Лабораторная работа №9

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 №9
Исследование возможностей ИСР Visual C# для создания приложений, использующие простые компоненты ввода и отображения текстовой информации.

Цель работы – исследовать возможностей интегрированной среды разработки Visual C# и получить практические навыки по созданию приложений, использующие простые компоненты ввода и отображения информации.

1. Теоретические сведения

Microsoft Visual Studio, являясь средой быстрой разработки, поддерживает технологию визуального проектирования, идея которой заключается в том, что среда разработки предоставляет в распоряжение программиста набор компонентов, которые можно поместить на форму.

Компонент — это объект, предназначенный для решения некоторой задачи. Компоненты обеспечивают ввод данных, отображение результатов (такие компоненты называют компонентами пользовательского интерфейса), доступ к базам данных, решение других задач.

1.1. Компонент Label

Компонент **Label** предназначен для отображения текстовой инфор мации. Задать текст, отображаемый в поле компонента, можно как во время разработки формы, так и во время работы программы, присвоив значение. Свойства компонента приведены в табл. 1.

Таблица 1. Свойства компонента Label

Свойство	Описание
Name	Имя (идентификатор) компонента. Используется в программе для доступа к компоненту
Text	Отображаемый текст
Location	Положение компонента на поверхности формы
Size	Размер компонента (области отображения текста)
Font	Шрифт, используемый для отображения текста
AutoSize	Включает автоматическое изменение размера в соответствии с размером шрифта. Чтобы увидеть длинный текст в метке, надо отключить свойство (значение равно **False**). После чего появятся анкерные точки, за которые поле метки можно растягивать или сжимать
ForeColor	Цвет текста, отображаемого в поле компонента
BackColor	Цвет закраски области вывода текста
TextAlign	Способ выравнивания (расположения) текста в поле компонента. Всего существует девять способов расположения текста. На практике наиболее часто используют выравнивание по левой и верхней границам (**TopLeft**), посередине (**TopCenter**) и по центру (**MiddleCenter**)
BorderStyle	Вид рамки (границы) компонента. По умолчанию граница вокруг поля **Label** отсутствует (значение свойства равно **None**). Граница компонента может быть объемной (**Fixed3D**) или тонкой (**FixedSingle**)

Чтобы в поле компонента **Label** вывести числовое значение, это значение на до преобразовать в строку. Сделать это можно при помощи метода **ToString()**.

Цвет текста (**ForeColor**) и фона (**BackColor**) можно задать, указав название цвета (**Color.Red**, **Color.Blue**, **Color.Green** и т.д.) или элемент цветовой схемы операционной системы (**System.Drawing.SystemColors.Control**, **System.Drawing.SystemColors.ControlText** и т.д.). Во втором случае цвет будет "привязан" к текущей цветовой схеме операционной системы и будет автоматически

меняться при каждой ее смене. По умолчанию для элементов управления используется второй способ кодирования цвета.

Цвет фона может быть "прозрачным" (**Color.Transparent**).

Пример 1. Создадим приложение "Конвертор", демонстрирующие возможности компонента **Label**. Приложение "Конвертор" (ее форма приведена на рис. 1, а текст функции обработки события **Click**, возникающего при щелчке на кнопке **ОК**— в лис тинге 1) показывает, как во время работы программы изменить цвет текста, отображаемого в поле компонента, как вывести в поле компонента значение переменной, а также как разбить отображаемый текст на строки. Программа пересчитывает цену из долларов в рубли. Если пользователь оставит какое-либо из полей неза полненным, то в результате щелчка на кнопке **ОК** в поле компонента **Label3** красным цветом отображается сообщение об ошибке. Если все поля формы заполнены, то в поле компонента **Label3** в две строки отображается результат расчета.

