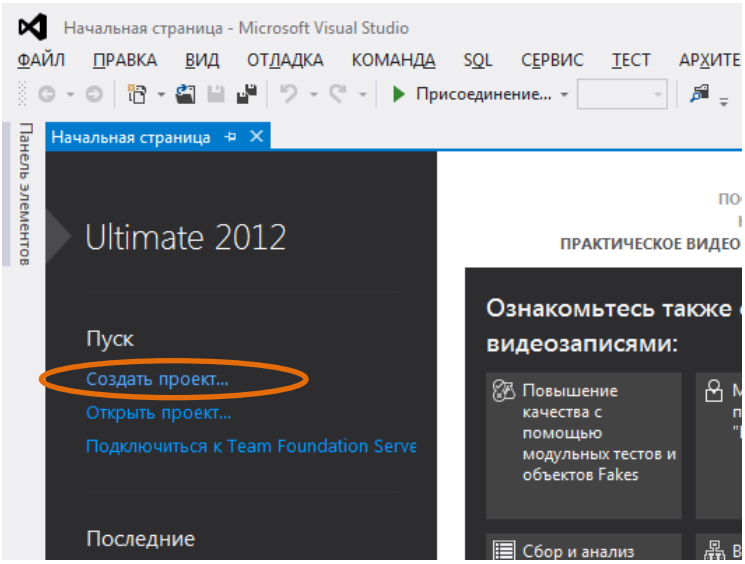
Тема 1. Базовые понятия и определения

Цель работы. Рассмотреть базовые понятия и определения, элементарные типы данных, основные операторы и выражения языка C#. Познакомиться с примерами простейших программ, демонстрирующих приемы работы с элементарными типами данных и операторами C#.

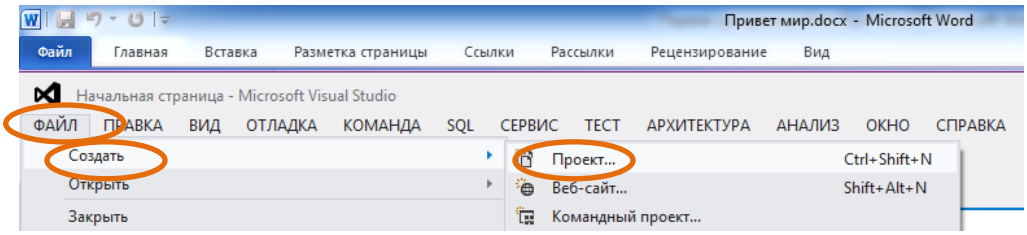
Задание 1. Создание консольного приложения «Привет мир!»

Практически все учебники по языкам программирования начинаются с изучения исходного текста программы, отображающей на консоли текстовую строку «Hello, world!».

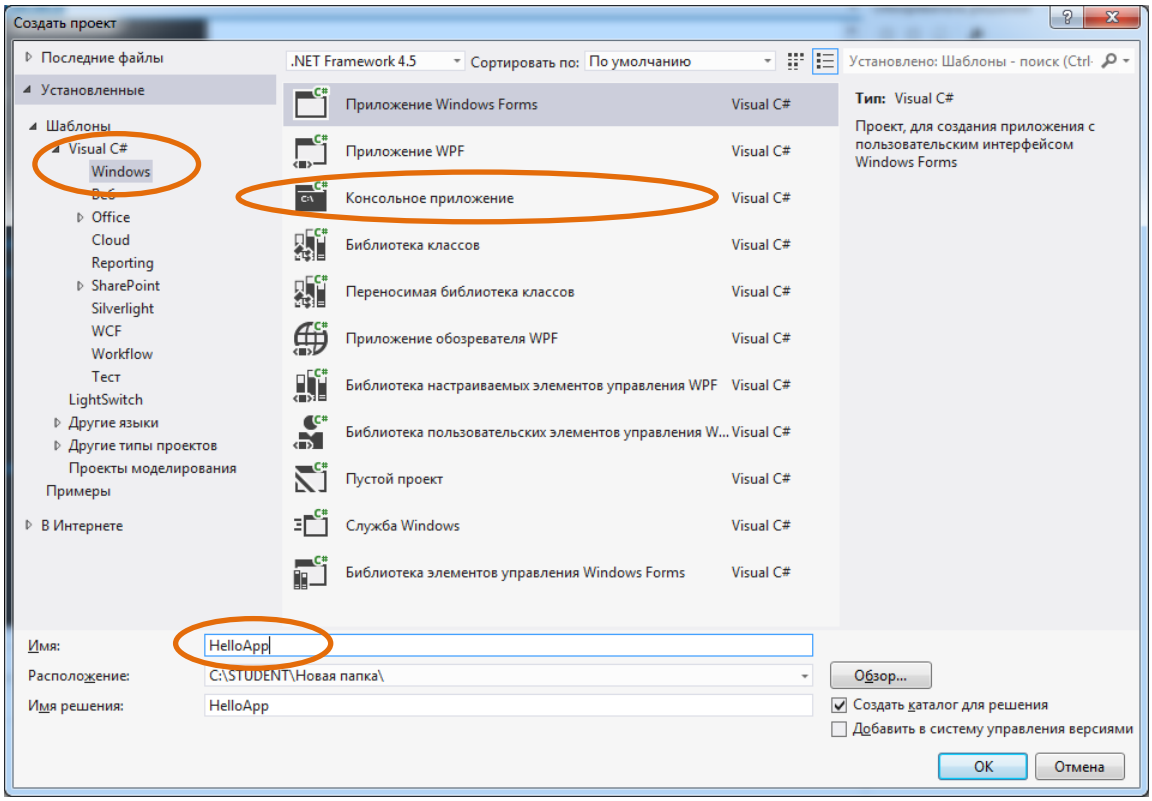
Для создания первого консольного проекта потребуется среда разработки Visual Studio. Запускаем Visual Studio и создаем проект консольного приложения.



