. Когда значение свойства `Enabled` равно `True`, элемент управления выглядит, как обычно, и может находиться в фокусе. В противном случае, если его значение `False`, Windows выводит затененный объект, который не может принимать фокус.

Если свойство `Locked` имеет значение `False`, элемент управления не отвечает на щелчки мыши или нажатия клавиш, невзирая на значение свойства `Enabled`. Однако если свойство `Enabled` имеет значение `True`, элемент управления может получать фокус и на экране будет выглядеть, как обычно.

Основы изменения элементов управления

Редактор Visual Basic позволяет вырезать, копировать и вставлять элементы управления по отдельности или целыми группами. Для этого применяются стандартные команды меню Windows или горячие клавиши. Кроме того, при помощи стандартных кнопок Office можно выполнять данные операции одним щелчком. В среде VBA при вставке из буфера обмена элемент управления помещается по центру формы, даже если эта часть формы в невидима. Однако если перед вставкой элемента управления выбран фрейм или многостраничный элемент управления, вставляемый элемент появится в центре выбранного элемента.

Чтобы удалить один или несколько выбранных элементов управления (не помещая их в буфер обмена), нажмите клавишу <Del> или выберите команду Edit⇨Delete (Правка⇨Удалить).



Клавиша <Backspace> при удалении не используется (в отличие от некоторых других программ).

Можно выбрать группу элементов управления и затем переместить их, изменить размер, вырезать или копировать их как один элемент или применять другие команды для их форматирования, описанные далее в этой главе. Изложенный способ также помогает устанавливать значения тех свойств объектов, которые должны быть одинаковыми. Чтобы выделить несколько элементов управления, используйте следующие способы.

- Обведите группу элементов управления рамкой выделения при помощи указателя Toolbox. Если часть элемента управления попадет в рамку, выделенным окажется весь элемент.
- Щелкните на первом элементе управления, входящем в группу, и затем, удерживая клавишу <Shift>, щелкните на элементе управления на другой стороне области выделения. Все элементы управления между этими двумя будут включены в выделение.
- Удерживая клавишу <Ctrl>, щелкните на отдельном элементе управления, чтобы добавить или исключить его из выделения.

Убрать изменения, произведенные над элементами управления, можно при помощи команды Undo (Отменить, <Ctrl+Z>). Undo не работает в случае изменения размеров формы, а также если изменения внесены в окне Properties.

Использование координатной сетки

Координатная сетка — набор горизонтальных и вертикальных линий, которые проходят через пользовательские формы. Координатная сетка выполняет две функции.

- **Служит визуальной подсказкой при расположении элементов управления в форме с помощью мыши**. Визуально координатная сетка представлена пунктирными линиями, которые видны на формах.
- **Автоматически привязывает элементы управления при изменении их положения или размеров**. Независимо от перемещений мыши, края элемента управления всегда привязаны к какой-либо из линий координатной сетки, что обеспечивает разумную последовательность при создании формы, хотя в какой-то мере и ограничивает гибкость.

Две эти функции используются независимо — можно отключить видимую координатную сетку и оставить включенной автоматическую привязку и наоборот.

Чтобы контролировать режим работы координатной сетки, выберите команду Tools⇨Options (Сервис⇨Параметры), в диалоговом окне Options (Параметры) перейдите на вкладку General (Общие), чтобы получить доступ к настройкам, представленным на рис. 56.8.

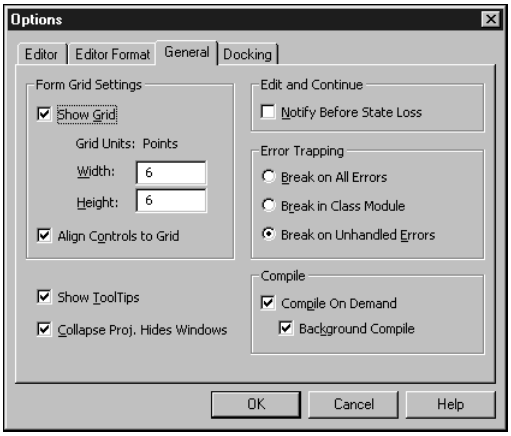


Рис. 56.8. Изменение настроек координатной сетки во вкладке General *диалогового окна* Options

Обратив внимание на левую верхнюю часть диалогового окна, можно увидеть насколько простых настроек сетки. В них входят следующие.

- **Show Grid** (Показать сетку). Снимите этот флажок, чтобы убрать с экрана визуальную сетку в виде пунктирных линий. Это действие никак не скажется на автоматической привязке к координатной сетке.
- **Width** (Ширина) и **Height** (Высота). Эти два поля позволяют изменять расстояние между горизонтальными и вертикальными линиями сетки независимо друг от друга. Параметр Width регулирует расстояние между вертикальными линиями, что, в свою очередь влияет на горизонтальные координаты элементов управления. Аналогично, Height служит для изменения расстояния между горизонтальными линиями координатной сетки и вертикальными координатами элементов управления.
- **Align Controls to Grid** (Привязать к сетке). Когда этот флажок установлен, функция привязки к координатной сетке включена. Сняв этот флажок можно абсолютно свободно располагать элементы управления на форме и изменять их размеры. Опять-таки, визуальная сетка может быть включена.

Форматирование элементов управления

К счастью, VBA предоставляет набор инструментов, которые помогают с легкостью достичь в форме симметрии, согласованности и порядка. Хотя создание хорошо организованной формы все еще требует определенного ручного труда, автоматические средства организации редактора Visual Basic могут выполнить большую часть работы.

Работа с меню Format

Меню Format (Формат) редактора Visual Basic (рис. 56.9) является центром управления командами, которые влияют на расположение элементов на форме. Близкое знакомство с этим меню и его многочисленными подменю могут сильно облегчить процесс создания форм.

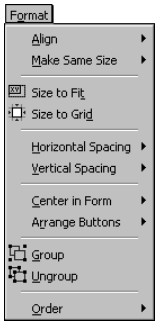


Рис. 56.9. Меню Format редактора Visual Editor

Панель инструментов UserForm

При работе в окне формы редактора Visual Basic панель инструментов UserForm (Пользовательская форма) пригодится как нельзя более кстати. Если она отсутствует на экране, ее можно вызвать, щелкнув правой кнопкой мыши в любом месте панели инструментов и выбрав в контекстном меню UserForm (рис. 56.10).

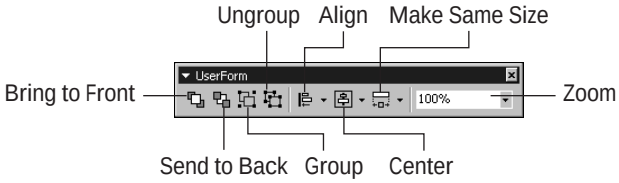


Рис. 56.10. Панель инструментов UserForm редактора Visual Basic

Большинство кнопок панели инструментов UserForm соответствуют командам меню Format. Некоторые из них являются многофункциональными кнопками и позволяют выполнять различные действия. Ели щелкнуть на основной части кнопки, VBA выполнит то же действие, что выполнялось в последний раз. Чтобы выбрать другое, щелкните на маленькой стрелке и выберите нужное действие из списка.

Группировка различных элементов управления

Хотя можно с легкостью выделить несколько элементов управления при помощи мыши, это не лучший выход, если приходится работать с ними как с единым целым. При объединении всех элементов управления в *группу* отпадает необходимость заново выделять одни и те же элементы управления каждый раз, когда необходимо произвести над ними какие-либо действия; кроме того, устраняется возможность ошибки при выделении.

Создание группы является простым действием. Выберите все элементы управления, которые нужно сгруппировать, и затем щелкните на кнопке Group (Группировка) на панели инструментов UserForm или выберите команду Format⇨Group (Формат⇨ Группировка).

Группировка позволяет применять команды форматирования не только к отдельным объектам, но и к целым группам. Если, например, необходимо сделать равное расстояние между тремя рядами кнопок, можно сгруппировать каждый ряд, выделить все три группы и использовать команду Horizontal Spacing⇨Make Equal (Горизонтальный интервал⇨Сделать равным), о чем рассказано ниже.

Расположение элементов управления друг над другом

Хотя лучше избегать наложения элементов управления, иногда используется именно этот подход (например, в случае, когда требуется изменить содержание формы во время работы программы). Программа может выводить или убирать элементы управления с экрана, изменяя значение свойства `Visible` с `True` на `False` и наоборот. В редакторе Visual Basic, однако, все элементы управления будут видимы, до тех пор, пока их не спрятать под другими элементами управления формы. Чтобы изменить порядок расположения элементов, можно использовать команды Order (Порядок) меню Format.

Ниже приведены рекомендации по использованию команд Order.

- Если хотя бы небольшая часть закрытого элемента управления видна, щелкните на ней с целью выделить элемент управления и затем выберите Format⇨Order⇨Bring to Front (Формат⇨Порядок⇨На передний план), чтобы расположить элемент управления поверх других.
- Если нужный элемент управления полностью скрыт под верхними, выберите верхний и затем отправьте его на задний план при помощи команды Format⇨Order⇨Send to Back (Формат⇨Порядок⇨На задний план). Необходимо повторять это действие, пока не появится нужный элемент управления.
- Если требуется упорядочить элементы управления, которые перекрываются во время выполнения программы, используйте команды Bring Forward (Переместить вперед) или Send Backward (Переместить назад). Эти команды сдвигают элемент на одну позицию вперед или назад.

Организация составных элементов управления

Многие из наиболее сложных команд меню Format работают только с составными элементами управления или с выделениями, которые включают в себя несколько групп. В данном разделе описываются такие команды, но сначала — информация о том, какое место занимают разные элементы управления.

Выбор основного элемента управления

Во время выполнения определенных команд, связанных с форматированием составных элементов управления, один из элементов служит ориентиром. В VBA он также называется *основным элементом управления* (dominant control). Например, если используется команда Format⇨Make Same Size (Формат⇨Установить равный

размер), для придания нескольким объектам одинакового размера VBA копирует выбранные параметры (высота, ширина или и то, и другое) основного элемента и переносит их на весь набор элементов. Аналогично действует команда Align (Выравнивание): все элементы выстраиваются по одной линии с основным элементом управления, который не изменяет своего положения. Результат действия команд Horizontal Spacing (Горизонтальный интервал) и Vertical Spacing (Вертикальный интервал) также зависит от того, какой элемент управления выбран основным.

На рис. 56.11 только один из выделенных элементов управления окружен белыми метками. Это и есть основной элемент управления (на что указывают белые метки). Все остальные элементы управления окружены черными метками.

