

Министерство науки и высшего образования Российской Федерации

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

ITMO University

ЛАБОРАТОРНАЯ РАБОТА №3

По дисциплине Объектно-ориентированное программирование

Тема работы Использование выражений

Обучающийся Крестьянова Елизавета Федоровна

Факультет факультет инфокоммуникационных технологий

Группа K3223

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. Реализация операторов выбора.....	4
1.1 Задание №1. Применение конструкции if-else-if.....	4
1.2 Задание №2. Применение оператора switch	7
1.3 Задание №3. Определение високосного года	10
2 Упражнение №2. Реализация циклов при работе с данными размерных типов	13
2.1 Задание 1. Использование операторов цикла while, do while и for.....	13
2.2 Задание 2. Расчет суммы, используя операторы перехода ..	19
2.3 Задание 3. Стрельба по мишени.....	20
ЗАКЛЮЧЕНИЕ	24

ВВЕДЕНИЕ

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

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

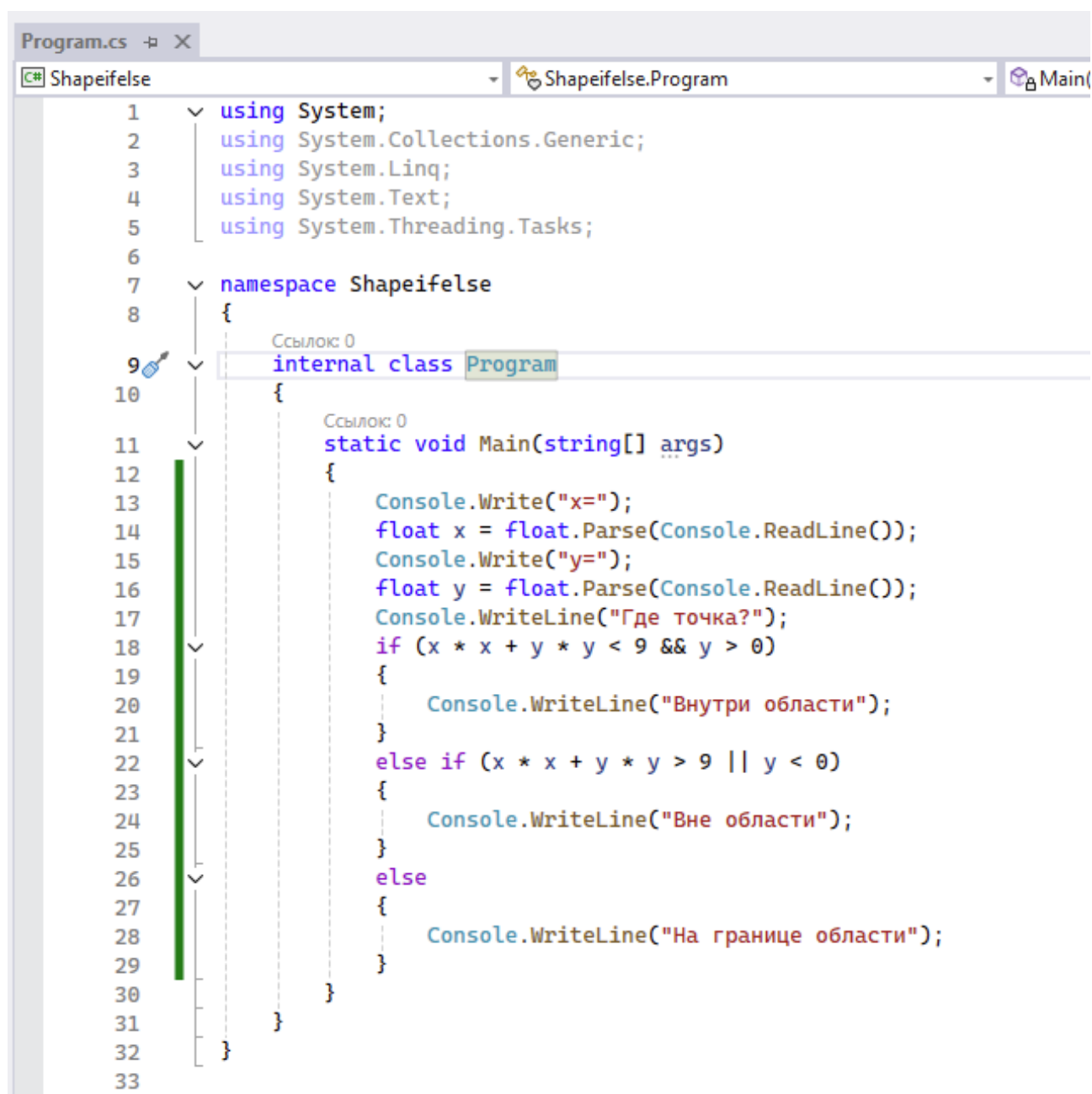
1 УПРАЖНЕНИЕ №1. РЕАЛИЗАЦИЯ ОПЕРАТОРОВ ВЫБОРА

1.1 Задание №1. Применение конструкции if-else-if

В этом задании программа должна выдать одно из сообщений "Внутри", "Вне" или "На Границе" в зависимости от того, лежит ли точка, чьи координаты ввёл пользователь, в полукруге с радиусом 3.

Был создан проект Shapeifelse.sln и в нём реализован ввод координат точки пользователем. Проверка на попадание внутри области выполняется строкой «*if (x * x + y * y < 9 && y > 0)*», на попадание вне «*else if (x * x + y * y > 9 || y > 0)*». Если координаты не прошли обе проверки, это значит, что точка находится на границе.

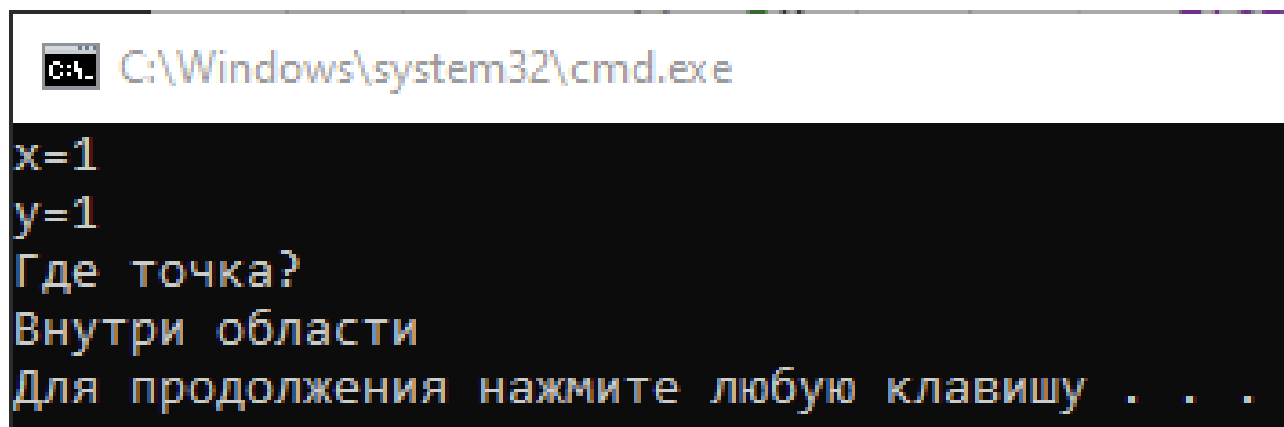
Код программы представлен на рисунке 1.1.



```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Shapeifelse
8  {
9      internal class Program
10     {
11         static void Main(string[] args)
12         {
13             Console.Write("x=");
14             float x = float.Parse(Console.ReadLine());
15             Console.Write("y=");
16             float y = float.Parse(Console.ReadLine());
17             Console.WriteLine("Где точка?");
18             if (x * x + y * y < 9 && y > 0)
19             {
20                 Console.WriteLine("Внутри области");
21             }
22             else if (x * x + y * y > 9 || y < 0)
23             {
24                 Console.WriteLine("Вне области");
25             }
26             else
27             {
28                 Console.WriteLine("На границе области");
29             }
30         }
31     }
32 }
33
```

