Министерство образования Красноярского края

Краевое государственное бюджетное профессиональное образовательное учреждение

«Красноярский колледж радиоэлектроники и информационных технологий»

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТE**

|  |
| --- |
| 09.02.07 Информационные системы и программирование |
| код и наименование специальности |
| МДК 07.01 Управление и автоматизация баз данных |
| код и наименование профессионального модуля |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | АБД-2.19 |  |  |  | Ю.А Зубоненко |
|  | номер группы, зачетной книжки |  | подпись, дата |  | инициалы, фамилия |
|  |  |  | оценка |  |  |
| Руководитель от колледжа | |  |  |  | В.С Лисянский |
|  | |  | подпись, дата |  | инициалы, фамилия |

Красноярск 2021

**КУРС ПО РАЗРАБОТКЕ С#**

**Цель работы:** Пройти курс по разработке с# и приложить разработанные приложения в репозиторий.

Ссылка на курс: <https://github.com/nVinz/CSharp-Course>

Ссылка на репозиторий: <https://github.com/nVinz/CSharp-Course>

**Первая программа:**

Сгенерированный шаблон.

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
  
namespace FirstApp  
{

class Program

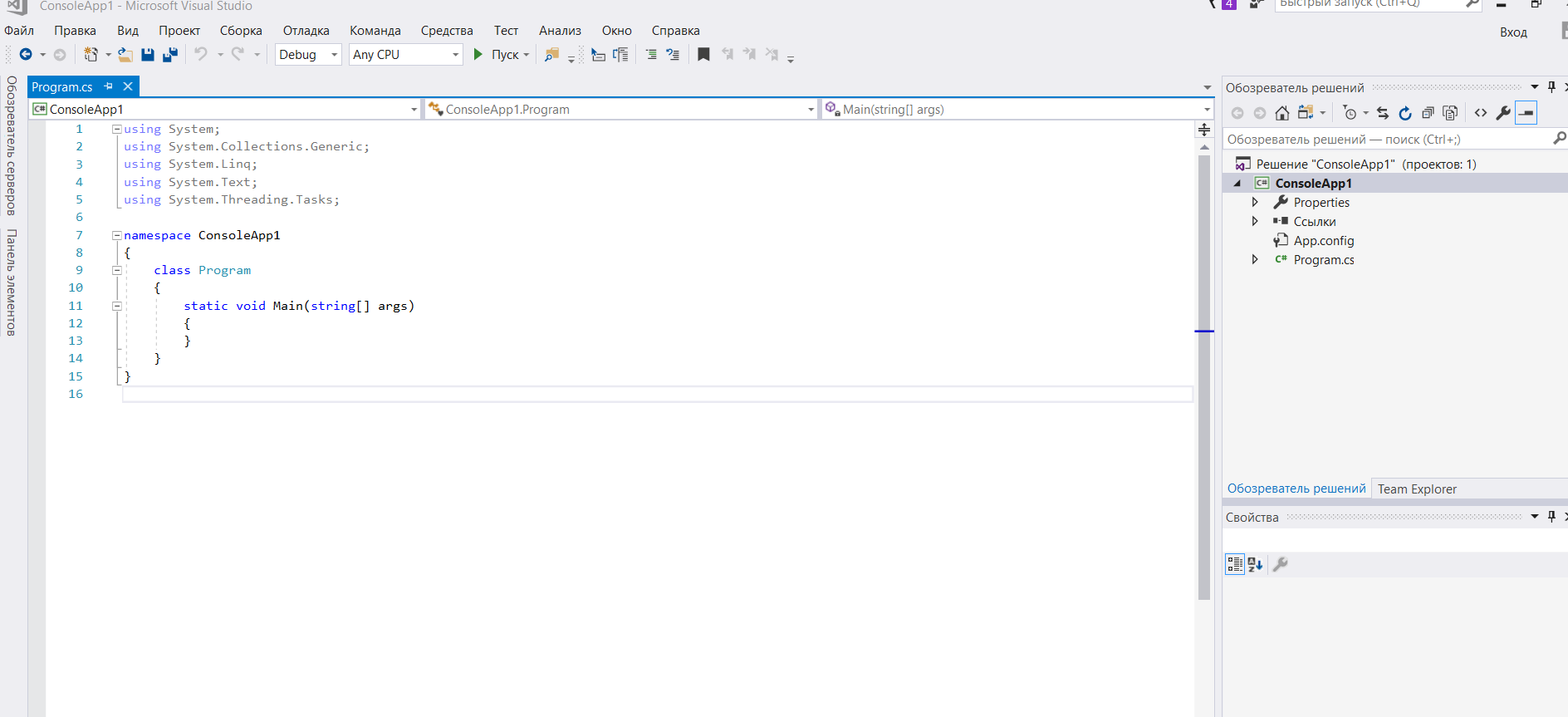
{

static void Main(string[] args)

{

}

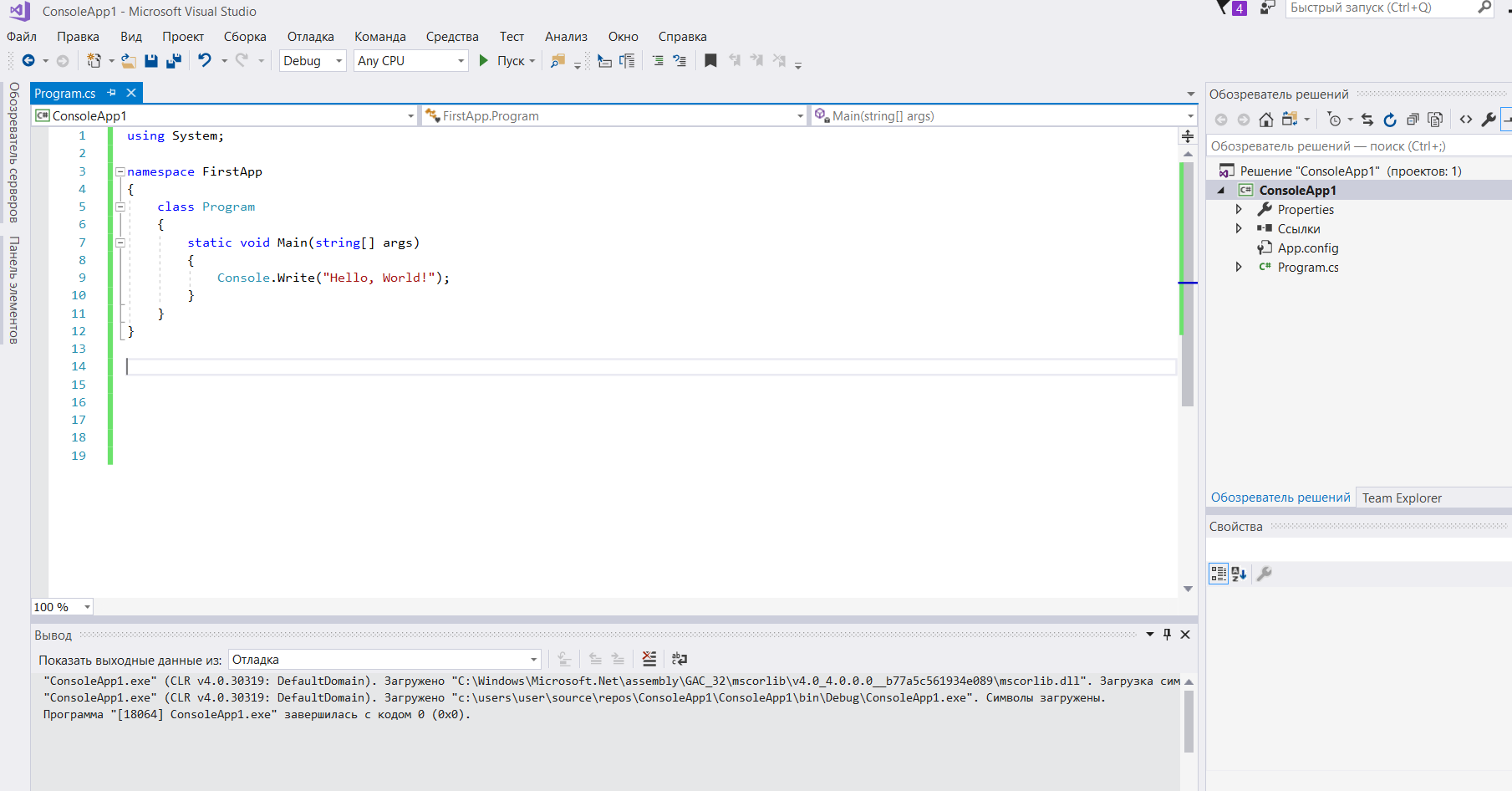
}  
}



Visual Studio сгенерировала следующий шаблон кода и это уже готовая программа, только она не делает ничего.

Hello, World!

using System;  
  
namespace FirstApp  
{  
    class Program  
    {  
        static void Main(string[] args)  
        {  
            Console.Write("Hello, World!");  
        }  
    }  
}



Она выводит на экран текст «**Hello, World!**».Если в коде есть какие то ошибки нам сообщают о не выполнении.

Все символы загружены и код прекрасно работает.

Вывод строк

Первый пример использует уже известный нам **Console.Write**. Данная программа выведет на одной строке склеенныестроки, то есть:

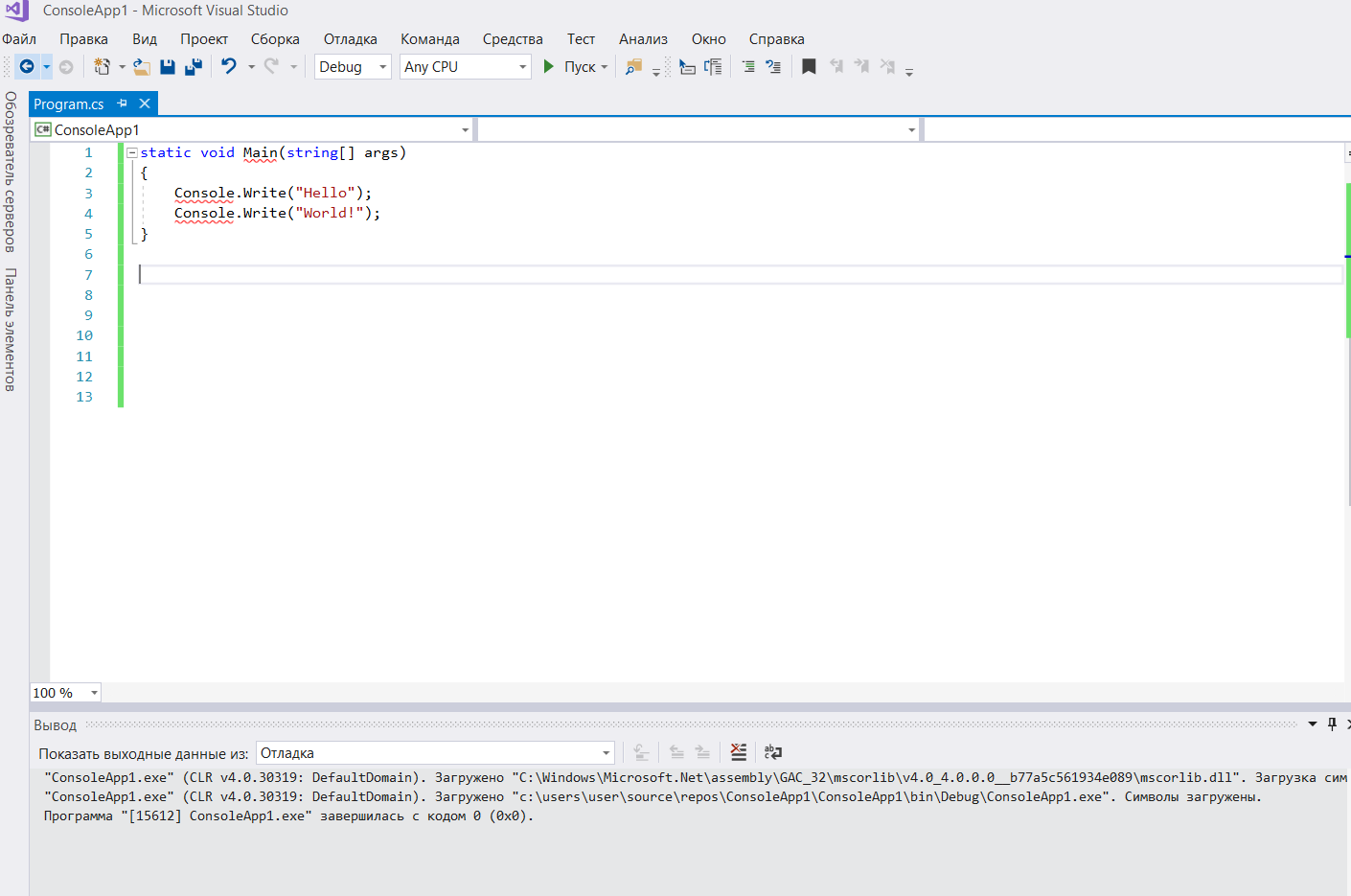
HelloWorld!

using System;

static void Main(string[] args)  
{  
   Console.Write("Hello");

  Console.Write("World!");

}

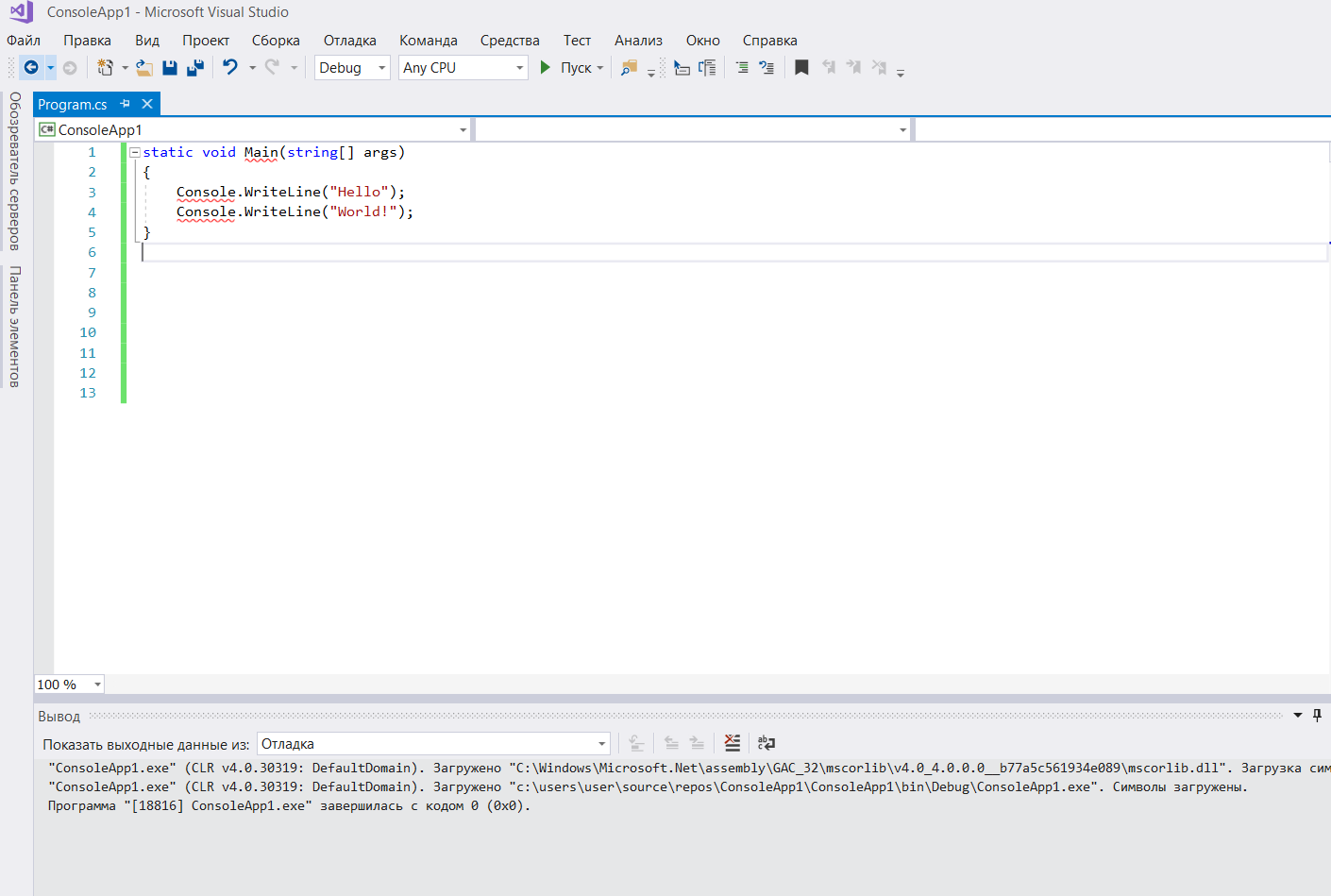


Команда успешно запустилась.

static void Main(string[] args)  
{

   Console.WriteLine("Hello");

  Console.WriteLine("World!");  
}



Вторая программа использует **Console.WriteLine()**. Данная команда после вывода переводит курсор на следующую строку, то есть, вывод будет следующим:

Hello

World!