либо

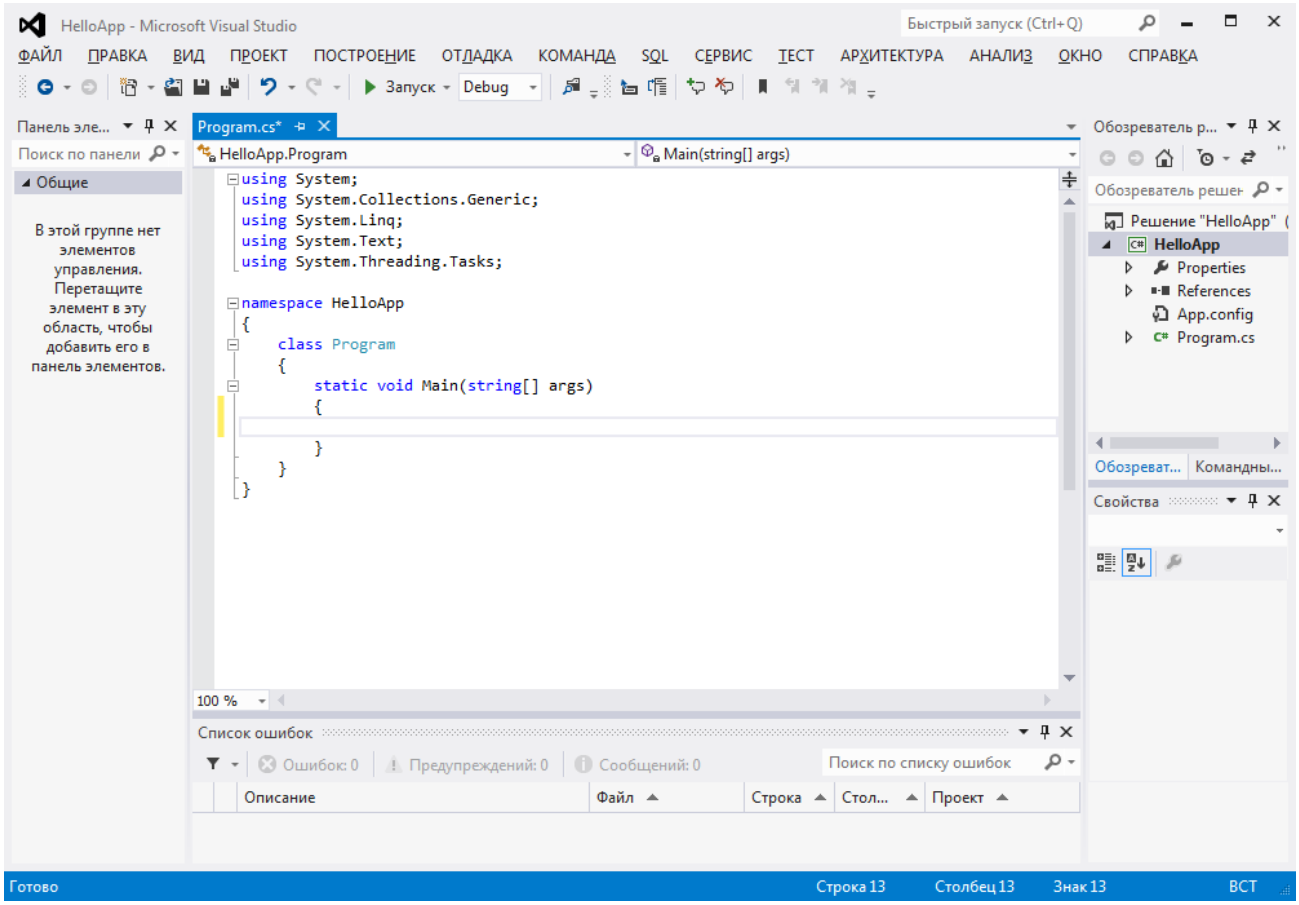


После этого откроется диалоговое окно создания нового проекта:



В левой колонке выберем язык C#, далее Windows, а в центральной части среди типов проектов - тип **Консольное Приложение** и дадим ему какое-нибудь имя в поле внизу. Например, назовем его *HelloApp*. После этого нажимаем OK.

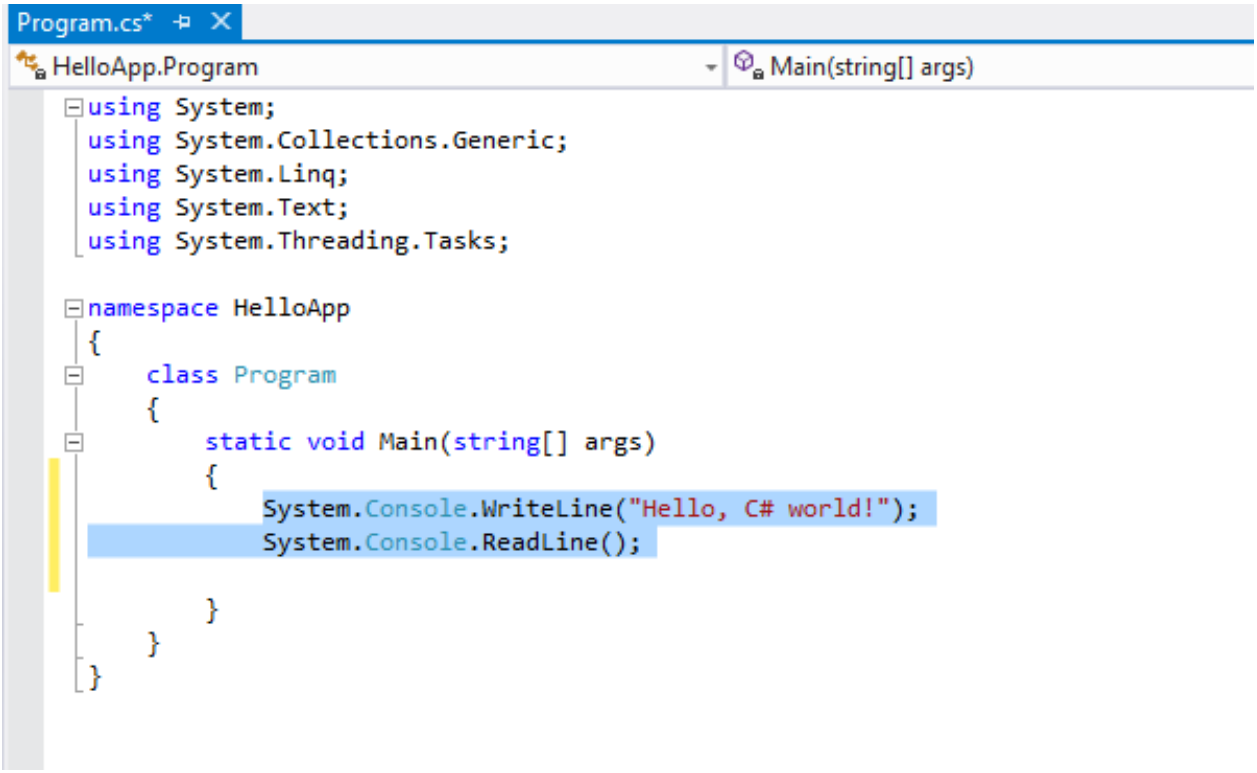
После этого Visual Studio откроет проект с созданными по умолчанию файлами:



Большую часть окна занимает поле для создания текста программы с заготовленными системными установками.

Исходный текст программы

Наша первая программа C# сразу после запуска выводит на консоль строку «Hello, C# world!», а затем ждет, когда вы нажмете клавишу Enter на клавиатуре компьютера. Исходный текст программы представлен ниже, наберите его в окне редактирования



Исходный текст программы состоит всего из нескольких строк, причем некоторые из них используются только для описаний программы и ее объектов, а некоторые непосредственно исполняются и приводят к появлению видимых результатов.