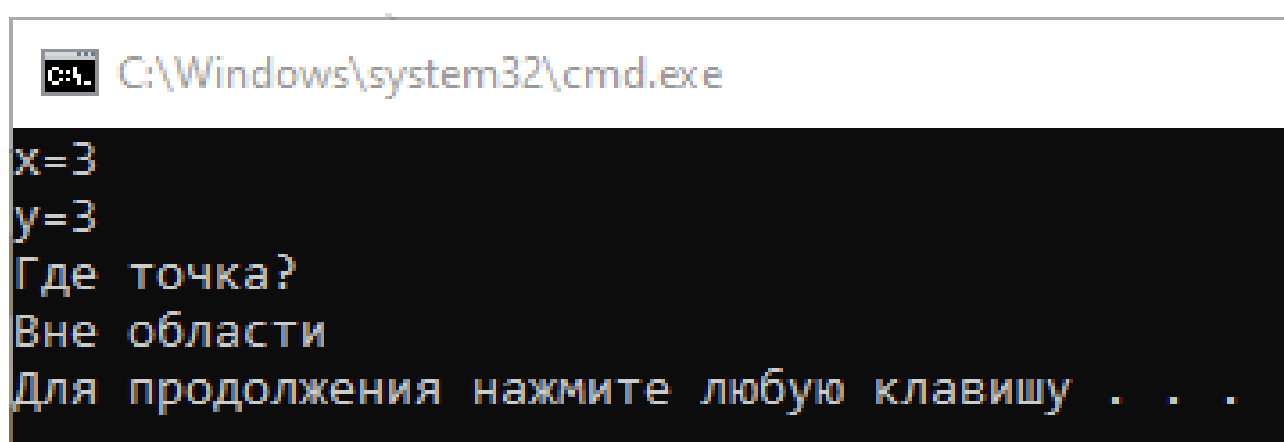
Рисунок 1.1 — Упр №1: Код проверки на попадание точки

Вывод программы с тремя разными сообщениями можно увидеть на рисунках



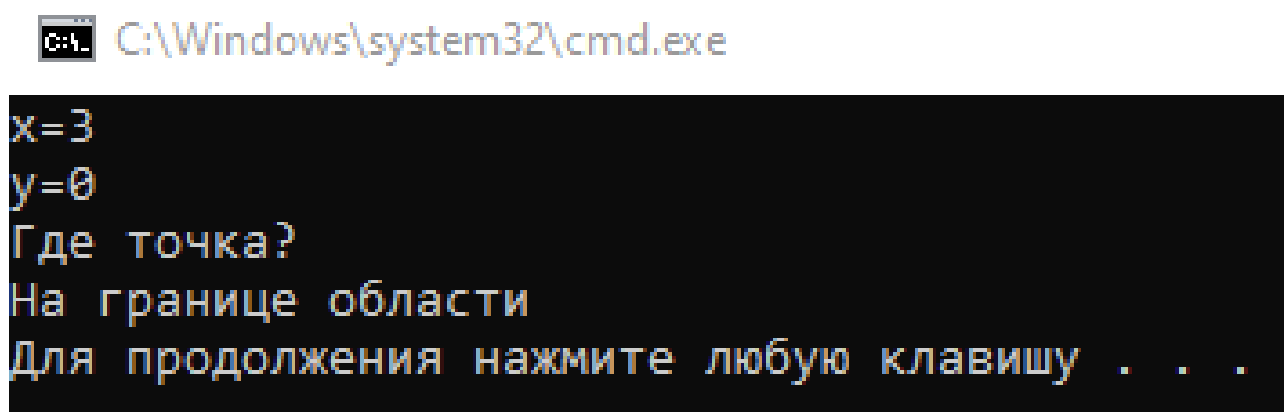
```
C:\Windows\system32\cmd.exe
x=1
y=1
Где точка?
Внутри области
Для продолжения нажмите любую клавишу . . .
```

Рисунок 1.2 — Упр №1: Точка внутри области



```
C:\Windows\system32\cmd.exe
x=3
y=3
Где точка?
Вне области
Для продолжения нажмите любую клавишу . . .
```

Рисунок 1.3 — Упр №1: Точка вне области



```
C:\Windows\system32\cmd.exe
x=3
y=0
Где точка?
На границе области
Для продолжения нажмите любую клавишу . . .
```

Рисунок 1.4 — Упр №1: Точка на границе области

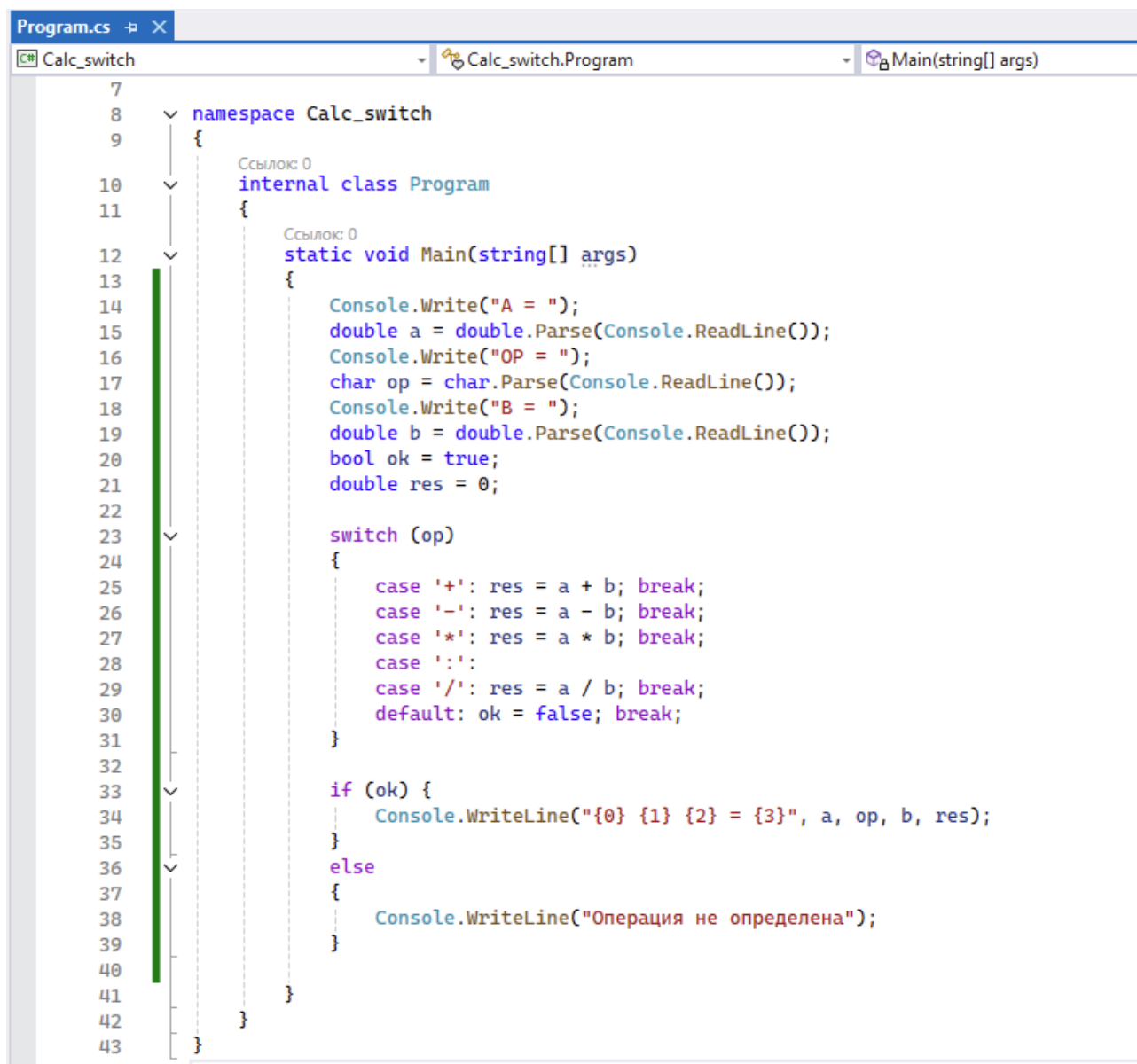
1.2 Задание №2. Применение оператора switch

Для выполнения этого задания будет создан калькулятор. Пользователь должен будет ввести первый операнд, требуемую операцию, и затем второй операнд.

