Санкт-Петербургский Национальный Исследовательский
Университет Информационных Технологий, Механики и Оптики

Факультет инфокоммуникационных технологий

Лабораторная работа №3
Использование выражений

Выполнил
Стафеев И.А.

Группа
K3221

Проверил
Иванов С.Е.

Санкт-Петербург,
2024

СОДЕРЖАНИЕ

Стр.

ВВЕДЕНИЕ

Цель работы: изучение и приобретение навыков использования управляющих конструкций для организации вычислений.

Для достижения цели необходимо выполнить следующие упражнения:

1. Реализация операторов выбора
2. Реализация циклов при работе с данными размерных типов

1 Упражнение 1

1.1 Задание 1

Задача: В этом задании вы составите программу, которая выдает одно из сообщений «Внутри», «Вне», «На границе» в зависимости от того, лежит ли точка внутри заштрихованной области (положительный по oY полукруг с центром в точке (0, 0) радиусом 3), вне заштрихованной области или на ее границе.

Код программы приведен на рисунке 1. Для обработки некорректных данных были добавлены исключения.

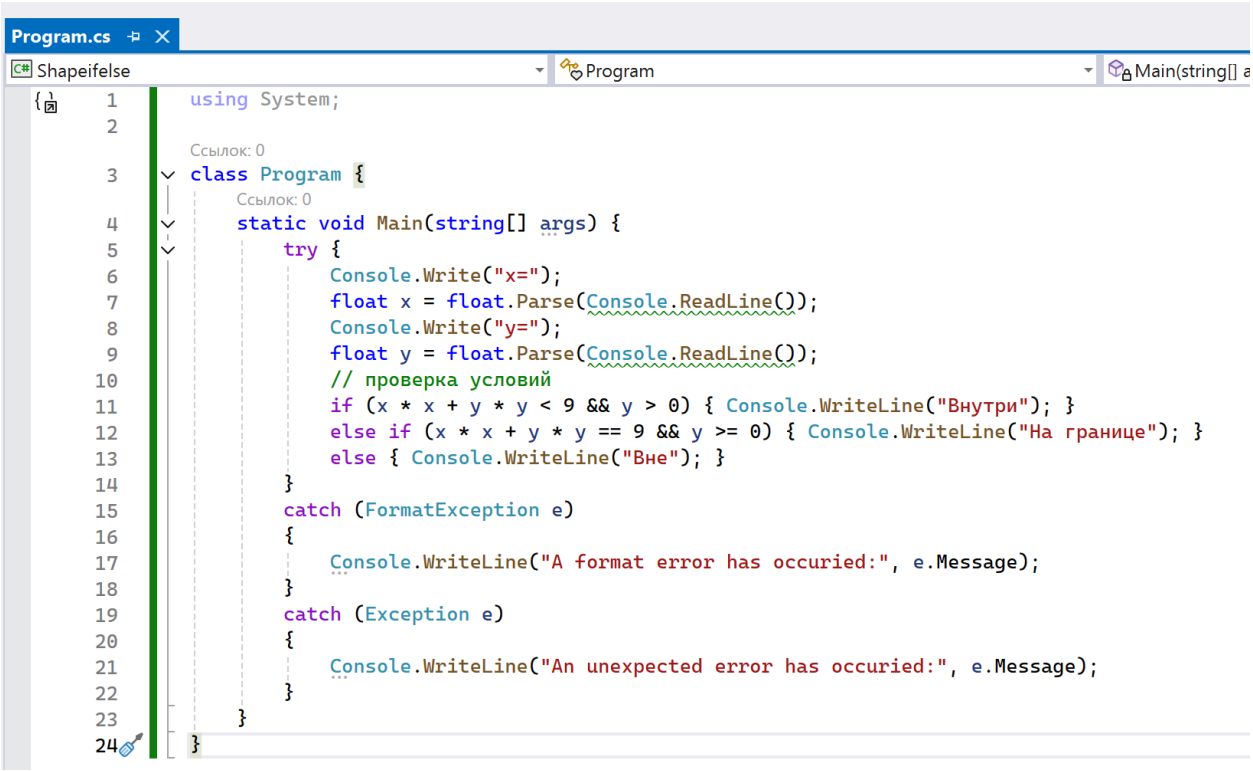


Рисунок 1 — Код программы

Примеры выполнения программы для всех 3 случаев показаны на рисунках 2, 3 и 4.

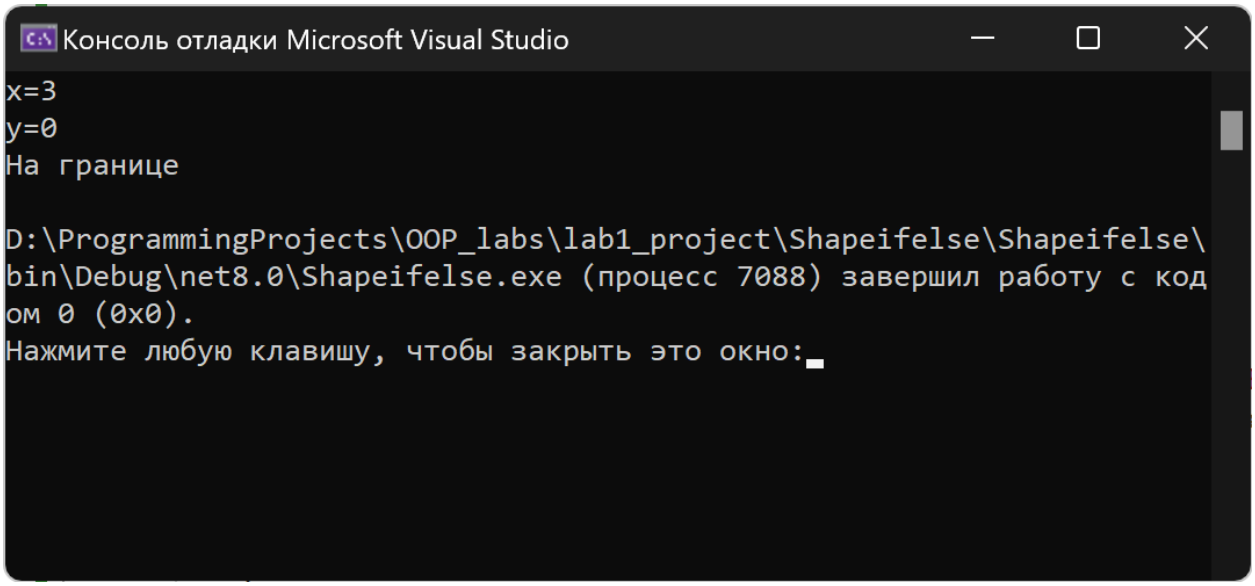


Рисунок 2 — Пример "На границе"

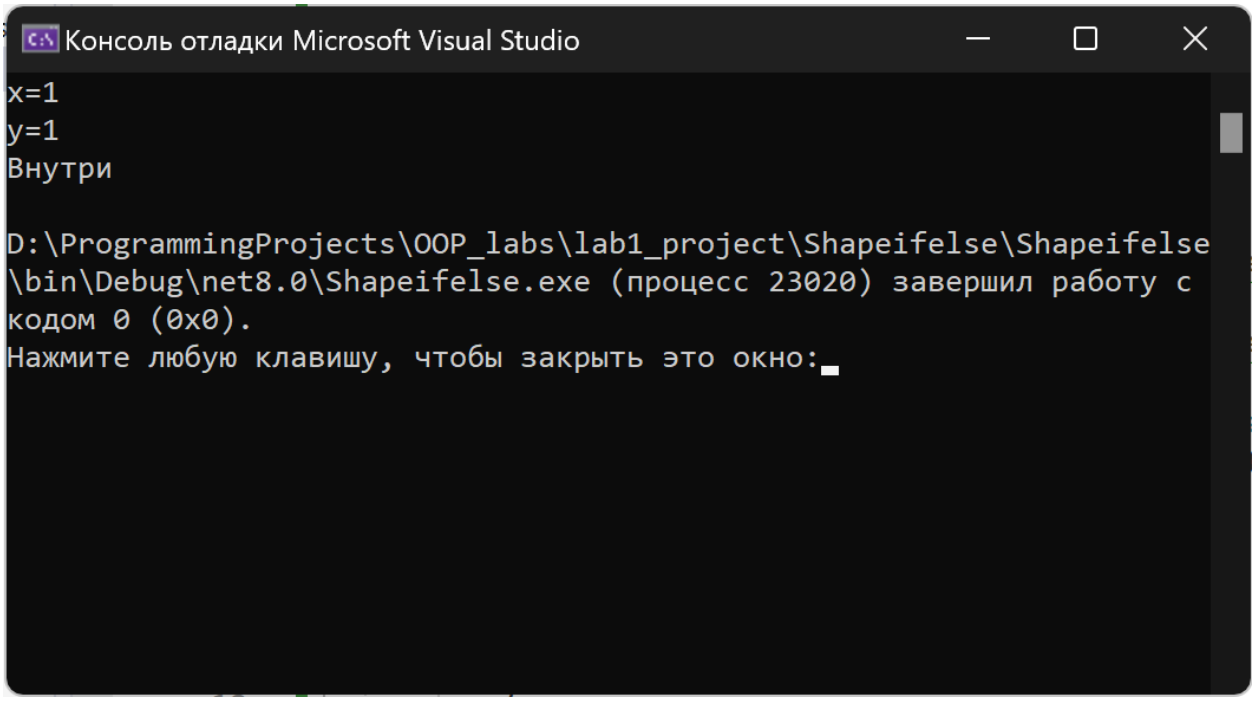


Рисунок 3 — Пример "Внутри"

