Министерство образования, науки и молодежной политики  
Нижегородской области

Государственное бюджетное профессиональное образовательное   
учреждение «Нижегородский автомеханический техникум»

**Методическая разработка**

**«Основные элементы управления.   
Списки»**

**по МДК.01.02. Прикладное программирование**

**для студентов специальности**

**09.02.03 Программирование в компьютерных системах**

Разработала преподаватель

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Г.Н. Атмайкина

Рассмотрено и утверждено на заседании ПЦК учетных и экономических дисциплин и программирования

Протокол №\_\_\_\_\_от \_\_\_\_\_\_\_\_\_\_\_2021 г.

Председатель комиссии

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.А. Мазунина

## г. Нижний Новгород

**2021 год**Содержание

[Аннотация 3](#_Toc30579253)

[Введение 4](#_Toc30579254)

[Основная часть 6](#_Toc30579255)

[Заключение 15](#_Toc30579256)

[Список использованных источников 16](#_Toc30579257)

Аннотация

Методическая разработка «Основные элементы управления. Списки» по МДК.01.02. Прикладное программирование предназначена для студентов специальности 09.02.03 Программирование в компьютерных системах и разработана в соответствии с рабочей программой МДК.01.02. Прикладное программирование.

Методическая разработка содержит:

- теоретические сведения по теме «Основные элементы управления. Списки»;

- практические задания по данной теме, которые носят и репродуктивный, и частично-поисковый, и поисковый характер;

- задания для самостоятельной работы, которые позволяют закрепить навыки работы учащихся;

- перечень использованных источников.

Введение

Visual Basic является мощным современным языком программирования. С помощью Visual Basic можно создавать приложения практически для любой области современных компьютерных технологий: бизнес-приложения, игры, мультимедиа, базы данных, интернет-приложения. При этом приложения могут быть как простыми, так и очень сложными, в зависимости от поставленной задачи.

Простота и мощность языка Visual Basic 2010 позволили сделать его встроенным языком для приложений Microsoft Office. В настоящее время Visual Basic уже не считается учебным языком - знание Visual Basic и его диалектов (VBA, VBScript) становится необходимостью для современного программиста любого уровня.

В Visual Basic используются все самые современные методы программирования: объектно-ориентированная модель, включая наследование визуальных классов, модель составных объектов СОМ (Component Object Model), технология программных компонентов ActiveX. Кроме того, Visual Basic позволяет создавать многопоточные программы, службы Windows разнообразные сетевые приложения и многое другое.

Среда Visual Basic включает множество элементов управления, позволяющих создать богатый пользовательский интерфейс. Одни элементы управления предназначены для ввода информации во время выполнения приложения, другие просто отображают определенные данные, третьи используются для задания запросов.

Основная часть

Списки, размещаемые в формах Visual Basic, позволяют пользователю выбрать один из возможных вариантов ответа. Для создания списков применяется несколько элементов управления (таблица 1).

Таблица 1 - Элементы управления, используемые при создании списков

|  |  |  |
| --- | --- | --- |
| Название | Кнопка | Описание |
| CheckedListBox |  | Список, являющийся объединением элементов управления ListBox и CheckBox и представляющий собой список флажков |
| ComboBox |  | Этот тип списка позволяет пользователю осуществлять выбор значения, вводимого в размещаемое сверху поле ввода, или выбирать значение из списка, открываемого нажатием кнопки со стрелкой, размещаемой с правой стороны. Список данного типа удобно использовать в том случае, если вводимых значений много, а места в форме для расположения обычного списка не хватает |
| DomainUpDown |  | Список, предназначенный для ввода текстовой информации и представляющий собой поле с двумя кнопками, по- зволяющими перемещаться по списку |
| ListBox |  | Список, в котором элементы расположены в одну или несколько колонок. В случае если элементы списка не помещаются в созданном объекте ListBox, в нем появляются вертикальные и/или горизонтальные полосы прокрутки |
| NumericUpDown |  | Список, предназначенный для ввода числовой информации и представляющий собой текстовое поле с двумя кнопками, позволяющими уменьшать или увеличивать размещенное в текстовом поле число на определенное значение. |

Элемент управления ListBox

Элемент управления ListBox, размещенный в форме, представляет собой список, из которого пользователь может выбрать одно из предложенных значений. Значения в списке могут размещаться в одну или несколько колонок в зависимости от значения свойства MultiColumn. Если элементы списка расположены в нескольких колонках, с помощью свойства ColumnWidth можно изменить заданную по умолчанию ширину колонок.



