ЛАБОРАТОРНАЯ РАБОТА №8

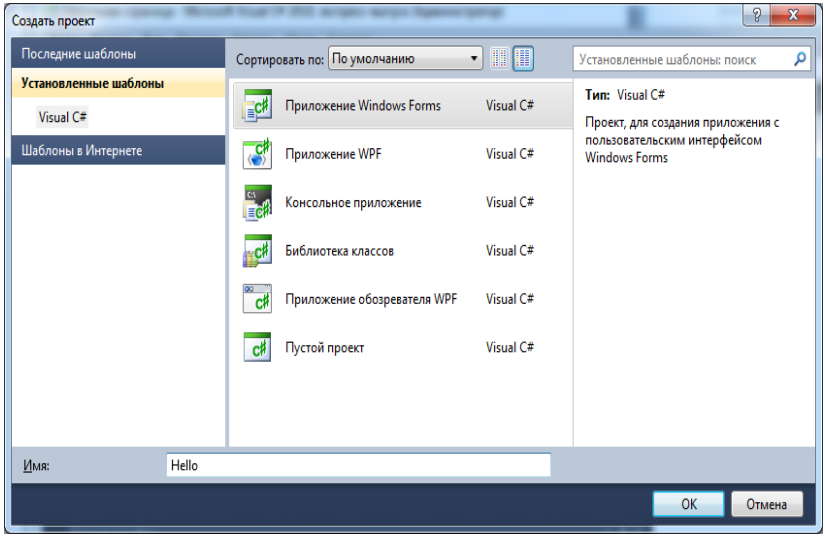
Windows Forms Hello

Создать проект Windows Forms приложения Hello. Оно должно при нажатии в форме кнопки «Нажми» выводить на экран фразу «Hello, World and Россия <Фамилия студента>».

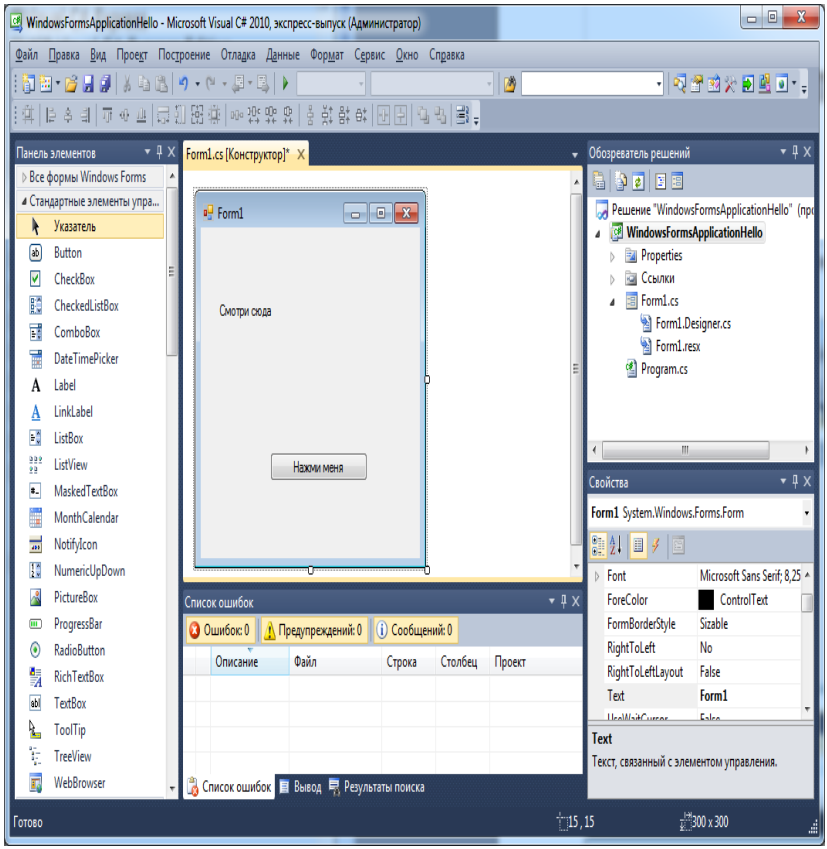
Варианты к заданию. Индивидуальные данные студента.