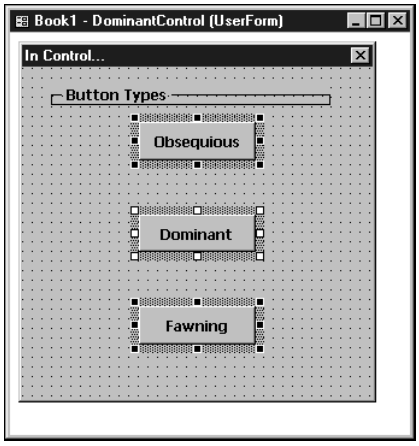


Рис. 56.11. Элемент управления, окруженный белыми метками, является основным

Ниже описывается, как назначить основной элемент управления при выделении группы.

Чтобы...	*Техника выделения элементов управления*
Выбрать основной элемент управления с помощью рамки выделения	Нужно начать перетаскивание выделения, чтобы указатель мыши был как можно ближе к нужному элементу управления
Сделать первый выбранный элемент управления основным	<Shift+щелчок>, чтобы выбрать каждый элемент управления в наборе
Сделать последний выбранный элемент управления основным	<Ctrl+щелчок>, чтобы выбрать каждый элемент управления в наборе
Выбрать другой основной элемент в имеющемся выделении	<Ctrl+двойной щелчок> на выбранном элементе управления

Выравнивание элементов управления относительно друг друга

Даже при работе с включенной координатной сеткой нередко бывает, что элементы управления, которые должны выстраиваться в четкий ряд, находятся не там, где должны. Вместо того, чтобы мучительно равнять их вручную, можно воспользоваться командами Format⇨Align (Формат⇨Выравнивание) или эквивалентными кнопками.

Можно выровнять элементы по центру или по любой из имеющихся сторон как по горизонтали, так и по вертикали.

Придание элементам управления одинакового размера

Три команды из группы Make Same Size (Установить равный размер) автоматически приводят размеры всех элементов управления к размерам основного элемента. Можно выровнять элементы по высоте, ширине или по обоим параметрам.

Выравнивание горизонтальных и вертикальных интервалов

Команды Horizontal Spacing и Vertical Spacing используются для изменения расстояния между двумя или более элементами управления четырьмя различными способами. Три из них чаще употребляются (хотя четвертый тоже доступен), если выделение включает в себя, по меньшей мере, три элемента управления. Ниже приведен список команд.

- **Make Equal** (Установить равными). Выравнивает расстояние тремя и более выбранными элементами управления. Крайние элементы управления остаются на месте, промежуточные передвигаются. Эта команда не работает, если выбраны только два элемента управления.
- **Increase** (Увеличить) и **Decrease** (Уменьшить). Увеличивают или уменьшают расстояние между выделенными элементами управления на одну ячейку сетки. Основной элемент управления остается на месте, остальные сдвигаются.
- **Remove** (Удалить). Сдвигает элементы управления так, что между ними не остается промежутков. Основной элемент управления остается на месте.

Другие параметры форматирования

По большей части, другие опции форматирования в меню Format предназначены для работы как с отдельными элементами управления, так и с группами. После выделения одного или нескольких элементов управления можно выполнить следующие действия.

- **Center in Form** (Расположить по центру формы). Центрирует элементы в форме как по горизонтали, так и по вертикали. Если выбрано два или более элементов управления, команда перенесет их все на осевую линию формы, а не будет рассматривать выделение как отдельный элемент (если нужен другой результат, можно попробовать предварительно сгруппировать элементы).
- **Arrange Buttons** (Упорядочить кнопки). Располагает одну или несколько управляющих кнопок внизу или в правом углу формы. Хотя команда Arrange Buttons работает, когда выделено несколько элементов управления, сдвигаются только командные кнопки.
- **Size to Fit** (Автоподбор). Предписывает VBA изменить размер элемента(ов) управления таким образом, чтобы был полностью показан содержащийся в них текст. Эта команда единоразовая (она не устанавливает значение свойства `AutoSize` равным `True`); размер остается постоянным, даже если впоследствии в текст будут внесены изменения.
- **Size to Grid** (По линиям сетки). Передвигает стороны выбранного элемента(ов) управления к ближайшей линии координатной сетки.

В редакторе Visual Basic отсутствуют команды для равномерного расположения элементов управления по горизонтали или вертикали. Однако это можно легко сделать следующим образом.

1. Визуально расположите элементы управления так, как хотелось бы.
2. Выделите элементы управления, затем используйте команду Align Tops или любую другую, производящую горизонтальное выравнивание.
3. Не снимая выделения с элементов управления, установите равные расстояния между ними командой Horizontal Spacing Make Equal.
4. Сгруппируйте выделенные элементы (командой Group).
5. Отцентрируйте элементы управления в форме, используя команду Center in Form Horizontally.

Аналогично, применив команды выравнивания по вертикали в пп. 2, 3 и 5, можно распределить элементы управления по вертикали.

Работа с элементами управления

Элементы управления, включенные в VBA, предоставляют все необходимые средства для построения диалоговых окон, которые выглядят и действуют как профессиональное программное обеспечение. В этом разделе речь пойдет о свойствах, общих для многих элементов управления (но недоступных для форм), а затем большинство элементов управления будет рассмотрено отдельно. Помните: чтобы заставить элемент управления выполнить нужное действие, необходимо написать код процедуры обработки события (см. раздел “Программирование форм” далее в этой главе).

Установка порядка перехода между элементами управления

В Windows существует соглашение, согласно которому нажатие клавиши <Tab> переводит фокус с одного элемента управления диалогового окна на другой. По умолчанию каждый элемент управления, добавляемый в форму VBA, занимает свое место в *порядке перехода*, последовательности, в которой выбираются элементы управления при нажатии <Tab>. (Между прочим, <Shift+Tab> проходит порядок перехода в обратном направлении.)

Чтобы проверить порядок перехода, не нужно запускать форму. Нажатие клавиши <Tab> в окне UserForm произведет переход между элементами управления в том же порядке. Вначале порядок перехода основывается на последовательности, в которой элементы управления добавлялись в форму. Но, как правило, элементы управления добавляются по мере необходимости, а не в порядке перехода. Возьмите в свои руки управление порядком перехода и, при желании, удалите некоторые элементы из этой последовательности.

Наиболее легкий способ изменить порядок перехода в форме — с помощью диалогового окна View⇨Tab Order (Вид⇨Порядок перехода). Для перегруппировки перечисленных в нем элементов щелкните на элементе управления, который нужно передвинуть, а затем — на кнопке Move Up (Вверх) или Move Down (Вниз). На самом деле, за порядок перехода отвечает свойство `TabInsdex` каждого элемента. У первого элемента последовательности значение `TabInsdex` равно 0, у второго — 1 и т.д. При изменении значения `TabIndex` VBA автоматически нумерует остальные. Чтобы удалить элемент управления из порядка перехода, установите его свойство `TabStop` равным `False`. Это действие не изменит его место в порядке перехода, поэтому, если вновь установить свойство `TabStop` равным `True`, элемент управления будет помещен на то же место в порядке перехода, которое он занимал ранее.

Назначение ускоряющих клавиш

Хотя многие пользователи выбирают элементы управления при помощи мыши, некоторые предпочитают работать с клавиатурой. Следует предоставить таким пользователям клавиши быстрого доступа, или *ускоряющие клавиши*, для каждого элемента управления. При работе формы нажатие клавиши <Alt> с ускоряющей клавишей переведет фокус на соответствующий элемент управления и может вызвать событие (например `Click`).

Чтобы создать связь, введите символ в поле `Accelerator` диалогового окна Properties. Символ должен входить в заголовок элемента управления и не должен повторять ускоряющую клавишу иного элемента управления в той же форме. VBA автоматически подчеркнет символ ускоряющей клавиши в заголовке.

Чтобы добавить ускоряющую клавишу для элемента управления, у которого нет свойства `Caption`, т. е. заголовка, (например, для текстового окна или полосы прокрутки), выполните следующие шаги.

1. Создайте такой элемент управления, как надпись (описан в следующем разделе).
2. Настройте порядок перехода таким образом, чтобы надпись находилась точно перед элементом управления.
3. Свяжите ускоряющую клавишу с надписью.

Секреты элементов управления

Каждый тип элемента управления VBA разработан для различных целей и каждый требует своего подхода со стороны пользователя.

Отправка сообщений с помощью меток

Элемент управления *надпись* — это прямоугольная область на форме, в которой можно отображать сообщения. С точки зрения пользователя, надпись не является в полном смысле элементом управления — она не позволяет чем-либо управлять. Надпись лишь отображает какой-либо текст или, возможно, рисунок. Вместо текста нельзя вводить свою информацию, его даже нельзя скопировать в буфер обмена.

Однако надписи очень важны с точки зрения программиста, поскольку позволяют отправлять сообщения пользователям. Как правило, надписи используются для обозначения элементов управления и их функций (рис. 56.12), что особенно полезно для таких элементов управления, у которых нет собственного заголовка (полосы прокрутки, счетчики и т.п.).

Текст, отображаемый в надписи, является ее заголовком. Поэтому, чтобы разместить в метке собственный текст, нужно изменить его в свойстве `Caption` окна Properties.

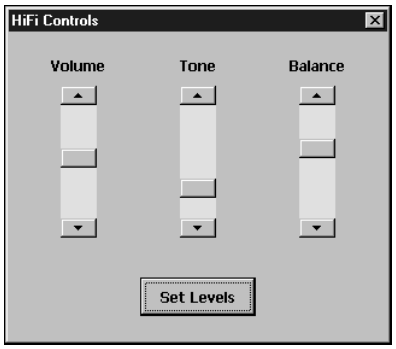


Рис. 56.12. Три полосы прокрутки, каждая с сопутствующей надписью

Автоматическая настройка надписей

При помощи опций в окне Properties надписи можно настроить так, чтобы они сами подстраивались под содержащийся текст.

- Оставьте свойство `WordWrap` равным `True` (по умолчанию), если необходимо, чтобы редактор VBA автоматически разбивал текст на строки, с целью размещения его в доступном пространстве, как это делает текстовый процессор. Если установить свойство `WordWrap` равным `False`, весь текст надписи останется в одной строке, даже если в области элемента управления недостаточно места.
- Установите свойство `AutoSize` равным `True`, если необходимо, чтобы размер надписи изменялся в зависимости от введенного текста. Если свойство `WordWrap` также установлено равным `True`, надпись будет расширяться по вертикали. Если же значение `WordWrap` равно `False`, надпись расширится так, чтобы поместить единственную строку текста.
- Используйте свойство `TextAlign` для управления выравниванием текста в надписи: по левому краю, по центру по правому краю.

Изменение текста надписи в коде

Хотя во время работы программы пользователь изменить текст надписи не может, программисту такое задание по плечу. Используя свойство `Caption`, достаточно добавить одну строку кода, чтобы изменить текст в соответствии с текущими требованиями программы:

```
lblInspirationalMessage.Caption = "Ищите и обрящете!"
```

Текстовые поля позволяют общаться с пользователем

Элемент управления *текстовое поле* (рис. 56.13) используется при необходимости получения информации от пользователя. Можно создать код для обработки данных, вводимых пользователем.

