**ФОРМЫ VBA**

До сих пор использовались диалоговые окна, которые встроены в VBA, а именно, функции *MsgBox* и *InputBox*. Хотя *MsgBox* и *InputBox* придают программам гибкость, которой могут обладать только интерактивные программы, их возможности в известной степени ограничены. При разработке более сложных программ, может понадобиться выводить диалоговые окна, позволяющие пользователям программ задавать при помощи одного диалогового окна несколько опций, выбирать пункты из списка или вводить в одном окне несколько значений. Часто возникает необходимость вместо встроенных окон, принадлежащих Excel, использовать диалоговые окна, соз­данные специально для разработанной программы.

VBA позволяет создавать и применять пользовательские диалоговые окна в написанных программах и процедурах при помощи добавления в проект объекта *UserForm*. Используя формы пользователя VBA (VBA User Forms), можно создавать диалоговые окна для вывода данных или получения значений от пользователя программы именно в том виде, который требуется программе. Например, можно вывести на экран диалоговое окно со списком различных вариантов формата даты и предоставить пользователю возможность выбрать из списка один из форматов.

Диалоговые окна позволяют программе общаться с пользователем наиболее удобным образом, обеспечивая гибкую форму ввода и вывода данных.

Пользовательская форма *UserForm* предоставляет пользователю возможность создавать диалоговые окна разрабатываемых приложений. Она служит базой пользовательского диалогового окна, на которой в зависимости от решаемой задачи размещают элементы управления.

Диалоговое окно в VBA создается добавлением в проект объекта *UserForm*. Объект *UserForm* – это пустое диалоговое окно. Настройку диалогового окна можно выполнить добавлением к объекту *UserForm* (обычно называют просто *форма*) элементов управления. Каждому объекту *UserForm* присущи определенные свойства, методы и события, которые он наследует от класса объектов *UserForm*. Каждый объект *UserForm* включает в себя также *модуль класса*, в который можно добавлять собственные методы и свойства или код обработки событий формы.

Первый шаг в создании пользовательского диалогового окна состоит в добавлении к проекту новой формы (объекта *User Form*). Форма содержит рабочую область, в которую можно поместить элементы, необходимые для выполнения некоторого диалога пользователя с приложением.

Для добавления к проекту новой формы надо использовать команду VB-редактоpa *Insert* → *UserForm* (Вставка → UserForm). Редактор VBA добавляет к текущему проекту новую форму, присваивая ей по умолчанию имя *UserFormN* и используя ту же систему нумерации, что и для модулей. Редактор VBA выводит новую форму *в режиме разработки*, как показано на рис. 9.9. В режиме разработки можно добавлять (или удалять) элементы управления к форме, устанавливать свойства формы или ее элементов управления и выполнять другие манипуляции с внешним видом формы в интерактивном режиме. Когда форма выведена и используется как часть выполняющейся программы, она находится в режиме *выполнения*.

На рис. 9.9 показана добавленная в проект форма *UserForm*1. Широкая рамка вокруг формы указывает на то, что форма выделена. Сетка из точек наповерхности формы помогает выравнивать и контролировать размеры элементов управления, помещаемых на форму, и появляется только в режиме *разработки*. Панель элементов является «палитрой», с помощью которой можно выбирать элементы управления и добавлять их к форме

Объект *UserForm* можно переименовать так же, как стандартный модуль или модуль класса. Для этого в Properties Window (окно свойств) надо редактировать свойство (*Name*) и *Caption* этого объекта.

Когда форма выводится на экран в *режиме разработки*, можно протестовать ее поведение, используя команду *Run* → *Run Sub/User Form* (Запуск → Запуск подпрограммы/UserForm). После этого редактор VBA выведет форму режиме запуска и все ее элементы управления будут активными.



Рис. 9.9. Новая UserForm в режиме разработки



Рис. 9.10. Новая UserForm в режиме запуска

**Свойства объекта UserForm**

Форма имеет некоторые встроенные свойства, и эти свойства можно устанавливать или программным образом, или в *Properties Window* (окне свойств) редактора VBA. Строго говоря, данные способы изменения свойств форм не являются эквивалентными. Некоторые из свойств могут быть установлены только посредством *Properties Window*. Программным способом свойства форм устанавливаются таким же образом, как и свойства других объектов: путем присвоения свойству нового значения. В табл. 9.7 перечислены свойства *UserForm*, на которые скорее всего, придется ссылаться или изменять их.