В программе было прописано принятие ввода пользователя для переменных операндов и операции. С помощью оператора switch определяется тип вычисления, и полученный результат записывается в переменную `res`. Если в переменной операции стоит знак, который не был задан в `switch`, в переменную `ok` записывается `false`.

Проверяется переменная `ok` - если она ложна, в консоль пишется ошибка. Иначе выдаётся полученный результат.

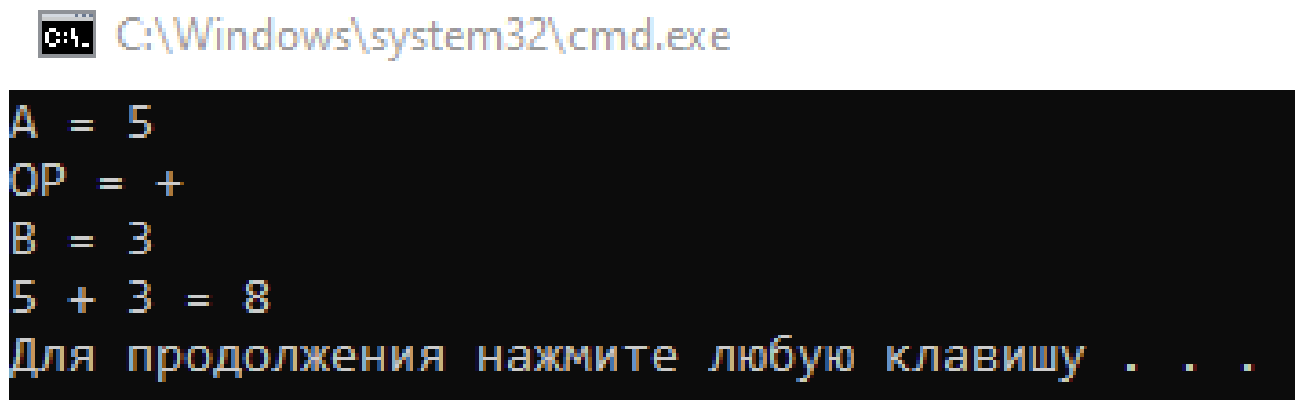
Код программы можно увидеть на рисунке 1.5.



```
7
8 namespace Calc_switch
9 {
10     internal class Program
11     {
12         static void Main(string[] args)
13         {
14             Console.Write("A = ");
15             double a = double.Parse(Console.ReadLine());
16             Console.Write("OP = ");
17             char op = char.Parse(Console.ReadLine());
18             Console.Write("B = ");
19             double b = double.Parse(Console.ReadLine());
20             bool ok = true;
21             double res = 0;
22
23             switch (op)
24             {
25                 case '+': res = a + b; break;
26                 case '-': res = a - b; break;
27                 case '*': res = a * b; break;
28                 case ':':
29                 case '/': res = a / b; break;
30                 default: ok = false; break;
31             }
32
33             if (ok) {
34                 Console.WriteLine("{0} {1} {2} = {3}", a, op, b, res);
35             }
36             else
37             {
38                 Console.WriteLine("Операция не определена");
39             }
40         }
41     }
42 }
43
```

Рисунок 1.5 — Упр №1: Код калькулятора

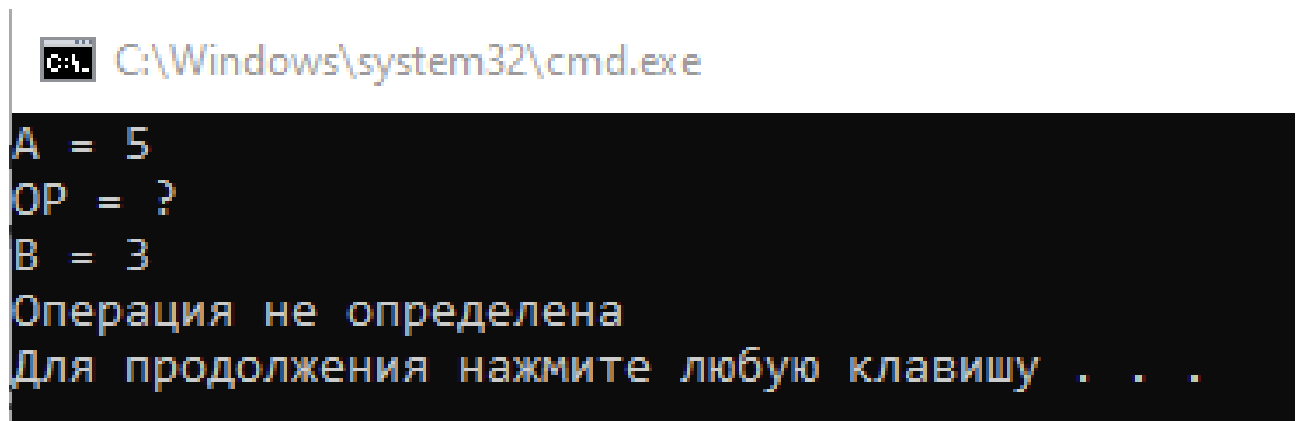
Успешная работа программы представлена на рисунке 1.6.



```
C:\Windows\system32\cmd.exe
A = 5
OP = +
B = 3
5 + 3 = 8
Для продолжения нажмите любую клавишу . . .
```

Рисунок 1.6 — Упр №1: Сложение чисел

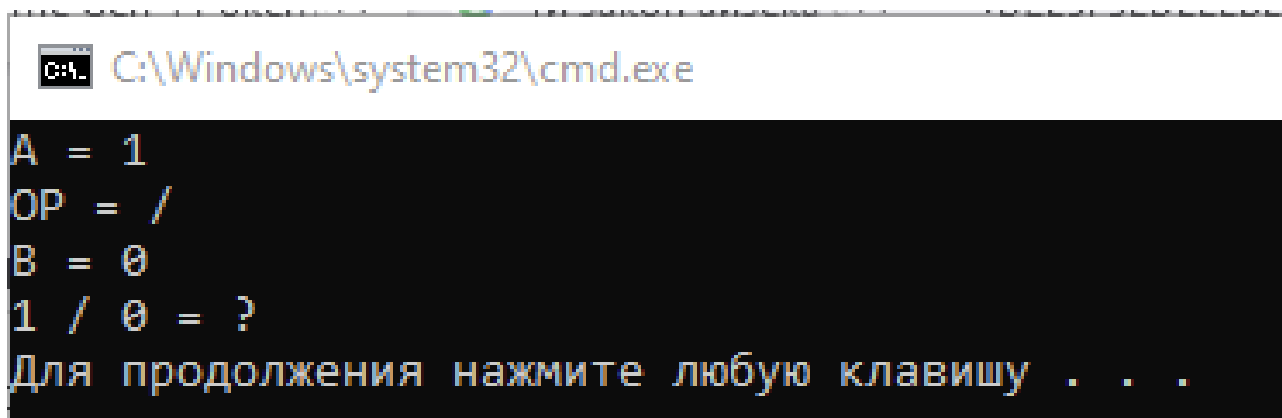
Неверно заданная операция пользователем приводит к результату, который можно увидеть на рисунке 1.7.



```
C:\Windows\system32\cmd.exe
A = 5
OP = ?
B = 3
Операция не определена
Для продолжения нажмите любую клавишу . . .
```

Рисунок 1.7 — Упр №1: Неверно заданная операция

При попытке поделить какое-либо число на ноль, в результат записывается знак бесконечности, который отображается как вопросительный знак в консоли. Его можно увидеть на рисунке 1.8.

