Введение в разработку приложений на VBA

Приложения

Приложение – это программа или комплекс программ специального назначения, написанные на каком либо языке программирования и решающие класс задач какого-либо направления.

База данных помимо всего прочего **хранит приложения**, занимающиеся обработкой данных этой базы. Такие приложения могут высчитывать какие-либо поля таблиц, проверять значения полей, выводить интерфейс взаимодействия с базой данных и т.д.

Важной частью разработки приложения является создание сервисных средств, обслуживающих **диалог с пользователем** приложения (формы, кнопки, меню, поля ввода, переключатели и т.д.).

Все языки программирования **с приставкой «Visual»** предоставляют для разработчиков приложений инструменты для создания таких форм, используя готовые «строительные блоки».

Объекты VBA

Одним из основных понятий VBA является **объект**. В VBA имеется более 100 встроенных объектов (рабочие книги (WorkBook), рабочие листы (Work-Sheet), рабочие ячейки (Cell), формы (UserForm), элементы управления (TextBox, CommandButton, Label и др.), диалоги и т. д.).

Объектом можно управлять с помощью программы на языке VBA. Каждый объект обладает не которыми характеристиками, или свойствами (шрифт, цвет, видимость на экране и т.д.). Изменяя свойства, можно менять характеристики объекта.

Свойство представляет собой атрибут объекта, определяющий его характеристики. Синтаксис применения свойства:

```
Объект.Свойство
```

Методы VBA

Объект содержит также **список методов**, которые к нему могут быть применены. Например, показать диалог (форму) на экране или убрать его можно с помощью **методов Show** и **Hide** соответственно.

Метод представляет собой действие, выполняемое над объектом.

Синтаксис применения метода:

```
Объект.Метод
```

Объекты VBA

Объект – это программный элемент, который имеет свое отображение на экране, содержит некоторые переменные, определяющие его свойства, и некоторые методы для управления объектом.

Наиболее часто в VBA используются следующие встроенные объекты:

Range	Диапазон ячеек (может включать только одну ячейку)
Cells	Ячейка
Sheet	Лист
Worksheet	Рабочий лист
DialogSheet	Диалоговое окно

Объекты VBA

Полноценное применение Access предполагает **широкое использование объектов** (таблиц, форм, отчетов, элементов управления и т.п.). Пользователь может создавать в дополнение к существующим свои объекты, которые называются пользовательскими.

Объект можно представить как содержимое модуля класса. В отличие от стандартных модули класса имеют **события**, **Initialize** (инициализация) и **Terminate** (завершение), связанные с созданием и уничтожением экземпляров класса. Все модули класса имеют **свойство Name**, которое определяет его имя, используемое для ссылок на класс (объект).

Объекты VBA

В разделе описаний модуля класса задаются **опции Option Compare Database** и **Option Explicit**, а также описываются **модульные переменные**, доступные для всех процедур данного модуля. Переменная, описанная с помощью **слова Public**, является свойством объекта, для доступа к которому нужно использовать полное имя в форме **ИмяОбъекта.ИмяСвойства**.

Рассмотрим пример простейшего объекта с **именем Ter** и со свойствами Tag и Value.

```
Option Compare Database
Option Explicit
Public Tag As String
Public Value As String
```

Для использования объекта нужно описать объектную переменную и создать экземпляр объекта:

Dim t As Ter
Set t = New Ter

Семейства VBA

Большинство объектов принадлежит к группе подобных объектов. Эти группы называются **семействами**.

Семейства можно использовать одним из двух способов: либо какое-либо действие совершается над всеми объектами семейства, (например, удалить – Delete), либо со ссылкой на семейство выбирается конкретный объект для работы с ним.

Семейство содержится в **объекте-владельце (контейнере)**. Так, например, **Forms**, содержит **семейство открытых форм**.

В VBA имеется **объект Collection**, который позволяет создавать и использовать собственные семейства объектов.

Семейства VBA

Семейство объектов напоминает массив, так как содержит набор однородных элементов, к которым можно получить доступ, используя числовой индекс. Однако семейства объектов предоставляют больше возможностей, поскольку содержат методы добавления и удаления объектов, а также организации ссылок на них.

