

Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики

Факультет инфокоммуникационных технологий

Направление подготовки 11.03.02

Лабораторная работа №3

Использование выражений

Выполнил:

Доценников Никита Андреевич

Группа: К3221

Проверил:

Иванов Сергей Евгеньевич

Санкт-Петербург

2025

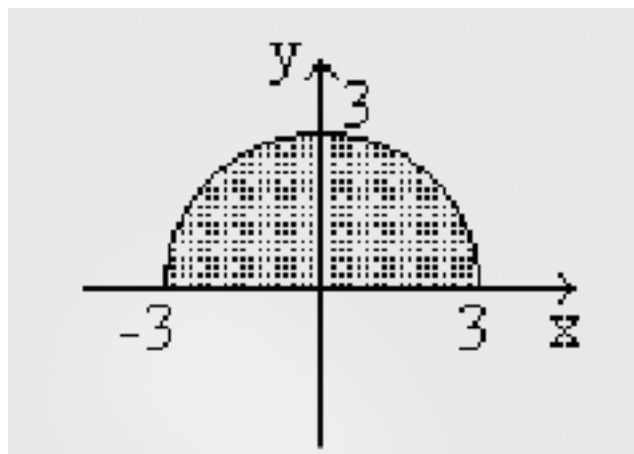
Цель работы:

Изучить и приобрести навыки использования управляющих конструкций для организации вычислений.

Упражнение 1. Реализация операторов выбора.

Задание 1. Применение конструкции if-else-if.

В этом задании я составил программу, которая выдает одно из сообщений “внутри”, “вне” или “на границе” в зависимости от положения точки относительно заштрихованной области на графике, представленном на рисунке.



```
using System;

public class Program
{
    static void Main(string[] args)
    {
        Console.Write("x = ");
        float x = float.Parse(Console.ReadLine());

        Console.Write("y = ");
        float y = float.Parse(Console.ReadLine());

        if (x * x + y * y < 9 && y > 0)
        {
            Console.WriteLine("внутри");
        }
        else if (x * x + y * y > 9 || y < 0)
        {
            Console.WriteLine("вне");
        }
        else
        {
            Console.WriteLine("на границе");
        }
    }
}
```

```

        Console.WriteLine("вне");
    }
    else
    {
        Console.WriteLine("на границе");
    }
}
}

```

Пример работы:

```

^ labs/lab3/Shapeifelse  ? main @ !? >
dotnet run                                     dotnet .NET @ 18:05
/home/nik/oop/labs/lab3/Shapeifelse/Program.cs(8,31): warning CS8604: P
ossible null reference argument for parameter 's' in 'float float.Parse
(string s)'.
/home/nik/oop/labs/lab3/Shapeifelse/Program.cs(11,31): warning CS8604:
Possible null reference argument for parameter 's' in 'float float.Parse
(string s)'.
x = 2
y = 2
Внутри

```

```

^ labs/lab3/Shapeifelse  ? main @ !? >
dotnet run                                     dotnet .NET @ 18:05
x = 0
y = 3
на границе

```

```

^ labs/lab3/Shapeifelse  ? main @ !? >
dotnet run                                     dotnet .NET @ 18:06
x = 10
y = 10
вне

```

Задание 2. Применение оператора switch.

Я создал программу моделирующую работу калькулятора.

Пользователем вводится первый операнд, требуемую операцию и второй операнд. Затем производится расчет результата.

```

using System;

public class Program

```

```

{
    static void Main(string[] args)
    {
        Console.Write("A = ");
        double a = double.Parse(Console.ReadLine());
        Console.Write("OP = ");
        char op = char.Parse(Console.ReadLine());
        Console.Write("B = ");
        double b = double.Parse(Console.ReadLine());

        bool ok = true;
        double res = 0;

        switch (op)
        {
            case '+': res = a + b; break;
            case '-': res = a - b; break;
            case '*': res = a * b; break;
            case '/':
            case ':':
                res = a / b; break;
            default: ok = false; break;
        }

        if (ok)
        {
            Console.WriteLine("{0} {1} {2} = {3}", a, op, b,
res);
        }
        else
        {
            Console.WriteLine("Операция не определена");
        }
    }
}

```

Пример работы:

1) с правильными значениями:

```
A = 3
OP = *
B = 5
3 * 5 = 15
```

2) при делении на нуль:

```
labs/lab3/Calc_switch | main | dotnet .NET 18:17
dotnet run
A = 5
OP = /
B = 0
5 / 0 = ∞
```

Бесконечность, так как в стандарте IEEE-754 прописан такой результат.

