**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технический университет**

**имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

Факультет «Информатика и вычислительная техника»

Кафедра ИУ5 «Системы обработки информации и управления»

**Отчет по лабораторной работе № 1**

Задание: разработать программу для решения биквадратного уравнения

Студент Левин Михаил Александрович

*фамилия, имя, отчество*

Группа ИУ5-32Б

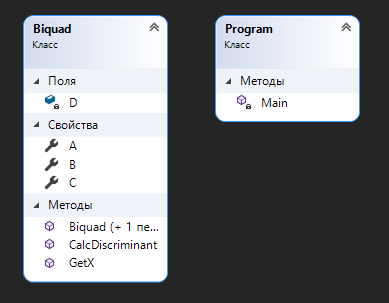
Москва, 2020 г.

# Задание

Разработать программу для решения биквадратного уравнения.

1. Программа должна быть разработана в виде консольного приложения на языке C#.
2. Программа осуществляет ввод с клавиатуры коэффициентов А, В, С, вычисляет дискриминант и корни уравнения (в зависимости от дискриминанта).
3. Если коэффициент А, В, С введен некорректно, то необходимо проигнорировать некорректное значение и ввести коэффициент повторно.
4. Первой строкой программа выводит ФИО разработчика и номер группы.
5. Корни уравнения выводятся зеленым цветом. Если корней нет, то сообщение выводится красным цветом.
6. ДОПОЛНИТЕЛЬНОЕ ТРЕБОВАНИЕ. Коэффициенты А, В, С задаются в виде параметров командной строки. Если они не указаны, то вводятся с клавиатуры в соответствии с пунктом 2. Проверка из пункта 3 в этом случае производится для параметров командной строки без повторного ввода с клавиатуры.

# Диаграмма классов



# Текст программы

**Biquad.cs**

using System;

using System.Collections.Generic;

namespace Lab1

{

public class Biquad

{

public double A { get; set; } //коэффициент А

public double B { get; set; } //коэффициент B

public double C { get; set; }

private double D;

public Biquad()

{

D = -1;

}

public Biquad(double a, double b, double c)

{

A = a;

B = b;

C = c;

CalcDiscriminant();

}

public void CalcDiscriminant()

{

D = B \* B - 4 \* A \* C;

}

public IList<double> GetX()

{

List<double> result = null;

if (A == 0) return null;

if (D >= 0)

{

result = new List<double>();

double y1 = (-B + Math.Sqrt(D)) / 2 \* A;

double y2 = (-B - Math.Sqrt(D)) / 2 \* A;

if (y1 > 0)

{

double x1 = Math.Sqrt(y1);

double x2 = -x1;

result.Add(x1);

result.Add(x2);

}

if (D > 0)

{

if (y2 > 0)

{

double x3 = Math.Sqrt(y2);

double x4 = -x3;

result.Add(x3);

result.Add(x4);

}

}

}

return result;

}

}

}

**Program.cs**

using System;

using System.Collections.Generic;

namespace Lab1

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Программа для решения бикдвардатных уровнений");

Console.WriteLine("Разработчик: Левин А.А. Группа: ");

double a = 0, b = 0, c = 0;

if (args.Length ==3)

{

try

{

a = double.Parse(args[0]);

b = double.Parse(args[1]);

c = double.Parse(args[2]);

}

catch

{

Console.WriteLine("Один или несколько параметров не являются числовыми значениями!");

}

}

else

{

do

{

Console.Write("Введите коэф-т А: ");

}

while (!double.TryParse(Console.ReadLine(),out a));

do

{

Console.Write("Введите коэф-т B: ");

}

while (!double.TryParse(Console.ReadLine(), out b));

do

{

Console.Write("Введите коэф-т C: ");

}

while (!double.TryParse(Console.ReadLine(), out c));

}

var biquad = new Biquad(a, b, c);

IList<double> result = biquad.GetX();

if (result == null)

{

ConsoleColor current = Console.ForegroundColor;

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Корни не найдены!");

Console.ForegroundColor = current;

}

else

{

ConsoleColor current = Console.ForegroundColor;

Console.ForegroundColor = ConsoleColor.Green;

for (int i = 0; i < result.Count; i++)

{

Console.WriteLine($"X{i+1} = {result[i]}");

}

Console.ForegroundColor = current;

}

}

}

}

# Экранные формы

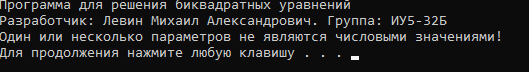


Рисунок 1 – Программа вызывается с тремя параметры, один из которых не является числом.

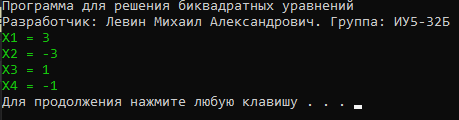


Рисунок 2 – Программа вызывается с тремя числовыми параметрами (1, -10, 9)

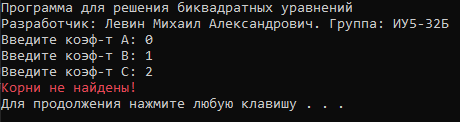


Рисунок 3 – Программа не может найти корней

# Контрольные вопросы:

1. Какими способами можно преобразовать значение строкового типа в значение числового типа?