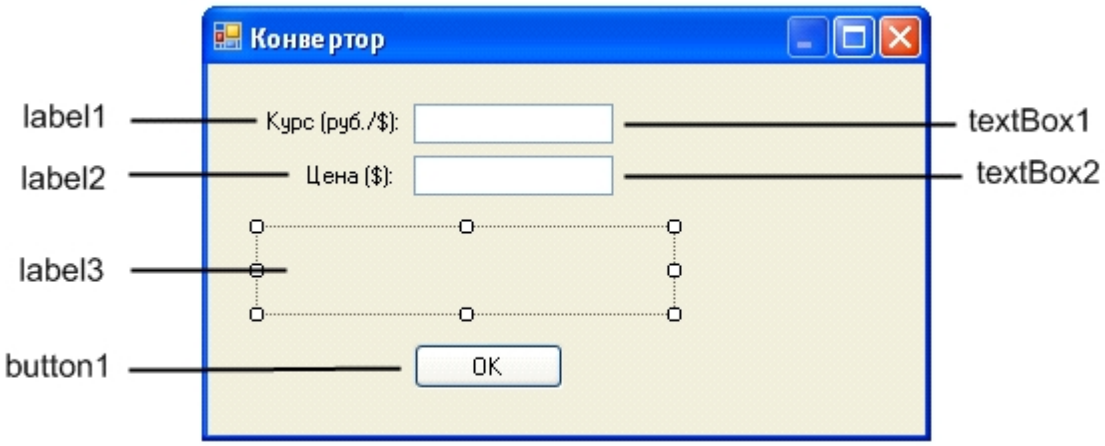

Рис. 1. Форма приложения «Конвертор»

Листинг 1. Обработка события Click

```
private void button1_Click(object sender, EventArgs e)
    {
      double usd;  // цена в долларах
      double k;    // курс
      double rub;  // цена в рублях
      if (( textBox1.Text  != "") && ( textBox2.Text != ""))
      {
        usd = Convert.ToDouble(textBox1.Text);
        k = Convert.ToDouble(textBox2.Text);
        rub = usd * k;
        label3.ForeColor = System.Drawing.SystemColors.ControlText;
        label3.Text = usd.ToString("n") + "$ = " + rub.ToString("c");
      }
      else
      {
        label3.ForeColor = Color.Red;
        label3.Text = "Надо ввести данные в оба поля";
      }
    }
```

1.2. Компонент Button

Компонент **Button** представляет собой командную кнопку. Свойства компонента приведены в табл. 2.

Таблица 2. Свойства компонента Button

Свойство	Описание
Name	Имя компонента. Используется в программе для доступа к свойствам компонента
Text	Текст (надпись) на кнопке
TextAlign	Положение текста (надписи) на кнопке. Надпись может располагаться в центре (**MiddleCenter**), быть прижата к левой (**MiddleLeft**) или правой (**MiddleRight**) границе. Можно задать и другие способы размещения надписи (**TopLeft**, **TopCenter**, **TopRight**, **BottomLeft**, **BottomCenter**, **BottomRight**)
FlatStyle	Кнопка может быть стандартной (**Standart**), плоской (**Flat**), "всплывающей" (**Popup**) или системной (**System**)
Location	Положение кнопки на поверхности формы. Уточняющее свойство X определяет расстояние от левой границы кнопки до левой границы формы, уточняющее свойство Y — от верхней границы кнопки до верх ней границы клиентской области формы (нижней границы заголовка)
Size	Размер кнопки
Font	Шрифт, используемый для отображения текста на кнопке
ForeColor	Цвет текста, отображаемого на кнопке
Enabled	Признак доступности кнопки. Кнопка доступна, если значение свойства равно **True**, и недоступна (например, событие **Click** в результате щелч ка на кнопке не возникает), если значение свойства равно **False**.
Visible	Позволяет скрыть кнопку (**False**) или сделать ее видимой (**True**)
Cursor	Вид указателя мыши при позиционировании указателя на кнопке
Image	Картинка на поверхности кнопки. Рекомендуется использовать gif-файл, в котором определен прозрачный цвет
ImageAlign	Положение картинки на кнопке. Значение свойства см. **TextAlign**.
ImageList	Набор картинок, используемых для обозначения различных состояний кнопки. Представляет собой объект типа **ImageList**. Чтобы задать значение свойства, в форму приложения надо добавить компонент **ImageList**.
ImageIndex	Номер (индекс) картинки из набора **ImageList,** которая отображается на кнопке. Это свойство связано со свойством **ImageList**, задающим ссылку на ряд изображений (пиктограмм), которые выбираются этим компонентом через диалоговое окно, открывающееся в поле этого свойства. Надо поместить **ImageList** в форму, тогда его имя станет видимым в поле свойства кнопки **ImageList**. Каждому изображению в списке присваивает ся свой порядковый номер (индекс), по которому впоследствии и можно будет выбирать нужное, или ключ, в качестве которого служит название файла-изображения. Свойство кнопки **ImageList** выбирается из списка (при открытии списка становятся видны все индексы, сформированные компонентом **ImageList**). Свойство кнопки **ImageKey** тоже формируется ; открытием списка значений, в качестве которых выступают имена файлов пиктограмм, сформированные компонентом **ImageList**. В рассматривае мом свойстве как раз и задается индекс нужного изображения из списка изображений. По нему в кнопке должна появиться пиктограмма.
TabIndex	В это свойство помещается сформированный средой про граммирования порядковый номер компонента в контейнере, например, в форме. Какие но мера имеют свойства **TabOrder**, в такой последовательности и станут ак тивизироваться (получать фокус ввода) компоненты в форме после за пуска приложения при последовательном нажатии клавиши **<Tab >.** Если надо изменить порядок активизации, то необходимо присвоить соответствующие значения свойствам компонентов**TabIndex**.

