

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

Кафедра программных систем

**Лабораторная работа №1**

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

Выполнил: Мороз

Илья Олегович

Группа № K3120

Проверила: Казанова П.П.

Санкт-Петербург

2019

**Цель работы:**

Целью работы является знакомство с языком C#, а также создание простых программ на данном языке.

**Ход работы:**

**Упражнение 1.**

1. Был создан пустой файл program.cs.
2. В данном файле был введён код на языке C#:

using System;

class Program

{

static void Main()

{

string myName;

Console.WriteLine("Please enter your name");

myName = Console.ReadLine();

Console.WriteLine("Hello, {0}", myName);

}

}

Листинг 1 – код программы program.cs

1. Была запущена командная строка разработчика для VS 2017. В ней была выполнена команда csc -?, котора позволяет узнать список аргументов командной строки в режиме командной строки компилятора C#.

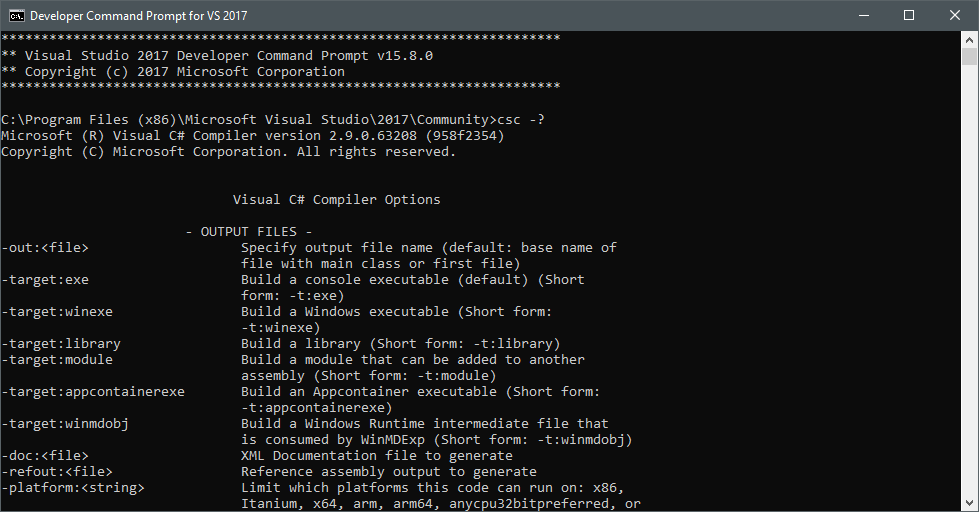


Рисунок 1 – командная строка разработчика

1. При помощи команды “csc /out:MyHelloProgram.exe program.cs” программа была скомпилирована. При помощи команды “MyHelloProgram.exe” программа была запущена и протестирована.

