

Министерство науки и высшего образования Российской Федерации

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

ITMO University

ЛАБОРАТОРНАЯ РАБОТА №4

По дисциплине Объектно-ориентированное программирование

Тема работы Создание и использование методов

Обучающийся Крестьянова Елизавета Федоровна

Факультет факультет инфокоммуникационных технологий

Группа К3223

Направление подготовки 11.03.02 Инфокоммуникационные технологии и
системы связи

Образовательная программа Программирование в
инфокоммуникационных системах

Обучающийся	_____	_____	<u>Крестьянова Е.Ф.</u>
	(дата)	(подпись)	(Ф.И.О.)

Руководитель	_____	_____	<u>Иванов С.Е.</u>
	(дата)	(подпись)	(Ф.И.О.)

СОДЕРЖАНИЕ

Стр.

ВВЕДЕНИЕ	3
1 Упражнение №1. Использование параметров в методах, возвращающих значения	4
2 Упражнение №2. Использование в методах параметров, передаваемых по ссылке	6
3 Упражнение №3. Использование возвращаемых параметров в методах	8
4 Упражнение №4. Расчёт площади треугольника с помощью метода	12
5 Упражнение №5. Вычисление корней квадратного уравнения	16
ЗАКЛЮЧЕНИЕ	19

ВВЕДЕНИЕ

В данном отчёте представлено выполнение лабораторной работы по дисциплине «Объектно-ориентированное программирование».

Цель данной работы - изучение и приобретение навыков работы с методами класса.

1 УПРАЖНЕНИЕ №1. ИСПОЛЬЗОВАНИЕ ПАРАМЕТРОВ В МЕТОДАХ, ВОЗВРАЩАЮЩИХ ЗНАЧЕНИЯ

Данное упражнение требует создания класса Utils с методом Greater, принимающим 2 целочисленных параметра и возвращающим больший из них.

Был создан проект Utils.sln и класс Utils, в котором был расписан метод Greater. Его содержание можно увидеть на рисунке 1.1.

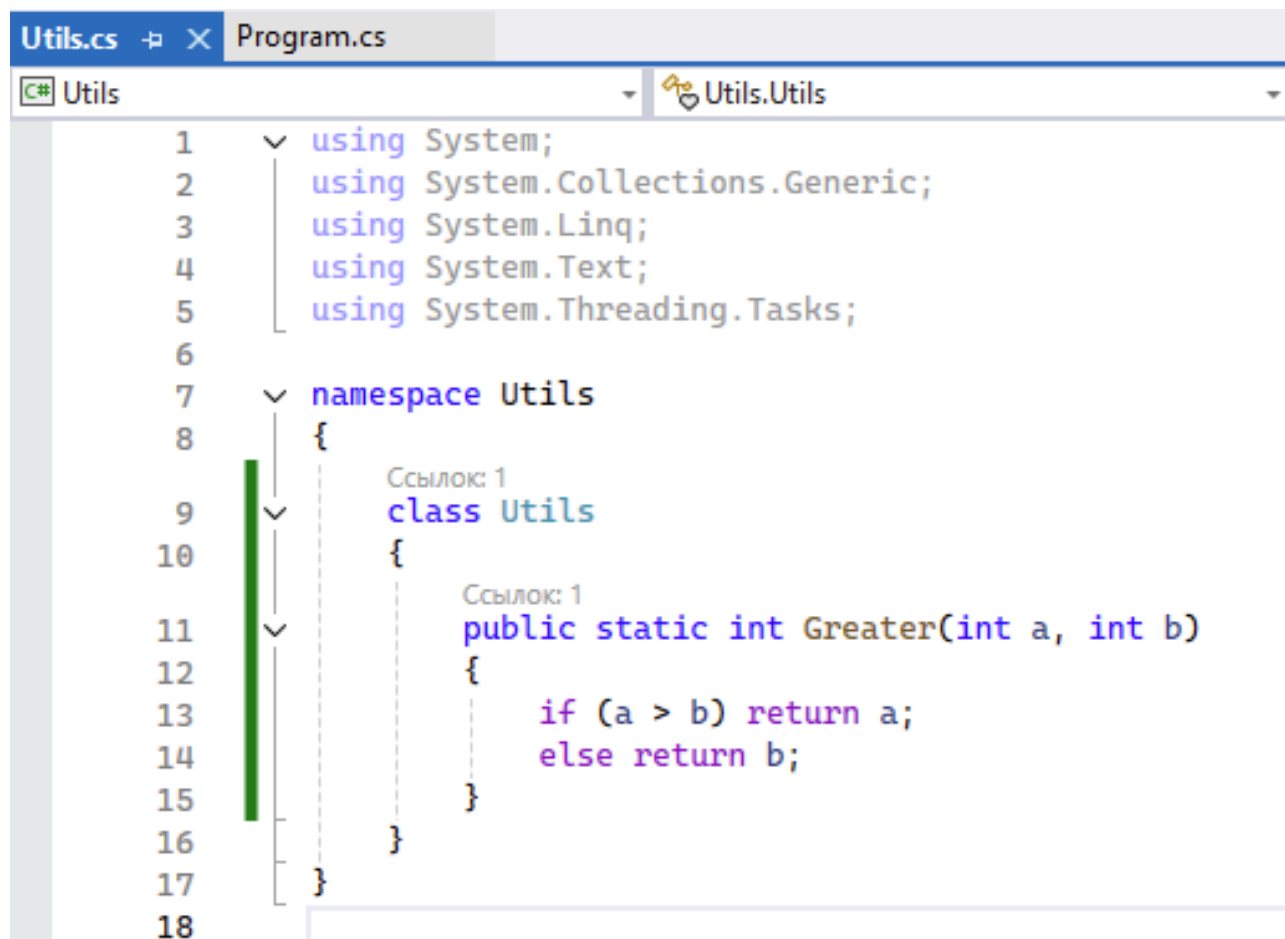
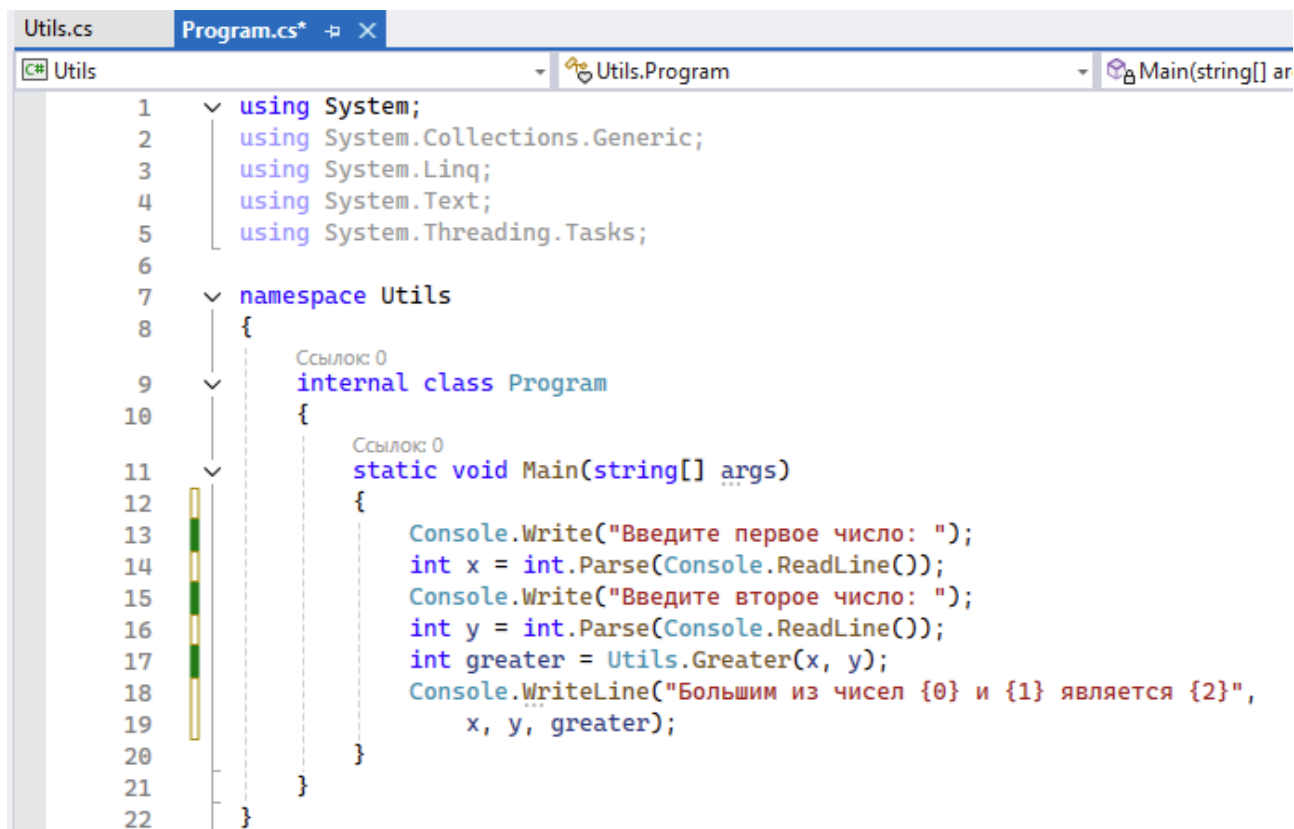


