**Отчёт по лабораторной работе №10**

**По дисциплине:** Технологии программирования

**Выполнил студент группы:** ИВТ-б-о-21-1

**Ф.И.О.:** Мальцев Николай Артёмович

**Номер индивидуального варианта:** 11

**Название лабораторной работы:** Основы работы с файлами.

**I. Ход выполнения работы:**

**Проработка примеров:**

Листинг программы:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Лабораторная\_10

{

public class Matrix

{

private float[,] matrix;

int m, n;

// Конструктор

public Matrix() { }

// Генерация матрицы заданного размера

public void GenerateMatrix(int M, int N)

{

m = M; n = N;

Random r = new Random(DateTime.Now.Millisecond);

matrix = new float[M, N];

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

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

matrix[i, j] = (float)r.Next(1000) / 973f;

}

// Сохранение сгенерированной матрицы в файл

public void SaveMatrix(string pFileName)

{

if (matrix.Length > 0)

{

if (File.Exists(pFileName))

File.Delete(pFileName);

FileInfo f = new FileInfo(pFileName);

TextWriter tw = f.CreateText();

tw.WriteLine(m.ToString());

tw.WriteLine(n.ToString());

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

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

tw.WriteLine(i.ToString() + " " + j.ToString() + " " + matrix[i, j].ToString("E10"));

tw.Close();

}

}

// Загрузка сохранённой матрицы из файла

public Boolean LoadMatrix(string pFileName)

{

if (File.Exists(pFileName))

{

try

{

TextReader tr = File.OpenText(pFileName);

m = Convert.ToInt32(tr.ReadLine());

n = Convert.ToInt32(tr.ReadLine());

matrix = new float[m, n];

string line;

string[] substring;

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

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

{

line = tr.ReadLine();

substring = line.Split(new char[] { ' ' }, 3);

matrix[i, j] = Convert.ToSingle(substring[2]);

}

tr.Close();

return true;

}

catch

{

return false;

}

}

return false;

}

public void PrintMatrix()

{

if (matrix.Length > 0)

{

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

{

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

Console.Write(matrix[i, j].ToString("E3") + " ");

Console.WriteLine();

}

}

}

}

class Program

{

static void Main(string[] args)

{

Console.BackgroundColor = ConsoleColor.White;

Console.ForegroundColor = ConsoleColor.Black;

Console.Clear();

Matrix m = new Matrix();

m.GenerateMatrix(10, 5);

m.SaveMatrix("Matrix.txt");

if (m.LoadMatrix("Matrix.txt"))

{

m.PrintMatrix();

}

Console.ReadKey();

}

}

}

Результат работы программы в консоли:

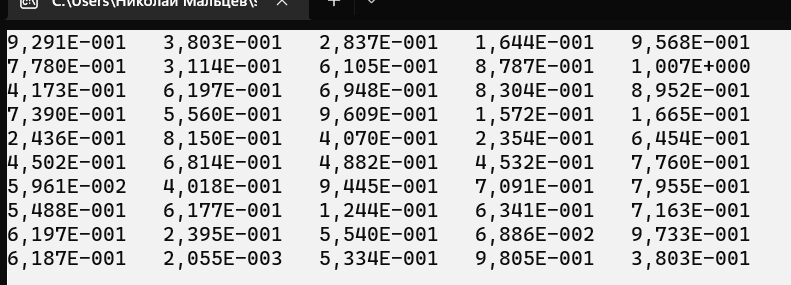


Рисунок 1. Результат работы программы в консоли

Результат работы программы в файле:

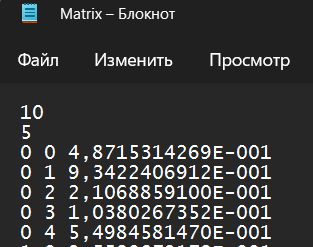


Рисунок 2. Файл Matrix после выполнения программы

**Индивидуальное задание:**

Листинг программы:

using System;

using System.IO;

using System.Security.Cryptography.X509Certificates;

using System.Xml.Linq;

namespace Lab\_10\_und

{

public class Matrix

{

private float[,] matrix;

int m, n;

string file { get; set; }

// Конструктор

public Matrix(int m, int n)

{

this.m = m;

this.n = n;

matrix = new float[m, n];

}

// Генератор матриц заданного размера

public void GeneratMatr(int M, int N)

{

m = M; n = N;

Random r = new Random(DateTime.Now.Millisecond);

matrix = new float[M, N];

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

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

matrix[i, j] = (float)r.Next(100);

}

// Сохранение сгенерированной матрицы из файла

public void SaveMatr(string pFileName)

{

if (matrix.Length > 0)

{

if (File.Exists(pFileName))

File.Delete(pFileName);

FileInfo f = new FileInfo(pFileName);

TextWriter tw = f.CreateText();

tw.WriteLine(m.ToString());

tw.WriteLine(n.ToString());

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

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

tw.WriteLine(i.ToString() + " " + j.ToString() + " " + matrix[i, j].ToString());

tw.Close();

}

}

// Загрузка сохранённой матрицы из файла

public Boolean LoadMatr(string pFileName)

{

if (File.Exists(pFileName))

{

try

{

TextReader tr = File.OpenText(pFileName);

m = Convert.ToInt32(tr.ReadLine());

n = Convert.ToInt32(tr.ReadLine());

matrix = new float[m, n];

string line;

string[] substring;

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

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

{

line = tr.ReadLine();

substring = line.Split(new char[] { ' ' }, 3);

matrix[i, j] = Convert.ToSingle(substring[2]);

}

tr.Close();

return true;

}

catch

{

return false;

}

}

return false;

}

// Вывод маттрицы на консоль

public void PrintMatr()

