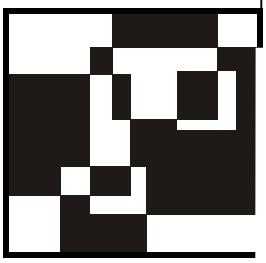
ГЛАВА 5



Формы, элементы управления и события

5.1. Для чего нужны формы

С самыми простыми возможностями организации взаимодействия с пользователем (применение встроенных функций `MsgBox()` и `InputBox()`) мы уже познакомились. Однако, конечно же, возможностей этих функций не всегда хватает. В этой главе речь пойдет о том, как создать графический интерфейс своего приложения с помощью VBA.

Чаще всего для предоставления пользователю графического интерфейса используются формы VBA. В принципе, многие элементы управления можно вставлять непосредственно на страницу документа (для этого используются панели инструментов **Формы** и **Элементы управления**), однако классический способ — это применение формы. Вне зависимости от того, используется ли форма или элементы управления размещаются напрямую в документе, набор элементов управления и приемы работы с ними одинаковы.

Как выглядит применение форм в приложении VBA? Обычно форма запускается при открытии пользователем документа. Пользователь выполняет на форме какие-то действия по вводу или выбору информации (например, выбирает значения в раскрывающемся списке, устанавливает значения для флажков и переключателей и т. п.), а потом, как правило, нажимает кнопку на этой форме, и введенная им информация передается в базу данных, отправляется по электронной почте, записывается в файл для распечатки и т. д.

5.2. Создание форм и самые важные свойства и методы форм

Создать форму очень просто: для этого достаточно в редакторе Visual Basic щелкнуть правой кнопкой мыши на проекте (т. е. на имени документа) в окне

Project Explorer и в контекстном меню выбрать **Insert | UserForm**. Откроется окно дизайнера форм (**Form designer**), в котором будет представлено пустое серое окно формы (по умолчанию она называется **UserForm1**) и рядом **Toolbox** — панель с набором элементов управления (рис. 5.1).

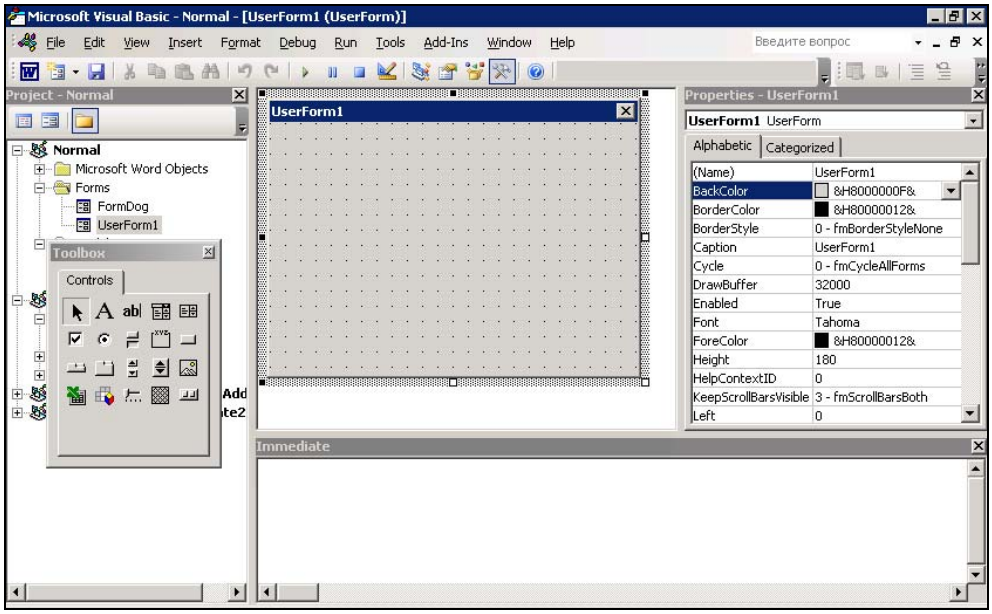


Рис. 5.1. Все готово для работы с формой

Если у вас включен показ окна свойств **Properties** (он включается по клавише <F4>), то в этом окне будут представлены свойства формы. Переход к редактору кода для этой формы (по умолчанию открывается событие `Click`) выполняется по клавише <F7>, возврат обратно в окно дизайнера форм — по <Shift>+<F7>.

Очень удобно, что для форм и элементов управления можно настраивать свойства при помощи графического интерфейса окна свойств — резко уменьшается количество программного кода, которое нужно писать вручную.

Некоторые самые важные свойства форм (кроме `ShowModal` все они применимы и для других элементов управления) приведены далее.

- `Name` — это свойство определяет имя формы. Пользователь вашей программы, скорее всего, его никогда не увидит. Имя формы используется только программистом в коде для этой формы (и в окнах редактора Visual Basic). После создания формы ее имя, предлагаемое по умолчанию (`UserForm1`), рекомендуется заменить на что-нибудь более значимое, чтобы

было проще ориентироваться в программе (это относится ко всем элементам управления).

- `Caption` — определяет заголовок формы (по умолчанию совпадает с именем формы). Рекомендуется ввести строку, которая будет напоминать пользователю о назначении формы (например, `"Выбор типа отчета"`).
- `Enabled` — если это свойство установлено в `False`, пользователь не сможет работать с формой. Используется для временного отключения формы, например, пока пользователь не обеспечит какие-то условия для ее работы.
- `ShowModal` — если свойство установлено в `True` (по умолчанию), то пользователь не может перейти к другим формам или вернуться в документ, пока не закроет эту форму (так называемый "модальный" режим работы).

Большая часть других свойств относится к внешнему виду, размерам и местонахождению формы.

Самые важные методы форм перечислены в следующем списке.

- В процессе редактирования формы (из окна редактора Visual Basic) ее можно запускать по нажатию клавиши <F5>. После того, как форма будет готова, вы должны обеспечить ее запуск в документе. Для запуска формы нужно воспользоваться методом `Show()`:

  ```
  UserForm1.Show
  ```

  Если форма уже была загружена в память, она просто станет видимой, если нет — то будет автоматически загружена (произойдет событие `Load`).

  Сам этот метод можно вызвать, например:

  - из обычного макроса, привязанного к кнопке или клавиатурной комбинации;
  - из автозапускаемого макроса (макроса с названием `AutoExec` для Word);
  - из кода для элемента управления, расположенного в самом документе (например, `CommandButton`) или на другой форме (для перехода между формами);
  - поместить его в обработчик события `Open` для документа Word или книги Excel, чтобы форма открывалась автоматически при открытии документа.
- После того, как пользователь введет или выберет нужные данные на форме и нажмет требуемую кнопку, форму необходимо убрать. Для этого можно воспользоваться двумя способами:

  - спрятать форму (использовать метод `Hide()`), например:

    ```
    UserForm1.Hide
    ```

Форма будет убрана с экрана, но останется в памяти. Потом при помощи метода `Show()` можно будет опять ее вызвать в том же состоянии, в каком она была на момент "прятанья", а можно, например, пока она спрятана, программно изменять ее и расположенные на ней элементы управления. Окончательно форма удалится из памяти при закрытии документа;

- если форма больше точно не потребуется, можно ее удалить из памяти при помощи команды `Unload`:

```
Unload UserForm1
```

Остальные методы относятся либо к обмену данными через буфер обмена (`Copy()`, `Cut()`, `Paste()`), либо к служебным возможностям формы (`PrintForm()`, `Repaint()`, `Scroll()`).