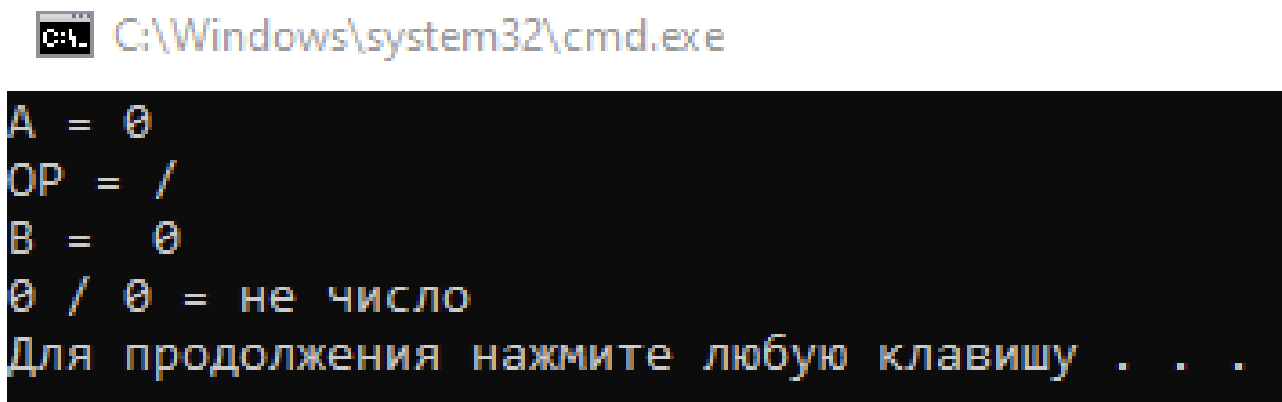


```
C:\Windows\system32\cmd.exe
A = 1
OP = /
B = 0
1 / 0 = ?
Для продолжения нажмите любую клавишу . . .
```

Рисунок 1.8 — Упр №1: Деление числа на ноль

Этот результат объясняется тем, что, во-первых, тип переменных `double` позволяет содержать значение бесконечности. Во-вторых, тип `double` не предназначен для крайне точных вычислений. Когда 1 делится на 0, на самом деле 1 делится на какое-то число, крайне близкое к нулю. Поэтому выводится бесконечность.

При делении 0 на 0 команда выводит ответ «не число», ведь в математике «0/0» не определено. Этот ответ можно увидеть на рисунке 1.9.



```
C:\Windows\system32\cmd.exe
A = 0
OP = /
B = 0
0 / 0 = не число
Для продолжения нажмите любую клавишу . . .
```

Рисунок 1.9 — Упр №1: Деление нуля на ноль

1.3 Задание №3. Определение високосного года

В разработанной программе необходимо выяснить, является ли введённый год високосным по его номеру.

Год считается високосным, если его номер кратен 4, но не кратен 100, либо кратен 400.

В новом проекте «`leap_year`» была указана переменная булева типа «`is_leap_year`». Если введённое пользователем число проходит проверку «`year % 4 == 0 && year % 100 != 0 || year % 400 == 0`», в консоль выводится сообщение «Этот год високосный!», иначе выводится «Этот год не високосный...».

Код программы можно увидеть на рисунке 1.10.

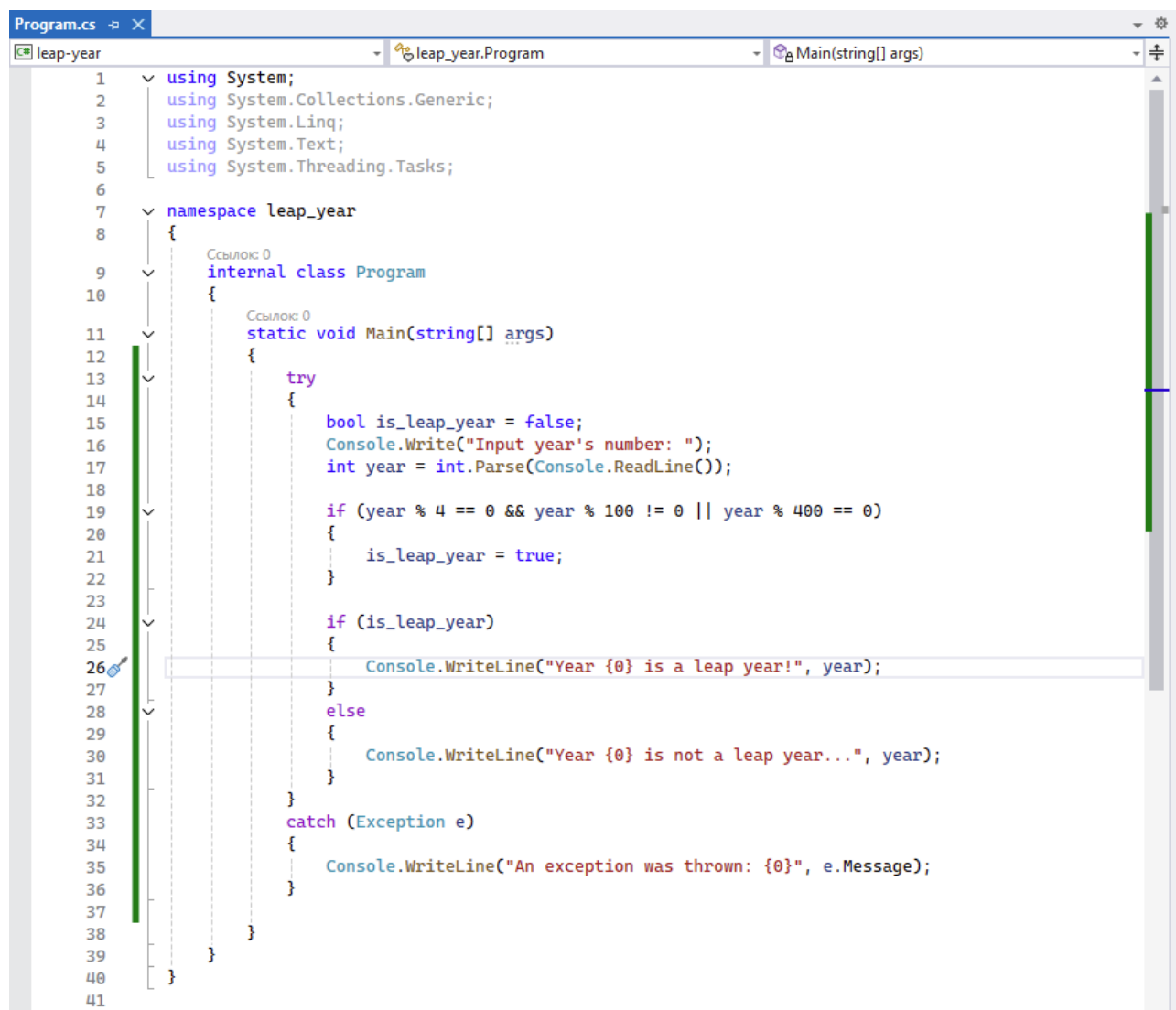


Рисунок 1.10 — Упр №1: Определение високосного года

Различные примеры работы программы показаны на рисунках 1.11, 1.12 и 1.13.