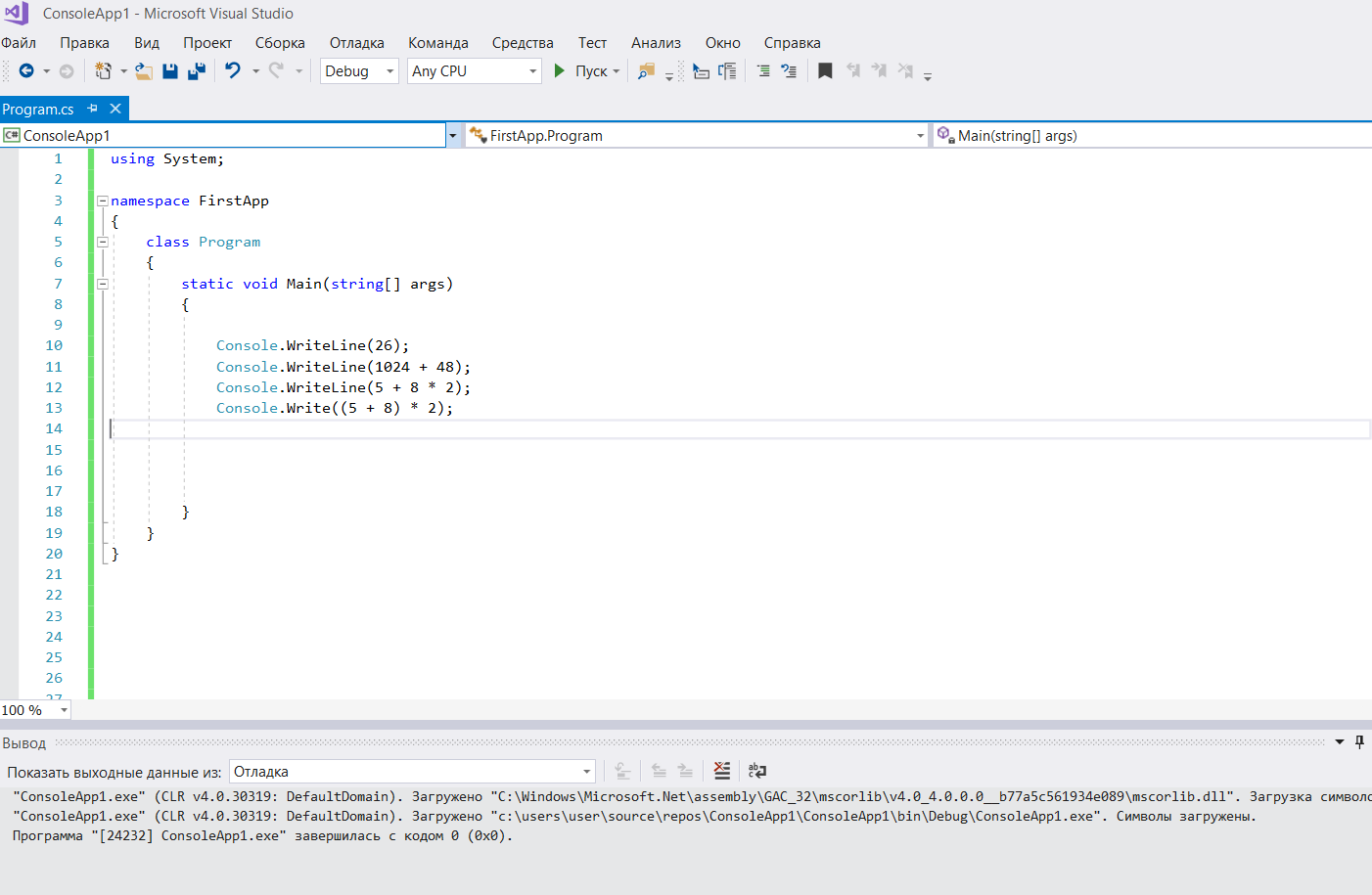
Команда успешно запустилась.

Вывод данных

static void Main(string[] args)

{  
  Console.WriteLine(26);   
  Console.WriteLine(1024 + 48);  
  Console.WriteLine(5 + 8 \* 2);  
  Console.Write((5 + 8) \* 2);

}



Операция над числами производится как в математике. На каждой строке выводится результат.

Форматированный вывод

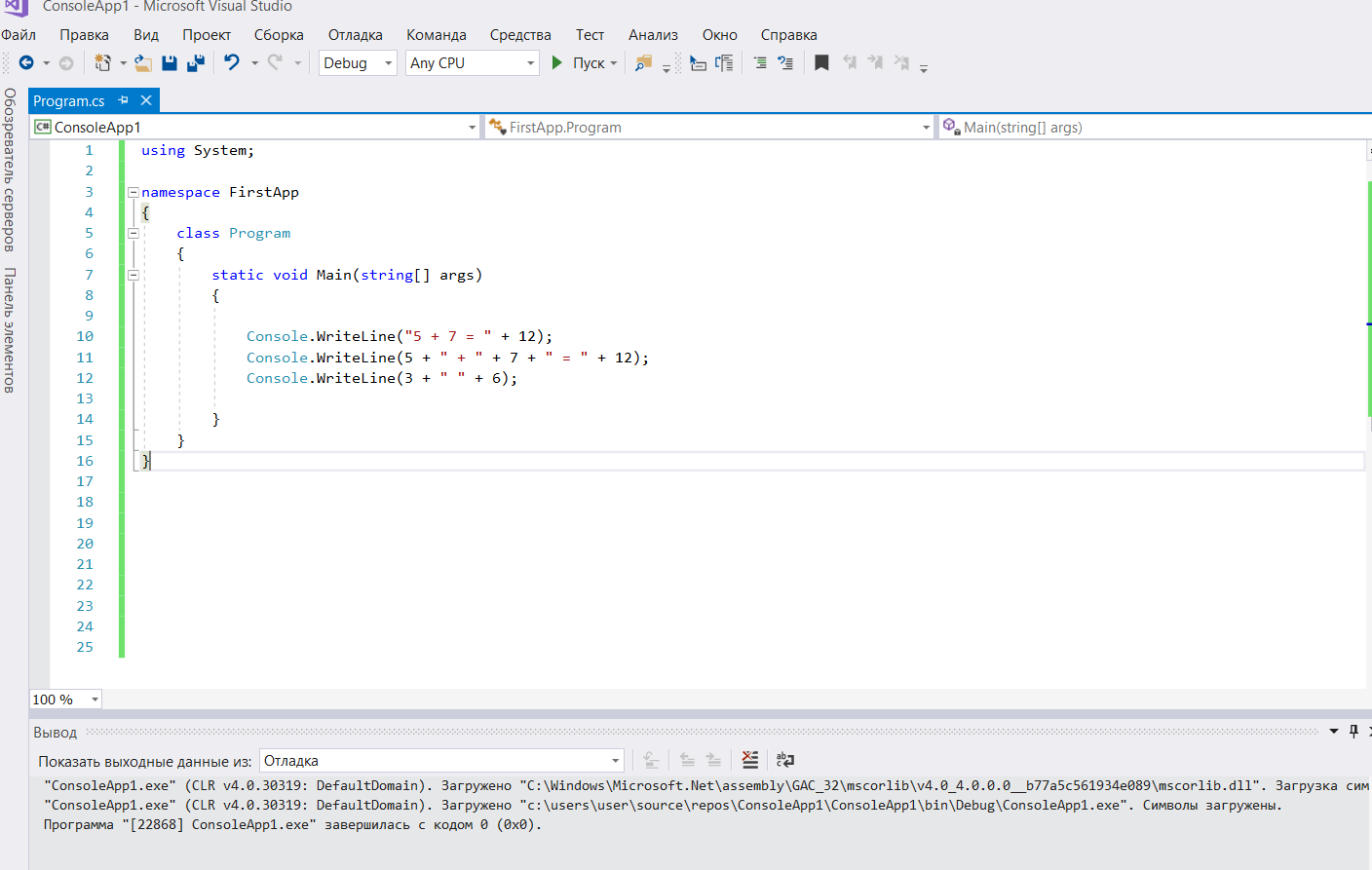
static void Main(string[] args)

{   
  Console.WriteLine("5 + 7 = " + 12);

  Console.WriteLine(5 + " + " + 7 + " = " + 12);   
  Console.WriteLine(3 + " " + 6);

}

Пример показывает, что один и тот же результат вывода можно достичь разными способами.



Переменные

Для хранения информации в программировании используются переменные.

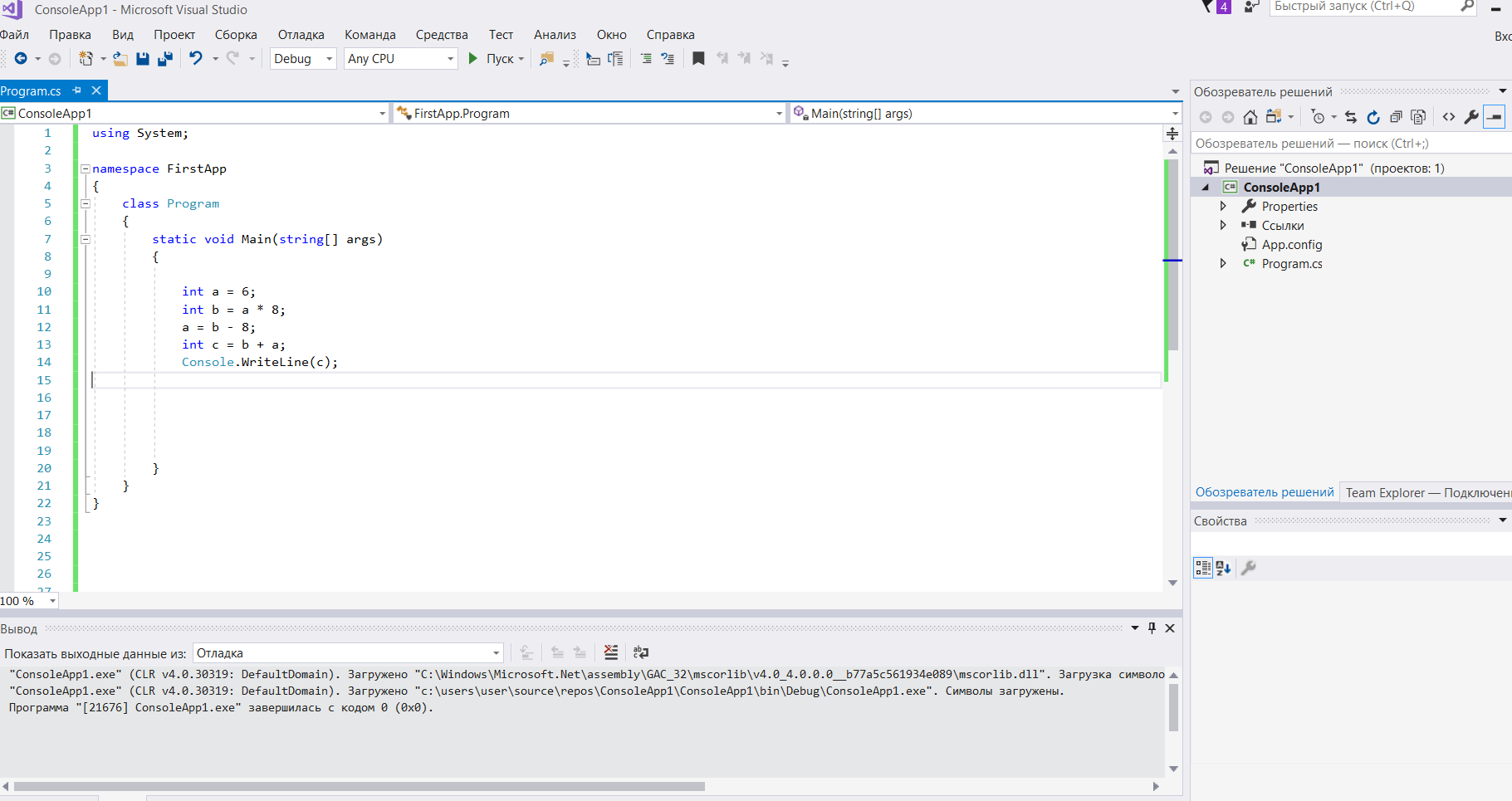
**Переменная** - это ячейка памяти определенного **типа**, имеющая **имя**.

Типы данных

Основные типы данных:

* **int** - целое число;
* **string** – строка;
* **double** - число с дробной частью;
* **char** – символ;
* **bool** - специальный тип, принимает только два значения - **true** или **false.**

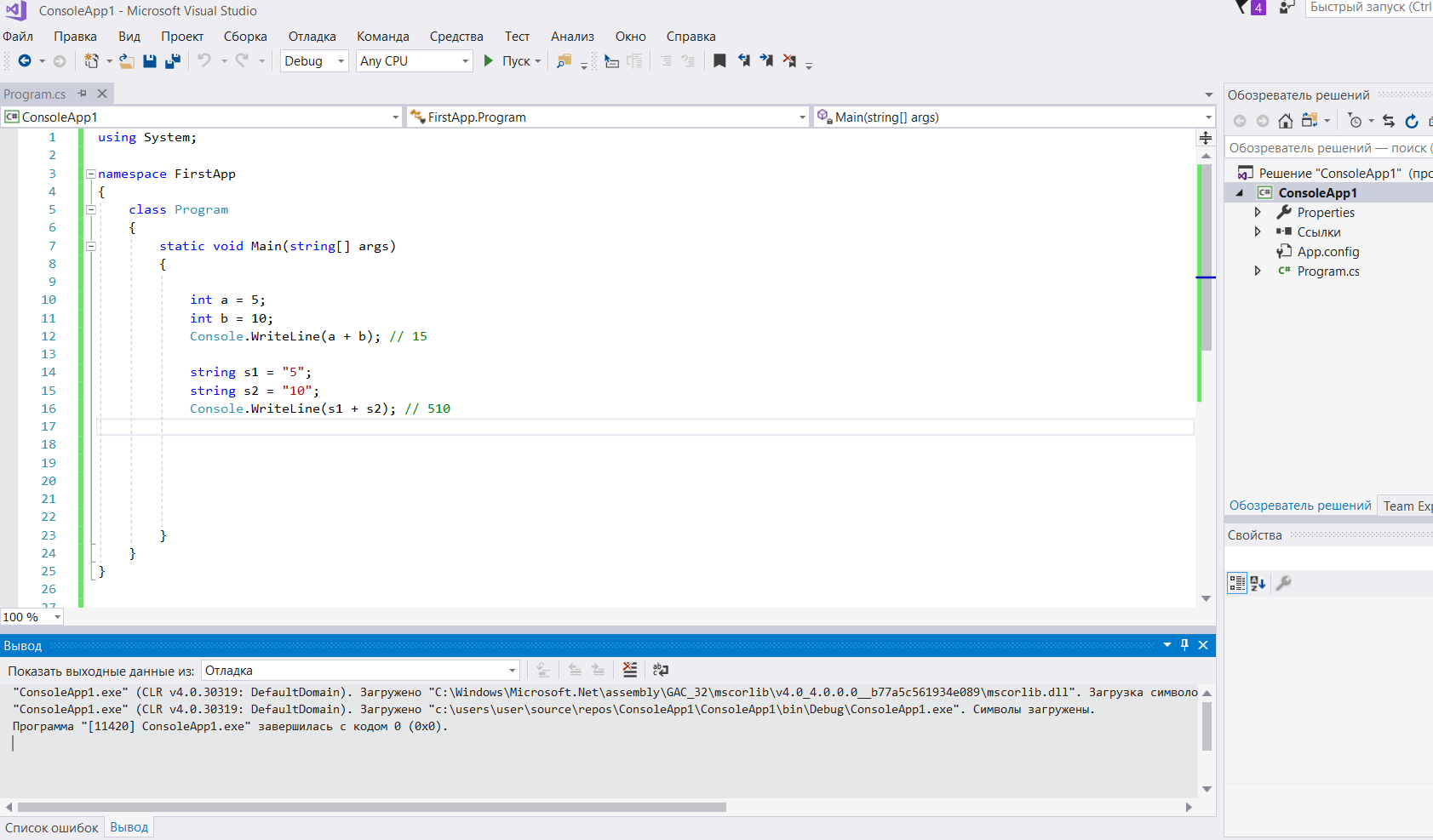
int a = 6;  
int b = a \* 8;  
a = b - 8;   
int c = b + a;   
Console.WriteLine(c);



От типа данных зависит, какая информация будет хранится в переменной. Также от типа переменной зависит поведение операций над ними.