Рисунок 1.1 — Упр №1: Метод Greater

В методе Main класса Program проекта был добавлен код считывания ввода пользователя, присваивание переменной greater результата функции Greater из утилиты Utils и вывод этой переменной.

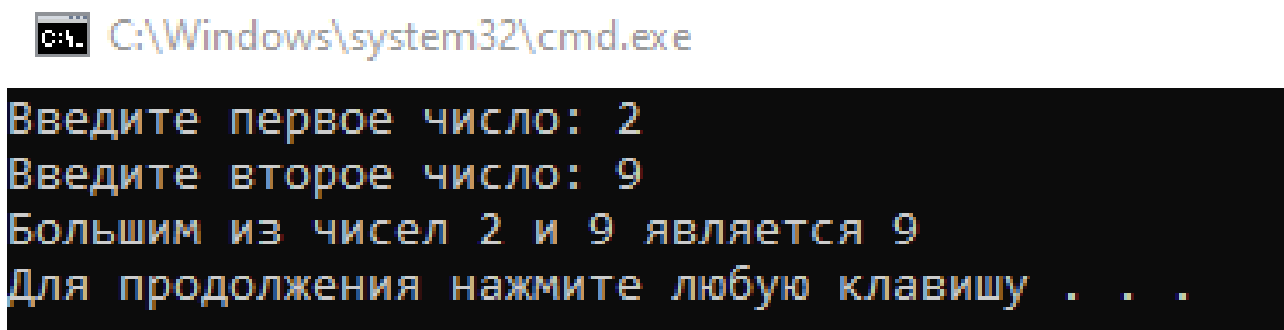
Код класса Program представлен на рисунке 1.2.



```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Utils
8  {
9      internal class Program
10     {
11         static void Main(string[] args)
12         {
13             Console.Write("Введите первое число: ");
14             int x = int.Parse(Console.ReadLine());
15             Console.Write("Введите второе число: ");
16             int y = int.Parse(Console.ReadLine());
17             int greater = Utils.Greater(x, y);
18             Console.WriteLine("Большим из чисел {0} и {1} является {2}",
19                             x, y, greater);
20         }
21     }
22 }
```

Рисунок 1.2 — Упр №1: Метод Main

Результат исполнения полученной программы можно увидеть на рисунке 1.3.



```
C:\Windows\system32\cmd.exe
Введите первое число: 2
Введите второе число: 9
Большим из чисел 2 и 9 является 9
Для продолжения нажмите любую клавишу . . .
```

Рисунок 1.3 — Упр №1: Исполнение программы

2 УПРАЖНЕНИЕ №2. ИСПОЛЬЗОВАНИЕ В МЕТОДАХ ПАРАМЕТРОВ, ПЕРЕДАВАЕМЫХ ПО ССЫЛКЕ

В данном упражнении было необходимо создать метод Swap, меняющий местами значения параметров.