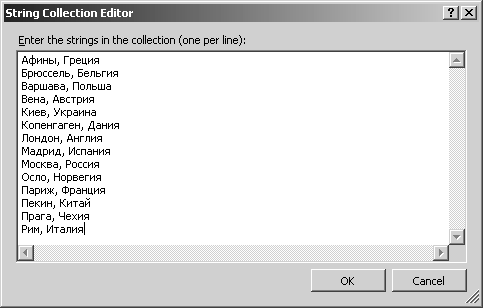
В том случае, если элементы списка не помещаются в выделенную для них в форме область, появляются полосы прокрутки, позволяющие просмотреть весь список. Чтобы полоса прокрутки элемента управления ListBox всегда отображалась, необходимо присвоить значение True свойству ScrollAlwaysVisible.

Возможность частичного отображения элемента списка задается с помощью свойства IntegralHeight. Если указано значение True, то в списке может отображаться только строка целиком.

Добавление элементов в список

Элементы в список могут добавляться во время разработки и программно. При формировании списка во время проектирования с помощью свойства Items вручную задается весь необходимый список. Для этого следует выбрать свойство и нажать расположенную справа кнопку с тремя точками. Откроется диалоговое окно String Collection Editor (Редактор элементов списка), показанное на рисунке 1.

Каждая строка поля ввода соответствует одному элементу. Пустая строка может также являться элементом списка. После задания всех элементов нужно нажать кнопку OK, окно редактора элементов списка закроется, а элемент управления ListBox будет отображать указанные значения.



### Рисунок 1 - Диалоговое окно String Collection Editor

Данные в список не обязательно вводить в алфавитном порядке, т. к. их можно упорядочить, установив для свойства Sorted (Сортировка) значение True. В этом случае вновь вводимые элементы списка будут располагаться в алфавитном порядке.

### Для добавления элементов в список программным способом предназначен метод Add коллекции Items элемента управления ListBox. Как правило, данный метод помещается в редактор кода после инициализации компонентов.

Для добавления сразу нескольких элементов в список можно использовать метод AddRange. Тогда предыдущий код можно заменить следующими строками:

Dim capitals() As String = {"Афины, Греция", "Брюссель, Бельгия",

"Минск, Беларусь"} ListBox1.Items.AddRange(capitals)

Чтобы в списке не было одинаковых строк, необходимо перед добавлением элемента осуществлять проверку на его существование с помощью метода Contains. Тогда код будет иметь следующий вид:

Dim list As String() = {"Афины, Греция", "Брюссель, Бельгия",

"Минск, Беларусь"} For Each item As String In list

If Not ListBox1.Items.Contains(item) Then ListBox1.Items.Add(item)

End If Next item

Удаление элементов из списка

Visual Basic позволяет удалять элементы из списка программно с помощью методов Remove и RemoveAt коллекции Items элемента управления ListBox.

Для удаления элемента по его тексту используется метод Remove: ListBox1.Items.Remove("Минск, Беларусь")

Чтобы удалить из списка элемент по его индексу, необходимо воспользоваться методом RemoveAt. Следующая строчка позволяет удалить первый элемент списка:

ListBox1.Items.RemoveAt(0)

С помощью этих методов также можно удалить из списка выделенный элемент:

ListBox1.Items.Remove(ListBox1.SelectedItem) ListBox1.Items.RemoveAt(ListBox1.SelectedIndex)

Свойства SelectedItem и SelectedIndex задают соответственно текст и индекс выделенного элемента списка.