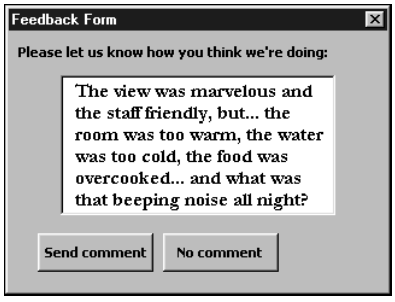


Рис. 56.13. Текстовые поля предоставляют возможность общаться с программой

Текстовые поля используются вместо окон ввода данных (`InputBox`), если:

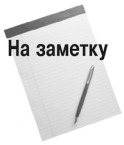
- требуется создать более привлекательное диалоговое окно, чем предоставляемое стандартной функцией `InputBox`;
- в диалоговом окне нужно расположить, по меньшей мере, еще один элемент управления, помимо текстового поля;
- необходимо проверить вводимые данные во время ввода. При помощи текстового поля можно запрограммировать процедуру события, которая будет вызываться при каждом нажатии клавиши и проверять, соответствует ли нажатая клавиша предъявляемым критериям. Используя `InputBox`, можно проверить всю введенную строку только после закрытия окна.

Использование текстовых полей в качестве надписей

Как и надписи, текстовые поля могут отображать сообщения для пользователя. В текстовом поле можно редактировать текст. Если вы хотите, чтобы пользователь не изменял содержимого текстового поля, установите значение свойства `Locked` равным `True`. Примите во внимание, что текст поля можно скопировать в буфер обмена, а текст надписи — нет. Можно предотвратить такой поворот событий, установив свойству `Enabled` значение `False`, но в этом случае текст будет появляться затемненным (см. раздел о свойствах `Locked` и `Enabled` в начале главы).

Содержание текстовых окон по умолчанию

Можно поместить в текстовое поле текст по умолчанию — пользователю не придется вводить свой вариант, если его устраивает существующий. Чтобы ввести текст по умолчанию, щелкните на текстовом поле, чтобы выбрать его, а затем щелкните еще раз для перехода в режим ввода текста (не используйте двойной щелчок). Можно ввести текст в поле `Value` в окне Properties.



У текстовых окон нет заголовков. Строка текста в текстовом поле — это его свойство `Value`.

Что такое пароль

Пароли используются для защиты важных данных от несанкционированного копирования или внесения изменений, с целью дать людям понять, что перед ними нечто важное, раз понадобилось защищать это паролем. Независимо от назначения пароля, для его ввода можно легко использовать текстовое поле.

Выберите текстовое поле в редакторе Visual Basic, определите значение свойства `PasswordChar` в окне Properties, указав символ, который будет отображаться в текстовом поле вместо вводимых значений. Текстовое поле будет воспринимать вводимую информацию, но никто не сможет ее увидеть, кроме, разумеется, программы.

Помните, запрос пароля сам по себе не защитит данные. Данные и список правильных паролей должны быть зашифрованы, и при вводе верного пароля программа должна их расшифровывать.

Обработка пользовательского ввода

Чтобы определить, что именно ввел пользователь, программа должна принять значение свойства `Value` текстового окна. Как правило, свойство `Value` присваивается строковой переменной оператором, как в следующем примере:

```
strTextBoxText = txtMessageFromUser.Value
```

Настройка текстового поля

У текстовых полей, как и у надписей, есть свойства `AutoSize`, `WordWrap` и `TextAlign`, и работают они практически так же (см. информацию об этих свойствах ранее в этой главе). Однако свойство `WordWrap` ничего работает только в многострочных текстовых полях. Читайте далее...

Многострочные текстовые поля

Текст, который переносится VBA, на самом деле хранится одной строкой. Многострочные текстовые поля дают пользователю возможность начать “настоящую” новую строку, нажав <Ctrl+Enter> или, если значение свойства `EnterKeyBehavior` установлено равным `True`, — просто <Enter>. Таким образом, разрыв строки может проходить не там, где он был бы установлен VBA. А управлять местоположением разрывов строк можно, используя код, приведенный в разделе о надписях ранее в этой главе.

Чтобы заставить текстовое окно правильно отображать текст, разбитый на строки, и уместить их в окно, установите свойство `MultiLine`

равным `True`. Иначе, даже если значение свойства `WordWrap` будет равно `True`, весь текст отобразится одной строкой, заходя за границы текстового окна.

Добавление полос прокрутки

Какими бы ни были значения свойств `MultiLine`, `WordWrap` и `EnterKeyBehavior`, текстовое окно отображает только определенное количество информации. Хотя пользователь, конечно, может переместить курсор при помощи клавиш управления с целью увидеть весь текст, лучше поместить полосы прокрутки во все текстовые поля при помощи свойства `ScrollBars`. VBA отображает полосы прокрутки только тогда, когда весь текст в поле не помещается.

Использование управляющих кнопок

Если необходимо что-либо сделать, причем срочно, ничто не создает такого чувства, будто все контролируешь, как щелчок на кнопке и немедленное получение обратной реакции. Обычная кнопка — это просто серое пятно с коротким пояснением, например Да, Отмена или "Подумай снова, Дружок". Если текст кажется слишком скучным, можно легко добавить небольшую пиктограмму. На рис. 56.14 показаны различные варианты управляющих кнопок.

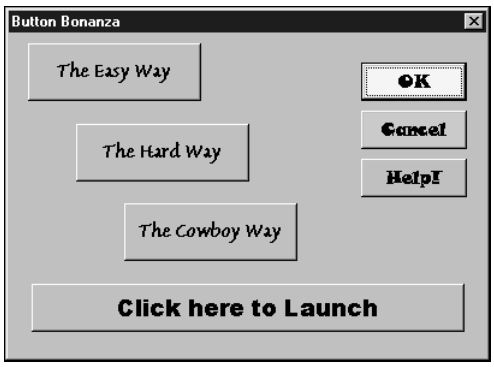


Рис. 56.14. Все многообразие управляющих кнопок

К сожалению, кнопки не *делают* ничего, пока их не запрограммируют. Щелчок на кнопке генерирует событие `Click`, но нужно написать код, чтобы сообщить VBA, какие действия необходимо предпринять.

Выбор кнопки по умолчанию

В большинстве диалоговых окон нажатие клавиши <Enter> запускает определенную кнопку. Она называется *кнопкой по умолчанию*, т.е. той, на которой находится фокус. Чтобы задать управляющую кнопку кнопкой по умолчанию, установите значение свойства `Default` равным `True`. Естественно, в каждой форме может быть только одна кнопка по умолчанию.

Создание кнопки Отмена

Если диалоговое окно может изменять данные или настройки программы, то необходимо предусмотреть возможность "дать задний ход", пока изменения не стали необратимыми. По общему соглашению, таким спасательным выходом является кнопка с надписью Отмена (Cancel). Если программист принадлежит к секте раскольников, он вполне может назвать кнопку "А, неважно" или "Забудем это", но из

соображений заботы о беспомощном пользователе следует создать пути отхода, как бы они ни назывались.

Также, по всеобщему соглашению, нажатие клавиши <Esc> приводит к закрытию диалогового окна, что равнозначно использованию кнопки Отмена. Установка свойства `Cancel` равным `True` подразумевает следующее: на нажатие клавиши <Esc> программа отреагирует, как на щелчок на этой кнопке. Последующие действия зависит уже от написанного кода.

Установка значения свойства `Cancel` равным `True` не означает, что по щелчку на кнопке диалоговое окно закроется. Просто клавиша <Esc> будет связана с событием `Click` данной кнопки.

Рамки объединяют другие элементы управления

Элементы управления *рамка* (или *фрейм*) являются скромными, но важными компонентами в наборе инструментов VBA. Визуально фрейм — простой прямоугольник с заголовком в верхней строке. Другие элементы управления помещаются во фрейм с целью объединения.

Фреймы служат таким целям:

- визуально отделять группу связанных между собой по какому-либо признаку элементов управления, показав их связь и упорядочив большую форму;
- функционально определять группу переключателей.

Подробности по использованию фреймов даны в разделе о переключателях далее в этой главе. Ниже приведены основы использования фреймов для организации всех типов элементов управления.

Расположение элементов управления во фреймах

После добавления фрейма в форму расположение любого последующего элемента управления во фрейме связывает элемент управления и фрейм. С этого момента при изменении положения фрейма изменит свое место и элемент управления, всегда находящийся в одном положении относительно фрейма.

Существуют два пути добавления элемента управления во фрейм.

- **Размещение во фрейме нового элемента управления**. Создайте, как обычно, новый элемент управления, щелкнув на соответствующей пиктограмме в панели инструментов и перетащив его в нужное место в форме (в данном случае оно находится во фрейме).
- **Размещение во фрейме существующего элемента управления**. Перетащите элемент управления при помощи мыши, пока указатель мыши не окажется в границах фрейма. Отпустите клавишу — и фрейм перехватит управление элементом.

После успешного размещения элемента управления во фрейме граница фрейма будет всегда выделяться при выделении содержащегося в нем элемента управления (рис. 56.15).

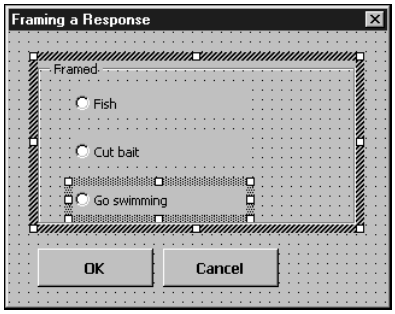


Рис. 56.15. Выбор кнопки во фрейме привел к выделению всего фрейма

Удаление элементов управления из фреймов

Разорвать связь между элементом управления и фреймом так же просто, как и перетащить элемент управления в другое место формы. Как только указатель мыши окажется за границами фрейма, отпустите кнопку мыши — связь будет разорвана, а элемент управления будет помещен в форму вне фрейма. Теперь два объекта (фрейм и элемент управления) независимы.

Многостраничные элементы управления

Многостраничный элемент управления воспроизводит вкладки диалоговых окон, часто используемые в коммерческом программном обеспечении под Windows. Каждая страница элемента управления действует как отдельный фрейм. Помещенные на страницу, элементы управления привязаны к странице и видимы только при ее отображении. Чтобы отобразить другую страницу, необходимо щелкнуть на соответствующей вкладке в верхней части формы.

Многостраничный элемент управления устанавливается так же, как и любой другой. Между прочим, VBA содержит подобный элемент управления — так называемую *полосу закладок* (tab strip). Полосы закладок полезны, когда необходимо, чтобы одни и те же элементы управления содержали различную информацию в зависимости от выбранной закладки. Они требуют от программиста большей работы, чем многостраничные элементы управления, следовательно, рассматриваются вне рамок данной книги.