В утилите Utils из предыдущего упражнения был добавлен метод Swap, принимающих параметры, передаваемые по ссылке. Ссылки используются, когда необходимо передать сами переменные для их изменения, а не просто их значения. Этот метод затем меняет местами значения параметров. Его код показан на рисунке 2.1.

Ссылка: 2

```
class Utils
```

```
{
```

Ссылка: 1

```
public static int Greater(int a, int b)
```

```
{
```

```
    if (a > b) return a;
```

```
    else return b;
```

```
}
```

Ссылка: 1

```
public static void Swap(ref int a, ref int b)
```

```
{
```

```
    int temp = a;
```

```
    a = b;
```

```
    b = temp;
```

```
}
```

```
}
```

Рисунок 2.1 — Упр №2: Метод Swap

В методе Main класса Program были добавлены выводы значений переменных x, y из предыдущего упражнения, до вызова метода Swap, и после. Добавленные строки представлены на рисунке 2.2.

Ссылка: 0

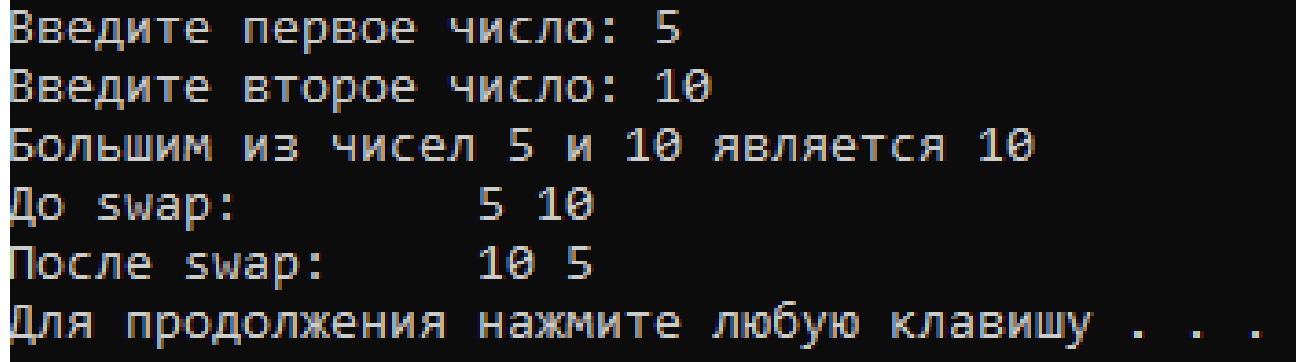
```
static void Main(string[] args)
{
    Console.Write("Введите первое число: ");
    int x = int.Parse(Console.ReadLine());
    Console.Write("Введите второе число: ");
    int y = int.Parse(Console.ReadLine());
    int greater = Utils.Greater(x, y);
    Console.WriteLine("Большим из чисел {0} и {1} является {2}",
        x, y, greater);

    Console.WriteLine("До swap: \t" + x + " " + y);
    Utils.Swap(ref x, ref y);
    Console.WriteLine("После swap: \t" + x + " " + y);
}
```

Рисунок 2.2 — Упр №2: Метод Main

Результат исполнения программы можно увидеть на рисунке 2.3.

 C:\Windows\system32\cmd.exe



```
Введите первое число: 5
Введите второе число: 10
Большим из чисел 5 и 10 является 10
До swap:          5 10
После swap:       10 5
Для продолжения нажмите любую клавишу . . .
```

Рисунок 2.3 — Упр №2: Исполнение программы

3 УПРАЖНЕНИЕ №3. ИСПОЛЬЗОВАНИЕ ВОЗВРАЩАЕМЫХ ПАРАМЕТРОВ В МЕТОДАХ

В данном упражнении было необходимо добавить метод Factorial в утилиты.

В утилитах был добавлен метод Factorial, которому передаются два целочисленных параметра n и answer типа out int, используемого для возвращения результата. Метод возвращает успешность выполнения метода через значение ok типа bool. В метод был добавлен обработчик исключительных случаев. Код Factorial можно увидеть на рисунке 3.1.

Ссылка: 0

```
public static bool Factorial(int n, out int answer)
{
    int k;
    int f = 1;
    bool ok = true;

    try
    {
        checked {
            for (k = 2; k <= n; k++)
            {
                f = f * k;
            }
        }
    }
    catch (Exception)
    {
        f = 0;
        ok = false;
    }

    answer = f;
    return ok;
}
```

Рисунок 3.1 — Упр №3: Метод Factorial

В метод Main класса Factorial был вставлен код, который вызывает метод Factorial, подсчитывающий факториал переменной x из предыдущих упражнений. Код представлен на рисунке 3.2.