Для удаления всех элементов из списка предназначен метод Clear коллекции Items элемента управления ListBox: ListBox1.Items.Clear()

Вставка элементов в список

Чтобы во время выполнения приложения вставить элемент списка, нужно воспользоваться методом Insert коллекции Items элемента управления ListBox. Этот метод позволяет создать новый элемент списка в определенном месте списка. Например, с помощью следующей строки можно вставить имя столицы в начало списка:

ListBox1.Items.Insert(0, "Brussels, Belgium")

Выбор нескольких элементов из списка

Visual Basic 2010 позволяет использовать списки, разрешающие пользователю выбирать из них несколько элементов. Для создания таких списков предназначено свойство SelectionMode.

Доступ к элементам списка

Порой необходимо иметь возможность обратиться к определенному элементу списка. Для доступа к элементам списка предназначено свойство Items элемента управления ListBox. Обращение к элементу осуществляется через индекс, определяющий его положение в списке. Нумерация элементов списка начинается с 0. Например, текст text1 и text2 первого и пятого элементов списка можно получить с помощью команд:

Dim text1 As String = ListBox1.Items(0) Dim text2 As String = ListBox1.Items(4)

Свойство Count коллекции Items позволяет определить количество элементов в списке. Это значение можно использовать, например, в том случае, если нужно обрабатывать элементы списка в цикле. С помощью следующей строки можно получить число строк num в списке:

Dim num As Integer = ListBox1.Items.Count

Выделенные элементы списка

Для определения выделенных пользователем элементов или их программного задания служат указанные в таблице 2 свойства.

Таблица 2 - Свойства, определяющие выделенных объектов

|  |  |
| --- | --- |
| Свойство | Описание |
| SelectedIndex | Задает или возвращает номер выделенного элемента списка. Если не выбран ни один элемент из списка, то возвращает значение –1 |
| SelectedIndices | Задает или возвращает номера выбранных элементов списка. Представляет собой коллекцию индексов выбранных элементов списка |
| SelectedItem | Задает или возвращает текст выбранного элемента списка |
| SelectedItems | Задает или возвращает коллекцию выбранных элементов списка |

Чтобы задать выделяемые по умолчанию элементы списка, при запуске приложения можно использовать свойства из таблицы 2 или метод SetSelected элемента управления ListBox, который имеет следующий синтаксис:

Sub SetSelected(ByVal *index* As Integer, ByVal *value* As Boolean)

где:

* *index* — номер элемента в списке;
* *value* — значение True позволяет выделить указанный элемент, а значение

False — снять с него выделение.

Если необходимо снять выделение со всех элементов списка, применяется метод ClearSelected элемента управления ListBox.

Рассмотрим небольшой пример, демонстрирующий использование свойства SelectedItems. Для этого разместим в форме два списка. В первый список элементы введем с помощью свойства Items, а второй список будет отображать выбранные в первом списке элементы. Для создания приложения выполните следующие действия:

1. Разместите в форме два списка, используя для этого элемент управления

ListBox.

1. С помощью свойства Items первого списка введите элементы списка.
2. Чтобы разрешить выбор из первого списка нескольких элементов, установите для его свойства SelectionMode значение MultiSimple.
3. Теперь необходимо задать процедуру формирования элементов второго списка при выборе элементов первого списка. Для этого в окне редактора кода из раскрывающегося списка Class Name (Имя класса) выберите элемент управления ListBox, соответствующий первому списку формы, а из раскрывающегося списка Method Name (Имя метода) — событие SelectedIndexChanged, вызываемое при изменении списка выбранных элементов.
4. Добавьте в тело процедуры следующий код:

Dim i As Integer ListBox2.Items.Clear()

For i = 0 To ListBox1.SelectedItems.Count — 1 ListBox2.Items.Add(ListBox1.SelectedItems(i))

Next

Этот код позволяет при щелчке кнопкой мыши на любом элементе сначала очистить содержимое второго списка, а затем заполнить его выделенными в первом списке элементами.

Результат работы приложения приведен на рисунке 2.