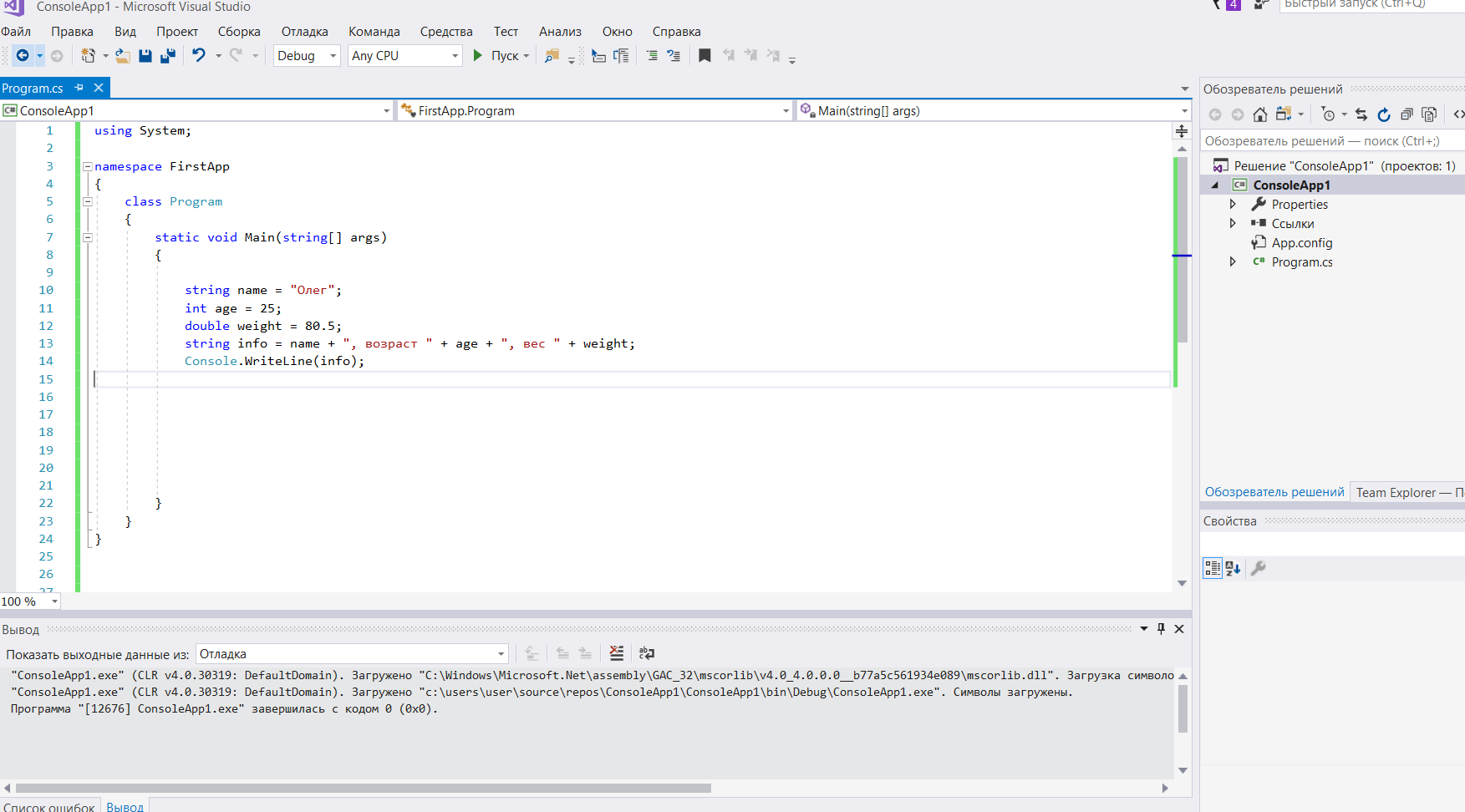
Для строк знак плюс **(+)** означает склеивание между собой, а для целых и вещественных чисел, плюс складывает их и возвращает результат:

int a = 5;  
int b = 10;  
Console.WriteLine(a + b); // 15  
  
string s1 = "5";  
string s2 = "10";  
Console.WriteLine(s1 + s2); // 510



Еще пример:

string name = "Олег";  
int age = 25;  
double weight = 80.5;  
string info = name + ", возраст " + age + ", вес " + weight;  
Console.WriteLine(info);



Ввод информации. Конвертация

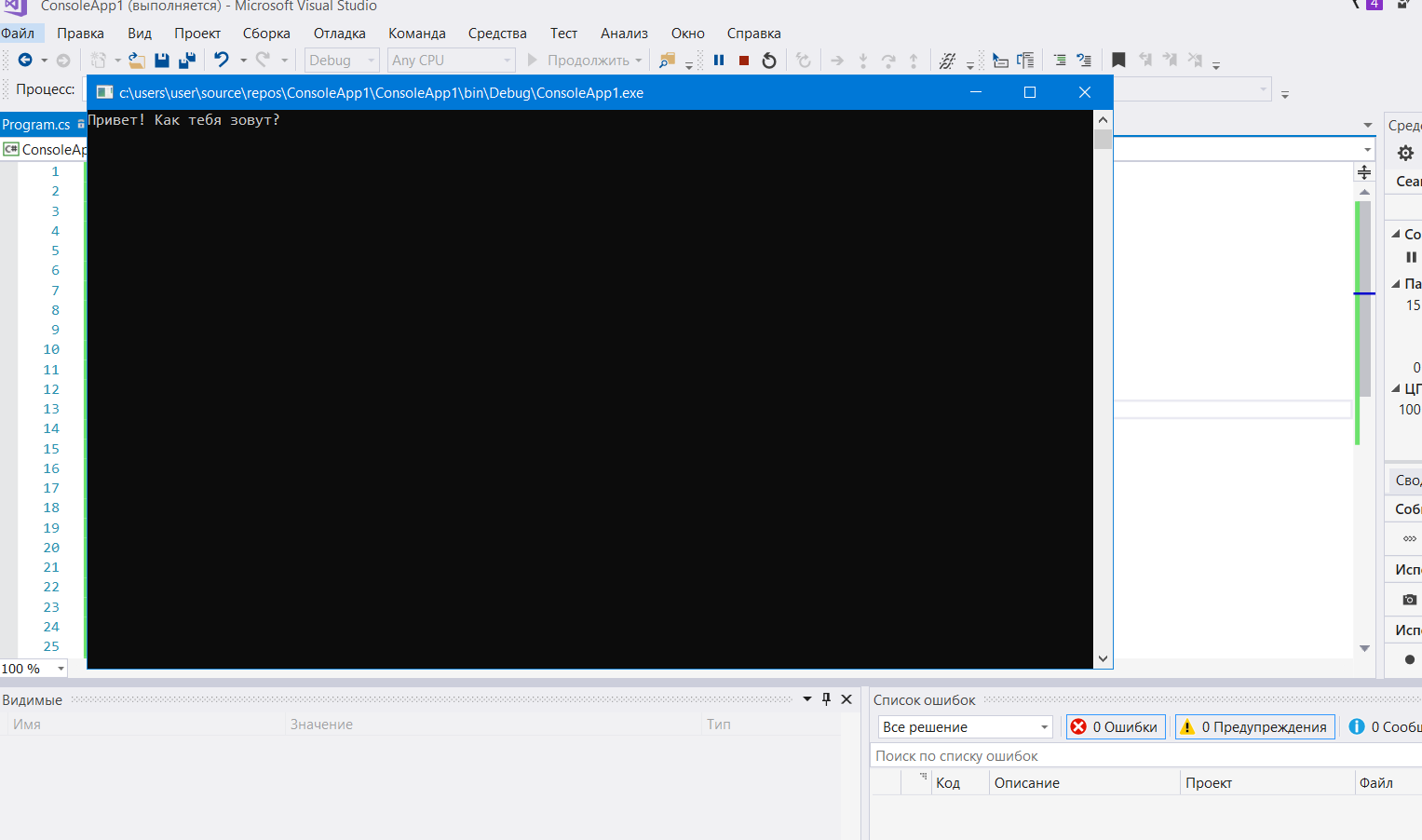
Присваивание переменной введенного числа

Напишем программу, которая будет спрашивать имя у пользователя и после здороваться с ним:

Console.WriteLine("Привет! Как тебя зовут?");

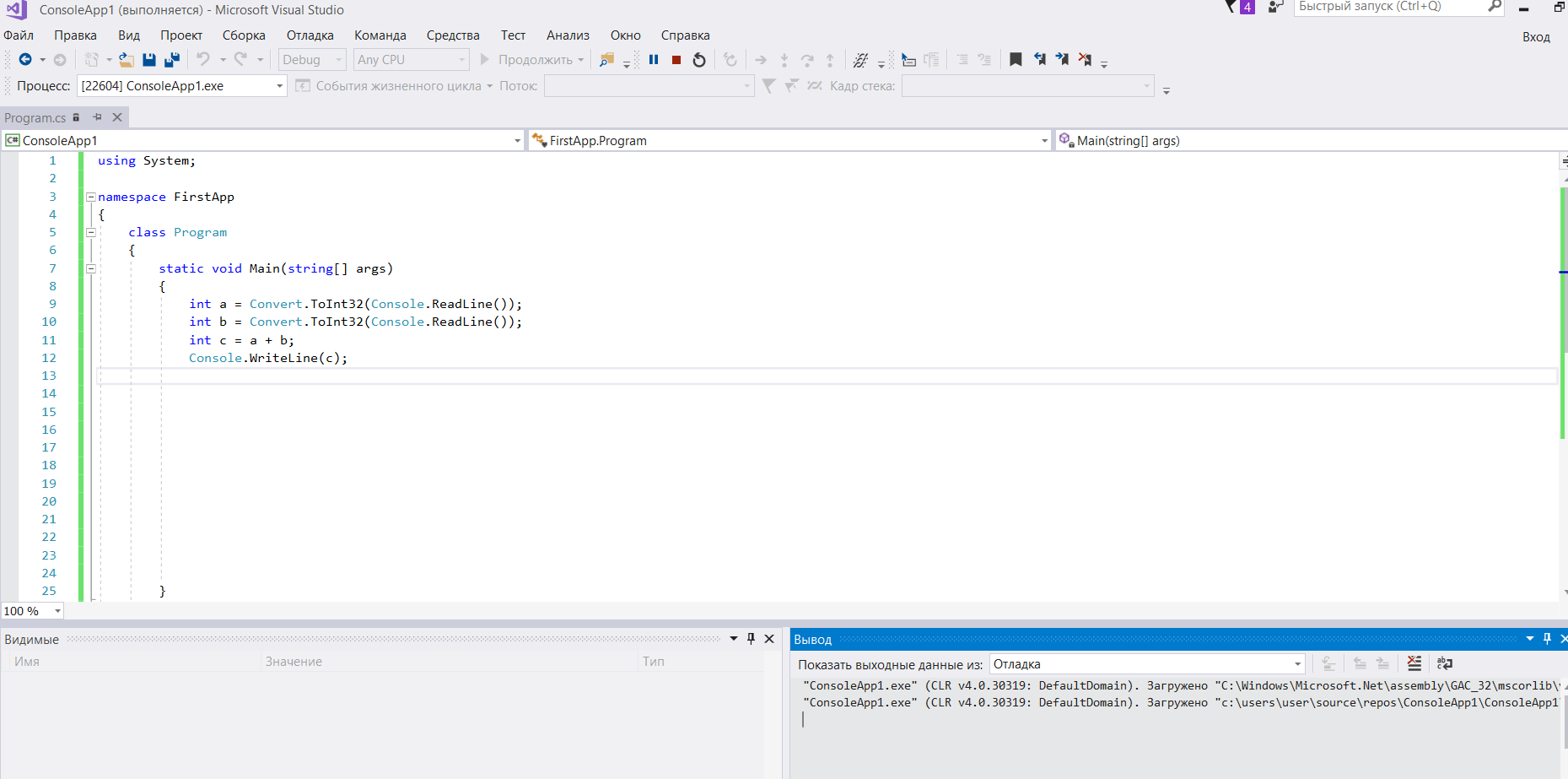
string name = Console.ReadLine();

Console.WriteLine("Привет, " + name);



Для лучшего понимания напишем программу, которая принимает два числа и выводит их сумму:

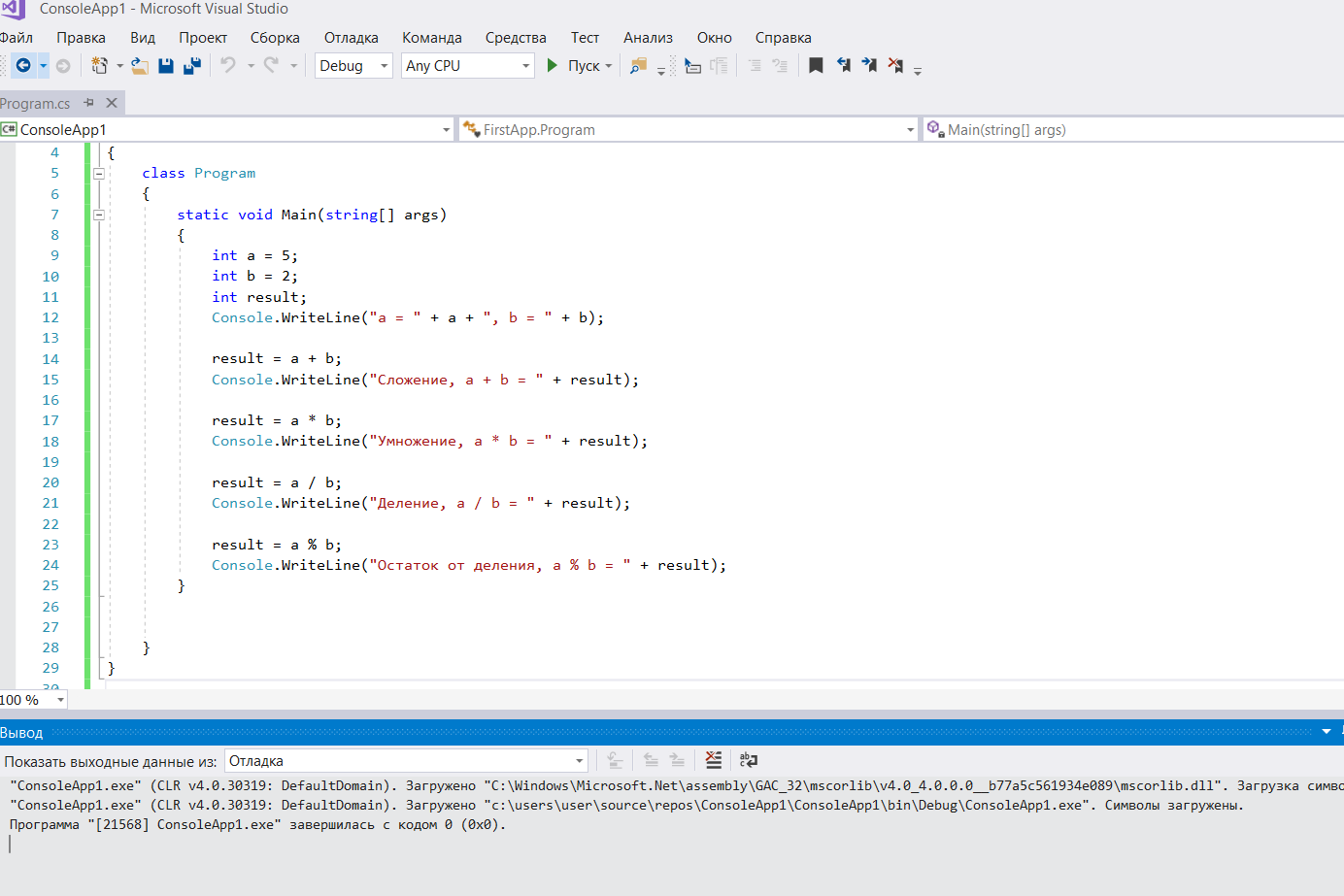
int a = Convert.ToInt32(Console.ReadLine());  
int b = Convert.ToInt32(Console.ReadLine());  
int c = a + b;   
Console.WriteLine(c);



**ОПЕРАЦИИ С ЧИСЛАМИ**

int a = 5;  
    int b = 2;

int result;  
    Console.WriteLine("a = " + a + ", b = " + b);  
   
    result = a + b;  
    Console.WriteLine("Сложение, a + b = " + result);  
   
    result = a \* b;  
    Console.WriteLine("Умножение, a \* b = " + result);  
   
    result = a / b;  
    Console.WriteLine("Деление, a / b = " + result);  
   
    result = a % b;  
    Console.WriteLine("Остаток от деления, a % b = " + result);  
}



Унарные операторы

Унарных арифметических операторов есть всего два:

* **инкрементация** «++»;
* **декрементация** «--».