Для преобразования значения строкового типа в значение числового типа есть два способа. Первый способом это использованием методов класса System.Convert, например, таких как ToInt32 для получения целочисленного значения из строки, ToDouble для получения вещественного числа из строки.

Второй способ использование методов Parse и TryParse у целочисленных и вещественных классов, например для получения целочисленного значения из строки используется Int32.Parse(«строка»).

1. Какие символьные и строковые типы данных существуют в C#?

В C# существуют следующие встроенные символьные типы данных: char. Ключевое слово типа char — это псевдоним для типа структуры System.Char .NET, представляющий символ UTF-16 в Юникоде.

Строковые типы данных: string. Тип string представляет последовательность, состоящую из нуля или более символов в кодировке Юникод. string является псевдонимом для System.String в .NET.

1. Какие средства консольного ввода/вывода существуют в C#?

За работу с консолью в C# отвечает статический класс Console. Класс Console предоставляет стандартные потоки для консольных приложений: входной, выходной и поток сообщений об ошибках. Для консольного ввода/вывода используются следующие основные методы: Write, WriteLine, Read, ReadLine.

1. Какие целочисленные типы данных существуют в C#?

Приведем список целочисленных типов и их краткое описание в виде таблицы:

|  |  |
| --- | --- |
| byte | хранит целое число от 0 до 255 и занимает 1 байт. Представлен системным типом System.Byte |
| sbyte | ранит целое число от -128 до 127 и занимает 1 байт. Представлен системным типом System.SByte |
| short | хранит целое число от -32768 до 32767 и занимает 2 байта. Представлен системным типом System.Int16 |
| ushort | хранит целое число от 0 до 65535 и занимает 2 байта. Представлен системным типом System.UInt16 |
| int | хранит целое число от -2147483648 до 2147483647 и занимает 4 байта. Представлен системным типом System.Int32. |
| uint | хранит целое число от 0 до 4294967295 и занимает 4 байта. Представлен системным типом System.UInt32 |
| long | хранит целое число от –9 223 372 036 854 775 808 до 9 223 372 036 854 775 807 и занимает 8 байт. Представлен системным типом System.Int64 |
| ulong | хранит целое число от 0 до 18 446 744 073 709 551 615 и занимает 8 байт. Представлен системным типом System.UInt64 |

1. Как задаются и обрабатываются аргументы командной строки в консольном приложении?

Для того, чтобы консольное приложение могло обработать переданные параметры в командой строке, необходимо определить метод main следующим образом: static int Main(string[] args).

Параметр метода Main — это массив String, представляющий аргументы командной строки. Как правило, определить, существуют ли аргументы, можно, проверив свойство Length, например:

if (args.Length == 0)

{

System.Console.WriteLine("Please enter a numeric argument.");

return 1;

}

Строковые аргументы также можно преобразовать в числовые типы с помощью класса Convert или метода Parse. Например, следующая инструкция преобразует string в число long с помощью метода Parse:

long num = Int64.Parse(args[0]);

1. Как работает механизм обработки исключений в C#?

Обработка исключений использует ключевые слова try, catch и finally для действий, которые могут оказаться неудачными.

Пример:

try

{

result = SafeDivision(a, b);

Console.WriteLine("{0} divided by {1} = {2}", a, b, result);

}

catch (DivideByZeroException e)

{

Console.WriteLine("Attempted divide by zero.");

}

В указанном примере обрабатывается ситуация попытки деления на ноль.

1. Какие условные операторы существуют в C#?

В C# cуществует три условных оператора:

1. Конструкция if/else

Пример:

if(num1 > num2)

{

Console.WriteLine($"Число {num1} больше числа {num2}");

}

1. Конструкция switch

Пример:

switch (selection)

{

case "Y":

Console.WriteLine("Вы нажали букву Y");

break;

case "N":

Console.WriteLine("Вы нажали букву N");

break;

default:

Console.WriteLine("Вы нажали неизвестную букву");

break;

}

1. Тернарная операция

Пример:

int z = selection=="+"? (x+y) : (x-y);

1. Как работает цикл foreach?

Ключевой особенностью цикла foreach является то, что он может использоваться только с коллекциями реализующими интерфейс IEnumerable или IEnumerable<T>, либо коллекция должна удовлетворять следующими условиям: тип коллекции реализует открытый метод GetEnumerator, открытое свойство Current, открытый метод MoveNext.

1. Как работает форматированный вывод в консоль?

В качестве входных данных для составного форматирования в .NET используется список объектов и строка составного формата. Строка составного формата состоит из фиксированного текста, в который включены индексированные местозаполнители, которые называются элементами форматирования и соответствуют объектам из списка. Операция форматирования создает результирующую строку, состоящую из исходного фиксированного текста, в который включено строковое представление объектов из списка.

1. Какие операторы цикла существуют в C#?

Существуют следующие операторы для реализации циклов в С#:

1. For - использует счётчик в своей работе, применяется когда известно точное количество итераций.
2. Do while – используется для того, чтобы выполнить хотя бы одну итерацию, далее проверяется условие выхода из цикла. Данный цикл с постусловием.
3. While – цикл с предусловием, сначала проверяется условие входа в цикл, а зачем, если условие соответствует, выполняется тело цикла. Данный цикл может не выполниться ни разу.
4. Foreach – цикл для обхода коллекций, соответствующих определенным требованиям (указаны в вопросе № 8).