**19 Документирование в формате xml**

Задание 1. 3 (Процедуры и функции – методы класса);

12 (Делегаты);

16 (Работа с файлами) выполнить XML документирование кода.

Листинг программы 3:

class Program

{

/// <summary>

/// Вычисление функции

/// </summary>

/// <param name="arg"></param>

/// <returns></returns>

static int func(int arg)

{ return (arg / 10) % 10; }

/// <summary>

/// Метод Main

/// </summary>

/// <param name="args"></param>

static void Main(string[] args)

{

int a = 123456, b = 345678, c = 987694, result;

result = func(a) + func(b) - func(c);

Console.Write($" Сумма вторых справа цифр = { result } ");

Console.ReadKey();

}

}

Таблица 19.1 – Входные и выходные данные

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

Анализ результатов:



Рисунок 19.1 – Результат работы программы

Листинг программы 12:

class Program

{

static void Main()

{

Console.WriteLine("Введите действие: \n1 Удалить первое слово \n2 Замена всех букв о на ОК \n3 Удаление точек ");

string a = Console.ReadLine();

Console.WriteLine("Введите строку: ");

string x = Console.ReadLine();

switch (a)

{

case "1":

Stroka calc = (c) => c;

string del = x.Remove(0, x.IndexOf(' ') + 1);

Console.WriteLine("Ответ:{0}", del);

break;

case "2":

Stroka calc1 = (c) => c;

string del1 = x.Replace("о", "ОК");

Console.WriteLine("Ответ:{0}", del1);

break;

case "3":

Stroka calc2 = (c) => c;

string zamena = x.Replace(".", "");

Console.WriteLine("Ответ:{0}", zamena);

break;

}

Console.ReadKey();

}

}

Таблица 19.2 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 1  sun rise s | rise s |

Анализ результатов:

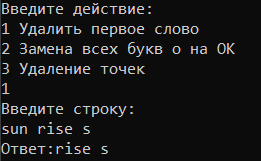


Рисунок 19.2 – Результат работы программы

Листинг программы 16:

class Program

{

static void Main(string[] args)

{

var directory = Directory.CreateDirectory("D:\\Folder16");

var file = new FileInfo($"{directory.FullName}\\File1.txt");

if (!file.Exists)

{

file.Create();

}

FillFile(filePath: file.FullName);

int count = default(int);

using (StreamReader reader = new StreamReader(file.FullName, Encoding.UTF8))

{

string s;

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

{

count++;

}

}

try

{

var numbers = ReadFile(filePath: file.FullName);

var minNumber = numbers.Min();

Console.WriteLine("Минимальное число в файле: " + minNumber);

var countPositiveNumbers = numbers.Count(number => number > 0);

Console.WriteLine("Количество положительных чисел в файле: " + countPositiveNumbers);

}

catch (FormatException ex)

{

Console.WriteLine(ex.Message);

}

Console.ReadLine();

}

/// <summary>

/// Запись в файл

/// </summary>

/// <param name="filePath"></param>

/// <param name="minValue"></param>

/// <param name="maxValue"></param>

/// <param name="countNumbers"></param>

private static void FillFile(string filePath, int minValue = -5, int maxValue = 6, int countNumbers = 10)

{

var random = new Random();

using (var writer = new StreamWriter(filePath, append: false, encoding: Encoding.UTF8))

{

while (countNumbers > 0)

{

writer.WriteLine(random.Next(minValue, maxValue));

countNumbers--;

}

}

}

/// <summary>

/// Чтение файла

/// </summary>

/// <param name="filePath"></param>

/// <returns></returns>

private static IEnumerable<int> ReadFile(string filePath)

{

using (var reader = new StreamReader(filePath, Encoding.UTF8))

{

var input = String.Empty;

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

{

yield return int.Parse(input);

}

}

}

}

Таблица 19.3 – Входные и выходные данные

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

Анализ результатов:



Рисунок 19.3 – Результат работы программы