Отчет по практической работе №5

Наименование работы: Обработка исключительных ситуаций.

Индивидуальные задания (вариант №9):

1. Предусмотреть исключительные ситуации в индивидуальном задании №1 практической работы №4.
2. Описать класс дробей — рациональных чисел, являющихся отношением двух целых чисел. Предусмотреть методы сложения, вычитания, умножения и вычитания дробей. Написать программу, демонстрирующую все разработанные элементы класса.

Вопросы выходного контроля:

1. Что такое перегрузка метода?

Ответ:

Перегрузка методов – это приём программирования, который позволяет разработчику в одном классе для методов с разными параметрами использовать одно и то же имя.

1. Для того чтобы перегрузить метод необходимо?

Ответ:

Иметь метод с названием как у перегружаемого но разным по кол-ву параметров, их порядку, их типу.

1. Что такое исключения?

Ответ:

Обработка исключительных ситуаций – механизм языков программирования, предназначенный для описания реакции программы на ошибки времени выполнения и другие возможные проблемы, которые могут возникнуть при выполнении программы и приводят к невозможности дальнейшей отработки программой её базового алгоритма.

1. Для чего нужны блоки catch, try, throw и finally в исключениях?

Ответ:

В try мы пишем блок кода, в catch мы пишем что будет выполнено во время исключения(если оно возникнет), finally выполнится в любом случае, а через throw мы можем вызвать исключение вручную.

1. Как перегружаются операторы?

Ответ:

Операторы перезагружаются благодоря ключевому слову operator и самому оператору.

1. Какие операторы можно перегружать, а какие нельзя?

Ответ:

Только данные операторы можно перегружать:

* унарные операторы +, -, !, ~, ++, --
* бинарные операторы +, -, \*, /, %
* операции сравнения ==, !=, <, >, <=, >=
* логические операторы &&, ||

Ход работы:

1. Разработал программу в соответствии с заданием №1. Были реализованы класс, метод, математические функции.

Листинг программы:

using System;

using System.Linq;

namespace Lab4

{

class Program

{

static void Main(string[] args)

{

GetNatue();

Console.ReadLine();

}

static void GetNatue()

{

try

{

Console.Write("Введите N: ");

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

for (int ch = 100; ch < 1000; ch++)

{

del(N, ch);

}

}

catch (Exception)

{

Console.WriteLine("Error!");

}

}

static void del(int N, int ch)

{

try

{

int Ni = 0;

int?[] listi = new int?[N + 1];

for (int i = 1; i <= ch; i++)

{

if (ch % i == 0)

{

listi[Ni] = i;

Ni++;

}

if ((Ni == N) && (i == ch))

{

Console.WriteLine("Найдено число: " + ch + " с делителями: " + String.Join(" ", listi));

return;

}

if (N < Ni) return;

}

}

catch (Exception)

{

Console.WriteLine("Error!");