Работа со свойствами многостраничных элементов управления и с их страницами

Многостраничные элементы управления, как и отдельные страницы, обладают свойствами. Перед изменением свойства убедитесь, что это *нужное* свойство. После помещения многостраничного элемента управления в форму он становится выделенным, а его свойства появляются в окне Properties. Щелчок на многостраничном элементе управления выбирает одну из страниц, а не весь элемент. Можно выбрать другую страницу, щелкнув на соответствующей вкладке. Чтобы выбрать элемент управления целиком, либо щелкните на границе элемента управления, либо используйте раскрывающийся список в верхней части окна Properties.

Добавление элементов управления на страницу

Чтобы добавить кнопки, фреймы и другие элементы управления на страницу многостраничного элемента управления, просто перенесите их на нужную страницу, как во фрейм. Однако сначала выберите необходимую страницу, щелкнув на нужной вкладке (вкладки работают, даже если форма не запущена).

Добавление и удаление страниц

Чтобы добавить в многостраничный элемент управления еще одну страницу, щелкните правой кнопкой мыши на полосе вкладок и выберите в контекстном меню команду New Page (Создать страницу). Удалить существующую страницу еще проще, но будьте внимательны, не удалите по ошибке нужную страницу! Редактор Visual Basic удаляет страницы немедленно, без запроса на подтверждение, причем команда Undo (Отменить) не сработает.

Изменение текста на вкладках для отдельной страницы

Чтобы изменить название, отображаемое на вкладке, измените ее заголовок. Переименуйте вкладку, поместив новый текст в поле `Caption` в окне Properties. Либо щелкните правой кнопкой мыши на вкладке элемента управления и выберите команду Rename (Переименовать) из контекстного меню и введите новый заголовок в диалоговом окне Rename.

Не следует пренебрегать словом “переименовать” — на самом деле изменяется заголовок страницы, а не ее имя (чтобы действительно переименовать страницу, нужно воспользоваться окном свойств Property). Диалоговое окно Rename дает возможность пользователю выбрать ускоряющую клавишу и ввести текст подсказки; обе опции можно установить в окне Properties.

Изменение порядка страниц

Чтобы изменить порядок, в котором отображаются многостраничные элементы управления, выберите команду Move (Переместить) из контекстного меню, появляющегося при щелчке правой кнопкой на ярлыке вкладки. В появившемся диалоговом окне щелкните на странице, которую следует передвинуть, а затем, используя кнопки Move Up и Move Down, измените ее положение в ряду.

Выбор одного элемента из группы переключателей

Часто как в жизни, так и в программном обеспечении можно выбрать только что-то одно из множества. Заказывая мороженное, можно выбрать для него сироп, шоколад или ром, но не все сразу. При покупке брюк вы выбираете только один размер из всех возможных.

В Windows наиболее частым способом представить взаимоисключающие элементы при выборе является набор *переключателей* — небольших круглых кнопок, названных так по причине подобия кнопкам автомагнитол. Как-никак, а можно одновременно слушать только одну радиостанцию (клинические случаи не рассматриваются). Обычный набор *переключателей* приведен на рис. 56.16.

Рис. 56.16. Переключатели

Переключатели должны находиться в наборе, поскольку выбор одного из них исключает выбор других. Включение одного переключателя автоматически выключает все остальные.

Группировка переключателей

Не следует беспокоиться о том, как создать группу кнопок выбора вариантов. Достаточно расположить их в одном месте в форме. VBA автоматически обращается с ними как с группой, и во время выполнения программы их поведение адекватно — когда один включен, остальные выключены.

Но что подразумевается под “одним местом в форме”? Идея состоит в следующем. Одна часть формы — это сама форма, ее полотно, если хотите. Каждый фрейм, который создает пользователь, добавляет еще одну часть. И каждая страница в многостраничном элементе управления тоже является отдельной частью. Можно даже помещать фреймы друг в друга или в страницы многостраничного элемента управления — в этом случае каждый фрейм тоже будет являться частью.

В форме, содержащей один или несколько фреймов, VBA обращается с переключателями, которые не входят во фрейм, и кнопками каждого фрейма как с отдельными группами. При желании, свойство `GroupName` позволяет определить несколько групп переключателей в одной части формы — достаточно указать одинаковое значение `GroupName` для всех переключателей группы.

Какой из переключателей выбран

Щелчок на переключателе выбирает его, но, как правило, не вызывает других быстрых изменений. Диалоговое окно остается прежним, давая пользователю возможность подумать и, возможно, изменить выбор. Только по щелчку на кнопке OK выбор пользователя принимается.

Для программиста задача заключается в следующем: определить, какая кнопка была выбрана в момент события подтверждения формы. Проверьте свойство `Value` каждой кнопки в группе. Хотя существуют и более хитрые способы — оператор `If...ElseIf` является вполне подходящим решением:

```
If OptionButton1.Value = True Then
    ChosenOption = "Bill"
ElseIf OptionButton2.Value = True Then
    ChosenOption = "Bob"
```

```
ElseIf OptionButton3.Value = True Then
    ChosenOption = "Barney"
Else
    ChosenOption = ""
```

Включение и выключение опций при помощи флажков и выключателей

Переключатели удобны при работе с многими взаимоисключающими вариантами. Однако когда выбор ограничен двумя вариантами, лучше использовать флажок или переключатель. *Флажки* и *переключатели* подходят для ситуаций альтернативного выбора: `Да` или `Нет`, `Включен` или `Выключен`, `Правда` или `Ложь`. На практике вся разница между флажками и переключателями заключается в том, как они выглядят.

Группы флажков

Флажки часто группируют, чтобы предоставить пользователю список вариантов выбора, не исключающих друг друга. Это все равно, как если бы человек пошел в магазин за пивом и принес бы домой упаковку шоколада, пару коробок конфет и пакет зефира. На рис. 56.17 приведены некоторые дополнительные примеры. Обратите внимание: каждый флажок представляет собой выбор типа “да/нет” для отдельного элемента.

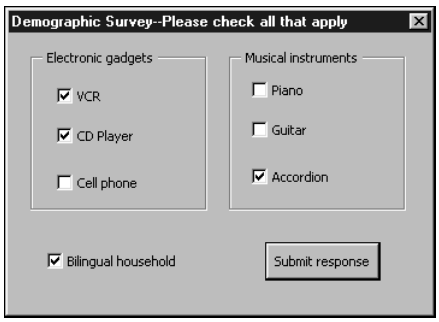


Рис. 56.17. Группы флажков позволяют выбрать несколько элементов для одного варианта

С целью поберечь и без того расшатанную психику пользователя размещайте группу флажков где-нибудь в одном месте формы так, чтобы они были визуально связаны. Для разграничения групп используйте фреймы.

Поскольку каждый флажок работает независимо от других, беспокоится об их группировке не нужно. Однако, при желании, можно определить их как членов группы, указав одинаковое значение в поле свойства `GroupName`. Это помогает отслеживать, какие флажки чему принадлежат, если форма каким-либо образом изменяется.

Как проверить, установлен ли флажок или включен выключатель

Как правило, свойство `Value` содержит основную информацию о взаимодействии пользователя с элементами управления. Если флажок установлен, его свойство `Value` равно `True`; если флажок снят — `False`. То же и с выключателями: `Value` равно `True`, если выключатель включен, и `False` — в противном случае. Приведенный далее код возвращает и обрабатывает значение:

```
If tglLightSwitch.Value = True
    TurnLightsOn
Else
    TurnLightsOff
End If
```

Выбор опций при помощи списка и поля со списком

Если установить более 4–5 взаимоисключающих вариантов выбора для одного элемента и представить их в виде переключателей, форма станет напоминать забитый склад. Если варианты не взаимоисключающие, можно использовать 10–12 флажков, ведь пользователь будет считать каждый флажок личной заслугой. Тем не менее, лес флажков — это не совсем нормально.

Знаете, что такое списки? Не торопитесь их использовать

Для работы со списками предпочтительнее использовать поля со списками, независимо от того, разрешено пользователю вводить данные в поле или нет. Забудьте о списках.

И вот почему: окно списка VBA не может отображать элементы в виде раскрывающегося списка и представляет собой просто прямоугольный участок в форме, со списком возможных вариантов. Таким образом, использование списка не решает проблемы экономии места в форме — если список содержит большое число элементов, он занимает много места и отвлекает внимание, даже когда не используется. Если вариантов выбора немного, для создания удобного списка вполне можно обойтись переключателями или флажками, которые и выглядят посимпатичнее.

В отличие от окон списка, поле со списком гораздо компактнее, так как отображает только один вариант из имеющихся в списке. При необходимости поле со списком может действовать как окно списка, не воспринимая пользовательского ввода. Нужно лишь изменить значение свойства `Style` на 2 (`fmStyleDropDownList`). Поэтому не стоит утруждать себя окнами списка.

Список

Список — решение подобных проблем в Windows. Окно списка представляет собой компактный набор именованных опций, из которого нужно выбрать одну. Примером могут служить списки штатов Америки или заправок для салата.

Элемент управления список в VBA позволяет относительно просто создавать окна списка, хотя для внесения данных в список необходимо уметь программировать. Список является альтернативой фрейма с группой переключателей, давая возможность пользователю выбрать только один из приведенных элементов. Кроме того, окно списка может служить в качестве набора флажков, предоставляя пользователю выбор нужного количества элементов. Список не может принимать значения, не находящиеся в списке, а также его нельзя представить в виде раскрывающегося списка. Чтобы обойти эти ограничения, лучше использовать поле со списком.

Поле со списком

Поле со списком сочетает в себе достоинства списка с достоинствами текстового поля: можно выбрать элемент из списка, а при отсутствии подходящего ввести его.

С точки зрения пользователя, поля со списками предпочтительнее, они дают возможность самовыражения. Однако, с точки зрения программиста, многие ситуации требуют ограничения выбора пользователя, чтобы избежать получения ошибочных данных. Например, в США 50 штатов, и бессмысленно разрешать пользователям вводить новые штаты в адресные базы данных.

Внесение данных в список или в поле со списком

Перейдем к сложному. Для ввода элементов в список или в поле со списком воспользоваться окном свойств Propertie не удастся. Придется писать код для метода элемента управления `AddItem` или связывать элемент управления с *источником данных*, т.е. со списком из рабочего листа Excel или базы данных Access.

Создание списка напрямую в коде требует написания процедуры для события формы `Activate`. Она должна содержать несколько операторов, подобных следующим:

```
Private Sub UserForm_Activate()
    cmbOpinionPoll.AddItem "Перенаселение"
    cmbOpinionPoll.AddItem "Глобальное потепление"
    cmbOpinionPoll.AddItem "Некогда нюхать розы"
    cmbOpinionPoll.AddItem "Нет роз чтобы понюхать"
    cmbOpinionPoll.AddItem "Налоги на богатство чересчур
высоки"
    cmbOpinionPoll.AddItem "Слишком много социальных служб"
    cmbOpinionPoll.AddItem "Несоответствующие социальные
службы"
    cmbOpinionPoll.AddItem "КЕЗы"
End Sub
```

О связывании списков или полей со списками с источниками данных речь идет в главе 50 (о формах Access).

