Урок 1

Введение. Базовые типы данных. Консоль. Классы и методы

Что такое .NET Framework? Создаем консольное приложение в VisualStudio. Переменные. Типы данных и их преобразование. Выводим и вводим данные через консоль. Первое знакомство с методами и классами.

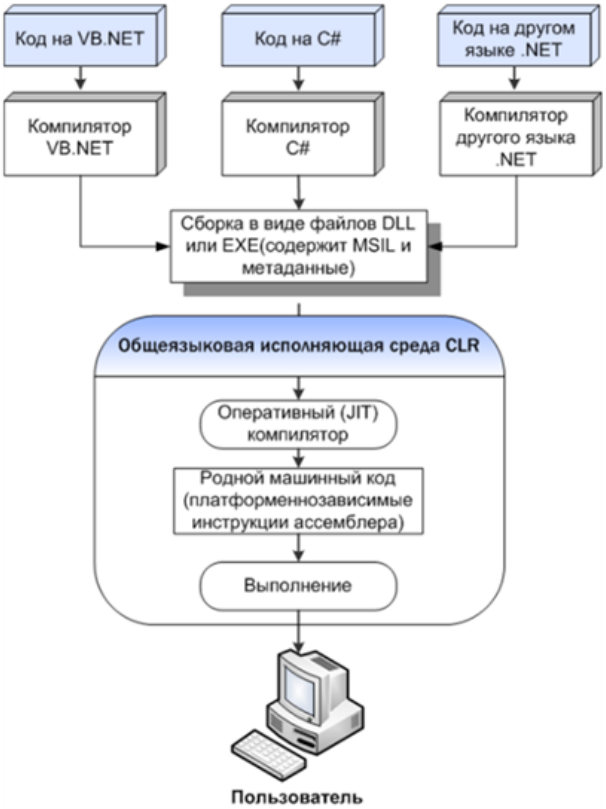
C# и .NET FRAMEWORK

C# и .Net Framework неразрывно связаны друг с другом. .Net Framework — это технология, разработанная Microsoft, которая упрощает написание программ для операционных систем, мобильных устройств, сайтов и других разработок Microsoft. C# является специально разработанным языком поддержки .Net Framework. Хотя под .Net Framework можно программировать и на других языках, в C# реализована полная поддержка этой технологии.

CLR, MSIL, управляемый код

Необходимо понимать, что при написании программы на C# по умолчанию программа компилируется в так называемый управляемый код MSIL (промежуточный язык), который выполняется с помощью CLR (общеязыковой среды выполнения). Это позволяет обеспечить перенос программы с одной платформы на другую,даёт дополнительную защиту от ошибок и ряд других преимуществ. Правда, с небольшой потерей в производительности.
Управляемый код — это код, который выполняется в CLR. В C# есть возможность выйти за рамки управляемого кода, если важны критерии производительности или есть другие потребности при написании программы.

Схема компиляции .NET-приложения:



Столбовский Д.Н.
Разработка Web-приложений ASP .NET
с использованием Visual Studio .NET

