Лабораторная работа №1

Тема: Создание, компилирование, отладка и выполнение проектов в интегрированной среде разработки.

Цель: Научится созданию, компиляции, отладке и выполнению проектов в интегрированной среде разработки.

Теоретические сведения:

Все программы условно можно подразделить на две категории: консольные и оконные.

Консольные программы — это наследие старых операционных систем типа MS DOS, работавших в текстовом режиме. Полной аналогии здесь нет, но создатели систем разработки пытаются сделать всё, чтобы было максимальное сходство. Для консольной программы операционная система или система разработки предоставляет специальное окно (обычно — черного цвета), в котором виден ход выполнения программы.

Оконное приложение всегда имеет своё собственное окно (обычно его называют главным окном приложения). Кроме главного окна в приложении могут использоваться множество других окон.

При разработке оконного приложения используются различные элементы управления: командные кнопки, окна ввода, надписи, меню и т.д.

Выполнение работы:

Создание первой программы

Рассмотрим вопрос о том, как с помощью среды разработки **Microsoft Visual Studio Express** создать программу. Итак, первое, что нам необходимо сделать — запустить среду разработки. В окне, которое откроется, необходимо создать новый проект. Для этого в меню **File выбираем** команду **New Project**, как это показано на рис. 1.

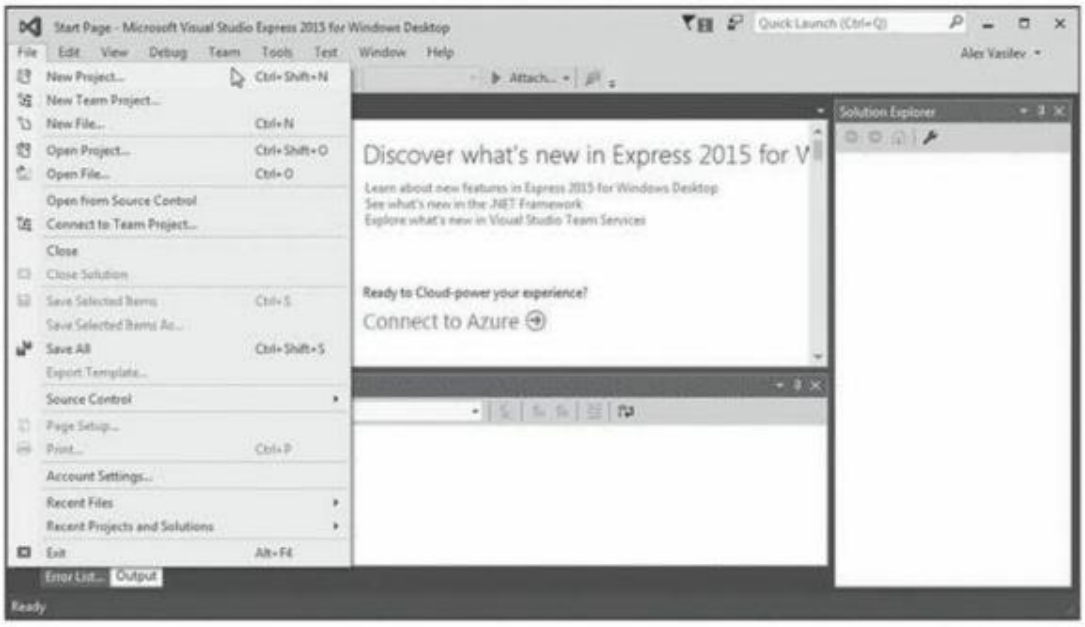


Рисунок 1 – Создание проекта в окне разработки Microsoft Visual Studio Express

В результате появляется окно создания нового проекта, показанное на рис. 2.

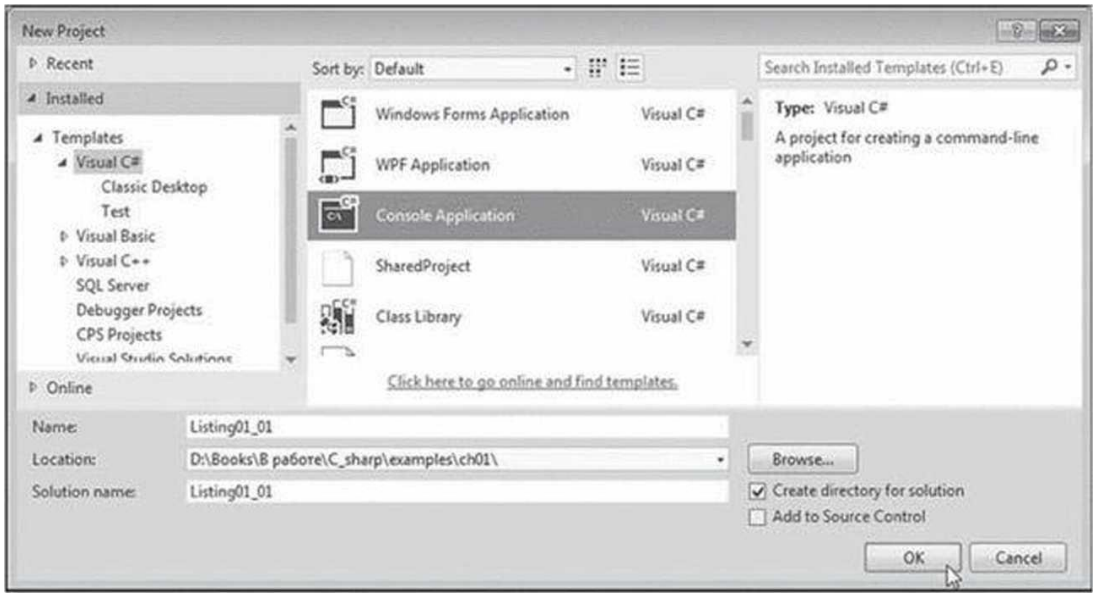


Рисунок 2 - Окно создания нового проекта New Project

В разделе **Templates** выбираем **Visual C#**, а в центральной части окна следует выбрать пункт **Console Application**. В поле **Name** указываем название проекта (мы используем название **Listing01_01**), а в поле **Location** выбираем место для сохранения файлов проектов.

После щелчка на кнопке **ОК** открывается окно проекта с шаблонным кодом, представленное на рис. 3.

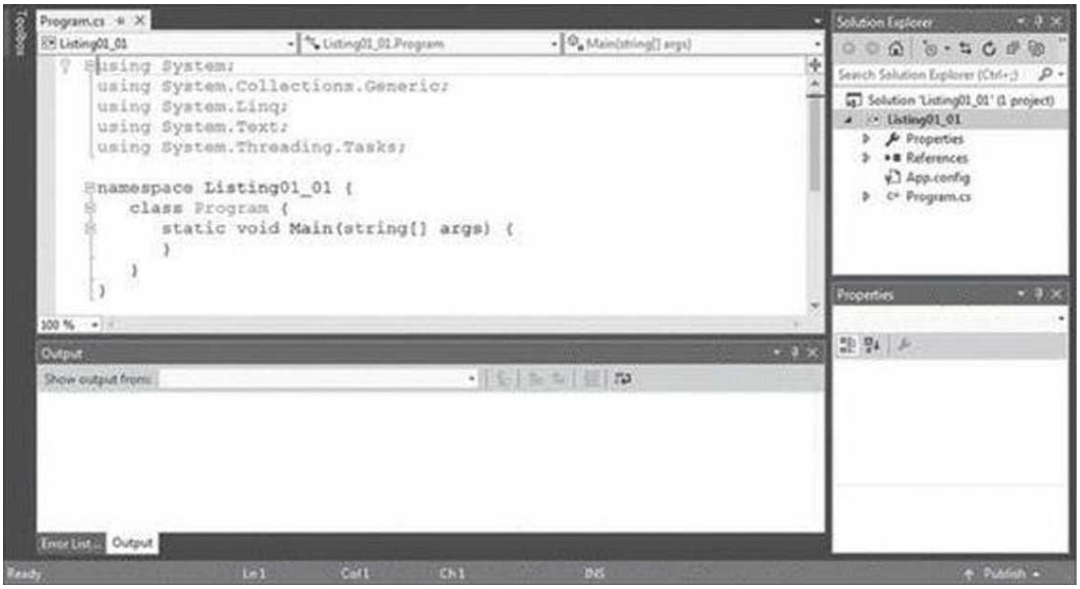


Рисунок 3 – Окно проекта с шаблонным кодом

В окне редактора (окно с программным кодом) заменяем шаблонный код на код нашей программы (см. листинг 1). Результат показан на рис. 4.