Возврат опций, выбранных пользователем

Чтобы получить элемент, который выбрал пользователь, используйте в коде свойство `Value`. Принцип тот же, что и при работе с текстовым полем. Просто свяжите свойство с подходящей переменной (строкой, числовой переменной или вариантом), например:

```
strOpinion = cmbOpinionPoll.Value
```

Выбор значений при помощи полос прокрутки и счетчика

Когда человек регулирует громкость музыки или поворачивает термостат нагревателя, для выбора значений из доступного диапазона он пользуется обычными элементами управления. Такой вариант выбора эмулируется в VBA двумя элементами управления — *полосами прокрутки* и *кнопками счетчика* (рис. 56.18).

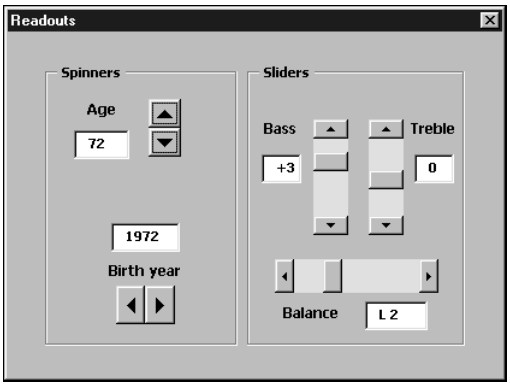


Рис. 56.18. Примеры полос прокрутки и кнопок счетчика

Конечно же, полосы прокрутки по большей части используются в Windows для пролистывания видимой части документа или диалогового окна, если вся информация не помещается одновременно. Но можно думать о полосе прокрутки и в общем плане, как о ползунке, который проходит через диапазон числовых значений. Чтобы выбрать значение, пользователь может воспользоваться ползунком, этакой прямоугольной штукой на полосе прокрутки, или щелкнуть на стрелке с краю.

Кнопки счетчика — те же полосы прокрутки, только без ползунков — остались только стрелки. Кнопки счетчика не считают ничего, кроме связанных с ними чисел.

Создание элементов управления

Полосы прокрутки и кнопки счетчиков располагаются в форме так же, как и другие элементы управления: щелкните на нужной пиктограмме в панели элементов Toolbox и перетащите в форму. Полосы прокрутки и кнопки счетчика можно располагать и горизонтально (со стрелками, направленными вправо и влево) и вертикально (вверх и вниз). По умолчанию VBA определяет расположение автоматически, основываясь на местоположении элемента в форме. Для настройки ориентации используйте свойство `Orienation`.

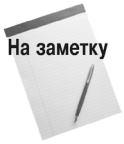
Настройка элементов управления

Если полоса прокрутки или кнопки счетчика расположены в форме, для обеспечения надлежащего уровня работы элементов управления выполните две задачи (в следующем порядке).

1. Определите диапазон величин, которые можно выбрать при помощи этих элементов управления.
2. Обеспечьте зрительное сопровождение, чтобы пользователь знал, что выбирает.

Выбор диапазона значений

Чтобы определить диапазон значений для полосы прокрутки или кнопок счетчика, используйте свойства `Max` и `Min`; допускаются только целые значения.



Несмотря на названия, эти свойства относятся к положениям на элементах управления, а не к числовым минимуму или максимуму. Другими словами, свойство `Max` может принимать меньшее значение, чем свойство `Min`. Свойство `Max` задает значение, которое принимает элемент управления по щелчку на верхней стрелке в вертикальном расположении или правой — в горизонтальной. `Min` отвечает за нижнюю или левую стрелку. Установка свойства `Min` меньшим, чем `Max`, изменяет полярность элемента управления, т.е. щелчок на стрелке “вниз” увеличивает значение.

Обратная связь

Полосами прокрутки и кнопками счетчика пользователь вряд ли воспользуется, если он не знает выбранного значения. К сожалению, элементы управления не оснащены шкалой по умолчанию. Придется писать код, чтобы соединить выбранное при помощи полосы прокрутки или кнопок счетчика значение с текстом, выводимым другим элементом управления. Это несложно и может быть выполнено одной строкой кода. Следующий код задает значение полосы прокрутки заголовку надписи при двойном щелчке на полосе прокрутки:

```
Private Sub sclWarpFactor_DblClick
    lblScrollBarReadout.Caption = sclWarpFactor.Value
End Sub
```

Обратите внимание: чтобы обеспечить немедленную обратную связь, следует разместить код (отвечающий за вывод выбранного значения) в процедуре обработки события `Click` полосы прокрутки или кнопок счетчика, как в примере (`sclWarpFactor` — название элемента управления полосы прокрутки). В следующем разделе подробно обсуждается программирование событий.

Программирование форм

Добавить элементы управления в форму просто, но чтобы заставить их выполнять нужные действия, придется приложить некоторые усилия и потрудиться над программой. Этот раздел поможет разобраться со всеми сложностями.

Загрузка и показ форм

После того, как было принято решение включить стандартные формы в программу VBA, первой и наиболее фундаментальной проблемой программирования стал вопрос вывода формы на экран. Поскольку программа VBA принадлежит к пользовательскому интерфейсу приложения, она не отражает форму сразу после запуска. В этом отношении VBA отличается от своего близкого родственника Visual Basic, в котором программа является формой, пока не приняты какие-либо чрезвычайные меры. Во всяком случае, чтобы в программе VBA отобразить форму, сделав ее доступной пользователю, необходимо добавить код.

Процесс отображения формы VBA состоит из двух этапов: загрузки формы в память и вывода ее на экран. Для выполнения обоих этапов можно использовать один оператор VBA. Однако (об этом пойдет речь в следующих разделах) иногда бывает полезным эти этапы разделять.

Вывод окон

Чтобы вывести форму на экран, используйте ее метод `Show`. Если, например, форма называется `FormICa`, достаточно набрать следующую строку:

```
FormICa.Show
```

Заметим, что `Show` — метод объекта UserForm, и его необходимо присоединить к имени формы после точки. В примере форма не загружена в память, метод `Show` загружает ее и затем выводит на экран. Если форма загружена, но скрыта, метод `Show` просто делает ее видимой.

Загрузка формы без ее отображения

Для загрузки формы в память без ее отображения на экране используется оператор `Load`. `Load`, который не является методом, поэтому его синтаксис, в отличие от `Show`, будет следующим:

```
Load FormA1DeHyde
```

Зачем загружать форму, не отображая ее? Например, если программа содержит многочисленные и сложные формы. Такой подход поможет быстрее выводить их на экран. Так как программа любой степени сложности выполняет множество различных операций по инициализации (чтение из файлов, просчет переменных и создание объектов), при запуске обычно приходится некоторое время ждать. Если в это же время загрузить формы, пользователи не обратят внимания на задержку, как при последующей загрузке форм.

Внесение изменений в форму перед ее выводом на экран

Можно также загрузить форму без использования оператора Load: с помощью операторов, работающих со свойствами или методами формы, либо с помощью одного из элементов управления. Такой подход позволяет программе внести изменения в форму перед выводом ее на экран. Программист не может предвидеть, как будут себя вести форма или элементы управления, пока не запустит программу.

В качестве простого примера предположим, что необходимо в заголовок формы внести дату и время. Поскольку время запуска программы предсказать невозможно, а запустить программу могут неоднократно, следует сделать так, чтобы VBA выдавал текущие дату и время. Это позволит сделать следующий простой пример:

```
Sub DisplayDateCaptionedForm ()
    DateCaptionedForm.Caption = Now
    DateCaptionedForm.Show
End Sub
```

Используя подобный подход, можно обеспечить наличие в надписи или текстовом поле отображения всей выделенной в данный момент в приложении VBA информации. На рис. 56.19 показано, что происходит при запуске соответствующей процедуры, объединенной с соответствующей формой в PowerPoint:

```
Sub DisplayShowSelectionForm ()
    Dim ItemCount As Integer, Message As String
    Items = ActiveWindow.Selection.ShapeRange.Count
    Message = CStr(Items) & " объектов выбрано."
    ShowSelectionForm.lblCountOfItems.Caption = Message
    ShowSelectionForm.Show
End Sub
```

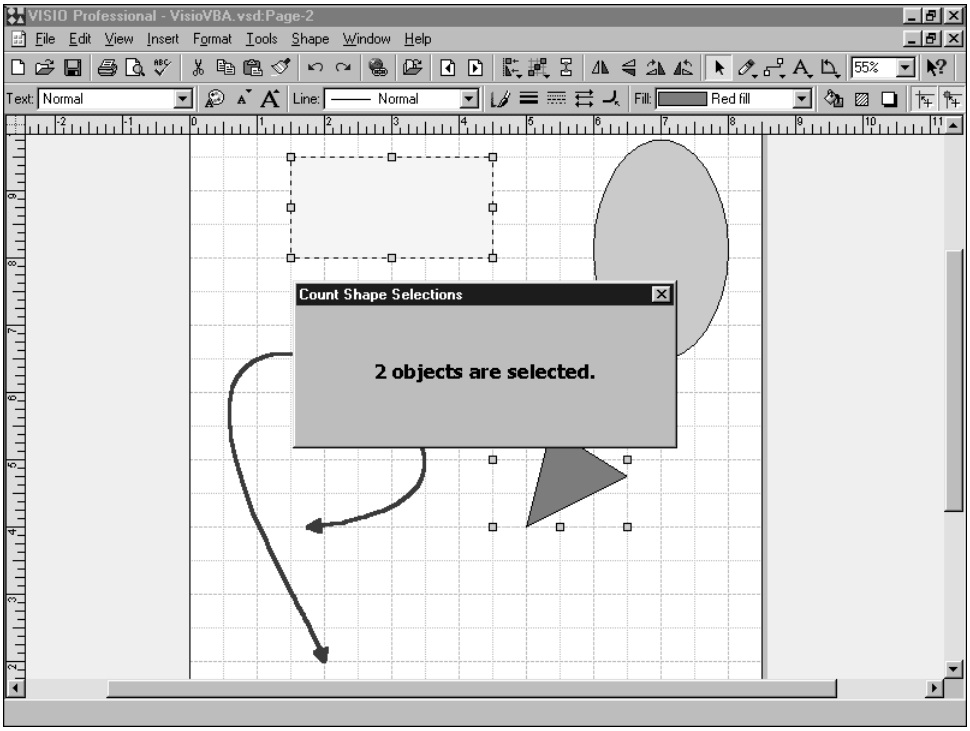


Рис. 56.19. Эта программа перед отображением формы определяет, что должно находиться в надписи

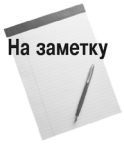
Изменение формы при помощи событий Initialize и Activate

Другим способом изменения формы перед выводом на экран является использование ее событий — `Initialize` или `Activate`. Код, помещенный в процедуры обработки этих событий, автоматически выполняется при возникновении события. Событие `Initialize` используется для помещения кода, который выполняется, когда VBA первый раз загружает форму, а `Activate` — для кода, который должен выполняться раз при каждом выводе формы на экран (включая первый). Создание кода для процедур обработки событий обсуждается в разделе "Программирование событий для форм и элементов управления" далее в этой главе.