Важнейшая концепция VBA — события. Событие *(event)* — это то, что происходит с программой и может быть ею распознано. Например, к событиям относятся щелчки мышью, нажатия на клавиши, открытие и закрытие форм, перемещение формы по экрану и т. п. VBA построен таким образом, чтобы можно было создавать на нем программы, управляемые событиями *(event-driven)*. Такие программы противопоставляются устаревшему процедурному программированию.

Самые важные события форм приведены далее.

- `Initialize` — происходит при подготовке формы к открытию (появлению перед пользователем). Обычно в обработчик для этого события помещается код, связанный с открытием соединений с базой данных, настройкой элементов управления на форме, присвоением значений по умолчанию и т. п.
- `Click` (выбирается по умолчанию) и `DblClick` — реакция на одиночный и двойной щелчок мыши соответственно. Для формы эти события используются не так часто. Обычно обработчики щелчков применяются для кнопок (элементов управления `CommandButton`).
- `Error` — это событие используется при возникновении ошибки в форме, предоставляя пользователю возможность исправить сделанную им ошибку. Подробнее — в *гл. 6*, которая посвящена ошибкам и отладке.
- `Terminate` — используется при нормальном завершении работы формы и выгрузке ее из памяти (например, по команде `Unload`). Обычно применяется для разрыва открытых соединений с базой данных, освобождения ресурсов, протоколирования и т. п. Если работа формы завершается аварийно (например, запустившее форму приложение выдало команду `End`), то это событие не возникает.

Остальные события связаны либо с изменением размера окна формы, либо с нажатиями клавиш, либо с активизацией (получением фокуса) или деактивизацией (потерей фокуса).

Поскольку форма — это во многом просто контейнер для хранения других элементов управления, главное ее событие — `Initialize`. Все остальные события обычно используются не для формы, а для расположенных на ней элементов управления.

Нужно отметить некоторые моменты, связанные с созданием и редактированием форм:

- формы, создаваемые в Microsoft Access, не являются стандартными, как формы остальных приложений Office, и набор свойств и методов у них несколько отличается. Тем не менее по функциональности они практически одинаковы;
- иногда для обсуждения форму удобно распечатать. Для этого предусмотрено специальное диалоговое окно, которое можно вызвать по нажатию клавиш <Ctrl>+<P> (при выбранной форме в дизайнере);
- если все нужные вам элементы управления трудно уместить на одной форме (даже большого размера), в вашем распоряжении два варианта: воспользоваться двумя формами (осуществляя переход между ними при помощи методов `Show()` и `Hide()`, подвязанных к элементам управления) или воспользоваться несколькими вкладками для формы. Для этой цели в вашем распоряжении — специальный элемент управления `Multipage`.

5.3. Элементы управления

5.3.1. Что такое элемент управления

Элемент управления — это специализированный объект, который можно размещать на формах VBA (или непосредственно в документах) и который используется для организации взаимодействия с пользователем. В VBA есть как стандартные элементы управления (`CommandButton`, `CheckBox`, `OptionButton`), так и нестандартные (любые другие, которые есть на вашем компьютере, например, Microsoft Web Browser, представляющий Internet Explorer, элемент управления `Calendar` и т. п.). Элементы управления реагируют на события, которые генерирует пользователь (нажатие на кнопку, ввод значения, перемещение ползунка и т. п.).

Добавление элементов управления на форму чаще всего производится из дизайнера форм при помощи панели **Toolbox**. Для этого необходимо выбрать элемент управления на **Toolbox** и перетащить его на форму или, что более

удобно, выделить элемент управления в **Toolbox**, а затем на форме выделить ту область экрана, которую будет занимать этот элемент управления.

Добавление элементов управления можно производить и программным способом (при помощи метода `Add()` коллекции `Controls`), однако при этом вам придется указывать огромное количество свойств создаваемого элемента управления, что не очень удобно.

5.3.2. Элемент управления *Label*

Это самый простой элемент управления. *Надпись* (`Label`) — это просто область формы, в которой выводится какой-то текст (рис. 5.2).

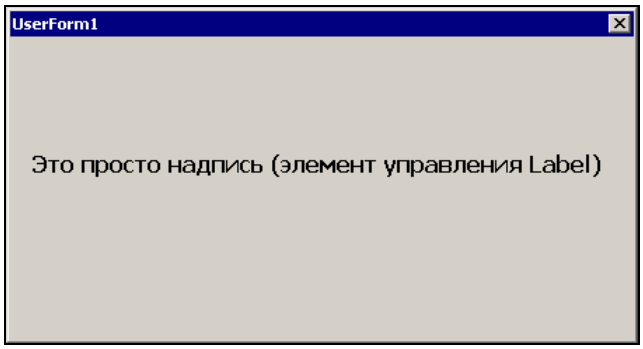


Рис. 5.2. Элемент управления `Label` на форме

Пользователь не может изменять этот текст. Чаще всего элемент управления `Label` используется как строка состояния с объяснением того, что сейчас произошло, или происходит, или должен сделать пользователь, и т. п. Этот элемент управления может использоваться и как пояснение для других элементов управления, таких как ползунок.

Главное свойство элемента управления `Label` — это `Caption`, тот текст, который будет выводиться на форме. Большая часть остальных свойств относится к форматированию этого текста или настройке внешнего вида этого элемента управления.

Несмотря на то, что для этого элемента управления предусмотрен набор событий (`Click`, `Error` и т. п.), использовать их не принято: пользователю обычно не приходит в голову, что по надписи нужно щелкнуть мышью.

5.3.3. Элемент управления *TextBox*

Текстовое поле (`TextBox`) — один из самых часто используемых элементов управления (рис. 5.3).

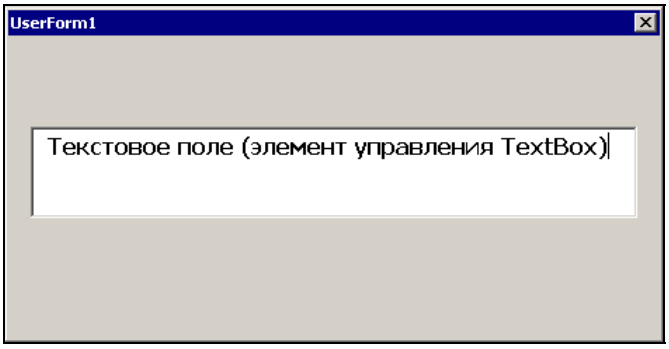


Рис. 5.3. Текстовое поле (элемент управления `TextBox`) на форме

Текстовое поле используется:

- для приема каких-либо текстовых данных, вводимых пользователем (например, для отправки по почте, для занесения в базу данных и т. п.);
- для вывода пользователю текстовых данных с возможностью их редактирования (из базы данных, листа Excel и т. п.);
- для вывода пользователю текстовых данных с возможностью копирования и печати, но без возможности изменения (классический пример — текст лицензионного соглашения).

Далее приведены некоторые важные свойства этого элемента управления.

