**20. Документирование в формате XML.**

Задание №1.

Задокументировать практическую работу «Процедуры и функции – методы класса»

Листинг программы:

z1.Program:

///<summary>Импортирование класса Math</summary>

using static System.Math;

///<summary>Переменная объявления a</summary>

Console.Write("Введите a: ");

double numA = Convert.ToDouble(Console.ReadLine());

///<summary>Переменная объявления a</summary>

Console.Write("Введите b: ");

double numB = Convert.ToDouble(Console.ReadLine());

///<summary>

///Данная проверка нужна для определения

///наименьшего значения возращаемся методом<c>Func(double x)</c>

///</summary>

if (Func(numA) > Func(numB))

{

Console.WriteLine($"В точке b:{numB} со значением y:{Func(numB):.###} " +

$"функция принимает наименьшее значение");

}

else

{

Console.WriteLine($"В точке a:{numA} со значением y:{Func(numA):.###} " +

$"функция принимает наименьшее значение");

}

///<param name="x">число x по которому ьудет находится

///значение функции</param>

///<returns>возращает результат выполнения функции

///cos(2x) + sin(x - 3)</returns>

double Func(double x) => Cos(2 \* x) + Sin(x - 3);

z2.Program:

///<summary>Обявление переменных a, b, h(шаг)</summary>

Console.Write("Введите a: ");

double numA = Convert.ToDouble(Console.ReadLine());

Console.Write("Введите b: ");

double numB = Convert.ToDouble(Console.ReadLine());

Console.Write("Введите шаг h: ");

double h = Convert.ToDouble(Console.ReadLine());

///<summary>

///цикл фор нанчинает отсчет с <c>numA</c>> до <c>numB</c> ключительно с

///шагом <c>h</c>

/// </summary>

for (double i = numA; i <= numB; i += h)

{

if (F.Fc(i) < 0)

{

//Задание 3

F.Fc(i, out double y);

Console.WriteLine($"x:{i} y:{y} a + b = {numA + numB}");

}

else if (0 <= F.Fc(i) && F.Fc(i) < 10)

{

Console.WriteLine($"x:{i} y:{F.Fc(i)} a - b = {numA - numB}");

}

else

{

Console.WriteLine($"x:{i} y:{F.Fc(i)} a \* b = {numA \* numB}");

}

}

/// <summary>

/// Класс <c>F</c> предназначен ля содержания функций

/// </summary>

class F

{

/// <summary>

/// Считает функцию по заданному x по формуле x^2 - 5 \* x

/// </summary>

/// <param name="x">переменная покоторой ведется расчет фк</param>

/// <returns></returns>

public static double Fc(double x) => x \* x - 5 \* x;

//Задание 3

/// <summary>

/// Перегрузка основного метода Fc с изменеными параметрами

/// и с типом возращаемых данных void

/// </summary>

/// <param name="x">Переменная покоторой ведется расчет</param>

/// <param name="y">Выходной параметр, который высчитывается

/// по формуле x^2 - 5 \* x</param>

public static void Fc(double x, out double y)

{

y = 0;

y = x \* x - 5 \* x;

}

}

z3.TwodimensionalArray:

namespace z3.Model

{

/// <summary>

/// Класс двухмерный массив строк

/// </summary>

public class TwodimensionalArray

{

private string[,] \_arr;

/// <value name="CountElem">Количество элемента во всем <c>\_arr</c></value>

public int CountElem

{

get { return \_arr.Length; }

}

/// <summary>

/// Базовый конструктор класса

/// </summary>

/// <param name="sizeI">кол-во строк</param>

/// <param name="sizeJ">кол-во столбцов</param>

public TwodimensionalArray(int sizeI, int sizeJ)

{

\_arr = RandomArr(sizeI, sizeJ);

}

/// <summary>

/// Перегруженный оператор +

/// </summary>

/// <param name="a1">Первый объект TwodimensionalArray для сложения</param>

/// <param name="a2">Второй объект TwodimensionalArray для сложения</param>

/// <returns>Возращает сумму двух элментов или же вернет null если размеры матриц не совпадают</returns>

public static TwodimensionalArray operator +(TwodimensionalArray a1,

TwodimensionalArray a2)

{

if (a1.\_arr.GetUpperBound(0) == a2.\_arr.GetUpperBound(0) &&

a1.\_arr.GetUpperBound(1) == a2.\_arr.GetUpperBound(1))

{

TwodimensionalArray newArr =

new TwodimensionalArray(a1.\_arr.GetUpperBound(0) + 1, a1.\_arr.GetUpperBound(1) + 1);

for (int i = 0; i < a1.\_arr.GetUpperBound(0) + 1; i++)

{

for (int j = 0; j < a1.\_arr.GetUpperBound(1) + 1; j++)

