Лабораторная работа №11

ХАРЬКОВСКИЙ НАЦИОНАЛЬНЫЙ АВТОМОБИЛЬНО-ДОРОЖНЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И МЕХАТРОНИКИ

Кафедра информационных технологий и мехатроники

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по проведению лабораторных работ по дисциплине
«Объектно-ориентированное программирование»
для студентов специальности 6.050101 “Компьютерные науки”

Разработчик - доцент кафедры информационных технологий и мехатроники
кандидат технических наук, старший научный сотрудник
Тимонин Владимир Алексеевич

Харків 2015

Лабораторная работа №11
Исследование возможностей ИСР Visual C# для создания приложений с использованием управляющих элементов.

Цель работы – исследовать возможностей интегрированной среды разработки Visual C# и получить практические навыки по созданию приложений с использованием управляющих элементов.

1. Теоретические сведения

В библиотеке визуальных компонентов Visual C# существует множество управляющих компонентов. Простейшей и часто используемым компонентом является кнопка Button (компонент рассмотрен в лабораторной работе «Исследование возможностей Visual C# для создания приложений, использующие простые компоненты ввода и отображения информации»).

1.1. Компонент CheckBox

Компонент **CheckBox** представляет флажок, который может нахо диться в одном из двух состояний: выбранном или невыбранном. Часто вме сто "выбранный" говорят "установленный", а вместо "невыбранный" — "сброшенный" или "выключенный". Рядом с флажком, как правило, находит ся поясняющий текст. Компоненты **CheckBox** обычно используют для выбора нескольких опций из ряда возможных. Свойства компонента **CheckBox** приведены в табл. 1.

Таблица 1. Свойства компонента CheckBox

Свойство	Описание
Text	Текст, поясняющий назначение флажка
Checked	Состояние (вид) флажка. Если флажок выбран, то значение свойства равно **True**. Если флажок сброшен, то значение свойства равно **False**.
CheckState	Устанавливает трех видовое состояние: флажок включен – **Checked,** флажок выключен – **Unchecked,** состояние неопределенности (квадрат зеленого цвета) – **Indeterminate.**
ThreeState	Задает поддержку двух (**ThreeState=false**) или трех (**ThreeState=true**) состояний
TextAlign	Положение текста в поле отображения текста. Текст может располагаться в центре поля (**MiddleCenter**), быть прижат к левой (**MiddleLeft**) или правой (**MiddleRight**) границе. Можно задать и другие способы размещения текста надписи (**TopLeft, TopCenter, TopRight, BottomLeft, BottomCenter, BottomRight**)
CheckAlign	Положение флажка в поле компонента. Флажок может быть прижат к левой верхней границе (**TopLeft**), прижат к левой границе и находиться на равном расстоянии от верхней и нижней границ поля ком понента (**MiddleLeft**). Есть и другие варианты размещения флажка в поле компонента
Enabled	Свойство позволяет сделать флажок недоступным (**False**)
Visible	Свойство позволяет скрыть (**False**) флажок
AutoCheck	Свойство определяет, должно ли автоматически изменяться состоя ние флажка в результате щелчка на его изображении. По умолчанию значение равно **True**
FlatStyle	Стиль (вид) флажка. Флажок может быть обычным (**Standart**), плоским (**Flat**), "всплывающим" (**Popup**) или системным (**System**). Стиль определяет поведе ние флажка при позиционировании указателя мыши на его изобра жении
Appearance	Определяет вид флажка. Флажок может выглядеть обычным образом (**Normal**) или как кнопка (**Button**)
Image	Картинка, которая отображается в поле компонента
ImageAlign	Положение картинки в поле компонента. Картинка может распола гаться в центре (**MiddleCenter**), быть прижата к левой (**MiddleLeft**) или правой (**MiddleRight**) границе. Можно задать и другие способы размещения картинки на кнопке (**TopLeft,**