Таблица 9.7. Наиболее часто используемые свойства объектов UserForm

| Свойство | Описание |
| --- | --- |
| ActiveControl | Возвращает объектную ссылку на элемент управления, находящийся в фокусе в данный момент. Только для чтения |
| BackColor | Целое число типа *Long* определяет цвет фона формы. Самый простой способ установить это свойство – использовать Properties Window; чтобы выбрать желаемый цвет (если необходимо), можно скопировать номер цвета из Properties Window в свою программу |
| BackColor | Возвращает цвет фона формы |
| BorderStyle | Устанавливает тип границы |
| Caption | Текст, выводимый в качестве заголовка формы. Запись/Чтение |
| Caption | Возвращает текст, отображаемый в строке заголовка формы |
| Controls | Возвращает коллекцию всех элементов управления формы. Только для чтения |
| Cycle | Определяет, должно ли нажатие клавиши табуляции вызывать последовательный выбор всех элементов управления во всех группах и на каждой странице многостраничных элементов управления или только в пределах текущей группы или страницы. Может содержать одну из двух встроенных констант: *fmCycleAllForms* или f*mCycleCurrentForm*. Чтение/Запись |
| Enabled | Содержит значение типа *Boolean*, указывающее, доступна ли форма. Если его значение равно *False*, ни один из элементов управления формы не доступен. Чтение/Запись |
| Font | Возвращает ссылку на объект *Font*, посредством которого можно выбрать параметры шрифта формы или элемента управления |
| ForeColor | То же самое, что и свойство *BackColor*, но устанавливает цвет используемый для переднего плана (обычно – это цвет текста) объекта формы |
| Height И Width | Возвращают высоту и ширину формы в пунктах |
| Left И Top | Возвращают местоположение верхнего левого угла формы в пунктах |
| Name | Возвращает имя пользовательской формы |
| Picture | Указывает рисунок, отображаемый как фон формы |
| StartUpPosition | Возвращает значение, определяющее расположение формы при ее первом отображении на экране. Допустимые значения: * Manual (начальное значение не устанавливается),
* CenterOwner (выравнивание по центру объекта, к которому принадлежит форма)
* CenterScreen (выравнивание по центру экрана)
* windows Default (положение верхнего левого угла экрана)
 |

**Методы объекта UserForm**

Всякий раз создавая в проекте новый объект *UserForm*, создается новый подкласс объекта *UserForm*. Любые процедуры или функции, написанные вами в разделе General (общий) модуля класса, относящегося к форме, становятся дополнительными методами для отдельного подкласса объекта. Можно также создать для формы новые свойства, добавив в ее модуль класса процедуры Property Get и Property Let. Можно также создавать экземпляры подкласса UserForm с помощью оператора Dim и ключевого слова New.

Однако чаще всего манипулируют объектом формы при помощи стандартных методов и свойств класса UserForm и при помощи собственных процедур обработки событий для выбранной формы и ее элементов управления.

В табл. 9.8 перечислены наиболее часто используемые методы для объектов UserForm, которыми можно воспользоваться, и кратко изложены их свойства. Эти методы будут доступны для каждой формы, которая добавляется в проект.

Таблица 9.8.Наиболее часто используемые методы объекта UserForm

| Метод | Назначение |
| --- | --- |
| Copy | Копирует выделенный в элементе управления текст в буфер обмена Windows |
| Cut | Вырезает выделенный в элементе управления текст и помещает его в буфер обмена Windows |
| Hide | Закрывает форму. Скрывает UserForm, не выгружая ее из памяти, сохраняя значения элементов управления формы и всех переменных, объявленных в модуле класса формы |
| Paste | Вставляет содержимое буфера обмена Windows в текущий элемент управления |
| PrintForm | Выводит на используемый в Windows по умолчанию принтер изображение формы, включая все данные, введенные в элементы управления  |
| Repaint | Перерисовывает форму, выведенную на экран. Метод используется, если надо перерисовать форму, не ожидая, когда она будет перерисована через обычный период времени |
| Show | Выводит форму на экран. Если форма еще не загружена в память, то данный метод сначала ее загружает |

**Синтаксис метода Show**

*FormName*.Show

В данной синтаксической конструкции *FormName* может быть любым объектом *UserForm* текущего проекта. *FormName* – имя формы в том виде, как оно отображается в *Project Explorer*. Например, если имеется форма *frmInsertFigure*, вывести ее на экран можно с помощью оператора:

frmlnsertFigure.Show

Если форма в данный момент не загружена в память, метод *Show* загрузит ее и выведет на экран. Если форма уже загружена, метод *Show* просто выведет ее на экран. В любом случае метод *Show* выводит форму и затем передает ей управление. Форма будет оставаться на экране до тех пор, пока или не будет выполнен метод *Hide* объекта *User Form*, или форма не будет выгружена при помощи оператора *Unload*.

**События и событийные процедуры**

*Событие* – это что-то, что может произойти с диалоговым окном или элементом управления диалогового окна. Типичные примеры событий: щелчок на кнопке, переключателе и т.д. Другие примеры событий: изменение содержимого окна редактирования или выбор элемента списка. Щелчок мышью, нажатие клавиши и действия, внутренние для вашего компьютера, – все они *запускают* или иными словами влекут за собой события.