Ссылка: 0

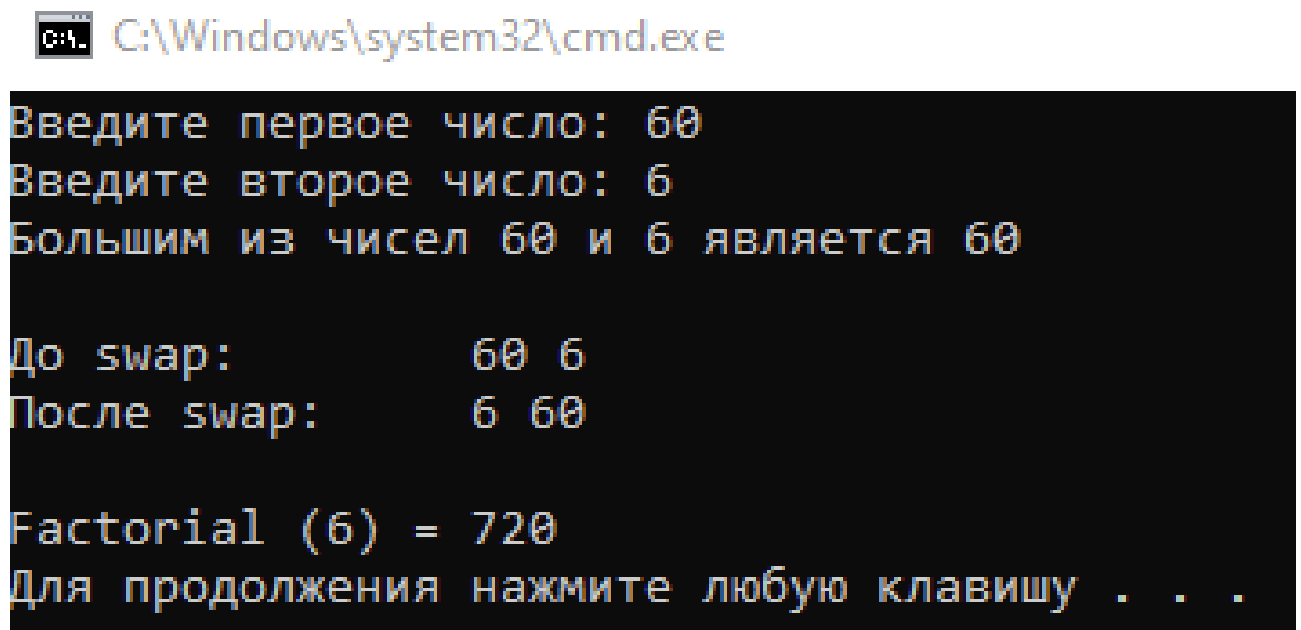
```
static void Main(string[] args)
{
    // Упражнение #1
    Console.Write("Введите первое число: ");
    int x = int.Parse(Console.ReadLine());
    Console.Write("Введите второе число: ");
    int y = int.Parse(Console.ReadLine());
    int greater = Utils.Greater(x, y);
    Console.WriteLine("Большим из чисел {0} и {1} является {2}",
        x, y, greater);
    Console.WriteLine();

    // Упражнение #2
    Console.WriteLine("До swap: \t" + x + " " + y);
    Utils.Swap(ref x, ref y);
    Console.WriteLine("После swap: \t" + x + " " + y);
    Console.WriteLine();

    // Упражнение #3
    int f;
    bool ok;
    // Console.WriteLine("Number for factorial: ");
    // x = int.Parse(Console.ReadLine());
    ok = Utils.Factorial(x, out f);
    if (ok)
        Console.WriteLine("Factorial (" + x + ") = " + f);
    else
        Console.WriteLine("Cannot compute factorial of {0}", x);
}
```

Рисунок 3.2 — Упр №3: Метод Main

Результат исполнения программы можно увидеть на рисунке 3.3.



C:\Windows\system32\cmd.exe

```
Введите первое число: 60
Введите второе число: 6
Большим из чисел 60 и 6 является 60

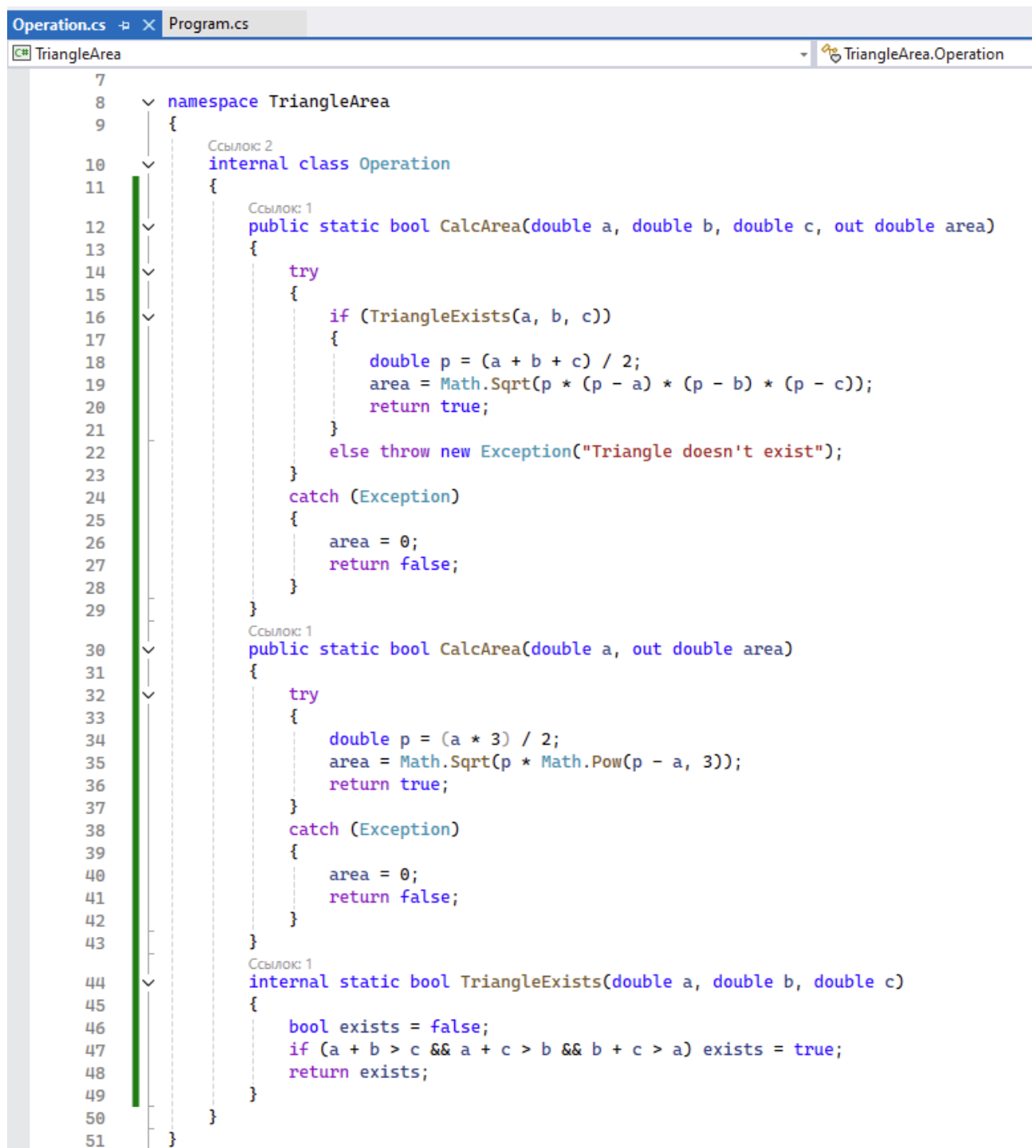
До swap:          60 6
После swap:       6 60

Factorial (6) = 720
Для продолжения нажмите любую клавишу . . .
```

