**4 Обработка исключительных ситуаций**

Задание 1. Вычислить значение выражений. В каждой разработанной программе должна быть осуществлена обработка исключительных ситуаций. Отдельно обработаны исключения DivideByZeroException, FormatException. Выражения для задания представлены на рисунке 4.1.

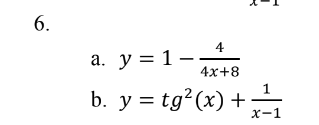


Рисунок 4.1 – Выражения для вычисления

Источник: задание 1

Листинг программы для выражения A:

try {

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

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

if (x == -2)

{throw new DivideByZeroException("Знаменатель не может быть равен 0"); }

double y = 1 - (4 / (4 \* x + 8));

Console.WriteLine("Значение выражения a. y = {0}", y); }

catch (DivideByZeroException ex) {Console.WriteLine(ex.Message);}

catch (FormatException ex) { Console.WriteLine("Ошибка ввода: " + ex.Message); }

catch (Exception ex) {Console.WriteLine("Произошла ошибка: " + ex.Message); }

Таблица 4.1 – Выходные и входные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| ff | Ошибка ввода: Input string was not in a correct format. |

Источник: собственная разработка

Анализ результатов:

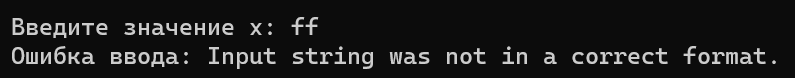


Рисунок 4.2 – Результат работы программы

Источник: собственная разработка

Листинг программы для выражения B:

try {

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

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

if (x == 1) {

throw new DivideByZeroException("Знаменатель не может быть равен 0");}

double y = Math.Pow(Math.Tan(x), 2) + (1 / (x - 1));

Console.WriteLine("Значение выражения b. y = {0}", y); }

catch (DivideByZeroException ex){Console.WriteLine(ex.Message);}

catch (FormatException ex){Console.WriteLine("Ошибка ввода: " + ex.Message);}

catch (Exception ex){Console.WriteLine("Произошла ошибка: " + ex.Message);}

Таблица 4.2 – Выходные и входные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 1 | Знаменатель не может быть равен 0 |

Источник: собственная разработка

Анализ результатов:

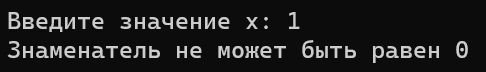


Рисунок 4.3 – Результат работы программы

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

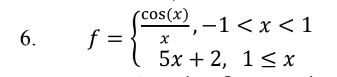


Рисунок 4.4 – Функция f

Источник: задание 2

Листинг программы:

try{ Console.Write("Введите вещественное число x: ");

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

double f;

if (x<-1){

throw new Exception("x должно быть больше или равно -1.");}

else if (x == 0){throw new DivideByZeroException("Деление на ноль."); }

else if (x >= 1) {f = 5 \* x + 2;} else{f = Math.Cos(x) / x;}

Console.WriteLine("f = " + f); }

catch (DivideByZeroException ex){

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

catch (FormatException){

Console.WriteLine("Ошибка: введено некорректное значение.");}

catch (Exception ex){

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

Таблица 4.3 – Выходные и входные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 3 | f = 17 |

Источник: собственная разработка

Анализ результатов:

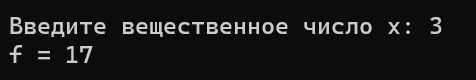


Рисунок 4.5 – Результат работы программы

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 DigitCountSum(K, C, S), находящую количество C цифр целого положительного числа K, а также их сумму S (K — входной, C и S — выходные параметры целого типа). С помощью этого метода найти количество и сумму цифр для каждого из пяти данных целых чисел.

Листинг программы:

static void Main(){

int[] numbers = new int[5]; for (int i = 0; i < 5; i++) {

Console.Write($"Введите {i + 1}-е число: ");

string input = Console.ReadLine();

try{int number = int.Parse(input); int count, sum;

DigitCountSum(number, out count, out sum);

Console.WriteLine($"Колв-во цифр: {count}, Их сумма: {sum}");}

catch (DivideByZeroException ex){

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

catch (FormatException ex){

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

catch (Exception ex){

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

static void DigitCountSum(int k, out int c, out int s){

if (k <= 0){

throw new ArgumentException("Число должно быть больше 0");}

c = 0; s = 0;

while (k > 0) {

int digit = k % 10;

c++;

s += digit; k /= 10; } }

Таблица 4.4 – Выходные и входные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 123; 1111; 4587; 1255; 1478 | Кол-во цифр: 3, Их сумма: 6  Кол-во цифр: 4, Их сумма: 4  Кол-во цифр: 4, Их сумма: 24  Кол-во цифр: 4, Их сумма: 13  Кол-во цифр: 4, Их сумма: 20 |

Источник: собственная разработка

Анализ результатов:

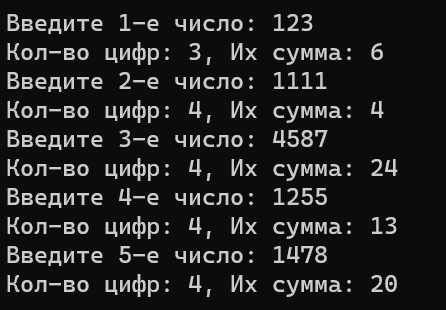


Рисунок 4.6 – Результат работы программы

Источник: собственная разработка