Модальные и немодальные формы

Классическим примером *модальной формы* является обычное диалоговое окно: когда диалоговое окно активно, работать с другой частью программы нельзя. Сначала нужно закрыть диалоговое окно, чтобы получить возможность работать с другим диалоговым окном или напрямую с документом. Когда модальная форма активна, единственными процедурами, которые может выполнять программа VBA являются процедуры, принадлежащие самой форме. Форма может реагировать на события, но остальная часть программы приостанавливает свою работу и приложение (к которому относится программа) за пределами досягаемости пользователя до тех пор, пока форму не закроют. Например, при закрытии формы можно выдать запрос о сохранении изменений. Модальную форму используют для отображения такого сообщения. Данный тип формы заставит пользователя ответить, щелкнув либо на кнопке Да, либо на кнопке Нет.

Версия VBA, поставляемая с пакетом Office XP, позволяет выбирать между модальным и немодальным поведением каждого диалогового окна. По умолчанию свойство формы `ShowModal` равно `True`, что означает модальность формы. Измените его на `False`, если необходимо дать возможность пользователю переключаться между открытыми формами и если нужно, и чтобы программа продолжала выполняться, пока форма видима.



Перемещаться между открытыми формами не удастся, пока все они не станут немодальными. При открытии одной модальной формы остальная часть программы прекращает свое выполнение до тех пор, пока эту форму не закроют. Если программа выводит на экран формы перед появлением модальной формы, они будут видимы на экране, но недоступны.

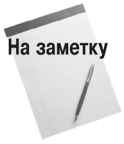
Ссылки на формы при помощи переменных

Хотя код может ссылаться на объект формы прямо по имени, можно воспользоваться и переменной. Если у формы длинное имя, такой подход сохранит время при создании кода. К тому же, при использовании форм в разных проектах код будет работать быстрее. Например:

```
Sub FormVariableDemo()


Dim frm1 As FormAnOpinion
Set frm1 = FormAnOpinion


' Изменяет свойства и использует методы через переменную
With frm1
.Caption = "Все из вышеупомянутого"
.Show
End With
End Sub
```



Переменную, связанную с определенной формой, следует объявлять не как родовую `UserForm`.

Сокрытие видимой формы

Метод формы `Hide` используется для сокрытия формы, чтобы программа могла вернуться в документ приложения VBA или активизировать другую форму. Для сокрытия формы используется оператор

```
FormErly.Hide
```

Но метод `Hide` нельзя использовать в любом месте программы. Следует расположить его в процедуре обработки события формы. Наиболее часто оператор `Hide` располагается в конце процедуры обработки события управляющих кнопок Да и Отмена. Сокрытие формы не выгружает ее из памяти. Можно вновь быстро вывести форму на экран, любое число раз используя метод `Show`.

Удаление формы из памяти

Если программе абсолютно точно больше не понадобится форма, ее можно удалить из памяти. При загрузке формы используется оператор, а не метод. То же происходит и при выгрузке:

```
Unload FormAtion
```

Если форма видима, при выгрузке она будет удалена с экрана.

Если известно, что форма больше не понадобится, можно в процедуре обработки события, закрывающего форму, заменить оператор `Hide` на оператор `Unload` (см. предыдущий раздел).

Программирование событий для форм и элементов управления

При активации процедуры `Sub`, которая не отображает форм, написанный программистом код полностью отвечает за поведение программы. После вывода формы на экран программа впадает в пассивное состояние, ожидая получения инструкций от пользователя. Нажатие клавиши, передвижение указателя мыши или щелчок мышью генерируют события. В свою очередь, программа отмечает каждое событие, проверяя, содержит ли форма процедуру обработки такого события. Если нет, событие проходит бесследно. А если форма имеет соответствующую процедуру обработки события, программа оживает, честно запуская нужную процедуру.

Процедура обработки события может делать все то, что и обычная процедура: высчитывать значения переменных, перечислять свойства и методы объектов и даже загружать и выводить на экран формы. После того как процедура обработки события завершит свое выполнение, управление передается обратно форме. Программа возвращается в режим ожидания до появления следующего события.

Общие события

Формы VBA и их элементы управления могут определять и распознавать большое количество событий. Наиболее часто используемые события приведены в табл. 56.2.

Таблица 56.2. Некоторые события форм и элементов управления

Событие(я)	*Объекты, которых касается*	*Когда происходит*
`Activate`	Формы	Каждый раз при активизации формы (форма получает фокус)
`AddControl`	Формы, фреймы и многостраничные элементы управления	Когда к элементу управления во время выполнения добавляется новый элемент управления
`AfterUpdate`	Все активные элементы управления, кроме управляющих кнопок (не надписи, рисунки, фреймы или многостраничные элементы управления)	После того, как VBA зарегистрирует новое значение для элемента управления, перед выходом из элемента управления для перехода к следующему
`Change`	Все активные элементы управления, кроме надписей и управляющих кнопок; также многостраничные элементы управления и полосы закладок	Когда изменяется свойство `Value` элемента управления
`Click`	Формы и элементы управления всех типов	При щелчке на объекте
`DblClick`	Формы и элементы управления всех типов	При двойном щелчке на объекте
`DropButtonClick`	Поле со списком и текстовое поле	Когда раскрывается список (по щелчку на кнопке списка либо нажатии <F4>)
`Enter`	Элементы управления всех типов	Перед тем, как элемент управления получает фокус от другого элемента управления той же формы
`Error`	Формы и элементы управления всех типов	Когда происходит ошибка и информация о ней не может быть возвращена в программу
`Exit`	Элементы управления	Перед тем, как фокус

	всех типов	переходит от текущего элемента управления к другому в той же форме
KeyUp, KeyDown, KeyPress	Формы и элементы управления всех типов	При нажатии или отпускании клавиши
Layout	Формы, фреймы и многостраничные элементы управления	При изменении размера объекта
RemoveControl	Формы, фреймы и многостраничные элементы управления	При удалении элемента управления во время выполнения формы
Scroll	Формы, фреймы, многостраничные элементы управления; текстовые поля, поля со списком и списки	При изменении положения ползунка полосы прокрутки
Zoom	Формы, фреймы и многостраничные элементы управления	При изменении размера объекта (свойство Zoom)

Написание и редактирование процедур обработки событий

Форма или элемент управления могут реагировать на многие события. Но когда они реагируют на определенное событие? Только тогда, когда текущая форма или элемент управления имеют процедуру обработки этого события. Написание процедуры обработки называется *перехватом* события.

Работа с формами и элементами управления в окне Code

Написание процедуры обработки события похоже на написание любого другого кода VBA. Достаточно знать, куда ставить операторы. Код для процедуры обработки события и любой другой код, связанный с определенной формой, находится в окне кода (Code) формы редактора Visual Basic. Процедуры обработки событий элементов управления формы, как и самой формы, создаются в окне кода формы.

Написание кода обработки события формы или одного из ее элементов управления нужно начать с активизации окна формы Code. Дважды щелкните на форме или элементе либо используйте команду View Code (Код) из контекстного меню.

Следующим шагом является выбор объекта, для которого будет создаваться процедура обработки события. Его можно выбрать из раскрывающегося списка Object (Объект) в верхнем левом углу окна Code (рис. 56.20). Затем выберите событие, для которого будет создаваться код, используя раскрывающийся список Procedure (Процедура) в правом верхнем углу окна Code.

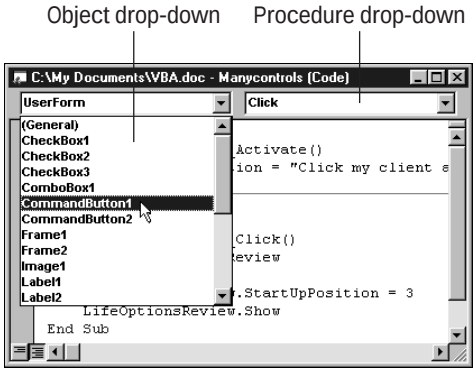


Рис. 56.20. Выбор элемента управления в окне формы Code — подготовка к написанию кода процедуры обработки события

Начало работы над процедурой обработки события

Когда из списка Procedure выбрано событие, VBA перейдет к процедуре обработки события в окне Code. Если для данной процедуры обработки события не был написан код, VBA создаст новый каркас процедуры, расположив курсор в чистой строке между объявлением и завершающим оператором. Если же процедура обработки события уже содержит код, курсор будет установлен в верхней строке существующего кода.

Синтаксис процедуры обработки события

Основной синтаксис процедуры обработки события такой же, как и для обычных процедур `Sub`, создаваемых в модулях VBA (см. главу 53). Единственное отличие заключается в имени. Чтобы процедуры обработки события работали, их имя должно включать имя объекта (формы или элемента управления), символ подчеркивания и специальное название события VBA. Если, например, кнопка называется `cmdCalculateSquareRoot`, процедура обработки события `Click` для этой кнопки должна называться `cmdCalculateSquareRoot_Click`. Обычно заботиться о названии процедуры обработки события не приходится — VBA создает объявление процедуры при выборе события в окне Code.

Единственная проблема, которая может возникнуть при работе с процедурой обработки события, появляется при изменении имени ассоциированного с ней объекта. Предположим, что написана процедура обработки события `Click` для кнопки, которая автоматически была названа `CommandButton1`. Имя процедуры обработки события, соответственно, — `CommandButton1_Click`. Затем, к примеру, вы решили дать кнопке более описательное имя, например `cmdTakeOutTheTrash`. В этой связи необходимо помнить: пока имя процедуры обработки события не изменить на `cmdTakeOutTheTrash_Click`, кнопка при запуске формы будет безжизненной, независимо от того, сколько раз на ней щелкнуть.

Работа с событием Click

Так как работа с мышью тесно интегрирована в Windows, хочется, чтобы и собственные VBA-формы реагировали на щелчки мыши. Для большинства элементов управления код обработки такого события писать не нужно, за одним, пожалуй, самым важным исключением: для управляющей кнопки код обработки события `Click` придется создавать вручную. Каждая управляющая кнопка определенно должна иметь процедуру обработки события `Click`, если, конечно, нужно, чтобы по щелчку на ней что-нибудь происходило. Следующий пример процедуры обработки события просто вычисляет число щелчков на кнопке `cmdCountClicks`:

```
Private Sub cmdCountClicks_Click()
' Объявление переменной intCount как статической сохраняет
' ее значение между вызовами процедуры
Static intCount As Integer
intCount = intCount + 1
cmdCountClicks.Caption = "Вы щелкнули на кнопке" _
& intCount & "раз"
End Sub
```

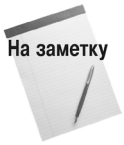
Данный код достаточно прямолинейный. При каждом запуске процедуры обработки события (что происходит по щелчку на кнопке) значение переменной `intCount` увеличивается на единицу. Это значение затем используется в строке, которая задает свойство `Caption` кнопки. Между прочим, объявление переменной `intCount` как `Static` (статической) сообщает VBA о том, что необходимо сохранить значение переменной в промежутках между вызовами процедуры обработки события. Если объявить эту переменную при помощи оператора `Dim`, VBA заново будет инициализировать переменную каждый раз при выполнении процедуры обработки события.

