ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ
(6 СЕМЕСТР)
ЛАБОРАТОРНАЯ РАБОТА №1.1
УСТАНОВКА И НАСТРОЙКА СРЕДЫ РАЗРАБОКИ

Скирневский И.П.

Томск – 2016

ОГЛАВЛЕНИЕ

ЛАБАРАТОРНАЯ №1.1. УСТАНОВКА И НАСТРОЙКА VISUAL STUDIO

1.1. Среда разработки Microsoft Visual Studio

1.1.1. Общая информация

Выполнение всех лабораторных работ будет выполнятся в среде Microsoft Visual Studio 2015 Community Edition. Visual Studio является одной из самых популярных сред разработки (IDE – Integrated development environment). На текущий момент Studio включает в себя все необходимое компоненты по созданию приложений для мобильных устройств, настольных приложений, веб-приложений и облачных решений. Доступна возможность писать код для iOS, Android и Windows в одной интегрированной среде разработки. Visual Studio обладает удобной и функциональной средой IntelliSense и рядом других преимуществ с которыми мы познакомимся по мере прохождения курса.

На рисунке 1 представлена инфо-графика по продукту Visual Studio 2013 предоставленная ресурсом tadviser.ru

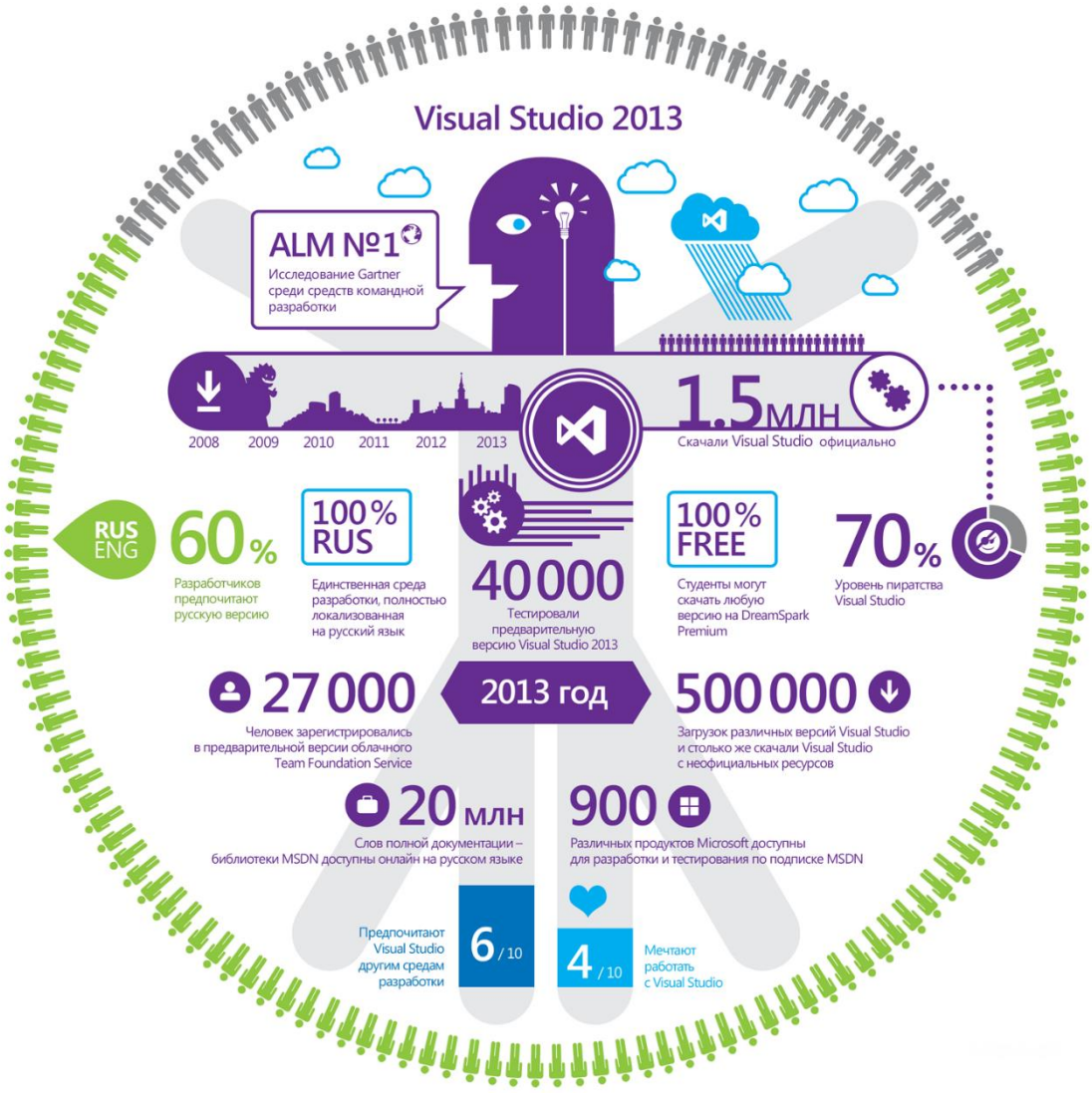

Рисунок 1 – Инфо-графика по продукту Visual Studio 2013

Продукт Visual Studio поставляется с конструкторами графических пользовательских интерфейсов, поддержкой фрагментов кода, инструментами манипулирования базами данных, утилитами для просмотра объектов и проектов, а также встроенной справочной системой. Visual Studio поддерживает множество дополнительных возможностей, наиболее важные из которых перечислены ниже:

- визуальные редакторы и конструкторы XML;
- поддержка разработки приложений для мобильных устройств Windows;
- поддержка разработки приложений для Microsoft Office;
- поддержка визуального конструктора для проектов Windows Workflow Foundation;
- встроенная поддержка рефакторинга кода;
- инструменты визуального конструирования классов.

С большинством перечисленных возможностей студенты познакомиться в процессе выполнения лабораторного практикума.

1.1.2. Загрузка и установка

Что бы скачать Microsoft Visual Studio 2015 Community Edition перейдите по ссылке и нажмите «Скачайте Community бесплатно» (рисунок 2).

