

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

ЛАБОРАТОРНАЯ РАБОТА №1

**Создание программы с помощью
среды разработки Visual Studio .NET**

Студент:

Швалов Даниил Андреевич

Факультет ИКТ

Группа: K32211

Преподаватель:

Иванов Сергей Евгеньевич

Упражнение 1. Создание простой программы в текстовом редакторе

Выполнив все шаги из задания, я получил следующий исходный код:

```
using System;

class Program
{
    static void Main()
    {
        string myName;
        Console.WriteLine("Please enter your name");
        myName = Console.ReadLine();
        Console.WriteLine("Hello, {0}", myName);
    }
}
```

Для компиляции программы я использовал следующую команду:

```
csc /out:MyHelloProgram.exe MyProgram.cs
```

Здесь аргумент командой строки `/out:MyHelloProgram.exe` устанавливает имя выходного файла (по умолчанию это базовое имя файла с классом `main` или имя первого файла).

Запустив программу и введя имя, получим следующий вывод программы:

```
Please enter your name
Hello, Daniil
```

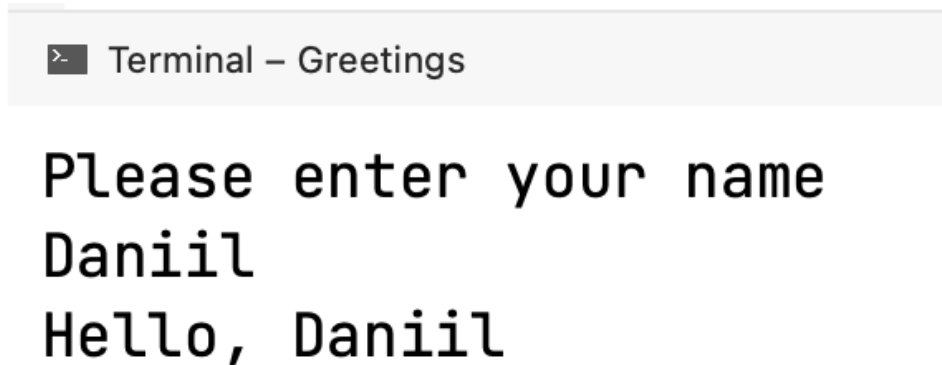
Упражнение 2. Создание программы с помощью среды разработки Visual Studio .NET

Создав новое консольное приложение C# в Visual Studio .NET, а также выполнив шаги из задания, я получил следующий исходный код:

```
namespace Greetings;
class Greeter
{
    static void Main(string[] args)
    {
        string myName;
        Console.WriteLine("Please enter your name");
        myName = Console.ReadLine();
        Console.WriteLine("Hello, {0}", myName);
    }
}
```

```
}  
}
```

Запустив программу и введя имя, я получил вывод программы, представленный на рис. 2.1.