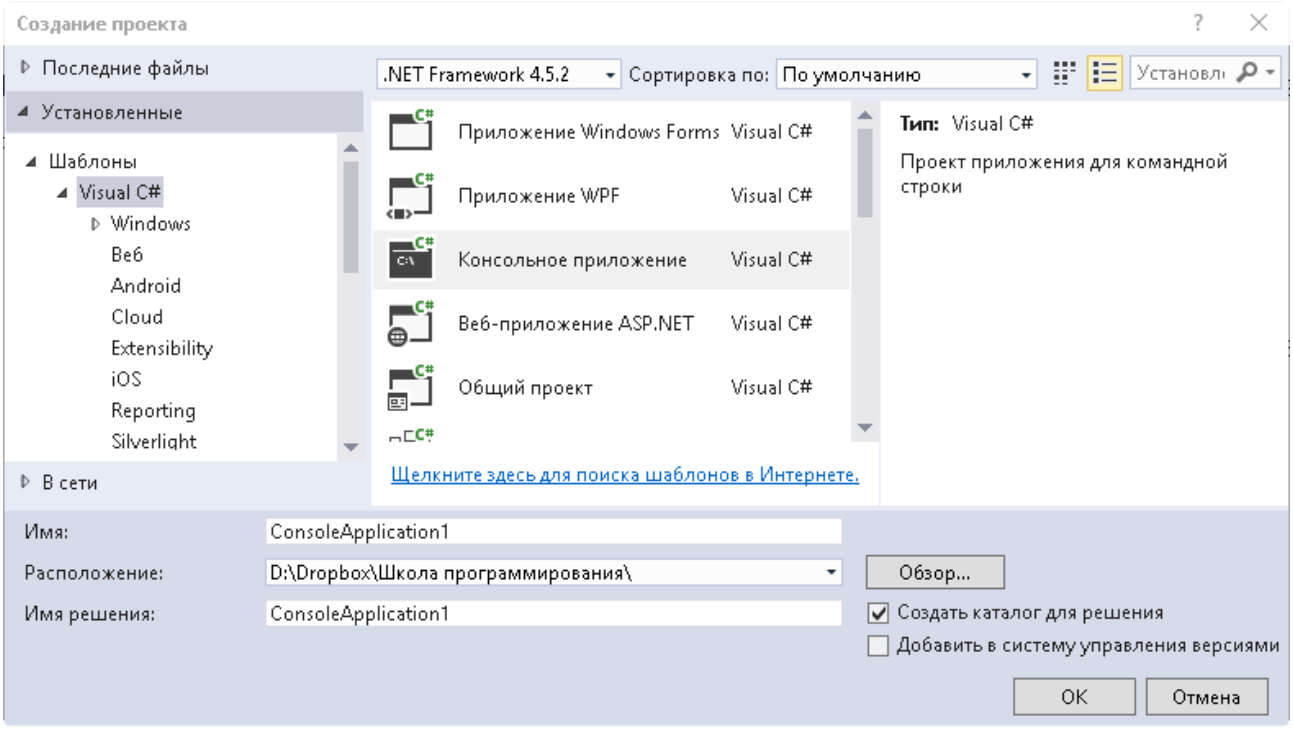
Visual Studio

Visual Studio (VS) — это интегрированная среда разработки (IDE), которая существенно облегчает жизнь программистам. С первого взгляда может показаться, что VS — это одна программа. На самом деле VS состоит из множества программ: компилятора, отладчика, редактора форм, утилиты для работы с базами данных и других.

Существует множество различных версий VS. Для обучения подходят бесплатные версии Visual Studio Express или Community.

Создание консольного приложения

Для создания проекта запустите VS. Выберите Шаблон Visual C# — Консольное приложение. Нажмите OK.



Расположение файлов

При создании нового решения, если не снять галочку с пункта «Создать каталог для решения», вы получите папку, внутри которой располагается файл решения с расширением sln и подпапки для каждого проекта.

IntelliSence

IntelliSence — технология автодополнения Microsoft, наиболее известна в Microsoft Visual Studio. Дописывает название функции при вводе начальных букв. Кроме прямого назначения, IntelliSense используется для доступа к документации и для устранения неоднозначности в именах переменных, функций и методов при помощи рефлексии (доступ к информации о структуре объекта).

Отладка программ

Существует отладчик, который позволяет управлять выполнением программы и смотреть, как изменяются переменные.

Наиболее часто используемые клавиши для взаимодействия с отладчиком и редактором VS:

Клавиша	Команда
<F9>	Добавление/снятие точки останова
<F12>	Переход к определению, объекта или метода
<Ctrl>+<M>	Разворот и сворачивание структуры кода в редакторе
<Ctrl>+<K>+<C>	Комментирование строки кода
<Ctrl>+<K>+<U>	Раскомментирование строки кода
<F5>	Запуск с отладкой
<Ctlr>+<F5>	Запуск без отладки
<F10>	Трассировка с обходом
<F11>	Трассировка со входом

Для отладки программ можно использовать окна «Отладка» и «Стек вызовов». Если вдруг их нет на экране, включите их в меню «Вид» — «Панели инструментов».

Директива region и комментарии

Прежде чем мы приступим к написанию программ, узнаем, как можно сделать код более удобочитаемым. Для этого предназначены директивы region, endregion и комментарии. Директивы — это конструкции, которые в C# начинаются со значка # и являются указаниями среде выполнения или компилятору, как нужно выполнять программу.