Рисунок 3.3 — Упр №3: Исполнение программы

4 УПРАЖНЕНИЕ №4. РАСЧЁТ ПЛОЩАДИ ТРЕУГОЛЬНИКА С ПОМОЩЬЮ МЕТОДА

В этом упражнении был создан новый проект TriangleArea с классом Operation, в котором были определены 3 статических метода: поиска площади треугольника CalcArea, равностороннего и обычного, а также закрытый метод проверки наличия треугольника. В обоих методах нахождения площади использовались формула Герона и обработчик исключительных ситуаций. В методе нахождения обычного треугольника проверяется его существование. Код этого класса можно увидеть на рисунке 4.1.



```
7
8 namespace TriangleArea
9 {
10     internal class Operation
11     {
12         public static bool CalcArea(double a, double b, double c, out double area)
13         {
14             try
15             {
16                 if (TriangleExists(a, b, c))
17                 {
18                     double p = (a + b + c) / 2;
19                     area = Math.Sqrt(p * (p - a) * (p - b) * (p - c));
20                     return true;
21                 }
22                 else throw new Exception("Triangle doesn't exist");
23             }
24             catch (Exception)
25             {
26                 area = 0;
27                 return false;
28             }
29         }
30         public static bool CalcArea(double a, out double area)
31         {
32             try
33             {
34                 double p = (a * 3) / 2;
35                 area = Math.Sqrt(p * Math.Pow(p - a, 3));
36                 return true;
37             }
38             catch (Exception)
39             {
40                 area = 0;
41                 return false;
42             }
43         }
44         internal static bool TriangleExists(double a, double b, double c)
45         {
46             bool exists = false;
47             if (a + b > c && a + c > b && b + c > a) exists = true;
48             return exists;
49         }
50     }
51 }
```

Рисунок 4.1 — Упр №4: Класс Operation

В методе Main класса Program был создан следующий функционал: принятие ввода пользователя на выбор треугольника (равносторонний он или нет), ввода сторон треугольника и вывод подсчитанной через созданную утилиту площади. Этот код представлен на рисунке 4.2.

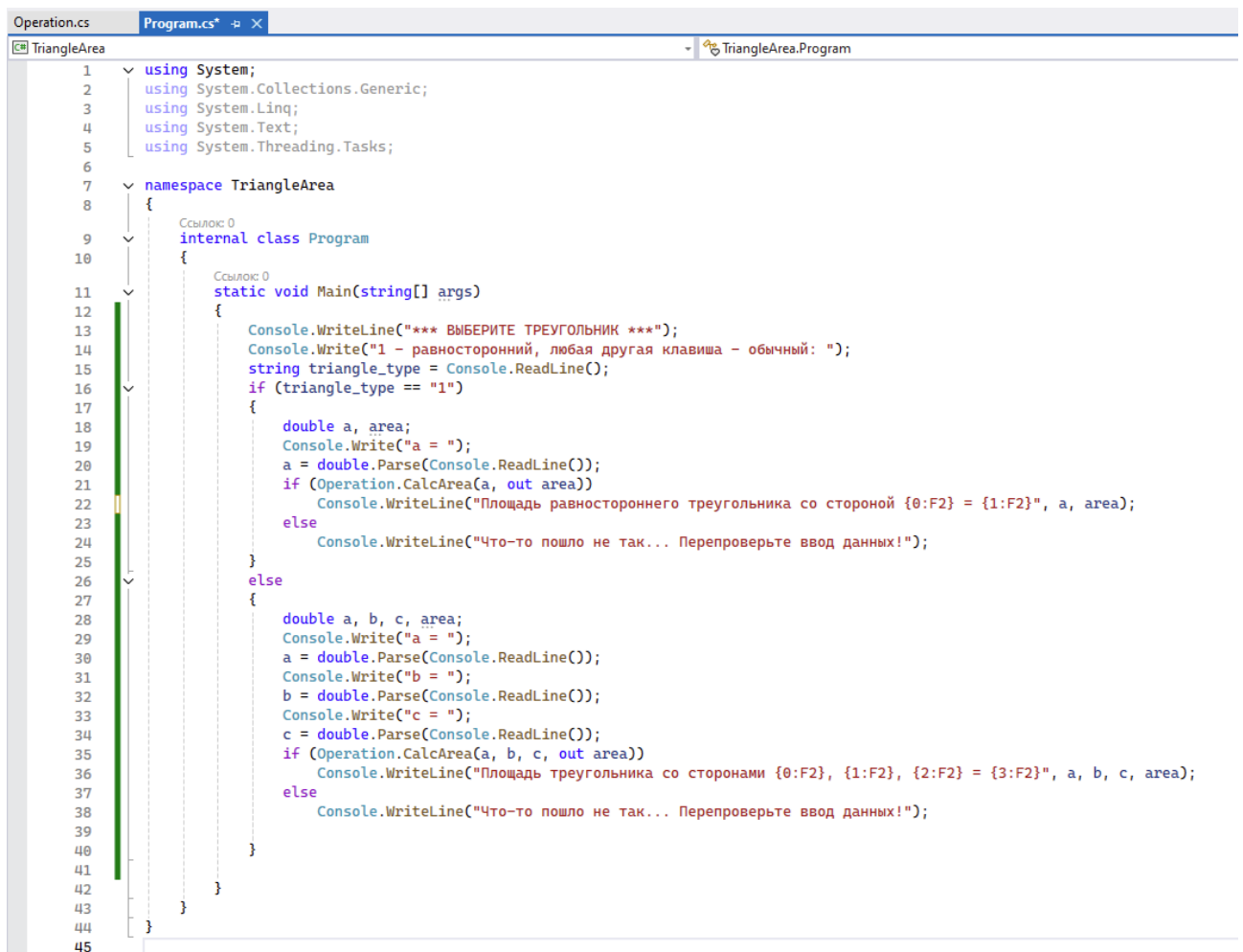


Рисунок 4.2 — Упр №4: Метод Main

Пример расчёта площади равностороннего треугольника можно увидеть на рисунке 4.3.