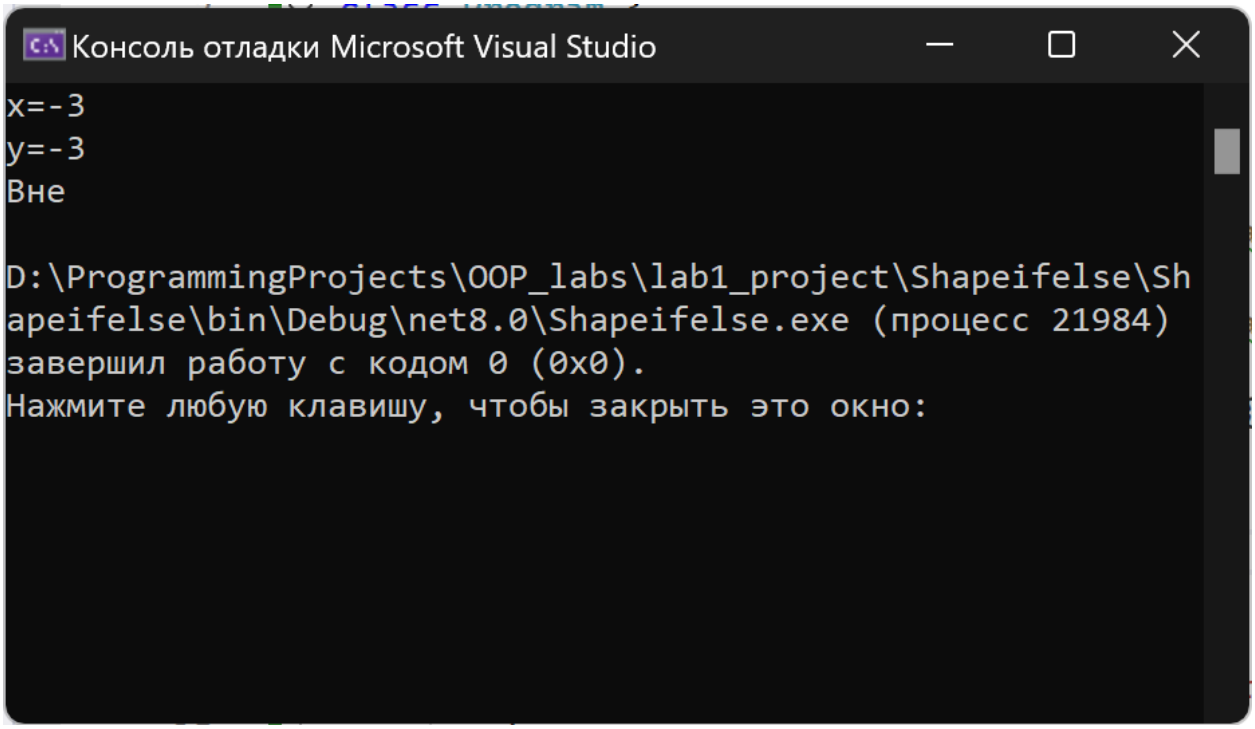


Рисунок 4 — Пример "Вне"

1.2 Задание 2

Задача: Вы создадите программу моделирующую работу калькулятора. Пользователь должен ввести первый операнд, затем требуемую операцию и второй операнд. В зависимости от знака операции будет проведен расчет результата.

Код программы приведен на рисунке 5. Для выполнения операций калькулятора использована конструкция switch, выполняющая определенную команду в зависимости от введенного значения операнда.

```
Program.cs
Calc_switch  Program  Main(string[] args)
1  using System;
2
   Ссылок: 0
3  class Program {
       Ссылок: 0
4      static void Main(string[] args)
5      {
6          Console.Write("A = ");
7          double a = double.Parse(Console.ReadLine());
8          Console.Write("OP = ");
9          char op = char.Parse(Console.ReadLine());
10         Console.Write("B = ");
11         double b = double.Parse(Console.ReadLine());
12         bool ok = true;
13         double res = 0;
14         // вычисления на "калькуляторе"
15         switch (op)
16         {
17             case '+': res = a + b; break;
18             case '-': res = a - b; break;
19             case '*': res = a * b; break;
20             case '/':
21             case ':': res = a / b; break;
22             default: ok = false; break;
23         }
24         // вывод результата
25         if (!ok) { Console.WriteLine("Error! Available operations are +, -, * and / (:)"); }
26         else { Console.WriteLine("{0} {1} {2} = {3}", a, op, b, res); }
27     }
28 }
```

Рисунок 5 — Код программы калькулятора

Пример выполнения с правильными данными показан на рисуке 6.

```
Консоль отладки Microsoft Visual Studio
OP = *
B = 0,12
1824 * 0,12 = 218,88

D:\ProgrammingProjects\OOP_labs\lab1_project\Calc_switch\Cal
c_switch\bin\Debug\net8.0\Calc_switch.exe (процесс 4348) зав
ершил работу с кодом 0 (0x0).
Нажмите любую клавишу, чтобы закрыть это окно:
```

Рисунок 6 — Выполнение программы с корректными данными

Пример выполнения при делении на ноль показан на рисунке 7. В отличие от целых чисел, для вещественных деление на ноль не вызывает ошибку, а результатом является бесконечность (подтверждение на рисунке 8). Бесконечность определена для вещественных чисел, а отличие от целых. А поскольку тип **double** не является математически точным, то любая дробь $\frac{n}{m}$ в десятичной записи будет является аппроксимацией, нахождением предела. При делении на ноль ситуация аналогична, предел будет равен бесконечности, отсюда и значение для выражение при размерном типе **double**.

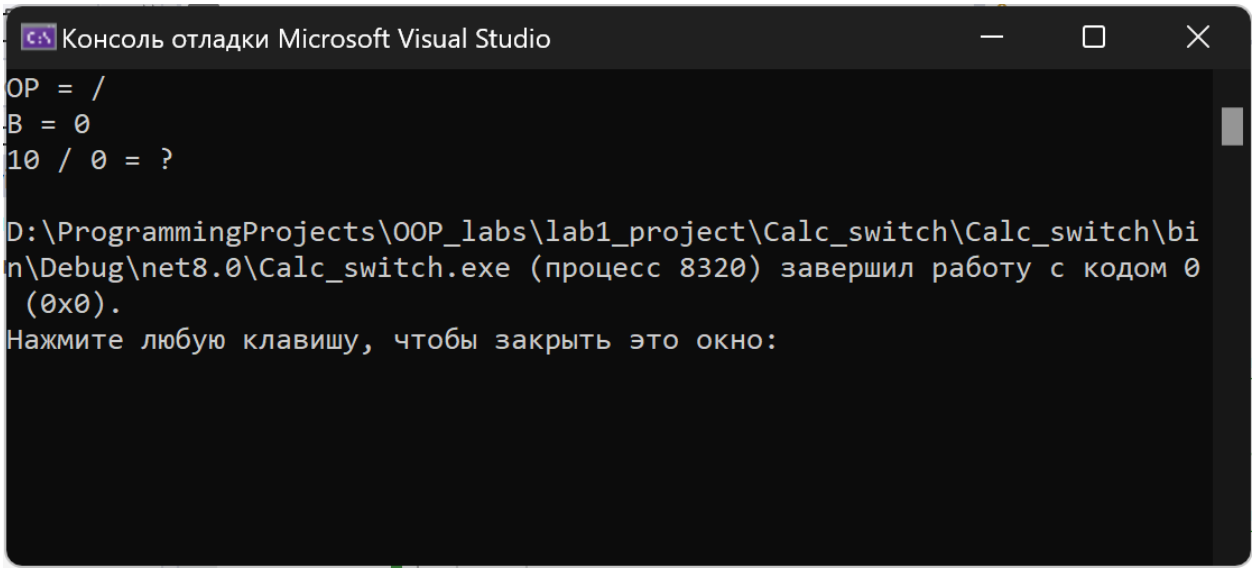


Рисунок 7 — Выполнение программы при делении на ноль

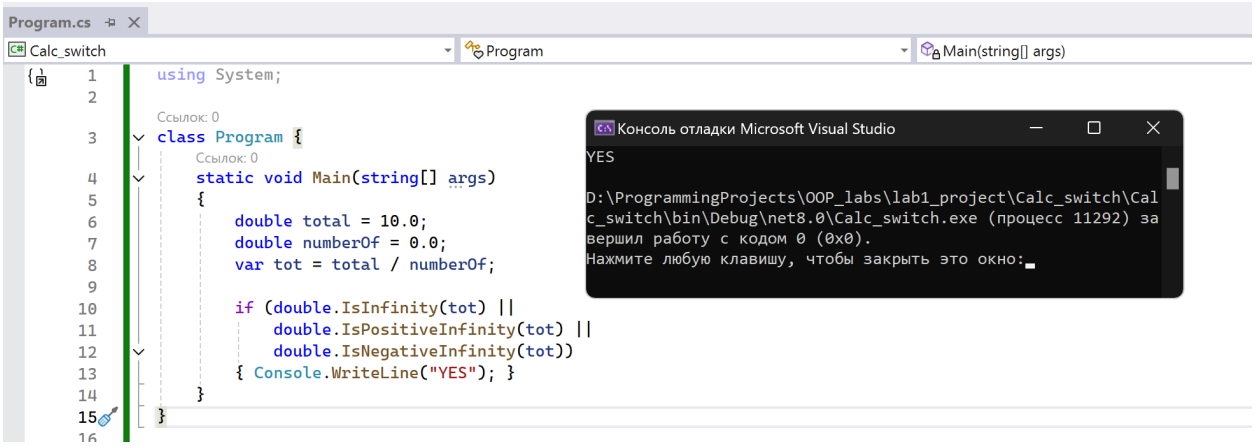


Рисунок 8 — Особое значение при делении на ноль

Пример выполнения при делении нуля на ноль приведен на рисунке 9. Результатом является NaN, поскольку значение выражения $0/0$ не определено.