Комментарии — это текст, с помощью которого можно описать свою программу. Комментарии пропускаются компилятором. Программисты часто используют их, чтобы убрать код, не удаляя его на самом деле из программы.

```
#region ***Директива #region позволяет указать блок кода, который можно разворачивать и сворачивать***
#region Пример однострочного комментария
//Одностройный комментарий
#endregion
#region Пример многострочного комментария
/* Можно запускать программу через Ctrl-F5,
 * тогда программно паузу создавать не обязательно
  Пример многострочного комментария
 */
#endregion
Простейшая программа
#endregion ***Конец region***
```

Простая программа

```
using System;
namespace Lesson1
{
  class Program
  {
      static void Main()
      {
      }
  }
}
```

Элементы простой программы

1. Директива using System объясняет, что мы можем не указывать пространство имен System в имени класса.
2. Пространства имен — контейнеры для классов.
3. Класс — логическая единица программы на C#, в которой содержатся методы и другие элементы.
4. Метод — подпрограмма программы, в которой содержится реализация алгоритма.

Как правило, программы содержат методы. Один метод является особенным.

Главный метод Main

В языке C# метод Main определяет в тексте программы место, где начинается исполнение самой программы. В тексте отдельной программы может существовать только один метод Main. Метод Main может отсутствовать только в случае, если ваша программа должна не выполняться, а использоваться другой программой. Например, когда вы пишете библиотеку классов.

Операции в C#

1. Арифметические бинарные операции (*, /, +, -, %).
2. Арифметические унарные операции (++, --).
3. Присваивание (=).
4. Операции отношения (<, >, ==, !=, >=, <=).

Приоритет операций от наивысшего к низшему:

1. Инкремент, декремент.

2. Умножение, деление, получение остатка.
3. Сложение, вычитание.

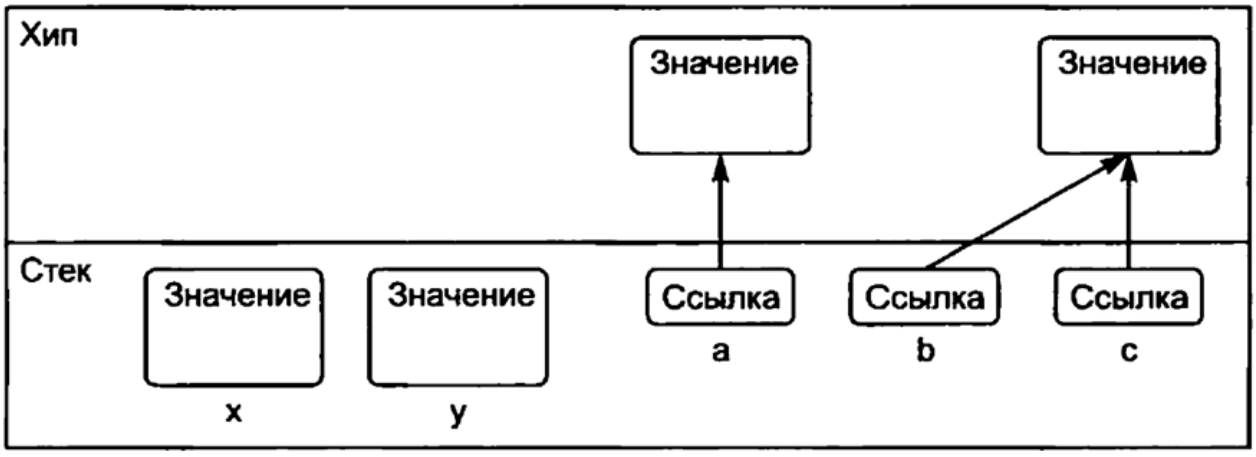
Переменные

Переменные — это именованные ячейки памяти, в которых хранятся данные. Какие данные может хранить ячейка, определяется её типом. Тип определяет также и размер ячейки. Программист должен уметь правильно выбирать типы данных для переменных в зависимости от решаемой задачи.

Типы данных

В C# типы делятся на значимые и ссылочные. Значимые хранят значения, а ссылочные — ссылки на ячейки, в которых уже содержатся значения.
Если переменная относится к типу значения, она содержит само значение, например 3,1416 или 2016.

Если переменная относится к ссылочному типу, она содержит ссылку на значение в общей памяти, которая называется «Куча» (Heap). По этой ссылке хранятся данные.