- `Value` (или `Text`, эти два свойства для текстового поля идентичны) — то текстовое значение, которое содержится в этом поле. Используется для занесения исходного значения и для приема значения, введенного пользователем, в строковую переменную.
- `AutoSize` — позволяет текстовому полю автоматически менять свой размер, чтобы поместить весь текст. Использовать не рекомендуется, т. к. может нарушиться весь дизайн вашей формы.
- `ControlSource` — ссылка на источник текстовых данных для поля. Может ссылаться, например, на ячейку в Excel, на поле в объекте `Recordset` и т. п. При изменении пользователем данных в текстовом поле автоматически изменится значение на источнике, определенном в `ControlSource`.
- `ControlTipText` — текст всплывающей подсказки, которая появляется, когда пользователь наводит указатель мыши на элемент управления. Рекомендуется к заполнению для всех элементов управления (для самой формы это свойство не предусмотрено).
- `Enabled` — если установить в `False`, то текст в поле станет серым и с содержимым поля ничего нельзя будет сделать (ни ввести текст, ни выде-

лить, ни удалить). Обычно это свойство используется, чтобы показать пользователю, что этот элемент управления отключен до выполнения каких-либо условий (это относится ко всем элементам управления).

- `Locked` — поле будет выглядеть как обычно, пользователь сможет выделять и копировать данные из него, но не изменять их. Используется для показа неизменяемых данных типа лицензионных соглашений, сгенерированных значений и т. п.
- `MaxLength` — максимальная длина значения, которое можно ввести в поле. Иногда можно использовать свойство `AutoTab` — при достижении определенного количества символов управление автоматически передается другому элементу управления.
- `MultiLine` — определяет, можно ли использовать в текстовом поле несколько строк или только одну. Если вам нужно текстовое поле для приема одного короткого значения, подумайте, нельзя ли вместо элемента управления обойтись функцией `InputBox()`.
- `PasswordChar` — позволяет указать, за каким символом будут "прятаться" вводимые пользователем значения. Используется, конечно, при вводе пароля.
- `ScrollBars` — определяет, будут ли показаны горизонтальная и вертикальная полосы прокрутки (в любом сочетании). Если текст будет длинным, без них не обойтись.
- `WordWrap` — настоятельно рекомендуется включать в тех ситуациях, когда значение `MultiLine` установлено в `True`. В этом случае будет производиться автоматический переход на новую строку при достижении границы текстового поля.

Остальные свойства по большей части относятся к оформлению текстового поля и его содержания, а также к настройкам редактирования.

Главное событие для текстового поля — это событие `Change` (т. е. изменение содержания поля). Обычно на это событие привязывается проверка вводимых пользователем значений или синхронизация введенного значения с другими элементами управления (например, сделать доступной кнопку, изменить текст надписи и т. п.)

5.3.4. Элемент управления *ComboBox*

Комбинированный список (`ComboBox`) также используется очень часто. Этот элемент управления позволяет пользователю выбирать "готовые" значения из списка, так и вводить значения самостоятельно (хотя это можно запретить). Пример элемента управления `ComboBox` представлен на рис. 5.4.

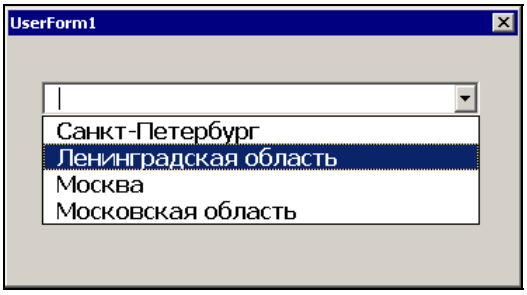


Рис. 5.4. Комбинированный список (элемент управления `ComboBox`) на форме

Обычно `ComboBox` используется в двух ситуациях:

- когда пользователю необходимо выбрать одно или несколько значений из списка размером от 4-х до нескольких десятков позиций. Если позиций меньше, то проще использовать переключатель (`OptionButton`), если больше — то ориентироваться в списке становится неудобно и необходимо использовать специальные приемы, когда пользователь вводит первые буквы нужного слова и в списке остаются только значения, которые начинаются с этих букв;
- когда список позиций для выбора необходимо формировать динамически на основании данных из источника (базы данных, листа Excel и т. п.).

К сожалению, через окно свойств заполнить список позициями не получится — для этой цели придется использовать специальный метод `AddItem()`. Обычно он помещается в обработчик события `Initialize` для формы. Применение его может выглядеть так:

```
Private Sub UserForm_Initialize()
    ComboBox1.AddItem "Санкт-Петербург"
    ComboBox1.AddItem "Ленинградская область"
    ComboBox1.AddItem "Москва"
    ComboBox1.AddItem "Московская область"
End Sub
```

Второй параметр `varIndex` (необязательный) этого метода может использоваться для определения положения элемента в списке, но он не может превышать значения свойства `ListCount` и поэтому для начальной загрузки `ComboBox` не подходит.

Самые важные свойства комбинированного списка представлены далее.

- `ColumnCount`, `ColumnWidth`, `BoundColumn`, `ColumnHeads`, `RowSource` — свойства, которые применяются при работе со списками из нескольких столбцов. Пользователи не любят такие списки, и поэтому к использованию они

не рекомендуются (гораздо проще сделать несколько комбинированных списков).

- `MatchEntry` — определяет, будут ли при вводе пользователем первых символов значения выбираться подходящие позиции из списка. Возможность очень удобная, рекомендуется сохранить значение, которое используется по умолчанию.
- `MatchRequired` — определяет, разрешается ли пользователю вводить то значение, которого нет в списке. По умолчанию `False`, т. е. разрешено.
- `Value` (или `Text`) — позволяет программным способом установить выбранное значение в списке или получить в переменную выбранное или введенное пользователем значение.

Остальные свойства (`AutoSize`, `Enabled`, `Locked`, `ControlText`, `ControlTipText`, `MaxLength`) применяются точно так же, как и для `TextBox`.

Главное событие для комбинированного списка — `Change`, то же, что и для `TextBox`. Обычно в обработчике этого события проверяются введенные пользователем значения, эти значения переносятся в текстовое поле или в `ListBox` (если нужно дать пользователю возможность выбрать несколько значений, поскольку свойства `MultiSelect` у `ComboBox` нет) и т. п.

5.3.5. Элемент управления *ListBox*

Элемент управления `ListBox` очень похож на комбинированный список, но применяется гораздо реже по двум причинам:

- в нем нельзя открыть список значений по кнопке. Все значения видны сразу в поле, аналогичном текстовому, и поэтому большое количество позиций в нем уместить трудно;
- пользователь не может вводить свои значения — только выбирать из готовых.

Пример этого элемента управления представлен на рис. 5.5.

Но у этого элемента управления есть и преимущества: в нем пользователь может выбирать не одно значение, как в `ComboBox`, а несколько.

Обычно `ListBox` используется:

- как промежуточное средство отображения введенных или выбранных пользователем через `ComboBox` значений (или любых других списков, например, для списка выбранных файлов);
- как средство редактирования списка значений, сформированных вышеуказанным образом или полученных из базы данных (для этого можно рядом с `ListBox` разместить кнопки **Удалить** или **Изменить**).