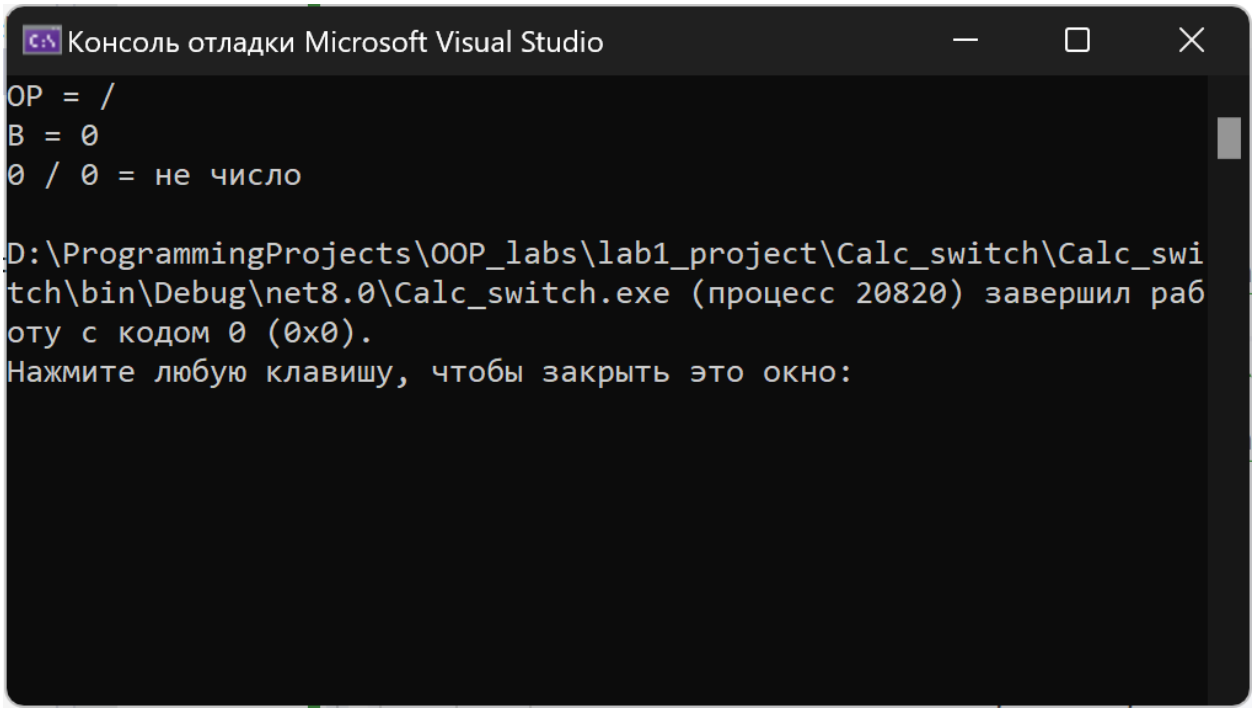


Рисунок 9 — Особое значение при делении нуля на ноль

Пример выполнения программы при некорректном операнде приведен на рисунке 10. Поскольку операнд не был соотнесен ни с одним из вариантов в конструкции switch, то была выполнена операция по умолчанию, которая была задана как присваивание булевой константе *ok* значения *false*. В теле метода есть условие, что при ложном значении этой переменной выводится сообщение об ошибке ввода операнда, что и произошло в примере.

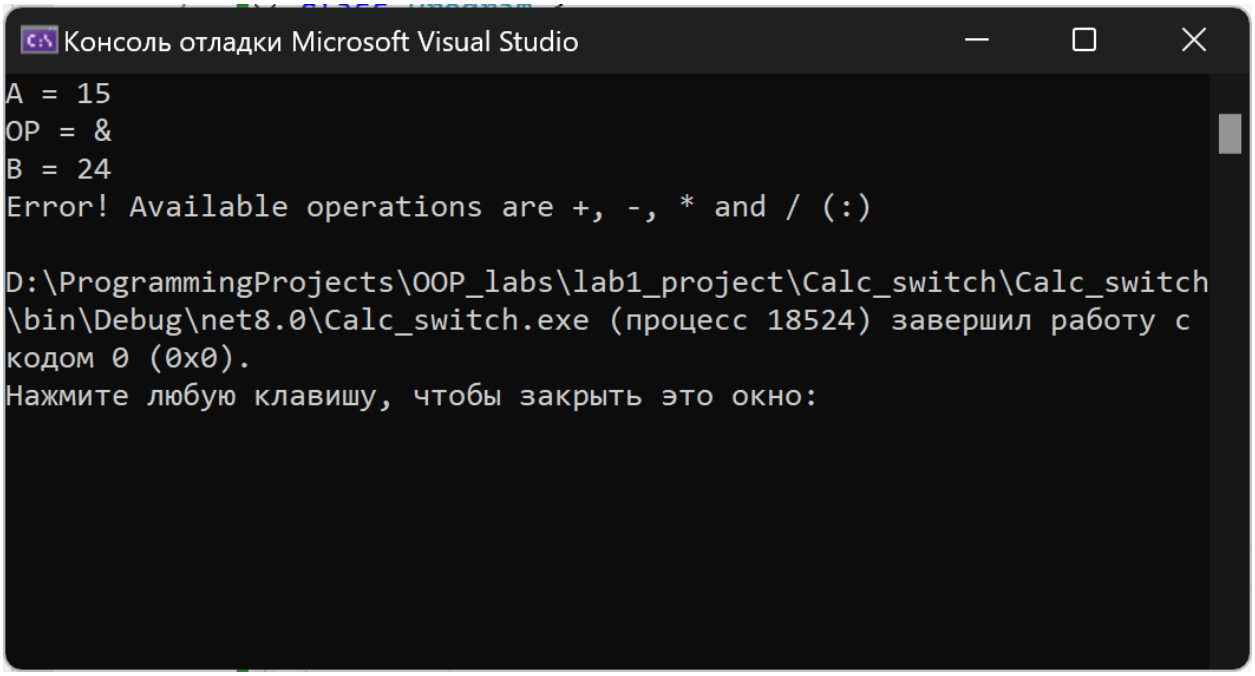


Рисунок 10 — Вывод при неправильном операнде

1.3 Задание 3

Задача: Дано натуральное число. Требуется определить, является ли год с данным номером високосным. Если год является високосным, то выведите YES, иначе выведите NO.

Код программы приведен на рисунке 11. Если введенный год делится на 400 или делится на 4 и не делится на 100, то он високосный, и программа выводит YES, иначе - NO.

```
using System;

Ссылок: 0
class Program {
    Ссылок: 0
    static void Main(string[] args) {
        try
        {
            Console.WriteLine("Enter the year number:");
            ulong year = ulong.Parse(Console.ReadLine());
            if ((year % 400 == 0) || (year % 4 == 0 && year % 100 != 0)) {
                Console.WriteLine("YES");
            } else {
                Console.WriteLine("NO");
            }
        }
        catch (FormatException e) { // если данные в неправильном формате
            Console.WriteLine("Format error:", e.Message);
        }
    }
}
```

Рисунок 11 — Код программы

Результат выполнения программы для 2024, 2023, 2000 и 1900 годов можно увидеть на рисунках 12, 13, 14 и 15.