Псевдонимы типов данных в C#

При описании типа данных можно использовать тип .NET, а можно псевдоним C#.

Со знаком		**Без знака**	
Тип .net	Псевдоним C#	Тип .NET	Псевдоним C#
System.Object	object	System.Enum	enum
System.String	string	System.Char	char
System.Sbyte	sbyte	System.Byte	byte
System.Int16	short	System.UInt16	ushort
System.Int32	int	System.UInt32	uint

System.Int64	long	System.UInt64	ulong
System.Single	float	System.Double	double
System.Decimal	decimal	System.Boolean	bool

Целочисленные типы

Тип	Разрядность в битах	Диапазон представленных чисел
byte	8	0 — 255
sbyte	8	-128 — 127
short	16	-32 768 — 32 767
ushort	16	0 — 65 535
int	32	-2 147 483 648 — 2 147 483 647
unit	32	0 — 4 294 967 295
long	64	-9 223 372 036 854 775 808 — 9 223 372 036 854 775 807
ulong	64	0 — 18 446 744 073 709 551 615

Типы для представления чисел с плавающей запятой

float	32 бита	диапазон значение от 5E-45 до 3,4E+38
double	64 бита	диапазон значений от 5E-324 до 1,7E+308

Десятичный тип данных

Тип decimal предназначен для ведения финансовых расчетов.

decimal	128 бит	Диапазон значений от 1E-28 до 7.9E+28

Символы

В C# символы представлены 16-разрядным кодом Unicode. Для работы с этим типом данных существует класс Char.

Пример:

```
Char.IsDigit(c)      // проверка, является ли символ числом
```

Строки

Строка описывается словом string или System.String. Строка является ссылкой на массив символов char. Строки можно склеивать между собой.
К каждому символу можно обращаться по его номеру:

```
string s="string";

string str=s[0].ToString()+s[1]+s[2]; // str="str"
```

Логический тип данных

Логический тип данных хранит в себе значение «Истина» или «Ложь». В переменной этого типа хранится значение операции отношения(<,>,<=,>=,==).

```
bool b;      // объявили переменную b типа bool

b=false;     // переменной b присвоили значение false

b=2*2==4;    // переменной b присвоили значение true
```

Логические операции и их таблицы истинности

Оператор	**Значение**
&	И
|	ИЛИ
^	Исключающее ИЛИ
&&	Укороченное И
||	Укороченное ИЛИ
!	НЕ

p	**q**	**p & q**	**p | q**	**p ^ q**	**!p**
false	false	false	false	false	true
true	false	false	true	true	false

false	true	false	true	true	true
true	true	true	true	false	false

Неявно типизированные переменные

Неявно типизированные переменные дают некоторое удобство для программистов, которое позволяет описать тип переменной альтернативным способом. В этом случае выбор типа возлагается на компилятор.

```
var a=10; // компилятор объявит переменную типом int
```

Не путайте неявно типизированные переменные с динамическими переменными, тип которых может изменяться в процессе выполнения программы.

Преобразование и приведение совместимых типов

При вычислении выражений может возникнуть необходимость в преобразовании типов. Если операнды, входящие в выражение, одного типа и операция для этого типа определена, то результат выражения будет иметь тот же тип. Если операнды разного типа и/или операция для этого типа не определена, перед вычислениями автоматически выполняется преобразование типа по правилам, обеспечивающим приведение более коротких типов к более длинным.

Если неявного преобразования из одного типа в другой не существует, программист может задать явное преобразование типа с помощью операции (<название типа>)<имя переменной>. Эту операцию можно использовать только для совместимых типов. Например, нельзя таким способом привести данные из целого в строку. Для этого нужно использовать специальные методы.

```
double a=3.14;
int b=(int)a;      // Так можно, но с потерей дробной части
string s="3.14";
int c=(int)s;      // Так нельзя
```

Область видимости переменных

У переменных существует понятие «область видимости». Если переменную объявили внутри некоторого блока { и }, то снаружи этого блока переменная не видна.

```
{
  int i=1;
}
                  // Здесь i уже не существует
```

Консоль

Консоль является способом взаимодействия пользователя с компьютером. Для программиста консоль — один из самых простых способов взаимодействия.