Листинг 1:

```
// Главный класс программы:
class HelloWorld{
   // Главный метод программы:
   static void Main(){
      // Отображение сообщения в консольном окне:
      System.Console.WriteLine("Изучаем язык C#");
   }
}
```

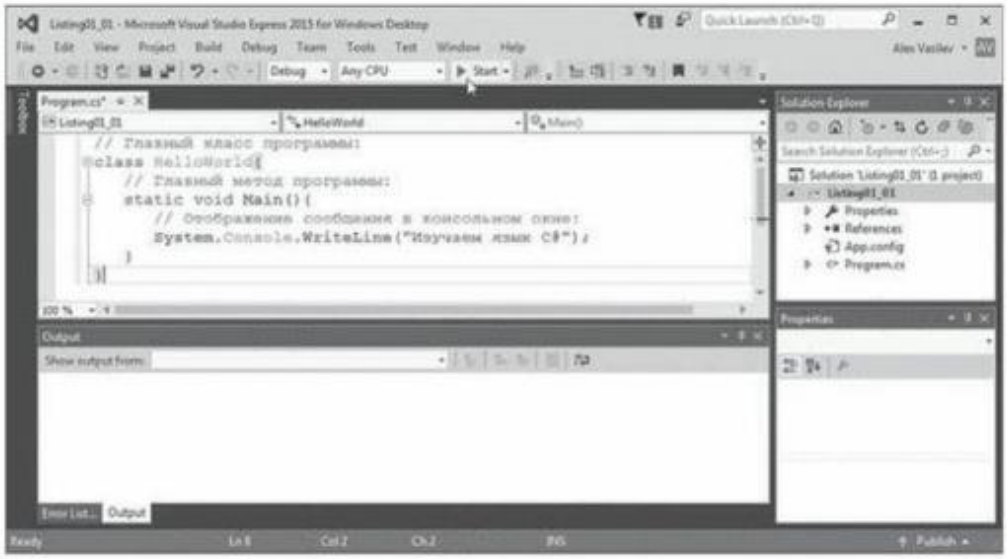


Рисунок 4 - Окно проекта с кодом программы

После того как код введен, программу можно компилировать и запускать на выполнение. Для этого можем воспользоваться специальной пиктограммой с зеленой стрелкой на панели инструментов (см. рис. 4). Получше воспользоваться командой **Start Without Debugging** из меню **Debug** (комбинация клавши **<Ctrl>+<F5>**), как показано па рис. 5.

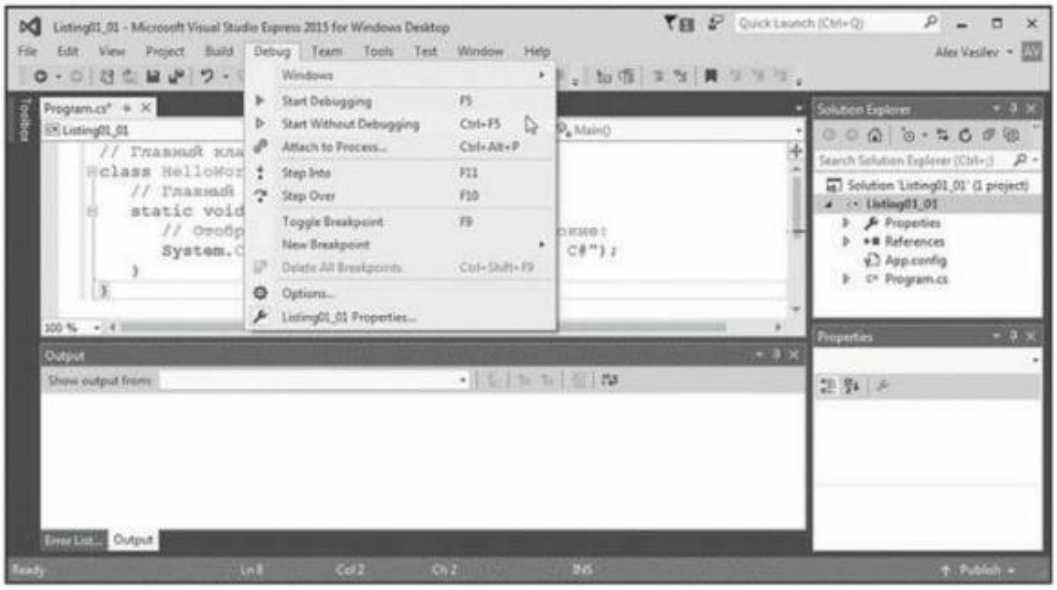


Рисунок 5 - Запуск программы на выполнение

Если компиляция прошла успешно, то в консольном окне появляется результат выполнения программы (в данном случае — сообщение), как показано на рис. 6.

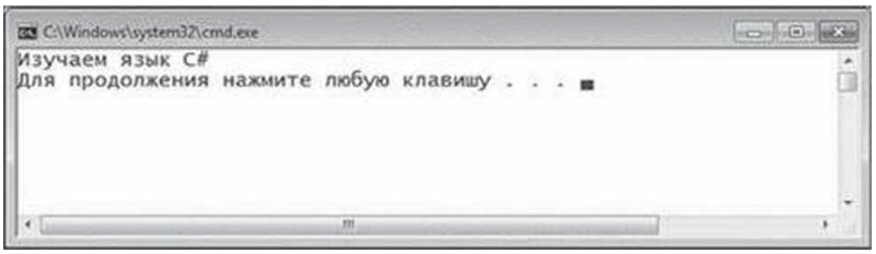


Рисунок 6 - Результат выполнения программы отображается в консольном окне

Порядок создания консольного приложения

Для создания консольного приложения в Microsoft Visual Studio необходимо сделать следующее. В меню **«File»(«Файл»)** выбрать пункт **«New»(«Новый»)** и в нем выбрать подпункт **«Project»(«Проект»).**

Далее в появившемся диалоговом окне, в разделе **Project types** необходимо выбрать пункт **Visual C#** и подпункт **Windows**. В разделе **Templates** выбрать **Console Application**. В нижней части окна необходимо задать имя проекта и папку, где он будет сохраняться. При этом необходимо

следить, чтобы папка, в которую сохраняется проект, была доступна для записи.

Ввод данных с клавиатуры осуществляется посредством метода **ReadLine()** класса **Console**. Данный метод возвращает строку, которую ввел с клавиатуры пользователь. Вывод данных на экран осуществляется посредством метода **WriteLine()** класса **Console**.