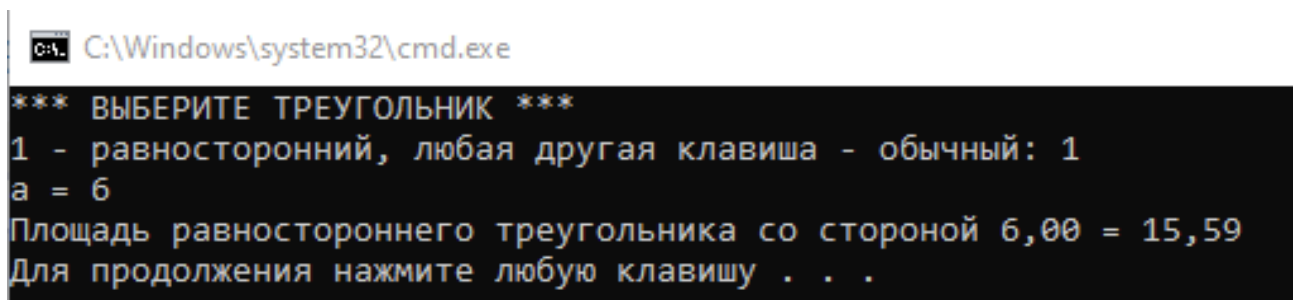


Рисунок 4.3 — Упр №4: Расчёт площади равностороннего треугольника

Пример расчёта площади иного треугольника можно увидеть на рисунке 4.4.

```
C:\Windows\system32\cmd.exe

*** ВЫБЕРИТЕ ТРЕУГОЛЬНИК ***
1 - равносторонний, любая другая клавиша - обычный: 2
a = 3
b = 4
c = 5
Площадь треугольника со сторонами 3,00, 4,00, 5,00 = 6,00
Для продолжения нажмите любую клавишу . . .
```

Рисунок 4.4 — Упр №4: Расчёт площади треугольника

5 УПРАЖНЕНИЕ №5. ВЫЧИСЛЕНИЕ КОРНЕЙ КВАДРАТНОГО УРАВНЕНИЯ

В этом упражнении необходимо реализовать метод вычисления корней квадратного уравнения.

Был создан новый проект QuadraticEquation с классом Utils. В нём был написан метод нахождения дискриминанта квадратного уравнения FindSolution. Если он больше нуля, ищется два корня, если он равен нулю, эти корни приравниваются, иначе они не существуют. Этот метод можно увидеть на рисунке 5.1.