Работа с консолью в C# происходит с помощью класса Console, который содержит в себе большое количество методов и свойств для взаимодействия с консолью.
Что такое методы, вы узнаете на этом уроке. Свойства пока можно воспринимать как переменные, которые принадлежат классу Console.

Вывод на экран консоли

Для вывода данных на экран консоли используются методы Write и WriteLine. Разница лишь в том, что WriteLine, в отличие от Write, переводит курсор на следующую строчку после вывода текста на экран консоли. То есть если следующий вывод (или ввод) нужно делать со следующий строки, используем WriteLine.

```
using System;

class Program
{
    static void Main(string[] args)
    {
        Console.Write("Не переходим на следующую строку.");
        Console.WriteLine("Переходим на следующую строку");
    }
}
```

У этих методов есть множество разновидностей, которые профессионально называются перегрузками.

Можно использовать перегрузку методов Write или WriteLine, которая преобразует заданное значение в строку и выведет полученную строку на экран консоли. А можно использовать перегрузку, в которой вначале задаётся строка форматирования, а потом значения для вывода. В этом случае у программиста появляется больше возможностей для управления выводом.

Управляющие последовательности символов

Управляющая последовательность	**Описание**
\n	Новая строка (перевод строки)
\r	Возврат каретки
\t	Горизонтальная табуляция
\'	Одинарная кавычка

\"	Двойная кавычка
\\	Обратная косая черта

Форматированный вывод

Различные спецификации формата в применении к целому числу 12 345.
Например:

```
Console.WriteLine("{0:D7}",12345);
```

Тип форматирования	Код формата	Результат
Currency (денежные суммы)	C C1 C7	$12,345.00 $12,345.0 $12,345.0000000
Decimal (десятичный)	D D1 D7	12345 12345 0012345
Exponential (экспоненциальный)	E E1 E7	1.234500E+004 1.2E+004 1.2345000E+004
Fixed point (с фиксированной точкой)	F F1 F7	12345.00 12345.0 12345.0000000
General (общий)	G G1 G7	12345 1E4 12345
Number (числовой)	N N1 N7	12,345.00 12,345.0 12,345.0000000
Percent (процент)	P P1 P7	1,234,500.00 1,234,500.0 1,234,500.0000000
Hexadecimal (шестнадцатеричный)	X X1 X7	3039 3039 0003039

Ввод данных с консоли

Программа должна обрабатывать данные, а данные нужно откуда-то брать. Ввод данных с консоли — наиболее простой способ, хотя нужно понимать, что данные можно вводить из разных мест.

Для чтения данных из консоли существует метод ReadLine. Этот метод приостанавливает выполнение программы, пока не получит признак конца ввода (обычно это нажатие клавиши Enter). После чего передаёт данные из консоли в переменную.
При работе с ReadLine может возникнуть исключение. Про исключения мы поговорим позже. На текущий момент воспринимайте их как ошибки.

Так как метод ReadLine класса Console возвращает строку (то есть результатом его работы является строка), часто её нужно преобразовать в другой тип данных. Это можно сделать разными способами:

1. Использовать метод класса Convert.
2. Использовать метод Parse или TryParse структур int, double, decimal и др.

Разница в их использовании на данном этапе нам не принципиальна, поэтому можно использовать любой из способов.

```
double x;
string str = Console.ReadLine();
x = Convert.ToDouble(str);
```

Функция или метод

Функции и методы — это технически одно и то же. Только функции могут не принадлежать классам, а методы принадлежат классу. В C# все функции являются методами.

Описание метода

Метод является частью класса, поэтому описываться он должен внутри класса. Чтобы описать метод, нужно придумать ему имя, определить, будет ли он возвращать значение. Если будет, то какого типа это значение. Далее в фигурных скобках надо описать тело метода. Так как мы ещё не очень знакомы с объектами, все методы у нас должны быть статическими. Если метод статический, он принадлежит классу, а для его вызова не нужно создавать объект класса.