Пример. Создать проект Windows Forms приложения Hello. Оно должно при нажатии в форме кнопки «Нажми меня» выводить на экран фразу «Hello, World and Россия от Акчурина!».

Активизировать ИСР. В главном меню выбирается команда File=>New Project. Вызывается окно выбора типа проекта с набором шаблонов. В нем выбираем Приложение WindowsForms. Задаем имя проекта Hello.



Вид ИСР меняется.



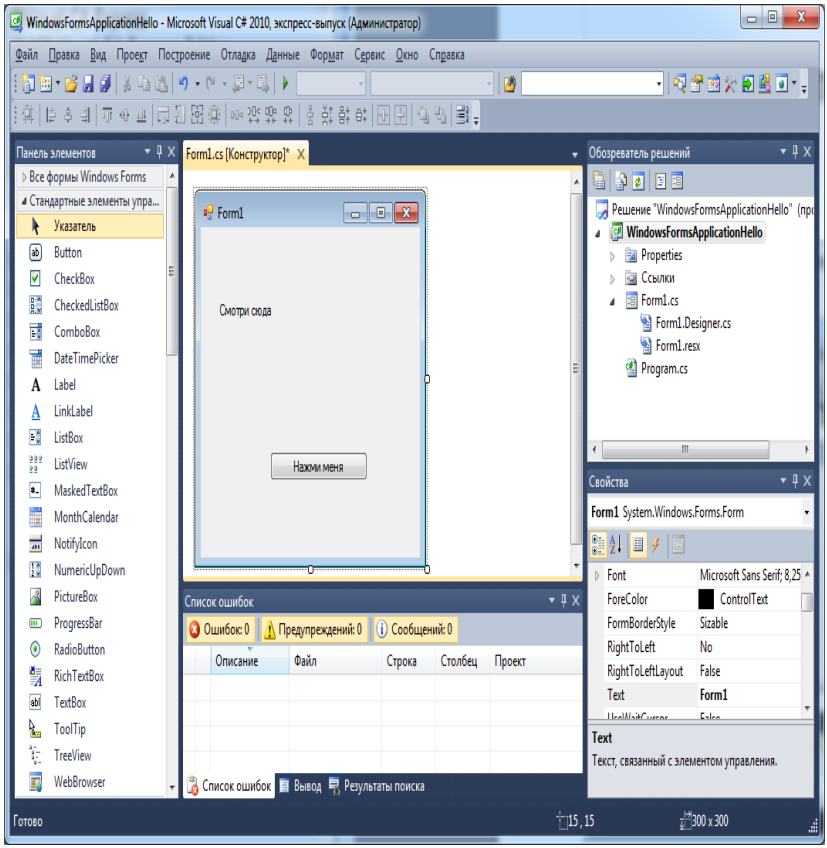
В центре в закладке **Form1.cs[Конструктор]** отображается окно **Конструктора** формы. Окно **Панель элементов** заполняется элементами для выбора. Окна **Обозреватель решений** содержит описание решения. **Конструктор** формы отображается по автоматически создаваемому коду (при желании его можно посмотреть двойным щелчком по **Form1.cs => Program.cs** в **Обозревателе решений**). **Редактор** кода модуля формы отображается командой **Перейти к коду**, которая находится в меню, выпадающем при щелчке по форме в кон-структоре правой кнопкой мыши. Редактор отображается в закладке с именем **Form1.cs**.

Большая часть кода в **Редакторе** ИСР сделала автоматически. Нужно доба-вить функциональность.

Окна **Конструктора** и **Редактора** можно переключать кнопками в заголовках их закладок.