{

newArr.\_arr[i, j] = a1.\_arr[i, j] + a2.\_arr[i, j];

}

}

return newArr;

}

return null;

}

/// <summary>

/// Вывод <c>\_arr</c> на консоль

/// </summary>

public void Write()

{

for (int i = 0; i <= \_arr.GetUpperBound(0); i++)

{

for (int j = 0; j < \_arr.GetUpperBound(1) + 1; j++)

{

Console.Write($" {\_arr[i,j]}\t");

}

Console.WriteLine("");

}

}

/// <summary>

/// Создание массива строк и заполнением его рандомными значениями

/// </summary>

/// <param name="sizeI">кол-во строк</param>

/// <param name="sizeJ">кол-во столбцов</param>

/// <returns>Рандомно заполненый массив строк</returns>

private string[,] RandomArr(int sizeI, int sizeJ)

{

string[,] newArr = new string[sizeI, sizeJ];

Random random= new Random();

for (int i = 0; i < sizeI; i++)

{

for (int j = 0; j < sizeJ; j++)

{

newArr[i,j] = random.Next(10).ToString();

}

}

return newArr;

}

}

}

z4. RealMatrix:

namespace z4.Model

{

/// <summary>

/// Вещественная матрица

/// </summary>

public class RealMatrix

{

private double[,] \_arr;

///<value name="CountElem">возращает кол-во всех элементов в <c>\_arr</c></value>

public int CountElem

{

get { return \_arr.Length; }

}

/// <summary>

/// Индексатор

/// </summary>

/// <param name="i">Номер строки</param>

/// <param name="j">Номер столбца</param>

/// <returns>элемент из матрицы по заданным параметрам</returns>

public double this[int i, int j]

{

get { return \_arr[i,j]; }

set { \_arr[i, j] = value; }

}

/// <summary>

/// Базовый конструктор

/// </summary>

/// <param name="sizeI">кол-во строк</param>

/// <param name="sizeJ">кол-во столбцов</param>

public RealMatrix(int sizeI, int sizeJ)

{

\_arr = RandomArr(sizeI, sizeJ);

}

/// <summary>

/// Презагрузка оператора + для сложения двух RealMatrix;

/// Каждый i,j элемент первой RealMatrix будет сложен с каждым i,j элементов второй RealMatrix

/// </summary>

/// <param name="a1">первый объект RealMatrix для сложения</param>

/// <param name="a2">второй объект RealMatrix для сложения</param>

/// <returns>сложение двух объектов RealMatrix></returns>

/// <exception cref="Exception">сробатывает, когда разные размеры матриц</exception>

public static RealMatrix operator +(RealMatrix a1,

RealMatrix a2)

{

if (a1.\_arr.GetUpperBound(0) == a2.\_arr.GetUpperBound(0) &&

a1.\_arr.GetUpperBound(1) == a2.\_arr.GetUpperBound(1))

{

RealMatrix newArr =

new RealMatrix(a1.\_arr.GetUpperBound(0) + 1, a1.\_arr.GetUpperBound(1) + 1);

for (int i = 0; i < a1.\_arr.GetUpperBound(0) + 1; i++)

{

for (int j = 0; j < a1.\_arr.GetUpperBound(1) + 1; j++)

{

newArr.\_arr[i, j] = a1.\_arr[i, j] + a2.\_arr[i, j];

}

}

return newArr;

}

throw new Exception("Разные размеры матриц");

}

/// <summary>

/// Прегрузка оператора + для сложения RealMatrix с числом

/// Каждый i,j элемент RealMatrix будет сложен с числом

/// </summary>

/// <param name="a1">объект RealMatrix для сложения с числом</param>

/// <param name="num">число для сложения</param>

/// <returns>объект RealMatrix с которым сложили число</returns>

public static RealMatrix operator +(RealMatrix a1,

double num)

{

RealMatrix newArr =

new RealMatrix(a1.\_arr.GetUpperBound(0) + 1, a1.\_arr.GetUpperBound(1) + 1);

for (int i = 0; i < a1.\_arr.GetUpperBound(0) + 1; i++)

{

for (int j = 0; j < a1.\_arr.GetUpperBound(1) + 1; j++)

{

newArr.\_arr[i, j] = a1.\_arr[i, j] + num;

}

}

return newArr;

}

/// <summary>

/// Перегрузка оператора - для вычитания RealMatrix с RealMatrix.