	TopCenter, TopRight, BottomLeft, BottomCenter, BottomRight)
ImageList	Набор картинок, используемых для обозначения различных состоя ний кнопки. Представляет собой объект типа **ImageList**. Чтобы задать значение свойства, в форму приложения следует добавить компонент **ImageList**
ImageIndex	Номер (индекс) картинки из набора **ImageList**, которая отображается в поле компонента

Состояние флажка изменяется в результате щелчка на его изображении (если значение свойства **AutoCheck** (равно **True**). При этом возникает событие **CheckedChanged**, а потом событие **Click**. Если значение свойства **AutoCheck** равно **False**, то в результате щелчка на флажке возникает событие **Click**, а затем, если процедура обработки этого события изменит состояние кнопки, возникает событие **CheckedChanged.**

Пример 1. Создадим приложение "Form1", демонстрирующие возможности компонента **CheckBox**. Приложение (форма приведена на рис. 1, вид приложения – на рис. 2, а текст — в листинге 1) отображает прямоугольники (панели) соответствующего цвета.

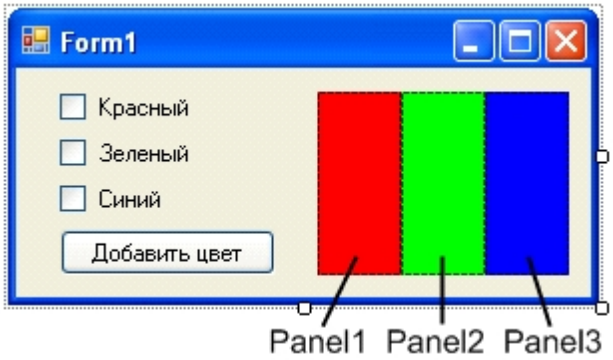

Рис. 1. Вид формы приложения

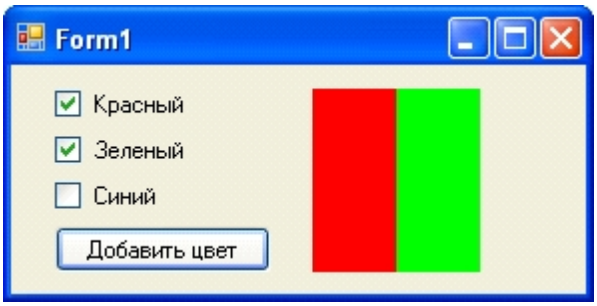

Рис. 2. Вид приложения

Листинг 1. Обработка события Click кнопки «Добавить цвет»

```
private void button1_Click(object sender, EventArgs e)
    {
      if (checkBox1.Checked == true) panel1.Visible = true;
      else panel1.Visible = false;
      if (checkBox2.Checked == true) panel2.Visible = true;
      else panel2.Visible = false;
      if (checkBox3.Checked == true) panel3.Visible = true;
      else panel3.Visible = false;
    }
```

1.2. Компонент RadioButton

Компонент **RadioButton** представляет собой переключатель, со стояние которого зависит от состояния других переключателей (компонентов **RadioButton**). Обычно компоненты **RadioButton** объединяют в группу (дости гается это путем размещения нескольких компонентов в поле компонента **GroupBox**). В каждый момент времени только один из переключателей группы может находиться в выбранном состоянии (возможна ситуация, когда ни один из переключателей не выбран). Состояние компонентов, принадлежа щих одной группе, не зависит от состояния компонентов, принадлежащих другой группе. Свойства компонента приведены в табл. 2.