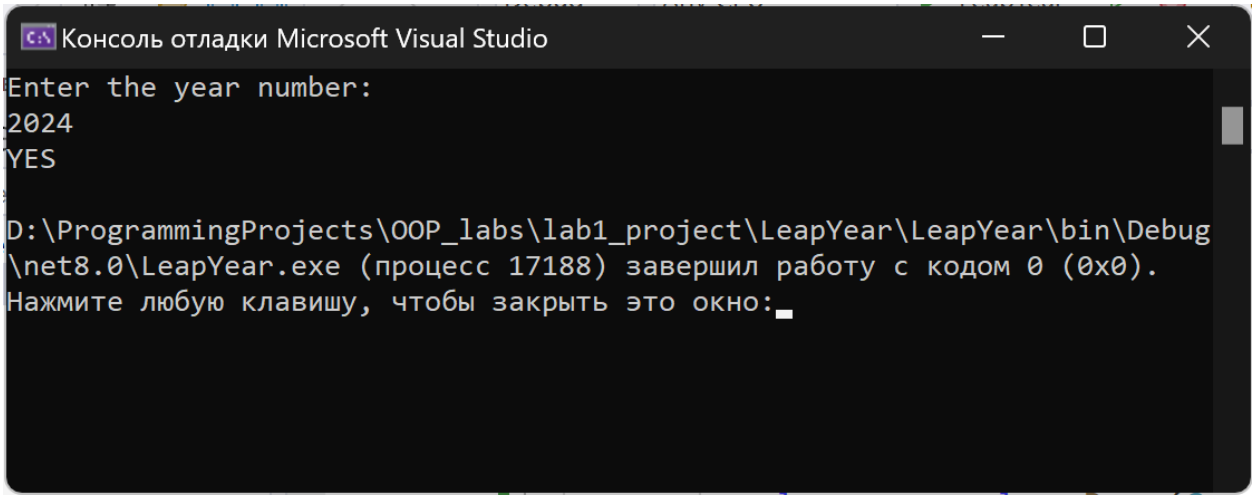


Рисунок 12 — Тест для 2024 года

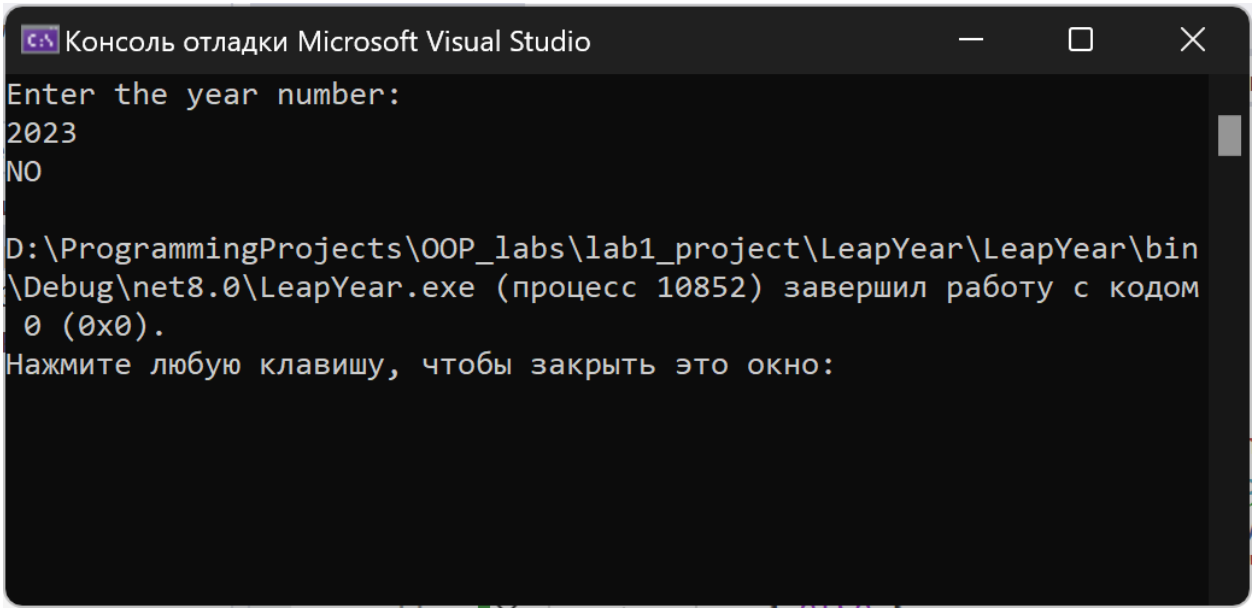


Рисунок 13 — Тест для 2023 года

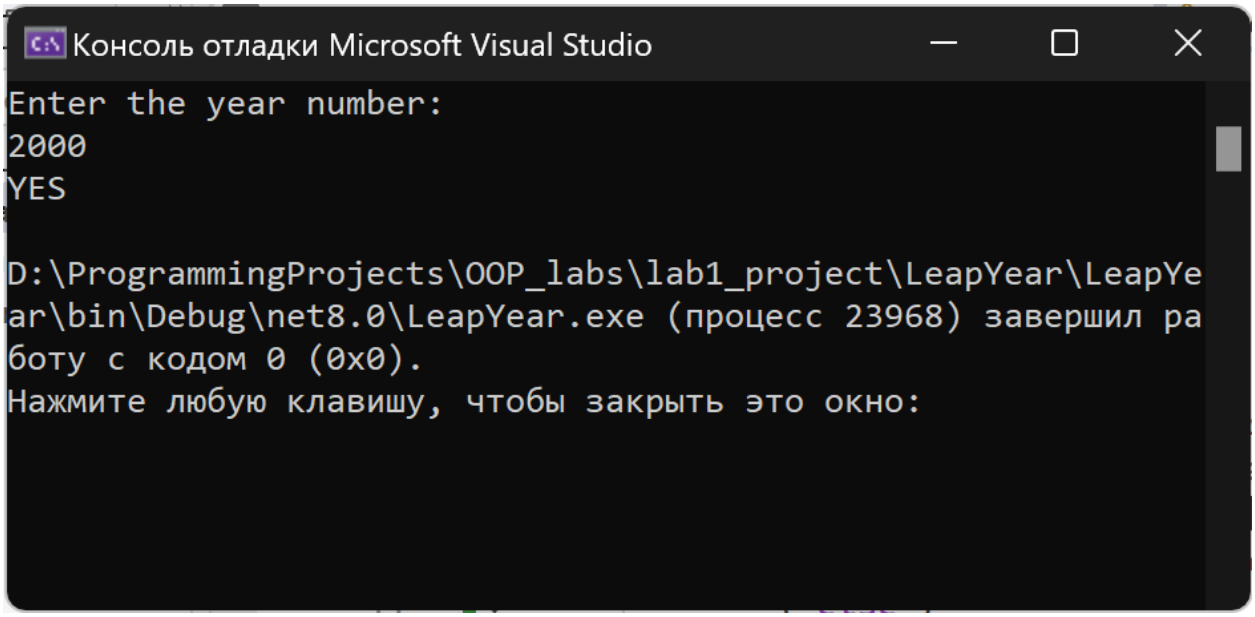


Рисунок 14 — Тест для 2000 года

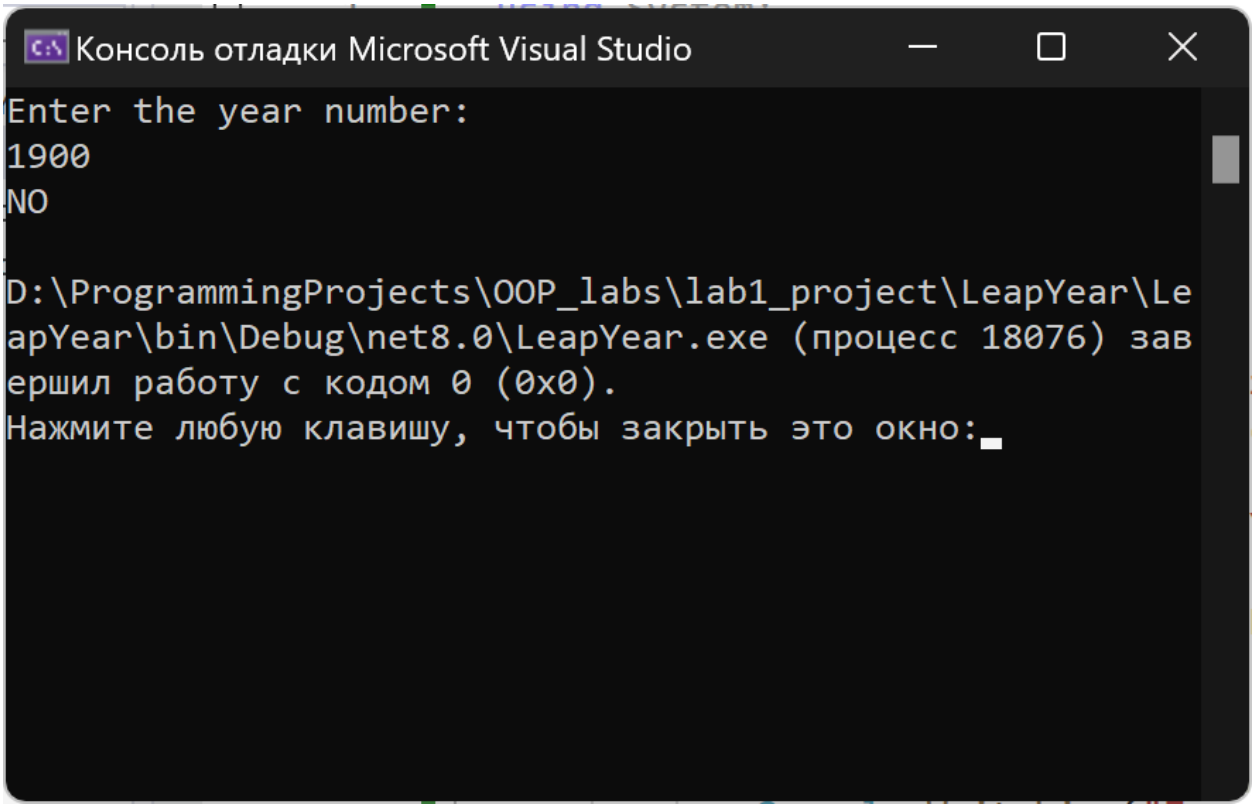


Рисунок 15 — Тест для 1900 года

2 Упражнение 2

2.1 Задание 1

Задача: В этом задании вы напишите программу, выводящую на экран последовательность целых нечетных чисел в строчку через пробел с помощью трех операторов цикла **while**, **do while** и **for**. Реализуйте задачу вывода значений функции с помощью цикла с предусловием, а алгоритм Евклида с постусловием. Сравните варианты реализации задач с помощью разных циклов и сделайте выводы о целесообразности выбора типа цикла.

Код для вывода n нечетных чисел с помощью трех разных циклов приведен на рисунке 16.

```
1   using System;
2
    Ссылок: 0
3   class Program {
        Ссылок: 0
4       public static void Main(string[] args) {
5           Console.Write("n=");
6           int n = 2 * int.Parse(Console.ReadLine());
7           // while cycle
8           Console.WriteLine("Cycle WHILE works:");
9           int i = 1;
10          while (i <= n) {
11              Console.Write(i + " ");
12              i += 2;
13          }
14          // do while cycle
15          Console.WriteLine("\nCycle DO WHILE works:");
16          i = 1;
17          do {
18              Console.Write(i + " ");
19              i += 2;
20          } while (i <= n);
21          // for cycle
22          Console.WriteLine("\nCycle FOR works:");
23          for (i = 1; i <= n; i+= 2)
24          {
25              Console.Write(i + " ");
26          }
27      }
28  }
```

Рисунок 16 — Код программы для вывода нечетных чисел

Результат выполнения для них одинаковый (рисунок 17), за исключением случая $n = 0$, потому что цикл **do while** выведет одно число, а другие циклы не выведут ничего (рисунок 18).