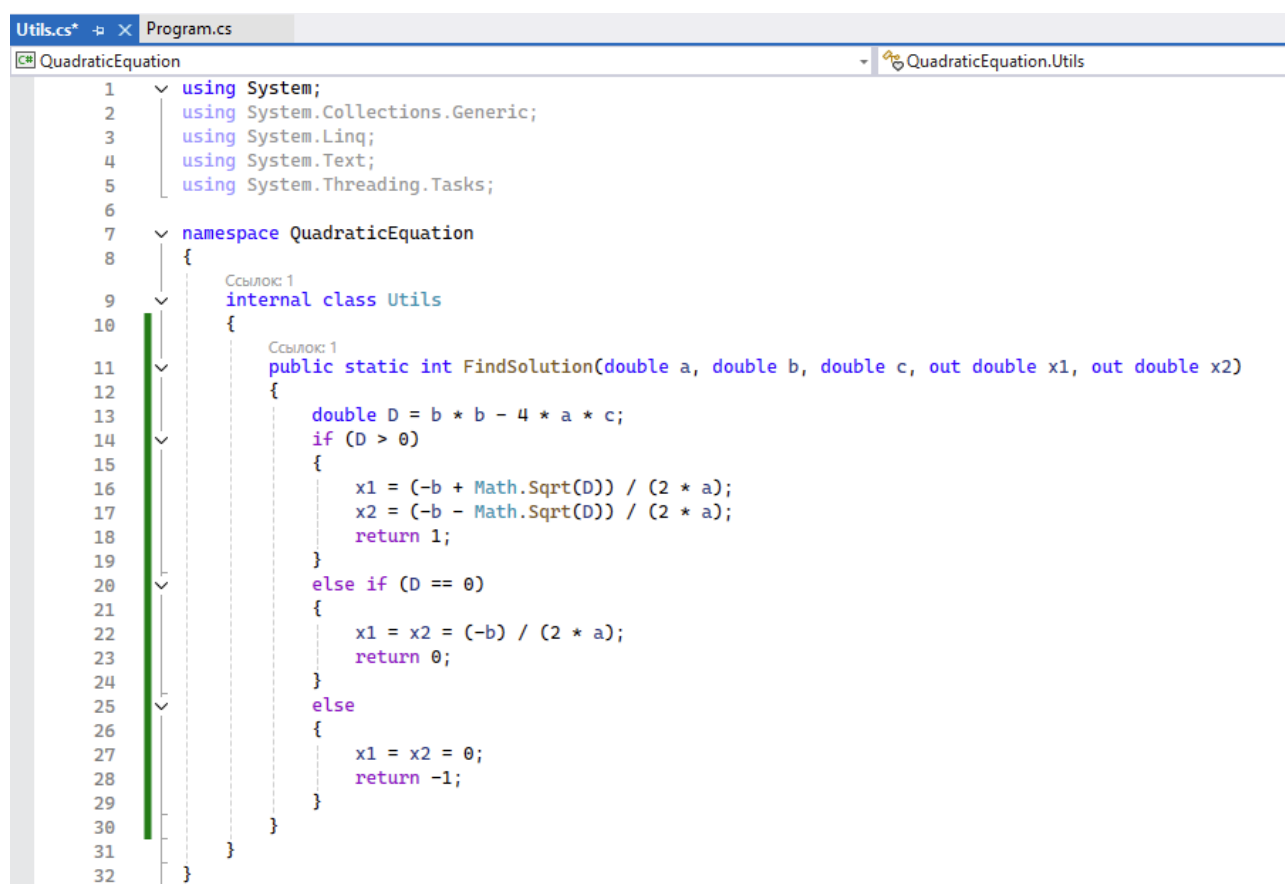
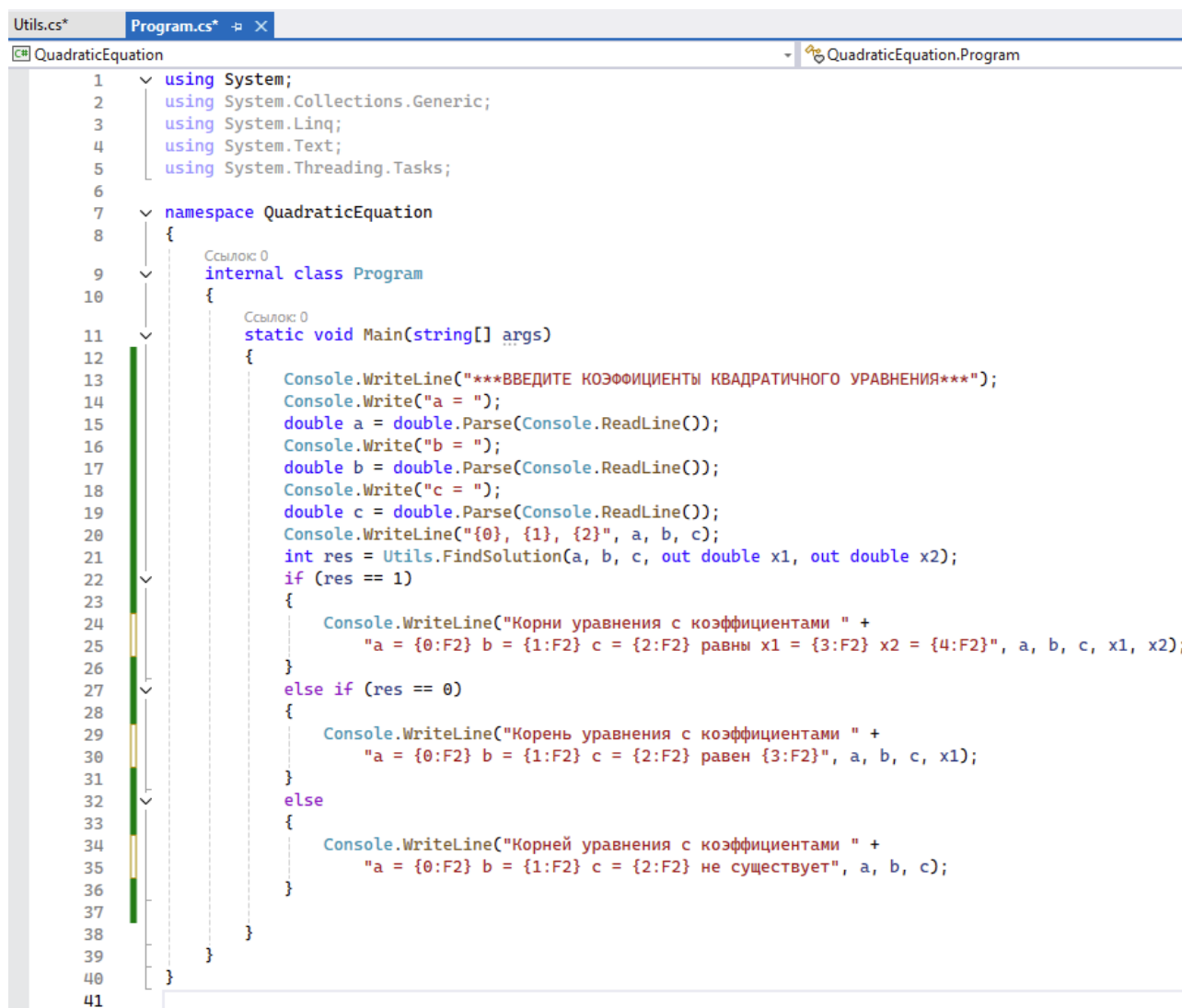


