**Министерство науки и высшего образования российской федерации**

**ФГАОУ ВО «Пермский государственный национальный исследовательский университет»**

Физический факультет

Технологии .NET Framework

Лабораторная работа №8 на тему

**«****Файловый ввод-вывод. Работа с каталогами. Работа с файлами»**

Выполнил:

студент 2 курса физического факультета

группа ПМИ-2-2021

направление «01.03.02 Прикладная математика и информатика»

профиль «Инженерия программного обеспечения»

Макарова Полина Фёдоровна

Пермь, 2023 г.

1. Цель работы

• научиться использовать механизмы файлового ввода-вывода;

• научиться применять классы для работы с файлами;

• научиться применять классы для работы с каталогами;

• научиться использовать потоки ввода-вывода.

1. **Постановка задачи.**

Необходимо спроектировать многомодульное приложение, использующее файлы для ввода и вывода информации.

Исходные данные программа получает из входного файла. Требуется разработать программу, которая рассчитывает сумму элементов чётных строк в двух матрицах, которые хранятся в разных файлах.

Программа выполняет действия:

— считывать матрицу из файла;

— выводить матрицу на экран;

— рассчитывать сумму элементов чётных строк в двух матрицах (т.е. все элементы нужных строк складываются между собой и прибавляется по такому же методу сумма у второй матрицы). Выводится результат сложения на экран.

1. **Декомпозиция задачи**

3.1. Осуществляем логическую декомпозицию данного приложения. В данном случае:

— чтение из файла;

— вывод на экран;

— рассчитывание суммы и вывод на экран.

3.2. Осуществляем физическую декомпозицию данного приложения. В данном случае:

— 1-ый модуль в виде файла \*.exe для запуска приложения;

— 2-ой модуль в виде файла \*.dll, содержащий методы рассчитывания суммы и вывод на экран;

— 3-ий модуль в виде файла \*.dll, содержащий методы чтения из файла и вывод на экран.

Декомпозиция, логическая и физическая, завершена.