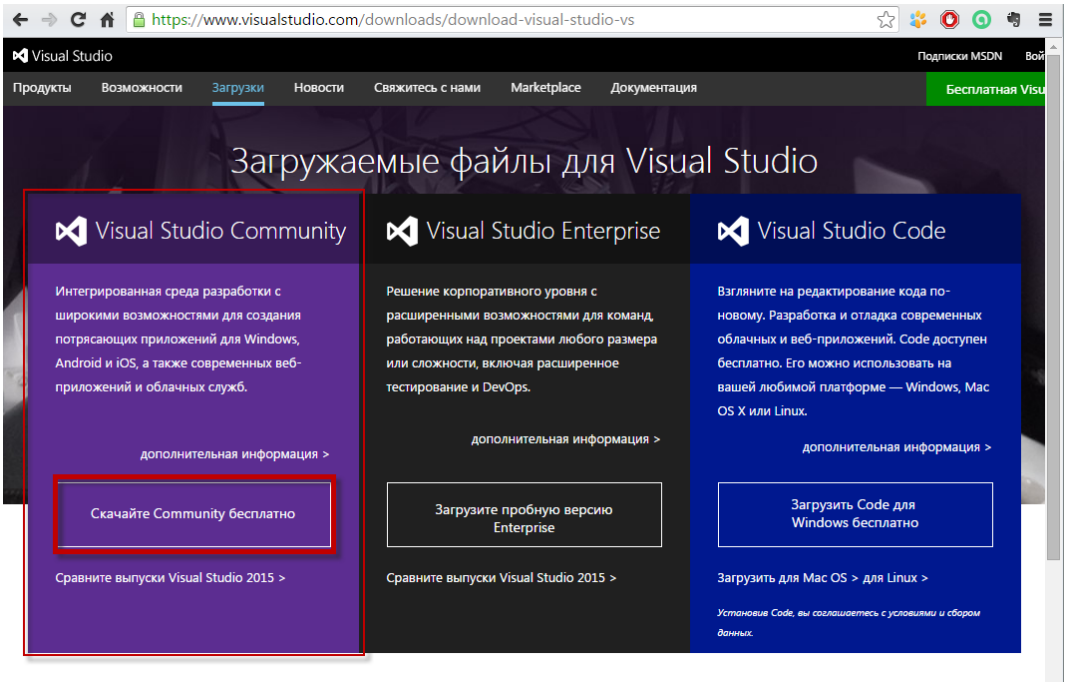

Рисунок 2 – Загрузка Visual Studio

Сохраните файл «vs_community_RUS.exe» в удобную для вас директорию на компьютере. Обратите внимание, что файл является веб-установщиком и все основные пакеты будут загружены на ваш компьютер во время процесса установки. Если вы планируете выполнить установку позже, перейти в раздел «Загружаемые файлы для Visual Studio» –

«Visual Studio 2015» – «Community 2015» и выберете формат загружаемого файла ISO (рисунок 3).

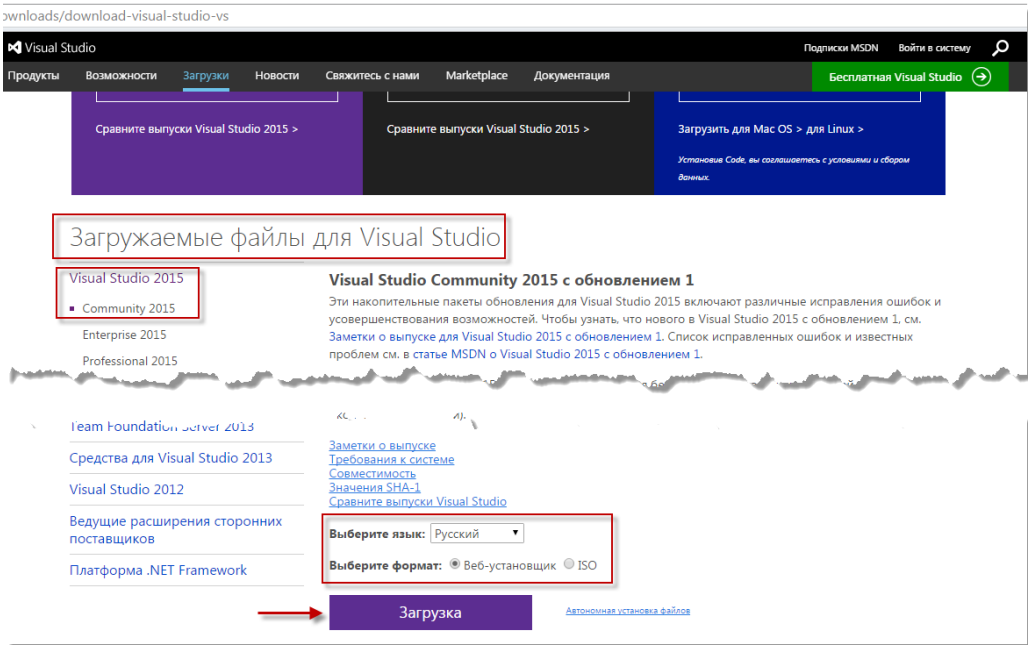

Рисунок 3 – Выбор формата

После запуска установочного файла выберите тип установки Typical и нажмите кнопку Install (в русской версии продукта названия кнопок могут отличаться) (рисунок 4).

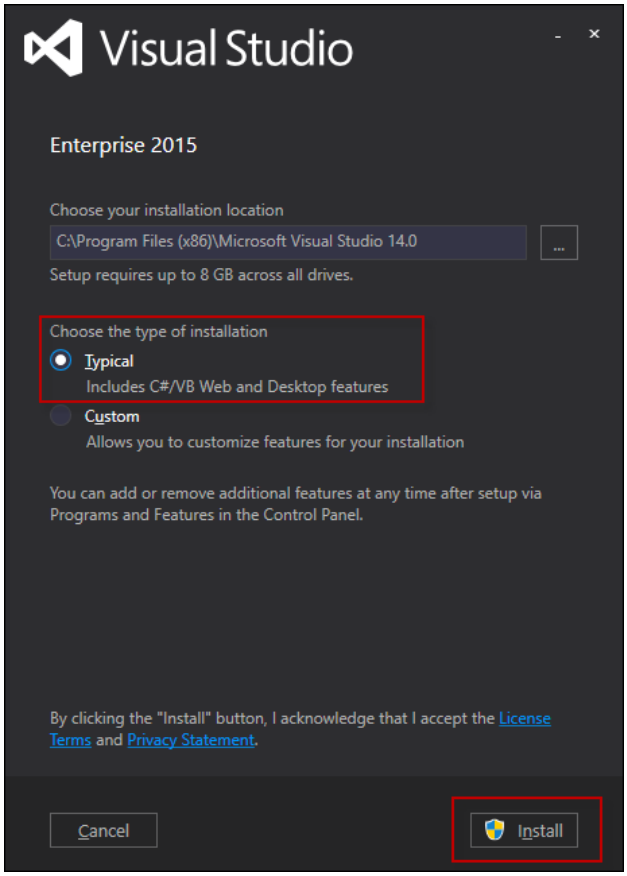

Рисунок 4 – Установка Visual Studio

1.1.3. Первый запуск

После успешной установки запустите Visual Studio из выбранной вами папки либо из меню Пуск. После запуска откроется экран приветствия, представленный на рисунке 5, разберем его ключевые области.