 C:\Windows\system32\cmd.exe

```
Input year's number: 2024
Year 2024 is a leap year!
Для продолжения нажмите любую клавишу . . .
```

Рисунок 1.11 — Упр №1: 2024 год

 C:\Windows\system32\cmd.exe

```
Input year's number: 1700
Year 1700 is not a leap year...
Для продолжения нажмите любую клавишу . . .
```

Рисунок 1.12 — Упр №1: 1700 год

 C:\Windows\system32\cmd.exe

```
Input year's number: 2000
Year 2000 is a leap year!
Для продолжения нажмите любую клавишу . . .
```

Рисунок 1.13 — Упр №1: 2000 год

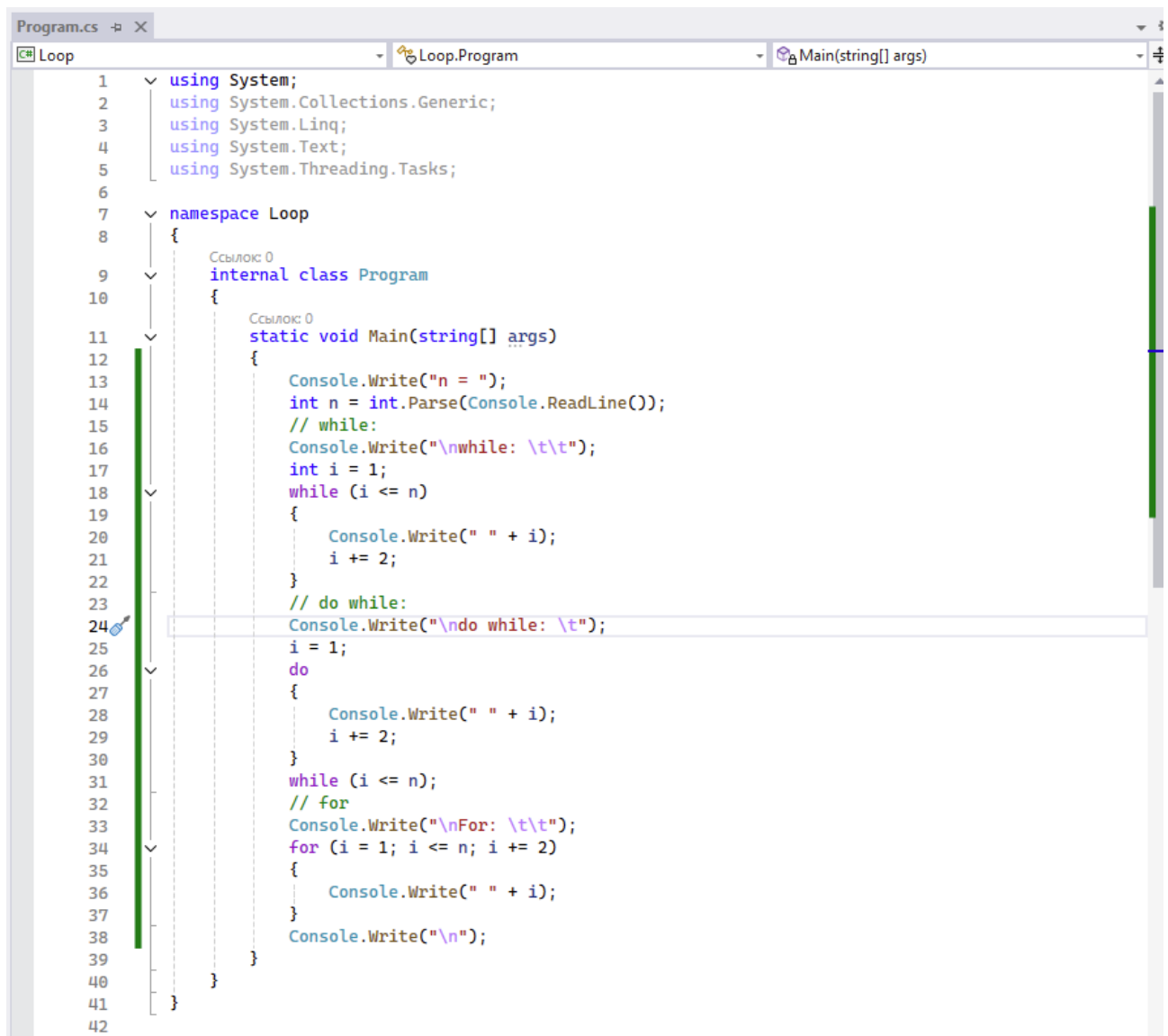
2 УПРАЖНЕНИЕ №2. РЕАЛИЗАЦИЯ ЦИКЛОВ ПРИ РАБОТЕ С ДАННЫМИ РАЗМЕРНЫХ ТИПОВ

2.1 Задание 1. Использование операторов цикла while, do while и for

В этом задании необходимо написать программы, выводящую на экран последовательность целых нечетных чисел в строчку.

Был создан проект Loop, в методе Main было прописано принятие ввода пользователя, отвечающего за количество чисел.

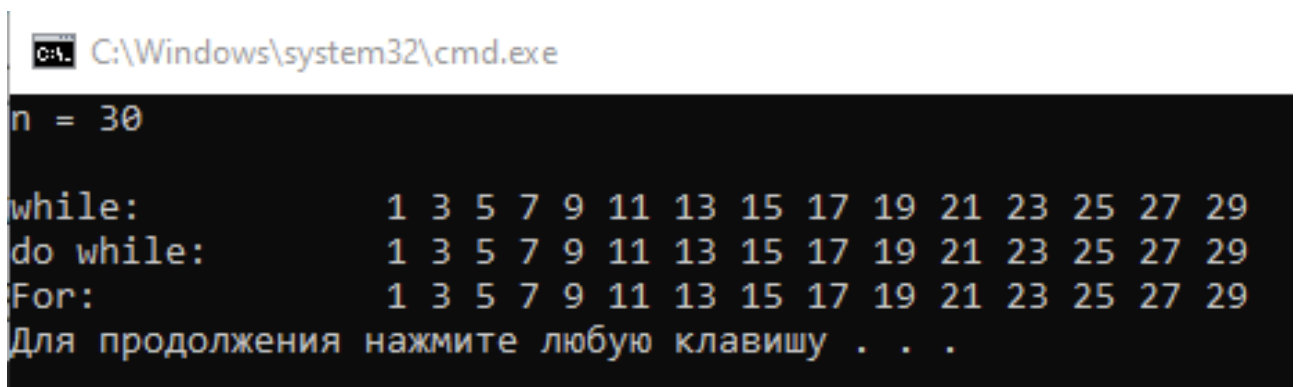
Вывод последовательности чисел был реализован тремя разными операторами цикла: while, do while и for. Эту реализацию можно посмотреть на рисунке 2.1.



```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Loop
8  {
9      Ссылка: 0
10     internal class Program
11     {
12         Ссылка: 0
13         static void Main(string[] args)
14         {
15             Console.Write("n = ");
16             int n = int.Parse(Console.ReadLine());
17             // while:
18             Console.Write("\nwhile: \t\t");
19             int i = 1;
20             while (i <= n)
21             {
22                 Console.Write(" " + i);
23                 i += 2;
24             }
25             // do while:
26             Console.Write("\ndo while: \t");
27             i = 1;
28             do
29             {
30                 Console.Write(" " + i);
31                 i += 2;
32             }
33             while (i <= n);
34             // for
35             Console.Write("\nFor: \t\t");
36             for (i = 1; i <= n; i += 2)
37             {
38                 Console.Write(" " + i);
39             }
40             Console.Write("\n");
41         }
42     }
43 }
```