Инкрементация **увеличивает** операнд на **единицу**, а декрементация - **уменьшает** на **единицу**.

static void Main()

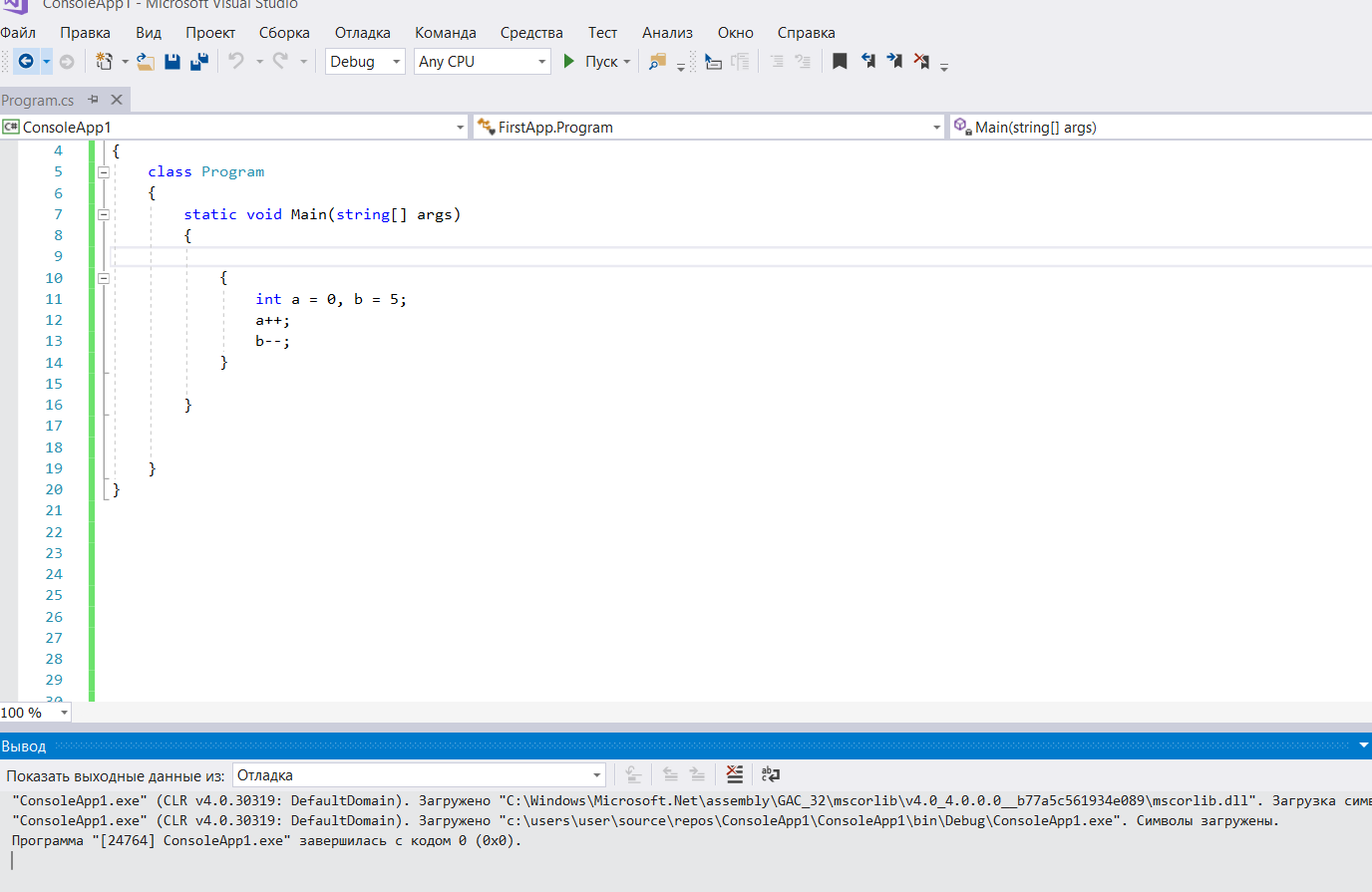
{

int a = 0, b = 5;

a++;

b--;

}



**ВЕЩЕСТВЕННЫЕ ЧИСЛА**

Данные вещественного (**double**) типа используются значительно чаще целого типа. Они необходимы в тех случаях, когда числовые значения могут содержать дробные части. Дробная часть отделяется от целой части при при помощи **точки**.

double d = 0.5;

Считываются вещественные переменные точно также как целые, только конвертация происходит в нужный тип данных, то есть:

double d = Convert.ToDouble(Console.ReadLine());

Чтобы получить **точный** результат с десятичной точкой, нужно, чтобы делимое и/или делитель были типа **double**. Тогда деление будет происходить как в математике:

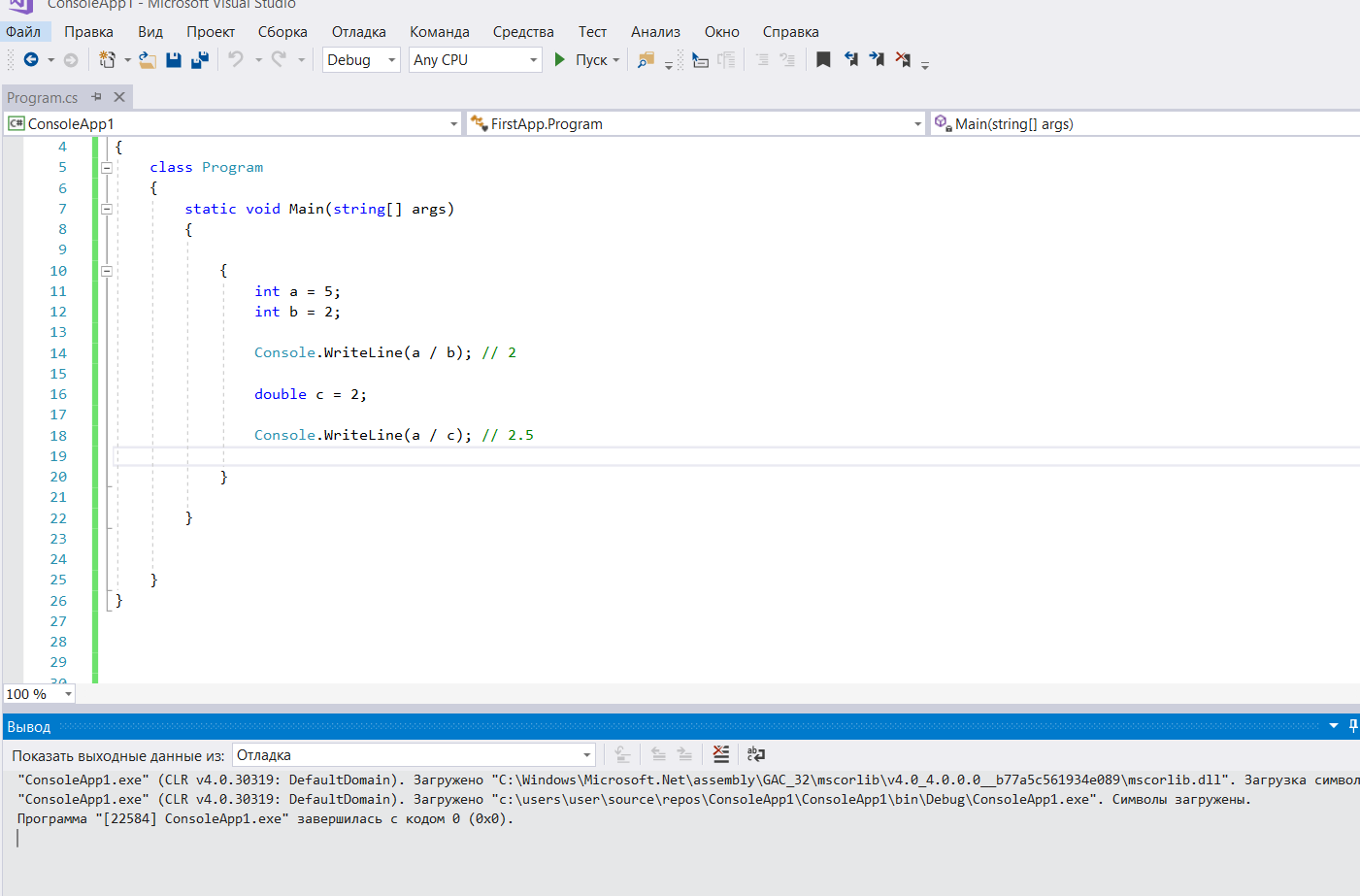
int a = 5;

int b = 2;

Console.WriteLine(a / b); // 2

double c = 2;

Console.WriteLine(a / c); // 2.5



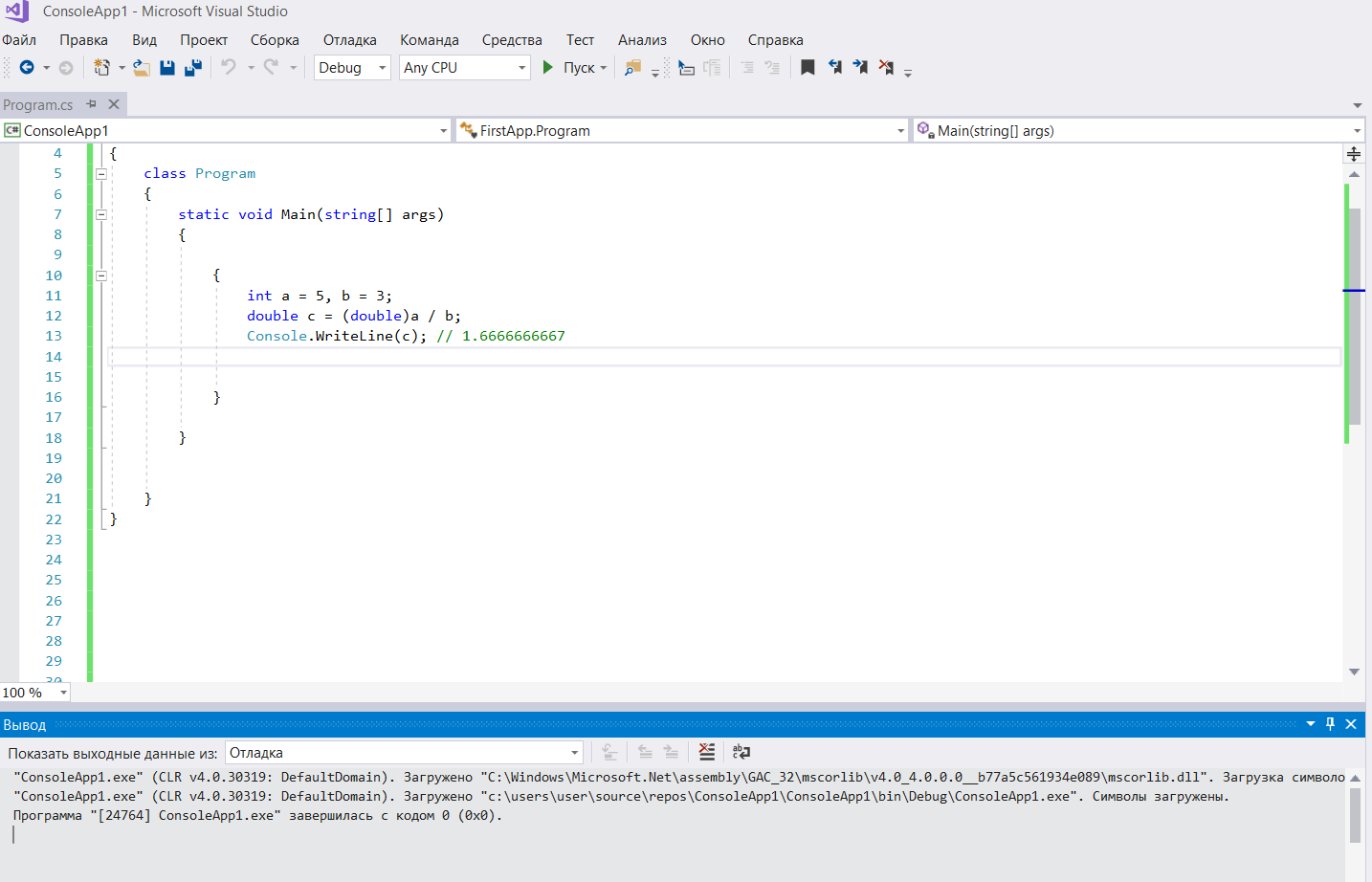
Целые и действительные числа в одном арифметическом выражении

Если мы хотим сохранить в вещественной переменной целое число, то достаточноприсвоить переменной типа **double** значение целочисленного арифметического выражения. Это возможно, так как множество вещественных чисел содержит в себе множество целых чисел:

int a = 8;  
double c = a;

Таким образом, чтобы в переменной "с" оказался правильный результат, нужно написать:

int a = 5, b = 3;  
double c = (double)a / b;  
Console.WriteLine(c); // 1.6666666667



Явное приведение типов

Мы не всегда можем присвоить значение переменной другой переменной, если они разных типов. Для этого нам необходимо **явно привести** значение одной переменной в значенгие второй переменной:

int a = 5;

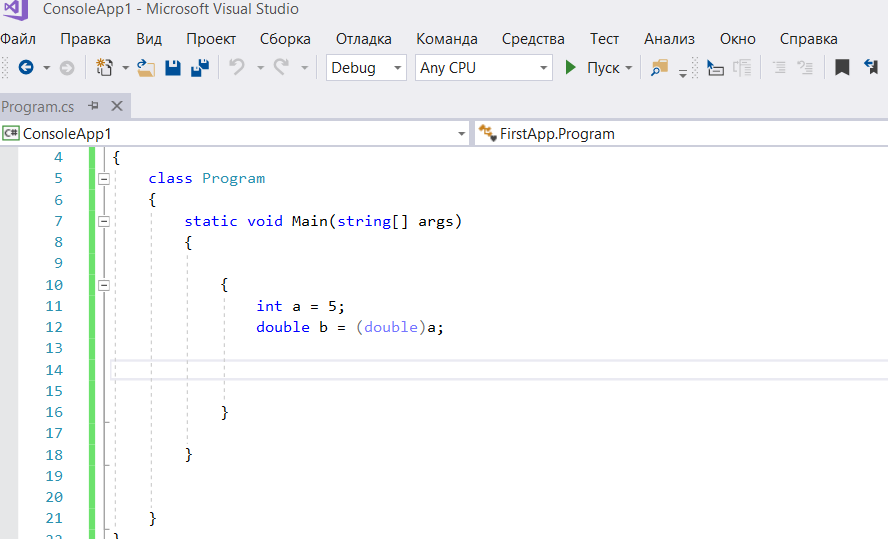
double b = (double)a;

int a = 5;

string b = (string)a;

double a = 5;

string b = (string)a;



Класс Math

В классе **Math** собраны все основные математические функции: модуль числа, функция возведение числа в степень, нахождение квадратного корня и другие.