В приложениях Access широко применяются **встроенные семейства Reports, TableDefs, RecordSet**. Элемент семейства сам может быть семейством (форма содержится в семействе Forms и в свою очередь содержит семейство элементов управления Controls). Некоторые элементы управления включают свои семейства (набор вкладок).

Обратиться к объекту, входящему в семейство:

ИмяСемейства.ИмяОбъекта

ИмяСемейства.ИндексОбъекта.

Семейства VBA

Семейство поддерживает ряд методов и свойств, которые дают возможность помещать в семейство и удалять из него другие объекты, а также ссылаться на них. Создавая собственные семейства объектов при помощи VBA, можно ограничиться только тремя методами и одним свойством:

- **Метод Add** добавляет объекты в семейство. В качестве параметров задаются указатель на объект и его уникальный идентификатор (имя);
- **Метод Remove** удаляет объекты из семейства. Параметрами метода являются имя или индекс объекта.
- **Метод Item** обращается к конкретному объекту семейства и возвращает указатель на него. В качестве параметра задается имя или индекс объекта.
- **Свойство Count** возвращает количество объектов в семействе.

События VBA

Событие представляет собой действие, распознаваемое объектом (например, щелчок мышью или нажатие клавиши), для которого можно запрограммировать отклик. События возникают в результате действий пользователя программы, или же они могут быть вызваны системой.

Процедуры обработки событий формы условно можно разделить на две группы: процедуры без параметров и с параметрами.

К первой группе относятся, например, **процедуры Initialize, Load, Click и DblClick**. Эти процедуры имеют следующий синтаксис:

```
Sub UserForm_Событие()
  Последовательность инструкций
End Sub
```

События VBA

Ко второй группе процедур относятся **Unload**, **MouseDown**, **MouseUp** и **KeyDown.** Эти процедуры имеют следующий синтаксис:

```
Private Sub Form_Событие(СписокПараметров)
  Последовательность инструкций
End Sub
```

Процедура обработки **события может быть связана с действием над любым элементом управления** (кнопкой, полем, списком, флажком и т.п.). **Имя** такой **процедуры** может быть выбрано пользователем самостоятельно или формироваться по умолчанию следующим образом:

```
ИмяЭлементаУправления_Событие(СписокПараметров)
```

Панель инструментов «Элементы управления»

Панель инструментов Элементы управления обычно содержит следующие кнопки: выбор объектов, надпись, поле, поле со списком, список, флажок, переключатель, выключатель, рамка, кнопка, набор вкладок, набор страниц, полоса прокрутки, счетчик, рисунок.

Каждый из данных элементов уже имеет описание собственного классового модуля и может быть использован для написания приложений базы данных.

Поле (TextBox)

Элемент управления TextBox позволяет ввести в форму информацию, которая затем может быть использована в программе, также может служить и для вывода информации.

Для добавления любого элемента управления в форму необходимо нажать соответствующую кнопку на панели элементов управления, а затем щелкнуть по форме в требуемой позиции.

После создания любого элемента управления желательно сразу же **присвоить ему новое имя**, иначе будет использоваться имя, заданное по умолчанию.

При присвоении имен полей используется следующее правило: **TxtИмяОбъекта** (TxtAge, TxtfirstName и т.д.).

Поле (TextBox)

Использование поля в VBA:

1. Создадим новую **форму** в режиме конструктора и поместим на нее объект Поле.
2. Зададим ему **имя TxtTest** и выберем процедуру обработки события **нажатия клавиши Enter**:

Private Sub TxtTest_Enter()

3. Для установки и получения содержимого поля используется **свойство Value**. Это свойство имеет тип **Variant**. Пропишем следующий код, сохраняющий введенное в поле значение в переменной X:

Dim X As Variant

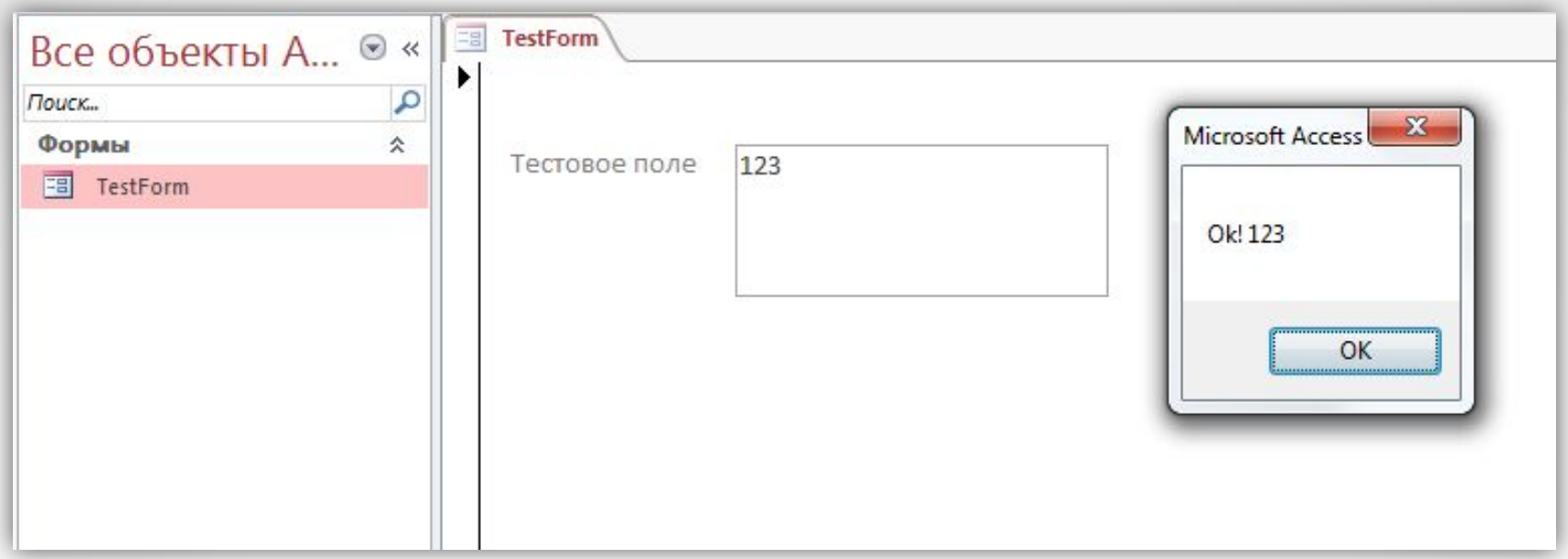
X = TxtTest.Value

4. Для подтверждения удачного сохранения введенных данных выведем на экран соответствующее сообщение:

MsgBox ("Ok!" & X)

5. Сохраним форму как **TestForm** и запустим созданную форму на исполнение.

Поле (TextBox)




```
Option Compare Database
Private Sub TxtTest_Enter()
Dim X As Variant
X = TxtTest.Value
MsgBox ("Ok! " & X)
End Sub
```

Поле (TextBox)

Использование поля в VBA:

1. Создадим новую **форму** в режиме конструктора и поместим на нее объект Поле.
2. Зададим ему **имя TxtTest** и выберем процедуру обработки события **нажатия клавиши Enter**:

```
Private Sub TxtTest_Enter()
```

3. Для установки и получения содержимого поля используется **свойство Value**. Это свойство имеет тип **Variant**. Пропишем следующий код, сохраняющий введенное в поле значение в переменной X:

```
Dim X As Variant
X = TxtTest.Value
```

4. Для подтверждения удачного сохранения введенных данных выведем на экран соответствующее сообщение:

```
MsgBox ("Ok!" & X)
```

5. Сохраним форму как **TestForm**, а модуль как **TestModule** и запустим созданную форму на исполнение.