Вызов метода

Для вызова метода без параметров достаточно написать имя метода и круглые скобки. Круглые скобки после названия — признак, что это метод, а не переменная или свойство. Если метод принимает параметры, в скобках перечисляем фактические параметры (то, что передаётся внутрь метода для обработки).

```
using System;

class Program
{
```

```
    static void Pause()
    {
        Console.ReadKey();
    }

    static void Main(string[] args)
    {
        Pause();      // Вызов метода
    }
}
```

Возвращаемое значение

Напишем для примера функцию возведения целого числа в квадрат. Для этого используем ключевое слово return. Если функция возвращает значение, вместо void нужно указать тип возвращаемого значения.

```
using System;

class Program
{
  static int Sqr(int x)
  {
    return x*x;
  }
 static void Main(string[] args)
  {
    int value = 8;
    Console.WriteLine("Квадрат числа "+ value + " = " + Sqr(value));
  }
}
```

Перегрузка методов

Перегрузка — это создание метода с таким же именем, но с другими параметрами.

Раньше для подпрограмм, которые совершали похожие действия, создавали функции с близкими, но разными именами. В C# для этого можно создать методы с одним именем, но с разными параметрами. Это существенно облегчает труд программистов, так как не требуется запоминать различные имена функций, а также упрощает чтение программы.

Несмотря на кажущуюся сложность, в перегрузке нет ничего сверхъестественного. Попробуйте сами написать несколько функций с одинаковыми названиями, но с разным количеством параметров. При попытке использовать в программе перегруженный метод Intellisense подскажет вам, что существуют разновидности данного метода.

```
using System;

class Program
```

```
{

        static void Pause()                     // Создали метод
        {
            Console.ReadKey();
        }

        static void Pause(string msg)    // Перегрузили метод
        {
            Console.WriteLine(msg);
            Console.ReadKey();
        }

    static void Main(string[] args)
    {
        Pause("Перегруженный метод");
    }

}
```

Класс Math

Класс Math содержит методы для вычисления математических функций. Например, Pow(a,b) — метод, который возвращает a в степени b.

У методов есть сигнатуры. Сигнатура — это тип возвращаемого значения и параметры. Если метод перегружен, он может иметь несколько сигнатур. У Pow(a,b) сигнатура double(double, double). Это означает, что Pow возвращает тип double и принимает два параметра типа double. Мы не можем передавать в Pow несовместимые типы (например, string), и количество параметров должно равняться двум.

Рекомендации по программированию

1. Приступая к написанию программы, чётко определите, что является её исходными данными, что требуется получить в результате.
2. Давайте переменным имена, отражающие их назначение. Общая тенденция такова: чем больше область действия переменной, тем более длинное у нее имя. Напротив, для переменных, вся «жизнь» которых проходит на протяжении нескольких строк кода, лучше обойтись однобуквенными именами типа i или k.
3. Аккуратно форматируйте текст программы так, чтобы его было удобно читать.

Практическая часть урока

Задача 1. Написать программу сложения двух чисел.

```
using System;
```

```
    class Program
    {
        static void Main(string[] args)
        {
            double x;
            double y;
      Console.Write("Введите первое число: ");
            string str = Console.ReadLine();
            x = Convert.ToDouble(str);
            Console.Write("Введите второе число: ");
            y = Convert.ToDouble(Console.ReadLine());
            double z = x + y;
              Console.WriteLine(x + "+" + y + "=" + z);      // Преобразование в
строку
            Console.ReadLine();
        }
    }
```

Задача 2. Вывести значение функции ax^2+bx+c в точке x. x — ввести с клавиатуры, a,b и c — присвоить в программе.

```
using System;

    class Program
    {
        static void Main(string[] args)
        {
            double a = 1;
            double b = 1;
            double c = 1;
            double x;
      Console.Write("Введите значение x: ");
            string s = Console.ReadLine();
            x = Convert.ToDouble(s);
            double f = a*Math.Pow(x, 2) + b * x + c;
              Console.WriteLine("{0}*x^2+{1}*x+{2}, при x={3}, f={4}", a, b, c,
x, f);
            Console.ReadLine();
        }
    }
```

Задача 3. Обменять значениями две переменные.

```
class Program
{
    static void Main(string[] args)
    {
            int a=10;      // Присваиваем начальное значение
            int b=20;      // Присваиваем начальное значение
            int t=a;         // В t запоминаем значение a
            a=b;            // В a записываем b
             b=t;            // В b записываем сохраненное a
    }
}
```

Задача 4. Разработать метод проверки чётности числа. Метод возвращает true, если число чётное, и false, если число нечётное.

```
using System;

class Program
{
    static bool Odd(int n)
       {
                return n%2==0;
       }
    static void Main(string[] args)
    {
        int value = 100500;
        Console.WriteLine(Odd(value));
    }
}
```

Задача 5. Работа с консолью и перегрузкой методов.