Пример 2. Создадим приложение " Мили-километры ", демонстрирующие возможности компонента **Button**. Программа «Мили-километры» (ее форма приведена на рис. 2, а текст— в листин ге 2) пере считывает расстояние из милей в километры. Расчет и отображение результата выполняет процедура обработки события **Click**. Следует обратить внимание, что кнопка **ОК** доступна только в том случае, если в поле редактирования находятся данные (хотя бы одна цифра). У правляет доступностью кнопки процедура обработки события **TextChanged** компонента **TextBox**. Процедура контролирует количество символов, которое находится в поле редактирова ния, и, если в поле нет ни одной цифры, присваивает значение **False** свойст ву **Enabled** и, тем самым, делает кнопку недоступной. В процессе создания формы свойству **Enabled** кнопки надо присвоить значение **False**.

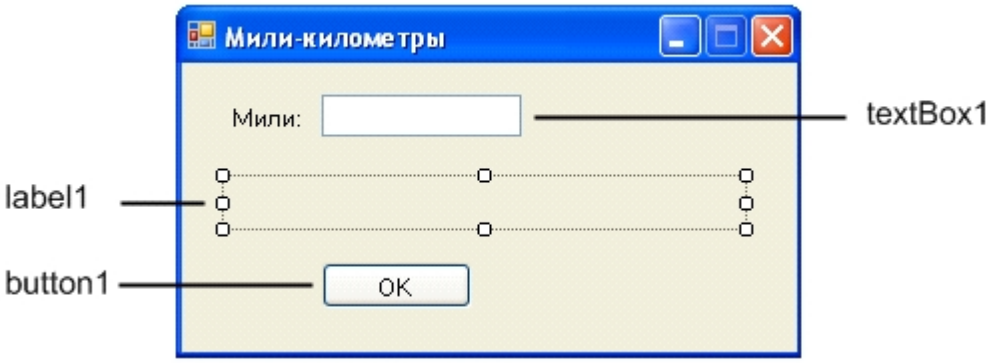

Рис. 2. Форма приложения «Мили-километры»