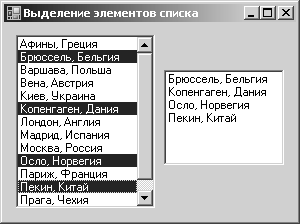


Рисунок 2 - Выбранные из первого списка значения отображаются   
во втором списке

Поиск элемента списка

Если необходимо определить индекс элемента списка, начинающегося с указанных символов, можно использовать метод FindString элемента управления ListBox, имеющий следующий синтаксис:

Function FindString(ByVal *s* As String,

ByVal *startIndex* As Integer) As Integer

где:

* *s*  - текст искомого элемента списка;
* *startIndex -* позиция, с которой начинается поиск элемента. Данный параметр можно опустить, тогда поиск будет осуществляться с начала списка.

Если элемент не найден, то возвращается значение ListBox.NoMatches, что соответствует числу –1.

Для нахождения элемента списка, текст которого соответствует указанному, предназначен метод FindStringExact элемента управления ListBox, имеющий аналогичный методу FindString синтаксис.

С помощью следующего кода можно удалить из списка все элементы, начинающиеся с буквы М:

Dim i As Integer = ListBox1.FindString("M") While Not i = ListBox1.NoMatches

ListBox1.Items.RemoveAt(i)

i = ListBox1.FindString("M", i) End While

Элемент управления ComboBox

Списки типа ComboBox называют раскрывающимися или полями со списком. Оба названия верны. Раскрывающимися их называют потому, что для выбора значения из списка сначала необходимо список открыть, нажав кнопку со стрелкой, расположенную с правой стороны поля ввода. Второе название — поле со списком — они получили из-за того, что по своим функциям список типа ComboBox совмещает функции списка ListBox и поля ввода TextBox. Иными словами, из списка ComboBox данные можно не только выбирать, но и вводить новое значение в находящееся в верхней части поле ввода. Использование списков ComboBox позволяет представлять большой объем информации, экономя при этом место в форме.



Стиль оформления списка

Стилем оформления списка типа ComboBox управляет свойство DropDownStyle. Оно может принимать значения, приведенные в таблице 3 и показанные на рисунке 3.

Таблица 3 - Значения свойства DropDownStyle

|  |  |
| --- | --- |
| Значение | Описание |
| Dropdown | Пользователь может вводить значение в текстовое поле, располагающееся в верхней части списка, либо открыть список, нажав кнопку с направленной вниз стрелкой с правой стороны поля, и выбрать из него требуемое значение. Выбранное из списка значение переносится в текстовое поле. Стиль, используемый по умолчанию |
| DropdownList | Пользователь может лишь выбрать значение из списка, открываемого нажатием кнопки с направленной вниз стрелкой с правой стороны поля. Выбранное из списка значение переносится в текстовое поле. |
| Simple | При данном значении список отображается в форме в открытом состоянии. Если все элементы не помещаются в нем, то появляется вертикальная полоса прокрутки. Пользователь может вводить значение в текстовое поле, располагающееся в верхней части списка, либо выбрать из списка требуемое значение, и оно переносится в текстовое поле |

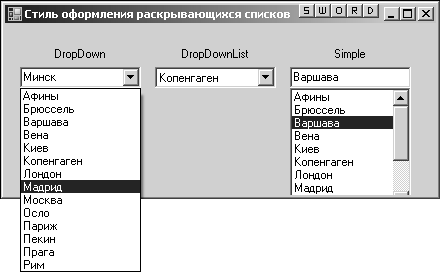


Рисунок 3 - Стили раскрывающегося списка

Параметры раскрывающегося списка

Для задания ширины ниспадающей части списка в пикселах служит свойство DropDownWidth. По умолчанию значение свойства соответствует ширине элемента управления ComboBox. При определении ширины ниспадающей части списка необходимо учитывать, что она не может быть меньше ширины самого списка.

Высота ниспадающей части списка задается свойством DropDownHeight.