Код программы пишем между фигурными скобками (внутри тела) этого метода:

```
public static void Main(string[] args)
{
}
```

Листинг 2:

```
using System;
public class Program
{
   public static void Main(string[] args)
   {
     Console.WriteLine("Введите свое имя");
     string st=Console.ReadLine();
     Console.WriteLine("Привет {0}",st);
     Console.ReadLine();
   }
}
```

Порядок выполнения программы с диалоговым окном

Следующая программа, которую мы рассмотрим, выводит сообщение диалоговом окне: при запуске программы на выполнение появляется диалоговое окно, содержащее текст (сообщение). В окне будет кнопка ОК. и щелчок на ней приводит к закрытию окна и завершению выполнения программы. Для отображения окна мы воспользуемся статическим методом Show () из класса MessageBox. Класс описан в пространстве имен Forms, которое входит в пространство имен Windows, которое, в свою очередь, входит в пространство имен System. Эта иерархия отображается инструкцией System. Windows . Forms. Соответственно, программный код начинается инструкцией using System. Windows . Forms, подключающей пространство имен Forms. Это дает возможность использовать класс MessageBox и его метод Show ().

Если при вызове метода Show () ему передать аргументом текстовое значение, то появится диалоговое окно, содержащее данный текст. Как выглядит код программы, в которой отображается диалоговое окно, показано в листинге 3.

Листинг 3. Отображение диалогового окна

```
// Использование пространства имен:
using System.Windows.Forms;
// Класс с главным методом программы:
class DialogDemo{
// Главный метод программы:
static void Main(){
// Отображение диалогового окна:
MessageBox.Show ("Продолжаем изучать C#);
}
}
```

Кроме инструкции подключения пространства имен в начале программы, в главном методе программы имеется команда **MessageBox. Show** ("Продолжаем изучать C#"). Как отмечалось ранее, в результате выполнения этой команды появится диалоговое окно. Но способ создания проекта в данном случае немного отличается от того, что рассматривался выше. Далее описываются основные этапы реализации программы.

Итак, на этот раз будем создавать пустой проект — для этого в окне создания проекта New Project в качестве типа проекта выбираем Empty Project, как показано на рис. 7.

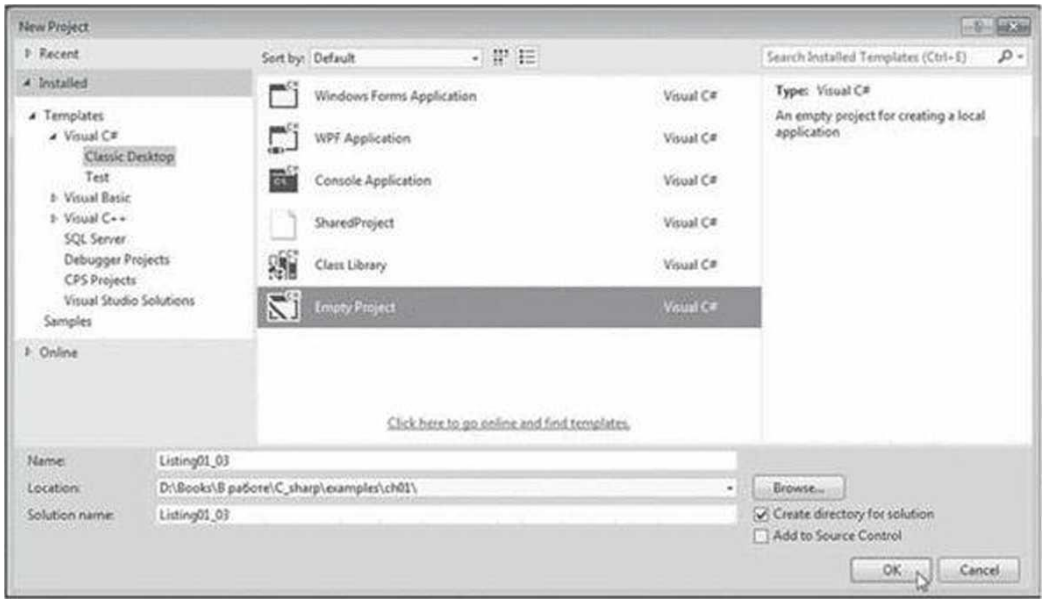


Рисунок 7 - Создание пустого проекта

После того как объект создан, в него нужно добавить файл, в который мы введем программный код. Для этого во внутреннем окне **Solution Explorer** в правом верхнем углу окна среды разработки выбираем узел проекта и щелкаем правой кнопкой мыши. Откроется контекстное меню для проекта, в котором в подменю **Add** выбираем команду **New Item**, как показано на рис. 8.

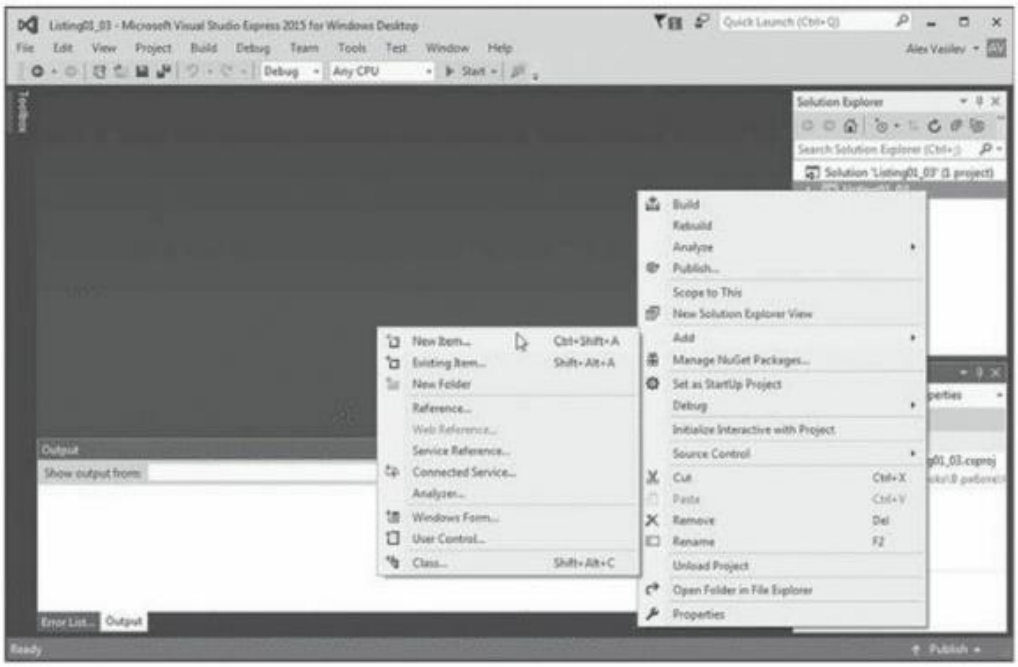


Рисунок 8 - Добавление элемента в созданный пустой проект

В результате откроется окно **Add New Item** для добавления элемента в проект. Окно показано на рис. 9.