Листинг 2. Обработка события Click

```
// содержимое поля редактирования изменилось
private void textBox1_TextChanged(object sender, EventArgs e)
    {
      // если в поле texBox1 нет данных,  сделать кнопку button1 недоступной
      if (textBox1.Text.Length == 0)
        button1.Enabled = false;
      else
        button1.Enabled = true;
    }

// нажатие клавиши в поле textBox
private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
    {
      /* Правильными символами считаются цифры, запятая, <Enter> и
       <Backspace>. Будем считать правильным символом также точку, но
       заменим ее запятой. Остальные символы запрещены. Чтобы запрещенный
       символ не отображался в поле редактирования, присвоим  значение true
       свойству Handled параметра e */
      if ((e.KeyChar >= '0') && (e.KeyChar <= '9'))
        return; // цифра


      if (e.KeyChar == '.')
      {
        e.KeyChar = ','; // точку заменим запятой
      }

      if (e.KeyChar == ',')
```

```
        {
            if (textBox1.Text.IndexOf(',') != -1)
            {
                e.Handled = true; // запятая уже есть в поле редактирования
            }
            return;
        }
        if (Char.IsControl(e.KeyChar))
        { // <Enter>, <Backspace>, <Esc>
            if (e.KeyChar == (char)Keys.Enter)
                // нажата клавиша <Enter> установить "фокус" на кнопку OK
                button1.Focus();
            return;
        }
        // остальные символы запрещены
        e.Handled = true;
    }

// щелчок на кнопке OK
private void button1_Click(object sender, EventArgs e)
    {
        double mile; // расстояние в милях
        double km;   // расстояние в километрах
        mile = Convert.ToDouble(textBox1.Text);
        km = mile * 1.609344;
        label1.Text = mile.ToString("n") + " miles - " +  km.ToString("n") + " км.";
    }
```

На кнопке может находиться картинка. Существуют два способа поместить картинку на кнопку. Первый— присвоить значение свойству **Image**. Вто рой — добавить в форму компонент **ImageList**, выполнить его настройку, ус тановить связь между компонентами **Button** и **ImageList** и затем, присвоив значение свойству **ImageIndex** компонента **Button**, задать картинку для кноп ки. Первый способ технически проще, но при его использовании нельзя за дать прозрачный цвет, поэтому картинка должна быть прямоугольной или цвет фона изображения должен совпадать с цветом кнопки. При использова нии второго способа есть возможность задать прозрачный цвет.

1.3. Компонент TextBox

Компонент **TextBox** предназначен для ввода данных с клавиатуры. В зависимости от настройки компонента, в поле редактирования можно вве сти одну или несколько строк текста. Свойства компонента приведены в табл. 3.

Если не предпринимать никаких усилий, то в поле компонента **TextBox** ото бражаются все символы, которые пользователь набирает на клавиатуре, что не всегда удобно. Программа может обеспечить фильтрацию вводимых сим волов путем запрета отображения запрещенных символов. Для того чтобы символ не отображался в поле редактирования, в процедуре обработки собы тия **KeyPress** параметру **Handled** надо присвоить значение **True**.

Таблица 3. Свойства компонента TextBox

Свойство	Описание
Name	Имя компонента. Используется в программе для доступа к компо ненту и его свойствам, в том числе к тексту, который находится в поле редактирования
Text	Текст, который находится в поле редактирования
Location	Положение компонента на поверхности формы

Size	Размер компонента
Font	Шрифт, используемый для отображения текста в поле компонента
ForeColor	Цвет текста, находящегося в поле компонента
BackColor	Цвет фона поля компонента
BorderStyle	Вид рамки (границы) компонента. Граница компонента может быть объемной (**Fixed3D**), тонкой (**FixedSingle**). Граница вокруг компо нента может отсутствовать (в этом случае значение свойства равно **None**)
TextAlign	Способ выравнивания текста в поле компонента. Текст в поле ком понента может быть прижат к левой границе компонента (**Left**), правой (**Right**) или находиться по центру (**Center**)
MaxLength	Максимальное количество символов, которое можно ввести в поле компонента
PasswordChar	Символ, который используется для отображения вводимых пользо вателем символов (введенная пользователем строка находится в свойстве **Text**)
MultiLine	Разрешает (**True**) или запрещает (**False**) ввод нескольких строк текста
Dock	Способ привязки положения и размера компонента к размеру формы. По умолчанию привязка отсутствует (**None**). Если значение свойства равно **Top** или **Bottom**, то ширина компонента устанавли вается равной ширине формы и компонент прижимается соответст венно к верхней или нижней границе формы. Если значение свой ства **Dook** равно **Fill**, а свойства **Multiline** — **True**, то размер компонента устанавливается максимально возможным.
Lines	Массив строк, элементы которого содержат текст, находящийся в поле редактирования, если компонент находится в режиме **Multiline**. Доступ к строке осуществляется по номеру. Строки нумеруются с нуля
ScrollBars	Задает отображаемые полосы прокрутки: **Horizontal** — горизон тальная; **Vertical** — вертикальная; **Both** — горизонтальная и вер тикальная; **None** — не отображать.
WordWrap	Указывает о возможности выполнения автоматического переноса на новую строку в многострочных полях редактирования