Чтобы задать максимальное число элементов, отображаемых в видимой области ниспадающей части списка, следует воспользоваться свойством MaxDropDownItems. Если число элементов списка превышает указанное для этого свойства значение, то в списке появляется вертикальная полоса про- крутки. Свойство MaxDropDownItems может принимать любое целочисленное значение в диапазоне от 1 до 100 включительно. По умолчанию для свойства задано значение 8.

С помощью свойства MaxLength для списков, имеющих стиль Dropdown или Simple, можно задать максимальное число символов, которые пользователь может ввести в редактируемое поле списка.

Добавление и удаление элементов списка

Элементы могут добавляться в список и удаляться из него во время разработки приложения с помощью свойства Items и программно с использованием методов коллекции Items элемента управления ComboBox, аналогичные методам коллекции Items элемента управления ListBox.

Доступ к элементам списка

Для получения доступа к выбранному элементу списка типа ComboBox можно использовать свойство Text. Свойство принимает введенное в текстовое поле списка значение (для списков, у которых значение DropDownStyle равно Dropdown или Simple) или значение, выбранное из раскрывающегося списка.

Чтобы получить доступ к элементам списка, также можно применять свойство Items. Значения этого свойства являются массивом, размер которого равен количеству элементов в списке. Например, значение первого элемента списка будет равно ComboBox1.Items(0), второго — ComboBox1.Items(1) и т. д.

Элемент управления CheckedListBox

Элемент управления CheckedListBox является сочетанием элементов ListBox, задающего стандартный список, и CheckBox, имеющего вид флажка и предназначенного для выбора одного из двух возможных значений. Таким образом, данный объект представляет собой список элементов, с левой стороны каждого из которых расположен флажок (рисунок 4).

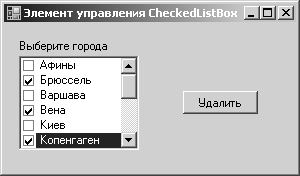


Рисунок 4 - Элемент управления CheckedListBox

Элемент управления CheckedListBox обладает основными свойствами списков, такими как наличие полосы прокрутки, возможностью задания более двух колонок, сортировкой элементов по алфавиту.

Элементы списка

Для обращения к элементам списка используется коллекция Items. Она также позволяет с помощью методов программно добавить или удалить элементы списка, а с помощью свойства Count определить общее число элементов списка.

Чтобы программно задать состояние флажка одного из элементов списка, применяются методы SetItemChecked и SetItemCheckState, имеющие следующий синтаксис:

SetItemChecked(ByVal index As Integer, ByVal value1 As Boolean) SetItemCheckState(ByVal index As Integer, ByVal value2 As CheckState)

где:

* index — номер элемента в списке;
* value1 — значение True позволяет установить флажок, а значение False — его сбросить;
* value2 — задает одно из трех возможных состояний флажка. Принимает значения перечисления CheckState: Checked (Установленный), Indeterminate (Неопределенный) и Unchecked (Сброшенный).

Для определения выбранных элементов списка, т. е. элементов с установленными флажками, служит коллекция CheckedItems элемента управления CheckedListBox. Например, с помощью приведенного далее кода можно при нажатии на кнопку удалить из списка все элементы с установленными флаж- ками. Для этого выполните следующие действия:

1. Создайте новое Windows-приложение и разместите на форме элемент управления CheckedListBox.
2. Используя свойство Items, задайте элементы списка.
3. Перетащите на форму элемент управления Button и присвойте значения

bRemove и Удалить его свойствам Name и Text соответственно.

1. Для задания процедуры обработки события нажатия кнопки дважды щелкните на кнопке Удалить. Откроется окно редактора кода с созданной процедурой bRemove\_Click. Добавьте в тело процедуры следующий код:

While Not CheckedListBox1.CheckedItems.Count = 0 CheckedListBox1.Items.Remove(CheckedListBox1.CheckedItems(0))

End While

При удалении выбранного элемента списка число элементов коллекции CheckedItems уменьшается на единицу. Поэтому для удаления всех элементов коллекции достаточно в цикле удалять лишь первый элемент.