{

Console.WriteLine("\n");

if (matrix.Length > 0)

{

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

{

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

{

Console.Write(matrix[i, j].ToString() + " ");

}

Console.WriteLine();

}

}

}

public float SumElement(Matrix pMatr\_1, Matrix pMatr\_2)

{

float sum = 0;

for (int i = 1; i < pMatr\_1.m; i=i+2)

{

for (int j = 0; j < pMatr\_1.n; j=j+2)

{

sum += pMatr\_1.matrix[i, j] + pMatr\_2.matrix[i, j];

}

}

return sum;

}

// Перегрузка оператора, не нужна в этой работе, сделал по приколу 😎

public static Matrix operator +(Matrix pMatr\_1, Matrix pMatr\_2)

{

Matrix tempMatr = new Matrix(pMatr\_1.m, pMatr\_1.n);

for (int i = 0; i < pMatr\_1.m; i++)

{

for (int j = 0; j < pMatr\_1.n; j++)

{

tempMatr.matrix[i, j] = pMatr\_1.matrix[i, j] + pMatr\_2.matrix[i, j];

}

}

return tempMatr;

}

}

class Program

{

static void Main(string[] args)

{

int m, n;

// Ввод размера матрицы

Console.Write("Введите m >> ");

m = Convert.ToInt32(Console.ReadLine());

Console.Write("Введите n >> ");

n = Convert.ToInt32(Console.ReadLine());

float[,] matr = new float[m, n];

Matrix first = new Matrix(m, n);

Matrix second = new Matrix(m, n);

Matrix third = new Matrix(m, n);

first.GeneratMatr(m, n);

first.SaveMatr("First\_matrix.txt");

second.GeneratMatr(m, n);

second.SaveMatr("Second\_matrix.txt");

if (first.LoadMatr("First\_matrix.txt") & second.LoadMatr("Second\_matrix.txt"))

{

first.PrintMatr();

second.PrintMatr();

// third = first + second; // Для проверки работы оператора

Console.WriteLine("\n" + third.SumElement(first, second));

// third.SaveMatr("Rezult\_matrix.txt"); // Для проверки работы оператора

// third.PrintMatr(); // Для проверки работы оператора

}

Console.ReadKey();

}

}

}

Результат работы программы:

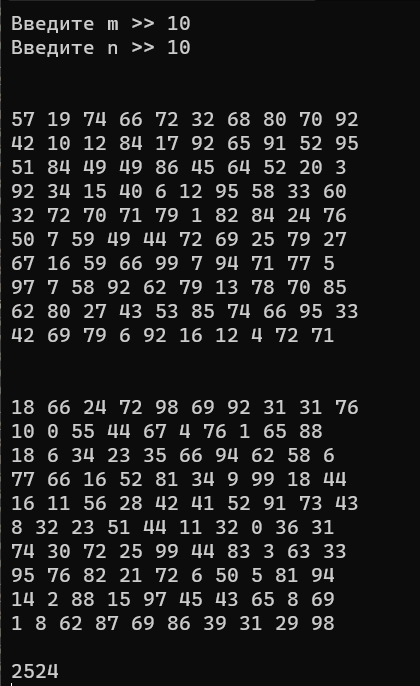


Рисунок 3. Результат работы программы в консоли

**II. Ответы на контрольные вопросы:**

**1. Какие классы для работы с файловой системой вы знаете?**

File - статический класс для работы с файлами, Directory – статический класс для работы с директориями, Path – класс для манипулирования путевыми именами, FileInfo – класс для представления физического файла на диске, имеет методы для манипуляции эти файлом.

**2. Что такое сборка?**

Сборка — это скомпилированный результат кода, который содержит все необходимые для работы программы файлы.

**3. Как определить проект по умолчанию в решении Visual Studio?**

Для этого необходимо правой кнопкой мыши нажать на необходимом проекте и во всплывающем меню выбрать пункт «Выбрать в качестве запускаемого проекта».

**4. Какие классы отвечают за представление файлов в программе?**

Классы File и FileInfo. Их отличие заключается в том, что первый является статическим, а второй динамическим. На практике это отличие проявляется в том, что для методов для методов второго необходимо передавать больше параметров. Так, например, FileInfo.Copy() принимает два параметра – путь к файлу, который нужно скопировать и путь к файлу, где должна создаться копия. Для File.Copy() необходим только один параметр – путь, куда необходимо скопировать файл.

**5. Что такое поток? Какие типы классов потоков используются при работе с файлами?**

Из документации Microsoft: «Поток – это последовательность байт, содержащая данный, записываемые в файл».

В С# существует абстрактный базовый класс Stream, который отвечает за чтение и запись байтов. От него наследуются такие классы, как FileStream – для чтения и записи в файл; MemoryStream – для чтения и записи в память; BufferedStream – для повышения быстродействия операций чтения и записи; CryptoStream – для связи потоков данных с криптографическими преобразованиями и др.

**6. Опишите последовательность действий при необходимости записать одну строку в файл. Приведите примеры использования различных классов.**

Для начал необходимо создать экземпляр класса FileInfo, передав в качестве параметра название файла, в который необходимо записать строку, далее создаём поток в созданный файл при помощи класса StreamWriter, при помощи Console.Write() выводим необходимую строку в файл.

**7. Перечислите классы для работы с каталогами.**

File, FileInfo, Directory и DirectoryInfo.

**8. Поясните принцип работы синтаксической конструкции try…catch.**

В случае, если при выполнении кода, помещённого в «блок» try происходит исключение (exception), то код в «блоке» try перестает работать с той строчки кода, где возникло это исключение и начинается выполнение кода в «блоке» catch.

**Дата: 10.12.2022**