Пример 3. Создадим приложение "TextBox", демонстрирующие возможности компонента **TextBox** для ввода данных различного типа. Приложение (форма приведена на рис. 3, а текст — в листинге 3) спроектировано таким образом, что в режиме ввода текста в поле редактирования можно ввести любой символ, в режиме ввода целого числа— только цифры. В режиме ввода дробного числа кроме цифр в поле компонента можно ввести символ-разделитель (за пятую или точку, в зависимости от настройки операционной системы).

При написании кода необходимо подключить в перечень пространств имен пространство **System.Globalization**. Это делается директивой:

using System.Globalization;

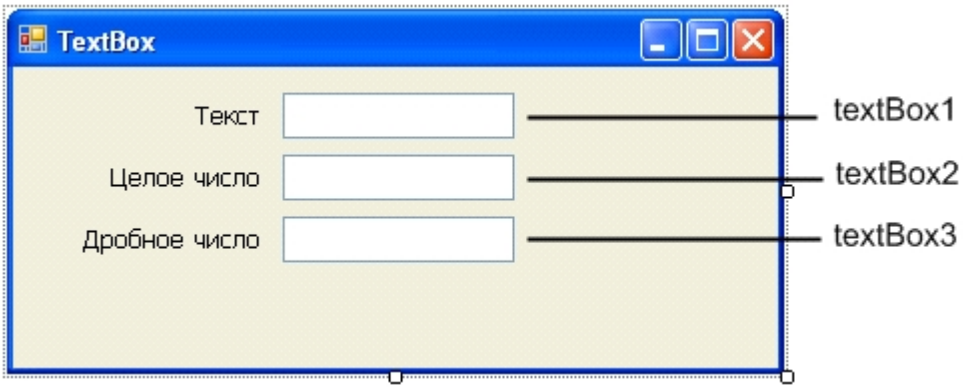

Рис. 3. Форма приложения «TextBox»

Листинг 1. Обработка события Click

```
// текст
private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
      {
            if (Char.IsControl(e.KeyChar))
            {
               if (e.KeyChar == (char)Keys.Enter)
               { // нажата клавиша <Enter> переместить фокус в поле textBox2
                  textBox2.Focus();
               }
            }
      }

// целое число
private void textBox2_KeyPress(object sender, KeyPressEventArgs e)
      {
         // в textBox2 можно ввести только целое число
         if ((e.KeyChar >= '0') && (e.KeyChar <= '9'))
            return; // цифра
         if (Char.IsControl(e.KeyChar))
         {
            if (e.KeyChar == (char)Keys.Enter)
            { // нажата клавиша <Enter> переместить фокус в поле textBox3
               textBox3.Focus();
            }
            return;
         }
         // остальные символы запрещены
         e.Handled = true;
      }

// дробное число
private void textBox3_KeyPress(object sender, KeyPressEventArgs e)
      {
         // CultureInfo.CurrentCulture.NumberFormat.NumberDecimalSeparator - строка.
         // нам нужен первый символ
         char aDecimalSeparator =
                  CultureInfo.CurrentCulture.NumberFormat.NumberDecimalSeparator[0];
         if ((e.KeyChar >= '0') && (e.KeyChar <= '9'))
            return; // цифра
         /* Заменим введенный десятичный разделитель (точку или запятую)  на
           "правильный" – CultureInfo.CurrentCulture.NumberFormat.NumberDecimalSeparator
           нам нужен первый символ */
         if ((e.KeyChar == '.') || (e.KeyChar == ','))
            e.KeyChar = aDecimalSeparator;
         if (e.KeyChar == aDecimalSeparator)
         { // Нажата запятая. Проверим,есть уже запятая в поле редактирования
            if ((textBox3.Text.IndexOf(aDecimalSeparator) != -1) ||
                                       (textBox3.Text.Length == 0))
            { // запятая уже есть. Запретить ввод еще одной
               e.Handled = true;
            }
            return;
```

```
    }
    if (Char.IsControl(e.KeyChar))
    {
      if (e.KeyChar == (char)Keys.Enter)
      { // нажата клавиша <Enter> переместить курсор в поле textBox1
        textBox1.Focus();
      }
      return;
    }
    // остальные символы запрещены
    e.Handled = true;
  }
```