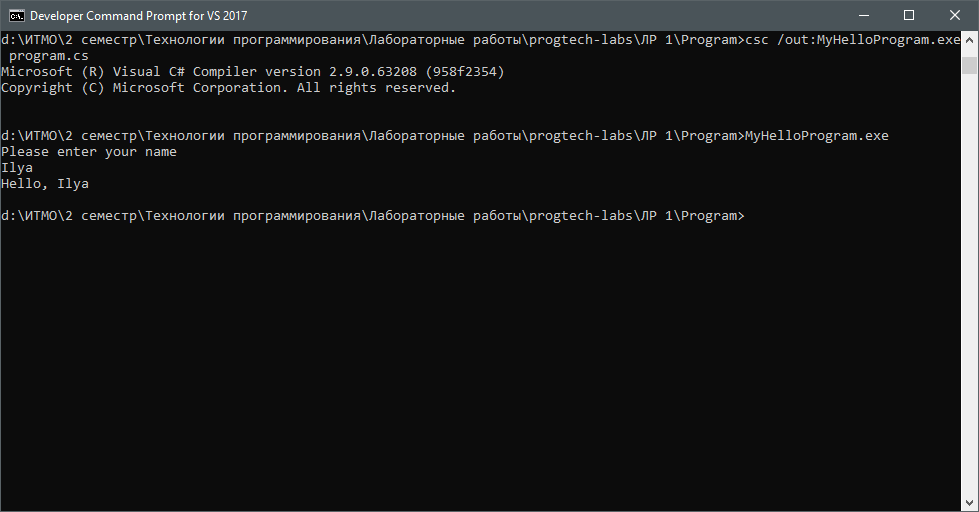


Рисунок 2 – результат компиляции и работы программы

**Упражнение 2.**

1. В среде разработки Visual Studio был создан новый проект Greeting.
2. В файле Program.cs было изменено имя класса на Greeter и написан данный код:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

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);

}

}

}

Листинг 2 – исходный код программы Greeting (файл Program.cs)

1. Программа была скомпилирована и запущена.

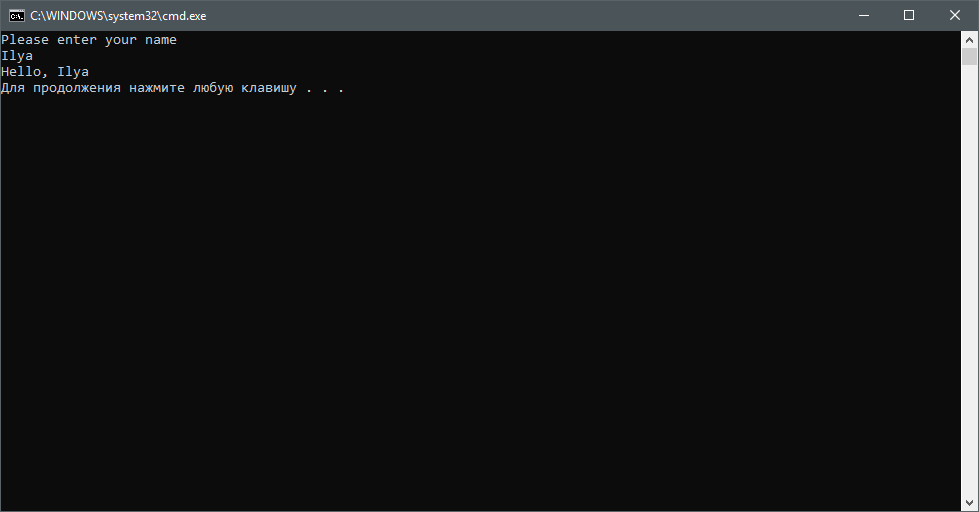


Рисунок 3 – результат работы программы Greeting

**Упражнение 3.**

1. На строке 14 была установлена точка остановки выполнения программы.
2. Программа была запущена в режиме Debug.