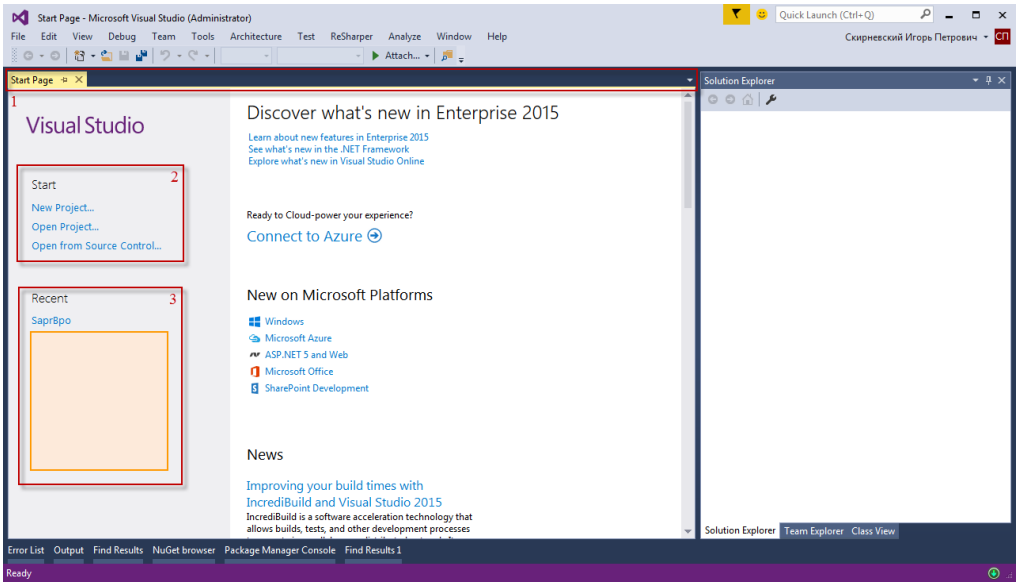

Рисунок 5 – Начальная страница

1) Область вкладок. В данной области будут размещается все открытие окна Visual Studio
2) Область создания/открытия проекта.
3) Область последних открытых проектов

В центральной части окна обычно размещены новости и другая вспомогательная информация. Стоит обратить внимание, что при использовании русской версии Visual Studio названия элементов будут отличаться. В правой части обычно размещается так называемый Обозреватель решения (Solution Explorer), представляющей из себя дерево всех файлов в рамках одного Решения (Solution) или проекта. Мы вернемся к изучению проводника по ходу выполнения лабораторных работ.

Для создания первого проекта нажмите на «Новый проект» (New Project…) в левой части стартового окна. Либо перейдите в меню File – New – Project (Рисунки 6,7).

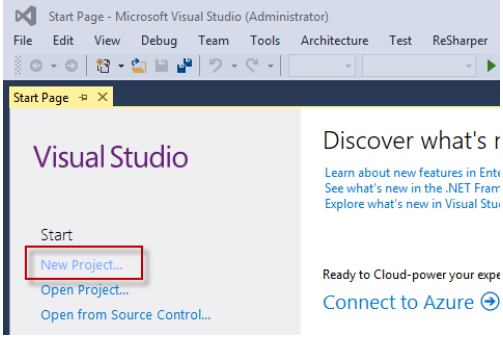

Рисунок 6 – Создание нового файла

Также создание нового проекта доступно через сочетание клавиш Ctrl + Shift + N. С подробным описанием всех «горячих клавиш» можно познакомится, перейдя по ссылке.

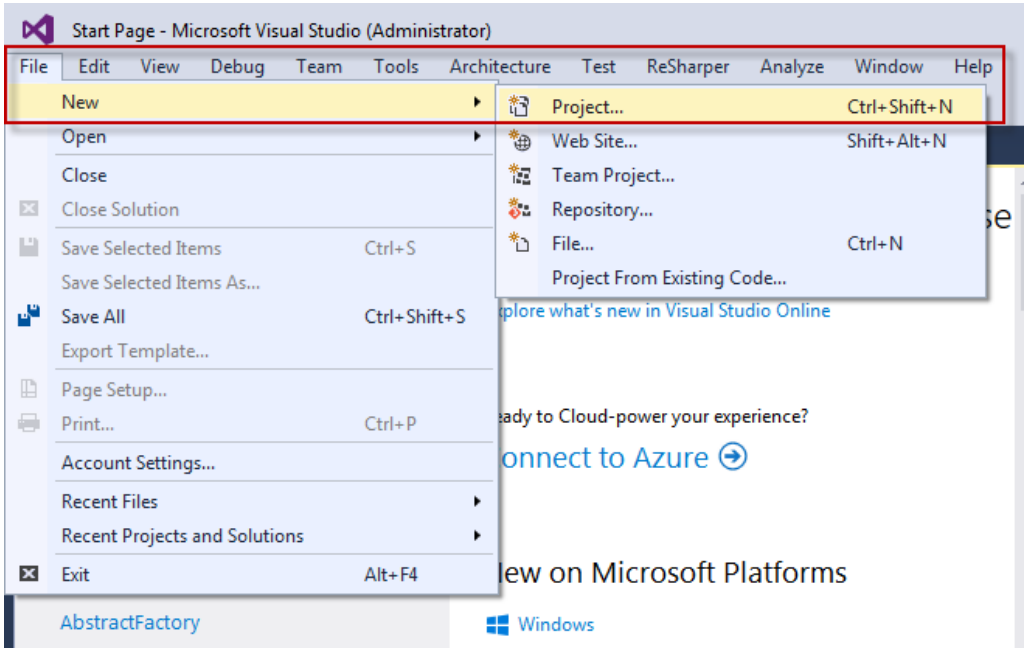

Рисунок 7 – Создание нового проекта через меню File

После нажатия на кнопку создания нового проекта, необходимо выбрать тип проекта. Для этого следует выбрать из раздела шаблонов (Templates) проект Visual C#. Все проекты, созданные в рамках нашего курса, будут выбраны из раздела Visual C#. Стоит также обратить внимание, что в среде Visual Studio существует множество других типов проектов: Visual Basic, F#, C++ и другие.

Для выполнения первой лабораторной работы необходимо выбрать проект Visual C# – Console Application. Обращаю ваше внимание на то, что шаблон проекта Console Application есть в различных подразделах меню. Чтобы не ошибиться перейдите в раздел Windows, как показано на рисунке 8.

В разделе выбора Framework можно ничего не менять и оставить предложенную средой версию. **Важно** корректно заполнить поля Name, Location и Solution Name, которые находится в нижней части экрана.

Поле Name – Название проекта. При заполнении поля Name, поле Solution заполняется тем же именем автоматически. Студентам не следует создавать отдельную директорию под свой проект, Visual Studio создаст ее автоматически.