Использование событий позволяет создавать действительно диалоговые приложения. В таких приложениях все действия пользователя приводят к определенной реакции приложения, если эти действия предусмотрены и не заблокированы. Если надо, чтобы пользователи чувствовали себя комфортно при работе с программами, начинать создавать диалоговые окна надо с разработки событийных процедур.

Такие объекты, как формы и элементы управления приводят в действие некоторые события. Можно написать собственные VBA-процедуры, реагирующие на события. Такие процедуры, называются *событийными процедурами* или *процедурами обработки событий.*

Событийные процедуры следует записывать в модуль класса, который является частью *UserForm*. При этом такие процедуры должны иметь имена в виде *ObjectName\_EventName,* где *ObjectName* – имя формы или элемента управления, a *EventName* – имя события, с которым следует работать. Такой формат имени, позволяет VBA сопоставлять заданному событию требуемую процедуру.

Большая часть программы, которую вы записывается в модуль класса формы, будет связана с обработкой событий. В табл. 9.9 перечислены события, для которых можно написать процедуры обработки.

Таблица 9.9. События объектов UserForm

| Событие | Синтаксис заголовка процедуры обработки события | Описание |
| --- | --- | --- |
| Activate | Private Sub objecf\_Activate() | Инициируется всякий раз, когда окно формы становится активным. Это событие используется для обновления содержимого диалоговых элементов управления, чтобы отразить любые изменения, которые произошли, пока окно формы было неактивным |
| Click | Private Sub object\_Click(index As Long) | Инициируется всякий раз, когда по форме (любой ее части, не занятой элементами управления) щелкают мышью |
| DblClic | Private Sub object\_DblClick(index As Long,ByVal *Cancel* AsMSForms.ReturnBoolean) | Инициируется всякий раз, когда по форме (любой ее части, не занятой элементами управления) дважды щелкают мышью |
| Deactivate | Private Sub object\_Deactivate() | Инициируется всякий раз, когда форма перестает быть активной |
| Initialize | Private Sub object\_Initialize() | Инициируется всякий раз, когда форма впервые загружается в память посредством выполнения оператора *Load* или с помощью метода *Show*. Это событие используется для инициализации элементов управления формы при ее появлении на экране. |
| Resize | Private Sub *UserForm*\_Resize() | Инициируется при изменении размеров формы |
| Terminate | Private Sub object\_Terminate() | Инициируется всякий раз, когда форма выгружается из памяти. Событие используется для осуществления любых специальных служебных задач, которые необходимо выполнить прежде, чем переменные формы будут выгружены |

**Примеры программ модуля класса формы**

Рассмотрим несколько простых примеров использования свойств и методов формы с помощью процедур обработки событий. Будем инициировать события, описанные в табл. 9.9, а в ответ на них с помощью окна функций MsgBox будем сообщать о типе события или менять видимые свойства формы. Начнем с события Click, так как процедуру обработки этого события легче всего создать: достаточно дважды щелкнуть на форме (используем созданную в начале главы форму) в режиме разработки, и появится шаблон процедуры. Запишем после заголовка процедуры следующий оператор MsgBox ("Событие: Click"):

Private Sub UserForm\_Click()

MsgBox ("Событие: Click")

End Sub

После запуска формы на выполнение на экране поверх формы появится форма. Щелкнув на ней мышью, на экране появится сообщение в виде сообщения "Событие: Click"

Если даже не часто будет нужно использовать подобное событие для формы, не стоит забывать о такой возможности. А еще лучше поступать следующим образом. Всегда использовать это событие на случай, когда пользователь форм при попытке щелкнуть на каком либо элементе управления не совсем точно позиционирует курсор мыши. Если при этом будет выполняться процедура обработки события *Click* для формы с выводом предупреждающего сообщения о недостаточной уверенности в действиях пользователя, с приложением станет приятно работать.

Рассмотрим событие *Activate*. Запишем в модуле рассматриваемой формы следующую процедуру:

Private Sub UserForm\_Activate()

MsgBox ("Событие: Activate")

End Sub

После запуска формы и ее появления на экране появится сообщение о том, что произошло событие *Activate*.

Рассмотрим события *Deactivate* и *Terminate*. Первое из них инициируется всякий раз, когда форма перестает быть активной, а второе – когда форма выгружается из памяти. Процедура обработки события *Terminate* просто добавляется к имеющейся коллекции программ формы UserForml и выполняется при завершении работы с формой. Для демонстрации же события *Deactivate* необходимо, не выходя из приложения, сделать форму UserForml неактивной. Это можно осуществить, если активной станет другая форма. И ее надо создать.

***Создадим новую форму*** либо при помощи кнопки *Insert UserForm*, либо командой *Insert* → *UserForm*. Пусть она имеет имя, которое ей присваивается по умолчанию: UserForm2 Наметим план тестирования:

1. Загружаем форму UserForml.

2. Событие *Click* формы UserForml вызывает метод *Show* для загрузки фомы UserForm2.