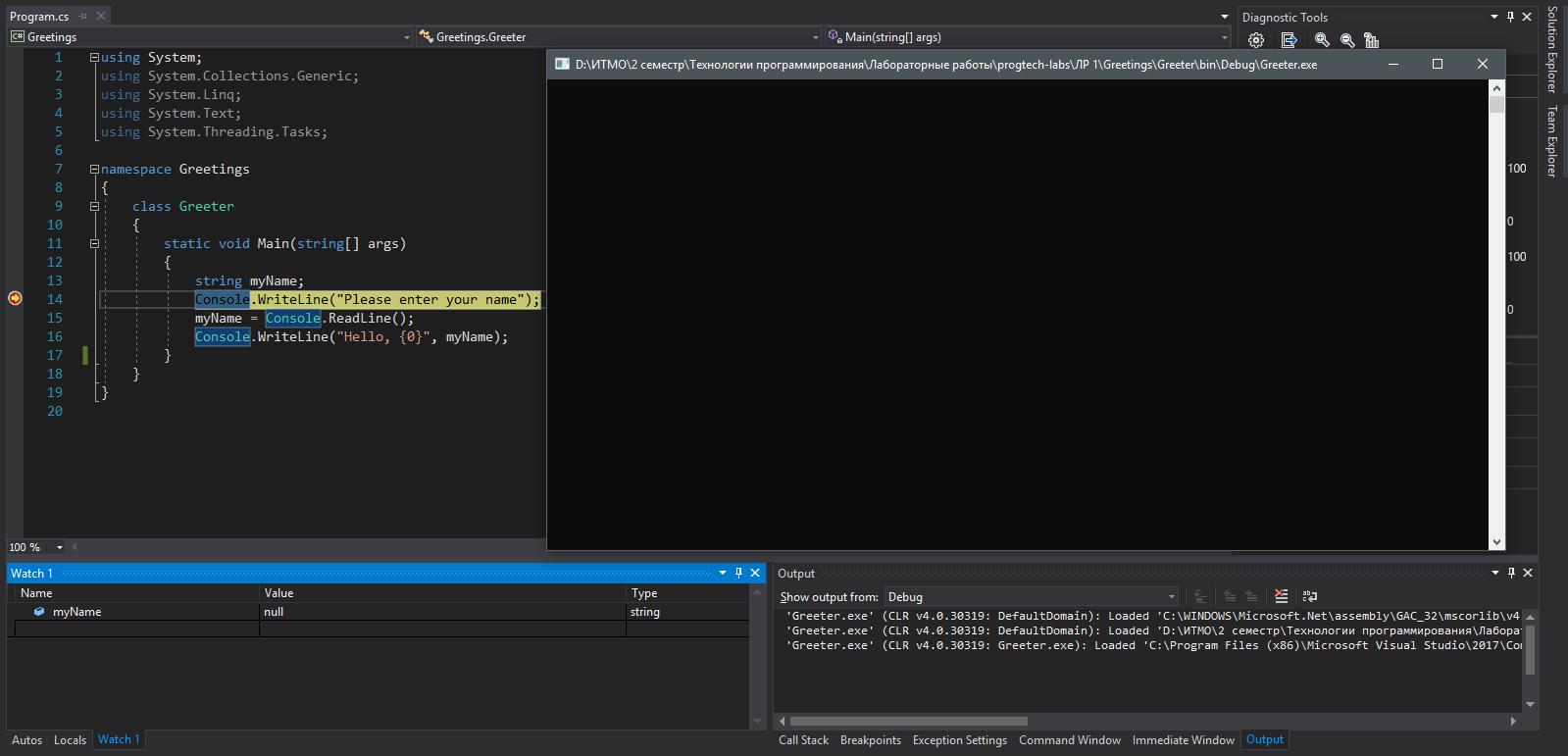


Рисунок 4 – процесс выполнения программы был остановлен

1. После выполнения команд в режиме Debug и ввода имени, значение переменной myName изменилось:

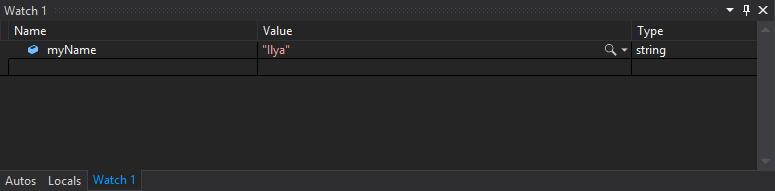


Рисунок 5 – окно Watch 1

1. С помощью команды Continue выполнение программы было завершено.

**Упражнение 4.**

1. Был создан новый проект Divider.
2. В методе Main был реализован ввод двух целых чисел:

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);

1. Было реализовано деление первого числа на второе:

int k = i / j;

Console.WriteLine("i / j = {0}", k);

1. Программа была скомпилирована и запущена.