Пример: Если пользователь создает проект с именем TestProject в директории VS/Projects/, то папка TestProject будет создана автоматически. Если предварительно создать директорию под проект получится следующий не самый удобный путь:

VS/Projects/ TestProjec/TestProject/

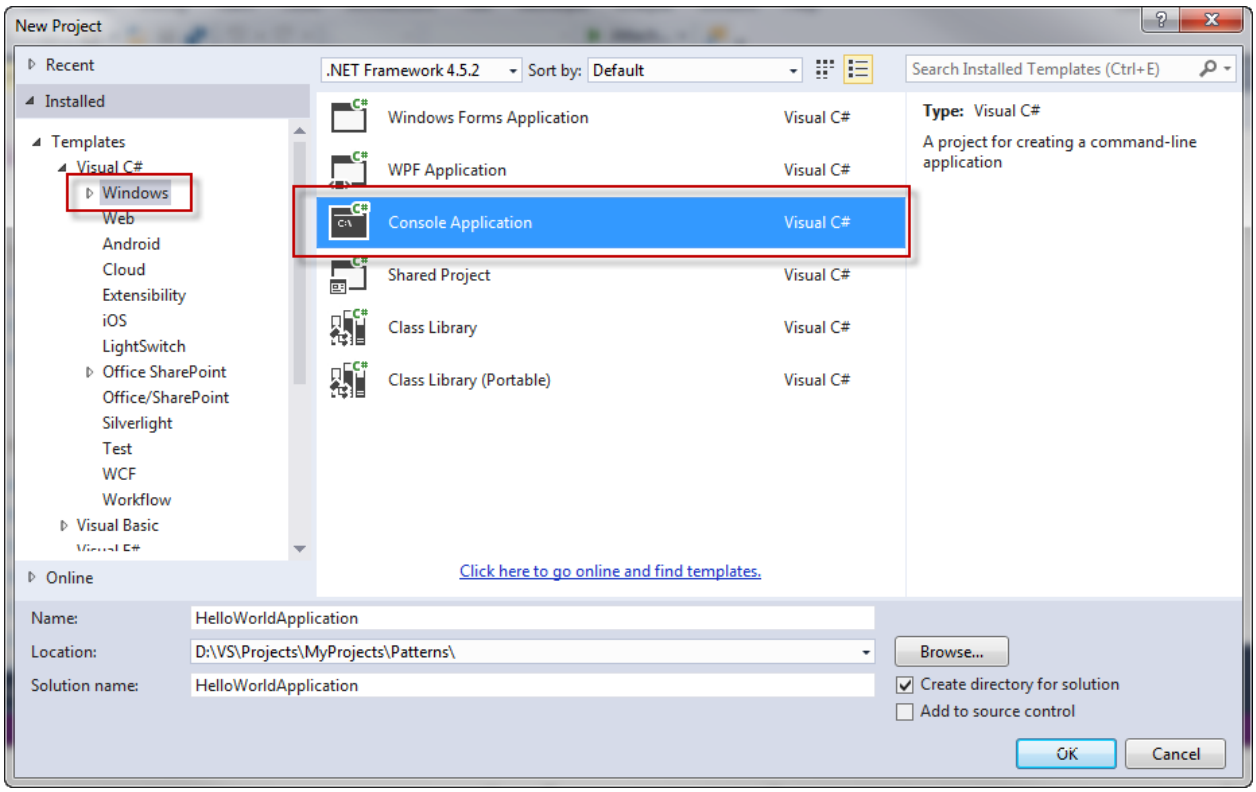

Рисунок 8 – Создание консольного приложения

После создания нового консольного приложения Visual Studio откроет файл Program.cs по умолчанию и Solution Explorer (Обозреватель решения) в правой части окна. (Рисунок 9)

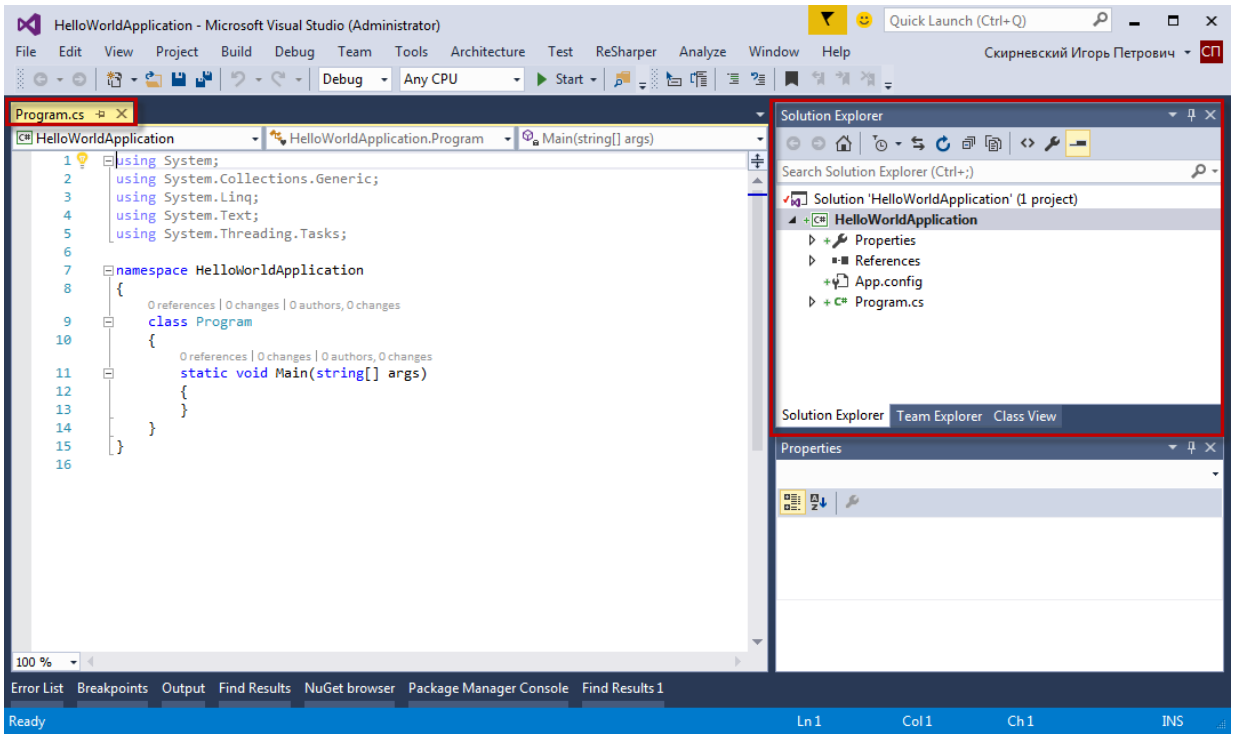

Рисунок 9 – Файл Program.cs после создания нового проекта

1.1.4. Про Solution Explorer

Утилита Solution Explorer (Проводник решений), доступная через меню View (Вид), позволяет просматривать набор всех файлов содержимого и ссылочных сборок, которые входят в состав текущего проекта (рисунок 9). Также обратите внимание, что заданный файл (например, Program.cs) можно раскрыть, чтобы просмотреть определенные в нем кодовые типы. По ходу изложения материала будут указываться и другие полезные особенности Solution Explorer. Однако при желании можете самостоятельно поэкспериментировать с каждой из опций. Кроме того, внутри папки References (Ссылки) в окне Solution Explorer отображается список всех сборок, на которые имеются ссылки. В зависимости от типа выбираемого проекта и целевой версии .NET Framework, этот список выглядит по-разному. Поскольку мы создали консольное приложение, набор автоматически включаемых библиотек минимален (System.dll, System.Core. dll, System.Data. dll и т.д.).