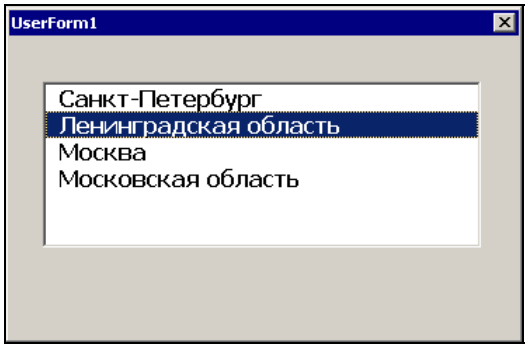


Рис. 5.5. Список (элемент управления `ListBox`) на форме

Основные свойства, методы и события у `ListBox` — те же, что и у `ComboBox`. Главное отличие — это свойство `MultiSelect`, которое позволяет пользователю выбирать несколько значений. По умолчанию это свойство отключено.

5.3.6. Элементы управления *CheckBox* и *ToggleButton*

Флажки (`CheckBox`) (пользователи часто называют их "галками" или "птичками") и *кнопки с фиксацией* (`ToggleButton`) используются для выбора невзаимоисключающих вариантов (если этих вариантов немного). Они представлены на рис. 5.6.

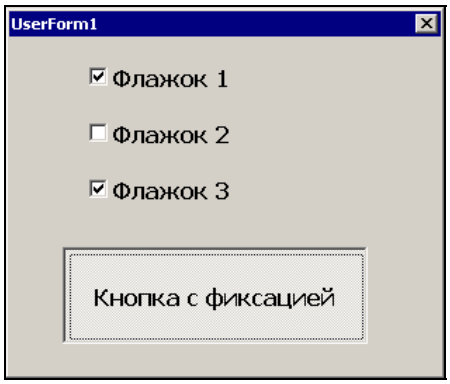


Рис. 5.6. Флажки (элементы управления `CheckBox`) и кнопки с фиксацией (`ToggleButton`)

Для `CheckBox` предусмотрено три главных свойства.

- `Caption` — надпись справа от флажка, которая объясняет, что выбирается этим флажком.

- TripleState — если это свойство установлено в False (по умолчанию), то флажок может принимать только два состояния: установлен или нет. Если для TripleState установить значение True, то появляется третье значение Null, когда установлен "серый" флажок. Такое значение часто используется, например, при выборе компонентов программы при установке, когда выбраны не все компоненты, а лишь некоторые.
- Value — само состояние флажка. Может принимать значения True (флажок установлен), False (снят) и Null — "серый" флажок (когда свойство TripleState установлено в True).

Главное событие элемента CheckBox — Change.

ToggleButton выглядит как кнопка, которая после щелчка на ней остается "нажатой" (рис. 5.6), а при повторном щелчке отключается. У нее могут быть те же два (или три, в соответствии со свойством TripleState) состояния, что и у CheckBox. Свойства и методы — те же самые. Единственное отличие — в восприятии их пользователем. Обычно ToggleButton воспринимается пользователем как переход в какой-то режим или начало выполнения продолжительного действия.

5.3.7. Элементы управления *OptionButton* и *Frame*

Если CheckBox предназначен для выбора невзаимоисключающих вариантов, то *переключатель* (OptionButton) используется как раз для выбора варианта в ситуации "или/или" (рис. 5.7).

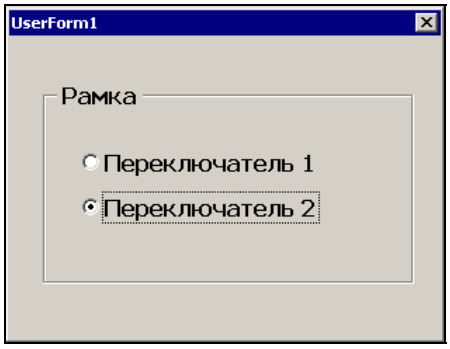


Рис. 5.7. Переключатели (2 объекта OptionButton) в рамке (объект Frame)

Классический пример, при помощи которого можно проиллюстрировать работу OptionButton, — выбор радиостанции на радиоприемнике: сразу две радиостанции слушать нельзя (поэтому иногда этот элемент управления называют RadioButton).

Главных свойств у этого элемента управления два.

- `Caption` — надпись для переключателя.
- `Value` — установлен переключатель или нет (только два состояния — `True` или `False`).

Главное событие тоже стандартное — `Change`.

Конечно, использовать один переключатель бессмысленно. Выбор должен предоставляться хотя бы из двух вариантов, и при выборе одного из них другой автоматически снимается. Однако в некоторых ситуациях нам необходимо выбрать из нескольких наборов вариантов (например, отчет за месяц/квартал/год, тип отчета, нужный филиал и т. п.). Решение простое — переключатели нужно сгруппировать.

Самый простой вариант группировки — просто использовать новую форму или вкладку на форме. Если переключатели находятся на одной форме (или на одной вкладке), они автоматически считаются взаимоисключающими. Если же нужно более точно выбрать группы, то необходимо использовать элемент управления `Frame`.

`Frame` — это просто рамка, которая выделяет прямоугольную область на форме и позволяет организовать элементы управления (рис. 5.7). Помещенные внутрь рамки переключатели считаются взаимоисключающими, остальные элементы управления ведут себя точно так же, хотя иногда бывает полезно с точки зрения наглядности свести вместе под одной рамкой, например, набор флажков. При желании рамку можно сделать невидимой, установив для свойства `BorderStyle` значение 1 и убрав значение свойства `Caption`.

5.3.8. Элемент управления *CommandButton*

Элемент управления `CommandButton` *(кнопка)* — самый распространенный элемент управления на формах (рис. 5.8).

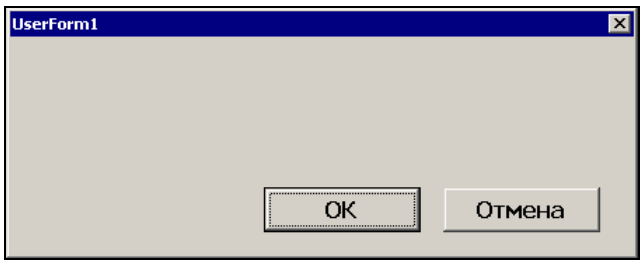


Рис. 5.8. Кнопки (объекты `CommandButton`)

В большинстве форм обязательно будет, по крайней мере, две кнопки: **OK** и **Отмена** (Cancel). По нажатию кнопки **OK** должно выполниться то действие,

ради чего создавалась форма, по нажатию кнопки **Отмена** форма должна закрыться. Ваша задача — обеспечить необходимый код для этих кнопок, который и будет выполнять эти действия.

Далее представлены самые важные свойства кнопки.

- `Cancel` — если для этого свойства установить значение `True`, то кнопка будет нажиматься автоматически при нажатии клавиши <Esc>. Как правило, на такие кнопки помещаются надписи типа "Отмена", "Выход", "Вернуться в окно приложения". Однако, кроме назначения клавише <Esc>, свойство ничего больше этой кнопке не дает. Необходимо еще добавить код в обработчик события `Click`, например:

```
Private Sub CommandButton1_Click()
    Unload Me
End Sub
```

  `Me` — это специальное зарезервированное слово, которое представляет текущий объект (в данном случае форму). Его можно использовать вместо имени формы.

- `Caption` — надпись, которая будет на кнопке.
- `Default` — если это свойство установлено в `True`, то такая кнопка будет считаться нажатой при нажатии пользователем клавиши <Enter>, даже если фокус находился в другом месте формы (но не на другой кнопке). Обычно такие кнопки являются главными, по которым выполняется действие, ради которого создавалась форма (печать отчета, занесение информации в базу данных, отправка почты и т. п.).
- `Picture` — если простая надпись вас не устраивает, можно назначить кнопке рисунок (пиктограмму).
- `TakeFocusOnClick` — определяет, будет ли передаваться управление этой кнопке при нажатии на нее. По умолчанию установлено `True`.