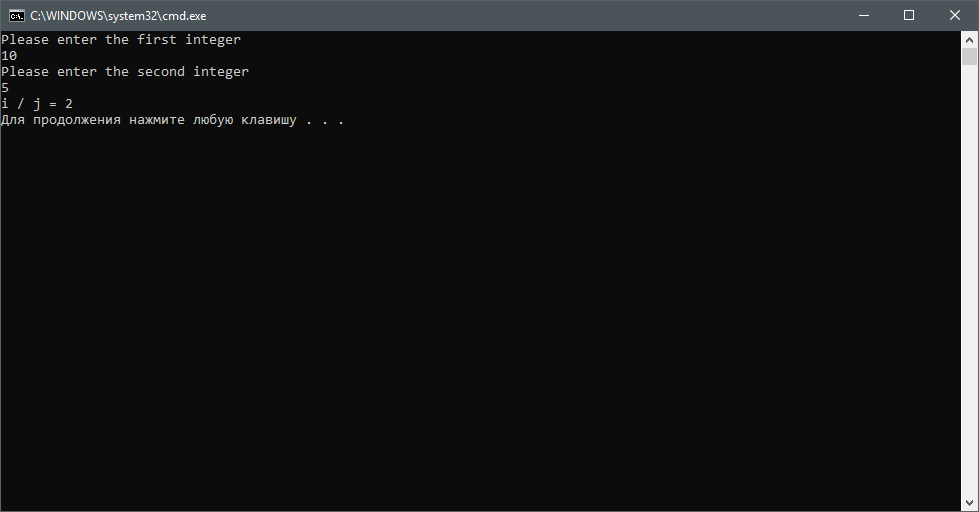


Рисунок 6 – результат работы программы

1. При делении на ноль во время выполнения программы происходит необработанное исключение.

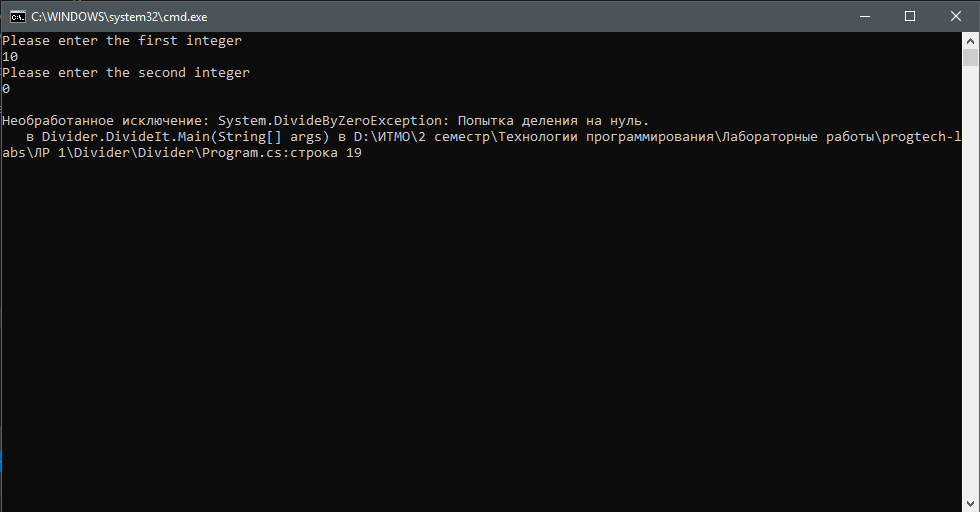


Рисунок 7 – деление на ноль

1. Для обработки исключений в код был добавлен блок try-catch.

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("i / j = {0}", k);

}

catch(Exception e)

{

Console.WriteLine("An exeption was thrown: {0}", e.Message);