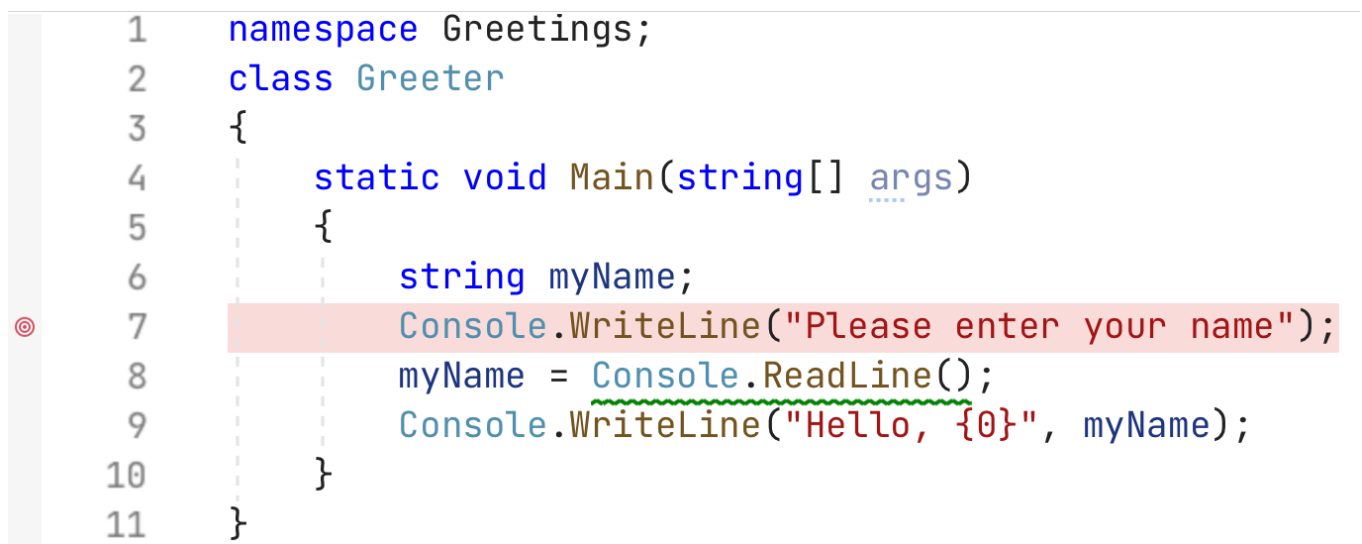


```
>- Terminal - Greetings  
  
Please enter your name  
Daniil  
Hello, Daniil
```

Рис. 2.1: Вывод программы Greetings

Упражнение 3. Использование отладчика Visual Studio .NET

На рис. 3.1 я добавил точку останова в месте, где впервые встречается команда `Console.WriteLine`.



```
1 namespace Greetings;  
2 class Greeter  
3 {  
4     static void Main(string[] args)  
5     {  
6         string myName;  
7         Console.WriteLine("Please enter your name");  
8         myName = Console.ReadLine();  
9         Console.WriteLine("Hello, {0}", myName);  
10    }  
11 }
```

Рис. 3.1: Добавление точки останова

На рис. 3.2 я добавил переменную `myName` в список выражений для мониторинга.

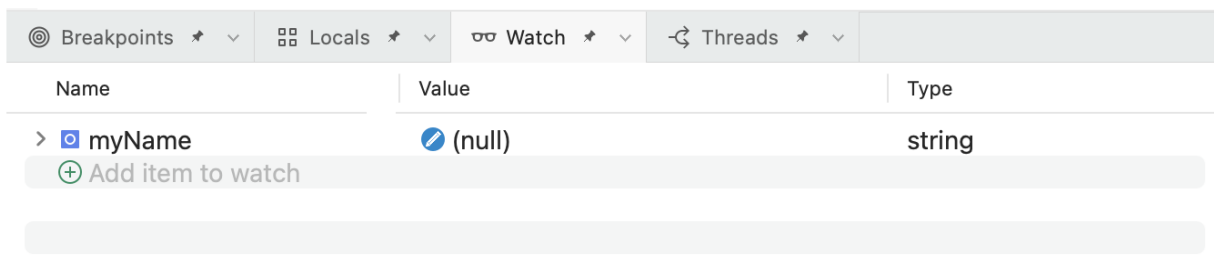


Рис. 3.2: Добавление переменной в мониторинг

На рис. 3.3 и рис. 3.4 я ввел имя, проверил значение переменной myName в отладчике.