Главное событие для кнопки — это, конечно, `Click`. Как правило, к этому событию и привязывается программный код, ради которого создавалась кнопка.

5.3.9. Элементы управления *ScrollBar* и *SpinButton*

Полосы прокрутки (`ScrollBars`) чаще всего встречаются в текстовых полях, когда введенный текст полностью на экране не помещается. Однако ничего не мешает вам использовать `ScrollBar` как отдельный элемент управления (пользователи часто называют его "ползунок") для выбора какого-то значения из диапазона (рис. 5.9). Обычно такой элемент управления используется для выбора плавно меняющихся значений, например: уровня громкости, яркости, сжатия, приоритета и т. п.

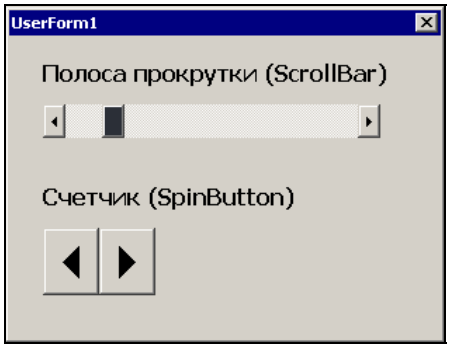


Рис. 5.9. Полоса прокрутки (`ScrollBar`) и счетчик (`SpinButton`)

Главное событие для `ScrollBar` — уже знакомое нам `Change`. Главные свойства этого элемента управления представлены далее.

- `Max` и `Min` — максимальное и минимальное значения, которые можно задать при помощи этого элемента управления. Возможный диапазон — от –32 767 до +32 767. При этом максимальное значение может быть и меньше минимального — просто ползунок придется тянуть в обратную сторону.
- `LargeChange` и `SmallChange` — определяют, какими шагами будет двигаться ползунок при перемещении его пользователем (путем щелчка на полосе около ползунка или при нажатии на одну из кнопок направления соответственно).
- `Orientation` — определяет расположение ползунка (вертикальное или горизонтальное). По умолчанию для этого свойства установлено значение 1, т. е. ориентация определяется автоматически в зависимости от конфигурации отведенного элементу управления пространства на форме (что больше — длина или высота). Однако при помощи этого свойства можно и явно указать вертикальное или горизонтальное расположение ползунка.
- `ProportionalThumb` — определяет размер ползунка: будет ли он пропорционален размеру полосы прокрутки (по умолчанию) или будет фиксированного размера.
- `Value` — главное свойство этого элемента управления. Определяет положение ползунка и то значение, которое будет возвращать этот элемент управления программе.

Как правило, использование ползунка без отображения выбранной при помощи его информации не очень приветствуется пользователями. В самом простом варианте то, что выбрано при помощи ползунка, следует просто отображать в текстовой надписи:

```
Private Sub ScrollBar1_Change()
    Label1.Caption = ScrollBar1.Value
End Sub
```

В более сложном варианте пользователю можно выбирать — использовать ли ползунок или вводить значение в текстовом поле. В этом случае в событии `Change` для текстового поля необходимо предусмотреть проверку вводимых пользователем значений и обратную связь с ползунком.

Элемент управления *счетчик* (`SpinButton`) — эта та же полоса прокрутки, лишенная самой полосы и ползунка (рис. 5.9). `SpinButton` используется в тех ситуациях, когда диапазон выбираемых значений совсем небольшой (например, надо выбрать количество копий для печати отчета). Все свойства, которые есть у `SpinButton`, совпадают со свойствами `ScrollBar`.

5.3.10. Элементы управления *TabStrip* и *MultiPage*

Набор вкладок (`TabStrip`) и *набор страниц* (`MultiPage`) применяются в одной и той же ситуации — когда элементов управления слишком много, чтобы уместить их на одной странице формы. Эти элементы управления позволяют создавать на форме несколько вкладок (страниц), между которыми сможет переходить пользователь. Принципиальное отличие между этими элементами управления заключается в том, что на вкладках `TabStrip` всегда располагаются одинаковые элементы управления, а на `MultiPage` — разные. Применение множества вкладок вы наверняка видели во многих программах (например, в Word в окне **Параметры**, открывающемся с помощью меню **Сервис | Параметры**). Пример использования элемента управления `MultiPage` представлен на рис. 5.10.

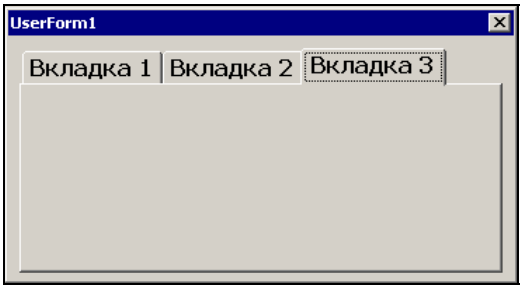


Рис. 5.10. Форма с несколькими вкладками (элементами управления `MultiPage`)

Элемент `TabStrip` используется реже. Например, его можно применить для занесения данных по одному шаблону для филиалов или сотрудников (если их не слишком много).

Свойства и события у этих элементов управления практически идентичны. Чаще всего используются следующие свойства.

- `MultiRow` — определяет, можно ли использовать несколько горизонтальных рядов вкладок.
- `TabOrientation` — определяет, где будут расположены заголовки вкладок (по умолчанию вверху).
- `Value` — номер вкладки, которая открыта в настоящий момент (нумерация начинается с 0).

Главное событие этих элементов управления — `Change` (т. е. переход между вкладками). К нему можно привязать, например, проверку уже введенных пользователем значений или вывод предупреждений.

5.3.11. Элемент управления *Image*

Наверное, *рисунок* (`Image`) — это самый простой из элементов управления. Он позволяет отобразить на форме рисунок в одном из распространенных форматов, который будет реагировать на щелчок мышью (а может просто использоваться для украшения формы). Отметим некоторые моменты, связанные с применением элемента управления `Image`:

- в качестве альтернативы можно использовать свойство `Picture` формы (особенно если вам нужен фоновый рисунок для всей формы);
- еще две альтернативы — это свойство `Picture` элементов управления `Label` или `CommandButton`. Функциональность рисунков получается практически одинаковая;
- при использовании этого элемента управления само изображение копируется внутрь документа и внешний его файл больше не нужен.

Главные свойства этого элемента управления представлены далее.

- `Picture` — позволяет выбрать само изображение для формы.
- `PictureAlignment` — позволяет выбрать местонахождение изображения в отведенной ему области. По умолчанию рисунок располагается по центру.
- `PictureSizeMode` — позволяет выбрать режим растяжения или уменьшения элемента в случае, если он не соответствует размеру области.
- `PictureTiling` — определяет, размножать ли маленький рисунок, чтобы он покрыл всю отведенную ему область ("замостить").

Главное событие элемента управления `Image` — `Click`.