3) при делении нуля на нуль:

```
labs/lab3/Calc_switch | main | dotnet .NET 18:19
dotnet run
A = 0
OP = /
B = 0
0 / 0 = NaN
```

NaN, опять же в соответствии со стандартом.

4) при неправильной операции:

```
dotnet run dotnet .NET 18:21
A = 1
OP = @
B = 2
Операция не определена
```

Мы проваливаемся к значению по умолчанию.

Задание 3. Определение високосного года.

Программа, по введенному натуральному числу, определяет, является ли год с номером, равным этому числу, високосным.

```

using System;

public class Program
{
    public static void Main(string[] args)
    {
        Console.WriteLine("input a year: ");
        int year = int.Parse(Console.ReadLine());

        if ((year % 4 == 0 && year % 100 != 0) || (year % 400
== 0))
        {
            Console.WriteLine("YES");
        }
        else
        {
            Console.WriteLine("NO");
        }
    }
}

```

Пример работы:

```

^ labs/lab3/IsLeapYear  ↵ main 18:36
dotnet run
/home/nik/ooop/labs/lab3/IsLeapYear/Program.cs(8,30): warning CS8604: Po
ssible null reference argument for parameter 's' in 'int int.Parse(strin
g s)'.
input a year:
1234
NO

```

```

^ labs/lab3/IsLeapYear  ↵ main 18:37
dotnet run
input a year:
2020
YES

```

Упражнение 2. Реализация циклов при работе с данными размерных типов.

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

Я написал программу, которая выводит на экран последовательность целых нечетных чисел в строку через пробел с помощью трех операторов цикла `while`, `do while` и `for`.

```
using System;

public class Program
{
    public static void Main(string[] args)
    {
        Console.Write("n = ");
        int n = int.Parse(Console.ReadLine());

        // while
        Console.Write("\nwhile: \t\t");
        int i = 1;
        while (i <= n)
        {
            Console.Write(" " + i);
            i += 2;
        }

        // do-while
        Console.Write("\ndo while: \t");
        i = 1;
        do
        {
            Console.Write(" " + i);
            i += 2;
        }
        while (i <= n);

        // for:
        Console.Write("\nfor: \t\t");
        for (i = 1; i <= n; i += 2)
        {
            Console.Write(" " + i);
        }
    }
}
```

```
}  
}
```

Пример работы:

```
A labs/lab3/Loop  ? main  !? >  
dotnet run                                     dotnet .NET  © 18:44  
/home/nik/oop/labs/lab3/Loop/Program.cs(8,27): warning CS8604: Possible  
null reference argument for parameter 's' in 'int int.Parse(string s)'  
.  
n = 12  
  
while:          1 3 5 7 9 11  
do while:       1 3 5 7 9 11  
for:            1 3 5 7 9 11
```

Была написана программа, которая печатает таблицу синусов и их аргументов с периодом 0.01 от `x1` до `x2`.

```
using System;  
  
public class Program  
{  
    static void Main(string[] args)  
    {  
        double x, x1, x2, y;  
  
        Console.WriteLine("input x1:");  
        x1 = double.Parse(Console.ReadLine());  
        Console.WriteLine("input x2:");  
        x2 = double.Parse(Console.ReadLine());  
  
        Console.WriteLine("x\t\t sin(x)");  
        Console.WriteLine("-----");  
  
        x = x1;  
        do  
        {  
            y = Math.Sin(x);  
            Console.WriteLine($"{x:F2}\t {y:F6}");  
            x = x + 0.01;  
        }  
        while (x <= x2);  
    }  
}
```

```
}  
}
```