Элемент управления NumericUpDown

Элемент управления NumericUpDown предназначен для ввода пользователем числовой информации и представляет собой текстовое поле и две кнопки с направленными в противоположные стороны стрелками (рисунок 5).



Каждое нажатие кнопки с направленной вверх стрелкой увеличивает, а нажатие кнопки с направленной вниз стрелкой — уменьшает размещенное в текстовом поле число на определенное значение.

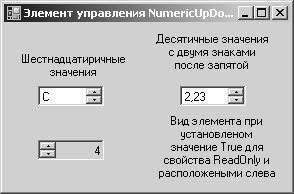


Рисунок 5 - Элемент управления NumericUpDown

Значения списка

Для определения значения списка используется свойство Value. С его по- мощью можно задать начальное значение, которое указывается в текстовом поле при запуске приложения.

Числовое значение текстового поля изменяется в заданных пределах. Для определения его максимального и минимального значений применяются свойства Minimum и Maximum. По умолчанию они принимают соответственно значения 0 и 100. Если задать минимальное значение больше максимального, то свойство Maximum автоматически примет значение свойства Minimum. Если же максимальное значение указано меньше минимального, то для свойства Minimum автоматически установится значение свойства Maximum.

С помощью свойства Increment задается шаг, с которым будет уменьшаться или увеличиваться числовое значение в текстовом поле элемента управления NumericUpDown при нажатии кнопки с направленной вниз или вверх стрелкой. По умолчанию указывается значение, равное 1.

Чтобы элемент управления отображал шестнадцатеричные значения вместо десятичных, необходимо задать значение True для свойства Hexadecimal.

Для определения числа десятичных разрядов, т. е. цифр после запятой, служит свойство DecimalPlaces. Оно по умолчанию принимает значение 0.

С помощью значения True свойства ThousandsSeparator можно задать отображение тысячного разделителя.

Значение True свойства ReadOnly элемента управления NumericUpDown запрещает пользователю вводить числа в текстовое поле и позволяет изменять значение списка лишь с помощью кнопок элемента управления NumericUpDown или клавиш <↑> и <↓>. Если требуется запретить использование клавиш-стрелок, нужно свойству InterceptArrowKeys присвоить значение False.

Для программного изменения значения списка предназначены методы UpButton и DownButton, соответствующие нажатию кнопок элемента управления NumericUpDown, увеличивающих и уменьшающих расположенное в текстовом поле число на определенное значение.

Внешний вид элемента управления

Для задания стиля обрамления элемента управления используется свойство BorderStyle, которое может принимать одно из следующих значений: Fixed3D (Объемная рамка), FixedSingle (Одномерная рамка) и None (Без рамки). По умолчанию элемент управления имеет трехмерную рамку.

Кнопки элемента управления NumericUpDown могут располагаться справа и слева от текстового поля. Для изменения расположения кнопок предназначено свойство UpDownAlign, принимающее значения Left (Слева) и Right (Справа).

Выровнять текст в текстовом поле элемента управления NumericUpDown можно по правому краю, левому краю или по центру с помощью соответствующих значений Right, Left или Center свойства TextAlign.

Элемент управления DomainUpDown

Элемент управления DomainUpDown предназначен для ввода пользователем текстовой информации и представляет собой текстовое поле и две кнопки с направленными в противоположные стороны стрелками (рисунок 6). Каждое нажатие кнопки с направленной вверх или вниз стрелкой позволяет перемещаться по списку заданных текстовых значений.

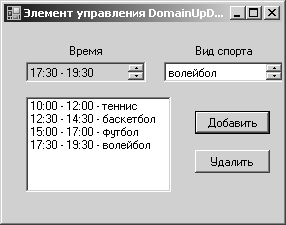


Рисунок 6 - Элемент управления DomainUpDown

Значения списка

Формирование списка во время проектирования осуществляется вручную с помощью свойства Items. Для добавления и удаления элементов списка программным способом предназначены методы коллекции Items элемента управления DomainUpDown. Данная коллекция также используется для обращения к элементам списка, а с помощью свойства Count позволяет определить их общее количество.