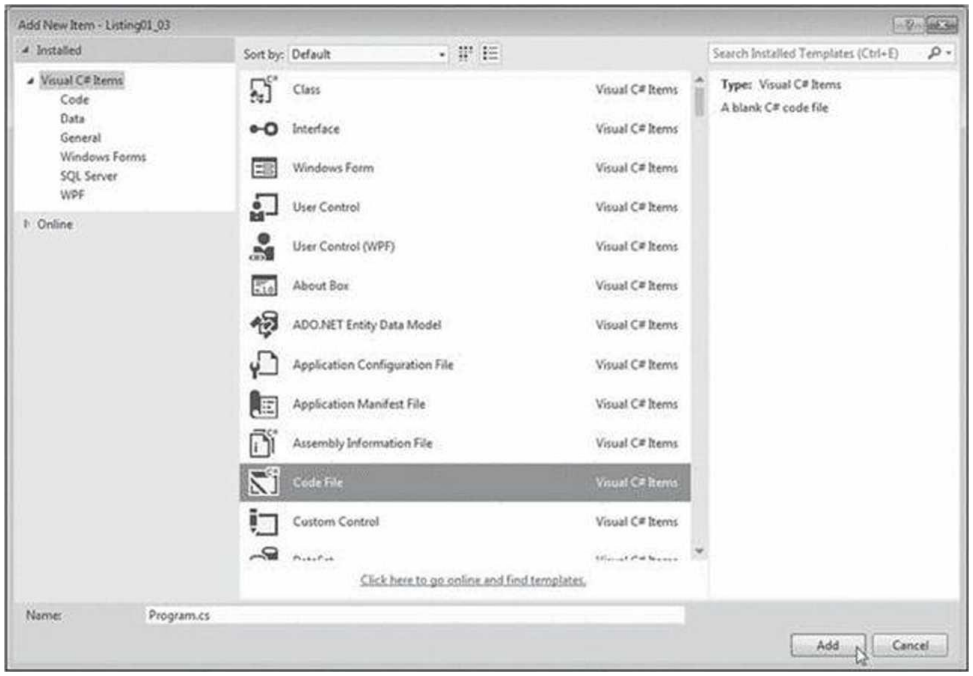


Рисунок 9 – Добавление в проект файла с программным кодом

В этом окне для типа добавляемого элемента выбираем **Code File**, а в ноле **Name** указываем название для файла (в данном примере мы назовем файл Program, cs). После щелчка на кнопке **Add** файл добавляется в проект. После этого в проекте есть файл, в который мы собираемся ввести программный код. Как выглядит проекте добавленным в него файлом (еще пустым), показано на рис. 10.

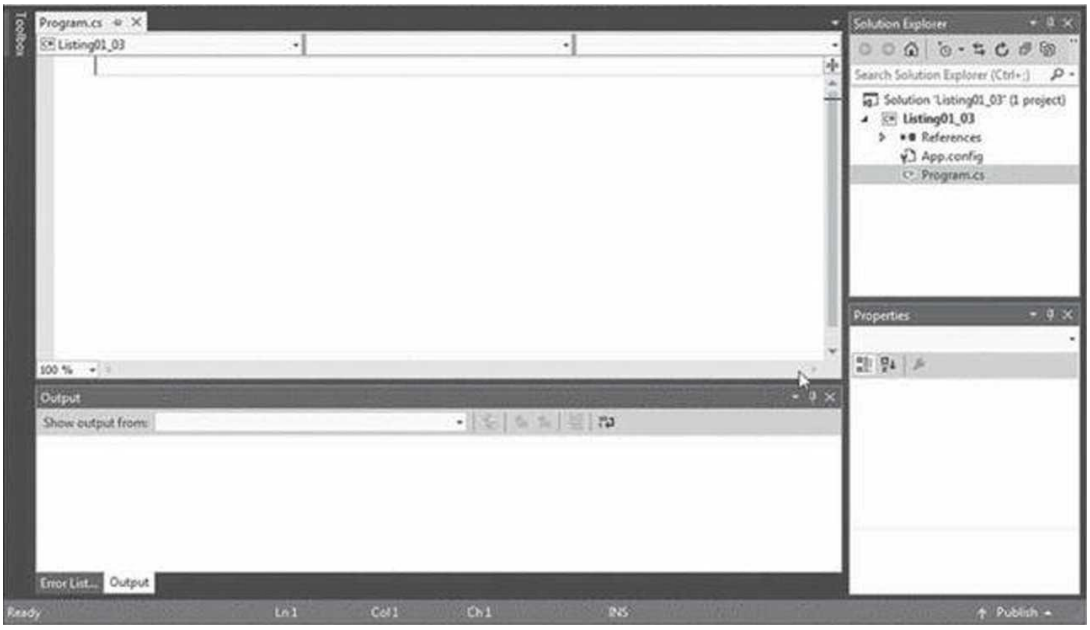


Рисунок 10 – Проект с файлом для ввода кода программы

После ввода программного кода окно среды разработки с открытым в ней проектом будет выглядеть так, как показано на рис. 11.

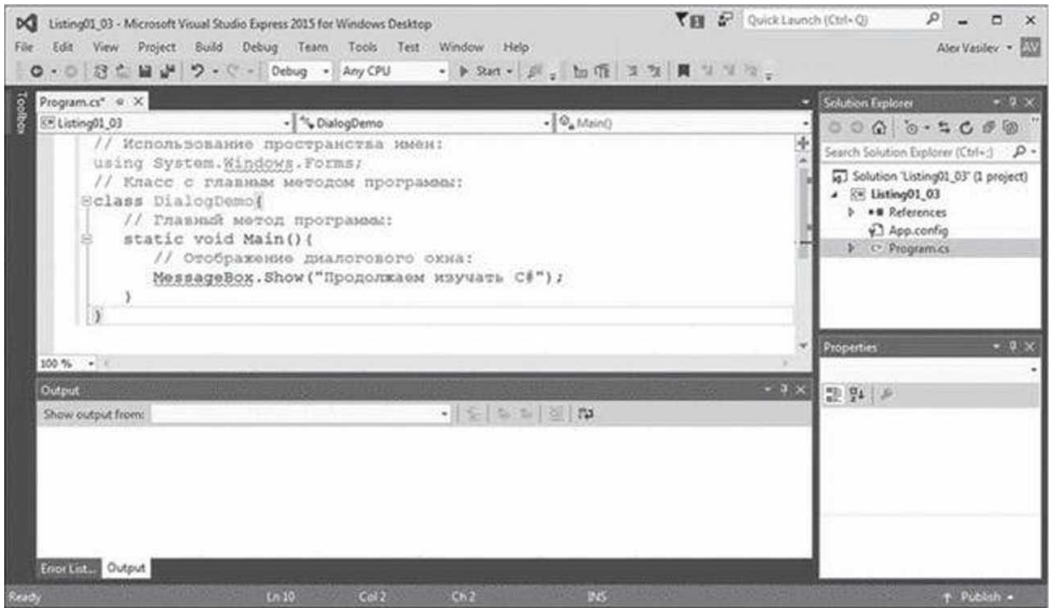


Рисунок 11 - Программный код введен в окно редактора кодов

Хотя мы уже ввели программный код, проект к работе еще не готов. В него необходимо добавить ссылку на пространство имен. Для этого в

контекстном меню проекта в подменю **Add** выбираем команду **Reference**. Ситуация проиллюстрирована на рис. 12.