1.2. Структура простой программы

Язык C# требует, чтобы вся логика программы содержалась внутри определения типа. В отличие от многих других языков, в C# не допускается создание ни глобальных функций, ни глобальных элементов данных. Вместо этого все данные-члены и все методы должны содержаться внутри определения типа. Говоря простым языком, весь код, который будет создан в рамках любого проекта будет написан внутри классов или структур, фактически не существует кода «весящего» в пространстве. Мы уже создали первый проект HelloWorldApplication и, как показано ниже, в исходном коде Program.es нет ничего особо примечательного, тем ни менее давайте разберем структуру программы:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace HelloWorldApplication
{
    class Program
    {
        static void Main(string[] args)
        {
        }
    }
}
```

С учетом приведенного кода, модифицируем метод Main () класса Program, добавив в него следующие операторы:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace HelloWorldApplication
{
    class Program
    {
        static void Main(string[] args)
        {
            // Вывести пользователю простое сообщение.
            Console.WriteLine("***** My First C# App *****");
            Console.WriteLine("Hello World!");
            Console.WriteLine();
            // Ожидать нажатия клавиши <Enter> перед завершением работы.
            Console.ReadLine();
        }
    }
  }
```

На заметку! *C# является языком программирования, чувствительным к регистру. Следовательно,Main — не то же самое, что main, а Readline — не то же самое, что ReadLine. Запомните, что все ключевые слова C# вводятся в нижнем регистре (например, public, lock, class, dynamic), а названия пространств имен, типов и членов начинаются (по соглашению) с заглавной буквы и содержат заглавные буквы в любых вложенных в них словах (как, например, Console.WriteLine, System.Windows.MessageBox и System.Data.SqlClient). Как правило, при каждом получении от компилятора ошибки, связанной с неопределенными символами, в первую очередь следует проверить регистр символов и точность написания имен.*

И так, мы имеем определение типа класса, который поддерживает единственный метод по имени *Main ()*. По умолчанию среда Visual Studio назначает классу, определяющему метод *Main ()*, имя Program, однако, при желании это можно изменить. Каждое исполняемое приложение C# (консольная программа, программа рабочего стола Windows или Windows-служба) должно содержать класс, определяющий метод *Main ()*, который используется для обозначения точки входа в приложение.

Формально класс, который определяет метод *Main ()*, называется объектом приложения. Хотя в одном исполняемом приложении разрешено иметь несколько объектов приложений (это может быть удобно при модульном тестировании), потребуется проинформировать компилятор о том, какой из методов *Main ()* должен использоваться в качестве точки входа.

Обратите внимание, что сигнатура метода *Main()* снабжена ключевым словом *static*, о котором будет написано ниже. Пока же достаточно знать, что область действия статических членов охватывает уровень класса (а не уровень объектов), поэтому они могут вызываться без предварительного создания нового экземпляра класса.

За время прочтения абзацев, написанных выше, студенты познакомились с рядом не знакомых слов и, если читателю начинает казаться, что он запутался и уже ничего не понимает, отчаиваться не стоит, в конце данного раздела мы разберем все новые слова подробно.

Кроме ключевого слова *static* этот метод *Main ()* принимает один параметр, который представляет собой массив строк *(string [ ] args)*. Хотя в текущий момент этот массив никак не обрабатывается, данный параметр может содержать любое количество входных аргументов командной строки. И, наконец, этот метод Main () был сконфигурирован с возвращаемым значением *void*, которое означает, что мы не определяем явно возвращаемое значение с помощью ключевого слова *return* перед выходом из области действия метода.

Логика Program содержится внутри самого метода *Main ()*. Здесь используется класс Console, который определен в пространстве имен System. В состав его членов входит статический метод *WriteLine (),* который позволяет отправлять строку текста и символ возврата каретки на стандартное устройство вывода. Кроме того, здесь вызывается метод *Console.ReadLine ()*, чтобы окно командной строки, запускаемое в IDE-среде Visual Studio, оставалось видимым во время сеанса отладки до тех пор, пока не будет нажата клавиша <Enter>.