/// Каждый i,j элемент первой RealMatrix будет вычитан из i,j элемент второй RealMatrix

/// </summary>

/// <param name="a1">первый объект RealMatrix для вычитания</param>

/// <param name="a2">второй объект RealMatrix для вычитания</param>

/// <returns>результат вычитки двух RealMatrix</returns>

/// <exception cref="Exception"></exception>

public static RealMatrix operator -(RealMatrix a1,

RealMatrix a2)

{

if (a1.\_arr.GetUpperBound(0) == a2.\_arr.GetUpperBound(0) &&

a1.\_arr.GetUpperBound(1) == a2.\_arr.GetUpperBound(1))

{

RealMatrix newArr =

new RealMatrix(a1.\_arr.GetUpperBound(0) + 1, a1.\_arr.GetUpperBound(1) + 1);

for (int i = 0; i < a1.\_arr.GetUpperBound(0) + 1; i++)

{

for (int j = 0; j < a1.\_arr.GetUpperBound(1) + 1; j++)

{

newArr.\_arr[i, j] = a1.\_arr[i, j] - a2.\_arr[i, j];

}

}

return newArr;

}

throw new Exception("Разные размеры матриц");

}

/// <summary>

/// Прегрузка оператора - для вычитания RealMatrix от числа.

/// Каждый i,j элемент RealMatrix будет вычитан от числа

/// </summary>

/// <param name="a1">объект RealMatrix для вычитания числа</param>

/// <param name="num">число для вычитания</param>

/// <returns>объект RealMatrix с которой вычли число</returns>

public static RealMatrix operator -(RealMatrix a1,

double num)

{

RealMatrix newArr =

new RealMatrix(a1.\_arr.GetUpperBound(0) + 1, a1.\_arr.GetUpperBound(1) + 1);

for (int i = 0; i < a1.\_arr.GetUpperBound(0) + 1; i++)

{

for (int j = 0; j < a1.\_arr.GetUpperBound(1) + 1; j++)

{

newArr.\_arr[i, j] = a1.\_arr[i, j] - num;

}

}

return newArr;

}

/// <summary>

/// Перегрузка оператора ">" для двух объектов RealMatrix

/// </summary>

/// <param name="a1">первый объект RealMatrix для сравнения</param>

/// <param name="a2">второй объект RealMatrix для сравнения</param>

/// <returns>true/false</returns>

/// <exception cref="Exception">если разные размеры RealMatrix.\_arr</exception>

public static bool operator >(RealMatrix a1,

RealMatrix a2)

{

if (a1.\_arr.GetUpperBound(0) == a2.\_arr.GetUpperBound(0) &&

a1.\_arr.GetUpperBound(1) == a2.\_arr.GetUpperBound(1))

{

for (int i = 0; i < a1.\_arr.GetUpperBound(0) + 1; i++)

{

for (int j = 0; j < a1.\_arr.GetUpperBound(1) + 1; j++)

{

if (a1.\_arr[i, j] < a2.\_arr[i, j])

{

return false;

}

}

}

return true;

}

throw new Exception("Разные размеры матриц");

}

/// <summary>

/// Перегрузка оператора "<" для двух объектов RealMatrix

/// </summary>

/// <param name="a1">первый объект RealMatrix для сравнения</param>

/// <param name="a2">второй объект RealMatrix для сравнения</param>

/// <returns>true/false</returns>

/// <exception cref="Exception">если разные размеры RealMatrix.\_arr</exception>

public static bool operator <(RealMatrix a1,

RealMatrix a2)

{

if (a1.\_arr.GetUpperBound(0) == a2.\_arr.GetUpperBound(0) &&

a1.\_arr.GetUpperBound(1) == a2.\_arr.GetUpperBound(1))

{

for (int i = 0; i < a1.\_arr.GetUpperBound(0) + 1; i++)

{

for (int j = 0; j < a1.\_arr.GetUpperBound(1) + 1; j++)

{

if (a1.\_arr[i, j] > a2.\_arr[i, j])

{

return false;

}

}

}

return true;

}

throw new Exception("Разные размеры матриц");

}

/// <summary>

/// Вывод на консоль RealMatrix.\_arr

/// </summary>

public void Write()

{

for (int i = 0; i <= \_arr.GetUpperBound(0); i++)

{

for (int j = 0; j < \_arr.GetUpperBound(1) + 1; j++)

{

Console.Write($" {\_arr[i, j]}\t");

}

Console.WriteLine("");

}

}

/// <summary>

/// Создание массива вещественных чисел и заполнением его рандомными значениями

/// </summary>

/// <param name="sizeI">кол-во строк</param>

/// <param name="sizeJ">кол-во столбцов</param>

/// <returns></returns>

private double[,] RandomArr(int sizeI, int sizeJ)

{

double[,] newArr = new double[sizeI, sizeJ];

Random random = new Random();

for (int i = 0; i < sizeI; i++)

{

for (int j = 0; j < sizeJ; j++)

{

newArr[i, j] = random.Next(10);

}

}

return newArr;

}

}

}