Для **нахождения модуля числа**, используется функция **Abs**([число]):

static void Main(string[] args)

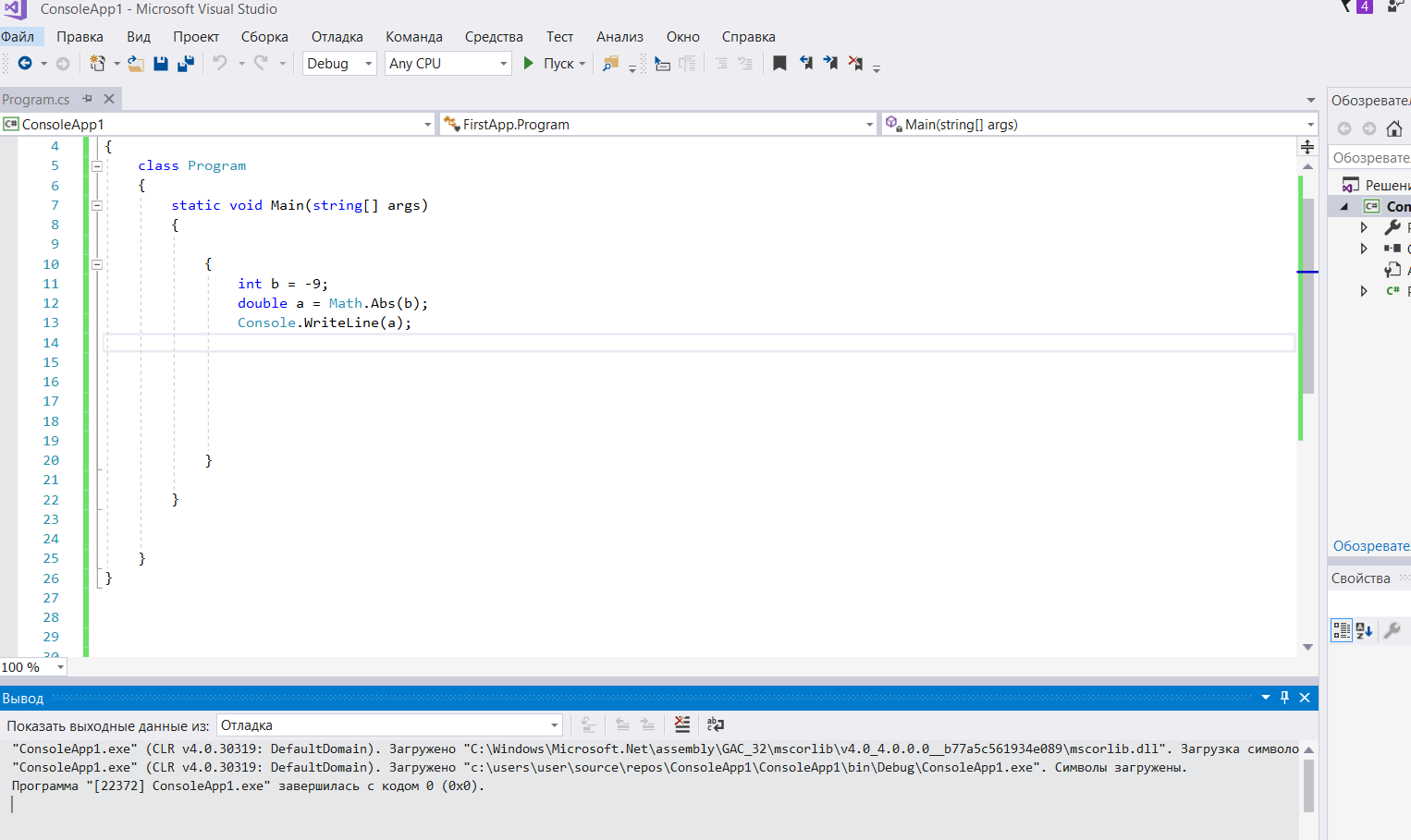
{

int b = -9;

double a = Math.Abs(b);

Console.WriteLine(a);

}



Для **возведения числа в степень**, используется функция **Pow**([число], [степень]):

static void Main(string[] args)

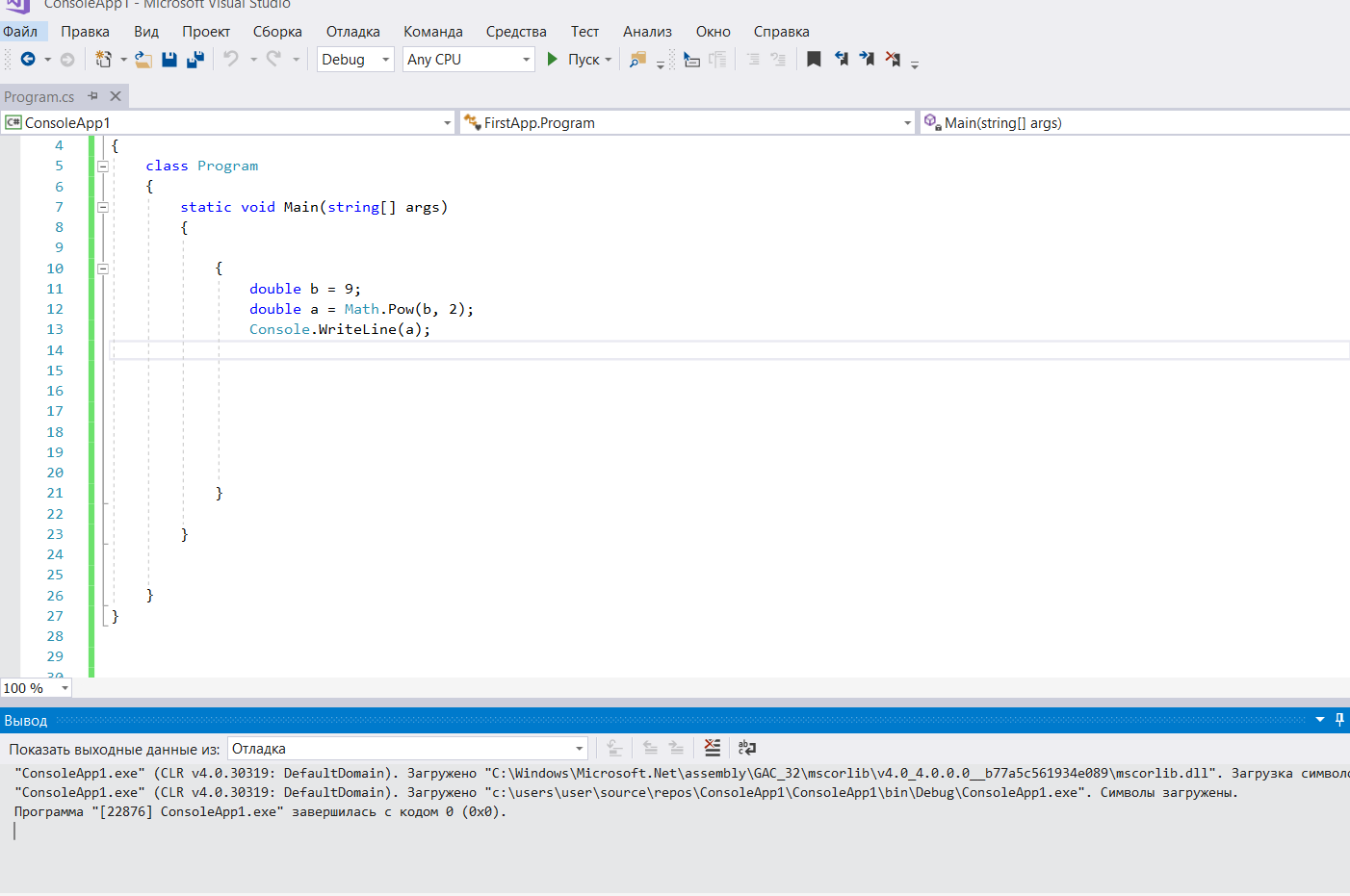
{

   double b = 9;

   double a = Math.Pow(b, 2);

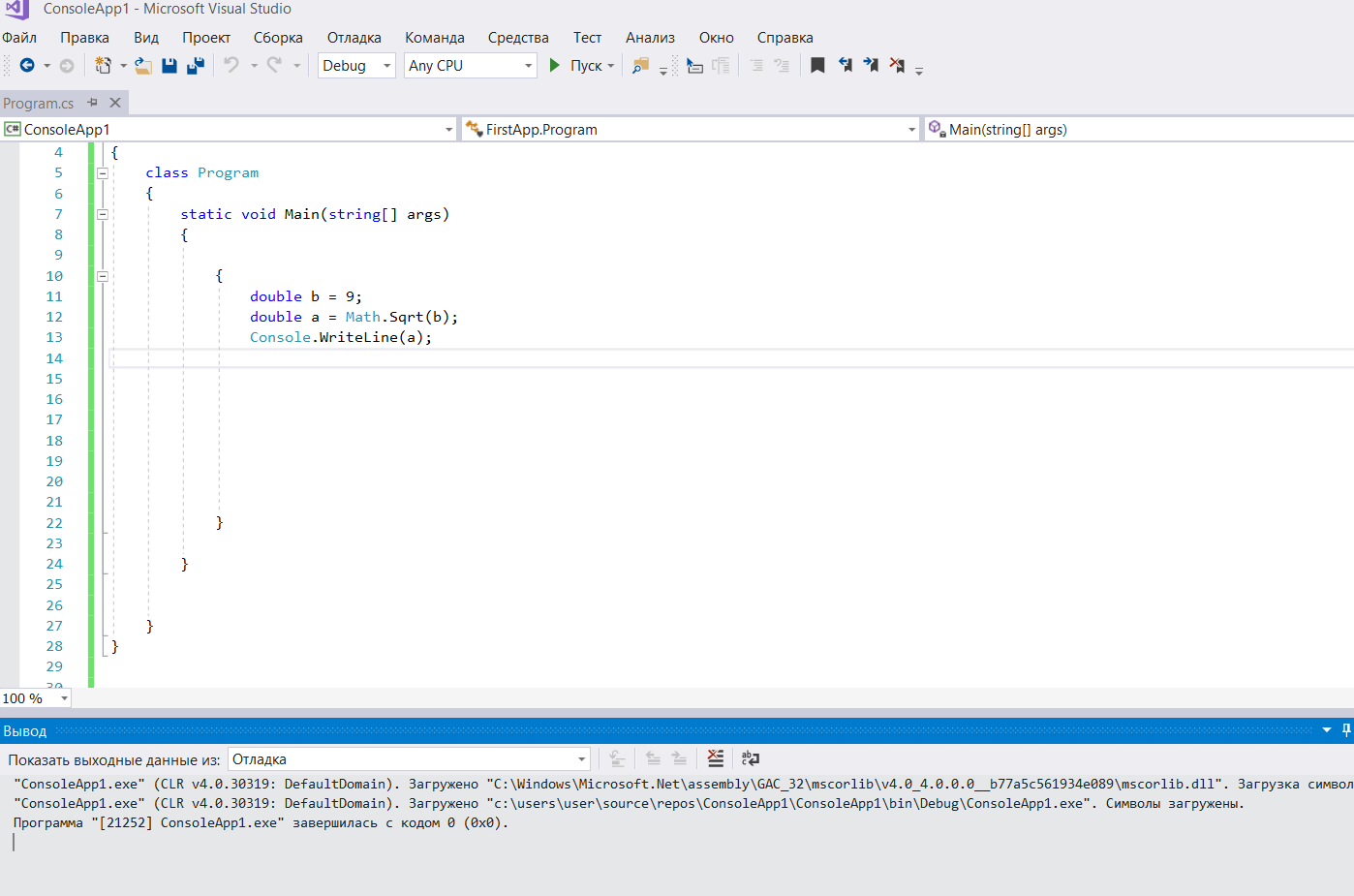
   Console.WriteLine(a);

}

  
  Для **нахождения квадратного корня** служит функция **Sqrt**([число]) возвращаемый тип данных – **double**:

static void Main(string[] args)  
{  
   double b = 9;

   double a = Math.Sqrt(b);  
   Console.WriteLine(a);   
}



Условный оператор

В общем виде **оператор** **if** записывается так:

if (выражение)  
{  
   инструкция;

   инструкция;

   инструкция;

}

Пример:

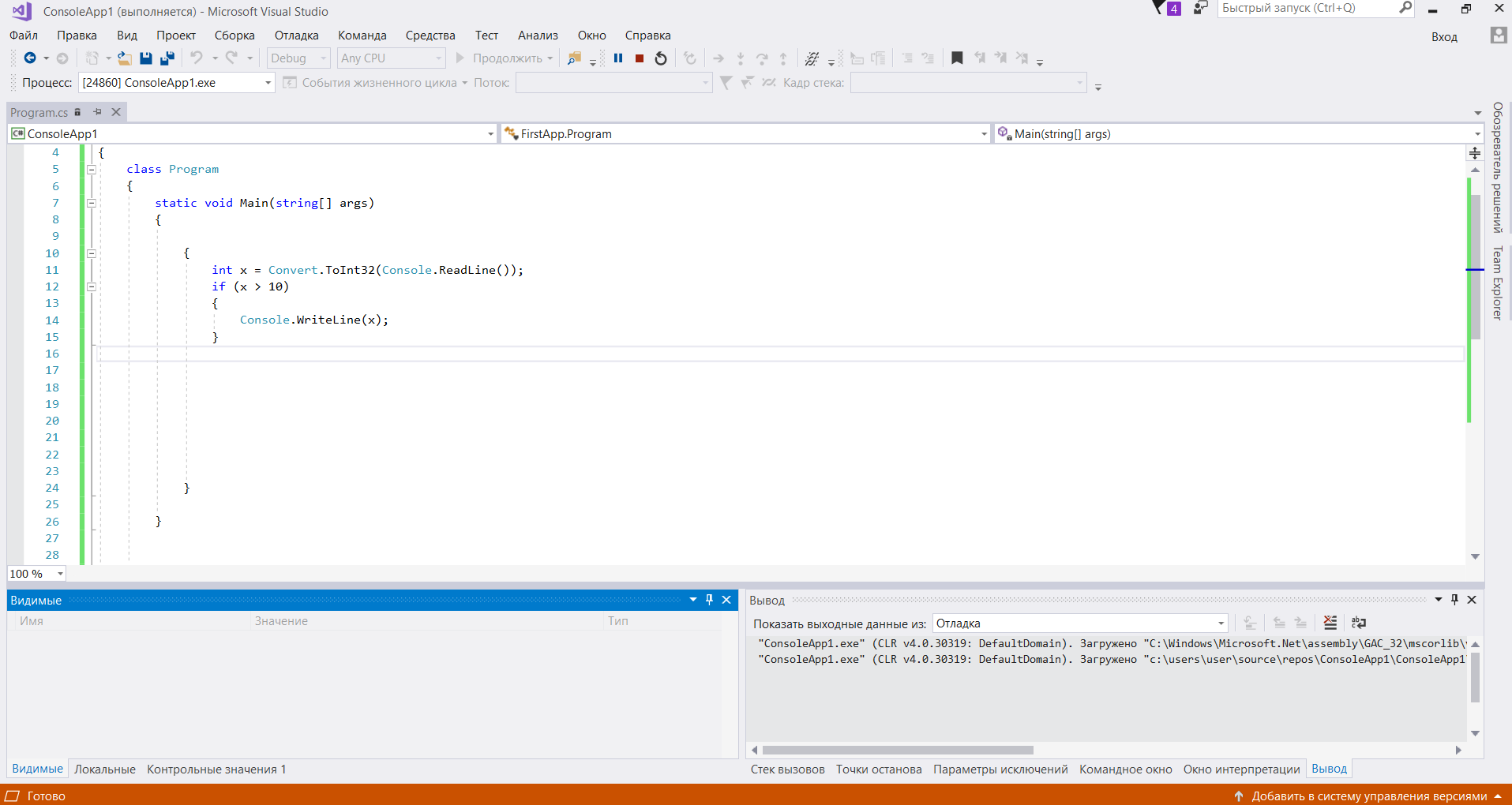
int x = Convert.ToInt32(Console.ReadLine());

if (x > 10)