Рисунок 2.1 — Упр №2: Код программы, выводящий три последовательности чисел

С выводом программы можно ознакомиться на рисунке 2.2.



```
C:\Windows\system32\cmd.exe
n = 30
while:      1 3 5 7 9 11 13 15 17 19 21 23 25 27 29
do while:   1 3 5 7 9 11 13 15 17 19 21 23 25 27 29
For:        1 3 5 7 9 11 13 15 17 19 21 23 25 27 29
Для продолжения нажмите любую клавишу . . .
```

Рисунок 2.2 — Упр №2: Вывод трёх последовательностей нечетных чисел

В этом же проекте были реализованы циклы с постусловием и предусловием. Цикл с постусловием помог реализовать вывод значений функции $\sin(x)$ на заданном интервале, а циклом с предусловием был реализован алгоритм Евклида.

Эти два алгоритма представлены на рисунке 2.3.

```

}
Console.WriteLine("\n");

// Таблица синусов, постусловие

double x, y;
Console.Write("x1 = ");
double x1 = double.Parse(Console.ReadLine());
Console.Write("x2 = ");
double x2 = double.Parse(Console.ReadLine());

x = x1;
Console.WriteLine("x \t sin(x) \t");
do
{
    y = Math.Sin(x);
    Console.WriteLine("{0} \t {1:F3}", x, y);
    x = x + 0.01;
}
while (x <= x2);

Console.WriteLine("\n");

// Алгоритм Евклида, предусловие
Console.Write("a = ");
int a = int.Parse(Console.ReadLine());
Console.Write("b = ");
int b = int.Parse(Console.ReadLine());
int temp;
temp = a;
while (temp != b)
{
    a = temp;
    if (a < b)
    {
        temp = a;
        a = b;
        b = temp;
    }
    temp = a - b;
    a = b;
}
Console.WriteLine("НОД заданных чисел = {0}", a);
}

```

Рисунок 2.3 — Упр №2: Постусловие синуса, предусловие Евклида

Результаты их выполнения представлены на рисунке 2.4.


```
x1 = 1,1
x2 = 1,2
x      sin(x)
1,1    0,891
1,11    0,896
1,12    0,900
1,13    0,904
1,14    0,909
1,15    0,913
1,16    0,917
1,17    0,921
1,18    0,925
1,19    0,928

a = 30
b = 18
НОД заданных чисел = 6
Для продолжения нажмите любую клавишу . . .
```

Рисунок 2.4 — Упр №2: Результаты постусловия синуса, предусловие Евклида

Поменяли циклы местами, что видно на рисунке 2.5.

```

85 // Таблица синусов, предусловие
86
87 Console.Write("x1 = ");
88 x1 = double.Parse(Console.ReadLine());
89 Console.Write("x2 = ");
90 x2 = double.Parse(Console.ReadLine());
91
92 x = x1;
93 Console.WriteLine("x \t sin(x) \t");
94 while (x <= x2)
95 {
96     y = Math.Sin(x);
97     Console.WriteLine("{0} \t {1:F3}", x, y);
98     x = x + 0.01;
99 }
100
101 Console.WriteLine("\n");
102
103 // Алгоритм Евклида, постусловие
104 Console.Write("a = ");
105 a = int.Parse(Console.ReadLine());
106 Console.Write("b = ");
107 b = int.Parse(Console.ReadLine());
108 temp = a;
109 do
110 {
111     a = temp;
112     if (a < b)
113     {
114         temp = a;
115         a = b;
116         b = temp;
117     }
118     temp = a - b;
119     a = b;
120 }
121 while (temp != b);
122 Console.WriteLine("НОД заданных чисел = {0}", a);
123 }
124 }
125 }

```

Рисунок 2.5 — Упр №2: Предусловие синуса, постусловие Евклида

При данных из рисунка 2.4 результат остаётся тем же, но некоторые данные приводят к разным выводам.

Если в функции синусов задать первую границу больше второй, то в цикле с постусловием программа выведет первое значение синуса левой границы, и только потом проверит условие и остановит цикл. В цикле с предусловием бы ничего в таблицу не вывелось.

В алгоритме Евклида цикл с постусловием некорректно себя ведёт при равных числах a , b . Допустим, пользователь задал числа 1 и 1. Если цикл с предусловием сразу их проверяет и останавливается, цикл с постусловием всё равно всегда делает хотя бы один шаг. К концу этого шага переменные a , b , $temp$ получают следующие значения: 1, 1, 0. Так как 0 не равно 1, цикл некорректно продолжается. На самом деле, он не остановится: из переменной $temp$ будет продолжать вычитаться переменная b .

Постусловие стоит использовать, когда нам необходимо совершить хотя бы один шаг. Если нам нужно убедиться, что все шаги отвечают условию, то тогда следует использовать цикл с предусловием.

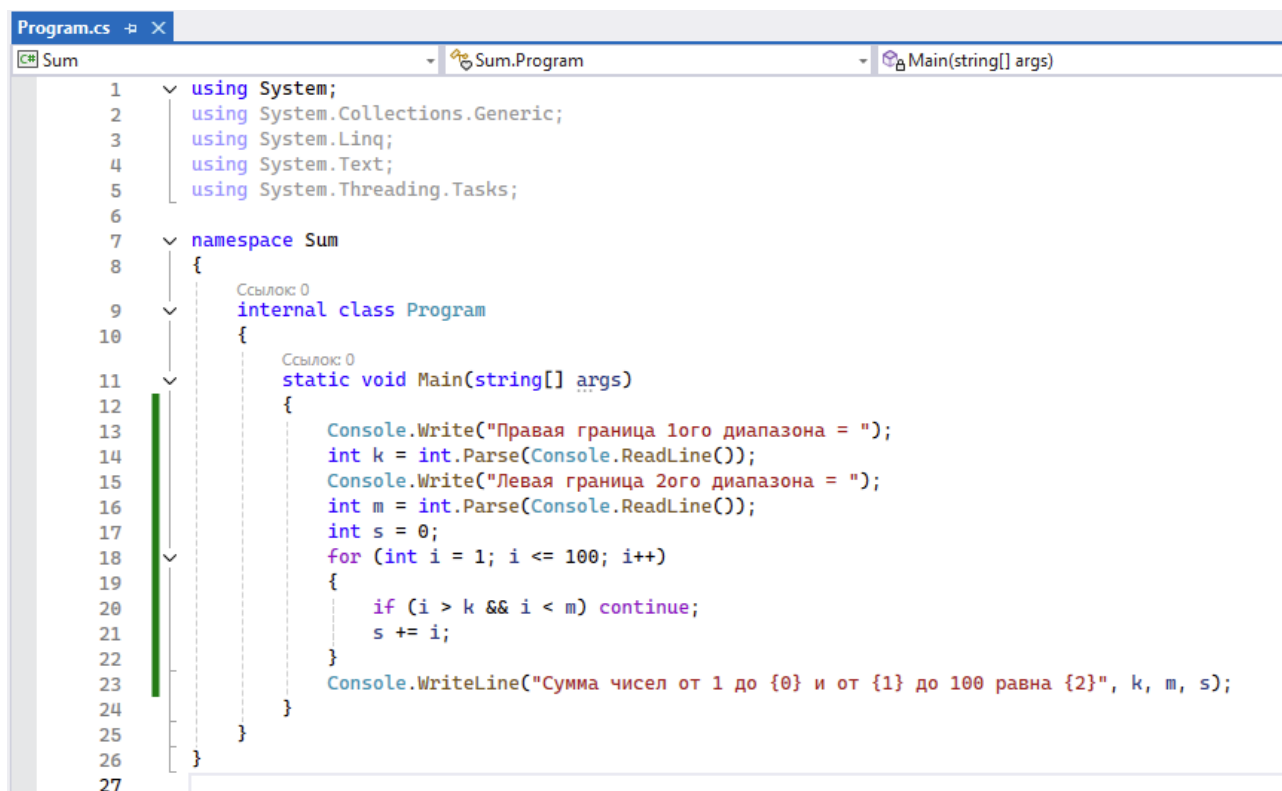
2.2 Задание 2. Расчет суммы, используя операторы перехода