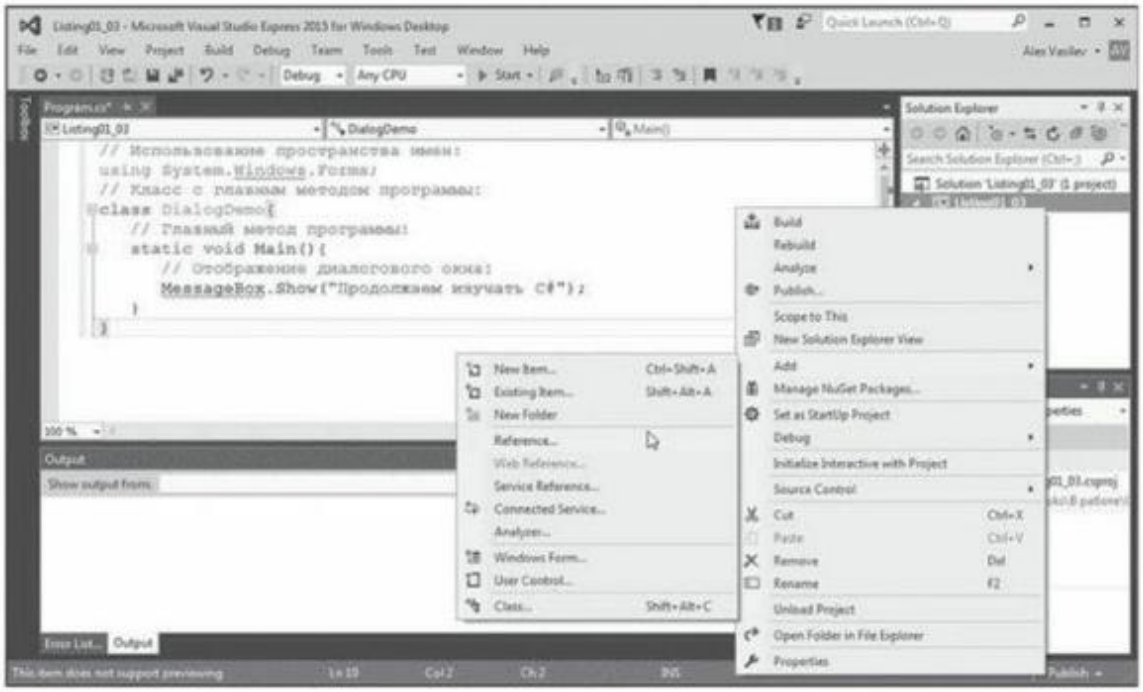


Рисунок12 - Добавление ссылок в проект

Откроется окно **Reference Manager**, в котором мы находим пункт **System. Windows.Forms** и устанавливаем возле него флажок (рис. 13).

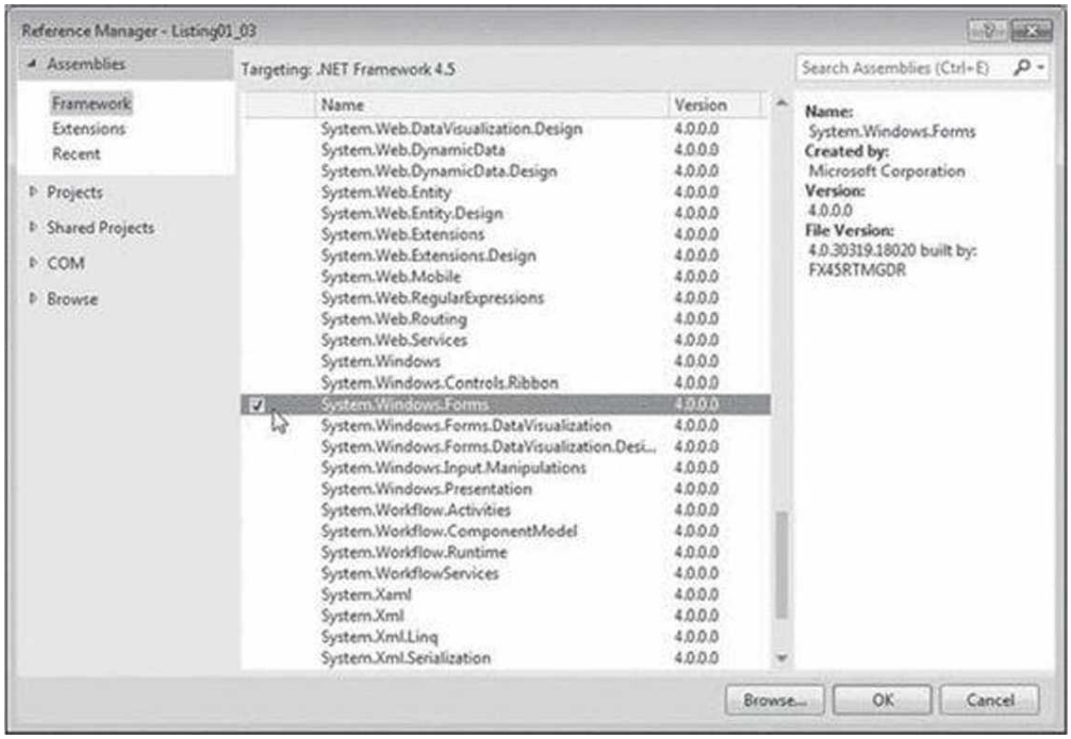


Рисунок 13 - Выбор ссылки для добавления в проект

Для подтверждения щелкаем кнопку ОК.

Технически проект готов к использованию. Осталось один маленький «штрих» добавить. А именно, мы установим тип приложения. Для этого в контекстном меню проекта выбираем команду **Properties** (рис. 14) или выбираем одноименную команду в меню **Project**.

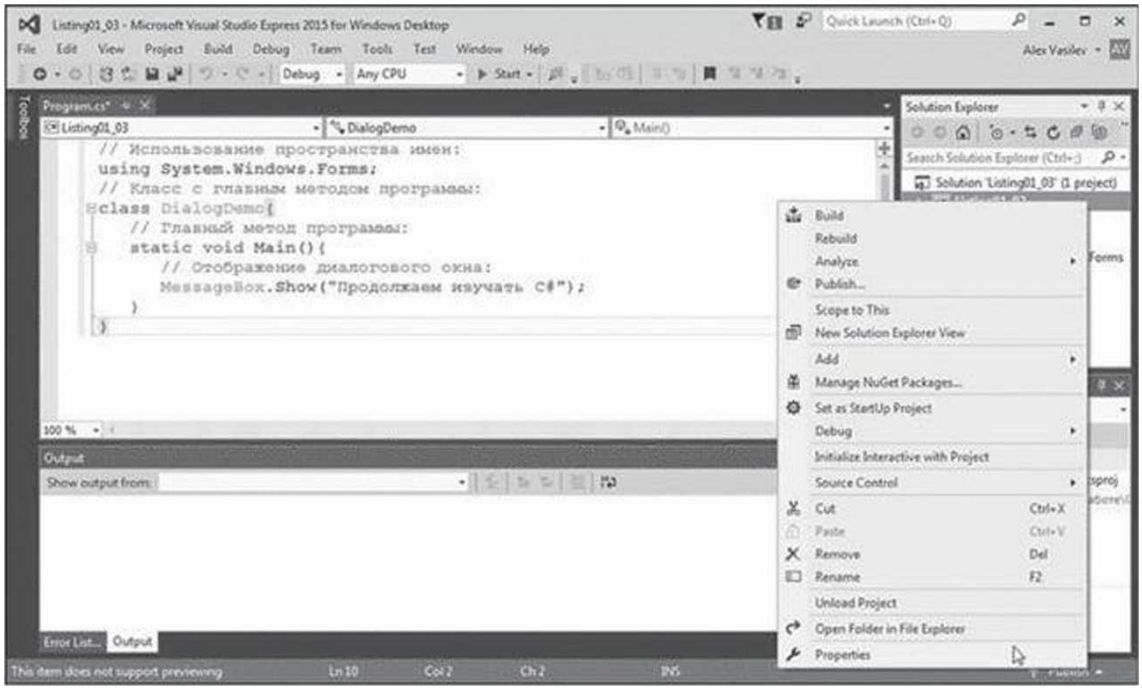


Рисунок 14 - Выбор команды Properties в контекстном меню проекта для перехода к вкладке свойств проекта

Во внутреннем окне редактора кодов откроется вкладка свойств проекта. Она показана на рис. 15.

