1. **Документирование в формате XML**

Задание 1. В темах: 3 (Процедуры и функции – методы класса), выполнить XML документирование кода.

Листинг программы:

namespace Task\_2

{

/// <summary>

/// Главный класс.

/// Выполнение метода Main.

/// </summary>

class Task\_2

{

/// <summary>

/// Главный метод.

/// Вводятся значения.

/// Создается ссылка на класс Program.

/// Выполняется цикл вывода значений результата функции.

/// </summary>

/// <param name="args">Можно передать массив со строками</param>

public static void Main(string[] args)

{

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

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

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

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

Console.Write("Введите шаг: ");

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

Program pr = new Program();

foreach (double num in pr.F(start, finish, step))

{

Console.WriteLine($"F(x)= {num}");

}

}

}

/// <summary>

/// Класс высчитывания функции

/// </summary>

class Program

{

/// <summary>

/// Функция вычисления значений на отрезке с завданым шагом.

/// </summary>

/// <param name="start">Начальное значение отрезка</param>

/// <param name="finish">Конечное значение отрезка</param>

/// <param name="h">Шаг передвижения по отрезку</param>

/// <returns>Возвращает List типа double.</returns>

public List<double> F(double start, double finish, double h)

{

List<double> list = new List<double>();

for (double i = start; i <= finish; i += h)

{

if (i < 0)

{

list.Add(0);

}

else if (i > 0 && i != 1)

{

list.Add(Math.Pow(i, 2) +1);

}

else

{

list.Add(1);

}

}

return list;

}

}

}

Таблица 1.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| -4,8 5 1,8 | 0 0 0 1,8099... 8,2900… 21,25 |

Анализ результатов:

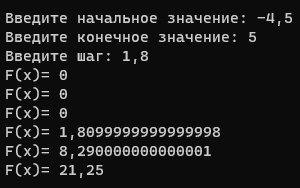


Рисунок 1.1 – Результат работы программы

Источник: собственная разработка

Задание 2. В темах: 3 (Процедуры и функции – методы классавыполнить XML документирование кода.

Листинг программы:

namespace Task\_4

{

/// <summary>

/// Главный класс выполнения программы методом Main.

/// </summary>

class Task\_4

{

/// <summary>

/// Главный метод выполнения программы.

/// Вводится размерность двумерного массива.

/// Создается ссылка на класс Array и выполняется два действия,

/// первое с конструктором без параметров;

/// второе с конструктором с параметрами.

/// </summary>

/// <param name="args">Можно передать массив со строками</param>

public static void Main(string[] args)

{

Console.Write("Введите размерность двумерного массива:\nСтрок= ");

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

Console.Write("Колонок= ");

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

Console.WriteLine("Action One");

Array array = new Array(row, colum);

InputNumInMas(array, row, colum);

Console.WriteLine($"Перемножением положительных элементов массива, меньших 10: {array.ArrayAction()}");

Console.WriteLine("\nAction Two");

Array ar = new Array();

ar.CreateArrayMas(row, colum);

InputNumInMas(ar, row, colum);

int[,] mas = ar.GetArray();

Console.WriteLine($"Перемножением положительных элементов массива, меньших 10: {ar.ArrayAction(mas)}");

}

/// <summary>

/// Метод ввода значений в двумерный массив.

/// </summary>

/// <param name="array">Ссылка на класс Array.</param>

/// <param name="row">Параметр количества строк в массиве.</param>

/// <param name="colum">Параметр количества колонок в массиве.</param>

private static void InputNumInMas(Array array, int row, int colum)

{

Console.WriteLine("Введите значения массива: ");

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

{

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

{

Console.Write(i + "," + j + "=");

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

array.SetNumArray(num, i, j);

}

}

}

}

/// <summary>

/// Класс обработки и создания массива типа Array.

/// </summary>

public class Array

{

/// <summary>

/// Закрытый, не инициализированный двумерный массив.

/// </summary>

private int[,] masDuo;

/// <summary>

/// Конструктор без параметров.

/// </summary>

public Array()

{

}

/// <summary>

/// Конструктор с параметрами.

/// </summary>

/// <param name="row">Параметр количества строк в массиве.</param>

/// <param name="colum">Параметр количества колонок в массиве.</param>

public Array(int row, int colum)

{

masDuo = new int[row, colum];

}

/// <summary>

/// Метод добавления значения в двумерный массив.