}

Листинг 3 – код программы с обработчиком исключений

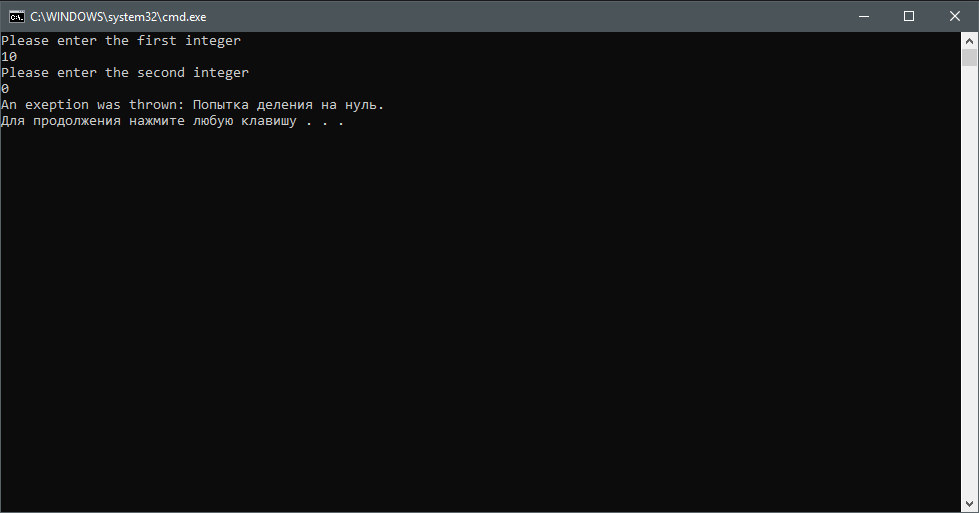


Рисунок 8 – теперь исключение обрабатывается

1. В обработчик ошибок были добавлены новые блоки catch, позволяющие отдельно ловить ошибки формата и деления на ноль.

catch(FormatException e)

{

Console.WriteLine("An format exception was thrown: {0}", e.Message);

}

catch(DivideByZeroException e)

{

Console.WriteLine("An DivideByZero exception was thrown: {0}", e.Message);

}

catch(Exception e)

{

Console.WriteLine("An exception was thrown: {0}", e.Message);

}

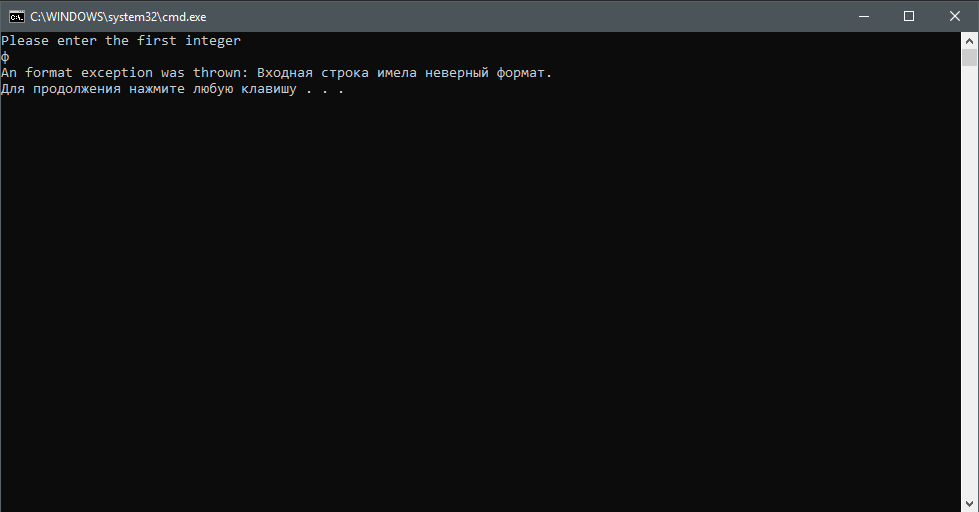


Рисунок 9 – исключение формата обрабатывается со специальным сообщением

**Упражнение 5.**

1. Был создан новый проект TriangleArea для расчёта площади треугольника по формуле Герона. В программе использовался обработчик исключений, чтобы обрабатывать исключения формата ввода и ввода отрицательного периметра.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace TriangleArea

{

class Program

{

static void Main(string[] args)

{

try

{

Console.WriteLine("Enter p");

int p = Int32.Parse(Console.ReadLine());

if (p <= 0)

{

throw new ArithmeticException();

}

double side = p / 3;

double area = Math.Sqrt(p \* (p - side) \* (p - side) \* (p - side));

Console.WriteLine("Side:\tArea:");

Console.WriteLine("{0:F}\t{1:F}", side, area);

}

catch (FormatException e)

{

Console.WriteLine("An FormatException was thrown: {0}", e.Message);

}

catch (ArithmeticException e)

{

Console.WriteLine("An ArithmeticException was thrown: {0}", e.Message);

}

}

}

}

Листинг 4 – код программы TriangleArea

1. Программы была скомпилирована и запущена.