В данном задании необходимо реализовать сумму $s = \sum_{i=1}^{100} i$ для i , находящихся на диапазонах от 1 до k и от m до 100.

Был создан новый проект Sum, в методе Main было написано принятие ввода пользователя для целочисленных переменных k и m .

С помощью цикла `for` программа добавляет к переменной s растущую переменную i . Если i не попадает в заданные диапазоны, то цикл продолжается оператором `continue`.

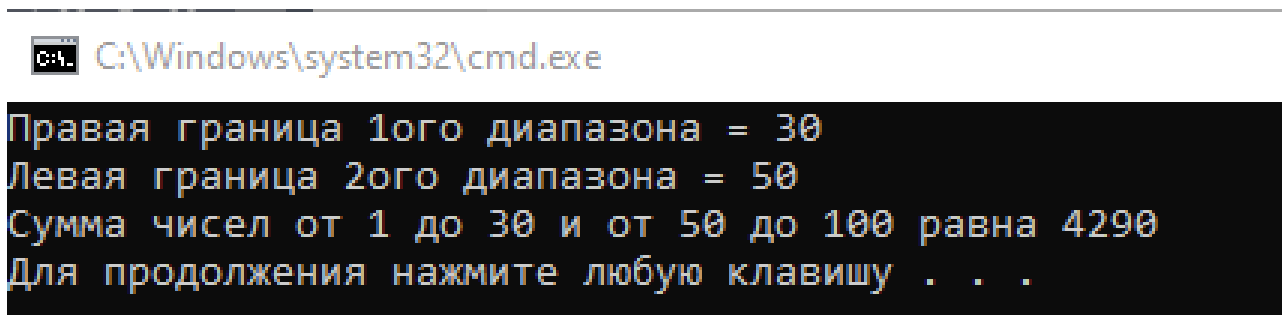
Код программы представлен на рисунке 2.6.



```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace Sum
8 {
9     internal class Program
10     {
11         static void Main(string[] args)
12         {
13             Console.Write("Правая граница 1ого диапазона = ");
14             int k = int.Parse(Console.ReadLine());
15             Console.Write("Левая граница 2ого диапазона = ");
16             int m = int.Parse(Console.ReadLine());
17             int s = 0;
18             for (int i = 1; i <= 100; i++)
19             {
20                 if (i > k && i < m) continue;
21                 s += i;
22             }
23             Console.WriteLine("Сумма чисел от 1 до {0} и от {1} до 100 равна {2}", k, m, s);
24         }
25     }
26 }
27
```

Рисунок 2.6 — Упр №2: Цикл for подсчёта суммы

Её вариант выполнения показан на рисунке 2.7.



```
C:\Windows\system32\cmd.exe
Правая граница 1ого диапазона = 30
Левая граница 2ого диапазона = 50
Сумма чисел от 1 до 30 и от 50 до 100 равна 4290
Для продолжения нажмите любую клавишу . . .
```

Рисунок 2.7 — Упр №2: Подсчёт суммы

2.3 Задание 3. Стрельба по мишени

В этом задании необходимо разработать программу, имитирующую стрельбу по мишени, в которой будет следующая функциональность:

- пользователь вводит данные о выстреле в виде пары чисел несколько раз;
- Вывод информации о сумме очков;

- 2 варианта мишени;
- случайное значение местоположения центра мишени;
- помехи при стрельбе.

Был создан новый проект Target. В нём была создана структура Точка, которой задаются координаты x и y.

Программа много раз просит ввод пользователя: сначала она просит выбрать тип мишени и количество выстрелов. Программа генерирует случайное местоположение мишени на квадрате от $[-1, -1]$ до $[1, 1]$. В версии программы, показанной в этом отчёте, в консоль выводится чёткое местоположение мишени для тестирования. Реализацию этих функций можно увидеть на рисунке 2.8.