/// </summary>

/// <param name="x">Параметр добавляемого значения</param>

/// <param name="i">Параметр номера строки</param>

/// <param name="j">Параметр номера колонки</param>

public void SetNumArray(int x, int i, int j)

{

masDuo[i, j] = x;

}

/// <summary>

/// Метод инициализации двумерного массива.

/// </summary>

/// <param name="row">Параметр количества строк в массиве.</param>

/// <param name="colum">Параметр количества колонок в массиве.</param>

public void CreateArrayMas(int row, int colum)

{

masDuo = new int[row, colum];

}

/// <summary>

/// Метод возвращения двумерного массива.

/// </summary>

/// <returns>Возаращает двумерный массив типа int.</returns>

public int [,] GetArray()

{

return masDuo;

}

/// <summary>

/// Метод суммирования значений двумерного массива в диапазоне 0&lt;i&lt;10.

/// </summary>

/// <returns>Возаращает сумму значений типа int.</returns>

public int ArrayAction()

{

int result = 0;

foreach (var i in masDuo)

{

if (i >= 0 && i < 10)

{

result += i;

}

}

return result;

}

/// <summary>

/// Метод суммирования значений двумерного массива в диапазоне 0&lt;i&lt;10.

/// </summary>

/// <param name="mas">Параметр передаваемого массива</param>

/// <returns>Возаращает сумму значений типа int.</returns>

public int ArrayAction(int[,] mas)

{

int result = 0;

foreach (var i in mas)

{

if (i >= 0 && i < 10)

{

result += i;

}

}

return result;

}

}

}

Таблица 2.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 2 4 (2 -5 -3 -6 -4 -6 -2 0) (2 2 2 5 45 7 -100 23) | 2 18 |

Анализ результатов:

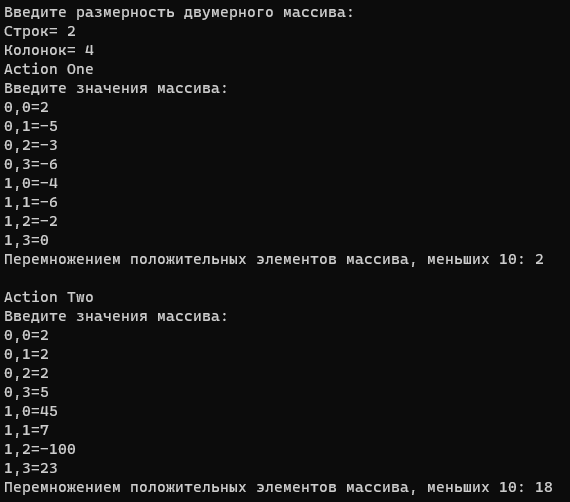


Рисунок 2.1 – Результат работы программы

Источник: собственная разработка

Задание 3. В темах: 12 (Делегаты) выполнить XML документирование кода.

Листинг программы:

namespace Task\_1

{

/// <summary>

/// Главный класс выполнения программы методом Main.

/// </summary>

class Program

{

/// <summary>

/// Главный метод программы реализующий вычисления окружности.

/// </summary>

/// <param name="args">Можно передать массив со строками</param>

public static void Main(string[] args)

{

CalkFigur cf = new CalkFigur(GetLength);

cf += GetArea;

cf += GetVolume;

Console.Write("Введите радиус: ");

cf(double.Parse(Console.ReadLine()));

}

/// <summary>

/// Делегат вызывающий методы вычисления длинны, площади и обьема окружности.

/// </summary>

/// <param name="radius">Параметр радиуса окружности.</param>

/// <returns></returns>

public delegate double CalkFigur(double radius);

/// <summary>

/// Метод вычисления длинны окружности.

/// </summary>

/// <param name="r">Параметр радиуса окружности.</param>

/// <returns></returns>

static double GetLength(double r)

{

double D = 2 \* Math.PI \* r;

Console.WriteLine($"Длинна окружности: {D}");

return D;

}

/// <summary>

/// Метод вычисления площади окружности.

/// </summary>

/// <param name="r">Параметр радиуса окружности.</param>

/// <returns></returns>

static double GetArea(double r)

{

double S = Math.PI \* Math.Pow(r, 2);

Console.WriteLine($"Площадь круга: {S}");

return S;

}

/// <summary>

/// Метод вычисления обьема окружности.