Таблица 2. Свойства компонента RadioButton

Свойство	Описание

Text	Текст, который находится справа от переключателя
Checked	Состояние, внешний вид переключателя. Если переключатель выбран, то значение свойства **Checked** равно **True**; если не выбран, то значение свойства **Checked** равно **False**
TextAlign	Положение текста в поле отображения текста. Текст может располагаться в центре поля (**MiddleCenter**), быть прижат к левой (**MiddleLeft**) или правой (**MiddleRight**) границе. Можно задать и другие способы размещения текста надписи (**TopLeft, TopCenter, TopRight, BottomLeft, BottomCenter, BottomRight**)
CheckAlign	Положение переключателя в поле компонента. Переключатель может быть прижат к левой верхней границе (**TopLeft**), прижат к левой границе и находиться на равном расстоянии от верхней и нижней границ поля компонента (**MiddleLeft**). Есть и другие варианты размещения переключателя в поле компонента
Enabled	Свойство позволяет сделать переключатель недоступным (**False**)
Visible	Свойство позволяет скрыть (**False**) переключатель
AutoCheck	Свойство определяет, должно ли автоматически изменяться со стояние переключателя в результате щелчка на его изображении. По умолчанию значение равно **True**
FlatStyle	Стиль переключателя. Переключатель может быть обычным (**Standart**), плоским (**Flat**) или "всплывающим" (**Popup**). Стиль переключателя определяет его поведение при позиционировании указателя мыши на изображении переключателя
Appearance	Определяет вид переключателя. Переключатель может выглядеть обычным образом (**Normal**) или как кнопка (**Button**)
Image	Картинка, которая отображается в поле компонента
ImageAlign	Положение картинки в поле компонента. Картинка может распола гаться в центре (**MiddleCenter**), быть прижатой к левой (**MiddleLeft**) или правой (**MiddleRight**) границе. Можно задать и другие способы размещения картинки на кнопке (**TopLeft, TopCenter, TopRight, BottomLeft, BottomCenter, BottomRight**)
ImageList	Набор картинок, используемых для обозначения различных состоя ний переключателя. Представляет собой объект типа **ImageList**. Чтобы задать значение свойства, в форму приложения надо доба вить компонент **ImageList**
ImageIndex	Номер (индекс) картинки из набора **ImageList**, которая отобража ется в поле компонента

Состояние переключателя изменяется в результате щелчка на его изображе нии (если значение свойства **AutoCheck** равно **True**). При этом возникает собы тие **CheckedChanged**, а затем событие **Click**. Если значение свойства **AutoCheck** равно **False**, то в результате щелчка на переключателе возникает событие **Click**, а затем, если процедура обработки этого события изменит состояние кнопки, возникает событие **CheckedChanged**.

 **Пример 2.** Создадим приложение "Form1", демонстрирующие возможности компонента **RadioButton**. Приложение (форма приведена на рис. 3, вид приложения – на рис. 4, а текст — в листинге 2) закрашивает прямоугольник (панель) в соответствии с выбранным цветом.

Рис. 3. Вид формы приложения

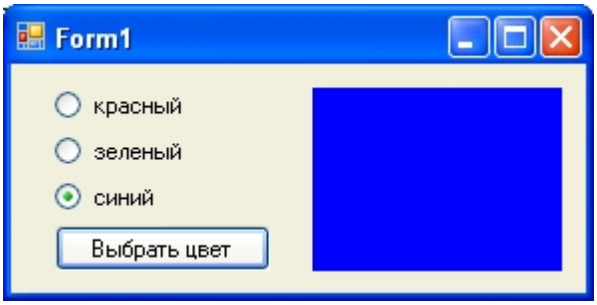

Рис. 4. Вид приложения

Листинг 2. Обработка события Click кнопки «Выбрать цвет»

```
private void button1_Click(object sender, EventArgs e)
    {
      if (radioButton1.Checked == true) panel1.BackColor = Color.Red;
      else if (radioButton2.Checked == true) panel1.BackColor = Color.Green;
      else if (radioButton3.Checked == true) panel1.BackColor = Color.Blue;
    }
```

1.3. Компонент GroupBox

Довольно часто вместо отдельных переключателей используют их групповой контейнер–компонент **GroupBox**. Компонент **GroupBox** представляет собой контейнер для других компонентов. Обычно он используется, во-первых, чтобы обеспечить разделение компонентов на различные группы, которые становятся для них родителями, с тем, чтобы компоненты унаследовали некоторые свойства своих родителей (обычно так делают, чтобы подразделить форму на несколько функций), и во-вторых, для объединения в группы компонентов **RadioButton** по функциональному признаку. Свойства компонента СгоирВох приведены в табл. 4.