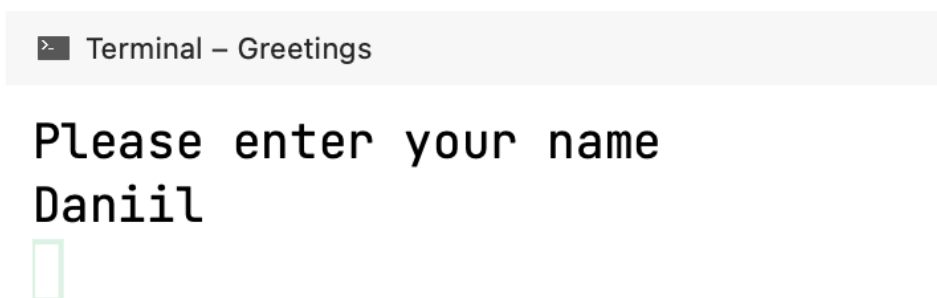


Рис. 3.3: Ввод имени

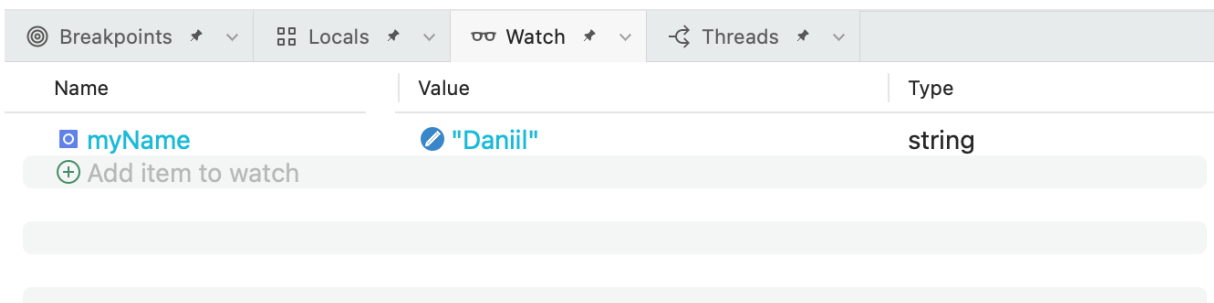


Рис. 3.4: Мониторинг значения переменной в отладчике

На рис. 3.5 я проверил вывод программы в консольном окне.