/// </summary>

/// <param name="r">Параметр радиуса окружности.</param>

/// <returns></returns>

static double GetVolume(double r)

{

double V = (4 / 3) \* Math.PI \* Math.Pow(r, 3);

Console.WriteLine($"Обьем шара: {V}");

return V;

}

}

}

Таблица 3.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 2 | 12,56… 12,56… 25,13… |

Анализ результатов:

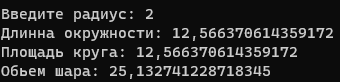


Рисунок 3.1 – Результат работы программы

Источник: собственная разработка

Задание 4. В темах: 12 (Делегаты) выполнить XML документирование кода.

Листинг программы:

namespace Task\_3

{

/// <summary>

///

/// </summary>

/// <param name="str"></param>

public delegate void StringOperation(string[] str);

/// <summary>

/// Главный класс выполнения программы методом Main.

/// </summary>

class Program

{

/// <summary>

/// Главный метод программы реализующий действия над солвами.

/// </summary>

/// <param name="args">Можно передать массив со строками</param>

public static void Main(string[] args)

{

try

{

Console.Write("Введите слово затем что заменить в слове на что: ");

string[] array = Console.ReadLine().Split(" ").ToArray();

StringOperation oper = new StringOperation(StringReplace);

oper(array);

Console.Write("Введите слово затем индекс и то что надо вставить: ");

array = Console.ReadLine().Split(" ").ToArray();

oper = StringInsert;

oper(array);

Console.Write("Введите слово затем индекс с какого момента произвести удаление\nи сколько удалить: ");

array = Console.ReadLine().Split(" ").ToArray();

oper = StringDelete;

oper(array);

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

}

/// <summary>

/// Метод заменяющий часть слова на другое слово.

/// </summary>

/// <param name="str">Параметр массив из слов.</param>

public static void StringReplace(string[] str)

{

Console.WriteLine(str[0].Replace(str[1], str[2]));

}

/// <summary>

/// Метод вставляющий на заданный индекс другое слово.

/// </summary>

/// <param name="str">Параметр массив из слов.</param>

public static void StringInsert(string[] str)

{

Console.WriteLine(str[0].Insert(int.Parse(str[1]), str[2]));

}

/// <summary>

/// Метод удаления части слова или предложения.

/// </summary>

/// <param name="str">Параметр массив из слов.</param>

public static void StringDelete(string[] str)

{

Console.WriteLine(str[0].Remove(int.Parse(str[1]), int.Parse(str[2])));

}

}

}

Таблица 4.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Пастух ух ила  Пожар 1 р  Пастух 3 2 | Пастила  Прожар  Пасх |

Анализ результатов:

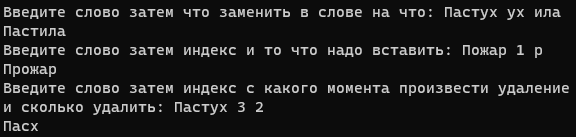


Рисунок 4.1 – Результат работы программы

Источник: собственная разработка

Задание 5. В темах: 16 (Работа с файлами) выполнить XML документирование кода.

Листинг программы:

namespace Task\_1

{

/// <summary>

/// Главный класс выполнения программы методом Main.

/// </summary>

class Program

{

/// <summary>

/// Главный метод вывода в консоль суммы наибольшего и наименьшего значений файла.

/// </summary>

/// <param name="args">Можно передать массив со строками</param>

static void Main(string[] args)

{

string path = "..\\..\\..\\f.txt";

FileStream file = new FileStream(@path, FileMode.Open);

StreamReader reader = new StreamReader(file);

string s;

int n = 0;

while ((s = reader.ReadLine()) != null)

{

n++;

}

reader.Close();

Console.WriteLine($"Сумма наибольшего и наименьшего значений компонент: " +

$"{SumMaxMinFromArray(FillArrayFromFile(n, path))}");

}

/// <summary>

/// Метод подсчитывает сумму максимального и минимального значения.

/// </summary>

/// <param name="array">Параметр массив.</param>

/// <returns></returns>

public static double SumMaxMinFromArray(double[] array)

{

return array.Max() + array.Min();

}

/// <summary>

/// Метод считывания файла в массив.

/// </summary>

/// <param name="n">Параметр еоличества строк массива.</param>

/// <param name="path">Параметр пути к файлу.</param>

/// <returns></returns>

public static double[] FillArrayFromFile(int n, string path)

{

FileStream file1 = new FileStream(@path, FileMode.Open);

StreamReader reader1 = new StreamReader(file1);

double[] array = new double[n];

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

{

array[i] = double.Parse(reader1.ReadLine());

}

file1.Close();

return array;

}

}

}

Таблица 5.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Путь к файлу | 2337,65655 |

Анализ результатов:



Рисунок 5.1 – Результат работы программы

Источник: собственная разработка

Задание 6. В темах: 16 (Работа с файлами) выполнить XML документирование кода.