5.3.12. Применение дополнительных элементов управления

Мы рассмотрели стандартные элементы управления, которые изначально помещены на панель **ToolBox** и доступны для размещения в формах. Однако возможности форм VBA этим не ограничиваются. В вашем распоряжении — сотни и тысячи элементов управления, встроенных в Windows, в другие продукты или поставляемые отдельно (в том числе третьими фирмами). Для того чтобы можно было разместить их на форме, щелкните правой кнопкой мыши по пустому пространству в **ToolBox**, выберите пункт контекстного меню **Additional Controls**, а затем в списке выберите нужный элемент. Правда, при использовании нестандартных элементов управления необходимо помнить, что при переносе программы (файла Office) на другой компьютер вам потребуется обеспечить на нем наличие необходимых библиотек.

Очень часто в программах используются дополнительные элементы управления Internet Explorer, Acrobat Reader, календарь, проигрыватели аудио- и видеофайлов и т. п. Например, чтобы разместить на форме элемент управления Microsoft Web Browser (в русифицированной версии Windows он называется **Обозреватель веб-страниц (Microsoft)**), который представляет окно Internet Explorer, нужно выполнить следующие действия:

- щелкнуть правой кнопкой мыши по пустому пространству в окне **Toolbox** и в контекстном меню выбрать **Additional Controls**;
- в открывшемся списке выбрать **Microsoft Web Browser** (или **Обозреватель веб-страниц (Microsoft)**);
- изменившимся курсором мыши очертить на форме ту область, которую будет занимать этот элемент управления.

Далее нужно позаботиться о программном коде для этого элемента управления. Созданный нами на форме элемент управления по умолчанию будет называться `WebBrowser1`. Можно выбрать любое из доступных событий этого элемента управления, а можно использовать этот элемент управления и в событиях других объектов. Например, если нам нужно, чтобы при открытии формы в окне Internet Explorer на ней открывалась определенная страница, можно воспользоваться событием `Initialize` для формы. Соответствующий код может быть таким:

```
Private Sub UserForm_Initialize()
    WebBrowser1.Navigate "http://www.AskIt.ru"
End Sub
```

Преимущества использования этого элемента управления очевидны — вы можете расширить функциональность своей формы за счет использования

Web-страниц (например, с формами HTML). Internet Explorer обычно установлен на любом компьютере под управлением Windows и поэтому с этим элементом управления не возникает никаких проблем при переносе программы на другой компьютер. Справку по этому элементу управления придется смотреть в MSDN.

Еще один часто используемый элемент управления, который есть практически на всех компьютерах — `Calendar` *(календарь)* (рис. 5.11). В зависимости от версии вашей операционной системы и установленного программного обеспечения он может называться по-разному, у меня на компьютере он называется **Calendar Control 8.0**. При помощи этого элемента управления пользователю будет очень удобно выбирать нужную дату.

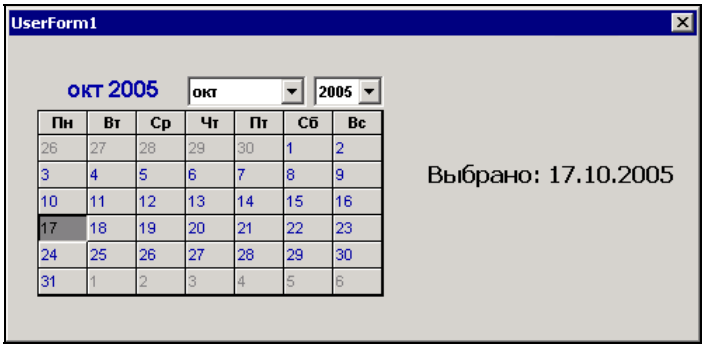


Рис. 5.11. Элемент управления `Calendar` и надпись, дублирующая значение, выбранное пользователем в `Calendar`

Главное свойство этого элемента управления — `Value`, т. е. та дата, которая выбрана пользователем. Остальные свойства предназначены для отображения внешнего вида календаря.

В Excel на панели **ToolBox** имеется еще один специфический элемент управления — `RefEdit` (в списке **Additional Controls** он называется как **RefEdit.Ctrl**). Он похож на текстовое поле с кнопкой справа. При нажатии на эту кнопку форма, на которой размещен этот элемент управления, "спрячется", а пользователю будет предоставлена возможность выбрать одну ячейку или диапазон ячеек Excel. После того как пользователь завершит выбор, он опять вернется в окно формы, а в `RefEdit` будет помещена информация об адресе выбранного диапазона. Такой же адрес, конечно, можно вводить и вручную. Главное свойство этого элемента управления — `Value`.

Большое количество дополнительных элементов управления предусмотрено для форм Access. Они являются специфическими для Access, и про них будет рассказано в *гл. 12*.

Задание для самостоятельной работы 5: Работа с элементами управления

Подготовка:

1. Создайте новую книгу Excel и сохраните ее как Prikaz.xls. Заполните ячейки с A1 по A5 значениями, аналогичными представленным на рис. 5.12. Данные о сотрудниках лучше ввести в родительном падеже, поскольку эти значения будут подставляться в автоматически создаваемый приказ в формате документа Word.

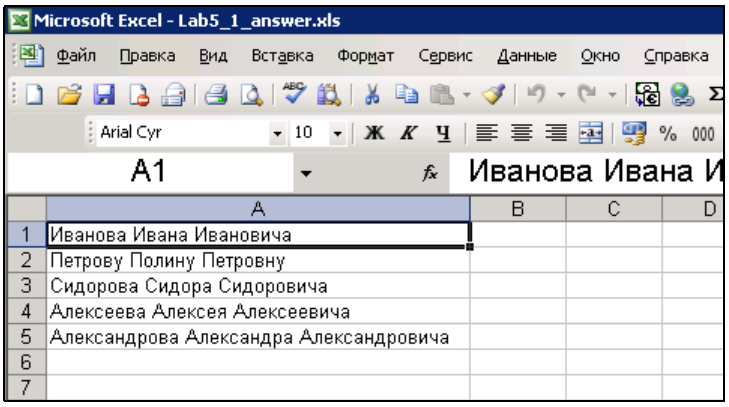


Рис. 5.12. Список сотрудников на листе Excel

2. Откройте редактор Visual Basic и в окне **Project Explorer** щелкните правой кнопкой мыши по объекту **Эта книга** и в контекстном меню выберите **View Code**.
3. В окне редактора кода для этой книги введите следующий код:

```
'При открытии рабочей книги показываем форму UF1
Private Sub Workbook_Open()
    UF1.Show
End Sub

'Специальная процедура, которая печатает приказ в Word
Public Sub DocWrite(sPovod As String, sFio As String, bFlagPremia As
Boolean, bFlagGramota As Boolean, nSummaPremii As Long, sOtvIsp As
String)

    Dim oWord As Word.Application
    Dim oDoc As Word.Document
    Set oWord = CreateObject("Word.Application")
```

```
    Set oDoc = oWord.Documents.Add()
    oWord.Visible = True
    oDoc.Activate
    With oWord.Selection
        .TypeText "Приказ"
        .Style = "Заголовок 1"
        .ParagraphFormat.Alignment = wdAlignParagraphCenter
        .TypeText vbCrLf
        .Style = "Обычный"
        .TypeText vbCrLf
        .TypeText "г.Санкт-Петербург" & Space(90) & Date
        .TypeText vbCrLf
        .TypeText vbCrLf
        .TypeText "За проявленные успехи в " & sPovod & _
                  " наградить " & sFio & ":"
        .TypeText vbCrLf
        If bFlagPremia Then
            .TypeText vbTab & "- денежной премией в сумме " & _
                      nSummaPremii & " рублей"

        End If
        If bFlagGramota Then
            .TypeText vbCrLf
            .TypeText vbTab & "- почетной грамотой."
        Else
            .TypeText "."
        End If
        .TypeText vbCrLf
        .TypeText vbCrLf
        .TypeText vbCrLf
        .TypeText vbCrLf
        .TypeText "Генеральный директор" & vbTab & vbTab & _
                  vbTab & "Иванов И. И."
        .ParagraphFormat.Alignment = wdAlignParagraphCenter
        .TypeParagraph
        .TypeText vbCrLf
        .TypeText vbCrLf
        .ParagraphFormat.Alignment = wdAlignParagraphLeft
        .TypeText Text:=("Отв. исполнитель " & sOtvIsp)
        .TypeParagraph

    End With
End Sub
```