Таблица 4. Свойства компонента GroupBox

Свойство	Описание
Dock	Определяет границу формы, к которой привязана (прикреплена) панель. Панель может быть прикреплена к левой (**Left**), правой (**Right**), верхней (**Top**) или нижней (**Bottom**) границе
Text	Заголовок — текст, поясняющий назначение компонентов, которые находятся в поле компонента **GroupBox**
Enabled	Позволяет управлять доступом к компонентам, находящимся в поле (на поверхности) компонента **GroupBox**. Если значение свойства равно **False**, то все находящиеся в поле **GroupBox** компоненты недоступны
Visible	Позволяет скрыть (сделать невидимым) компонент **GroupBox** и все компоненты, которые находятся на его поверхности
Location	Положение компонента на поверхности формы
FlatStyle	Стиль компонента: может быть обычным (**Standart**), плоским (**Flat**), "всплывающим" (**Popup**) или определяется системой (**System**)

Пример 3. Создадим приложение "Выбор", демонстрирующие возможности компонент **GroupBox, CheckBox** и **RadioButton**. Приложение (форма приведена на рис. 5, вид приложения – на рис. 6, а текст — в листинге 3) закрашивает прямоугольник (панель) в соответствии с выбранным цветом.

Рис. 5. Вид формы приложения

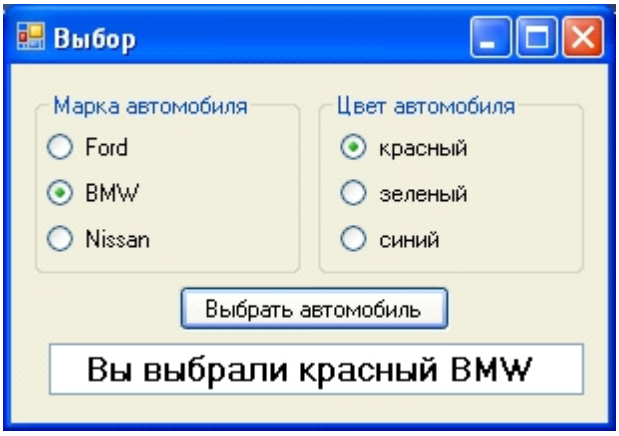

Рис. 6. Вид приложения

Листинг 3. Обработка события Click кнопки «Выбрать автомобиль»

```
private void button1_Click(object sender, EventArgs e)
    {
      string Str = "   Вы выбрали ";
      if (radioButton4.Checked == true) Str = Str + radioButton4.Text + " ";
      else if (radioButton5.Checked == true) Str = Str + radioButton5.Text + " ";
           else if (radioButton6.Checked == true) Str = Str + radioButton6.Text + " ";
      if (radioButton1.Checked == true) Str = Str + radioButton1.Text + " ";
      else if (radioButton2.Checked == true) Str = Str + radioButton2.Text + " ";
           else if (radioButton3.Checked == true) Str = Str + radioButton3.Text + " ";
      textBox1.Text = Str;
    }
```

1.4. Компонент CheckedListBox

Компонент **CheckedListBox** является расширением **ListBox**. Он делает почти все, что делает **ListBox**, но дополнительно выводит окна контроля (флажки-переключатели), в которых можно делать отметку галочкой. Вид компонента показан на рис. 7. Компонент **CheckedListBox** представляет собой список, перед каж дым элементом которого находится переключатель**CheckBox**. Пользователь может помечать элементы списка, щелкая мышью на одной или нескольких позициях (устанавливать флажок). Повторный щелчок снимает включение флажка. Свойства ком понента **CheckedListBox** приведены в табл. 3.