Пример работы:

```
^ labs/lab3/Sin  ? main @ !? >  
dotnet run                                     dotnet .NET @ 18:48  
/home/nik/oop/labs/lab3/Sin/Program.cs(10,27): warning CS8604: Possible  
null reference argument for parameter 's' in 'double double.Parse(str  
ing s)'.  
/home/nik/oop/labs/lab3/Sin/Program.cs(12,27): warning CS8604: Possible  
null reference argument for parameter 's' in 'double double.Parse(str  
ing s)'.  
input x1:  
2  
input x2:  
3  
x                sin(x)  
-----  
2.00      0.909297  
2.01      0.905091
```

```
using System;  
using System.Globalization;  
  
class Program  
{  
    static void FuncWhile()  
    {  
        var ci = CultureInfo.InvariantCulture;  
        double x1 = double.Parse(Console.ReadLine(), ci);  
        double x2 = double.Parse(Console.ReadLine(), ci);  
        double h = double.Parse(Console.ReadLine(), ci);  
        double x = x1;  
        while (x <= x2 + 1e-12)  
        {  
            double y = Math.Sin(x);  
            Console.WriteLine($"while: {x.ToString(ci)}  
{y.ToString(ci)}");  
            x += h;  
        }  
    }  
}
```

```

static void FuncDoWhile()
{
    var ci = CultureInfo.InvariantCulture;
    double x1 = double.Parse(Console.ReadLine(), ci);
    double x2 = double.Parse(Console.ReadLine(), ci);
    double h = double.Parse(Console.ReadLine(), ci);
    double x = x1;
    if (h <= 0) return;
    do
    {
        double y = Math.Sin(x);
        Console.WriteLine($"do-while: {x.ToString(ci)}{y.ToString(ci)}");
        x += h;
    } while (x - h < x2 + 1e-12);
}

static int GcdWhile(int a, int b)
{
    a = Math.Abs(a);
    b = Math.Abs(b);
    while (b != 0)
    {
        int temp = a % b;
        a = b;
        b = temp;
    }
    return a;
}

static int GcdDoWhile(int a, int b)
{
    a = Math.Abs(a);
    b = Math.Abs(b);
    int temp;
    if (b == 0) return a;
    do
    {
        temp = a % b;
        a = b;
        b = temp;
    } while (b != 0);
}

```

```

        return a;
    }

    static void Main()
    {
        Console.WriteLine("=== Функция с while ===");
        FuncWhile();

        Console.WriteLine("=== Функция с do-while ===");
        FuncDoWhile();

        Console.WriteLine("Введите два числа для НОД (while):");
        int a = int.Parse(Console.ReadLine());
        int b = int.Parse(Console.ReadLine());
        Console.WriteLine($"НОД (while): {GcdWhile(a, b)}");

        Console.WriteLine("Введите два числа для НОД (do-while):");
        a = int.Parse(Console.ReadLine());
        b = int.Parse(Console.ReadLine());
        Console.WriteLine($"НОД (do-while): {GcdDoWhile(a, b)}");
    }
}

```

- **while** подходит, когда важно проверять условие до входа в цикл (может выполниться 0 раз).
- **do...while** удобен, когда тело должно выполниться минимум один раз.
- Для функций на интервале **do...while** проще гарантирует вывод крайней точки.
- Для алгоритма Евклида оба варианта дают одинаковый результат; **while** короче, **do...while** более наглядно выражает шаг алгоритма.

Задание 2. Расчет суммы, используя операторы перехода.

Составлена программа, которая реализует сумму

$$s = \sum_{i=1}^{100} i,$$

для i , находящихся от 1 до k и от m до 100.

```
using System;

class Program
{
    static void Main()
    {
        Console.Write("Введите k: ");
        int k = int.Parse(Console.ReadLine());
        Console.Write("Введите m: ");
        int m = int.Parse(Console.ReadLine());

        int s = 0;
        for (int i = 1; i <= 100; i++)
        {
            if (i > k && i < m) continue;
            s += i;
        }

        Console.WriteLine("Сумма = " + s);
    }
}
```

Пример работы:

```
^ labs/lab3/Sum | main @ !? >
dotnet run                                     dotnet .NET © 19:04
/home/nik/oop/labs/lab3/Sum/Program.cs(8,27): warning CS8604: Possible
null reference argument for parameter 's' in 'int int.Parse(string s)'.
/home/nik/oop/labs/lab3/Sum/Program.cs(10,27): warning CS8604: Possible
null reference argument for parameter 's' in 'int int.Parse(string s)'
.
Введите k: 2
Введите m: 99
Сумма = 202
```

Задание 3. Стрельба по мишени.

Разработка программа, имитирующая стрельбу по мишени.

```

using System;

class Program
{
    static int VariantFromStudentNumber(int n) => (n % 2 !=
0) ? 1 : 2;

    static int ScoreVariant1(double x, double y)
    {
        var r = Math.Sqrt(x * x + y * y);
        if (r <= 1) return 10;
        if (r <= 2) return 5;
        if (r <= 3) return 2;
        return 0;
    }

    static int Sector(double x, double y)
    {
        var a = Math.Atan2(y, x);
        if (a > Math.PI / 4 && a <= 3 * Math.PI / 4) return 1;
        if (a > -Math.PI / 4 && a <= Math.PI / 4) return 5;
        if (a > -3 * Math.PI / 4 && a <= -Math.PI / 4) return
2;
        return 3;
    }

    static int ScoreVariant2(double x, double y)
    {
        var r = Math.Sqrt(x * x + y * y);
        if (r <= 1) return 10;
        if (r <= 3) return Sector(x, y);
        return 0;
    }

    static void Main()
    {
        Console.Write("Номер по списку: ");
        int student = int.Parse(Console.ReadLine());
        int variant = VariantFromStudentNumber(student);

        Console.Write("Сколько выстрелов: ");
        int n = int.Parse(Console.ReadLine());
    }
}