1.4. Компонент Panel

Компонент **Panel** представляет собой поверхность, на которую можно поместить другие компоненты. Панели обеспечивают общее (родовое) поведение для компонентов, помещенных в них: панельные компоненты могут содержать в себе другие компоненты, объединяя их в единое целое. При перемещении панели такие компоненты перемещаются вместе с ней. Обычно он используется для привязки компонентов к одной из границ формы. Например, если панель привязана к нижней границе формы, то независимо от размера окна (при изменении раз мера окна во время работы программы) кнопки, находящиеся на панели, все гда будут находиться в нижней его части Панель позволяет легко управлять компонентами, которые на ней находятся. Например, чтобы сделать недос тупными компоненты панели, достаточно присвоить значение **False** свойству **Enabled** панели.

Свойства компонента **Panel** приведены в табл. 4.

Таблица 4. Свойства компонента Panel

Свойства	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
BorderStyle	Вид границы панели: **FixedSingle** — рамка; **Fixe3D** — объемная граница; **None** — граница не отображается
Enabled	Свойство позволяет сделать недоступными (False) все компоненты, которые находятся в панели
Visible	Позволяет скрыть (**False**) панель
AutoScroll	Признак необходимости отображать полосы прокрутки, если компонен ты, которые находятся в панели, не могут быть выведены полностью

Свойство **Dook** позволяет "привязать" панель и, следовательно, находящиеся в ней компоненты к границе формы. В результате привязки панели к границе формы размер панели автоматически изменяется— ширина панели стано вится равной ширине формы (при привязке к верхней или нижней границе) или высота панели становится равной высоте формы (при привязке к левой или правой границе).

2. Рабочее задание

Задание 1. Самостоятельно разработать приложение, которое позволяет продемонстрировать отображение или скрытие компонентов панели. Используя компоненты **Label, Button** и **Panel,** разработайте интерфейс приложения, внешний вид которого представлен на рис. 4. Компонент **Panel** представлен в виде зеленого прямоугольника, на котором размещены компоненты **Label** и **Button**.