Таблица 3. Свойства компонента CheckedListBox

Свойство	Описание
Items	Элементы списка — коллекция строк
Items.Count	Количество элементов списка
Sorted	Признак необходимости автоматической сортировки (**True**) списка после добавления очередного элемента
CheckOnClick	Способ пометки элемента списка. Если значение свойства равно **False**, то первый щелчок выделяет элемент списка (строку), а второй устанавливает в выбранное состояние переключатель. Если значение свойства равно **True**, то щел чок на элементе списка выделяет элемент и устанавливает во включенное состояние переключатель
CheckedItems	Свойство **CheckedItems** представляет собой коллекцию, элементы которой содержат выбранные элементы списка
CheckedItems.Count	Количество выбранных элементов списка, переключатели которых установлены в выбранное состояние
CheckedIndices	Свойство **CheckedIndices** представляет собой коллекцию, элементы которой

	содержат номера выбранных (помеченных) элементов списка
ScrollAlwaysVisible	Признак необходимости всегда отображать вертикальную полосу прокрутки. Если значение свойства равно **False**, то полоса прокрутки отображается, только если все элементы списка нельзя отобразить при заданном размере компонента
MultiColumn	Признак необходимости отображать список в несколько колонок. Количество отображаемых колонок зависит от количества элементов и размера области отображения списка (рис. 8)
ColumnWidth	Указывает, какой должна быть ширина столбца в **ListBox** с несколькими столбцами
ThreeDCheckBoxes	Определяет стиль окна флажка (стиль **Flat** – значение свойства равно **True**, стиль **Normal** - значение свойства равно **False**)
Location	Положение компонента на поверхности формы
Size	Размер компонента без (для компонентов типа **DropDown** и **DropDownList**) или с учетом (для компонента типа **Simple**) размера области списка или области ввода
Font	Шрифт, используемый для отображения содержимого поля редактирования и элементов списка

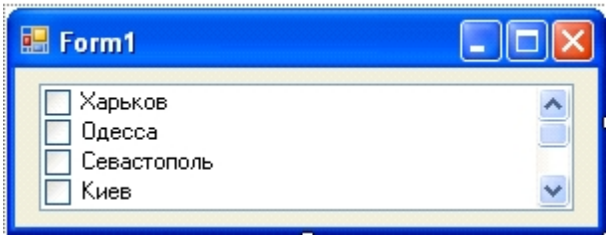

Рис. 7. Вид компонента **CheckedListBox** в одноколончатом варианте

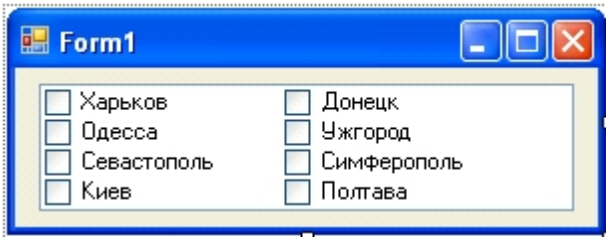

Рис. 8. Вид компонента **CheckedListBox** в многоколончатом варианте

Выбор элемента списка еще не означает, что флажок устанавливается/снимается. Существует свойство **CheckOnClick**, которое разрешает/запрещает делать пометку (устанавливать/снимать флажок). Если это свойство установлено в **False**, то при щелчке мышью на позиции галочка (флажок) в ней не появляется. Но одновременно со щелчком на позиции идет подсветка строки, повторно щелкнуть на отмеченной строке, то галочка появится. Таким образом, при значении свойства **CheckOnClick**, установленным в **False**, для включения флажка надо сначала отметить элемент списка (щелчком на строке), потом сделать повторный щелчок. А выключается флажок при щелчке на строке или на нем самом. Если же **CheckOnClick** установлено в **True**, то флажок включается одновременно с выбором элемента (и выключается при повторном щелчке на нем).

Компонент **CheckedListBox** поддерживает три состояния флажка (**Checked** — флажок включен; **Unchecked** — флажок выключен; **Indeterminate** — состояние неопределенности (флажок закрашен серым цветом). Такое состояние можно устанавливать только в режиме испол нения, т. е. программно).