3. Загрузка/выгрузка формы UserForm2 не сопровождаются никакими событиями.

4. Форма UserForml имеет процедуры обработки событий: *Click*, *Activate*, *Deactivate*, *Terminate*.

Для выполнения поставленной задачи в модуль формы UserForml необходимо поместить код листинга 43.

**Пример 43.** Процедуры событий UserForml.

'Обработчик события Click формы UserForml

Private Sub UserForm\_Click()

'Вывод окна сообщения на экран:

MsgBox ("Событие: Click. Выводим UserForm2")

'Загрузка формы UserForm2:

UserForm2.Show

End Sub

'Обработчик события Activate формы UserForml

Private Sub UserForm\_Activate()

'Вывод окна сообщения на экран:

MsgBox ("Событие: Activate для формы UserForml")

End Sub

Private Sub UserForm\_Deactivate()

'Вывод окна сообщения на экран:

MsgBox ("Событие: Deactivate для формы UserForml")

End Sub

Private Sub UserForm\_Terminate()

'Вывод окна сообщения на экран:

MsgBox ("Событие: Terminate для формы UserForml")

End Sub

После запуска формы на выполнение можно проследить за инициализацией всех четырех событий. Сразу после за грузки UserForml выдается сообщение "Событие: Activate для формы UserForml" (процедура UserForm\_Activate). Щелчок на форме приводит к выводу сообщения "Событие: Click. Выводим UserForm2" (процедура UserForm\_Clic), загрузке формы UserForm2 (процедура UserForm\_Click) и выводу сообщения "Событие: Deactivate для формы UserForml" (процедура UserForm\_Deactivate) в результате того, что форма UserForm1 становится неактивной, Если бы была записана процедура события *Activate* и для формы UserForm2, то один щелчок мыши вызвал бы инициализацию трех событий. При выгрузке из памяти формы UserForml на экран дается сообщение процедуры UserForm\_Terminate.

Событие *Initialize* инициализируется при загрузке формы при помощи оатора *Load* или метода *Show*. При запуске формы это событие не инициализируется. Это событие следует использовать при первой загрузке формы для установки каких-либо свойств формы ее элементов управления. В примере 44 для этого coбытия устанавливается свойство формы *BackColor*.

Для загрузки формы UserForm2 использется форма UserForml с программой обработки событий из примера 43. В примере 44 для установки свойства формы *BackColor* используется функция RGB, которая возвращает RGB-значение типа *Long*, используемое далее для присвоения значению UserForm2.BackColor. Синтаксис функции RGB:

RGB(*red*, *green*, *blue)*

Именованные аргументы:

|  |  |
| --- | --- |
| Аргумент | Описание |
| Red | Обязательный; тип: **Variant (Integer).** Число в диапазоне 0-255; представляет красный компонент цвета |
| Green | Обязательный; тип: **Variant (Integer).** Число в диапазоне 0-255; представляет зеленый компонент цвета |
| Blue | Обязательный; тип: **Variant (Integer).** Число в диапазоне 0-255; представляет синий компонент цвета |

**Пример 44.** Процедуры событий UserForm2

Dim sRED, sGREEN, sBLUE ' переменные задания цвета формы

'Процедура обработки события Initialize

'Инициализируется один раз: при загрузке

Private Sub UserForm\_Initialize()

'задаем начальный цвет формы

sRED = 100

sGREEN = 100

SBLUE = 200

UserForm2.BackColor = RGB(sRED, sGREEN, sBLUE)

End Sub

'Процедура обработки события Click

'При каждой инициализации меняет цвет формы

'При недопустимых значениях sRED, sGREEN, sBLUE

'выдает ошибку времени исполнения

Private Sub UserForm\_Click()

Dim i

'Меняем цвет формы:

sRED = sRED + 20

sGREEN = sGREEN +10

sBLUE = sBLUE - 20

i = RGB(sRED, sGREEN, sBLUE)

UserForm2.BackColor = i

'Изменить заголовок формы:

UserForm2.Caption = "Цвет: " & Str(i)

End Sub

После вывода формы с именем *UserForm*2 на экран событие *Click* , будет приводить к изменению цвета и заголовка формы.

**Элементы управления**

Объект *UserForm* может содержать те же элементы управления, что и находящиеся в диалоговых окнах Word, Excel или других приложений Windows *Элементы управления* – это элементы диалогового окна, позволяющие пользователю взаимодействовать с программой. Они включают в себя кнопки-переключатели, текстовые поля, линейки прокрутки, командные кнопки и т.д.

Многие производители программного обеспечения сделали доступным» пакеты, расширяющие набор элементов управления. Можно добавлять элементы управления (известные как объекты ActiveX и Automation) к панели *Toolbox* (панели элементов), предварительно создавая ссылку из проекта на библиотеку, содержащую расширенный набор элементов управления. После этого можно добавлять данные элементы управления на панель *Toolbox*.