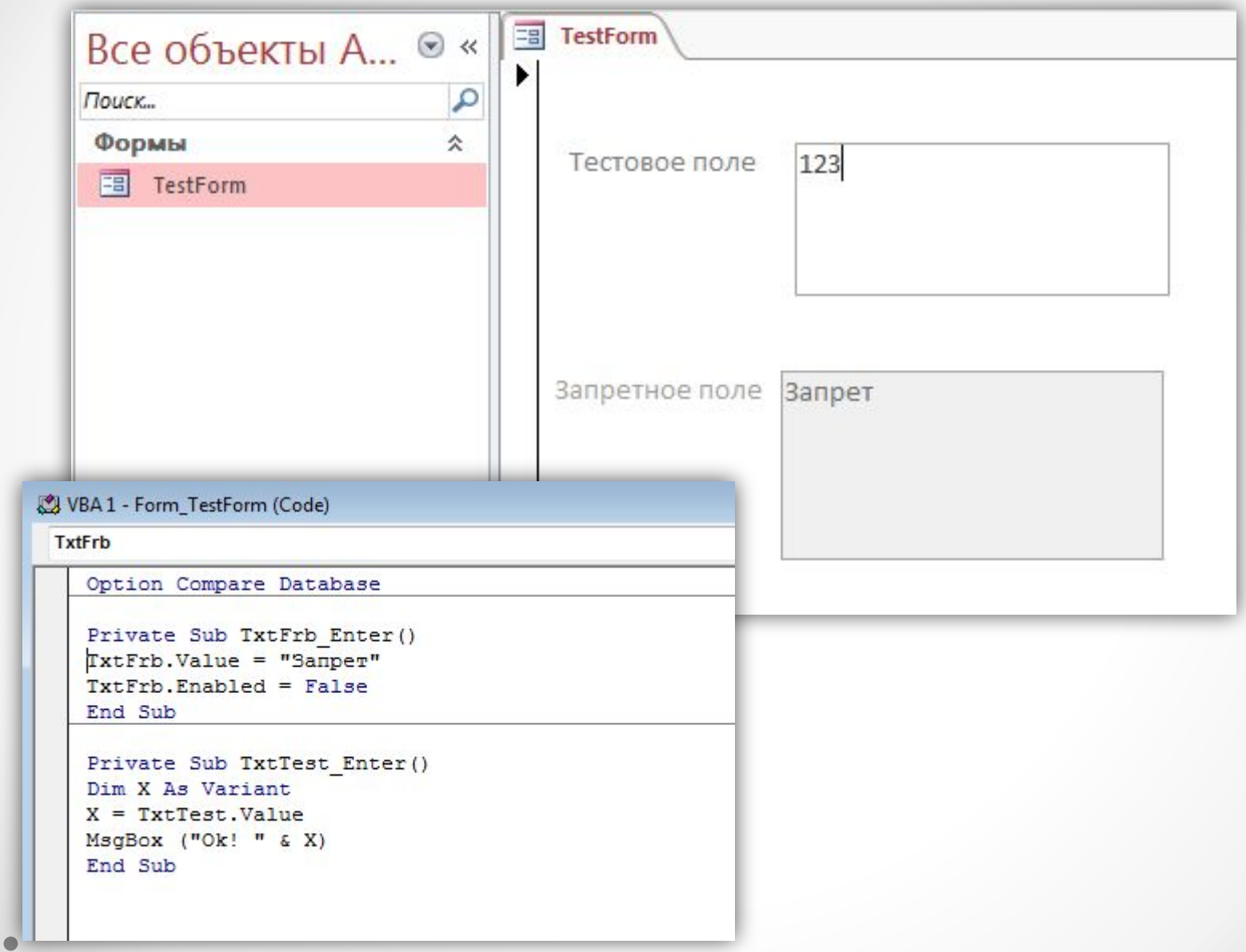
Поле (TextBox)

Если нужно запретить изменение содержимого поля (например, объект TextBox применяется для отображения доступной только для чтения информации, такой, как имена файлов), следует «отключить» поле с помощью **свойства Enabled**, установив его равным значению False. Если значение свойства равно **True**, то изменение содержимого поля разрешено.

Для того чтобы заблокировать поле:

1. Создайте новое поле и дайте ему имя;
2. Создайте обработку какого-либо события;
3. Передайте свойству Value созданного поля какой-либо текст;
4. Поставьте свойство Enabled созданного поля в значение False;
5. Сохраните код программы и запустите форму на исполнение.