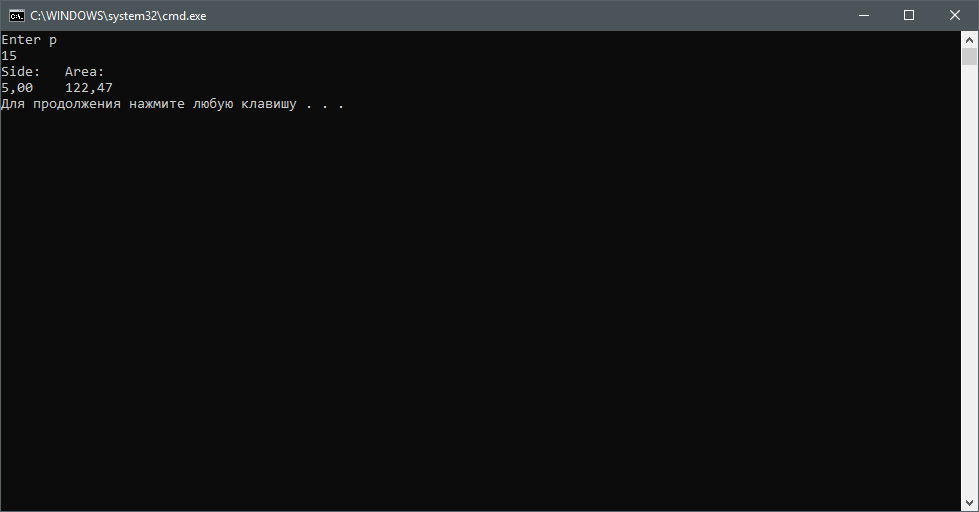


Рисунок 10 – результат работы программы

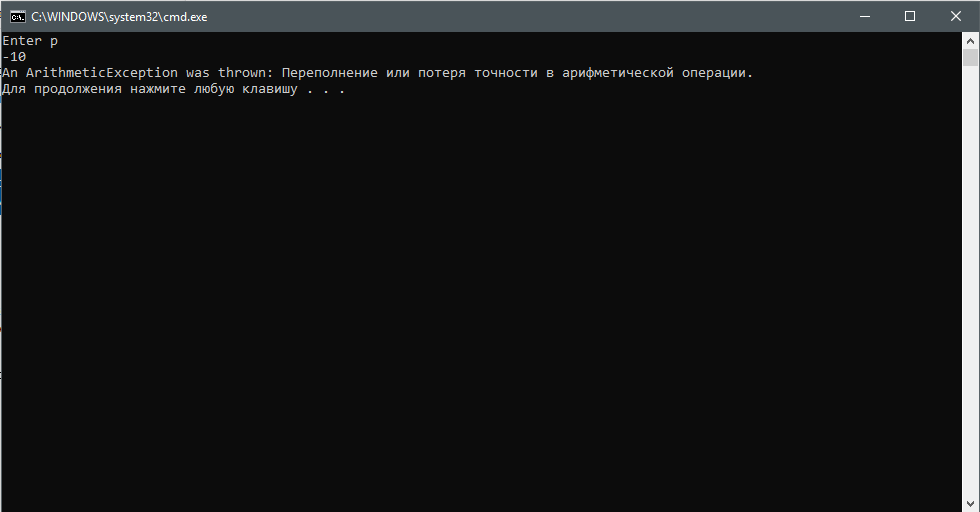


Рисунок 11 – обработка исключения

**Вывод:**

В ходе выполнения данной программы были изучены основы языка C#. Были созданы простейшие программы и изучена возможность компиляции через командную строку. Был изучен режим Debug. Также в ходе написания программ были использованы обработчики исключительных ситуаций.

**Ответы на контрольные вопросы:**

1. Какие виды Int существуют?

Int16, Int32, Int64.

1. Можно ли сделать так, чтобы не писать Console каждый раз?

Нет, нельзя, так как это некоторый класс со своими методами, а не пространство имён.