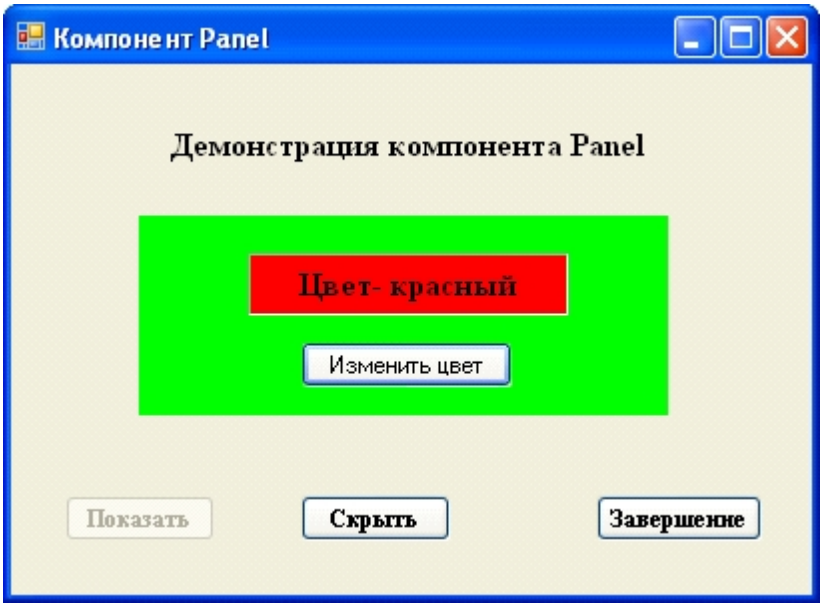

Рис. 4. Вид приложения «Компонент Panel» при первоначальном запуске

При запуске приложения кнопка **«Показать»** (компонент **Button2**) должна быть недоступна. Кнопка **«Изменить цвет»** (компонент **Button1**) должна изменять цвет компонента **Label** с красного на зеленый и наоборот. Кнопка **«Скрыть»** (компонент **Button3**) должна скрывать компонент **Panel1**, а кнопка **«Показать»** (компонент **Button2**) - отображать компонент **Panel1**. Кнопка **«Закрыть»** закрывает приложение.

Задание 2. Разработать программу расчета диапазона значений начальной скорости, с которой необходимо бросить тело под заданным углом бросания с целью попадания в круг радиусом **R**, центр которого расположен на дальности **D**. Дальность **D** полета тела, брошенного под углом **α** к горизонту с начальной скоростью **V**, определяется выражением:

$$D = V^2 \sin(2\alpha) / g,$$

где **g = 9,81 м/с^2** – ускорение свободного падения.

При разработке интерфейса приложения использовать компоненты **Label, TextBox, Button** и **Panel** (один из вариантов интерфейса представлен на рис. 25). Математические функции необходимо брать из класса **Math**, например, **Math.Cos(x).**

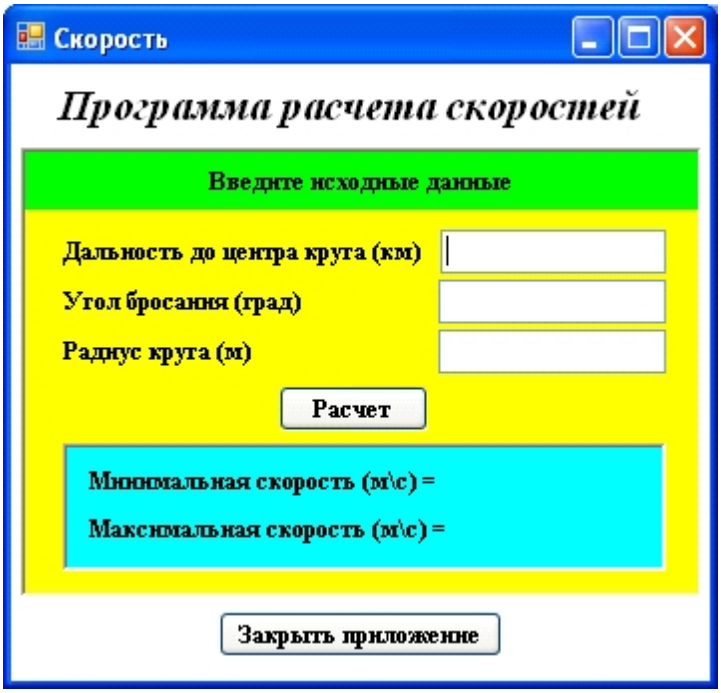

Рис. 5. Внешний вид приложения «Скорость»

Задание 3. Разработать программу «Окружность» для поиска центра и радиуса окружности. Уравнение $Ax^2 + Bx + Ay^2 + Cy + D = 0$ представляет окружность при условии, что коэффициенты **A, B, C, D** удовлетворяют неравенству $B^2+C^2-4AD > 0$. Тогда центр (**a, b**) и радиус **R** окружности можно найти по формулам:

$$a = -\frac{B}{2A}, \quad b = -\frac{C}{2A}, \quad R = \frac{B^2 + C^2 - 4AD}{4A^2}$$

В случае отсутствия решения уравнения предусмотреть вывод сообщения о невозможности нахождения центра и радиуса окружности.