Существуют методы **CheckedListBox,** с помощью которых можно определять и устанавливать состояние флажка:

- GetItemCheckState(int index);

- SetItemCheckState(int index, CheckState value);

где **index** — это индекс того элемента, состояние флажка которого определяется; **value** — значение состояния флажка. Состояние определено классом **CheckState** и имеет три значения (**Checked**, **Unchecked**, **Indeterminate** — им соответствуют числовые значения 1, 0 и 2).

Пример 4. Создадим приложение "Список", демонстрирующие возможности компонента **CheckedListBox** при выполнении программы устанавливать флажки (форма приведена на рис. 8, вид приложения – на рис. 9, а текст — в листинге 4).

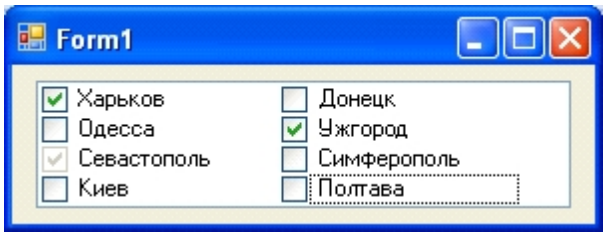

Рис. 9. Программно установленные флажки

Листинг 4. Обработка события DoudleClick

```
private void checkedListBox1_DoubleClick(object sender, EventArgs e)
    {
      checkedListBox1.SetItemCheckState(0, CheckState.Checked);
      checkedListBox1.SetItemCheckState(1, CheckState.Unchecked);
      checkedListBox1.SetItemCheckState(2, CheckState.Indeterminate);
      checkedListBox1.SetItemCheckState(5, CheckState.Checked);
      int i = (int)checkedListBox1.GetItemCheckState(0);
    }
```

Программа задаёт состояния для флажков 1, 2, 3 и 6-й строк. Последний оператор определяет состояние 1-й строки. Так как тип результата, выдаваемого методом **GetItemCheckState(0)**, **CheckState**, то чтобы увидеть состояние, надо этот тип привести к типу **int** принудительно. Поэтому после знака присвоения стоит **(int)**

Чтобы узнать, помечен ли элемент списка, можно выполнить оператор:

```
bool z = checkedListBox1.GetItemChecked(i);
```

Результат метода **GetItemChecked()** — логическая переменная, **i** — индекс элемента.

Можно и другим способом установить флажок. Для этого следует выполнить оператор:

```
checkedListBox1.SetItemChecked(3, true);
```

где **3** — индекс элемента (4-я строка).

Сформировать список компонента **CheckedListBox** можно во время работы программы, применив метод **Add()** к свойству **Items**. В инструкции вызова ме тода **Add()** в качестве второго параметра можно указать константу **True**. В этом случае переключатель перед элементом списка будет установлен в выбранное состояние.

2. Рабочее задание

Задание 1. Разработать приложение «Операции в C#» для исследования результатов действий арифметических и логических операций. При разработке приложения предусмотреть выдачу аварийного сообщения об отсутствии исходных данных или невыбранной операции.

При разработке интерфейса приложения использовать компоненты **Label, Button, Panel, RadioButton** и **GroupBox** (один из вариантов интерфейса представлен на рис. 10).