Теперь приступаем к проектированию в **Конструкторе**. Из окна Панели эле-ментов перетаскиваем в форму объекты

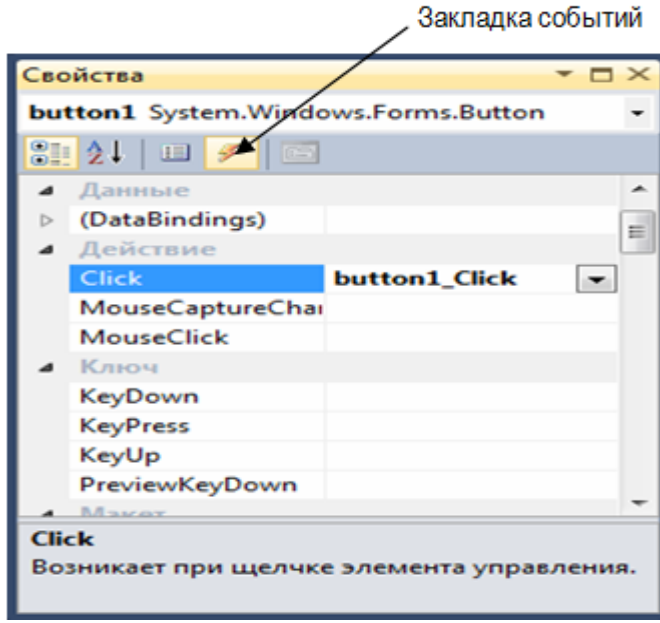
- button1 – кнопка для запуска обработчика события. Выделяем объект, в окне свойств отображаются свойства кнопки. Свойству Text присваиваем значение - Нажми меня.
- label1 – метка, поле для отображения сообщения. Свойству Text присваи-ваем значение – Смотри сюда.



Для создания обработчика события щелчка по кнопке дважды щелкаем по кнопке в форме. Автоматически отображается окно **Редактора**, в котором в

код добавлен шаблон обработчика события button1_Click, но без функцио-нальности. Курсор устанавливается в место ввода кода, который будет зада-вать функциональность проекта.

Чтобы обработчик события срабатывал, нужно в окне свойств кнопки button1 в закладке событий выбрать реакцию на щелчок по кнопке из списка:



В шаблон кода, начиная с позиции курсора, нужно ввести инструкции. В приме-ре свойству Text объекта label1 нужно присвоить строку “Hello, world and Рос-сия от меня!!”. Чтобы исключить повторный доступ к кнопке, сделаем ее после вывода текста невидимой. Для этого вводим код

```
label1.Text= "Hello, world!";
button1.Visible = false;
```

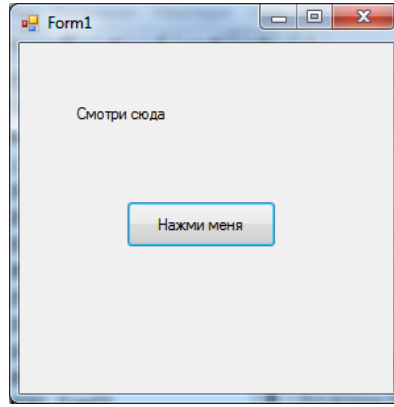
Листинг программы

```
using System;
using System.Windows.Forms;
namespace Hello
{
```

