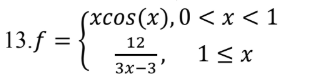
**4 Обработка исключительных ситуаций**

**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.

Сгенерируйте пользовательское исключение с помощью оператора throw при

проверке условий выхода за диапазон.



Листинг программы:

using System;

class Program

{

static double CalculateFunction(double x)

{

if (x <= 0 || x > 10) // Задаём диапазон, например, (0, 10]

{

throw new ArgumentOutOfRangeException("x", "Значение x должно быть в диапазоне (0, 10].");

}

if (x > 0 && x < 1)

{

return x \* Math.Cos(x); // x \* cos(x) при 0 < x < 1

}

else if (x >= 1)

{

if (Math.Abs(3 \* x - 3) < 0.000001) // Проверка деления на ноль

{

throw new DivideByZeroException("Знаменатель в выражении равен нулю.");

}

return 12 / (3 \* x - 3); // 12 / (3x - 3) при x >= 1

}

throw new InvalidOperationException("Неизвестное состояние функции.");

}

static void Main()

{

try

{

Console.Write("Введите значение x: ");

double x = double.Parse(Console.ReadLine()); // Чтение значения x

double result = CalculateFunction(x);

Console.WriteLine($"Результат функции f({x}) = {result:F6}");

}

catch (FormatException)

{

Console.WriteLine("Ошибка: Введено некорректное значение. Ожидалось вещественное число.");

}

catch (DivideByZeroException ex)

{

Console.WriteLine($"Ошибка: {ex.Message}");

}

catch (ArgumentOutOfRangeException ex)

{

Console.WriteLine($"Ошибка: {ex.Message}");

}

catch (Exception ex)

{

Console.WriteLine($"Неизвестная ошибка: {ex.Message}");

}

}

}

Таблица 4.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 0,5 | 0,438791 |

Анализ результатов:

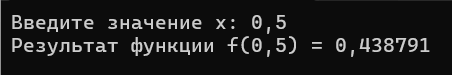


Рисунок 4.1 – Результат работы программы