4. В окне **Project Explorer** щелкните правой кнопкой мыши по проекту Prikaz.xls и в контекстном меню выберите **Insert | UserForm**. Выделите созданный вами объект формы и нажмите клавишу <F4>. В окне **Properties** введите для свойства `Name` этой формы значение `UF1`.

   Поместите на форму из **Toolbox** единственную кнопку — элемент управления `CommandButton1`. Установите для этой кнопки значение свойства `Caption` как `"Напечатать приказ"` (без кавычек) и измените размеры и местонахождение этой кнопки, чтобы форма выглядела так, как показано на рис. 5.13.

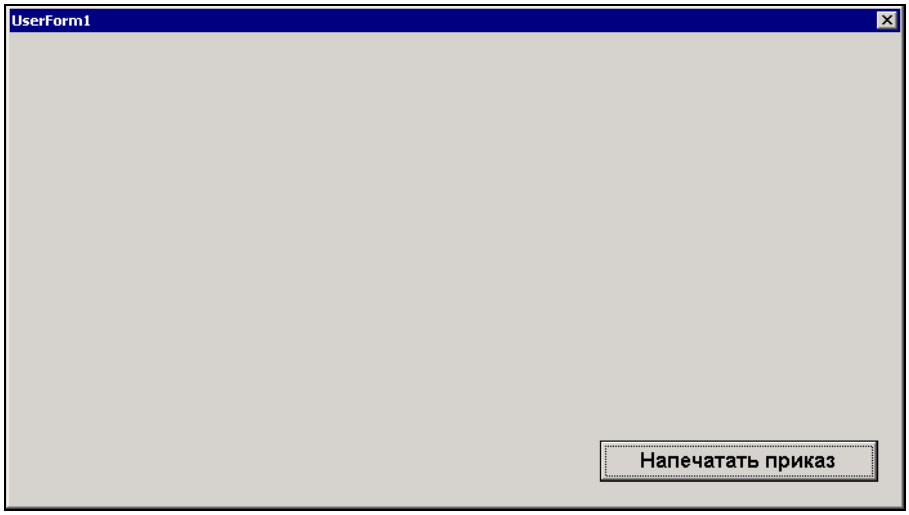


Рис. 5.13. Заготовка для формы с единственной кнопкой

5. Щелкните правой кнопкой мыши по кнопке `CommandButton1` на вашей форме, в контекстном меню выберите **View Code** и добавьте в код событийной процедуры для события `Click` этой кнопки следующий код:

```
Private Sub CommandButton1_Click()
    Dim sPovod As String
    Dim sFio As String
    Dim bFlagPremia As Boolean
    Dim bFlagGramota As Boolean
    Dim nSummaPremii As Long
    Dim sOtvIsp As String

    'Подставить данные из формы
    sPovod = "освоении новых информационных технологий"
    sFio = "Иванова Ивана Ивановича"
```

```
    bFlagPremia = True
    bFlagGramota = True
    nSummaPremii = 100000
    sOtvIsp = "Петрова П. П."
    'Конец подстановки данных

    Call ЭтаКнига.DocWrite(sPovod, sFio, bFlagPremia, bFlagGramota, _
        nSummaPremii, sOtvIsp)
End Sub
```

6. Запустите вашу форму на выполнение и убедитесь, что она работает: выводит в создаваемый документ Word приказ с фиксированными значениями.

ЗАДАНИЕ:

Измените форму таким образом, чтобы вместо присвоения переменным в выделенном комментариями блоке заранее определенных значений пользователь мог выбирать данные при помощи формы. При этом:

1. Значение переменной `sPovod` должно выбираться из трех возможных значений: "освоении новых информационных технологий", "внедрении новых программных продуктов" и значение, которое пользователь может ввести через текстовое поле. Используйте для этого набор из трех переключателей и текстовое поле (оно должно быть спрятано, если пользователь выбрал один из первых двух переключателей). По умолчанию должно подставляться "освоении новых информационных технологий".
2. Значение переменной `sFio` должно выбираться пользователем при помощи комбинированного списка. В этот комбинированный список должны автоматически помещаться значения из всех непустых ячеек столбца А листа Excel. По умолчанию должно выбираться значение "Иванова Ивана Ивановича".

Примечание

Образец для работы с ячейками столбца можно получить из ответов к предыдущим лабораторным работам.

3. Значения переменных `bFlagPremia` и `bFlagGramota` должны устанавливаться в зависимости от состояния двух флажков — "Премия" и "Грамота". По умолчанию оба флажка должны быть установлены. Если пользователь снял оба флажка, то ему должно выводиться предупреждающее сообщение "Не выбрана ни премия, ни почетная грамота!" с отменой вывода документа.
4. Пользователь должен иметь возможность задавать значение переменной `nSummaPremii` либо при помощи полосы прокрутки с диапазоном значений

от 0 руб. до 100 000 руб., либо при помощи текстового поля. Если флажок "Премия" снят, то полоса прокрутки и текстовое поле должны быть спрятаны от пользователя.

Ход полосы прокрутки (увеличение или уменьшение значения при щелчке на кнопках со стрелками) должен быть равен 100 руб.

По умолчанию размер премии должен быть равен 100 руб.

5. Поместите на форму еще одну кнопку **Отмена**. Эта кнопка должна закрывать текущую форму и срабатывать при нажатии клавиши <Esc>.
6. В заголовке формы должно выводиться значение "Формирование приказа о выплате премии".

Общий вид формы может выглядеть, например, так, как представлено на рис. 5.14.