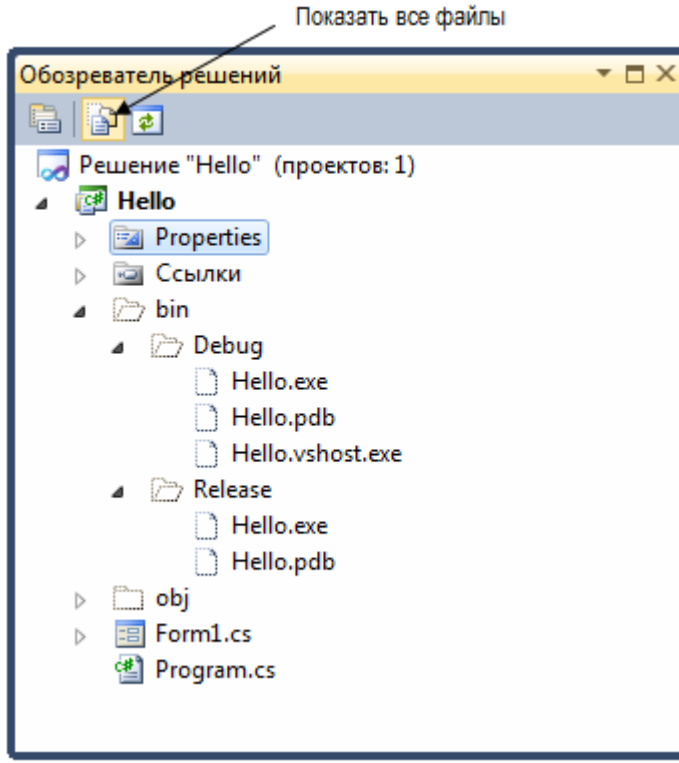
```
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void button1_Click(object sender, EventArgs e)
        {
            label1.Text = "Hello, world!!"; button1.Visible =
            false;
        }

        private void Form1_Load(object sender, EventArgs e)
        {
        }
    }
}
```

Проект готов, проверим его командой **Отладка=>Запуск без отладки**. Полу-чим окно приложения.



Проект готов, сохраняем его командой **Файл=>Сохранить все** в папке Hello. В результате проект сохраняется в структуре папок (чтобы увидеть все файлы, нужно активизировать кнопку, показанную на рисунке):



Решение включает:

Решение Hello	Решение
Hello	Проект
Properties	Свойства
Ссылки	Ссылки
bin	Двоичные файлы
Debug	Файлы отладки
Hello.exe	Управляемый исполняемый файл
Hello.pdb	База данных для JIT компилятора
Hello.vshost.e xe Release	Служебный файл Файлы выпуска
obj	Объектные файлы

Исполняемые (bin) и объектные (obj) файлы образуются при компиляции (по-строении). Возможны два режима:

- Команда **Построение=>Построить решение**. Построение в режиме от-ладки, в компонуемые файлы включаются символы отладки и режим оп-тимизации исключается. Это может увеличить размеры файлов. Файлы размещаются в папках Debug.
- Команда **Построение=>Перестроить решение**. Построение отлаженного проекта, когда в компонуемые файлы символы отладки не включаются и компилятор использует режим оптимизации кода (например, исключает не использованные переменные). Это может уменьшить размеры файлов. Файлы размещаются в папках Release.

Отладка и тестирование в C#

Предмет исследований

- Отладка программ.
- Способы отладки.
- Тестирование созданных программ.

Контрольные вопросы

1. Возможности по отладке в Visual C# 2010.
2. Использование точек останова при отладке программ.
3. Проверка значения отдельных переменных в процесс отладки.
4. Настройка параметров отладки в Visual C# 2010.
5. Одновременный просмотр значения нескольких переменных в процесс отладки.

Задание. Напишите и протестируйте программу.

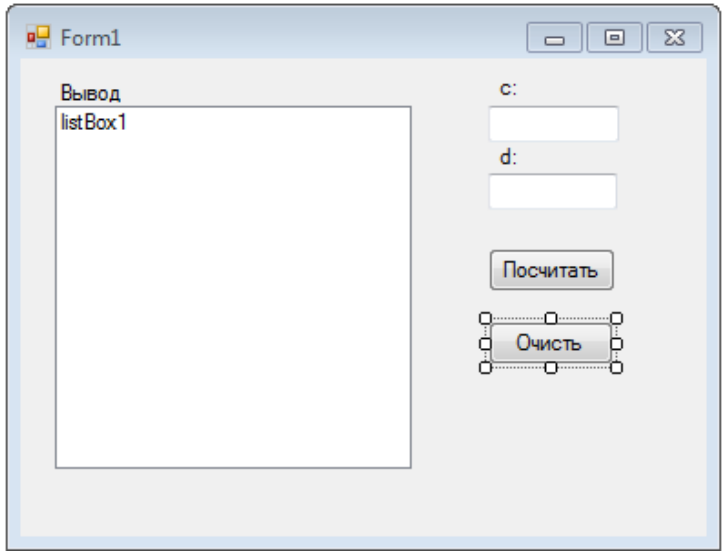
Пример. В примере представлена программа, в которой по паре вводимых чисел c, d в цикле для i=1...10 вычисляятся значения пары других параметров a=(c+d)*i, b=(c-d)*i. Функция Ample вычисляет квадратный корень из суммы квадратов целых частей от a, b, и эти значения суммируются в переменной s. По завершении цикла вычисляется среднее арифметическое sm от выходных параметров функции Ample. Функция Ample оформлена в виде подпрограммы. В программе определены метки, задающие точки останова.