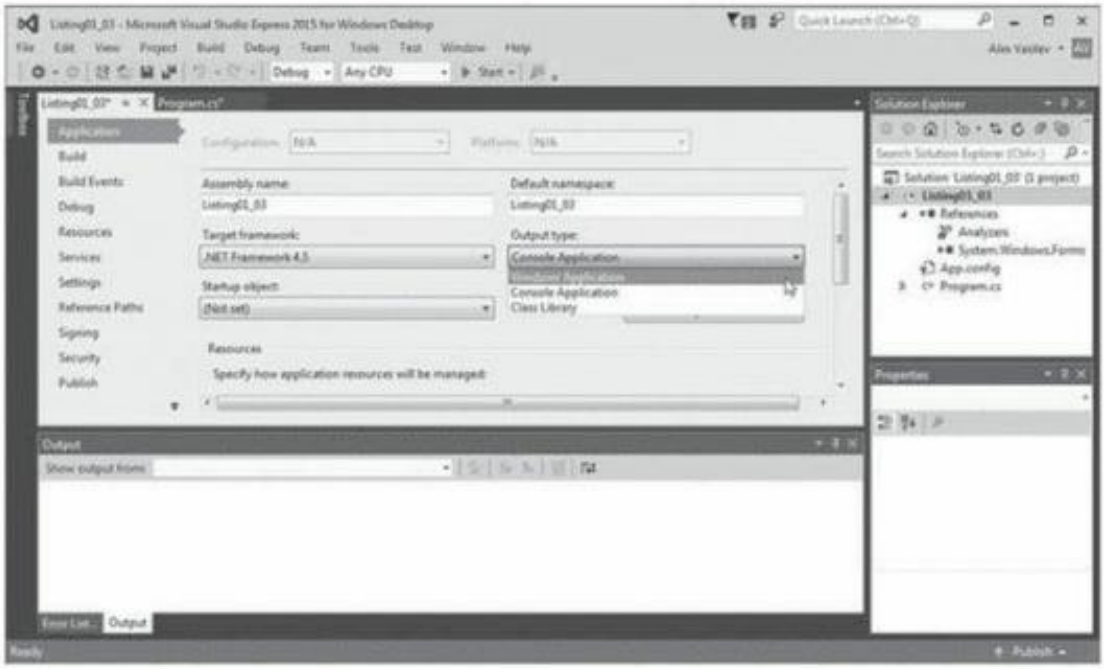


Рисунок 15 – Выбор типа приложения на вкладке свойств проекта

На вкладке свойств проекта следует выбрать раздел **Application**, а в раскрывающемся списке **Output type** устанавливаем значение **Windows Application** (вместо используемого по умолчанию значения **Console Application**). Чтобы закрыть вкладку свойств проекта, щелкаем системную пиктограмму (рис. 16).

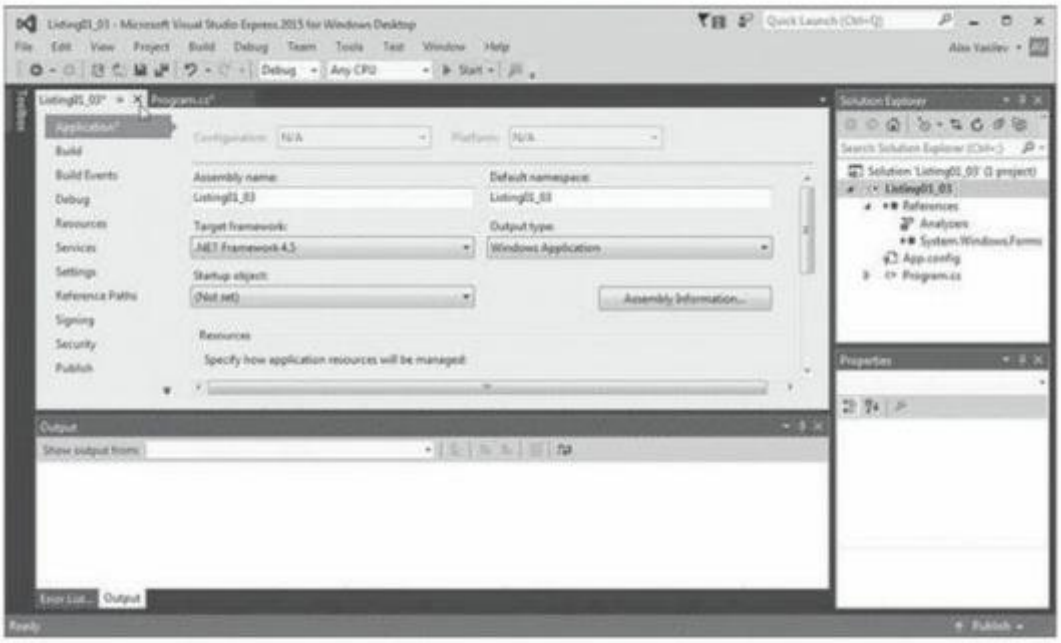


Рисунок16 - Закрытие вкладки свойств проекта

Для компилирования и запуска программы на выполнение выбираем, например, команду **Start Debugging** в меню **Debag** (рис. 17).

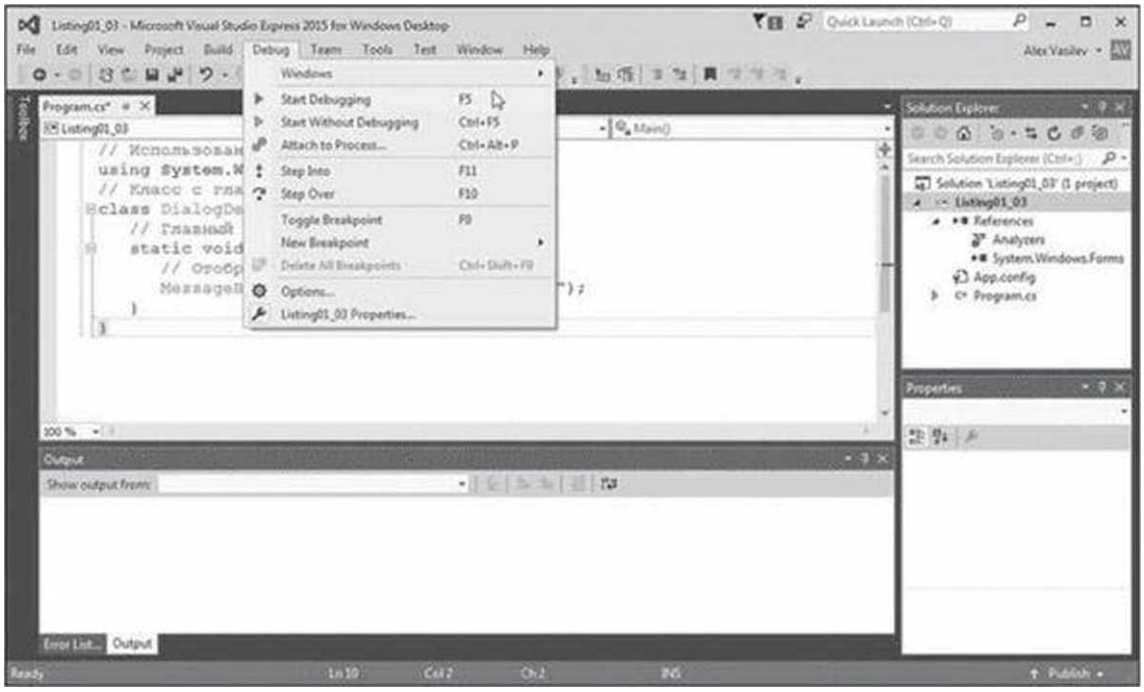


Рисунок 17 - Запуск приложения на выполнение с помощью команды Start Debugging в меню Debug

Если программа компилируется без ошибок, появляется диалоговое окно, представленное на рис. 18.