Элементы списка можно упорядочить по алфавиту. Для этого необходимо присвоить значение True свойству Sorted.

Для определения выбранного значения списка служат свойства SelectedItem и Text. С их помощью можно задать значение (необязательно являющееся элементом списка), которое указывается в текстовом поле при запуске приложения.

Для того чтобы можно было циклично перемещаться по списку, т. е. при достижении конца (начала) списка продолжать его просмотр с первого (последнего) элемента, необходимо задать значение True для свойства Wrap элемента управления DomainUpDown.

Значение True свойства ReadOnly элемента управления DomainUpDown запрещает пользователю вводить текст в поле и позволяет изменять значение списка лишь с помощью кнопок элемента управления DomainUpDown или клавиш <↑> и <↓>. Если требуется запретить использование клавиш-стрелок, нужно свойству InterceptArrowKeys присвоить значение False.

Внешний вид элемента управления

Для задания стиля обрамления элемента управления используется свойство BorderStyle, которое может принимать одно из следующих значений: Fixed3D (Объемная рамка), FixedSingle (Одномерная рамка) и None (Без рамки). По умолчанию элемент управления имеет трехмерную рамку.

Кнопки элемента управления DomainUpDown могут располагаться справа и слева от текстового поля. Для изменения расположения кнопок предназначено свойство UpDowmAlign, принимающее значения Left и Right.

Выровнять текст в поле элемента управления DomainUpDown можно с помощью свойства TextAlign, принимающего значения Right, Left и Center.

Пример

Рассмотрим небольшой пример, позволяющий составить расписание спортивных тренировок. Результат работы приложения показан на рисунке 6. Для его создания выполните следующие действия:

1. Создайте новое Windows-приложение и разместите на форме два элемента управления Label с текстом Время и Вид спорта.
2. Перетащите на форму два элемента управления DomainUpDown. Чтобы запретить ввод произвольного времени тренировки, присвойте значение True свойству ReadOnly первого элемента. Затем, используя свойство Items, задайте элементы списков.
3. Для отображения выбранных в списках значений воспользуйтесь элемен- том управления ListBox. Для этого расположите его на форме. Чтобы эле- менты списка сортировались по алфавиту, укажите значение True для свойства Sorted.
4. Разместите на форме две кнопки, которые позволят добавлять элементы в список и удалять их из него. Задайте текст и имена для кнопок.
5. Для обработки события нажатия кнопки, добавляющей элемент в список, щелкните дважды на ней и в созданную процедуру добавьте следующий код:

If (ListBox1.FindString(DomainUpDown1.Text) = ListBox1.NoMatches And Not DomainUpDown1.Text = "" And Not DomainUpDown2.Text = "") Then

ListBox1.Items.Add(DomainUpDown1.Text + " — " +

DomainUpDown2.Text)

End If

Данный код запрещает добавление элемента в список в случае существования записи с тем же временем.

1. Для обработки события нажатия кнопки, удаляющей выделенный элемент из списка, щелкните дважды на ней и в созданную процедуру добавьте следующие строки:

If Not ListBox1.SelectedIndex = ListBox1.NoMatches Then ListBox1.Items.RemoveAt(ListBox1.SelectedIndex)

End If

Приложение готово.

Задания для самостоятельной работы

**Создание пользовательской формы и элементов управления**

**Тема**: форма и элементы управления

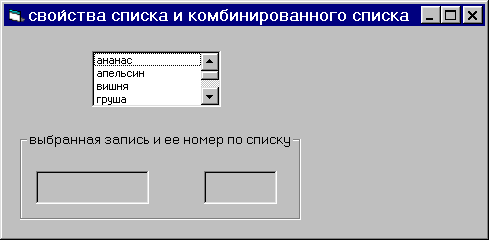
**Цель**: закрепить навыки работы с объектами **список** и **комбинированный список**.