Давайте поупражняемся с работой в консоли. Напишем для этого несколько функций, которые будут выводить данные в определённом месте консоли. А также создадим перегрузку метода, в котором можно будет задать цвет символов:

```
using System;

    class Program
    {

        static void Print(string msg, int x, int y)
        {
            // Установим позицию курсора на экране
            Console.SetCursorPosition(x, y);
            Console.WriteLine(msg);
        }

        static void Print(string msg, ConsoleColor foregroundcolor)
        {
```

```
            // Установим цвет символов
            Console.ForegroundColor = foregroundcolor;
            Console.WriteLine(msg);
        }


             // Создайте перегрузку функции, в которой будет указана позиция и
цвет
        static void Main()
        {
           Print("Привет!\n", 10,10);
           Print("Привет еще раз!", ConsoleColor.Red);
          Console.ReadLine();
        }
    }
```

Задача 6. Написать программу для подсчета площади треугольника.

Подсчёт площади и определение правильности треугольника сделаем в виде методов:

```
using System;

    class Program
    {
        static bool IsTriangle (double a,double b,double c)
        {
            return a + b > c && a + c > b && c + b > a;
        }

        static double S(double a,double b,double c)
        {
            double p=(a+b+c)/2;
            return Math.Sqrt(p * (p - a) * (p - b) * (p - c));
        }

        static void Main(string[] args)
        {
            Console.Write("Введите a:");
            double a = double.Parse(Console.ReadLine());
            Console.Write("Введите b:");
            double b = double.Parse(Console.ReadLine());
            Console.Write("Введите c:");
            double c = double.Parse(Console.ReadLine());
                  Console.WriteLine("Может существовать треугольник с такими
сторонами:" + IsTriangle (a, b, c));
            Console.WriteLine("Площадь треугольника:" + S(a, b, c));
      Console.ReadLine();
        }
    }
```

Домашнее задание

1. Написать программу «Анкета». Последовательно задаются вопросы (имя, фамилия, возраст, рост, вес). В результате вся информация выводится в одну строчку:

   а) используя склеивание;
   б) используя форматированный вывод;
   в) используя вывод со знаком $.

2. Ввести вес и рост человека. Рассчитать и вывести индекс массы тела (ИМТ) по формуле I=m/(h*h); где m — масса тела в килограммах, h — рост в метрах.

3. а) Написать программу, которая подсчитывает расстояние между точками с координатами x1, y1 и x2,y2 по формуле r=Math.Sqrt(Math.Pow(x2-x1,2)+Math.Pow(y2-y1,2). Вывести результат, используя спецификатор формата .2f (с двумя знаками после запятой);
   б) *Выполнить предыдущее задание, оформив вычисления расстояния между точками в виде метода.
4. Написать программу обмена значениями двух переменных:
   а) с использованием третьей переменной;
   б) *без использования третьей переменной.
5. а) Написать программу, которая выводит на экран ваше имя, фамилию и город проживания.
   б) *Сделать задание, только вывод организовать в центре экрана.
   в) **Сделать задание б с использованием собственных методов (например, Print(string ms, int x,int y).
6. *Создать класс с методами, которые могут пригодиться в вашей учебе (Print, Pause).

Достаточно решить 3 задачи. Записывайте в начало программы условие и свою фамилию. Все программы создавайте в одном решении. Со звездочками задания выполняйте в том случае, если вы решили задачи без звездочек.

Дополнительные материалы

1. Эффективное обучение C# разработчиков или Правильное программирование на C# с нуля.
2. Язык C# и .NET Framework.
3. Правила хорошего тона в программировании.
4. Индекс массы тела(BMI).

Используемая литература

Для подготовки данного методического пособия были использованы следующие ресурсы:

1. Павловская Т.А. Программирование на языке высокого уровня. — СПб: Питер, 2009.
2. Петцольд Ч. Программирование на C#. Т1. — М.: Русская редакция, 2001.
3. Шилдт Г. C# 4.0. Полное руководство”. — М.: ООО «И.Д. Вильямс», 2011.
4. MSDN.