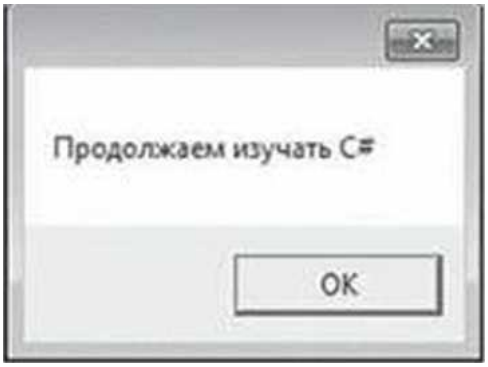


Рисунок 18 - При выполнении программы отображается окно с сообщением

Щелчок на кнопке **ОК** в этом окне (или на системной пиктограмме в строке названия окна) приводит к закрытию окна и завершению выполнения программы.

Порядок создания оконного приложения:

Создание оконного приложения начинается с создания проекта соответствующего типа. Порядок работы следующий:

1. Запускаем **Visual Studio** в **Стартовой странице (Start)** выбираем **New Project…**
2. В открывшемся окне диалога **New Project (Новый проект)** выбираем тип шаблона **Windows Forms Application (Оконное приложение)** (рис. 19)

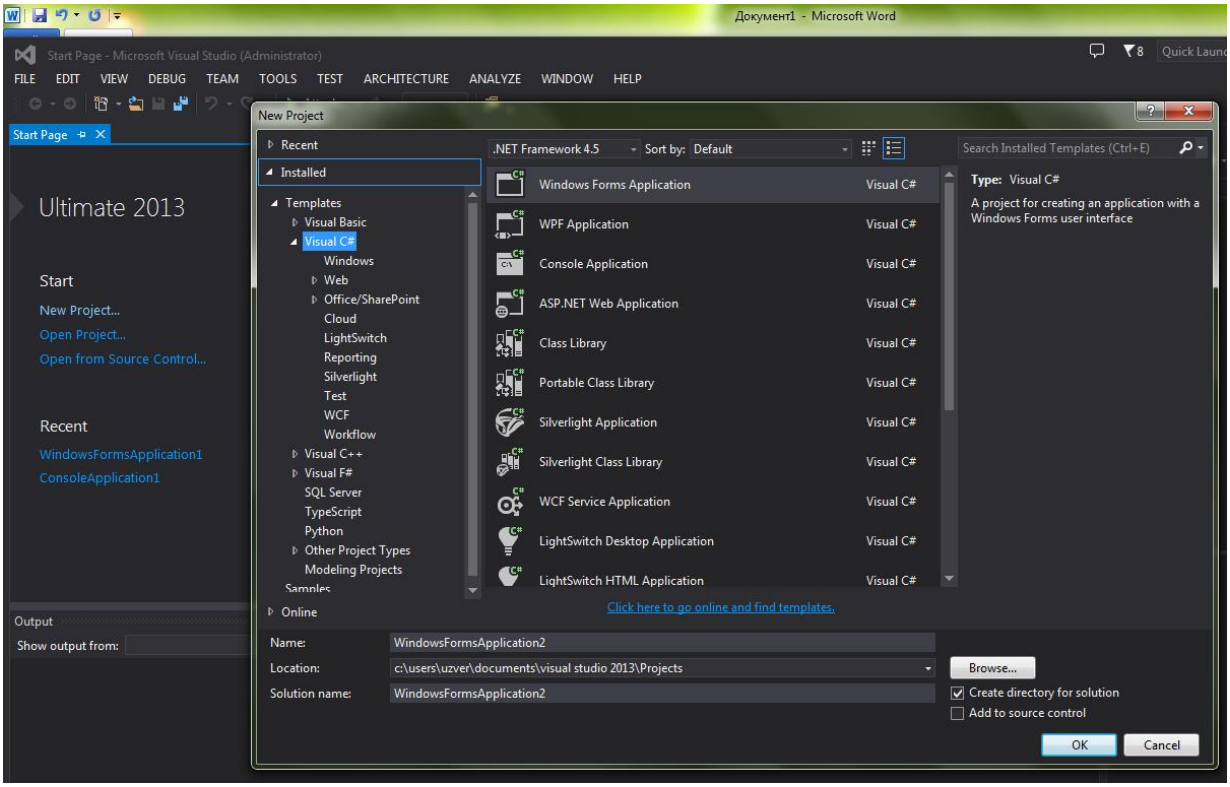


Рисунок 19 - Создание оконного приложения

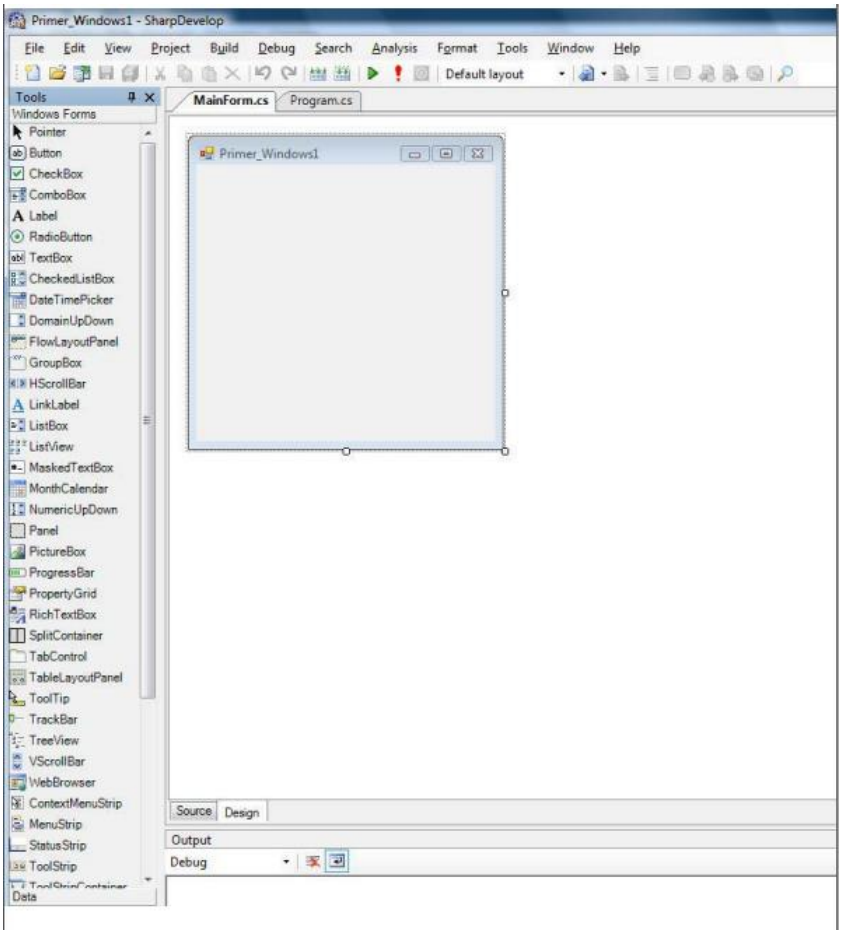


Рисунок 20 – Окно формы

Когда открыта вкладка **Design**, то становится доступно окно **Tools** (на рис. 20 оно расположено слева) — это окно с панелями инструментов, т.е. готовых элементов управления, которые можно размещать на форме.