**Порядок выполнения:**

**Часть 1.**

1. Установите на форме следующие объекты и измените им свойства:

|  |  |  |
| --- | --- | --- |
| **Объект** | **Свойство** | **Значение** |
| Рамка | Text | Выбранная запись и ее номер по списку |
|  | Font | Тип, размер, начертание шрифта (самостоятельно) |
| Метки | Name | Lbl1 |
| Text | (отсутствует) |
| Name | Lbl2 |
| Text | (отсутствует) |
| Список | Name | LstFruct |
| Items | Груша  Слива  Персик  Ананас  Вишня  Апельсин  Лимон  Мандарин |
| Sorted | True |
|  | BorderStyle | сначала FixedSingle; затем по умолчанию (Fixed3D) |

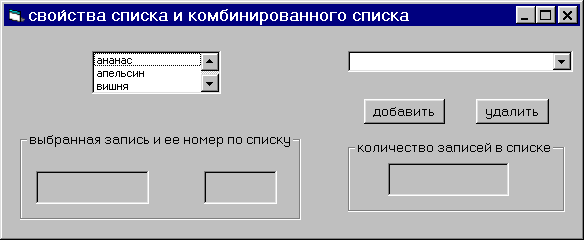
1.  Приведите свою форму в соответствие с образцом.
2. Самостоятельно измените значения свойств **Backcolor**, **ForeColor**, **Font** для объектов формы.
3. В окне кода процедуры обработки события **Click** для объекта **LstFruct** набрать с комментарием для каждой строки

**Lbl1.Text = LstFruct.SelectedIndex**

**Lbl2. = LstFruct.Text**

1. Проверьте работу созданной программы (обратите внимание, что список отсортирован по алфавиту, нумерация элементов списка начинается с нуля).
2. Добавьте на форму следующие объекты и измените им свойства:

|  |  |  |
| --- | --- | --- |
| **Объект** | **Свойство** | **Значение** |
| Рамка | Text | Количество записей в списке |
|  | Font | Тип, размер, начертание шрифта (самостоятельно) |
| Метка | Name | Lbl3 |
| Text | (отсутствует) |
| Раскрывающийся список | Name | CboProduct |
| Text  Items | (отсутствует)  Хлеб  Молоко  Сыр  Сметана  Масло  Творог  Йогурт |
| Sorted | False |
|  | DropDownWidth DropDownHeight  MaxDropDownItems | сначала 500; затем по умолчанию (по ширине списка)  сначала 300; затем по умолчанию (106)  сначала 5; затем по умолчанию (8) |
|  | DropDownStyle | DropDown |
| Командные кнопки | Name  Text | CmdAdd  Добавить |
|  | Name | CmdDel |
|  | Text | Удалить |

1. Приведите свою форму в соответствие с образцом
2. В окне кода процедуры обработки события **Click** для объекта **CmdAdd** набрать

**CboProduct.Items.Add (CboProduct.Text)** ‘добавляет элемент в список

**Lbl3.Text = CboProduct.Items.Count** ‘определяет число элементов списка

1. В окне кода процедуры обработки события **Click** для объекта **CmdDel** набрать

**n = CboProduct.SelectedIndex**

**CboProduct.Items.RemoveAt (n)**

**Lbl3. Text = CboProduct.Items.Count**

1. Проверить работу программы и результат показать преподавателю.
2. Ответить на вопрос: почему элементы раскрывающегося списка добавляются в конец списка?

Заключение

Изучив данную тему Вы узнали, как размещать на форме элементы управления, относящиеся к спискам, добавлять, выделять и удалять элементы списка. Теперь дело только за практикой.

Список использованных источников

1. Дукин А.Н. Самоучитель Visual Basic 2010 / А.Н. Дукин, А.А. Пожидаев. - СПб.: БХВ-Петербург, 2010. - 560 с.
2. Зиборов В.В. VisualBasic 2012 на примерах. - СПб.: БХВ-Петербург, 2013. - 448 с.
3. Федорова Г.Н. Разработка модулей программного обеспечения для компьютерных систем: учебник для студ. учреждений сред. проф. образования. / Г.Н. Федорова. – 3-е изд., испр. - М.: Издательский центр «Академия», 2019. – 384 с.