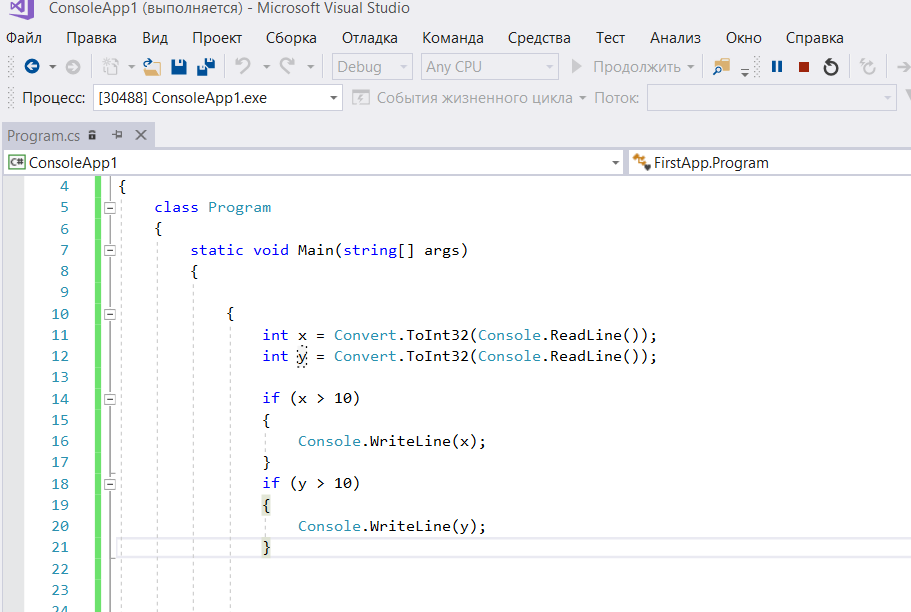
{

   Console.WriteLine(x);

}



Операторы **if** могут идти последовательно, проверяя несколько условий подряд:

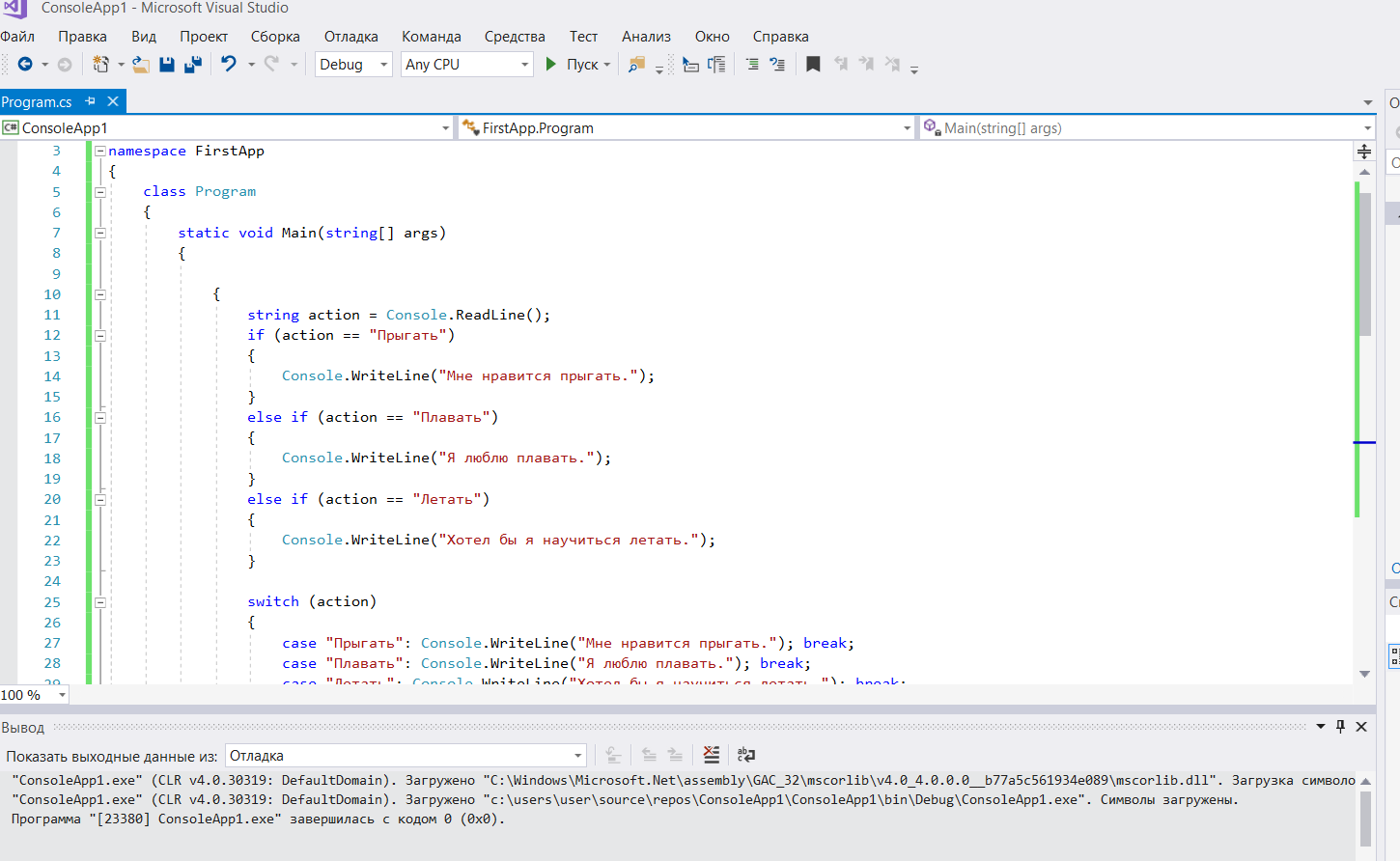


Оператор switch

Оператор **switch** сравнивает значение условного выражения с несколькими значениями.

Представим себе переменную **action**, которая может иметь значения **"Прыгать"**, **"Плавать"** и **"летать"**. Оператор **switch** позволяет легко определить блок кода, который должен исполняться для каждого из этих значений.

  В общем виде конструкция **switch-case** записывается так:

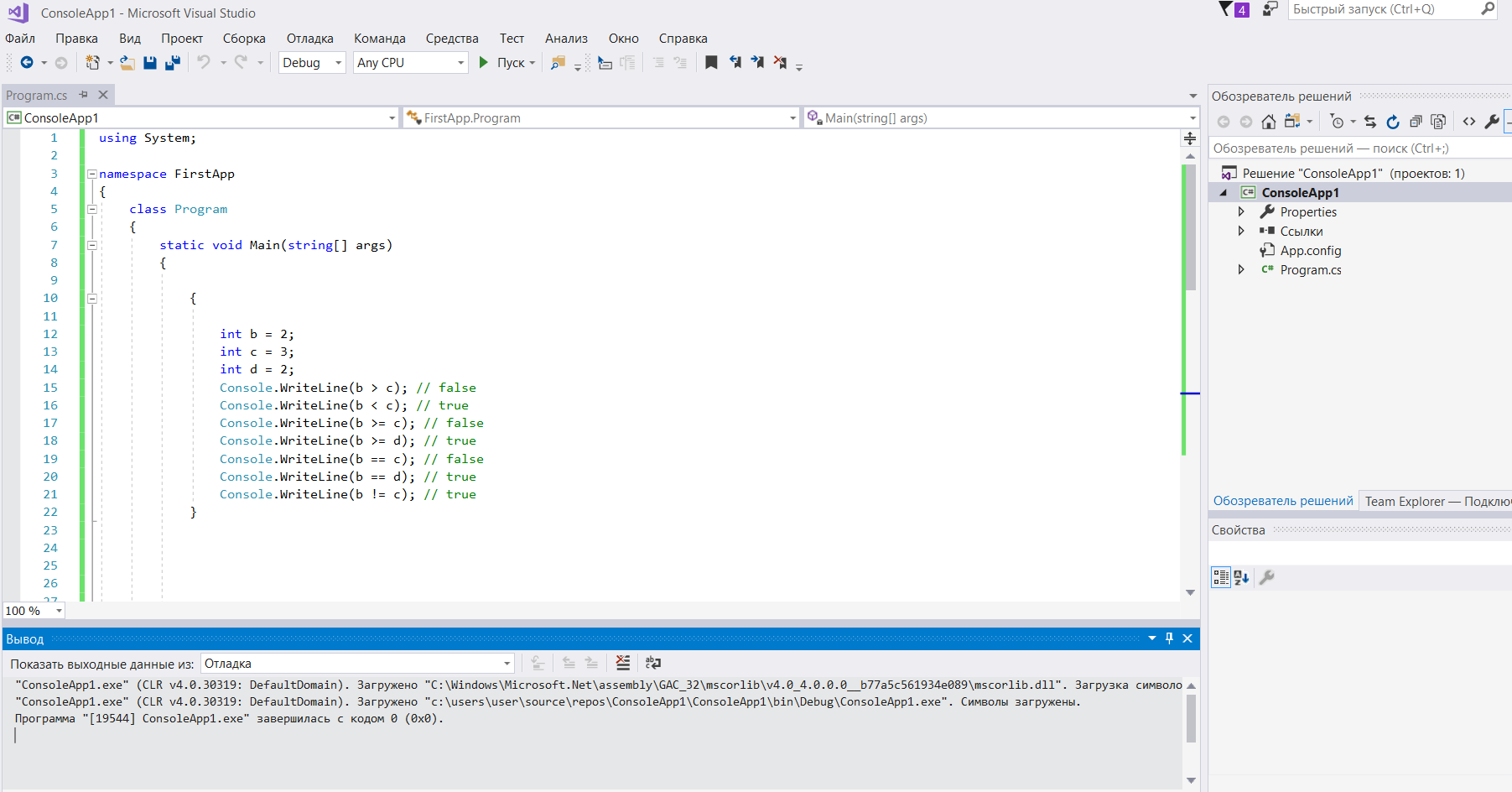


Операторы сравнения

В условных выражениях нужно пользоваться операторами сравнения:

* **>, <** - больше, меньше
* **>=, <=** - больше либо равно, меньше либо равно
* **==, !=** - равно, не равно

Стоит отметить, что "**=**" - оператор **присвоения**, а два раза равно "**==**" - это оператор **сравнения**, которое сравнивает на равенство два объекта. Для лучшего понимания рассмотрим пример:



цикл for

Допустим нам необходимо написать программу, которая выводит 100 раз строку "**Привет**!". Для решения данной задачи, мы напишем примерно следующий код:

Цикл for имеет следующую структуру:

for (инициализация счетчика; условие продолжения; итерация)

{

блок кода, который будет повторяться

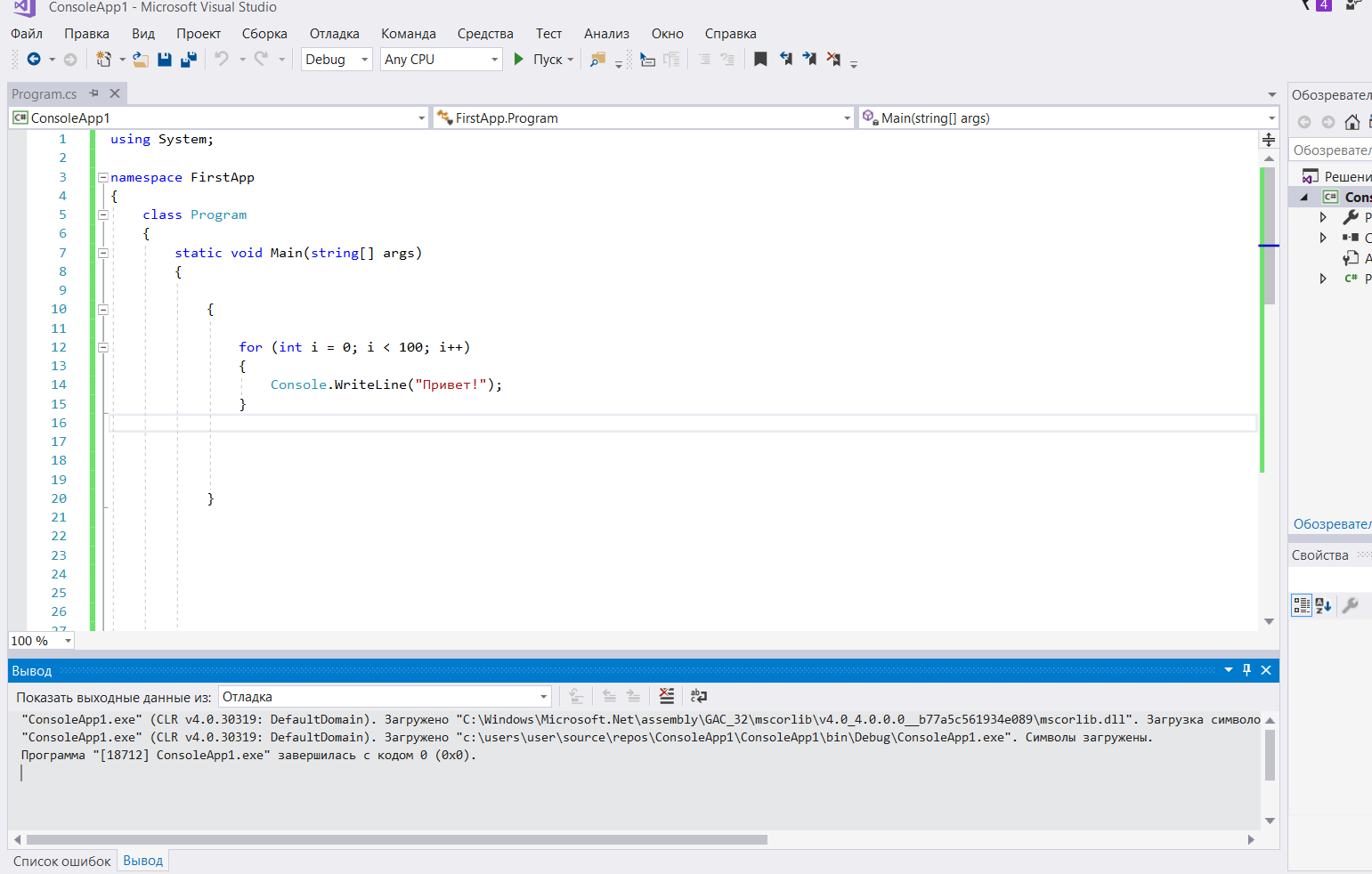
}

Пример:

for (int i = 0; i < 100; i++)

{

Console.WriteLine("Привет!");  
}



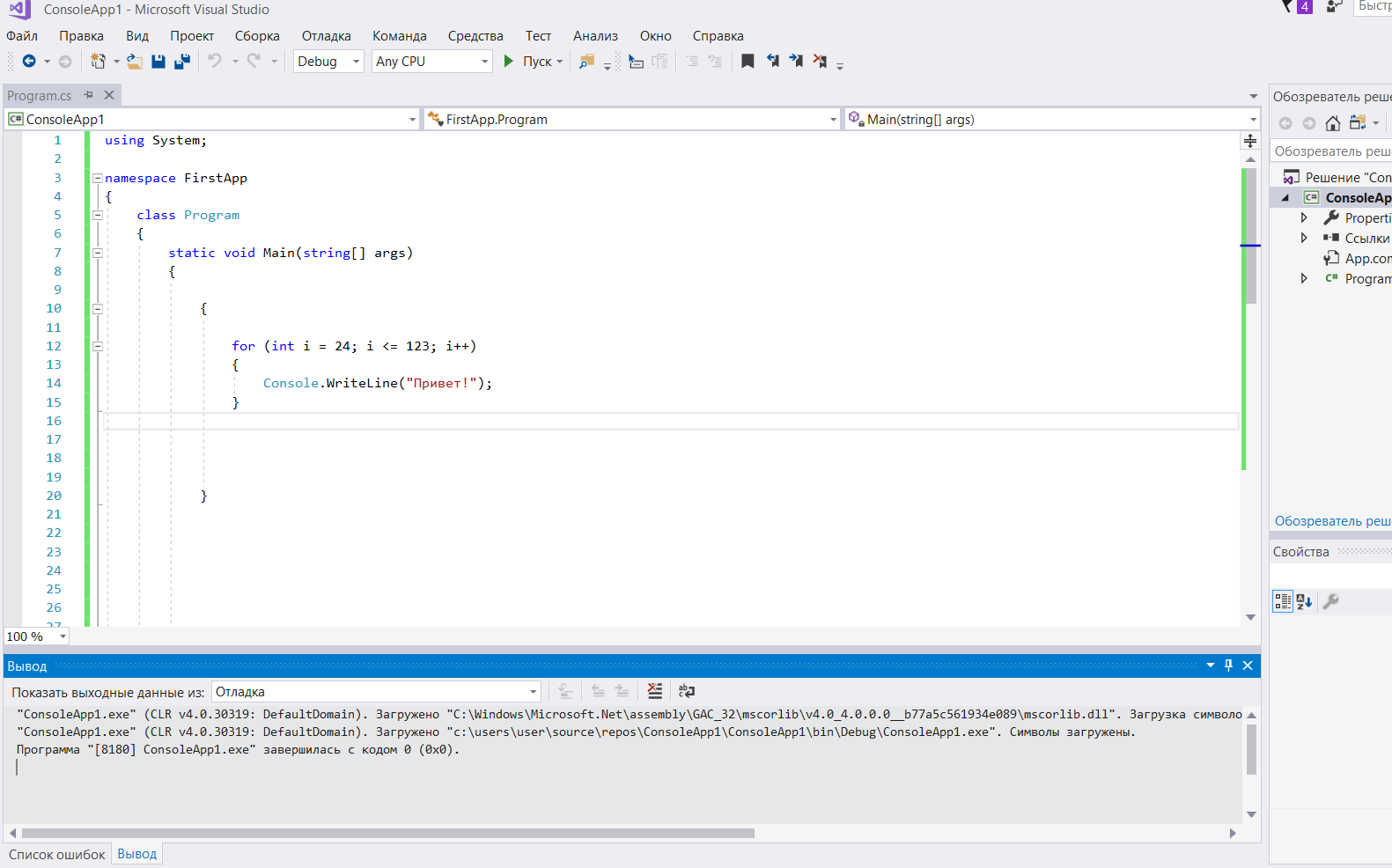
Итерация

for (int i = 24; i <= 123; i++)