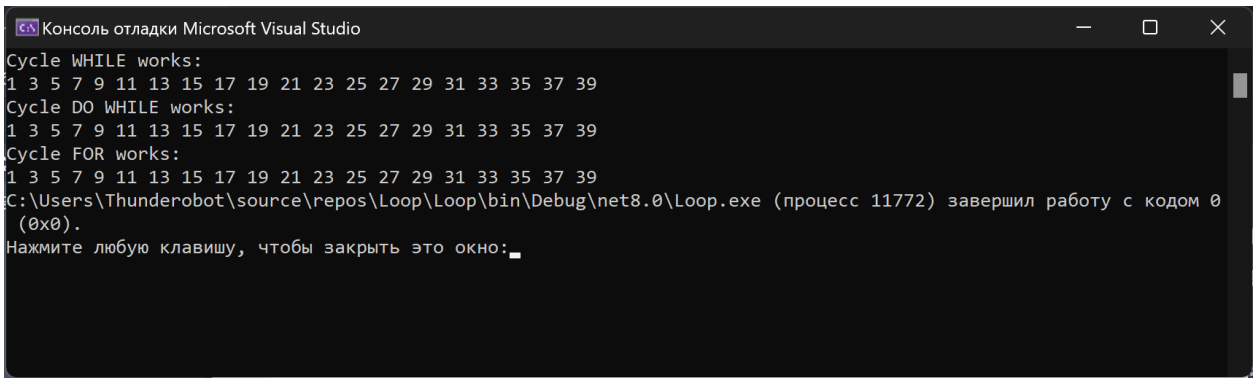


Рисунок 17 — Результат выполнения при $n > 0$

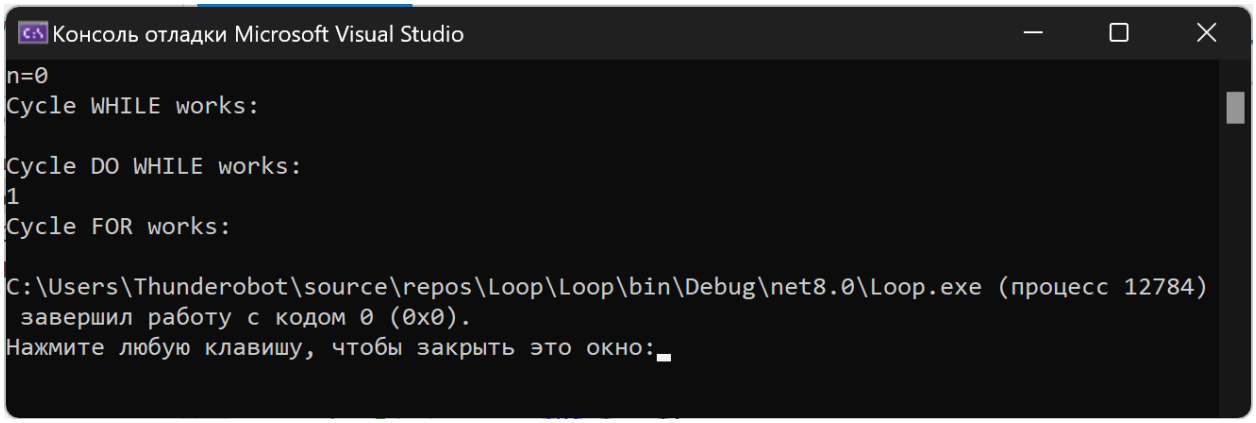


Рисунок 18 — Результат выполнения при $n = 0$

Код программы для вывода значений функции на промежутке приведен на рисунке 19. Реализован через цикл с постусловием.

```
1   using System;
2
    Ссылок: 0
3   class Program
4   {
        Ссылок: 0
5       public static void Main(string[] args)
6       {
7           Console.Write("x1=");
8           double x1 = double.Parse(Console.ReadLine());
9           Console.Write("x2=");
10          double x2 = double.Parse(Console.ReadLine());
11          double x = x1;
12          Console.WriteLine("{0,-10} | {1,-10}", "x", "y");
13          do
14          {
15              double y = Math.Sin(x);
16              Console.WriteLine("{0,-10:f2} | {1,-10:f4}", x, y);
17              x += 0.01;
18          } while (x <= x2);
19      }
20  }
```

Рисунок 19 — Код программы для вывода значений функции на промежутке

Та же программа, но с циклом с предусловием приведена на рисунке 20.

```
Ссылок: 0
3   class Program
4   {
        Ссылок: 0
5       public static void Main(string[] args)
6       {
7           Console.Write("x1=");
8           double x1 = double.Parse(Console.ReadLine());
9           Console.Write("x2=");
10          double x2 = double.Parse(Console.ReadLine());
11          double x = x1;
12          Console.WriteLine("{0,-10} | {1,-10}", "x", "y");
13          while (x <= x2)
14          {
15              double y = Math.Sin(x);
16              Console.WriteLine("{0,-10:f2} | {1,-10:f4}", x, y);
17              x += 0.01;
18          }
19      }
20  }
```

Рисунок 20 — Код программы для вывода значений функции на промежутке

Пример выполнения обеих программ приведен на рисунке 21 и на 22. Разницы в выполнении нет.

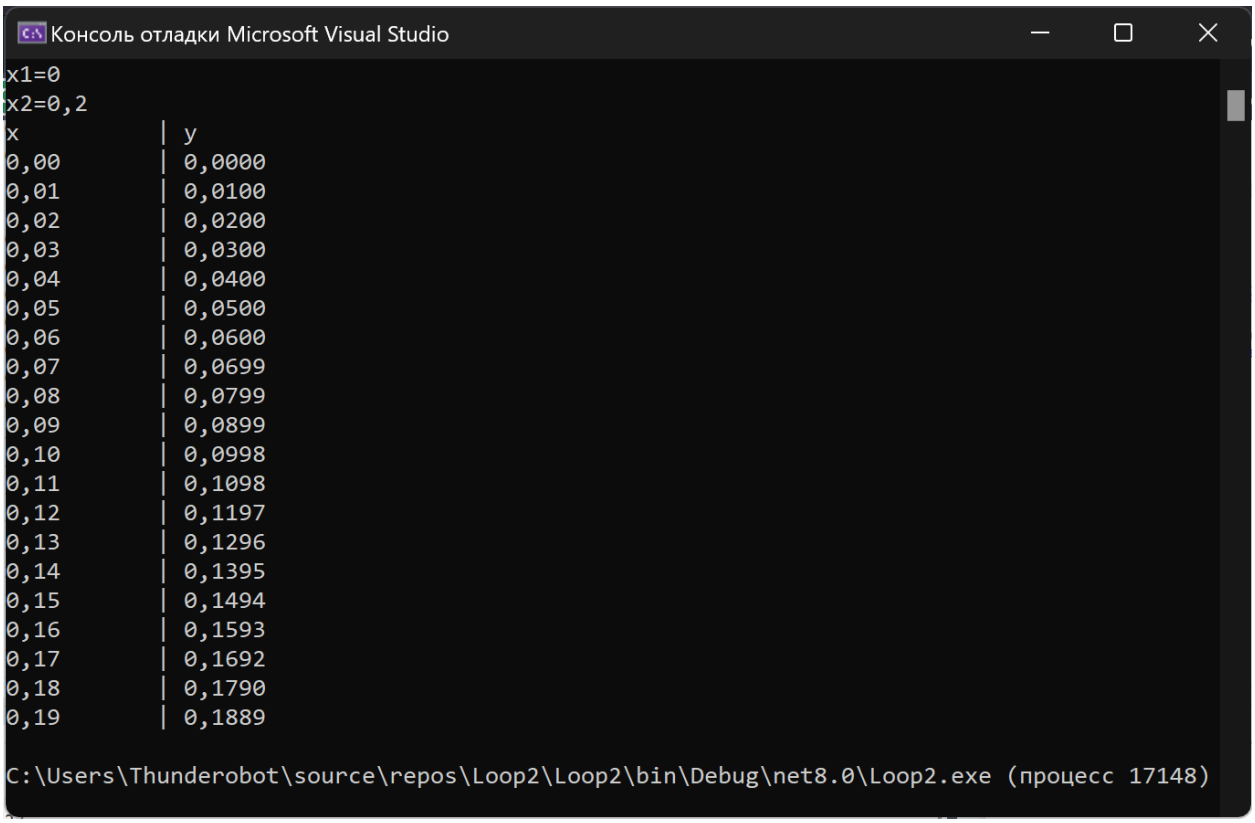


Рисунок 21 — Пример выполнения программы

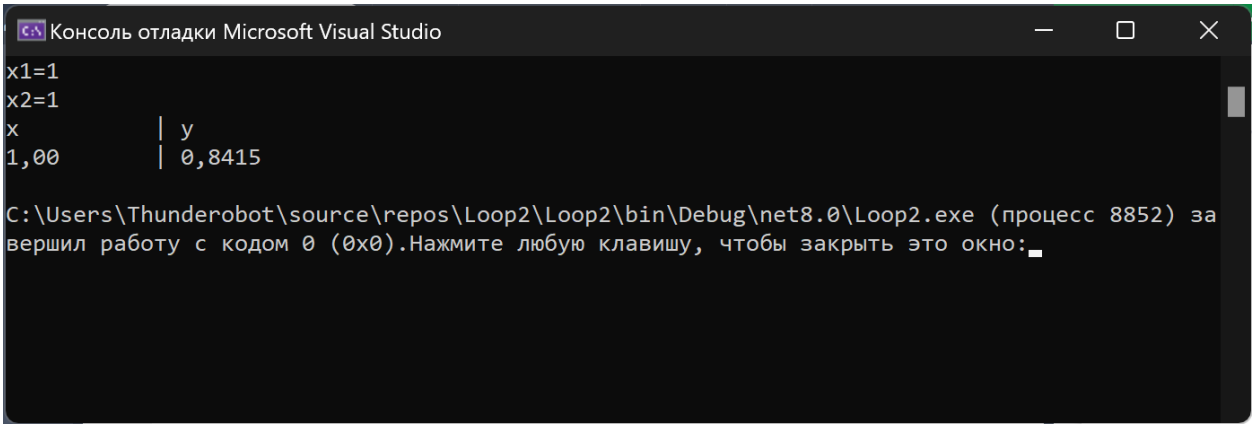


Рисунок 22 — Код программы с постусловием при промежутке из одной точки

Однако если задать $x_2 < x_1$, то программы будут выполнены по-разному. Программа с циклом с постусловием выведет одно значение (23), а вторая программа не выведет ничего (24), что более логично, потому что промежуток задан некорректно, следовательно, программа не должна пытаться обрабатывать значения из него.