Также станет доступно окно **Properties** (Свойства). В этом окне в выпадающем списке вверху можно выбирать тот объект на форме, который нужен, и изменять его свойства на этапе проектирования внешнего вида программы.

Разработка оконного приложения состоит из двух основных этапов:

- проектирование внешнего вида приложения;
- написание обработчиков сообщений.

Пример. Даны основание a и высота h треугольника. Вычислить его площадь s.

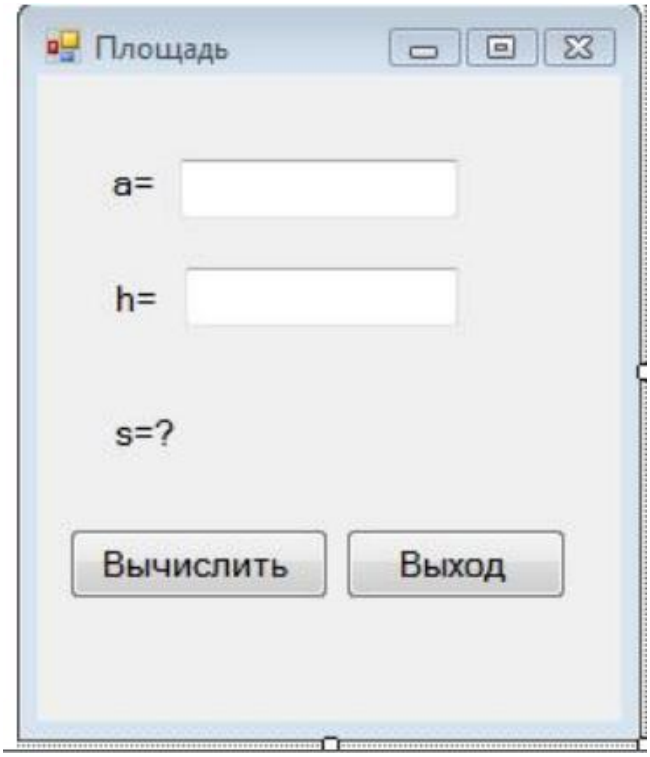
Задача сводится к вводу исходных данных (a и h), вычислению площади и выводу полученного результата.

Вначале на форме разместим два объекта типа **TextBox,** предназначенные для ввода данных (TextBox – простой текстовый редактор, используется для ввода или вывода данных). Объекты по-умолчанию получат имена **textBox1** и **textBox2**. Первый будем использовать для ввода a, второй — для ввода h.

Слева от текстовых редакторов расположим два объекта типа **Label** (Label — надпись, используется для вывода каких либо поясняющих текстов). Их имена будут **label1** и **label2**. В свойствах **Text** этих объектов (находим их в окне **Properties**) напишем строки **«a=»** и **«h=»**, чтобы было понятно, что будет вводится в текстовых редакторах.

Ниже расположим ещё один объект типа **Label (label3)**, напишем в его свойстве **Text** строку **«s=?»**. Этот объект будем использовать для вывода результатов вычисления, т.е. площади треугольника.

Ещё ниже разместим две кнопки — объекты типа **Button**. Они получат имена **button1** и **button2**. В свойстве **Text** для этих кнопок напишем строки **«Вычислить»** и **«Выход»**. Основное назначение объектов этого типа — выполнение каких-либо действий по щелчку. Т.е. пользователь программы щёлкает мышью по кнопке, и программа в ответ выполняет какое-то действие. Но чтобы это произошло, необходимо написать обработчики событий. Так в нашем случае первая кнопка предназначена для вычисления площади, а вторая — для завершения работы программы. В свойстве **Text** самой формы (**MainForm** либо **Form1**) напишем что-нибудь соответствующее решаемой задачи, например: «Площадь». Всё, этап проектирования внешнего вида программы закончен. Вот так будет выглядеть наша форма:



Рисуно 21 – Форма программы

Если сейчас запустить программу на исполнение, то можно будет вводит данные в поля ввода, щёлкать по кнопкам, но программа ни на что не будет реагировать.

Приступаем ко второй части разработки программы — написанию обработчиков событий.

Обработчик событий — это метод особого вида со стандартным перечнем параметров. Используется для реакции программы на те или иные действия пользователя. Обработчик не вызывается непосредственно в тексте программы. Он вызывается средой выполнения. Каждый обработчик связан с каким-либо объектом.

Создать обработчики событий для наших кнопок очень просто. Если открыта вкладка **Design**, то выполните двойной щелчок мышью по каждой из кнопок. Сразу же после двойного щелчка мыши по первой кнопке (**«Вычислить»**) открывается вкладка **Source** и в ней можно увидеть добавленный Дизайнером код заготовки метода **Button1Click()**. Снова перейдите во вкладку **Design** и выполните двойной щелчок по второй кнопке (**«Выход»**). Дизайнер добавит ещё один обработчик — метод **Button2Click()**.

Внутри полученных таким способом заготовок методов (обработчиков событий) напишем необходимый код. Вот текст файла **Form1** (для краткости убраны все комментарии, которые были добавлены системой разработки):

```
using System;
using System.Collections.Generic;
```

```
using System.Drawing;
using System.Windows.Forms;
namespace Primer_Windows1
{
    public partial class MainForm : Form
    {
        public MainForm()
        {
            InitializeComponent();
            }
            void Button1Click(object sender, EventArgs e)
            {
                double a, h, s;
                a = double.Parse(textBox1.Text);
                h = double.Parse(textBox2.Text);
                s = a * h / 2;
                label3.Text = "s=" + s.ToString();
            }
            void Button2Click(object sender, EventArgs e)
            {
                Close();
        }
    }
}
```

Ниже показано окно программы в процессе работы:

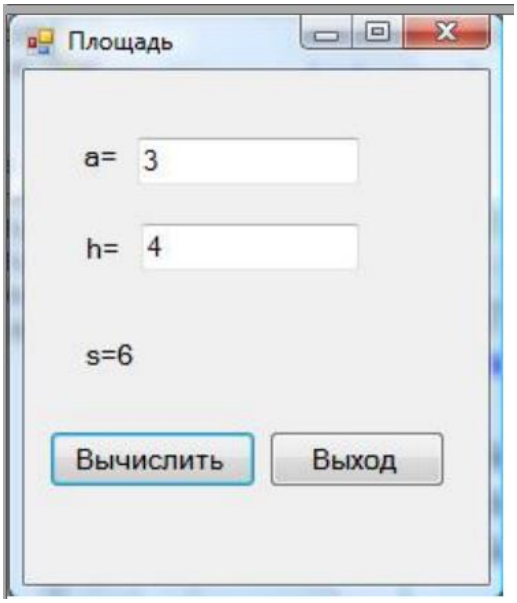


Рисунок 22 – Окно программы после выполнения