Ключевые элементы приложения HelloWorldApplication

Давайте рассмотрим структуру первого приложения еще раз, для удобства читателя, код, который приведен выше скопирован еще раз в данный раздел.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace HelloWorldApplication
{
    class Program
    {
        static void Main(string[] args)
        {
            // Вывести пользователю простое сообщение.
            Console.WriteLine("***** My First C# App *****");
            Console.WriteLine("Hello World!");
            Console.WriteLine();
            // Ожидать нажатия клавиши <Enter> перед завершением работы.
            Console.ReadLine();
        }
    }
  }
```

1. Директива Using

В C# ключевое слово using упрощает процесс добавления ссылок на типы, определенные в заданном пространстве имен. Другими словами, using подключает необходимые библиотеки к вашему проекту. Так, если вы решили в коде своего приложения использовать функции работы с файлами вам следует добавить в проект пространство имен System.IO (приведено только в качестве примера). Возникает логичный вопрос, как держать в голове, все нужные подключения и где получить информацию по нужным библиотекам?

На самом деле, среда Visual Studio сама подскажет, какого пространства имен не хватает для работы и предложит добавить его в код проекта.

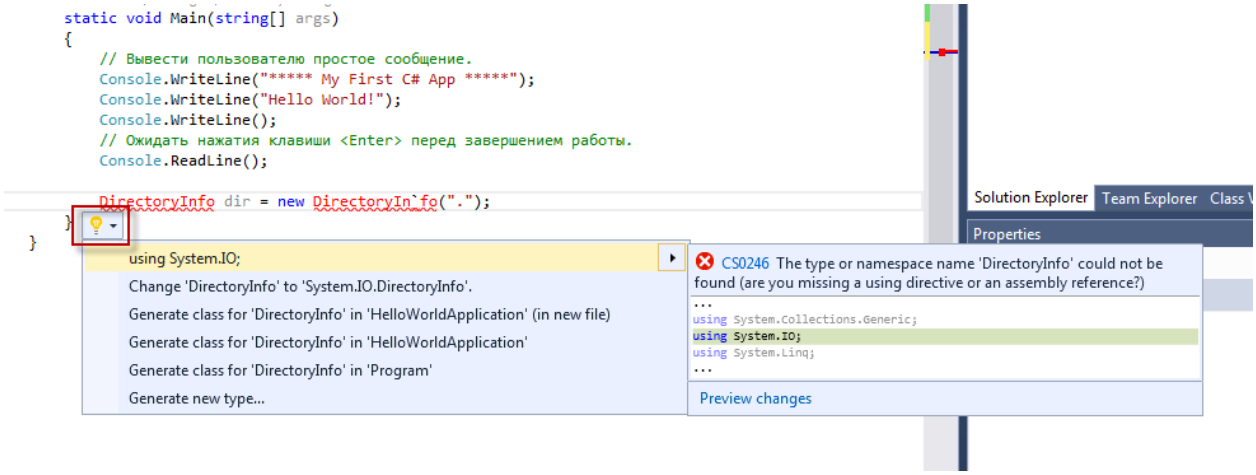

Рисунок 10 – Подсказка о пропущенном пространстве имен

Также всегда можно воспользоваться справкой, для этого выделите участок кода и нажмите F1.

Ниже приведена таблица 1 с основными типами, которые могут понадобится студентам в своих проектах. Но, как можно было заметить Visaul Studio при создании пустого проекта самостоятельно добавляет ряд подключений самых популярных библиотек, что значительно упрощает разработку.

Таблица 1 – Пространства имен .NET

Пространство имен .NET	**Описание**
System	Внутри пространства имен System содержится множество полезных типов, предназначенных для работы с внутренними данными, математическими вычислениями, генерацией случайных чисел, переменными среды и сборкой мусора, а также ряд часто применяемых исключений и атрибутов
System.Collections System.Collections.Generic	Эти пространства имен определяют набор контейнерных типов, а также базовые типы и интерфейсы, которые позволяют строить настраиваемые коллекции

System.Data System.Data.Common System.Data.EntityClient System.Data.SqlClient	Эти пространства имен используются для взаимодействия с базами данных через ADO.NET
System.IO System.IO.Compression System.IO.Ports	Эти пространства имен определяют множество типов, предназначенных для работы с файловым вводом-выводом, сжатием данных и портами
System.Drawing System.Windows.Forms	Эти пространства имен определяют типы, применяемые для построения настольных приложений с использованием исходного инструментального набора .NET для создания пользовательских интерфейсов (Windows Forms)
System.Windows System.Windows.Controls System.Windows.Shapes	Пространство имен System.Windows является корневым для нескольких пространств имен, которые представляют инструментальный набор для построения пользовательских интерфейсов Windows Presentation Foundation (WPF)