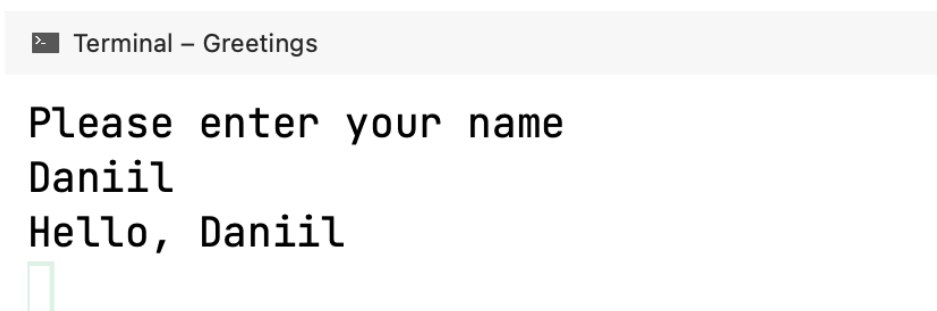


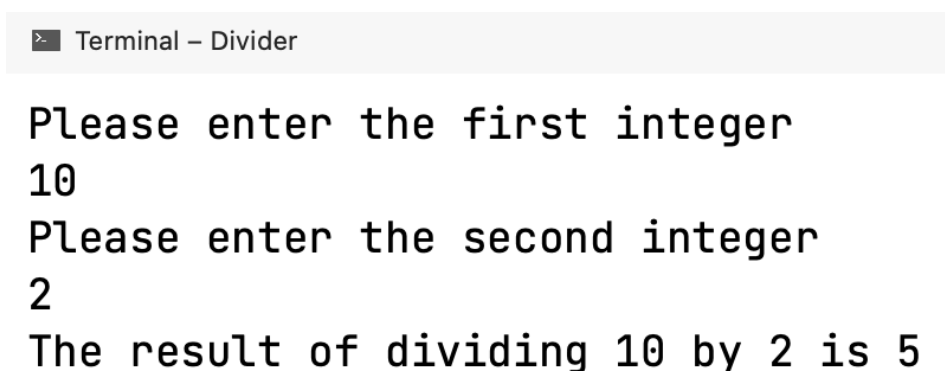
Рис. 3.5: Результат работы программы

Упражнение 4. Добавление в C#-программу обработчика исключительных ситуаций

Проектируя первые шаги, я получил следующую программу:

```
namespace Divider;
class DivideIt
{
    static void Main(string[] args)
    {
        Console.WriteLine("Please enter the first integer");
        string temp = Console.ReadLine();
        int i = Int32.Parse(temp);
        Console.WriteLine("Please enter the second integer");
        temp = Console.ReadLine();
        int j = Int32.Parse(temp);
        int k = i / j;
        Console.WriteLine("The result of dividing {0} by {1} is {2}", i, j, k);
    }
}
```

Я протестировал программу, введя числа 10 и 5, и получил корректный результат (см. рис. 4.1).



```
> Terminal - Divider

Please enter the first integer
10
Please enter the second integer
2
The result of dividing 10 by 2 is 5
```

Рис. 4.1: Ввод чисел 10 и 5

Затем я вновь протестировал программу, введя числа 10 и 3 (см. рис. 4.2). Я получил не совсем корректный результат. Вместо 3.33 программа вывела 3. Это произошло потому, что для хранения чисел мы использовали целочисленный тип `int`. Поэтому при делении целых чисел мы также получили целое число вместо числа с плавающей запятой.

```
Please enter the first integer
10
Please enter the second integer
3
The result of dividing 10 by 3 is 3
```

Рис. 4.2: Ввод чисел 10 и 3

Теперь проверим что будет, если 10 разделить 0 (см. рис. 4.3). В программе возникла исключительная ситуация.

```
Terminal – Divider
Please enter the first integer
10
Please enter the second integer
0
Unhandled exception. System.DivideByZeroException: Attempted to divide by zero.
   at Divider.DivideIt.Main(String[] args) in /Users/danilshvalov/Documents/dev/projects/oop/labs/lab1/Divider/Divider/Program1.cs:line 12
/bin/bash: line 1: 91803 Abort trap: 6           "/usr/local/share/dotnet/dotnet" "/Users/danilshvalov/Documents/dev/projects/oop/labs/lab1/Divider/Divider/bin/Debug/net7.0/Divider.dll"
```

Рис. 4.3: Ввод чисел 10 и 0

Добавим в программу обработчик исключительных ситуаций:

```
namespace Divider;
class DivideIt
{
    public static void Main(string[] args)
    {
        try
        {
            Console.WriteLine("Please enter the first integer");
            string temp = Console.ReadLine();
            int i = Int32.Parse(temp);
            Console.WriteLine("Please enter the second integer");
            temp = Console.ReadLine();
            int j = Int32.Parse(temp);
            int k = i / j;
            Console.WriteLine("The result of dividing {0} by {1} is {2}", i, j, k);
        }
        catch (Exception e)
        {
            Console.WriteLine("An exception was thrown: {0}", e.Message);
        }
    }
}
```

Теперь проверим работу программы при вводе 10 и 0 (см. рис. 4.4). Программа обработала ошибку корректно.

```
Terminal - Divider

Please enter the first integer
10
Please enter the second integer
0
An exception was thrown: Attempted to divide by zero.
```

Рис. 4.4: Ввод чисел 10 и 0 после добавления обработчика

Теперь попробуем ввести букву вместо числа (см. рис. 4.5). Программа также корректно обработала эту ситуацию.

```
Terminal - Divider

Please enter the first integer
a
An exception was thrown: The input string 'a' was not in a correct format.
```