Поле (TextBox)



Надпись (Label)

Элемент управления Label предназначен для вывода текста в форме, на пример для вывода заголовка для тех элементов управления, у которых отсут ствует собственное **свойство Caption**. В качестве примера таких элементов можно назвать поле или рисунок в форме. В этом случае надпись находится около этого элемента управления, указывая его назначение. По умолчанию надписи имеют имена Label1, Label2 и т. д.

Для задания текста надписи можно использовать **свойство Caption**. Например:

Label1.Caption="Адрес"

Задание: создайте на тестовой форме надпись и поменяйте текст, находящийся в нем по какому-либо событию.

Кнопка (CommandButton)

Элемент управления CommandButton задает выполнение некоторого действия, например запуск, прерывание или останов некоторого процесса. По умолчанию кнопкам присваиваются имена CommandButton1, CommandButton2 и т. д. Для изменения имени кнопки откройте окно свойств и введите новое имя в **поле Имя (Name).**

Можно задать текст, который будет выводиться на кнопке вместо установленного по умолчанию значения **CommandButtonN** Для этого установите новое значение **свойства Caption**.

Например:

CommandButton1.Caption="Моя новая программа"

Кнопка (CommandButton)

Если в форме имеется несколько кнопок, то одну из них можно назначить применяемой по умолчанию. Например, при выводе окна сообщений, в котором содержится запрос на подтверждение удаления данных, кнопка Да обычно за

дана по умолчанию. Если по ошибке нажать клавишу «Пробел» или Enter, то вся информация будет уничтожена. Поэтому нужно назначить применяемой по умолчанию кнопку Нет.

Для того чтобы назначить кнопку по умолчанию, нужно присвоить **значение True** ее **свойству Default**. Тогда **свойству Default** остальных кнопок формы **автоматически будет присвоено значение False**.

Например:

CommandButton1.Default=True

Кнопка (CommandButton)

С нажатием кнопки можно связать выполнение **некоторого действия**, если назначить эту кнопку некоторому событию, например **Нажатие кнопки (Click).** Процедура **обработки события Click** не имеет параметров.

Например, следующая процедура обработки события выводит в **окне отладки Debug** сообщение «Моя новая программа», после того как нажата кнопка **CommandButton1:**

```
Private Sub CommandButton1_Click()
Debug.Print "Моя новая программа"
End Sub
```

Можно изменить состояние кнопки: запретить пользователю нажатие кнопки, если оно приведет к опасным или нежелательным последствиям. На пример, можно отключить кнопку печати, пока не выбран принтер. При запрете доступа кнопка выглядит серой. **Для отключения объекта** используется **значение False свойства Enabled.**

Список (ListBox)

Элемент управления ListBox предназначен для хранения списка значений, из которого можно выбрать один или несколько элементов.

Существуют следующие варианты **выбора элементов в списке**:

1 – один элемент,

2 – несколько последовательно расположенных элементов,

3 – несколько произвольно расположенных элементов.

Способ **выбора элементов** в списке определяется **свойством MultiSelect**, значение которого (числовое или заданное константой) можно указать в окне свойств или в программе:

Вариант	Значение	Константа
1	0	`fmMultiSelect`
2	1	`fmMultiSelectExtended`
3	2	`fmMultiSelectMulti`

Список (ListBox)

Для добавления новых элементов в список используется **метод AddItem**. При этом нужно задать параметр, который определяет строку с названием добавляемого в список элемента:

ListBox1.AddItem элемент

В качестве элемента может выступать, в частности, число, имя переменной, элемент массива (a(i)), в этом случае в список добавляется их значение.

В следующей процедуре **метод AddItem** добавляет в список названия месяцев года:

```
Public Sub Months()
ListBox1.AddItem "January”
ListBox1.AddItem "February”
ListBox1.AddItem "December”
End Sub
```

Список (ListBox)

Для заполнения списка последовательными числами можно использовать процедуру:

```
Public Sub NumberList()
For i=1 To 20ListBox1.AddItem I
Next i
End Sub
```

Для удаления элемента из списка используется **метод RemoveItem**, при этом в качестве параметра метода указывается номер удаляемого пункта:

```
NumberList()For i=0 To ListBox1.ListCount-1
ListBox1.RemoveItem i
Next i
End
```

Задание: создать кнопку по нажатию на которую список заполнялся бы какими-либо значениями, и другую кнопку – удаляющую все значения в списке.