Рисунок 5.1 — Упр №5: Метод FindSolution

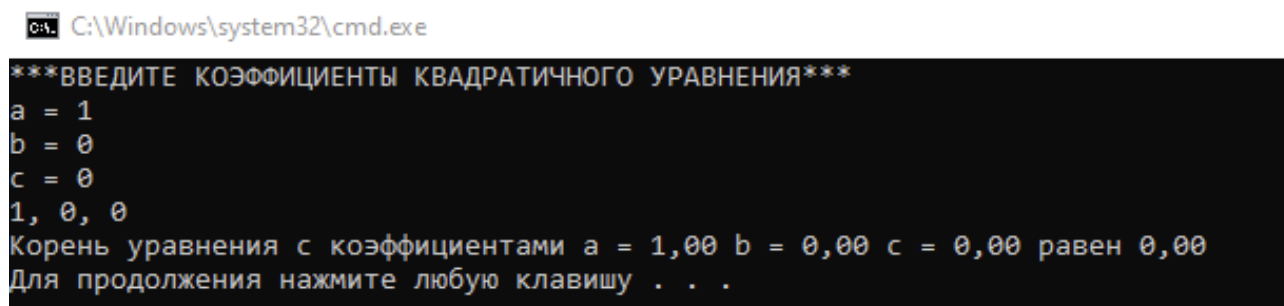
В методе Main класса Program был написан код принимающий ввод пользователя, задающий коэффициенты квадратного уравнения, и выводится определённая строка, в зависимости от результируемого дискриминанта. Этот код показан на рисунке 5.2.



```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace QuadraticEquation
8 {
9     internal class Program
10     {
11         static void Main(string[] args)
12         {
13             Console.WriteLine("***ВВЕДИТЕ КОЭФФИЦИЕНТЫ КВАДРАТИЧНОГО УРАВНЕНИЯ***");
14             Console.Write("a = ");
15             double a = double.Parse(Console.ReadLine());
16             Console.Write("b = ");
17             double b = double.Parse(Console.ReadLine());
18             Console.Write("c = ");
19             double c = double.Parse(Console.ReadLine());
20             Console.WriteLine("{0}, {1}, {2}", a, b, c);
21             int res = Utils.FindSolution(a, b, c, out double x1, out double x2);
22             if (res == 1)
23             {
24                 Console.WriteLine("Корни уравнения с коэффициентами " +
25                     "a = {0:F2} b = {1:F2} c = {2:F2} равны x1 = {3:F2} x2 = {4:F2}", a, b, c, x1, x2);
26             }
27             else if (res == 0)
28             {
29                 Console.WriteLine("Корень уравнения с коэффициентами " +
30                     "a = {0:F2} b = {1:F2} c = {2:F2} равен {3:F2}", a, b, c, x1);
31             }
32             else
33             {
34                 Console.WriteLine("Корней уравнения с коэффициентами " +
35                     "a = {0:F2} b = {1:F2} c = {2:F2} не существует", a, b, c);
36             }
37         }
38     }
39 }
40
41
```

Рисунок 5.2 — Упр №5: Метод Main

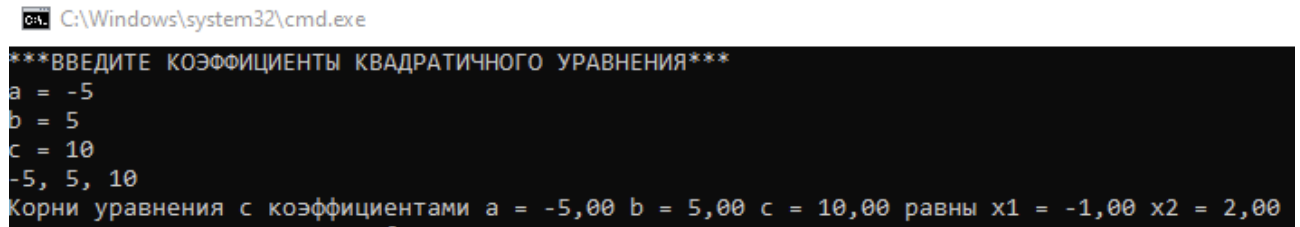
Пример работы программы над квадратичным уравнением с нулевым дискриминантом показан на рисунке 5.3.



```
C:\Windows\system32\cmd.exe
***ВВЕДИТЕ КОЭФФИЦИЕНТЫ КВАДРАТИЧНОГО УРАВНЕНИЯ***
a = 1
b = 0
c = 0
1, 0, 0
Корень уравнения с коэффициентами a = 1,00 b = 0,00 c = 0,00 равен 0,00
Для продолжения нажмите любую клавишу . . .
```

Рисунок 5.3 — Упр №5: Нулевой дискриминант

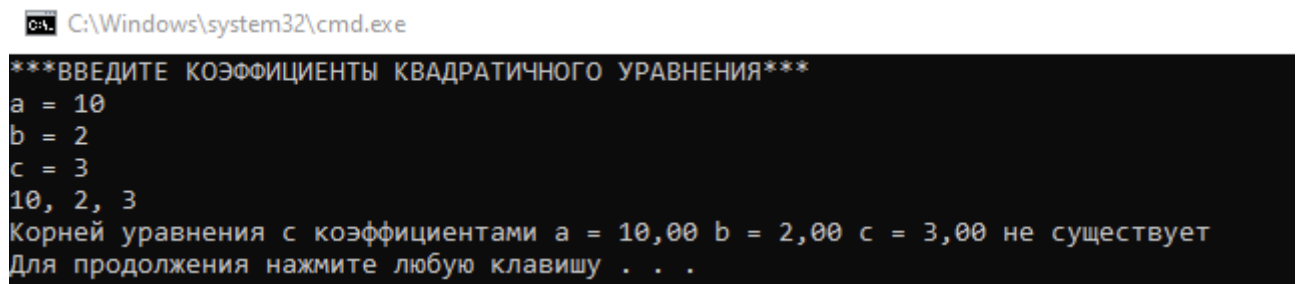
Пример работы программы над квадратичным уравнением с положительным дискриминантом показан на рисунке 5.4.



```
C:\Windows\system32\cmd.exe
***ВВЕДИТЕ КОЭФФИЦИЕНТЫ КВАДРАТИЧНОГО УРАВНЕНИЯ***
a = -5
b = 5
c = 10
-5, 5, 10
Корни уравнения с коэффициентами a = -5,00 b = 5,00 c = 10,00 равны x1 = -1,00 x2 = 2,00
```

Рисунок 5.4 — Упр №5: Положительный дискриминант

Пример работы программы над квадратичным уравнением с отрицательным дискриминантом показан на рисунке 5.5.



```
C:\Windows\system32\cmd.exe
***ВВЕДИТЕ КОЭФФИЦИЕНТЫ КВАДРАТИЧНОГО УРАВНЕНИЯ***
a = 10
b = 2
c = 3
10, 2, 3
Корней уравнения с коэффициентами a = 10,00 b = 2,00 c = 3,00 не существует
Для продолжения нажмите любую клавишу . . .
```

Рисунок 5.5 — Упр №5: Отрицательный дискриминант

ЗАКЛЮЧЕНИЕ

При выполнении лабораторной работы были созданы различные утилиты-методы для класса `Utils`, передачи данных к ним - как значений, так и самих переменных через ссылки; и возвращения их.

Цель изучения и приобретения навыков работы с методами класса была выполнена.