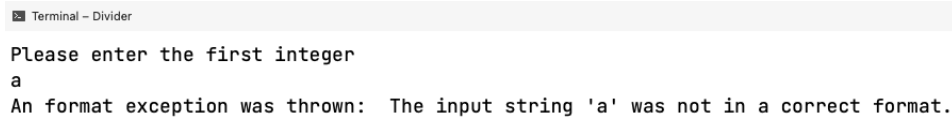
Рис. 4.5: Ввод буквы вместо числа

Добавим в программу обработчик исключительной ситуации при вводе данных неверного формата:

```
namespace Divider;
class DivideIt
{
    static void Main(string[] args)
    {
        try
        {
            Console.WriteLine("Please enter the first integer");
            string temp = Console.ReadLine();
            int i = Int32.Parse(temp);
            Console.WriteLine("Please enter the second integer");
            temp = Console.ReadLine();
            int j = Int32.Parse(temp);
            int k = i / j;
            Console.WriteLine("The result of dividing {0} by {1} is {2}", i, j, k);
        }
        catch (FormatException e)
        {
            Console.WriteLine("An format exception was thrown: {0}", e.Message);
        }
        catch (Exception e)
        {
            Console.WriteLine("An exception was thrown: {0}", e.Message);
        }
    }
}
```

```
}  
}
```

Проверим программу и введем букву вместо числа (см. рис. 4.6). Программа верно обработала эту ситуацию.



```
Terminal - Divider  
Please enter the first integer  
a  
An format exception was thrown: The input string 'a' was not in a correct format.
```

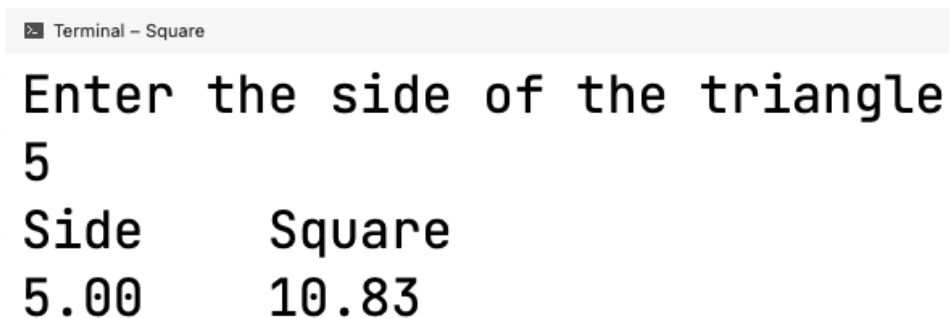
Рис. 4.6: Ввод буквы вместо числа

Упражнение 5. Расчет площади треугольника

В своей программе я считываю сторону треугольника в переменную `side`. После этого я создаю переменную `p` и присваиваю ей значение полу периметра треугольника. Для расчета площади треугольника я использую формулу Герона:

$$S = \sqrt{p(p-a)(p-b)(p-c)} = \sqrt{p(p-x)^3}.$$

Нахожу площадь, сохраняю ее в переменную `square` и вывожу информацию в виде таблицы. Кроме того, в программе я обрабатываю исключительные ситуации и вывожу информацию об ошибках пользователю на экран. Пример использования программы представлен на рис. 5.1.



```
Terminal - Square  
Enter the side of the triangle  
5  
Side      Square  
5.00      10.83
```

Рис. 5.1: Пример расчета площади

Исходный код программы представлен в следующем листинге:

```
namespace Square;  
  
class Program  
{  
    static void Main(string[] args)  
    {  
        try
```



```

{
    Console.WriteLine("Enter the side of the triangle");
    string temp = Console.ReadLine();
    double side = Double.Parse(temp);

    double p = side * 3 / 2;
    double square = Math.Sqrt(p * Math.Pow(p - side, 3));

    Console.WriteLine("Side\tSquare");
    Console.WriteLine("{0:f2}\t{1:f2}", side, square);
}
catch (FormatException)
{
    Console.WriteLine(
        "The side of the triangle must be a floating point number"
    );
}
catch (Exception e)
{
    Console.WriteLine("An exception was thrown: {0}", e.Message);
}
}
}

```