Список (ListBox)

Пусть в программе требуется определить выбранные элементы списка, например, узнать их значения. Если в списке задан выбор только одного элемента, то **свойство Text элемента управления ListBox** содержит выделенный элемент, в противном случае **свойство Text** равно пустой строке. **Свойство ListIndex** содержит номер выделенного пункта в списке. Выбранный в списке элемент можно вывести, например, в **окне отладки Debug** с помощью инструкции:

Debug.Print ListBox1.Text

Если известно, что в списке выделено несколько элементов, то необходимо проверить каждый пункт списка, чтобы определить, выделен он или нет.

Список (ListBox)

Для этого используется **свойство Selected**, которое по индексу пункта возвращает **значение True**, если пункт выбран, и **значение False** – в противном случае.

Например, если необходимо найти сумму выделенных элементов в списке, то можно воспользоваться следующей инструкцией:

```
For i = 0 To listBox1.ListCount-1
If ListBox1.Selected(i) = True
Then S = S + ListBox1.List(i)
End If
Next I
```

Свойство ListCount содержит общее количество элементов (пунктов) в списке. **Свойство List** возвращает по номеру пункта его текст.

Поле со списком (ComboBox)

Если используется поле со списком, то необходимый элемент можно выбрать сразу из списка или ввести его имя вручную для автоматического поиска. Текущее значение в **элементе управления ComboBox** отображается в поле, а список всех возможных значений выводится при нажатии кнопки со стрелкой.

Элемент управления ComboBox отличается **от элемента управления ListBox** тем, что в нем можно явно выделить требуемое значение.

Существует **два типа** полей со списком. С помощью объектов первого типа можно ввести в поле данные, которые затем можно использовать как:

- критерий выбора элементов в списке; например, если список содержит на звания месяцев года и вводится слово «May», то осуществляется перемещение на этот пункт списка;
- новое значение; например, для задания новой величины масштаба изображения; таким образом, в программе должен быть предусмотрен случай, когда введенного значения нет в списке.

Поле со списком (ComboBox)

Если **элемент управления ComboBox** относится ко второму типу, то для выбора элемента необходимо открыть список, нажав кнопку со стрелкой, а затем указать в списке требуемый элемент списка. Этот элемент появится в поле **элемента управления ComboBox**.

Тип объекта ComboBox можно указать с помощью **свойства Style,** значение которого в программе указывается либо числом (0, 2), или константой:

Тип	Значение	Константа
Ввод данных	0	`fmStyleDropDownCombo`
Выбор значения	2	`fmStyleDropDownList`

Поле со списком (ComboBox)

Для заполнения поля со списком применяется **метод AddItem**. Для получения значения, содержащегося в поле **элемента управления ComboBox**, можно использовать **свойства Value и Text**.

Например, два следующих оператора выполняют одно и то же действие выводят в окне **отладки Debug** текст, содержащийся в поле **элемента управления ComboBox**:

Debug.Print *ComboBox1.****Value***

Debug.Print *ComboBox1.****Text***

При присвоении значения **свойству Text** автоматически выполняются следующие действия:

- заданный текст выводится в поле **элемента управления ComboBox** (если заданный текст не является элементом списка, то выдается сообщение об ошибке);
- свойству ListIndex элемента ComboBox присваивается индекс элемента списка, соответствующего заданному значению.

Флажок (Checkbox)

Элемент управления CheckBox имеет вид маленького квадрата. С флажком можно связать некоторый заголовок (надпись). Если этот квадрат пуст, то при щелчке по нему в нем появляется галочка, и, наоборот, если квадрат был помечен галочкой, то при щелчке по нему галочка исчезает. Если флажок установлен (есть галочка), то **свойство Value элемента управления CheckBox** имеет **значение True**, в противном случае его **значение False**.