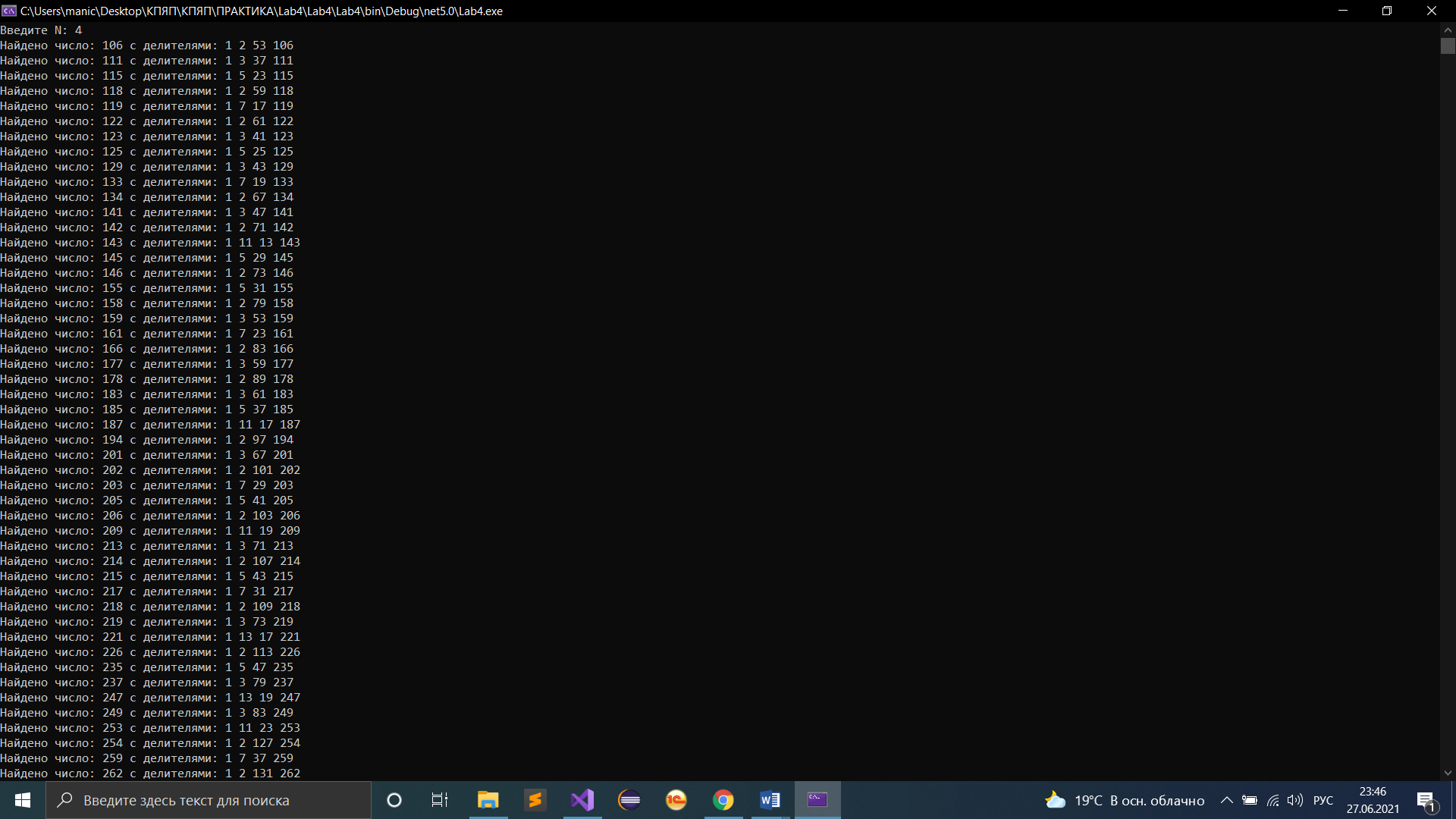
}

}

}

}

Результат выполнения программы Рис 1.1:



**Рисунок 1.1 – результат программы 5.1.**

1. Разработал программу в соответствии с заданием №1. Были реализованы класс, метод, математические функции.

Листинг программы:

using System;

namespace Lab5.\_2

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Введите +, -, /, \*");

string wib = Console.ReadLine();

double x1 = double.Parse(Console.ReadLine());

double y1 = double.Parse(Console.ReadLine());

switch (wib)

{

case "+":

{

Uan.Plus(x1, y1);

break;

}

case "-":

{

Tu.Minus(x1, y1);

break;

}

case "/":

{

Fri.Del(x1, y1);

break;

}

case "\*":

{

Fier.Umn(x1, y1);

break;

}

}

}

}

public class Uan

{

public static void Plus(double x, double y)

{

Console.WriteLine(x + y);

}

}

public class Tu

{

public static void Minus(double x, double y)

{

Console.WriteLine(x - y);

}

}

public class Fri

{

public static void Del(double x, double y)

{

Console.WriteLine(x / y);

}

}

public class Fier

{

public static void Umn(double x, double y)

{

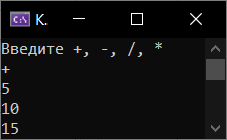
Console.WriteLine(x \* y);

}

}

}

Результат выполнения программы Рис 2.1:



**Рисунок 2.1 – результат программы 5.2.**

Отметка\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Подпись преподавателя\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_