Листинг программы:

namespace Task\_5

{

/// <summary>

/// Главный класс выполнения программы методом Main.

/// </summary>

class Program

{

/// <summary>

/// Главный метод программы реализующий действия над файлвми.

/// </summary>

/// <param name="args">Можно передать массив со строками</param>

public static void Main(string[] args)

{

string fileNaameOne = "f1.txt";

string fileNaameTwo = "f2.txt";

string fileNaameThree = "f3.txt";

Console.WriteLine($"Файл {fileNaameOne}");

string pathOne = CreateNewFile(fileNaameOne);

SetInFile(pathOne);

OutputFile(pathOne);

Console.WriteLine($"Файл {fileNaameTwo}");

string pathTwo = CreateNewFile(fileNaameTwo);

SetInFile(pathTwo);

OutputFile(pathTwo);

Console.WriteLine($"Файл {fileNaameThree}");

string pathThree = CreateNewFile(fileNaameThree);

Generalization(pathOne, pathTwo, pathThree);

SortedFile(pathThree);

OutputFile(pathThree);

}

/// <summary>

/// Метод соединения значений из двух файлов в третий.

/// </summary>

/// <param name="pathOne">Название первого файла</param>

/// <param name="pathTwo">Название второго файла</param>

/// <param name="pathWriter">Параметр пути к файлу записи</param>

public static void Generalization(string pathOne, string pathTwo, string pathWriter)

{

string[] file1 = File.ReadAllLines(pathOne);

string[] file2 = File.ReadAllLines(pathTwo);

StreamWriter writer = File.CreateText(pathWriter);

int lineNum = 0;

while (lineNum < file1.Length || lineNum < file2.Length)

{

if (lineNum < file1.Length)

writer.WriteLine(file1[lineNum]);

if (lineNum < file2.Length)

writer.WriteLine(file2[lineNum]);

lineNum++;

}

writer.Close();

}

/// <summary>

/// Метод создания файла и присвоения ему названия.

/// </summary>

/// <param name="fileName">Параметр название файла</param>

/// <returns>Возвращает путь к файлу.</returns>

public static string CreateNewFile(string fileName)

{

string path = @"..\..\..\..\" + fileName;

FileStream file = new FileStream(path, FileMode.Create, FileAccess.Write);

file.Close();

return path;

}

/// <summary>

/// Метод генерации списка чисел.

/// </summary>

/// <returns>Возвращает список значений.</returns>

public static List<int> GenerationIntegerNumbers()

{

Random rnd = new Random();

List<int> array = new List<int>();

for (int i = 0; i < 10; i++)

{

array.Add(rnd.Next(100));

}

array.Sort();

return array;

}

/// <summary>

/// Метод добавления в файл значений из списка.

/// </summary>

/// <param name="path">Параметр пути к файлу.</param>

public static void SetInFile(string path)

{

FileStream file = new FileStream(@path, FileMode.Open);

StreamWriter writer = new StreamWriter(file);

List<int> array = GenerationIntegerNumbers();

for (int i = 0; i < array.Count; i++)

writer.WriteLine(array[i]);

writer.Close();

}

/// <summary>

/// Метод вывода значений из файла.

/// </summary>

/// <param name="path">Параметр пути к файлу.</param>

public static void OutputFile(string path)

{

FileStream file = new FileStream(@path, FileMode.Open);

StreamReader reader = new StreamReader(file);

string line = "";

while ((line = reader.ReadLine()) != null)

{

Console.Write(line + " ");

}

Console.WriteLine();

reader.Close();

}

/// <summary>

/// Метод сортировки значений файла.

/// </summary>

/// <param name="path">Параметр пути к файлу.</param>

public static void SortedFile(string path)

{

List<string> lines = File.ReadAllLines(path).ToList();

List<int> list = lines.Select(int.Parse).ToList();

list.Sort();

StreamWriter writer = new StreamWriter(path);

for (int i = 0; i < list.Count; i++)

writer.WriteLine(list[i]);

writer.Close();

}

}

}

Таблица 6.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  |  |

Анализ результатов:

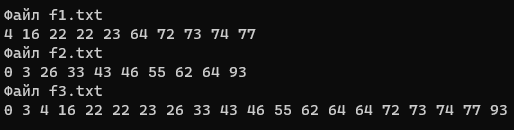


Рисунок 6.1 – Результат работы программы

Источник: собственная разработка