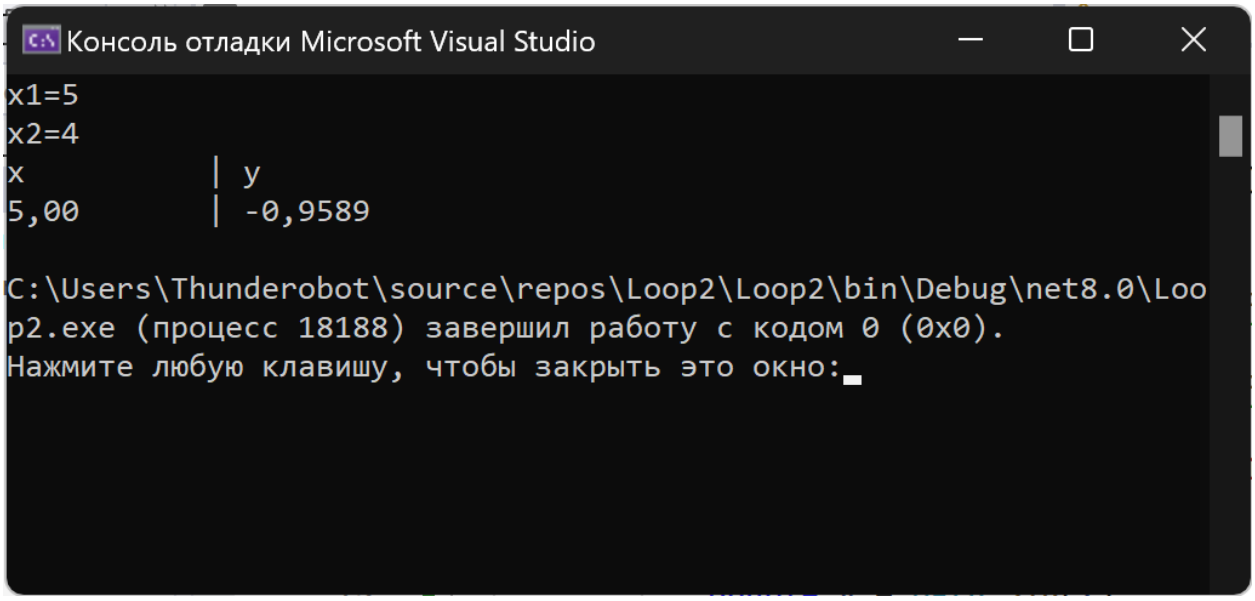


Рисунок 23 — Некорректная работа программы при некорректном промежутке

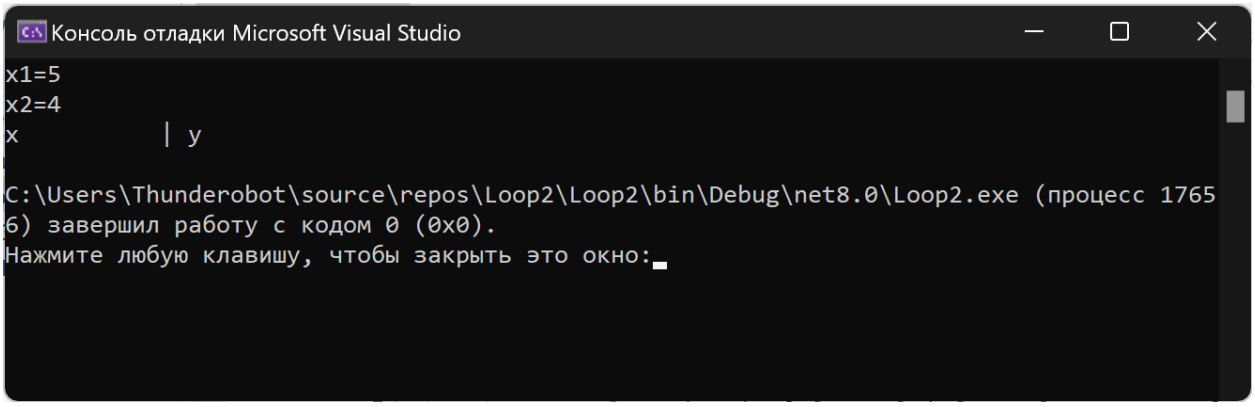
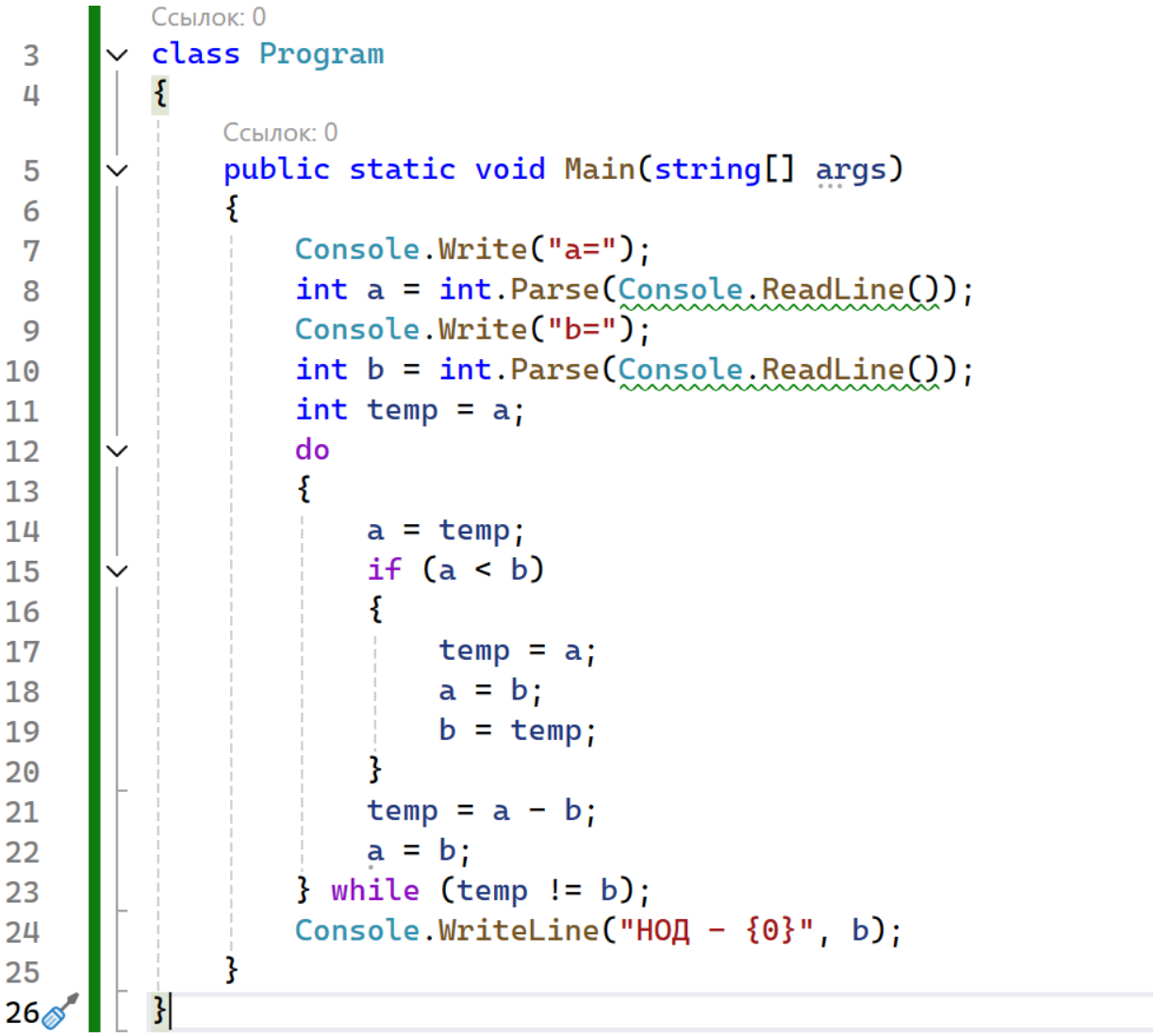


Рисунок 24 — Корректная работа программы при некорректном промежутке

Код программы для реализации алгоритма Евклида с циклом с предусловием приведен на рисунке 25, с постусловием - на 26.

```
1   using System;
2
    Ссылок: 0
3   class Program
4   {
        Ссылок: 0
5       public static void Main(string[] args)
6       {
7           Console.Write("a=");
8           int a = int.Parse(Console.ReadLine());
9           Console.Write("b=");
10          int b = int.Parse(Console.ReadLine());
11          int temp = a;
12          while (temp != b) {
13              a = temp;
14              if (a < b) {
15                  temp = a;
16                  a = b;
17                  b = temp;
18              }
19              temp = a - b;
20              a = b;
21          }
22          Console.WriteLine("НОД - {0}", b);
23      }
24  }
```

Рисунок 25 — Алгоритм Евклида с циклом с предусловием



```
Ссылок: 0
class Program
{
    Ссылок: 0
    public static void Main(string[] args)
    {
        Console.Write("a=");
        int a = int.Parse(Console.ReadLine());
        Console.Write("b=");
        int b = int.Parse(Console.ReadLine());
        int temp = a;
        do
        {
            a = temp;
            if (a < b)
            {
                temp = a;
                a = b;
                b = temp;
            }
            temp = a - b;
            a = b;
        } while (temp != b);
        Console.WriteLine("НОД - {0}", b);
    }
}
```

Рисунок 26 — Алгоритм Евклида с циклом с постусловием

Пример выполнения обеих программ приведен на рисунке 27.