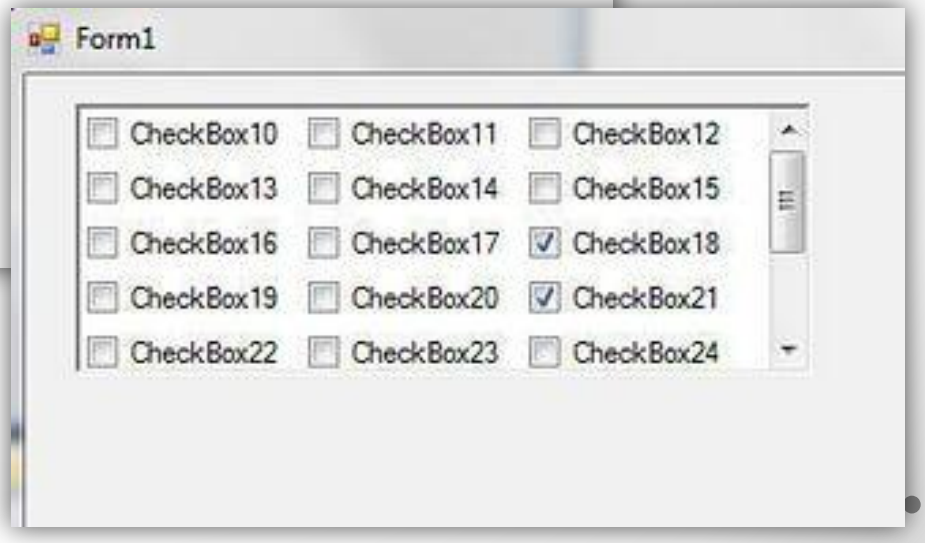
Состояние флажка используется в процедурах **обработки события Click** флажка или в других процедурах, где требуется анализ установки этого флажка.

Если флажок недоступен (блокирован **свойством Enabled**), то его значение (**свойство Value**) равно **константе Null**.

Флажок (Checkbox)

Описать работу **элемента управления CheckBox** можно, например, с помощью следующих инструкций:

```
Public Sub CheckBox1_Click()
    If CheckBox1.Value=True Then
    ' инструкции 1
    Else
    ' инструкции 2
    End If
End Sub
```



Переключатель (OptionButton)

Элемент управления OptionButton предназначен для выбора одного варианта из нескольких. Переключатель может менять значения независимо от других или входить в состав группы переключателей. В любое время в группе может быть выбран только один переключатель. Отмена выбора одного **элемента управления OptionButton** при выделении другого в группе осуществляется автоматически.

Группировка переключателей может быть **выполнена двумя способами**:

- С помощью **элемента управления Рамка (Frame).** Все **объекты управления OptionButton**, расположенные в одной рамке, рассматриваются как члены одной группы. Для каждого набора переключателей должна использоваться своя рамка.
- С помощью **свойства для группировки объектов – GroupName**. При выборе **элемента управления OptionButton** отменяется выбор всех переключателей, значение **свойства GroupName** которых совпадает со значением того же свойства выделенного элемента управления OptionButton. При использовании **свойства GroupName** отпадает необходимость в создании **элемента управления Frame**. **Свойство GroupName** может быть установлено как в окне свойств, так и в программе.

Переключатель (OptionButton)

Свойство Value активизированного переключателя имеет **значение True**.

Процедура, описывающая работу трех переключателей, может иметь вид:

```
Public Sub CheckOptionButton()
If OptionButton1.Value Then
' инструкции 1
Elself OptionButton2.Value Then
' инструкции 2
```

```
Elself OptionButton3.Value Then
' инструкции 3
End If
End Sub
```

Рамка (Frame)

Элемент управления Frame предназначен для группирования элементов в форме. По умолчанию рамки имеют имена Frame1, Frame2 и т. д. Установить новое значение имени рамки можно с помощью **свойства Name**.

Свойство Caption определяет текст (надпись), который появляется вверху рамки. Например:

Frame1.Caption ="Варианты заданий"

Задание на семинар

1) Ознакомиться

Лабораторная работа №7

1) Интегрировать изученные элементы в свои проекты;

2) Предоставить отчет и скриншоты выполненной работы.