System.Linq System.Xml.Linq System.Data.DataSetExtensions	Эти пространства имен определяют типы, применяемые во время программирования с использованием API-интерфейса LINQ
System.Xml	В пространствах имен, связанных с XML, содержатся многочисленные типы, используемые для взаимодействия с XML-данными

2. Namespace (пространство имен)

Мы не будем подробно останавливаться на следующем участке кода, а именно пространстве имен. Заметим лишь, что при создании нового проекта, в нашем случае, HelloWorldApplication Visual Studio самостоятельно создает одноименное пространство имен.

Объявление собственного пространства имен поможет в управлении областью действия имен классов и методов в крупных программных проектах.

3. Class Program

В первой лабораторной работе не будет затрагиваться тема объектно-ориентированного программирования, по этому строчку class Program мы пропустим и вернемся к ней в следующих лабораторных работах. Единственное, что стоит отметить, как писалось выше, весь код, написанный в рамках платформы .NET должен принадлежать какому-либо классу или структуре. В нашем случае, даже самая примитивная программа находится в рамках класса Program и размещается в нем автоматически.

4. static void Main(string[] args)

Данная строка описывает функцию Main – входную точку программы. Функции будут детально рассмотрены в следующей лабораторной работе.

Внутри фигурных скобок функции Main расположен основной код программы, структура которого в данной лабораторной выводит строчки ***** My First C# App ***** и Hello World!" на экран.

Вывод на консоль осуществляется при помощи статического метода WriteLine класса Console, который принимает на вход строку. Под словом «метод» в контексте данной лабораторной работы можно понимать некоторую функцию. Как вы думаете, функция WriteLine имеет возвращаемый тип? Или является void (т.е. ничего не возвращает?)

Строка Console.ReadLine() необходима для того, чтобы консоль не закрылась автоматически после вывода сообщений на экран, а дождалась нажатия клавиши пользователем. Попробуйте закомментировать данную строку и запустите приложение еще раз (запуск проекта подробно описан в следующей части).

Справка!

Закомментированный код игнорируется компилятором и нужен для создания примечаний и пояснений к коду. В языке C# существует два вида комментариев. Однострочный: «//» два слеша – комментирует только одну строчку. И многострочный: « / текст */ » .*

1.3. Запуск программы

После того, как код написан необходимо выполнить запуск и отладку приложения. Для того, чтобы запустить приложения необходимо нажать F5 или воспользоваться кнопкой Start в среде Visual Studio (рисунок 11)

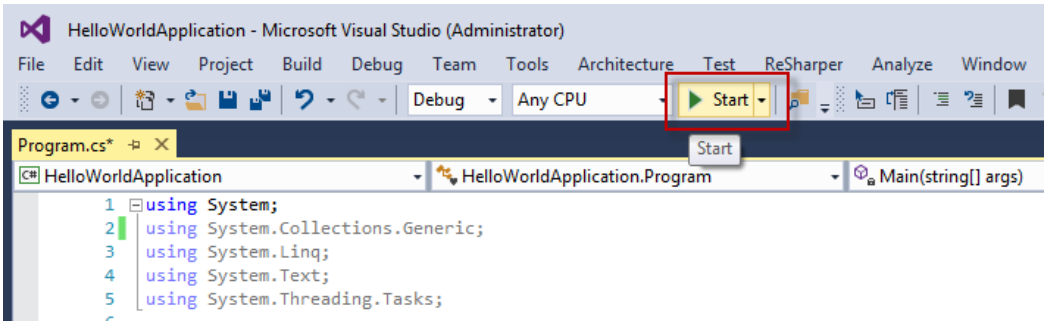

Рисунок 11 – Запуск проекта

Расширенный список опций доступен в разделе меню Debug (рисунок 12).

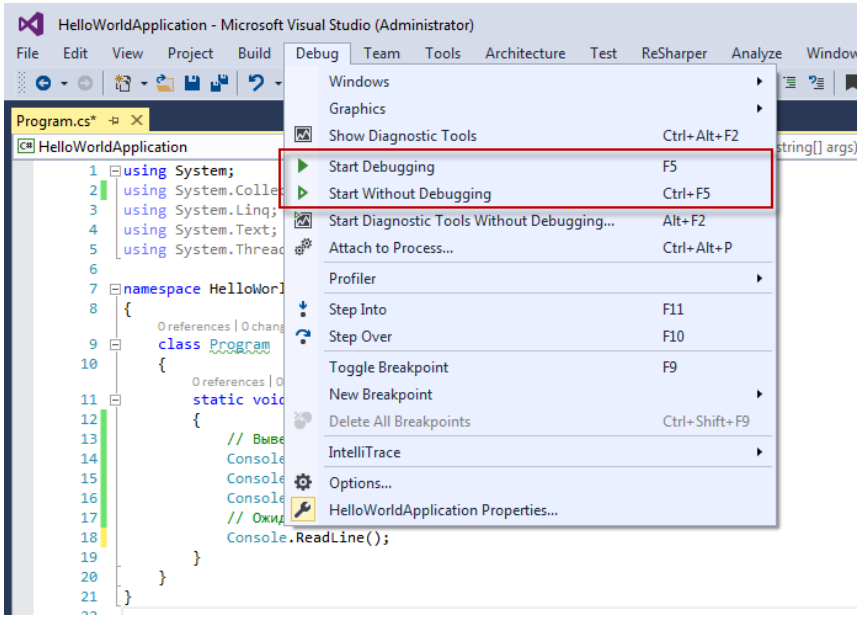

Рисунок 12 – Меню Debug

Если ни каких ошибок не было найдено приложение будет успешно запущено и откроется консоль с текстом (рисунок 13).

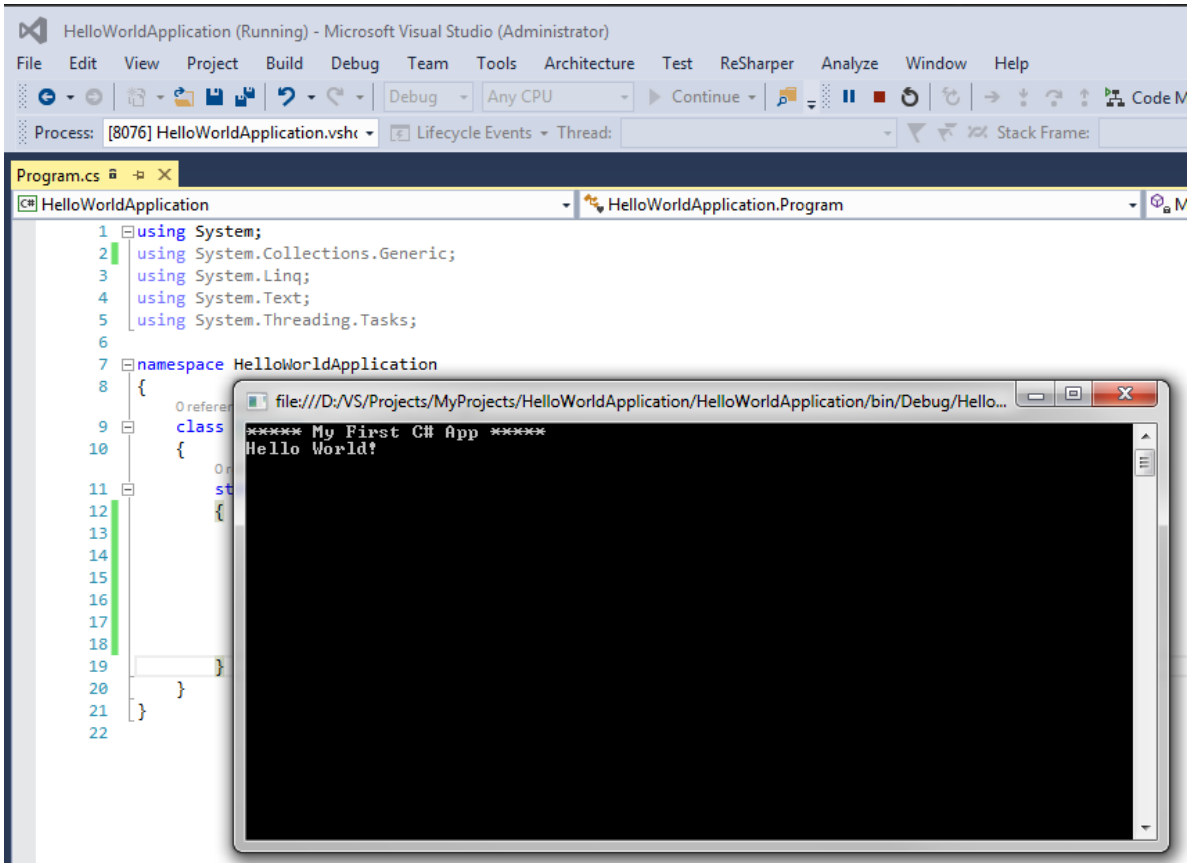

Рисунок 13 – Запуск проекта

Для завершения работы приложения можно нажать любую клавишу, либо закрыть окно. Также остановить запущенную программы позволяет сочетание клавиш Shft + F5.