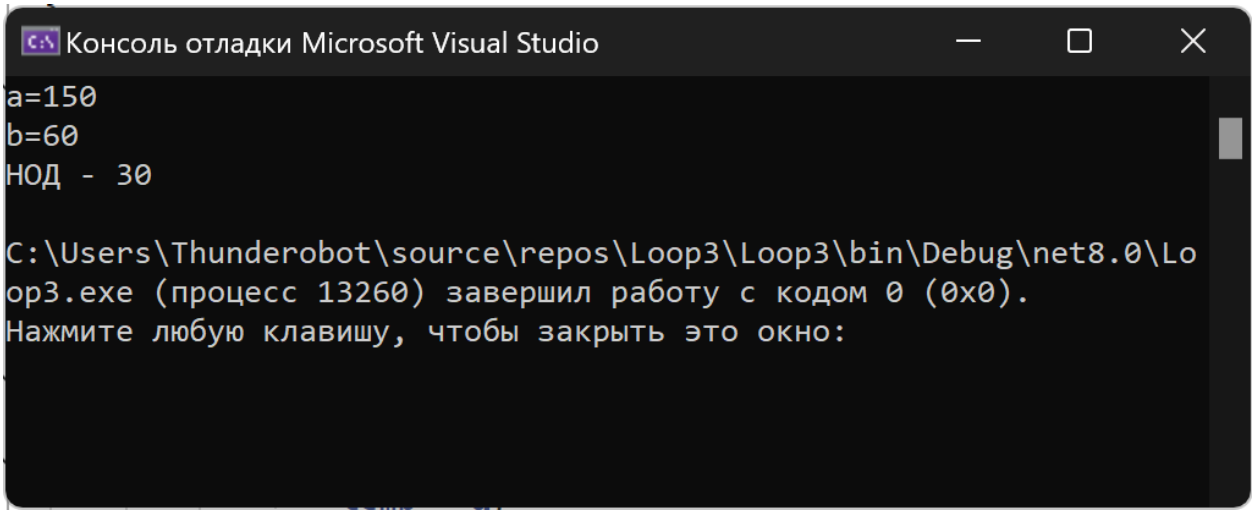


Рисунок 27 — Пример выполнения алгоритма Евклида

Разница между ними заключается при $a = b$. Тогда программа с циклом с постусловием никогда не выполнится, а первая программа сразу даст правильный ответ.

Разница между циклами **while** и **do while** состоит в следующем. Первый цикл используется тогда, когда его тело может не выполниться, а условие является более важным на начальном этапе выполнения, нежели тело цикла, или когда есть зависимость от начальных данных. Например, посылание запросов в базу данных (если соединение не установлено, произойдет ошибка, соответственно, сначала надо проверить наличие подключения). Второй цикл используется тогда, когда тело цикла необходимо выполнить хотя бы один раз, а последующие разы уже в зависимости от выполненности условия. Например, обработка игрового цикла.

2.2 Задание 2

Задача: В этом задании составьте программу, реализующую сумму $\sum_{i=1}^{100} i$ ($i \in [1; k] \cup [m; 100]$, $i \in \mathbb{Z}$).

Код программы приведен на рисунке 28. Использован оператор перехода continue, а также реализовано исключением для выпадающих из диапазона k и m.

```
using System;

Ссылок: 0
class Program {
    Ссылок: 0
    static void Main(string[] args) {
        try {
            Console.Write("k=");
            int k = int.Parse(Console.ReadLine());
            Console.Write("m=");
            int m = int.Parse(Console.ReadLine());
            // если k и m не в нужном диапазоне
            if (!(1 <= k && k <= m && m <= 100)) { throw new ArithmeticException(); }
            int sum = 0;
            for (int i = 1; i <= 100; i++) {
                if (i > k && i < m) continue;
                sum+=i;
            }
            Console.WriteLine("Сумма равна {0}", sum);
        }
        catch (ArithmeticException e)
        {
            Console.WriteLine("Числа k и m должны удовлетворять условию 1 <= k <= m <= 100");
        }
    }
}
```

Рисунок 28 — Код программы

Результат выполнения для корректных и некорректных данных приведен на рисунках 29 и 30.