При разработке интерфейса приложения использовать компоненты **Label, TextBox, Panel** и **Button** (один из вариантов интерфейса представлен на рис. 6).

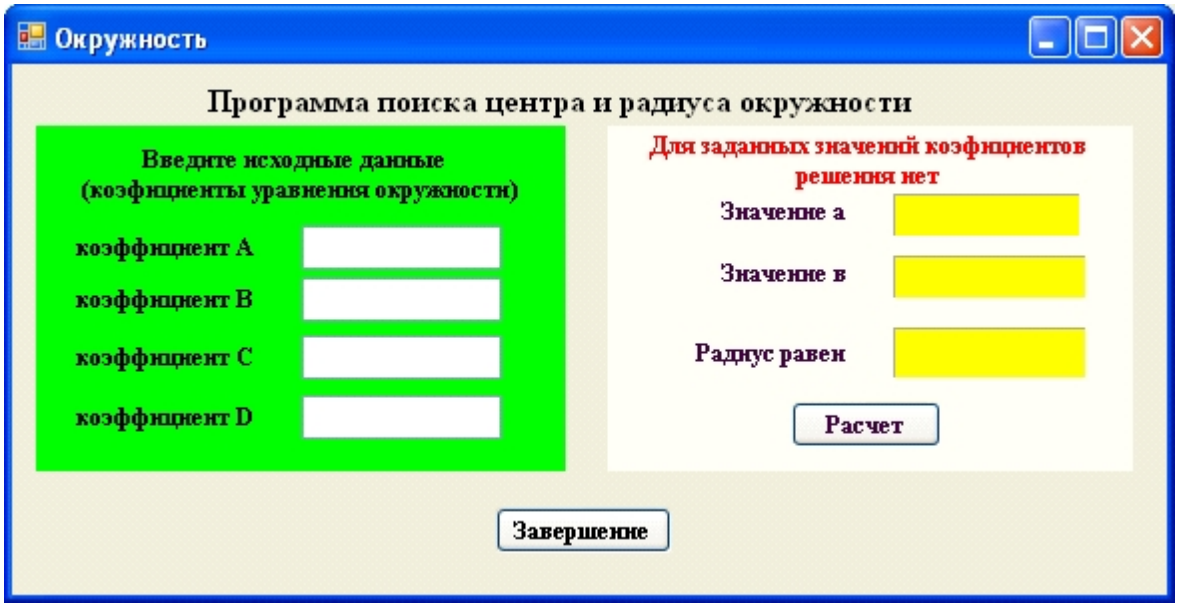

Рис. 6. Внешний вид приложения «Окружность»

Задание 4. Разработать программу «Калькулятор», с помощью которой можно проводить простейшие арифметические операции. В окне «Результат» должны отображаться исходные числа (вводятся с помощью соответствующих кнопок), знак операции и результат.

При разработке интерфейса приложения использовать компоненты **TextBox, Panel** и **Button** (один из вариантов интерфейса представлен на рис. 7).

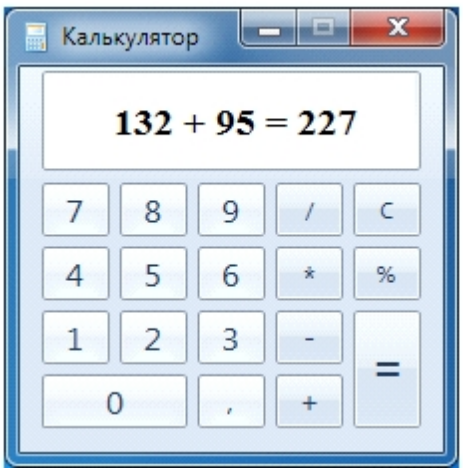

Рис. 7. Внешний вид приложения «Калькулятор»

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.