Каждый элемент управления – это объект с определенными свойствами, методами и событиями. Как и для формы, их содержащей, можно устанавливать свойства элементов управления программным путем или посредством *Properties Window* редактора VBA. В программе можно присваивать или восстанавливать значения свойств элементов управления так же, как для любых других объектов.

**Использование *Toolbox* (панели элементов)**

Редактор VBA в режиме разработки вместе с формой выводит на экран *Toolbox* (панель элементов) (если вывод этой панели не отключен при помощи меню *View* → *Toolbox*). На рис. 9.11 *Toolbox* показана в виде плавающего окна, кнопки на панели *Toolbox* активизируют различные инструменты, которые позволяют помещать на форму элементы управления.

*Toolbox* (подобно всем другим панелям инструментов) можно настраивать. Если установили дополнительные элементы управления независимых производителей или произвели настройку панели, она может выглядеть иначе, чем та, что показана на рисунке 9.11. На рис. 9.11 присутствуют только стандартные элементы управления, поставляемые вместе с VBA.



Рис. 9.11. Панель Toolbox:

1 – Select Objects (выбор объектов), 2 – Label (надпись), 3 – TextBox (поле), 4 – ComboBox (поле со списком), 5 – ListBox (список), 6 – CheckBox (флажок), 7 – OptionButton (переключатель), 8 – ToggleButton (выключатель), 9 – Frame (рамка) 10 – CommandButton (кнопка), 11 – TabStrip (набор вкладок), 12 – MultlPage (набор страниц), 13 – ScrollBar (полоса прокрутки), 14 – SpinButton (счетчик), 15 – Image (рисунок), 16 –RefEdit

Чтобы воспользоваться панелью *Toolbox* для добавления элементов управления к форме, надо выполнить следующие действия:

1. Щелкнуть на *Toolbox* по кнопке, соответствующей элементу управления, который надо добавить к форме. Указатель мыши изменит форму на перекрестие, когда будет находиться на форме.

2. Установить перекрестие в то место формы, в который нужно поместить верхний левый угол нового элемента управления.

3. Нажать и удерживать левую кнопку мыши.

4. Переместить мышь вниз и вправо, пока элемент управления не примет нужный размер, после чего отпустите кнопку мыши. Редактор VBA вставит в форму элемент управления, и указатель мыши вновь примет форму стрелки.

Можно изменить размер самой формы. Для этого надо щелкнуть по строке заголовка формы для ее выделения и затем, переместиь один из маркеров изменения размеров, увеличить или уменьшить размер формы до желаемого. *Маркеры изменения размеров –*это маленькие квадратики, которые появляются в углах и на серединах сторон графического объекта (наряду с толстой серой границей) при выделении объекта.

**Добавление к форме элементов управления**

Когда создается новая форма, редактор VBA предоставляет чистую (без элементов управления) форму (смотри рис. 9.10). К этой форме можете добавить к форме элементы управления с использованием различных инструментов панели Toolbox.

Чтобы добавить к форме элемент управления надо использовать панель Toolbox, как описано выше. В качестве примера добавления к форме конкретного элемента добавим на новую форму кнопку, выполнив следующие действия:

1. Выберем команду *Insert* → *UserForm*. Редактор VBA добавит новую форму, отображая ее в режиме разработки, и, кроме того на экране появится *Toolbox*.

2. Щелкнуть на элементе CommandButton панели Toolbox (рис. 9.11. позиция 10). Кнопка перейдет в «утопленное» состояние, указывая выбранный элемент.

3. Поместить курсор мыши на форму. Курсор при этом изменит свою форму на перекрестие с прикрепленным к нему символом выбранного элемента.

4. Поместить перекрестие в ту точку формы, в которой должен быть верхний левый угол кнопки.

5. Переместить курсор мыши вниз и вправо, чтобы нарисовать элемент управления *CommandButton*. В процессе перемещения курсора мыши редактор VBA отображает прямоугольный контур, показывающий размер кнопки.

6. Отпустить кнопку мыши, когда *CommandButton* будет иметь нужный размер. Редактор VBA создаст элемент управления *CommandButton* и поместит его на форму (рис. 9.12)



Рис. 9.12. UserForm1 после добавления элемента *CommandButton*

Добавив к форме кнопку, можно заметить, что верхний левый угол кнопки (также как и сама граница кнопки) автоматически выравнивается по шаблону сетки на форме. Это действие, называемое *привязка к сетке*, помогает выровнять размещенные на форме элементы управления и текст.

Все элементы управления формы должны иметь уникальные имена. Эти имена следует использовать при ссылках на элемент управления в своей программе. Всякий раз, когда добавляется к форме новый элемент управления, VBA присваивает ему имя по умолчанию, состоящее из имени типа элемента и номера. VBA обеспечивает уникальность имени включением в имя элемента управления числа. Число в имени больше, чем число любого другого подобного элемента управления. Если кнопка, которую добавили в начале этого раздела, является первой кнопкой, она получит имя *CommandButton*l. Следующая добавленная кнопка будет *CommandButton*2 и так далее. Если после этого добавлялось поле, оно получит имя *TextBox*l.