```

```

    Console.Write("Случайный центр? (y/n): ");
    bool rndCenter = Console.ReadLine().Trim().ToLower()
== "y";

    Console.Write("Случайная помеха? (y/n): ");
    bool noise = Console.ReadLine().Trim().ToLower() ==
"y";

    var rnd = new Random();
    double cx = 0, cy = 0;
    if (rndCenter)
    {
        cx = rnd.NextDouble() * 2 - 1;
        cy = rnd.NextDouble() * 2 - 1;
    }

    int sum = 0;
    for (int i = 0; i < n; i++)
    {
        Console.Write($"Выстрел {i + 1} (x y): ");
        var parts = Console.ReadLine().Trim().Split(new[]
{ ' ', '\t' }, StringSplitOptions.RemoveEmptyEntries);
        double x = double.Parse(parts[0]);
        double y = double.Parse(parts[1]);

        if (noise)
        {
            x += (rnd.NextDouble() * 0.2 - 0.1);
            y += (rnd.NextDouble() * 0.2 - 0.1);
        }

        x -= cx;
        y -= cy;

        int s = variant == 1 ? ScoreVariant1(x, y) :
ScoreVariant2(x, y);
        sum += s;
        Console.WriteLine($"Очки: {s}, сумма: {sum}");
    }

    Console.WriteLine($"Итоговая сумма: {sum}");

```

```
}  
}
```

Пример работы:

```
eference of a possibly null reference.  
/home/nik/oop/labs/lab3/Shooting/Program.cs(46,22): warning CS8602: Der  
eference of a possibly null reference.  
/home/nik/oop/labs/lab3/Shooting/Program.cs(60,25): warning CS8602: Der  
eference of a possibly null reference.  
Номер по списку: 1  
Сколько выстрелов: 2  
Случайный центр? (y/n): y  
Случайная помеха? (y/n): y  
Выстрел 1 (x y): 1 1  
Очки: 10, сумма: 10  
Выстрел 2 (x y): 2 2  
Очки: 2, сумма: 12  
Итоговая сумма: 12  
^ labs/lab3/Shooting | main 0 !? >  
|
```

Code review. (by zzzcode.ai)

Резюме

Код выполняет несколько задач, включая определение положения точки относительно круга, выполнение арифметических операций, проверку високосного года, вывод чисел и вычисление НОД. Однако, в коде присутствуют некоторые недостатки, которые могут быть улучшены для повышения его качества и удобства использования.

Ошибка

В коде отсутствует обработка исключений при парсинге пользовательского ввода. Это может привести к сбоям программы, если пользователь введет некорректные данные. Рекомендуется использовать `TryParse` для безопасного парсинга.

Стиль кода

Стиль кода в целом соответствует стандартам C#. Однако, в некоторых местах можно улучшить читаемость, добавив комментарии и более описательные имена переменных. Например, вместо `x1`, `x2` можно использовать `startX`, `endX`.

Структура кода

Код разбит на несколько классов, что хорошо для организации. Однако, каждый класс содержит метод `Main`, что не является хорошей практикой. Рекомендуется создать один класс с единственным методом `Main`, а остальные функции вынести в отдельные методы.

Читаемость

Читаемость кода можно улучшить, добавив комментарии, объясняющие логику выполнения. Также стоит использовать более понятные имена переменных и методов, чтобы другие разработчики могли быстрее понять, что делает код.

Производительность

Код выполняется достаточно быстро для небольших входных данных. Однако, в случае больших объемов данных, стоит рассмотреть возможность оптимизации, например, путем использования более эффективных алгоритмов. Масштабируемость

Код не очень масштабируем, так как он жестко привязан к конкретным задачам. Рекомендуется использовать более модульный подход, чтобы можно было легко добавлять новые функции без изменения существующего кода.

Безопасность

Безопасность кода можно улучшить, добавив обработку исключений и проверку входных данных. Это поможет избежать потенциальных уязвимостей, связанных с некорректным вводом.

Обработка ошибок

Обработка ошибок в коде отсутствует. Рекомендуется использовать конструкции `try-catch` для обработки возможных исключений, особенно при работе с пользовательским вводом и парсингом данных.

Заключение

В целом, код выполняет свои функции, но требует улучшений в области обработки ошибок, читаемости и структуры. Рекомендуется внести изменения, чтобы сделать код более безопасным, масштабируемым и удобным для чтения.

Выводы.

В ходе проделанной работы, я изучил и приобрел навыки использования управляющих конструкций для организации вычислений.