{

  Console.WriteLine("Привет!");  
}

**max** = 123, **min** = 24, следовательно количество итераций равно 123 - 24 + 1  = **100**.



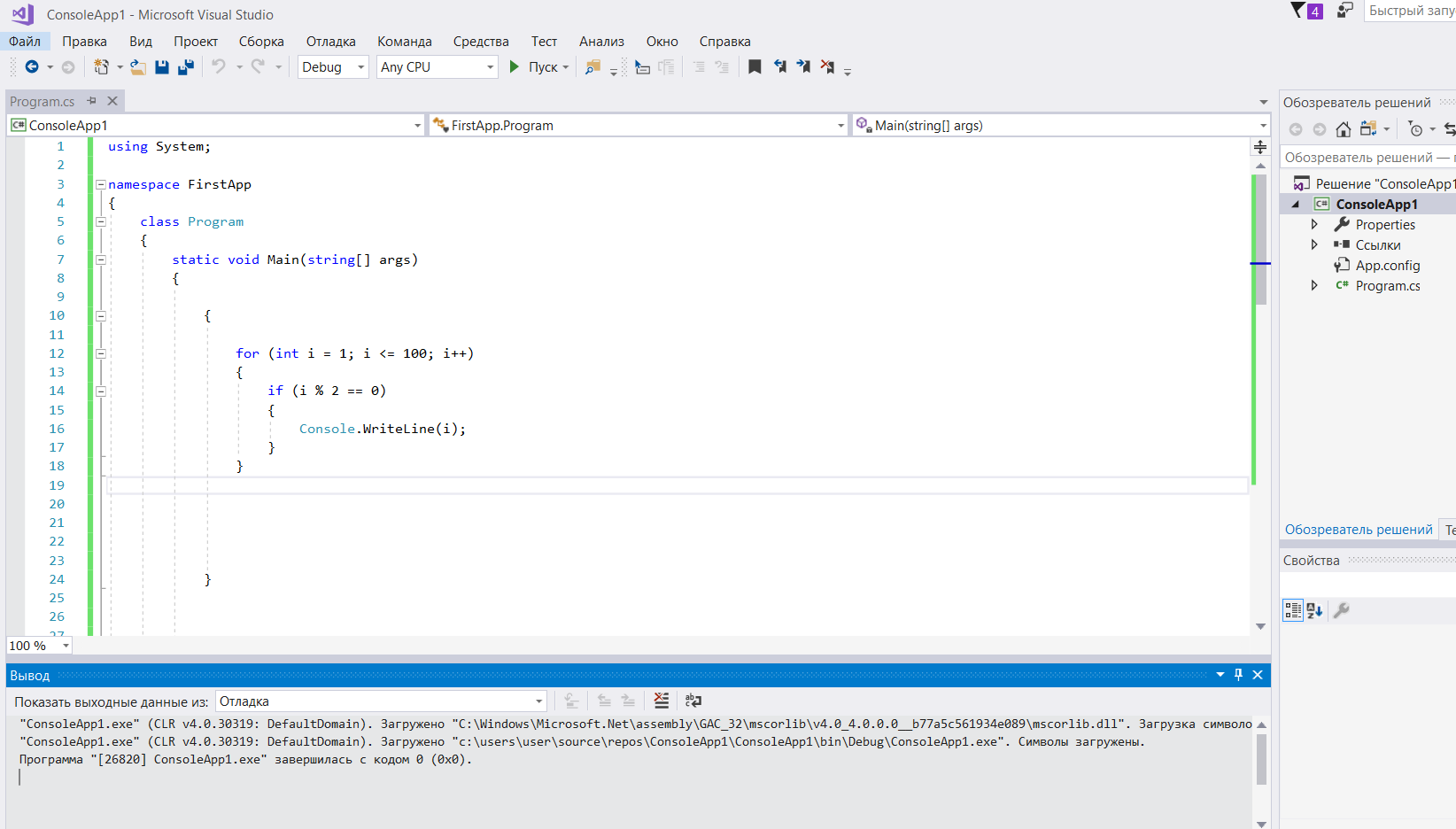
Переменная цикла

Разберем задачу, в котором надо вывести **четные числа от 1 до 100**.

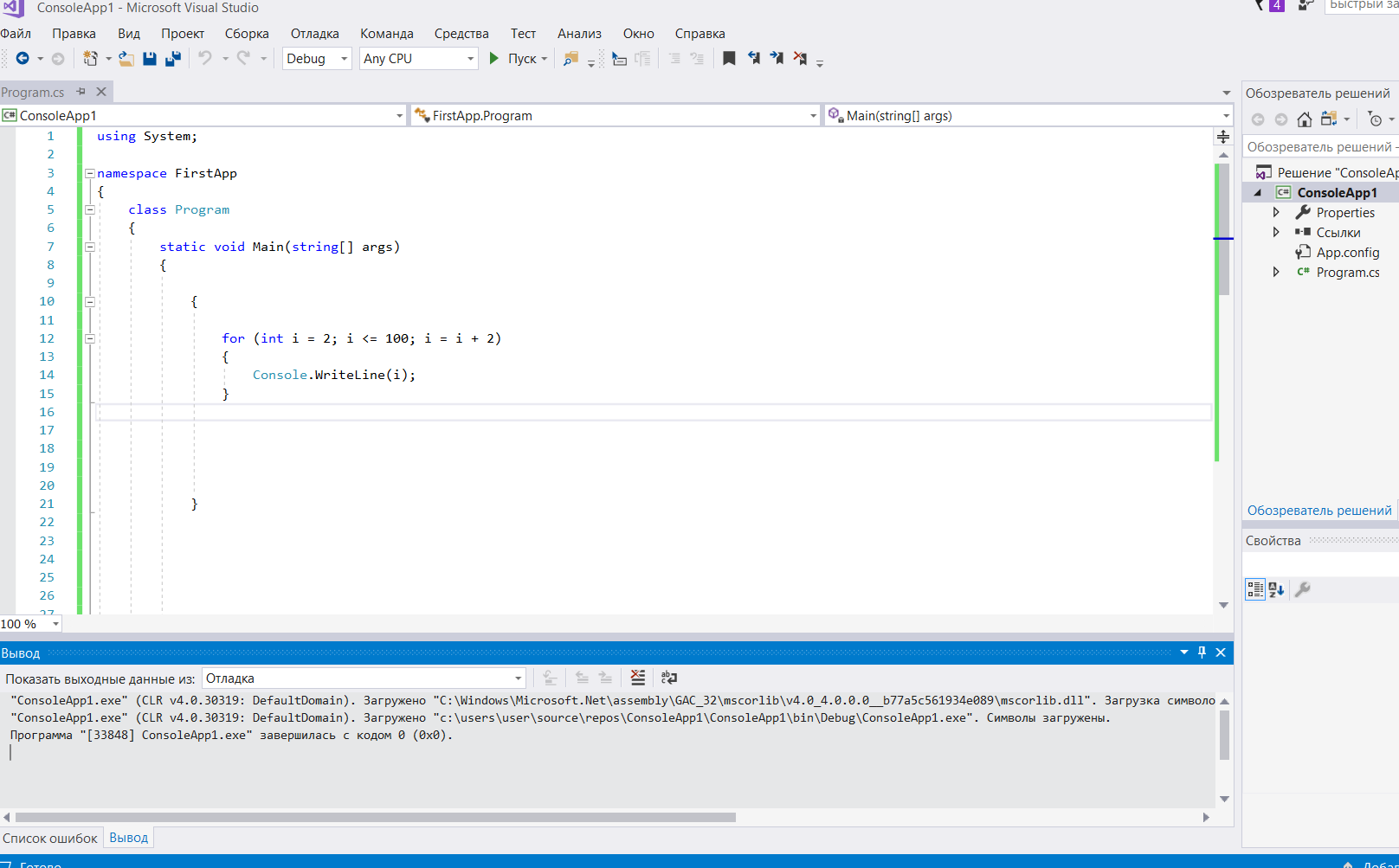
Воспользуемся циклом **for**, в котором переменная цикла будет принимать все значения от 1 до 100.

for (int i = 1; i <= 100; i++)  
{

   if(i % 2 == 0)  
   {  
      Console.WriteLine(i);  
   }  
}



Можно решить эту задачу при помощи манипулирования переменной цикла, а именно ее изменением.



Накопление суммы (сумматор)

Данный алгоритмический прием используется, когда надо просуммировать большое количество чисел. Для этого переменной, в которую будет записываться сумма, в начале присваивается **нулевое** значение, затем делается цикл, где **на каждом** шаге к этой переменной добавляется очередное число.

**Пример:** Просуммировать все целые числа от 1 до 100.

int sum = 0;

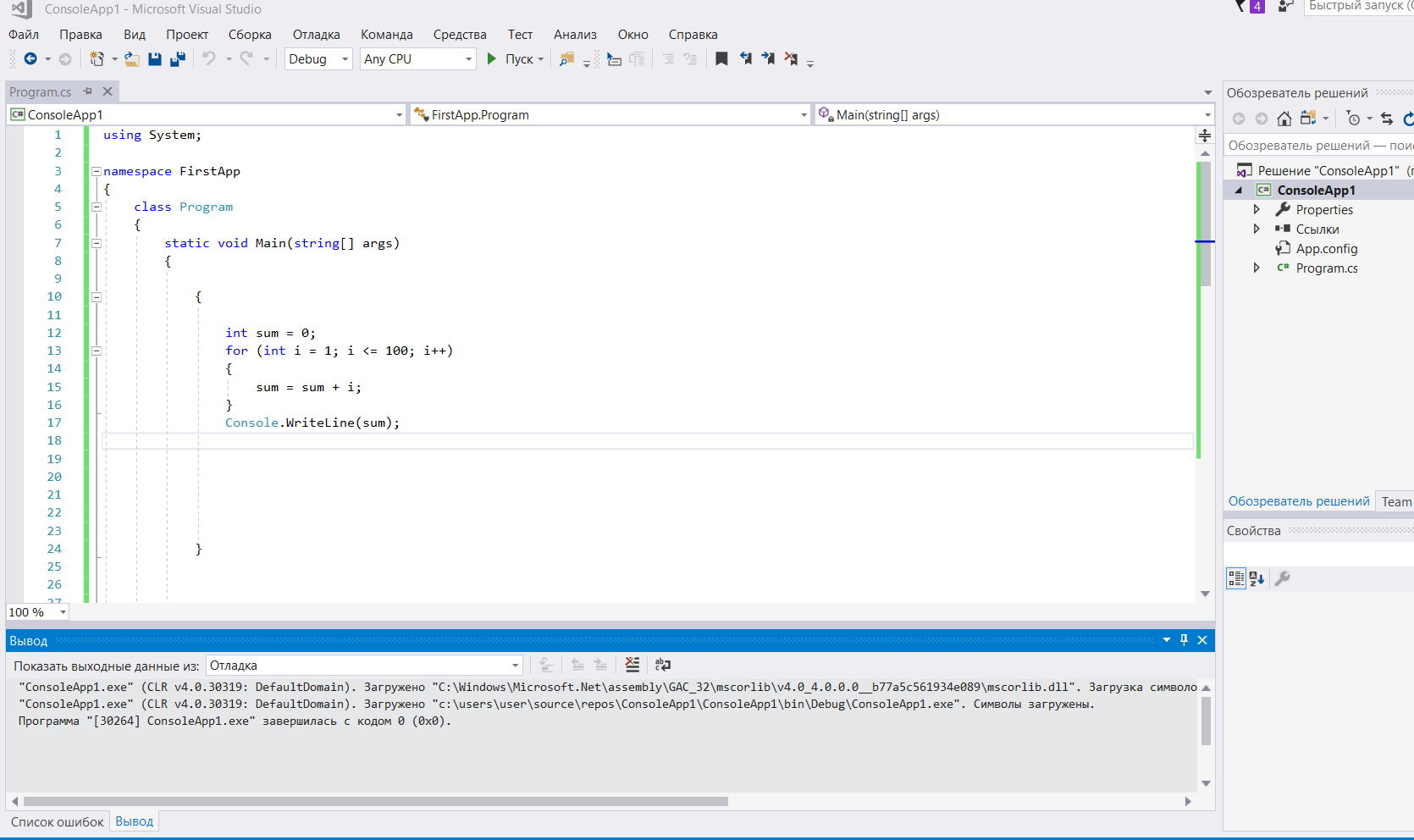
for (int i = 1; i <= 100; i++)

{

   sum = sum + i;

}

Console.WriteLine(sum);



Сумма 10 чисел

Задача следующая: пользователь вводит 10 чисел. Нужно найти сумму введенных чисел. Будем решать следующим образом: В цикле которая выполняется 10 раз, будем **вводить** число и **суммировать в переменную-**сумматор.

int sum = 0;

int number;

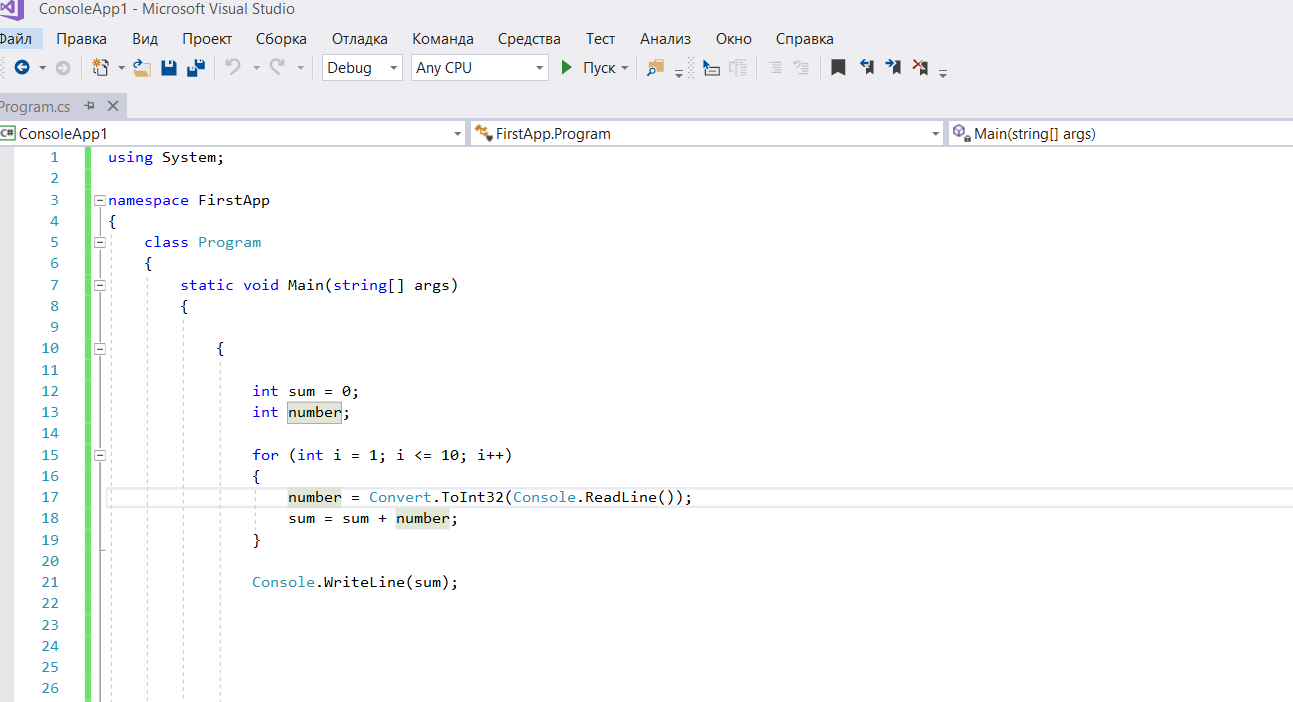
for (int i = 1; i <= 10; i++)