Очевидно, пример достаточно банальный. Однако, он иллюстрирует тот факт, что процедура обработки события выглядит и работает, как любой другой код.

Часто нет необходимости создавать процедуру обработки события Click

Большинство элементов управления VBA, которые можно расположить на форме, имеют (т.е. распознают) событие `Click`. Однако, помимо случая с управляющими кнопками, обычно нет необходимости в написании процедуры обработки события `Click`. Причина заключается в следующем: эти элементы управления реагируют на щелчки автоматически — так, как нужно программисту.

Например, в форме VBA размещены переключатели. После запуска формы по щелчку на переключателе он выбирается, поэтому программировать данное действие не нужно. Подобным образом VBA обрабатывает щелчки на выключателе или флажке, а текстовое поле автоматически реагирует на щелчок, устанавливая курсор в том месте, где щелкнул пользователь.



Код придется написать для передачи в программу тех настроек элементов управления, которые сделаны пользователем, — этот шаг VBA самостоятельно сделать не сможет. Подробнее см. разделы “Реакция на изменения элемента управления” и “Проверка ввода”.

События Click для пользовательских форм

Даже формы имеют события `Click`. При желании вся форма может работать как одна большая кнопка; щелчок в любом месте формы приводит к определенному действию программы. Во время работы программы VBA вызывает процедуру только по щелчку на той части формы, которая не закрыта элементами управления.

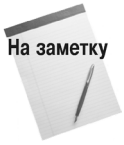
Реакция на изменение элемента управления

При щелчке на стрелке “вверх” в верхней части полосы прокрутки или на кнопке счетчика необходимо убедиться в ответной реакции. Как значение изменится после щелчка? И был ли щелчок на кнопке вообще? Перехват события `Change` позволяет написать код для обеспечения такой реакции. Событие `Change` происходит при изменении значения элемента управления (когда изменяется значение свойства `Value` элемента управления) и наиболее часто применяется с переключателями, флажками, выключателями, счетчиком и полосами прокрутки.

Самая простая реакция на событие `Change` такова — отобразите измененное значение свойства `Value` как заголовок надписи. Конечно, процедура обработки события `Change` может выполнять с новым значением элемента управления и другие действия, а не только выводить его на экран:

- проверить, отвечает ли значение определенным критериям (см. “Проверка ввода” далее в этой главе);
- выполнить вычисления на основании значения или другие действия в зависимости от результата вычислений;
- использовать значение элемента управления для изменения других установок, например, громкости звука.

Событие `Change` происходит каждый раз при изменении значения элемента управления, при каждом щелчке мышью или нажатии клавиши. Иногда целесообразнее внести изменения перед обработкой. В этой ситуации можно воспользоваться событиями `BeforeUpdate` и `AfterUpdate`. Событие `BeforeUpdate` лучше применять при проверке пользовательского ввода (см. раздел “Проверка ввода” далее в этой главе). Событие `AfterUpdate` происходит, когда пользователь заканчивает работу с элементом управления и переходит к другому элементу управления. Это событие подходит для выполнения расчетов с измененным значением элемента управления или использования этого значения для изменения других установок.



Одно предупреждение: событие `AfterUpdate` происходит только в том случае, когда значение элемента управления изменяется

пользователем. Если изменить текст в поле с помощью оператора типа `txtHerTextBox = strNewsFlashSc1`, следует воспользоваться событием `Change` — лишь тогда изменение текста приведет к выполнению определенных действий.

Определение нажатия клавиши

Чтобы определить, что пользователь нажал на клавиатуре, используйте события `KeyPress`, `KeyDown` и `KeyUp`. Событие `KeyPress` хорошо подходит для чтения обычных "печатаемых" клавиш, когда происходит ввод в поле или поле со списком. Также это событие распознает многие комбинации: <Ctrl+клавиша> и <Backspace>. События `KeyDown` и `KeyUp` определяют практически любую комбинацию, включая такие, как <Alt+Shift+Ctrl+F9>. Хотя работать с ними тяжелее, чем с `KeyPress`, эти события обладают большей гибкостью при создании горячих клавиш. Например, можно написать процедуру обработки события `KeyDown`, чтобы комбинации <Ctrl+левая клавиша управления курсором> и <Ctrl+правая клавиша управления курсором> уменьшали и увеличивали значение полосы прокрутки или счетчика на 10.

Описание общих задач по программированию форм

Разобравшись в том, как работают процедуры обработки событий, можно сконцентрировать внимание на создании диалоговых окон, которые будут действовать ожидаемым для программиста и пользователя образом. В этом разделе приводятся подсказки, описывающие многие общие сценарии при построении диалоговых окон.

Добавление кнопки Закрыть или Отмена

Большинство диалоговых окон имеют, как минимум, одну главную кнопку: ту, что удаляет диалоговое окно с экрана. В зависимости от того, как функционирует остальная часть диалогового окна, эта кнопка называется Закрыть (Close) или Отмена (Cancel) (но названия типа Выход, Завершение, Все, хватит или Сдаюсь тоже подойдут). Такая кнопка должна быть в каждой форме.

По негласному соглашению, как кнопка Закрыть, так и кнопка Отмена скрывают или выгружают форму. Вот как определить, какая из них должна быть в форме.

- Используйте кнопку Закрыть для формы, которая просто отображает информацию или немедленно выполняет задачи, не изменяя ни настроек программы, ни переменных, используемых далее в программе.
- Применяйте кнопку Отмена в форме, которая изменяет переменные или настройки программы. По щелчку на кнопке Отмена диалоговое окно закрывается без записи внесенных изменений — все остается в том же состоянии, что было перед запуском диалогового окна. В форме также должна присутствовать кнопка Да — на случай, если захочется подтвердить внесенные изменения.

Простые процедуры обработки событий кнопок Закрыть и Отмена

Как и любой другой командной кнопке, кнопке Закрыть или Отмена требуется процедура обработки события для выполнения работы в ответ на щелчок мыши. В большинстве случаев процедура обработки события нуждается в наличии только одного оператора, как в следующих двух примерах:

```
Private Sub cmdClose_Click()
    Hide ' ссылка на текущую форму
End Sub


Private Sub cmdCancel_Click()
    Unload frmOptions
End Sub
```

Конечно, процедура обработки события, привязанная к кнопке Закрыть или Отмена, может выполнять перед закрытием формы и другую работу. Например, выводить сообщение с запросом, действительно ли пользователь хочет закрыть диалоговое окно:

```
Private Sub cmdCancel_Click()
    Message = "Вы действительно хотите закрыть" _
        & "диалоговое окно и снять все внесенные" _
        & "изменения?"
    If MsgBox(Message, vbYesNo) = vbYes Then
        Hide ' Скрыть по щелчку на кнопке Yes
    End If ' Иначе ничего не делать
End Sub
```

Клавиатурный аналог

Свяжите с кнопкой Закрыть или Отмена клавишу <Esc>. Ею обычно пользуются, чтобы выйти из диалогового окна без сохранения внесенных изменений — не разочаровывайте пользователей. Добавлять процедуру обработки события `KeyDown` не нужно, просто в окне Properties установите значение свойства `Cancel` равным `True`.

Программирование кнопки Да

Предположим, что пользователь щелкнул на кнопке Да. Он ожидает, что программа воспримет текущие значения формы как окончательные, внесет изменения в вид, в поведение программы или содержащиеся в ней данные, после чего форму необходимо будет удалить с экрана.

Оправдайте ожидания пользователя, создав процедуру обработки события `Click` для кнопки Да. Код должен передавать значения из элементов управления формы переменным программы или использовать значения элементов управления в условных

операторах. Последний оператор процедуры должен содержать метод Hide или оператор Unload. В следующих примерах txtCName и txtCaddress — текстовые поля. Первые два оператора передают их содержимое соответствующим переменным программы. Затем программа проверяет значение выключателя tglSend и, если он включен, запускает процедуру SendBillToCustomer. После чего при помощи метода Hide форма скрывается.

```
Private Sub cmdOk_Click()
    strCustomerName = txtCName.Value
    strCustomerAddress = txtCAdrress.Value
    ' проверка состояния переключателя
    If tglSend.Value = True Then
       SendBillToCustomer
    End If
    Hide
End Sub
```

Проверка ввода

Одной из наиболее частых задач, которые решаются в процедурах обработки событий, является проверка значения, введенного в элемент управления. Зачастую программа способна воспринять только определенные величины, а пользователь может ввести любой текст в поле или поле со списком и выбрать произвольное значение при помощи ползунка или кнопок счетчика. Решением проблемы является использование в процедурах обработки событий проверочного кода. Код проверяет введенную пользователем информацию на соответствие определенным критериям. Если данные удовлетворяют критериям, они сохраняются или передаются в другую часть программы. А если нет, можно вывести сообщение, информирующее пользователя о проблеме (рис. 56.21). Также код может преобразовать введенную информацию в приемлемую для себя (скажем, конвертировав все символы в заглавные).

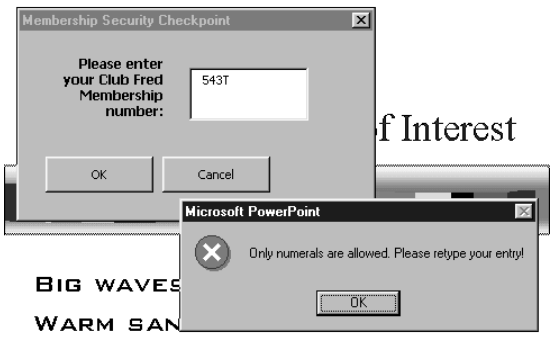


Рис. 56.21. Для проверки ввода в текстовом окне использована процедура обработки события Change. На экран выведено сообщение об ошибке

Коды для процедур проверки

Проверка значений элемента управления требует использования операторов `If...Then`, `Select Case` или и тех, и других. В следующем примере проверяется, не принимает ли счетчик недопустимые значения:

```
Private Sub spnVolumeControl_Change()
    If spnVolumeControl.Value = 11 Then
        MsgBox "11 — недопустимое значение"
    End If
End Sub
```

События, используемые для проверки ввода

Проверять значение ввода стоит в нескольких местах взаимодействия пользователя и формы. Каждая из точек проверки отвечает различным процедурам обработки событий элемента управления или формы как целого:

Когда проводится проверка	*Используемая процедура обработки события*
Каждый раз при изменении значения элемента управления	Событие `Change` элемента управления (оценивает само значение)
Каждый раз по нажатию клавиши	Событие `KeyPress` элемента управления (оценивает каждую нажатую клавишу)
По окончании работы с элементом управления перед переходом к следующему	`BeforeUpdate` (позволяет отменить событие и вернуть пользователя к элементу управления)
По закрытии формы	Событие `Click` для кнопок Да и Отмена формы

Отвод или изменение отдельных символов

Возможно, понадобится, чтобы в текстовом поле или поле со списком нельзя было ввести определенные символы. Для того чтобы избавляться от них по мере ввода, используйте процедуру обработки события `KeyPress`. Следующий код отвергает все символы, кроме букв и цифр:

```
Private Sub txtSerialNumber_KeyPress(ByVal KeyAscii _
    As MSForms.ReturnInteger)
' Весь следующий блок является условием
If  Chr(KeyAscii) < "0" Or _
    (Chr(KeyAscii) > "9" And Chr(KeyAscii) < "A") Or _
    (Chr(KeyAscii) > "Z" And Chr(KeyAscii) < "a") Or _
```

```
    Chr(KeyAscii) > "z"  _ Then


    MsgBox "Неверный символ!"
    KeyAscii = 0 ' Отбрасывает символ


End If
End Sub
```

После того как оператор `If...Then` определяет неверный символ, на экран выводится сообщение. Затем выполняется оператор `KeyAscii = 0`. `KeyAscii` — аргумент процедуры обработки события `KeyPress`, работающий как локальная переменная процедуры. Изменение ее значения изменяет код символа, отправленного в текстовое поле. Поскольку поле само по себе не воспринимает символ с кодом 0, введенный пользователем символ исчезает без следа.