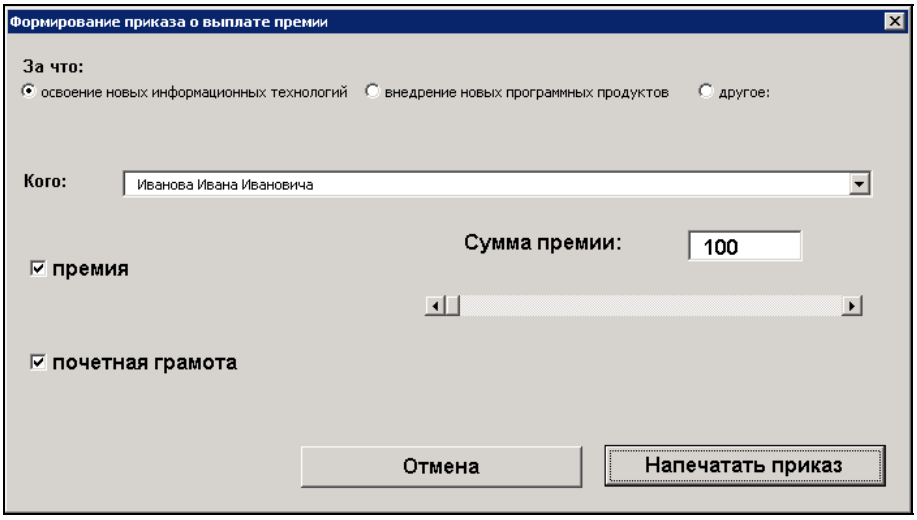


Рис. 5.14. Готовая форма

Ответ к заданию 5

К пункту 1 задания (работа с переключателями и текстовым полем):

1. В окне **Project Explorer** два раза щелкните мышью по объекту формы `UF1`. Затем в **ToolBox** щелкните по объекту `Label` и отведите место этому элементу управления в верхней части формы. Щелкните правой кнопкой мыши по созданному элементу управления `Label1` и в контекстном меню выберите **Properties**. Измените значение свойства `Caption` на `"За что:"` и при помощи свойства `Font` подберите подходящий шрифт и его размер.

2. В **ToolBox** щелкните по элементу управления `OptionButton` и отведите на форме место этому элементу управления. Повторите эту операцию еще два раза.
3. Откройте свойства первого переключателя. Измените значение свойства `Name` на `optOsvoenie`, а значение свойства `Caption` — на `"освоение новых информационных технологий"`. Для второго переключателя поменяйте значение свойства `Name` на `optVnedrenie` и свойство `Caption` — на `"внедрение новых программных продуктов"`, для третьего — на `optDrugoe` и `"другое:"` соответственно.
4. В **ToolBox** щелкните по элементу управления `TextBox` и поместите его в нужное место формы. Установите для свойства `Name` этого элемента управления значение `txtDrugoe`.
5. Щелкните правой кнопкой мыши по пустому месту на форме и в контекстном меню выберите **View Code**. В списке событий в верхней части окна редактора кода выберите событие `Initialize` для `UserForm` и введите для него следующий код:

```
optOsvoenie.Value = True
txtDrugoe.Visible = False
```

6. Для события `Change` переключателя `optDrugoe` введите следующий код:

```
If optDrugoe.Value = True Then
    txtDrugoe.Visible = True
Else
    txtDrugoe.Visible = False
End If
```

7. Перейдите к коду события `Click` для `CommandButton1` и вместо строки:

```
sPovod = "освоении новых информационных технологий"
```

   введите следующий код:

```
If optOsvoenie.Value = True Then sPovod = _
    "освоении новых информационных технологий"
If optVnedrenie.Value = True Then sPovod = _
    "внедрении новых программных продуктов"
If optDrugoe.Value = True Then sPovod = txtDrugoe.Value
```

8. Запустите форму на выполнение, напечатайте приказ и убедитесь, что все работает согласно поставленным условиям.

К пункту 2 задания (работа с комбинированным списком):

1. Разместите на форме еще один элемент управления `Label` с надписью `"Кого:"` и настройте для него шрифт.

2. Щелкните в **Toolbox** по элементу управления ComboBox и выделите для него место на форме. Присвойте созданному элементу управления ComboBox имя cbFIO.

3. Откройте код для события Initialize нашей формы UserForm и дополните его следующими строками:

```
Dim oColumn As Range
Dim oCell As Range
Set oColumn = Columns("A")
For Each oCell In oColumn.Cells
    If oCell.Value <> "" Then
        cbFIO.AddItem oCell.Value
    End If
Next
cbFIO.ListIndex = 0
```

4. Перейдите к коду события Click для CommandButton1 и вместо строки:

```
sFio = "Иванова Ивана Ивановича"
```

введите следующий код:

```
sFio = cbFIO.Value
```

5. Запустите форму на выполнение и убедитесь, что все работает нормально.

К пункту 3 задания (работа с флажками):

1. При помощи **ToolBox** поместите на форму два элемента управления CheckBox. Для первого элемента свойству Name присвойте значение chPremia и для свойства Caption — значение "Премия", для второго — chGramota и "Почетная грамота" соответственно.

2. Откройте код события Initialize формы UserForm и дополните его следующими строками:

```
chPremia.Value = True
chGramota.Value = True
```

3. Перейдите к коду события Click для CommandButton1 и вместо строк:

```
bFlagPremia = True
bFlagGramota = True
```

введите следующий код:

```
bFlagPremia = chPremia.Value
bFlagGramota = chGramota.Value
```

```
If bFlagPremia = False And bFlagGramota = False Then
    MsgBox "Не выбрана ни премия, ни почетная грамота!"
    Exit Sub
End If
```

4. Запустите форму на выполнение и убедитесь, что все работает нормально.

К пункту 4 задания (применение полосы прокрутки и дублирующего текстового поля):

1. Поместите на форму еще один элемент управления `Label` с надписью `"Сумма премии:"`. Присвойте его свойству `Name` значение `lblSum`.
2. Поместите рядом текстовое поле и присвойте его свойству `Name` значение `txtSum`.
3. Разместите под текстовым полем элемент управления `ScrollBar` и присвойте следующие значения его свойствам:
   - `Name` — значение `sbSum`;
   - `Min` — значение 0;
   - `Max` — значение 100 000;
   - `SmallChange` — значение 100.
4. Для события `Change` элемента управления `sbSum` введите следующий код:

```
txtSum.Value = sbSum.Value
```

5. Для события `Change` элемента управления `txtSum` введите следующий код:

```
sbSum.Value = CLng(txtSum.Value)
```

Примечание

Такой код является потенциально опасным, поскольку не проверяется вводимое пользователем в текстовом поле значение. Если это значение будет невозможно преобразовать в числовое или оно окажется больше 100 000, то возникнет ошибка времени выполнения. Как предупреждать появление ошибок и перехватывать их, будет рассмотрено в *гл. 6*.

6. Для события `Initialize` нашей формы `UserForm` добавьте следующий код:

```
sbSum.Value = 100
txtSum.Value = 100
```

7. Для события `Change` элемента управления `chPremia` добавьте следующий код:

```
If chPremia.Value = False Then
    lblSum.Visible = False
```

```
    txtSum.Visible = False
    sbSum.Visible = False
Else
    lblSum.Visible = True
    txtSum.Visible = True
    sbSum.Visible = True
End If
```

8. Для кода `Click` кнопки `CommandButton1` вместо кода:

   ```
   nSummaPremii = 100000
   ```

   впишите код:

   ```
   nSummaPremii = sbSum.Value
   ```

9. Запустите форму на выполнение и убедитесь, что все работает нормально.

К пункту 5 задания (применение кнопки):

1. Разместите на поле еще одну кнопку и настройте значения ее свойств следующим образом:
   - `Name` — значение `btnEscape`;
   - `Caption` — значение "Отмена";
   - `Cancel` — значение `True`.
2. Для события `Click` этой кнопки поместите код

   ```
   Unload Me
   ```

К пункту 6 задания (изменение заголовка формы):

1. Щелкните правой кнопкой мыши по пустому месту на форме и в контекстном меню выберите **Properties**.
2. Для свойства `Caption` настройте значение "`Формирование приказа о выплате премии`".
3. Запустите форму на выполнение и убедитесь, что приказы печатаются правильно.