Протестировать учебную программу с условиями:

- С помощью отладочной печати контролировать массив s[i].
- Переменные для окна «Список наблюдения» - a,b,s,i.
- Точки останова - m1,m2,m4.

Варианты		
№	Переменные для окна «Список наблюдения»	Строки останова
1.	a, b, f	m1,m2,m3
2.	c, d, s	m4,m5,m6
3.	a, c, s	m2,m3,m4
4.	b, d, f	m1,m5,m6
5.	a, d, c	m3,m4,m5
6.	b, a, s	m1,m2,m6
7.	c, d, f	m4,m5,m6
8.	d, c, a	m1,m2,m3
9.	f, d, b	m5,m6,m1
10.	s, f, c	m4,m2,m3
11.	a, s, f	m6,m1,m2
12.	b, a, s	m3,m4,m5
13.	c, d, f	m1,m3,m5
14.	s, d, c	m4,m5,m3
15.	a, c, f	m2,m1,m6

В программе использован графичесчкий интерфейс. Форма окна



Листинг программы

```
using System;
using System.Windows.Forms;

namespace LabDeb
{
   public partial class Form1 : Form
   {
      double a, b, c, d, s, f, sm;
      public Form1()
      {
         InitializeComponent();
      }
      public static double Aml(double x, double y)
      {
         double x1, y1;
         x1 = Math.Truncate(x);
         y1 = Math.Truncate(y);
         double result = Math.Sqrt(x1 * x1 + y1 * y1);
         return result;
      }
      private void button1_Click(object sender, EventArgs e)
      {
         string z;
         c = Convert.ToDouble(textBox1.Text);
         d = Convert.ToDouble(textBox2.Text);
         s = 0;
         for (double i = 0; i < 10; i++)
         {
            a = (c + d) * i;
            b = (c - d) * i;
            f = Aml(a, b);
            s = s + f;
            z = "s[" + Convert.ToString(i) + "] =" + Convert.ToString(s);
            listBox1.Items.Add(z);
         }
         sm = s / 10;
         listBox1.Items.Add(" ");
         z = "Результат = " + sm;
         listBox1.Items.Add(z);
      }
```

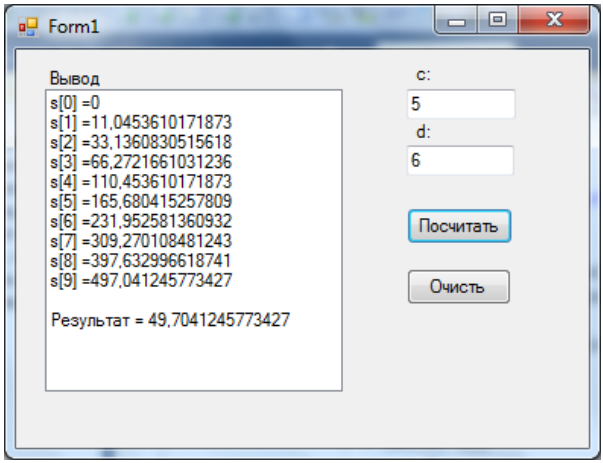
```
    private void button2_Click(object sender, EventArgs e)
    {
        listBox1.Items.Clear();
    }
}
```