Изменение значения `KeyAscii` также позволяет превратить неправильный ввод в правильный. Например, можно использовать следующую процедуру обработки событий для вывода и хранения набранных пользователем символов в верхнем регистре:

```
Private Sub txtSerialNumber_KeyPress(ByVal KeyAscii _
    As MSForms.ReturnInteger)
KeyAscii = Asc(Ucase(Chr(KeyAscii)))
End Sub
```

Оператор применяет три вложенные функции. Поскольку `KeyAscii` — цифровой код, его необходимо конвертировать в строку при помощи оператора `Chr`, затем при помощи `Ucase`, перевести в верхний регистр, после чего конвертировать обратно в целое число при помощи `Asc`.

Проверка в событии BeforeUpdate

Иногда нужно отложить проверку ввода, пока пользователь не закончит работу с ним. Некоторым нравится, когда позволяется исправлять ошибки перед вводом информации в программу и последней проверкой. Проверку можно отложить и по другой причине, например, из-за значительного интервала времени, необходимого для запуска кода проверки.

Чтобы проверить значение элемента управления после того, как пользователь сообщил об окончании работы, создайте процедуру обработки события `BeforeUpdate`. Эти события происходят при щелчке на другом элементе управления, при нажатии клавиши <Tab> или комбинации клавиш для выбора другого элемента управления. VBA определяет событие `BeforeEvent` перед тем, как оставить исходный элемент. Оно позволяет отменить переход к новому элементу управления и заставить пользователя решить проблему. Вот пример использования оператора `Cancel`:

```
Private Sub txtSerialNumber_Change()
    If len(txtSerialNumber.Value) > 5 Then
```

```
        MsgBox "Ваша строка чересчур длинная. Попробуйте еще
раз."
        Cancel
    End If
End Sub
```

Ожидание закрытия формы

Иногда полезно отложить проверку элемента управления, пока пользователь не щелкнет на кнопке Да, чтобы закрыть всю форму. Это разумно, если критерии проверки зависят от значений нескольких элементов управления. В данном случае, чтобы вставить код проверки, необходимо поместить в процедуру обработки события `Click` кнопку Да.

Трансформация форм и элементов управления

При наличии процедур обработки событий форма превращается в динамичный объект, реагирующий на события путем изменения внешнего вида, а не только данных в программе. Это является ключевым свойством для создания профессионально выглядящих форм.

Метод трансформации 1: изменение свойств элемента управления

Самый простой путь изменения внешнего вида формы —изменение ее свойств или свойств ее элементов управления в коде процедуры обработки событий. Как видно из нескольких предыдущих примеров, основное применение данной идеи состоит в изменении надписи, что позволяет отсылать сообщения пользователям исходя из текущих условий.

В следующем примере в форме присутствует управляющая кнопка `cmdClickMe` и надпись `lblLittleMessage`. Щелчок на кнопке изменяет заголовок надписи. Процедура обработки события `Click` выглядит следующим образом:

```
Private Sub cmdClickMe_Click()

' Объявляется статическая переменная для подсчета числа
' запусков процедуры
Static X as Integer

X =X + 1
Select Case X Mod 3
    Case 0
        lblLittleMessage.Caption = Now
    Case 1
```

```
        lblLittleMessage.Caption = CStr (X/2 * X)
    Case 2
        lblLittleMessage.Caption = _
            "Жизнь — это цепь событий"
    End Select
End Sub
```

Конечно, никто не ограничивает программиста изменением текста надписей. Можно изменять любое свойство любого объекта формы, кроме свойств только-для-чтения, доступ к которым запрещен. Установите значение свойства `Enabled` равным `False`, чтобы сделать элемент недоступным для пользователя. Можно даже изменить значение `Value`, отображаемое в элементе управления.

Метод трансформации 2: многочисленные наборы элементов управления

Изменить вид формы в ответ на какое-либо событие достаточно просто, но для изменения функций элементов управления потребуется гораздо больше усилий. Предположим, программа выводит тестовые вопросы, получает ответ пользователя и затем показывает сообщение о том, является ли выбранный ответ верным. Существует и более удачный метод. Можно использовать одну форму, выводя один набор элементов управления, когда задается вопрос, и другой, когда нужно дать ответ. Такой подход требует б*о*льших усилий, но ядерной физикой там и не пахнет.

Простой способ разработать одну форму, работающую как две (или больше) — добавить отдельный элемент управления для каждой функции, а затем воспользоваться свойством `Visible`, чтобы скрыть или вывести их по своему усмотрению.

Установка и использование элементов управления ActiveX

Помимо репутации программного империалиста, фирма Microsoft прилагает усилия, чтобы сделать средства разработки “открытыми”. Основываясь на спецификации ActiveX, каждый может создать новые элементы управления, которые будут работать с любым языком программирования под Windows, включая C++, HTML, Visual Basic и VBA. Преимущество заключается в том, что с помощью элементов управления ActiveX в программу можно добавить новые возможности, которых нет в VBA.

Добавление новых элементов управления на панель инструментов

Перед использованием элемента управления ActiveX в VBA-программе, нужно установить на жесткий диск компьютера программное обеспечение для элемента управления и зарегистрировать элемент управления в Windows. Существуют разные способы регистрации элементов управления, но во время инсталляции он будет зарегистрирован автоматически. Если приходится выполнять регистрацию самому, проще и удобнее это делать в редакторе Visual Basic.

Регистрация элемента управления

Чтобы зарегистрировать новый элемент управления, убедитесь, что известно имя файла, содержащего элемент управления, и его расположение на жестком диске. Затем выполните следующие шаги.

1. Выберите команду Tools⇨References (Сервис⇨Ссылки). Появятся ссылки ActiveX, доступные в проекте.
2. В диалоговом окне щелкните на кнопке Browse (Обзор). Вы увидите стандартное диалоговое окно Windows для открытия файлов.
3. Выберите ActiveX controls из раскрывающегося списка Files of Type (Тип файла).
4. Найдите файл с нужным элементом управления и дважды щелкните на нем, чтобы открыть. Вы вернетесь в диалоговое окно Referencies.
5. Пролистывая элементы управления, убедитесь, что рядом с нужным файлом установлен флажок.
6. Закройте диалоговое окно.

Как поместить элемент управления в панель инструментов

Когда новый элемент управления зарегистрирован, активизируйте его, разместив его пиктограмму в панели элементов (Toolbox).

1. Вызовите панель элементов, выбрав любое окно пользовательской формы, а затем — команду View⇨Toolbox (Вид⇨Элементы управления).
2. Щелкните правой кнопкой мыши на панели Toolbox и выберите либо в контекстном меню Additional Controls (Другие элементы), либо команду Tools⇨Additional Controls (Сервис⇨Другие элементы). Появится одноименное диалоговое окно (рис. 56.22).

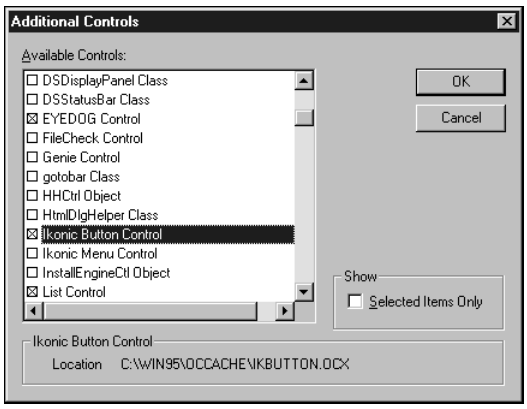


Рис. 56.22. Диалоговое окно для активизации новых элементов управления ActiveX

3. Пролистайте список доступных элементов управления до появления необходимого и установите рядом с ним флажок.

4. Закройте диалоговое окно. На панели Toolbox появится пиктограмма активизированного элемента управления. На рис. 56.23 показана панель Toolbox с набором новых элементов управления.

Рис. 56.23. Настроенная панель инструментов со множеством дополнительных элементов управления ActiveX

Если используется большое количество других элементов управления, их можно разместить на отдельных страницах панели Toolbox. Чтобы добавить в панель элементов новые страницы, щелкните правой кнопкой на вкладке вверху и выберите New Page (Создать страницу) (как в многостраничном элементе управления в форме). Пиктограмма активизированного элемента управления будет размещена на странице, открытой во время его активизации.

Использование элементов управления ActiveX в своих программах

Как только элемент управления ActiveX попал на панель элементов, его можно добавлять в формы тем же способом, что и любые элементы управления VBA. Однако, чтобы заставить такой элемент управления выполнить полезные действия, нужно знать, как работают его свойства и методы. Здесь понадобится документация и файлы справки, прилагаемые к элементу управления. Если элемент управления разработан и установлен правильно, справку по любому его свойству можно получить, нажав <F1> (когда свойство выбрано в окне Properties).

Невидимые элементы управления

Многие действительно ценные элементы управления во время работы программы невидимы. Вместо непосредственного взаимодействия с пользователем, такие элементы управления выполняют работу за программиста — ему не приходится создавать собственные объекты и процедуры для такой работы (например, счетчик или элемент управления, который выполняет расчеты в фоновом режиме). Элементы, невидимые во время работы формы, можно увидеть во время ее проектирования.

Чтобы использовать невидимый элемент управления в программе, добавьте его в форму, как и любой другой элемент управления. Поскольку для пользователя он невидим, его место расположения значения не имеет. Главное — разместите его там, где он не будет пересекаться с другими элементами управления.