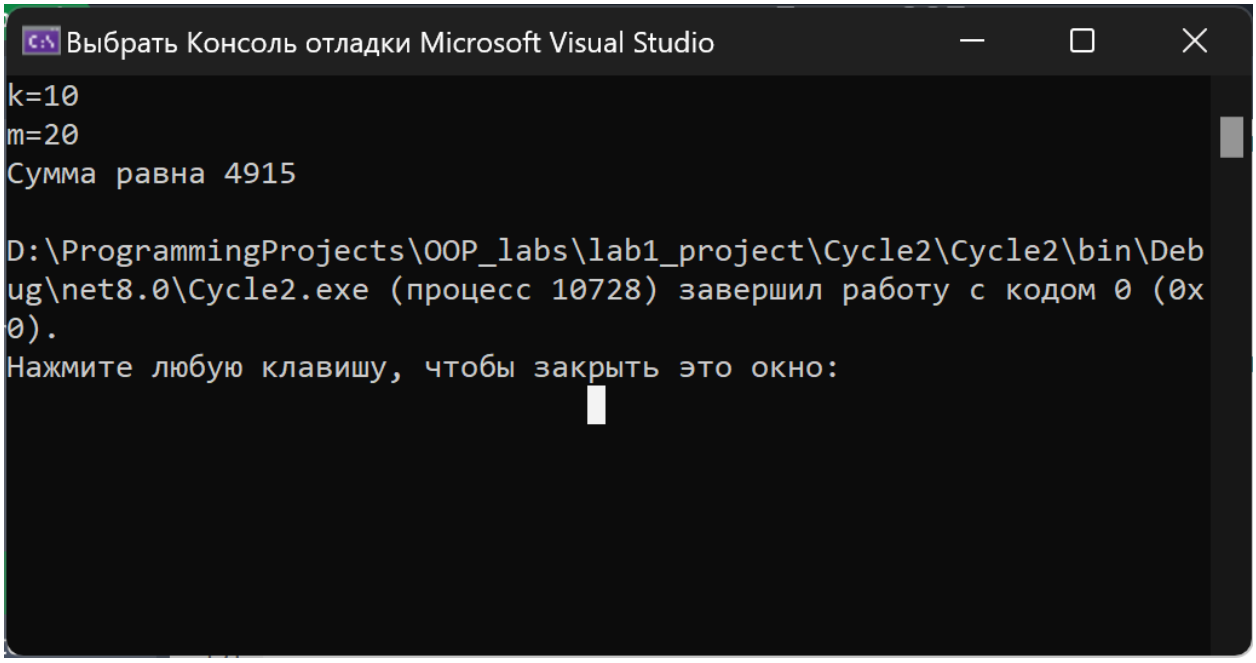


Рисунок 29 — Тест на корректных данных

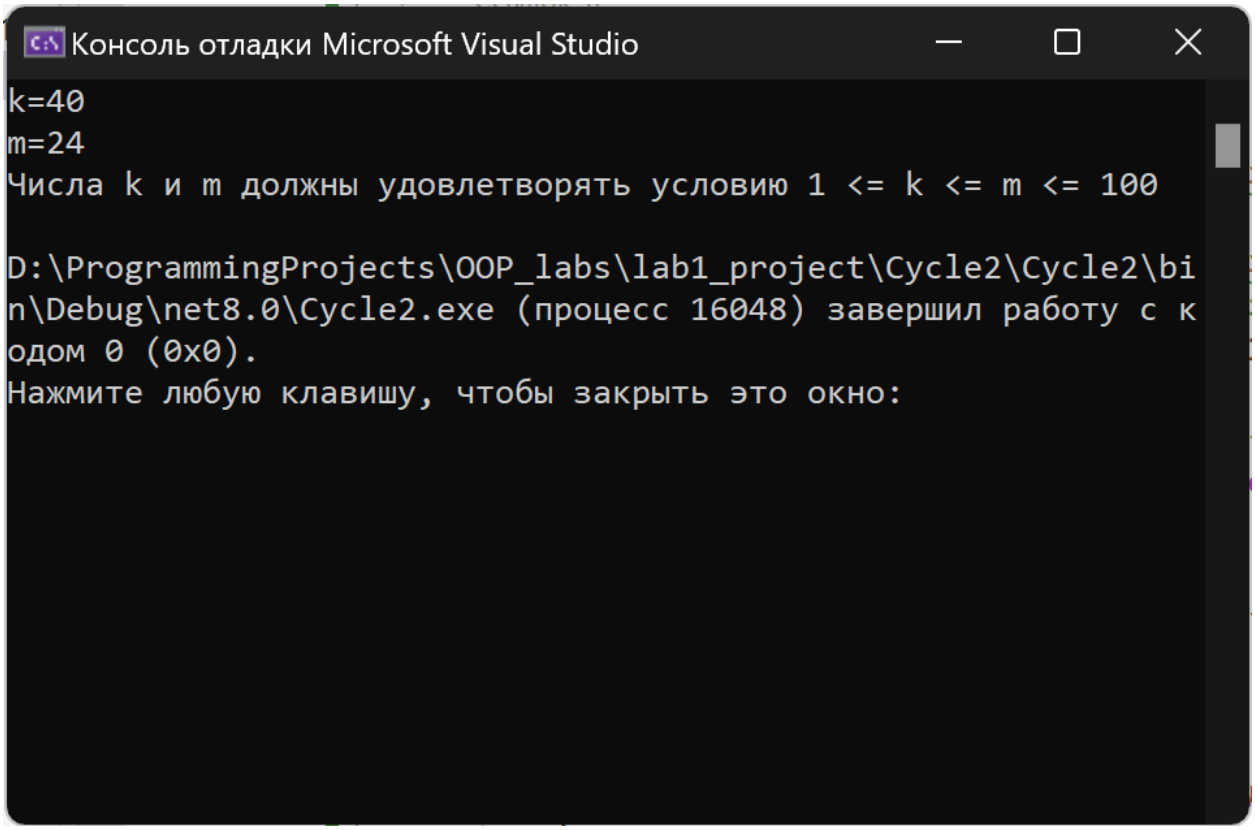


Рисунок 30 — Тест на некорректных данных

2.3 Задание 3

Задача: В этом задании разработайте программу, имитирующую стрельбу по мишени. Реализуйте следующую функциональность:

- Пользователь вводит данные о выстреле в виде пары чисел – координат x и y несколько раз.
- Повтор ввода следует организовать в цикле. После «стрельбы» пользователю выводится информация о сумме очков.

Для решения задачи была реализована структура для точки в декартовой системе координат (рисунок 31). Ее поля - координаты x и y. Также добавлен оператор вычитания, который используется в дальнейшем для вычисления расстояния между двумя точками.

```
Ссылок: 10
public struct Point {
    // Класс точки в декартовой системе
    public double x, y;
    Ссылок: 1
    public static Point operator -(Point a, Point b)
    {
        return new Point
        {
            x = a.x - b.x,
            y = a.y - b.y
        };
    }
}
```

Рисунок 31 — Код структуры точки в ДСК

Затем был создан класс для получения случайного числа, по модулю не превосходящего заданного. Реализован с использованием класса **Random**. Применяется для задания положения мишени и смещения координат при выстреле. Код показан на рисунке 32.

```
Ссылок: 4
public class RandomNumber {
    // Получение случайного числа, по модулю не превосходящего заданного
    Ссылок: 4
    public static double Get(int max_abs_value)
    {
        var rnd = new Random();
        return Math.Sign(rnd.Next(-1, 1)) * rnd.Next(max_abs_value + 1);
    }
}
```

Рисунок 32 — Код класса RandomNumber

В основном классе программы был создан метод для получения расстояния между двумя точками - тцентром мишени и точкой, вводимой игроком (рисунок 33). В методе запрашиваются координаты точки для выстрела, затем происходит случайное смещение координат, после чего при использовании оператора структуры находится расстояние между двумя точками.

```
Ссылок: 1
static double GetDistance(Point target) {
    // Получение точки от игрока и подсчет квадрата расстояния до
    // центра мишени
    Console.Write("x=");
    string command = Console.ReadLine();
    if (command == "exit") throw new ArgumentException();
    double x = double.Parse(command);
    Console.Write("y=");
    double y = double.Parse(Console.ReadLine());
    Point user = new Point { x = x + RandomNumber.Get(1), y = y + RandomNumber.Get(1) };
    Point diff = user - target;
    return diff.x * diff.x + diff.y * diff.y;
}
```

Рисунок 33 — Код метода GetDistance

Код метода - точки входа в программу - показан на рисунке 34. В этом методе создется объект структуры Point со случайными координатами, который будет играть роль центра мишени. Затем в бесконечном цикле программа запрашивает точку для выстрела, и в зависимости от результата попадания выводит соответствующее сообщения. Для выхода из цикла надо написать "exit".

```
27  class Program
28  {
        Ссылок: 0
29      static void Main(string[] args)
30      {
31          Point target = new Point { x = RandomNumber.Get(10), y = RandomNumber.Get(10) };
32          int total_sum = 0;
33          Console.WriteLine("Type \'exit\' to quit the game");
34          while (true)
35          {
36              try
37              {
38
39                  double dist = Program.GetDistance(target);
40                  if (dist > 9) Console.WriteLine("Не попал! Сумма очков - {0}", total_sum);
41                  else
42                  {
43                      if (dist <= 1) total_sum += 10;
44                      else if (dist <= 4) total_sum += 5;
45                      else total_sum += 1;
46                      Console.WriteLine("Попал! Сумма очков - {0}", total_sum);
47                  }
48              }
49              catch (ArgumentException) { break; }
50              catch (Exception e)
51              {
52                  Console.WriteLine("An error occuried: {0}", e.Message);
53                  continue;
54              }
55
56          }
57          Console.WriteLine("Сумма очков за игру - {0}", total_sum);
58      }
```

Рисунок 34 — Код метода Main

Пример выполнения программы приведен на рисунке 35.

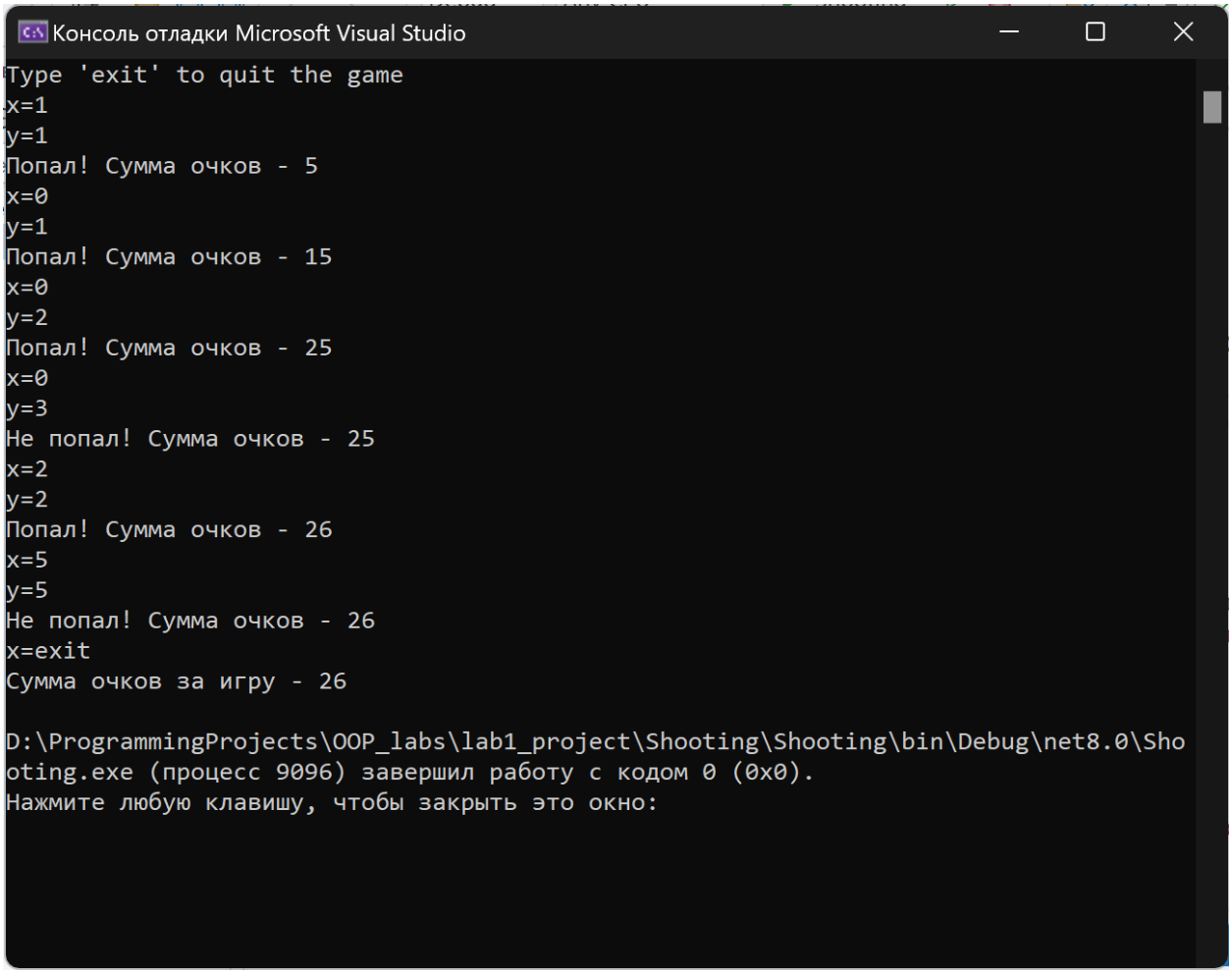


Рисунок 35 — Пример выполнения программы

ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы были выполнены все требуемые упражнения. Цель работы достигнута. Получены навыки использования управляющих конструкций для организации вычислений.