**4 ОБРАБОТКА ИСКЛЮЧИТЕЛЬНЫХ СИТУАЦИЙ**

Задание №1. Вычислить значение выражений. В каждой разработанной программе должна быть осуществлена обработка исключительных ситуаций. Отдельно обработаны исключения DivideByZeroException, FormatException.

Листинг программы:

static void Main(string[] args)

{

while (true)

{

try

{

Console.Write("Enter x value: ");

double x = double.Parse(Console.ReadLine());

Console.WriteLine($"First Expression result = {FirstExpression(x)}");

Console.WriteLine($"Second Expression result = {SecondExpression(x)}");

break;

}

catch (FormatException)

{

Console.WriteLine("Format exception (value cannot be a letter, please, enter number)");

}

catch (DivideByZeroException)

{

Console.WriteLine("Divide by zero (Value cannot be divided by zero)");

}

catch (Exception e)

{

Console.WriteLine($"Error! {e}");

}

}

static double FirstExpression(double x)

{

try

{

return Math.Round(((Math.Pow(x, 2) - 3) / (Math.Pow(x, 2) - 4)), 4);

}

catch (Exception)

{

throw new DivideByZeroException();

}

}

static double SecondExpression(double x)

{

try

{

return Math.Round((Math.Log10(x) / (x - 1)), 4);

}

catch (Exception)

{

throw new DivideByZeroException();

}

}

}

Таблица 4.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 87 | 1.0001, 0.0226 |

Источник: собственная разработка

Анализ результатов представлен на рисунке 4.1.

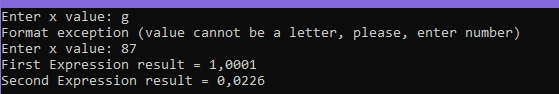


Рисунок 4.1 – Результат работы программы

Источник: собственная разработка

Задание №2. Для данного вещественного x найти значение следующей функции f, принимающей вещественные значения. В каждой разработанной программе должна быть осуществлена обработка исключительных ситуаций. Отдельно обработаны исключения DivideByZeroException, FormatException.

Сгенерировать пользовательское исключение с помощью оператора throw при проверке условий выхода за диапазон.

Листинг программы:

static void Main(string[] args)

{

while (true)

{

try

{

Console.Write("Enter x value : ");

double x = double.Parse(Console.ReadLine());

Console.WriteLine($"Expression result = {Expression(x)}");

break;

}

catch (FormatException)

{

Console.WriteLine("Format exception (value cannot be a letter, please, enter number)");

}

catch (DivideByZeroException)

{

Console.WriteLine("Divide by zero (Value cannot be divided by zero)");

}

catch (Exception e)

{

Console.WriteLine($"Error! {e}");

}

}

static double Expression(double x)

{

try

{

if ((x > -5) && (x < -1)) return Math.Round(Math.Pow(x, 3),4);

else if (x >= -1) return Math.Round(x / (5 \* x + 5),4);

else throw new Exception($"Value '{x}' is not in range!");

}

catch (DivideByZeroException)

{

throw new DivideByZeroException();

}

}

}

Таблица 4.2 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| -4 | -64 |

Источник: собственная разработка

Анализ результатов представлен на рисунке 4.2.

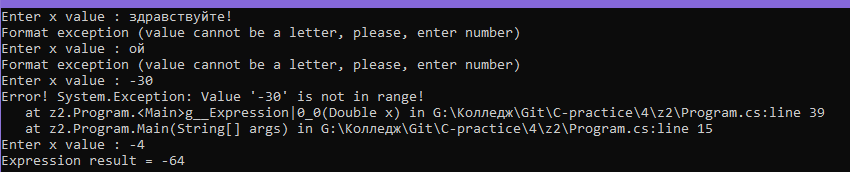


Рисунок 4.2 – Результат работы программы

Источник: собственная разработка

Задание №3. В каждой разработанной программе должна быть осуществлена обработка исключительных ситуаций. Отдельно обработаны исключения DivideByZeroException, FormatException. По возможности сгенерируйте пользовательское исключение с помощью оператора.

Описать метод SortInc3(A, B, C), меняющий содержимое переменных A, B, C таким образом, чтобы их значения оказались упорядоченными по возрастанию (A, B, C — вещественные параметры, являющиеся одновременно входными и выходными). С помощью этого метода упорядочить по возрастанию два данных набора из трех чисел: (A1, B1, C1) и (A2, B2, C2).

Листинг программы:

static void Main(string[] args)

{

while (true)

{

try

{

Console.Write("Enter A1 value: ");

double A = int.Parse(Console.ReadLine());

Console.Write("Enter B1 value: ");

double B = int.Parse(Console.ReadLine());

Console.Write("Enter C1 value: ");

double C = int.Parse(Console.ReadLine());

SortInc3(ref A, ref B, ref C);

Console.WriteLine($"Sorted values: \nA - {A} \nB - {B} \nC - {C}");

Console.Write("Enter A2 value: ");

A = int.Parse(Console.ReadLine());

Console.Write("Enter B2 value: ");

B = int.Parse(Console.ReadLine());

Console.Write("Enter C2 value: ");

C = int.Parse(Console.ReadLine());

SortInc3(ref A, ref B, ref C);

Console.WriteLine($"Sorted values: \nA - {A} \nB - {B} \nC - {C}");

break;

}

catch (FormatException)

{

Console.WriteLine("Format exception (value cannot be a letter, please, enter number)");

}

catch (DivideByZeroException)

{

Console.WriteLine("Divide by zero (Value cannot be divided by zero)");

}

catch (Exception e)

{

Console.WriteLine($"Error! {e}");

}

}

static void SortInc3(ref double A, ref double B, ref double C)

{

double[] mass = { A, B, C };

double[] ordered = mass.OrderBy(x => x).ToArray();

A = ordered[0];

B = ordered[1];

C = ordered[2];

}

}

Таблица 4.3 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 8, 1, 9, 5, 9, 1 | 1, 8, 9, 1, 5, 9 |

Источник: собственная разработка

Анализ результатов представлен на рисунке 4.3.

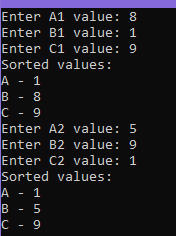


Рисунок 4.3 – Результат работы программы

Источник: собственная разработка