Выполнение

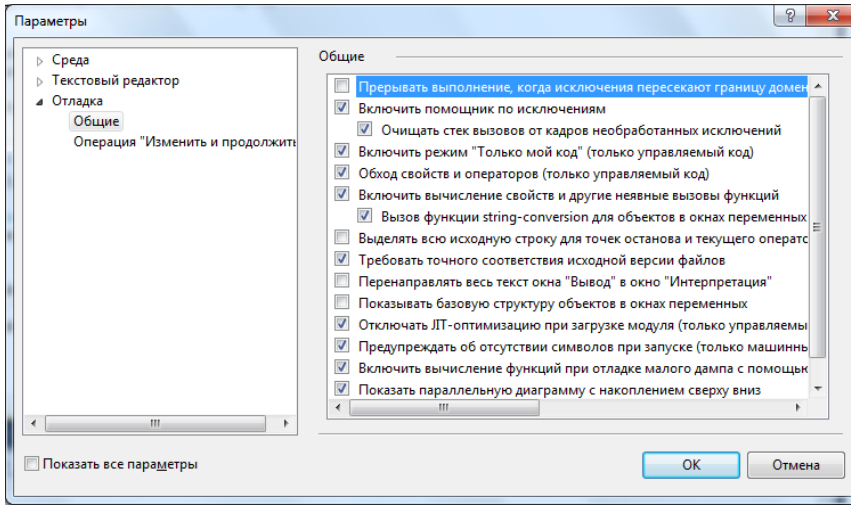
1. Создаем приложение Windows Form.

2. Размещаем в нем код учебной программы.

3. Проверка значений s[i] с помощью отладочной печати. Отладочная печать - вывод значений переменных в окно приложения осуществляется при прогоне программы. Ниже показано окно приложения при прогоне

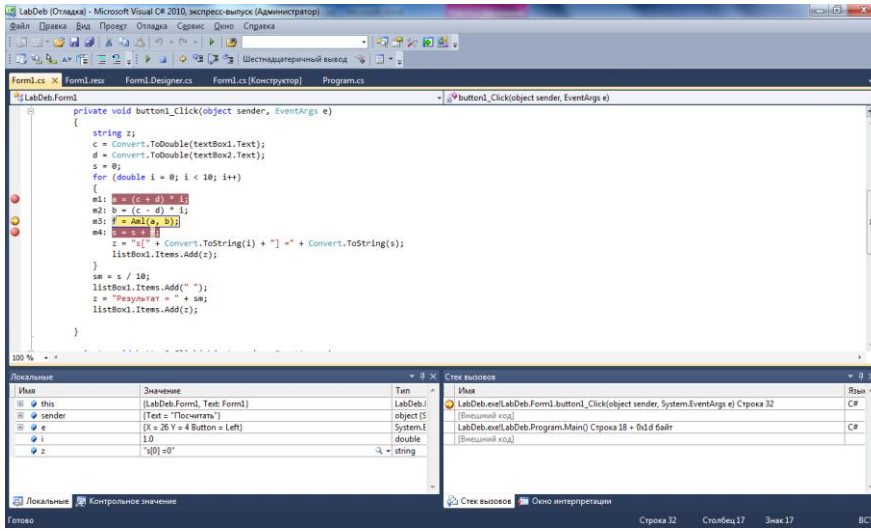


Настройка параметров отладки производится в меню "Отладка => Параметры и настройки.



4. Создание точек останова в Visual C# производится следующим образом: Выбираем нужную строку за тем "Отладка => Точка останова" или нажать F9

Ниже показан просмотр результата в точке останова при запущенной программе.



5. Проверяем значения переменных при остановке программы в точках останова наведя на переменны курсор мыши .

6. Для проверки значения нескольких переменных в процессе выполнения программы выделяем нужную переменную щелкаем правой кнопкой мыши и "Добавить контрольное значение" после этого возможен просмотр значения переменных в момент остановки программ в точках останова.