Если в процессе анализа исходного кода компилятор обнаружит синтаксические ошибки (ошибки компиляции), Visual Studio сообщит о месте локализации ошибки.

Добавим в проект строчку, как показано на рисунке и попробуем запустить программу.

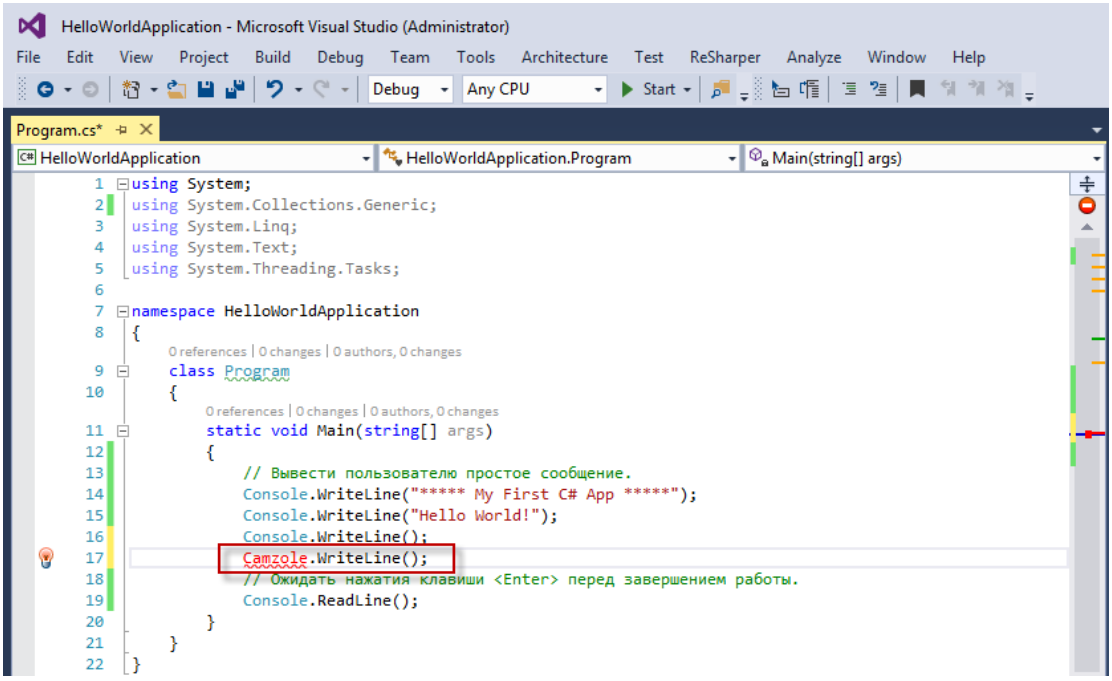

Рисунок 14 – Синтаксическая ошибка в коде проекта

После запуска приложения (кнопка F5, для тех, кто забыл) компилятор сообщит об ошибке и приложения не будет запущено. (рисунок 15)

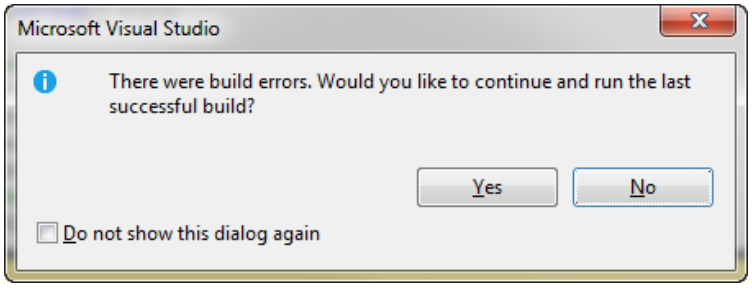

Рисунок 15 – Предупреждение о наличии ошибок

После нажатия на кнопку «No», автоматически всплывет окно Error List с детальным отчетом об ошибке. (рисунок 16).

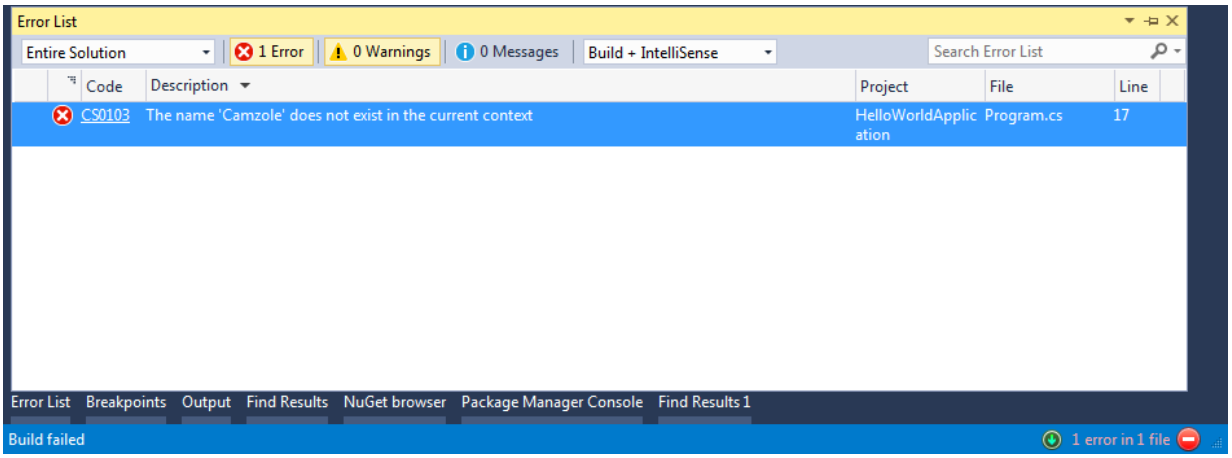

Рисунок 16 – Сообщение об ошибке

Таблице состоит из нескольких столбцов. Наиболее интересные из которых «Description» и «Line». В нашем случае ошибка звучит как: «*The name 'Camzole' does not exist in the current context*», что можно трактовать как, не существует имени Camzole в текущем контексте. Это значит, что в нашем проекте нет функции или класса с таким именем, ровно, как и в подключенных нами ранее библиотеках.

Колонка Line указывает номер строки, где произошла ошибка, в нашем случае это строка 17. Двойное нажатие на строку ошибки перебросил курсор на строчку в коде проекта.

Другие тонкости отладки будут рассмотрены в следующих лабораторных работах. На этом выполнение первой лабораторной закончено.

ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ №1

1. Установить среду Microsoft Visual Studio
2. Создать проект Консольное приложение c#. Для названия проекта используйте следующий синтаксис Lab#_Familia. Пример Lab1_Ivanov. *Лабораторные с неправильно оформленным названием могут быть не приняты или случайно удалены.*
3. Вывести на экран две строчки:
   a. В первой строке «Hello World!»
   b. Во второй строке: ФИО и номер зачетной книжки.
4. Удалить из проекта все не нужные блоки using (которые были добавлены средой по умолчанию)
5. Добавить в проект строчку содержащую ошибку и попытаться запустить приложение
6. Исправить ошибку и повторить запуск
7. Оформить отчет с описанием хода выполнения вашей лабораторной работы.

Требования к оформлению отчета

1 Формат листа – А4 (210 х 297 мм). Поля: слева – 25 мм; справа – 15 мм; сверху, снизу – 20 мм.

2 Основной текст документа должен быть напечатан шрифтом Times New Roman Cyr, размером 12 пт. Межстрочный интервал – 1.5, отступ красной строки – 1.25. Не добавлять разрыв между абзацами одного стиля. Выравнивание по ширине.

3 Размер заголовков – 12 пт. Заголовок первого уровня – Все заглавные, жирный, заголовок второго и последующих уровне – жирные, как в предложении. Отступ заголовка – 1.25. Выравнивание по левому краю.

4 Нумерация страниц производится автоматически.

5 Если позволяет структура документа, автоматически оформить оглавление.

6 Титульный лист оформлен по стандарту ТПУ.

7 Рекомендуемая структура отчета: Титульный лист, Оглавление (если требуется) Введение, Ход работы, Заключение. В разделе ход работы описывается последовательность выполнения заданий, если надо приводятся участки кода и скриншоты.

Требования к защите

Лабораторная считается защищенной при наличии трех файлов:

1. Архив с исходным кодом
2. Исполняемый файл
3. Отчет по лабораторной работе