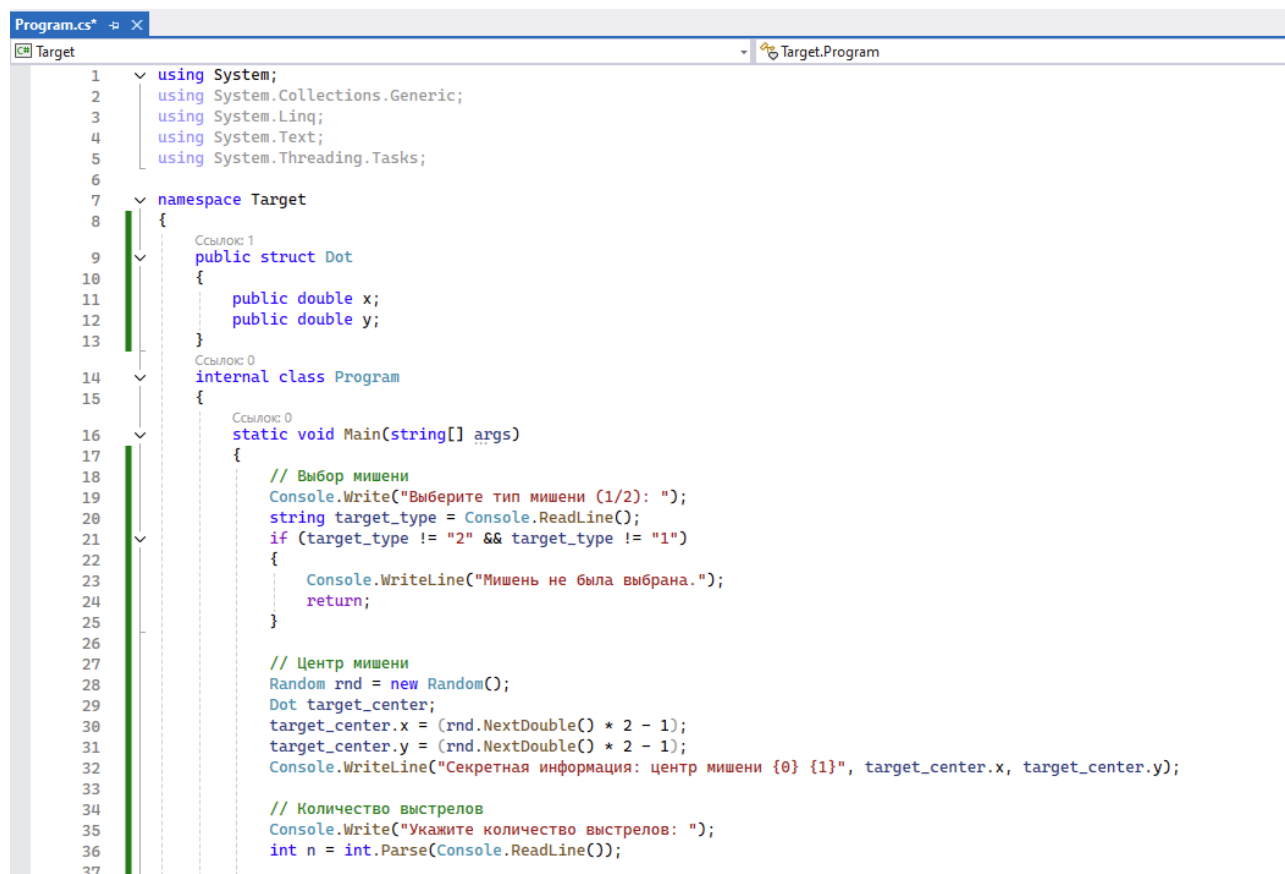
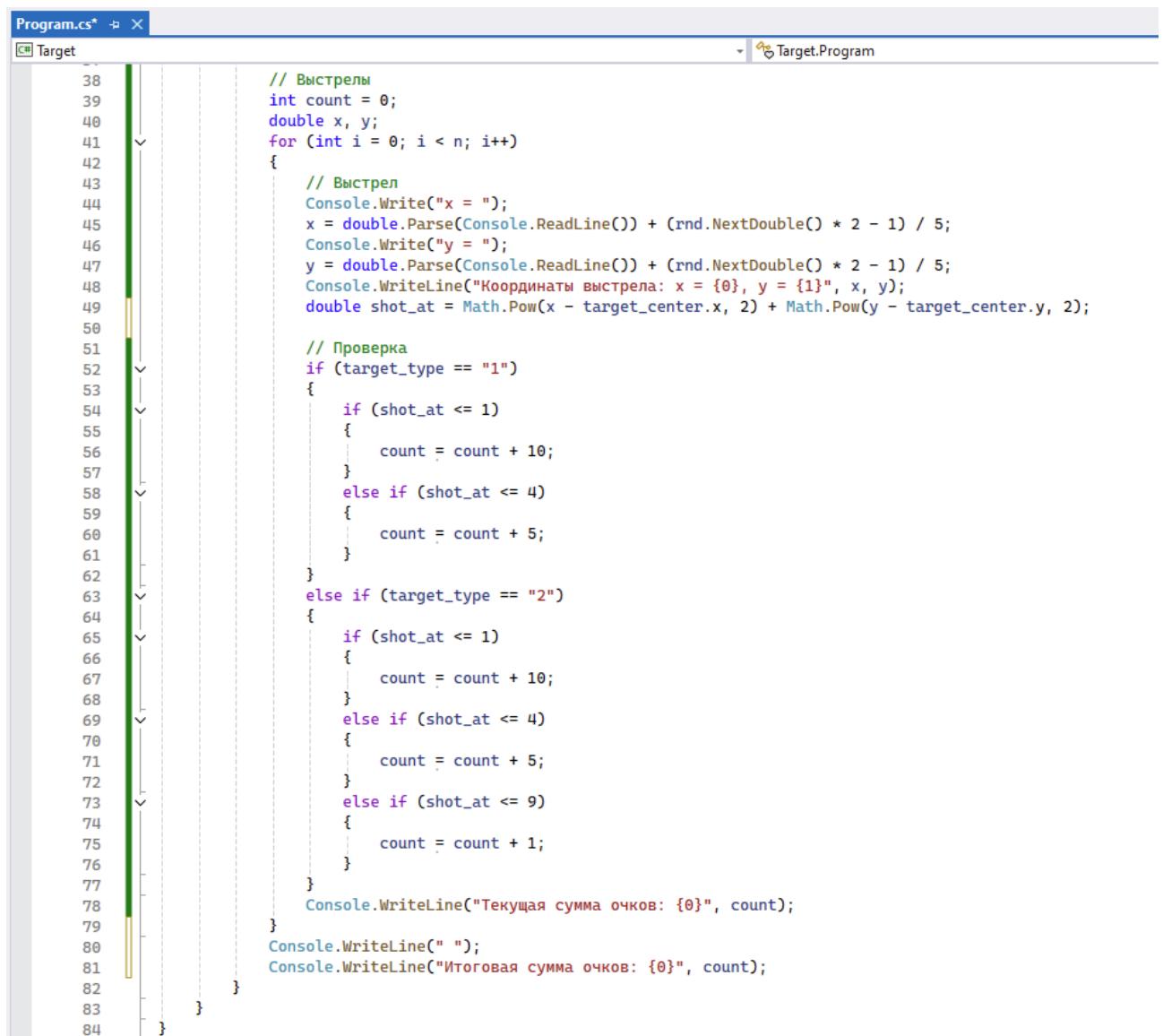


Рисунок 2.8 — Упр №3: Первая половина программы

Затем через цикл for пользователь прописывает координаты каждого выстрела. К ним добавляются случайные помехи через метод `NextDouble()`. Программа сверяет их с координатами мишени и добавляет очки, выводя

их в консоль. Когда все выстрелы были совершены, программа выводит итоговый счёт. Реализацию этих функций можно увидеть на рисунке 2.9.



```
38 // Выстрелы
39 int count = 0;
40 double x, y;
41 for (int i = 0; i < n; i++)
42 {
43     // Выстрел
44     Console.Write("x = ");
45     x = double.Parse(Console.ReadLine()) + (rnd.NextDouble() * 2 - 1) / 5;
46     Console.Write("y = ");
47     y = double.Parse(Console.ReadLine()) + (rnd.NextDouble() * 2 - 1) / 5;
48     Console.WriteLine("Координаты выстрела: x = {0}, y = {1}", x, y);
49     double shot_at = Math.Pow(x - target_center.x, 2) + Math.Pow(y - target_center.y, 2);
50
51     // Проверка
52     if (target_type == "1")
53     {
54         if (shot_at <= 1)
55         {
56             count = count + 10;
57         }
58         else if (shot_at <= 4)
59         {
60             count = count + 5;
61         }
62     }
63     else if (target_type == "2")
64     {
65         if (shot_at <= 1)
66         {
67             count = count + 10;
68         }
69         else if (shot_at <= 4)
70         {
71             count = count + 5;
72         }
73         else if (shot_at <= 9)
74         {
75             count = count + 1;
76         }
77     }
78     Console.WriteLine("Текущая сумма очков: {0}", count);
79 }
80 Console.WriteLine(" ");
81 Console.WriteLine("Итоговая сумма очков: {0}", count);
82 }
83 }
84 }
```

Рисунок 2.9 — Упр №3: Вторая половина программы

Пример работы программы показан на рисунке 2.10.

```
CA. C:\Windows\system32\cmd.exe
Выберите тип мишени (1/2): 2
Секретная информация: центр мишени 0,0926523493102995 0,159344900939308
Укажите количество выстрелов: 3
x = 1,09
y = 1,1
Координаты выстрела: x = 1,22244543035163, y = 1,29107984657915
Текущая сумма очков: 5
x = 2,4
y = -0,5
Координаты выстрела: x = 2,39187332568312, y = -0,556968845174168
Текущая сумма очков: 6
x = 0,05
y = 0,05
Координаты выстрела: x = 0,056338124818326, y = -0,0251727075666062
Текущая сумма очков: 16

Итоговая сумма очков: 16
Для продолжения нажмите любую клавишу . . .
```

Рисунок 2.10 — Упр №3: Результат работы программы

ЗАКЛЮЧЕНИЕ

При выполнении лабораторной работы было создано много различных программ на применение конструкции if-else-if, оператора switch, операторов цикла while, do while и for, оператора перехода, предусловий и постусловий, а также генераторов случайности.

Цель изучения и приобретения навыков использования управляющих конструкций для организации вычислений была выполнена.