{

  number = Convert.ToInt32(Console.ReadLine());  
  sum = sum + number;

}

Console.WriteLine(sum);



Цикл While

Цикл **while** позволяет выполнить одну и ту же последовательность действий пока проверяемое условие **истинно**. При этом условие записывается до тела цикла и проверяется до выполнения тела цикла.

while (условие продолжения)

{

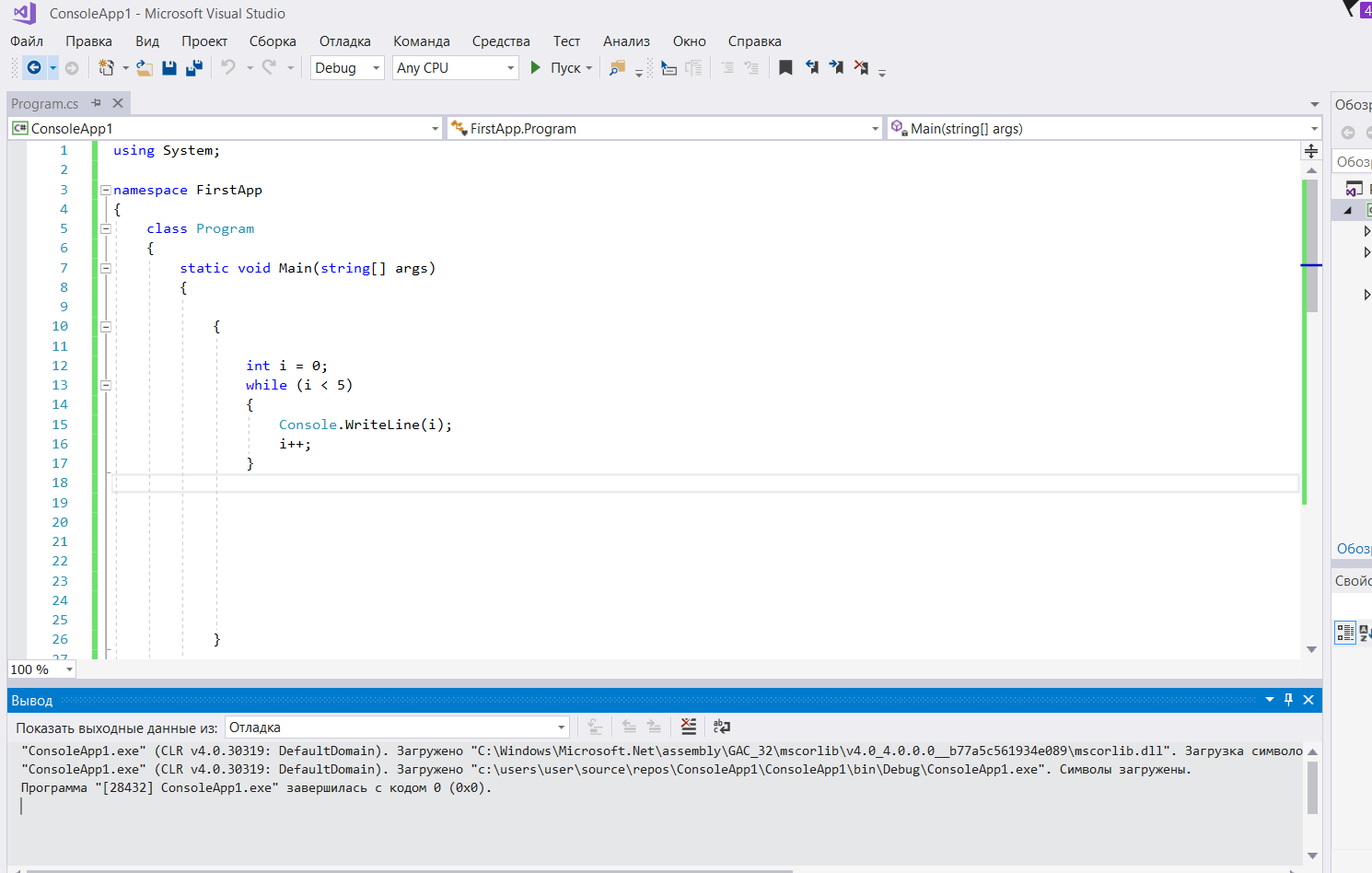
  //блок кода, который будет повторяться

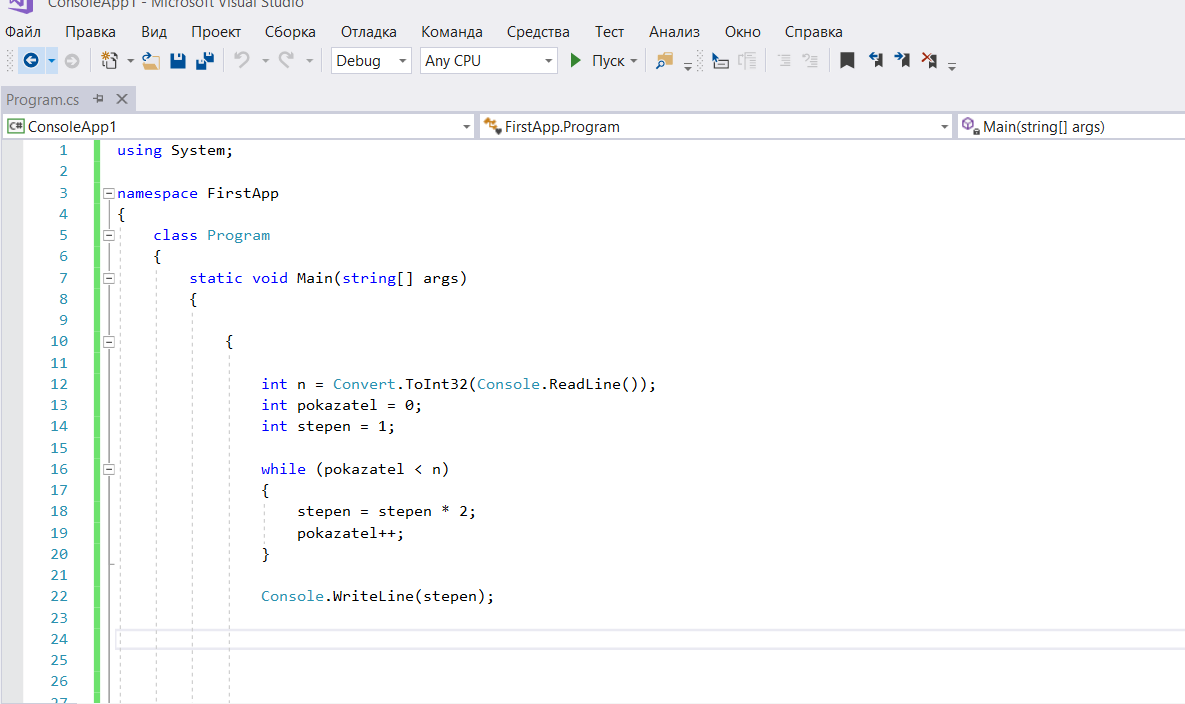
Пример:

int i = 0;

while (i < 5)  
{  
  Console.WriteLine(i);

  i++;  
}





Вложенные циклы

Циклы позволяют повторять выполнение любого набора операторов

Разберем следующую задачу: напечатать числа в виде следующей таблицы:

    5 5 5 5 5 5

    5 5 5 5 5 5

    5 5 5 5 5 5

Данная таблица состоит из трех строчек, в каждой из которых число 5 напечатано 6 раз. Строчку из шести чисел можно напечатать с помощью одного цикла **for**:

for (int i = 1; i <= 6; i++)

{

   Console.Write(5 + " ");

}

Чтобы повторить вывод строчки 3 раза, вставляем этот цикл внутрь другого:

for (int k = 1; k <= 3; k++)

{  
   for (int i = 1; i <= 6; i++)

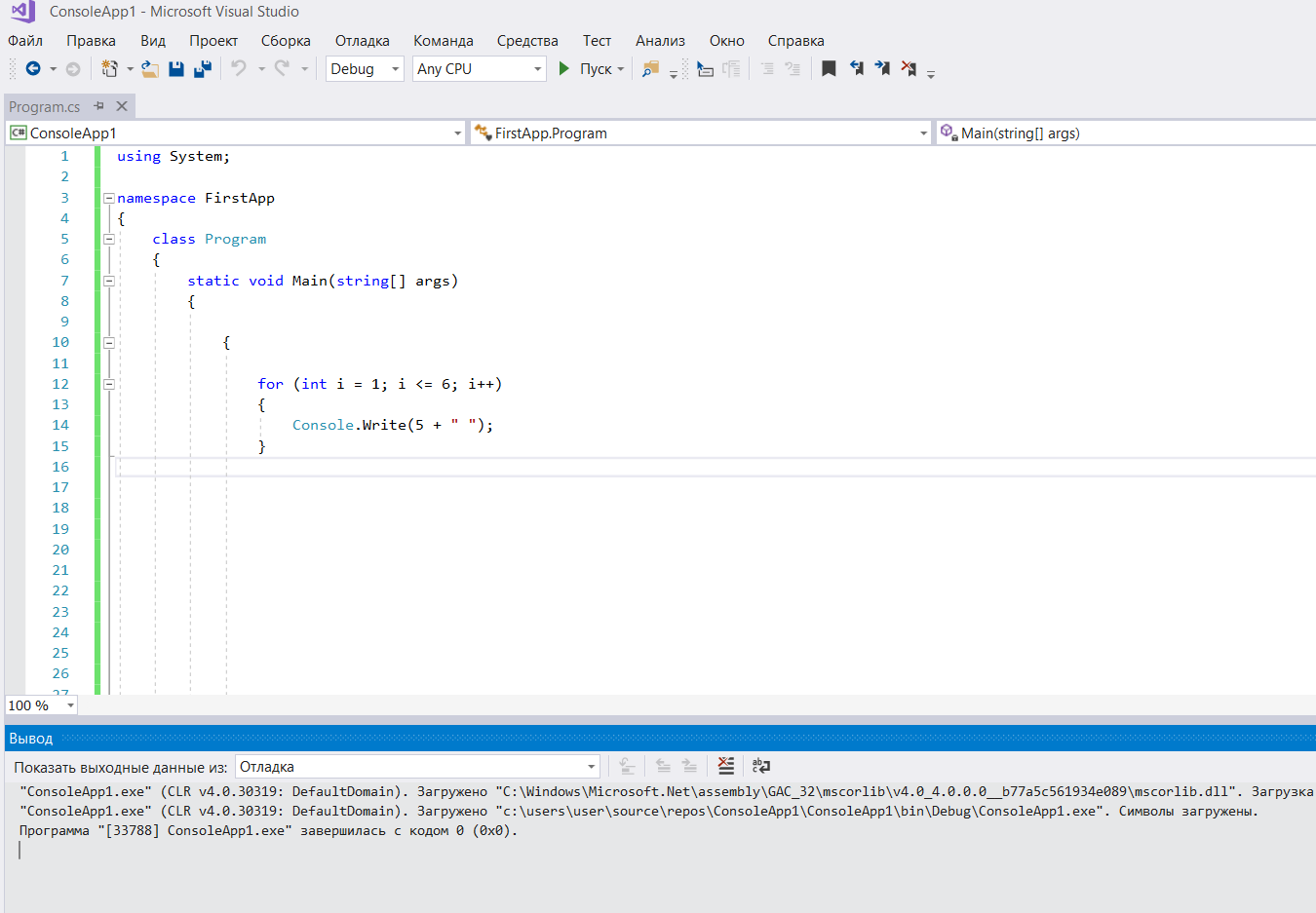
   {

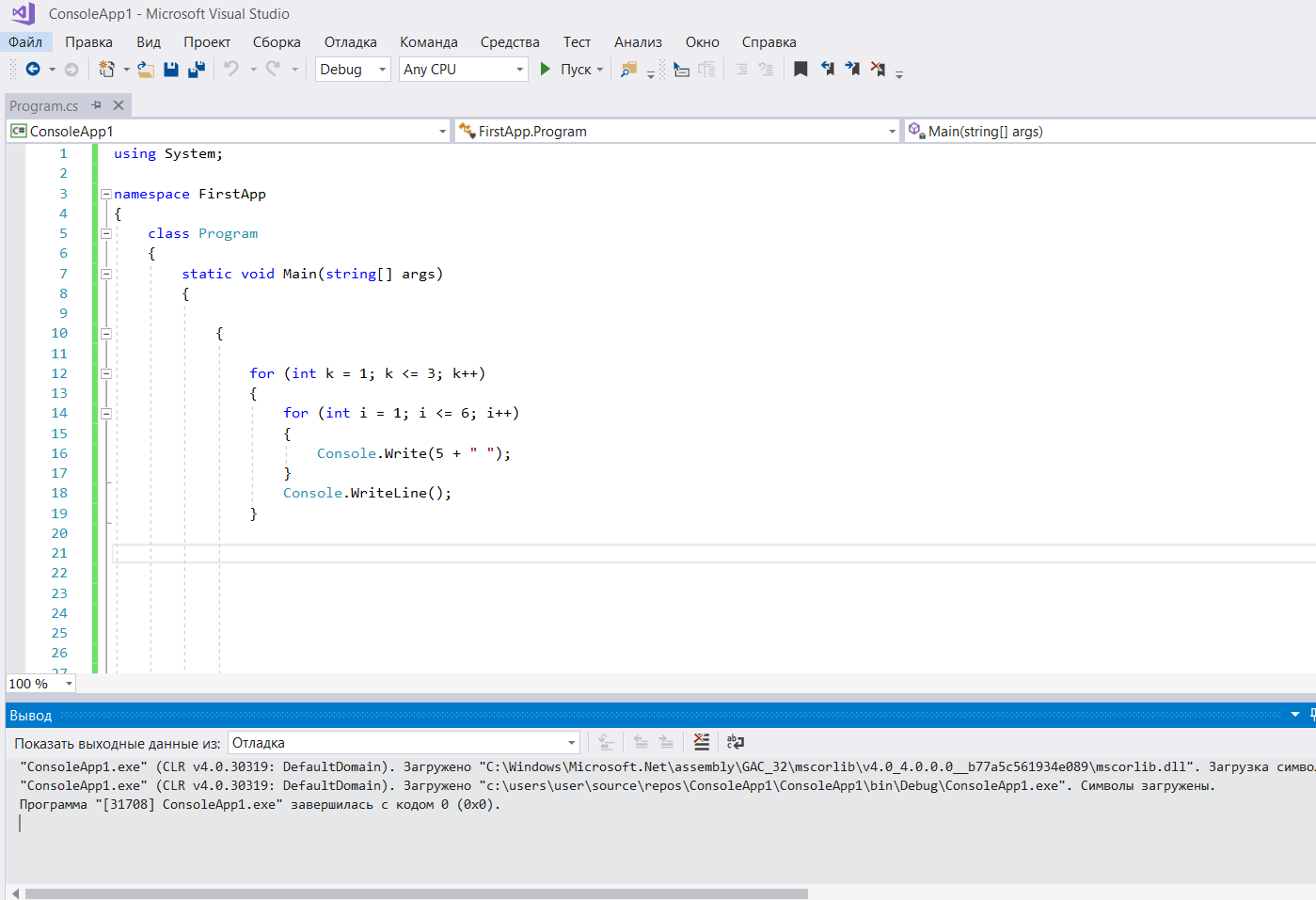
      Console.Write(5 + " ");

   }

   Console.WriteLine();

}





Вывод: Прошли курс по разработке с# и приложили разработанные приложения в репозиторий.