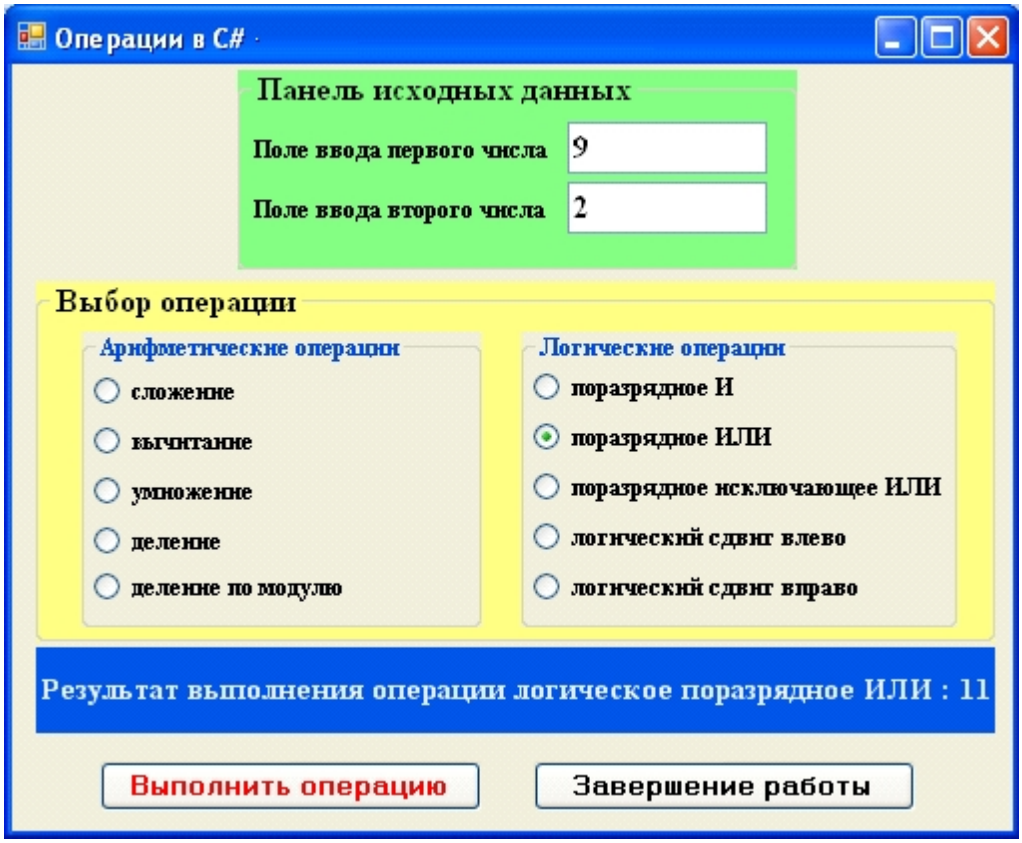

Рис. 10. Внешний вид приложения «Операции в C#»

Задание 2. Разработать приложение «Заказ комплектующих компонентов ПК», которое позволяет выбирать комплектующие компоненты ПК (типы процессоров, чипсетов и видеокарт) для формирования заказа. В соответствии с выбранными компонентами по щелчку кнопки «Сделать заказ» выдается сообщение о сделанном заказе.

При разработке интерфейса приложения использовать компоненты **Label, Panel, Button, CheckBox, RadioButton** и **GroupBox** (один из вариантов интерфейса представлен на рис. 11).