Все окна программ «солидных» разработчиков имеют общие черты. На рис. 9.13 приведена схема, которой можно придерживаться при разработке своих форм.



Рис. 9.13. Схема типичного диалогового Windows-окна

Наименование окна задается свойством *Caption* (рис. 9.9) формы. Кнопка закрытия окна, если ничего специально не предпринимать по умолчанию всегда будет в окне приложения. Если она не нужна, то лучшеот нее избавиться.

В «полезной» части окна помещаются функционально необходимые элементы управления, которые определяются назначением окна. Здесь могутбыть, например, элементы для ввода некоторой информации или изменения настроек приложения.

С кнопкой **ОК** обычно связывают событие, которое подводит некоторый итог. Обычно в профессиональных приложениях все, что вводится в диалоговом окне, не сразу вступает в действие, а только после того, как пользователь щелкнет кнопку **ОК**. Это дает возможность отказаться от недостаточно обдуманных действий. Разработчику таких окон не очень трудно скопировать данные из диалогового окна в несколько переменных прежде, чем присвоить их каким-либо свойствам элементов управления, а пользователю удобнее работать с окном, из которого всегда можно выйти без последствий.

Назначением кнопки **Отмена** также является выход из процедуры обработки событий формы. Кнопка же **ОК** может разрешить пользователю продолжить путешествие «внутрь» приложения, которое открылось данным окном.

Назначение кнопка **Справка** понятно всем. К сожалению, не так просто организовать помощь, подобную обычной Windows-справке, но можно сначала использовать эту кнопку для простых подсказок, которые создаются простым помещением на пустую форму элемента **Label** с текстом подсказки.

При создании достаточно сложных форм следует руководствоваться рекомендациями фирмы Microsoft. Они имеют отношение к следующим аспектам:

– Размещение элементов управления на форме (в том числе и установка интервалов между элементами).

– Выравнивание меток (метки должны быть хорошими путеводителями формы).

Использование стандартных шрифтов (часто пользователи собирают по всему свету интересные шрифты и забывают о том, что такие шрифты могут отсутствовать у других).

Использование цвета.

При размещении на форме нескольких элементов управления, которые можно условно разделить на отдельные группы, следует использовать элемент *frame* (рамка). При этом желательно пользоваться свойством, определяющим заголовок рамки.

Для более легкого чтения полей, расположенных друг под другом, необходимо выравнивать их по левому краю.

Необходимо стараться поддерживать стандартными высоту и ширину элементов управления, а также расстояния между ними. Интервалы между элементами измеряются в твипах (один пиксель равен 15 твипам). Microsoft рекомендует для высоты элемента управления 300 твипов, для ширины командной кнопки – 1200твипов, для расстояния по вертикали между элементами – 60 твипов между связанными (по смыслу) и 90 твипов между несвязанными элементами.

Для всех элементов управления рекомендуется использовать в качестве шрифта нормальный Sans Serif 8 пунктов. Конечно, следует учитывать и запросы потенциально пользователя продукта, но при подготовке тестового варианта для обсуждения сначала необходимо придерживаться рекомендаций Microsoft.

Самые обычные серые цвета, которые можно видеть во всех известных продуктах Windows, рекомендуются и для разрабатываемых приложений.

В таблице 9.10 приведены значения свойств элементов управления, которые были изменены. Для своих приложений в качестве документации тоже можно создавать такие таблицы для облегчения сопровождения своего программного продукта.

Таблица 9.10.

| **Тип элемента** | **Свойство, которое изменено** | **Значение** | **Примечание** |
| --- | --- | --- | --- |
| UserForm | Name | Frmfirst | Имя формы, на которое можно ссылаться в коде |
| Caption | Тестирование кнопок | Заголовок окна (формы) в верхней части. |
| CommandButton | Name | CmdOK | Имя кнопки, на которое можно ссылаться в коде |
| Caption | OK | Текст на кнопке |
| Default | True | При нажатии на клавишу Enter инициируется событие *Click* кнопки |
| CommandButton | Name | CmdCancel | Имя кнопки, на которое можно ссылаться в коде |
| Caption | Отмена | Текст на кнопке |
| Cancel | True | При нажатии на клавишу Esc инициируется событие *Click* кнопки |
| CommandButton | Name | CmdHelp | Имя кнопки, на которое можно ссылаться в коде |
| Caption | Справка | Текст на кнопке |

В VBA кнопки (элементы CommandButton) имеют специальное свойство *Cancel*, которое определяет кнопку Отмена (независимо от того, называется ли эта кнопка «Отмена»). Только одна кнопка формы может иметь свойство Cancel, установленное равным *True*. Если задать это свойство для одной из кнопок аналогичное свойство для всех остальных кнопок формы будет установлено равным *False*. Если форма содержит кнопку с установленным свойством *Cancel*, то нажатие клавиши Esc приводит к тому же результату, что и щелчок по кнопке. Создадим форму (рис. 9.14).