На данном этапе вам нужно представлять себе только общую структуру этой программы, не вникая в подробности.

Сохраните проект. Запустите его и убедитесь в его работоспособности.

Пространство имен System

Первая строка нашей программы содержит ключевое слово using и предписывает компилятору просматривать в процессе своей работы так называемое пространство имен System:

```
using System;
```

В состав среды выполнения программ Microsoft Framework .NET входит обширная библиотека, насчитывающая десятки тысяч классов. Сильно упрощая, скажем, что классы представляют собой описания некоторых данных и методов работы с этими данными.

Пользуясь классами как кирпичиками (или как прототипами), можно создавать весьма и весьма сложные программы, не затрачивая на это колоссальных усилий. Чтобы компилятор мог ориентироваться в названиях классов, а также пределенных в рамках этих классов символических именах, в языке C# используются пространства имен.

Указывая при помощи ключевого слова using пространство имен System, мы открываем компилятору доступ к классам, необходимым, в частности, для ввода текстовых строк с клавиатуры и вывода их на консоль. В своих примерах программ мы постоянно будем использовать пространство имен System и другие пространства имен.

Определение собственного пространства имен

Любая, даже простейшая программа C# создает свои классы. Она также может определять для этих классов собственные пространства имен. Такое определение делается при помощи ключевого слова **namespace:**

```
namespace Hello
```

После ключевого слова namespace указывается параметр— имя определяемого пространства имен. В данном случае наше пространство имен будет называться Hello. С помощью фигурных скобок мы ограничиваем строки программы, имеющие отношение к определяемому пространству имен.

Класс HelloApp

Как вы скоро узнаете, все данные в языке C# представляются в виде объектов некоторых классов. Наша программа тоже создает класс HelloApp, в котором определен единственный метод Main:

```
class HelloApp
{
static void Main()
     {
          System.Console.WriteLine("Hello, C# world!");
          System.Console.ReadLine();
     }
}
```

Название класса задается параметром оператора class, а содержимое класса располагается внутри фигурных скобок:

```
class HelloApp
```

Для своей первой программы мы выбрали произвольное название содержащего ее класса— HelloApp. Это сокращение от Hello, Application. В названии класса мы отразили назначение класса и всего приложения — отображение приветственного сообщения. Но, строго говоря, здесь мы могли бы использовать любое допустимое название.

Метод Main

Как мы уже говорили, классы представляют собой некоторые данные и методы для работы с ними. В нашем приложении определен класс HelloApp, а в этом классе — метод Main:

```
static void Main()
     {
          System.Console.WriteLine("Hello, C# world!");
          System.Console.ReadLine() ;
```

```
}
```

Отвлечемся пока от ключевых слов static и void, а также от круглых скобок, расположенных после имени метода Main. Чтобы система Microsoft .NET Framework могла запустить приложение, в одном из классов приложения необходимо определить метод с именем Main. Этот метод нужно сделать статическим, снабдив ключевым словом static, иначе ничего не получится. **Запомните** пока просто как аксиому, что в программе обязательно должен быть статический метод Main, определенный подобным образом. Именно этот метод получает управление при запуске приложения. Позже мы проясним ситуацию с ключевыми словами static и void.

Тело метода Main ограничено фигурными скобками, внутри которых находятся два оператора:

```
System.Console.WriteLine("Hello, C# world!");
System.Console.ReadLine();
```

Первый из них выводит строку «Hello, C# world!» на консоль, а второй ожидает, пока кто-нибудь не введет с клавиатуры произвольную строку и не нажмет клавишу Enter.

В первой строке нашего метода мы обращаемся к методу WriteLine, предназначенному для вывода данных на консоль. Этот метод определен в классе Console, который принадлежит упоминавшемуся ранее пространству имен System. В круглых скобках методу WriteLine передаются параметры, определяющие, что собственно нужно выводить на консоль. В данном случае мы выводим текстовую строку «Hello, C# world!», ограничив ее двойными кавычками.

Метод ReadLine тоже определен в классе Console из пространства имен System. Он предназначен для получения текстовой строки, введенной с консоли. Мы не передаем методу ReadLine никаких параметров. Единственное назначение метода ReadLine в нашей программе — приостановить ее работу после вывода на консоль строки сообщения «Hello, C# world!». Если этого не сделать, то при запуске программы в среде ОС Microsoft Windows консольное окно с сообщением появится на очень короткое время, а затем будет автоматически уничтожено. В результате вы не успеете ничего рассмотреть. Поэтому многие примеры консольных программ будут завершаться вызовом метода ReadLine или ReadKey.