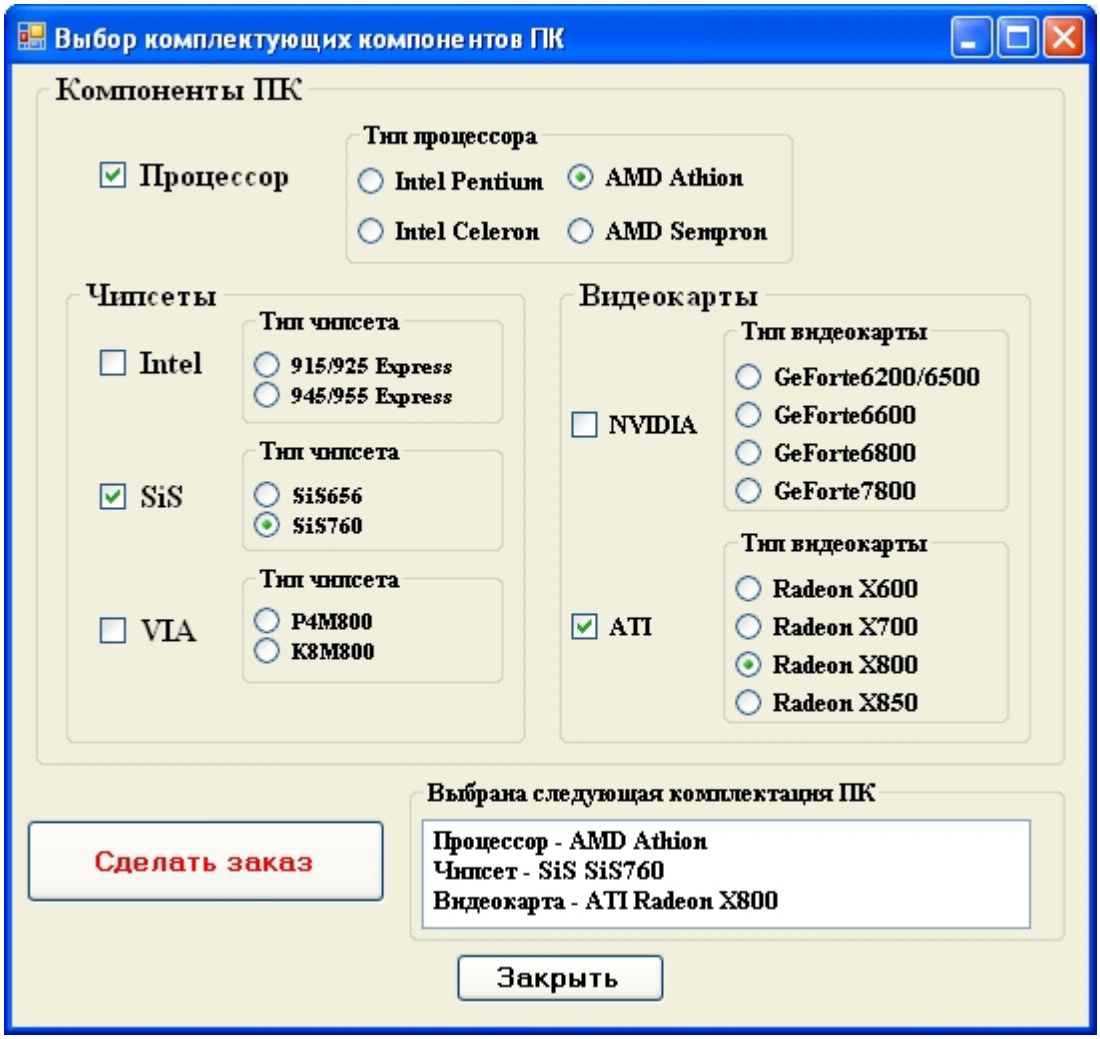

Рис. 11. Внешний вид приложения «Выбор комплектующих компонентов ПК»

3. Контрольные вопросы

Литература

1. Голощапов А.Л. Microsoft Visual Studio 2010. – СПб.:БХВ-Петербург, 2011. – 544 с.: ил.
2. Культин Н.Б. Microsoft Visual C# в задачах и примерах. – СПб.: БХВ-Петербург, 2009. – 320 с.: ил.
3. Лабор В.В. Си Шарп: Создание приложений для Windows. – Мн.: Харвест, 2003. – 384 с.
4. Петцольд Ч. Программирование для Microsoft Windows на C#. В 2-х томах. Том 1. Пер. с англ. - М.: «Русская Редакция», 2002.- 576 с.: ил.
5. Петцольд Ч. Программирование для Microsoft Windows на C#. В 2-х томах. Том 2. Пер. с англ. - М.: «Русская Редакция», 2002.- 624 с.: ил.
6. Троелсен Э. Язык программирования C# 2010 и платформа .NET 4.0. Пер. с англ. - М.: Издательский дом "Вильямс", 2011. — 1392 с.: ил.
7. Фаронов В.В. Программирование на языке C#. – СПб.: Питер, 2007. – 240 с.: ил.
8. Фленов М.Е. Библия C#. - СПб.: БХВ-Петербург, 2011.– 560с.: ил.