Рис. 9.14. Простейшее окно программы

Если запустить форму на выполнение, то ни щелчок на кнопке, ни нажатие клавиши Esc, не приведут ни к какому результату, так как с кнопкой не связано никакого события. Но диалоговое окно появится на экране в режиме выполнения и будет создавать впечатление чего-то работающего (рис. 9.14).

Если щелкать кнопки этого окна, то можно убедиться в том, что окно совершенно «нечувствительно» к манипуляциям. Единственная кнопка, которая будет реагировать на действия, находится вверху справа (с изображением перекрестия). Эта кнопка закроет окно.

Чтобы кнопки, помещенные на диалоговой панели, заработали, необ­ходимо для каждой из них написать процедуры обработки событий Click (щелчок на кнопке). Начнем с кнопки **Отмена**, чтобы всегда иметь возмож­ность выйти из приложения. Свяжем с щелчком на кнопке **Отмена** програм­му, которая выгрузит форму, что приведет к окончанию работы с диалоговым окном. Для *создания процедуры обработки* события надо дважды щелкните на кнопке **Отмена** в режиме разработки в появившемся шаблоне процедуры обработки события ввести оператор выгрузки формы, как это представлено в примере 45.

 **Пример 45.** Процедура обработки события – щелчок на кнопке **Отмена**

Private Sub CommandButton2\_Click()

Unload Me ' Выгружаем форму, заканчивая приложение

End Sub

Если снова запустить форму на выполнение, то появится то же диалоговое окно, что и на рис. 9.14, но теперь это окно – настоящее диалоговое, потому что при щелчке на кнопке **Отмена** выполняется программа обработки события и окно закрывается. Более того, при нажатии на клавишу Esc получается точно тот результат. Понятно, что процедура обработки события *Click* (именно это событие и связано с кнопкой) может выполнить и более серьезную работу, чем просто выгрузка формы. Чтобы это было действительно понятно, добавим к процедуре вывод сообщения о том, что возможен выход из диалога (см пример 46).

 **Пример 46.** Процедура обработки события – щелчок на кнопке **Отмена**

Private Sub CommandButton2\_Click()

'объявление переменных

Dim Msg, Title, Response As String

'выдаваемое в окне MsgBox сообщение

Msg = "Хотите закончить работу?"

'состав кнопок и тип значка

Style = vbYesNo + vbCritical + vbDefaultButton2

'заголовок окна

Title = "Выход из программы"

' вызов функции MsgBox с возвращаемым значением

Response = MsgBox(Msg, Style, Title)

'анализ возвращаемого значения

If Response = vbYes Then

Unload Me ' выгрузка формы

End If

End Sub

Если запустите форму с такой процедурой обработки события *Click* и щелкнуть на кнопке **Отмена** или нажать клавишу **Esc**, то на экран будет вы­дано сообщение, подобное приведенному на рис. 9.15. Эта процедура стала более дружественной, поскольку не сразу заканчивает работу окна диалога. Случайный щелчок на кнопке **Отмена** или нажатие на клавишу **Esc** не приведут к окончанию работы с диалогом.



Рис. 9.15. Процедура обработки сообщения от кнопки **Отмена**

После того, как появилась возможность покинуть диалоговое окно, можно написать простые процедуры обработки событий для кнопок **ОК** и **Справка**. Кнопка **ОК** в начале разработки приложения может быть связана с процедурой, выдающей сообщение о том, что приложение пока не готово к эксплуатации. С кнопкой **Справка** можно поступить по-разному: например, одним из решений может быть выдача длинного сообщения посредством функции **MsgBox**, а можно создать новое диалоговое окно с элементом **Label**, свойством **Caption** которого будет также длинное сообщение о полезности приложения.

Очень полезной может оказаться процедура обработки события *Change* – при любом изменении свойств элементов управления (в конечном итоге, все что происходит в диалоговом окне – это изменение тех или иных свойств) выполнять некоторые проверки на допустимость изменений и сообщать пользователю о возможных проблемах. Можно также использовать событие *Change* для подсказки пользователю о происходящих изменениях со свойствами элементов управления. В качестве примера создадим форму **UserForm1** так, чтобы она была похожа на представленную на рис. 9.16. Чтобы соответствующее форме диалоговое окно позволило продолжить работу с приложением, в этом окне необходимо в одном из текстовых полей ввести либо фамилию, либо им пользователя. При изменении содержимого любого поля (свойство **Text**) инициируется процедура обработки события *Change*, в которой свойству *Enable* кнопки **OK\_button** (с заголовком ОК) присваивается значение *True*. Кнопка **OK\_button** предназначена для изменения свойства **Caption** формы **UserForm2** и вывода этой формы на экран. Кнопка с заголовком **Отмена** всегда доступна и предназначена для выхода из программы. Свойство **Cancel** этой кнопки имеет значение **True**.



Рис. 9.16. Форма тестирования события Change

Для создания такой формы выполните следующее:

1. Создать новую форму с именем по умолчанию **UserForm1**. В **Properties Window** изменить свойство **Caption** этого элемента на *Тестирование события Change*.

2. Поместить на форму **UserForm1** элемент *Frame*. По умолчанию его имя будет **Frame1**. Изменить свойство **Caption** этого элемента на *Данные о пользователе*. (Размеры элемента не важны.)

3. Поместить на форму в область **Frame1** элемент *TextBox* (с именем по-умолчанию **TextBox1**) и элемент *Label* с заголовком (свойство **Caption**) **Фамилия**. Имя метки для данного случая не имеет значения.

4. Поместите на форму в область **Frame1** элемент *TextBox* (с именем по-умолчанию **TextBox2**) и элемент *Label* с заголовком (свойство **Caption**) **Имя**.

5. Поместить на форму элемент *CommandButton* и изменить его свойство *Name* на *Cancel\_button*,а свойство **Caption** на **Отмена**. Установить свойство *Cancel* в *True*, чтобы при нажатии на клавишу Esc программа выполняла ту же процедуру, что и при щелчке на кнопку **Отмена**.

6. Поместить на форму элемент *CommandButton* и изменить его свойство *Name* на *OK\_button*, а свойство *Caption* на *ОК*. Изменить свойство *Enabled* этой кнопки на *False*. Этим вы кнопка ОК сделается недоступной для пользователя после выдачи диалогового окна на экран в режиме выполнения. Чтобы кнопка стала доступной, и приложение могло выполнить какую-либо работу с использованием кнопки ОК, пользователь должен будет изменить содержимое одного из текстовых окон.

7. Открыть модуль класса формы **UserForm1** и записать в него код, представленный в примере 47.

8. Создайте форму **UserForm2** (без элементов управления) и запишите в модуль класса этой формы код из примера 44.

| **Тип элемента** | **Свойство, которое изменено** | **Значение** | **Примечание** |
| --- | --- | --- | --- |
| UserForm | Name | UserForm1 | Имя главной формы, на которое можно ссылаться в коде |
| Caption | Тестирование события Change | Заголовок окна (формы) в верхней части |
| Frame | Name | Frame1 | Имя, на которое можно ссылаться в коде |
| Caption | Данныео пользователе |  |
| TextBox | Name | TextBoxl | Имя, на которое можно ссылаться в коде  |
| Label | Name | Label1 |  |
| Caption | Фамилия | Заголовок для текстового окна TextBoxl |
| TextBox | Name | TextBox2 | Имя, на которое можно ссылаться в коде |
| Label | Name | Label2 |  |
| Caption | Имя | Заголовок для текстового окна TextBox2 |
| CommandButton | Name | Cance\_button | Имя кнопки, на которое можно ссылаться в коде |
| Caption | Отмена | Текст на кнопке: кнопка для окончания работы приложения |
| CommandButton | Name | OK\_button |  |
| Caption | OK | Текст на кнопке: кнопка для продолжения работы приложения |
| Enabled | False | Кнопка недоступна для пользователя сразу после выдачи диалогового окна на экран |
| UserForm | Name | UserForm2 | Имя формы, загружаемой из главной при щелчке на кнопке ОК |

Запустим форму на выполнение. Если содержимое текстовых окон не будет редактироваться (изменяться), будет доступна только кнопка **Выход**. Как только будет изменено содержимое какого-либо текстового окна **(Фамилия** или **Имя**), инициируется событие **Change** и свойству *Enabled* кнопки ОК будет присвоено значения *True* (строка 17 примера 47). Так как с кнопкой ОК связана процедура обработки события **Click**, которая загружает Форму **UserForm2**, пользователь получает возможность дальнейшей работы При этом свойству *Caption* формы присваивается сумма содержимого полей **TextBox1** и **TextBox2**.

 **Пример 47.** Обработка события Change

'Обработка события Click кнопки Выход

Private Sub Cancel\_button\_Click()

MsgBox ("Заканчиваем программу!")

Unload Me

End Sub

'Обработка события Click кнопки OK

Private Sub OK\_button\_Click()

UserForm2.Show ' загрузка окна UserForm2

'Изменить заголовок диалогового окна UserForm2

UserForm2.Caption = TextBox2.Text + TextBox1.Text

End Sub

'Обработка события Change текстового окна с меткой Фамилия

Private Sub TextBox1\_Change()

End Sub

'Обработка события Change текстового окна с меткой Имя

Private Sub TextBox2\_Change()

End Sub