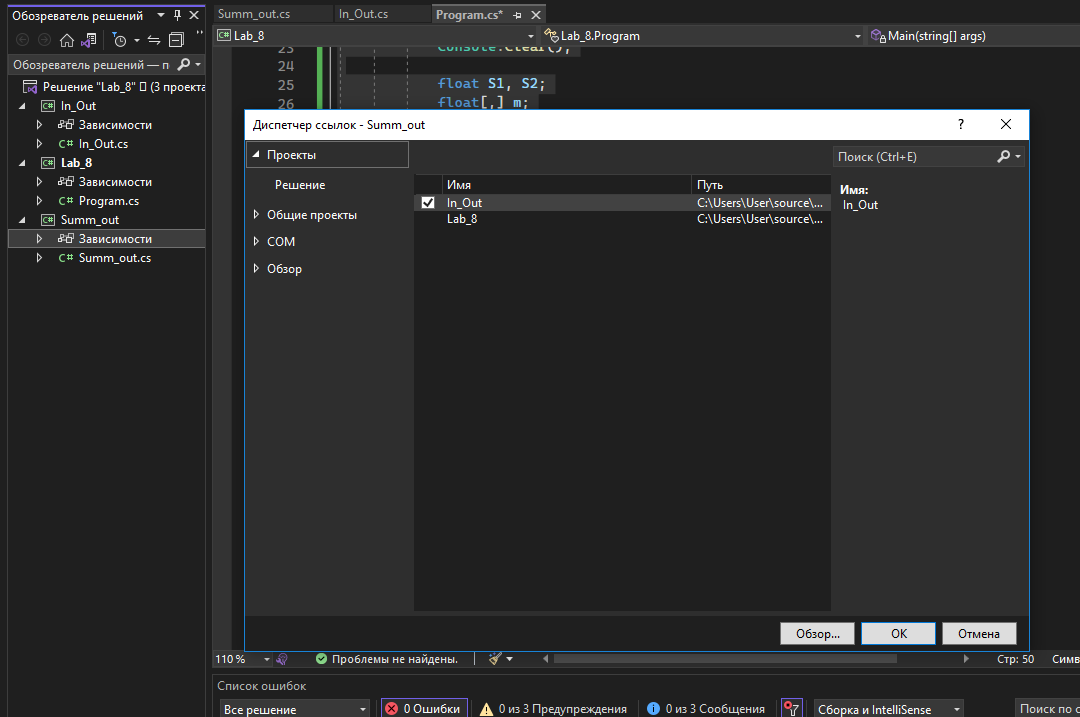


Рисунок 1 — Ссылки на данные проекта 1

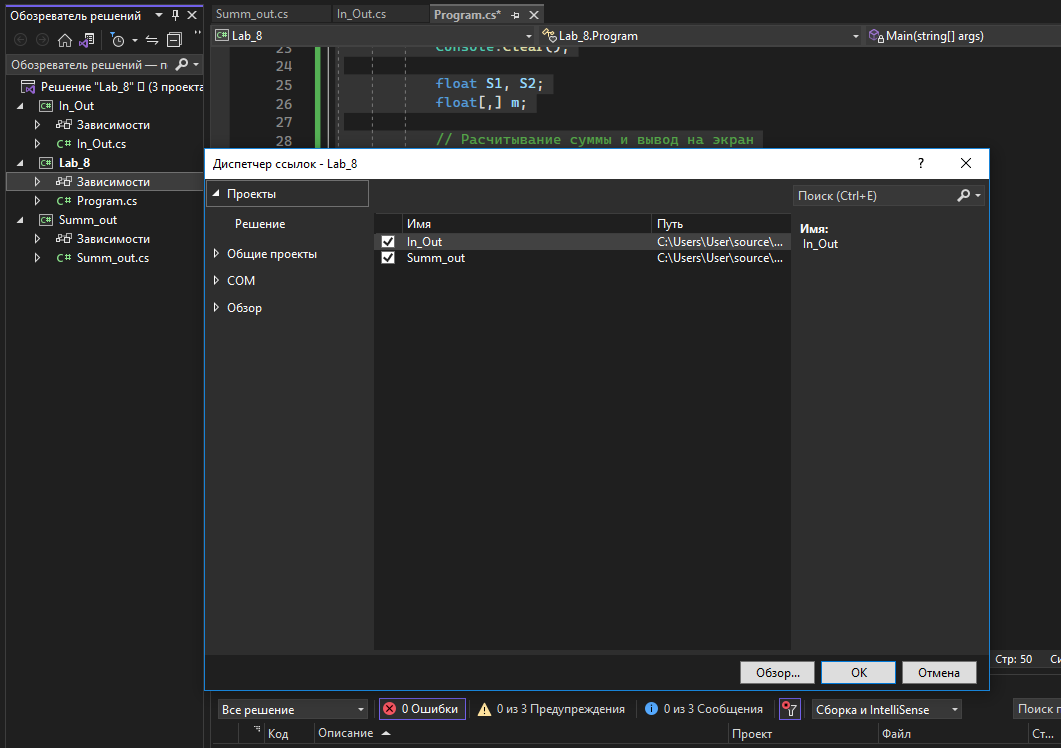


Рисунок 2 — Ссылки на данные проекта 2

1. Листинг программы

|  |
| --- |
| *Листинг 1 программы Program* |
| /\* Макарова Полина ПМИ-2  \* Вариант 10  \* Лабораторная работа 8  \*/  /\* В каждом варианте необходимо спроектировать многомодульное приложение (минимум 2 модуля).  \* Исходные данные в каждом варианте программа получает из входного файла.  \*  \* Программа рассчитывает сумму элементов чётных строк в двух матрицах,  \* которые хранятся в разных файлах.  \*/  using System;  namespace Lab\_8  {  internal class Program  {  private static void Main(string[] args)  {  Console.BackgroundColor = ConsoleColor.White;  Console.ForegroundColor = ConsoleColor.Black;  Console.Clear();  float S1, S2;  float[,] m;  // Расчитывание суммы и вывод на экран  Summ\_out.Summ\_out s = new Summ\_out.Summ\_out();  In\_Out.In\_Out m1 = new In\_Out.In\_Out();  // Загружаем матрицу из файла  m = m1.LoadMatrix("FileMatrix1.txt");  // Выводим матрицу из файла  m1.PrintMatrix(1);  // Находим сумму элементов чётных строк матрицы  S1 = s.Summ(m);  m = m1.LoadMatrix("FileMatrix2.txt");  m1.PrintMatrix(2);  S2 = s.Summ(m);    // Находим общую сумму элементов двух матриц  s.Print\_Summ(S1, S2);  DirectoryInfo dir1 = new DirectoryInfo(@"C:\Users\User\source\repos\Lab\_8\Lab\_8\bin\Debug\net6.0");  if (dir1.Exists)  {  Console.WriteLine("\nDirectoryInfo dir1:");  Console.WriteLine("FullName: {0}", dir1.FullName);  Console.WriteLine("Name: {0}", dir1.Name);  Console.WriteLine("Parent: {0}", dir1.Parent);  Console.WriteLine("CreationTime: {0}", dir1.CreationTime);  Console.WriteLine("Attributes: {0}", dir1.Attributes);  Console.WriteLine("Root: {0}", dir1.Root);  Console.WriteLine();  }  // Создаём подкаталог  dir1.CreateSubdirectory("In");  // Создаём вложенный каталог  dir1.CreateSubdirectory(@"Out\3");  Console.ReadKey();  }  }  } |

|  |
| --- |
| *Листинг 2 программы In\_Out* |
| using System;  using System.IO;  namespace In\_Out  {  public class In\_Out  {  int m, n;  float[,] matrix;  // Загрузка матрицы из файла  public float[,] LoadMatrix(string pathFileName)  {  if (File.Exists(pathFileName))  {  try  {  TextReader textReader = File.OpenText(pathFileName);  m = Convert.ToInt32(textReader.ReadLine());  n = Convert.ToInt32(textReader.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 = textReader.ReadLine();  substring = line.Split(new char[] { ' ' }, 3);  matrix[i, j] = Convert.ToSingle(substring[2]);  }  }  textReader.Close();  return matrix;  }  catch  {  return matrix;  }  }  return matrix;  }  // Вывод матрицы в консоль  public void PrintMatrix(int g)  {  if (matrix.Length > 0)  {  Console.WriteLine("Матрица " + g + ":\n");  for (int i = 0; i < m; i++)  {  for (int j = 0; j < n; j++)  {  Console.Write(matrix[i, j].ToString("E3") + " ");  }  Console.WriteLine();  }  Console.WriteLine();  }  }  }  } |

|  |
| --- |
| *Листинг 3 программы Summ\_out* |
| namespace Summ\_out  {  public class Summ\_out  {  int m, n;  float S;  // Конструктор  public Summ\_out()  {  m = 0;  n = 0;  S = 0;  }  // Рассчитывание суммы чётных строк матрицы  public float Summ(float[,] matrix) //In\_Out.In\_Out m1  {  S = 0;  // Узнаём размер матрицы  m = matrix.GetLength(0); //Для [i]  n = matrix.GetLength(1); // для [j]  Console.WriteLine("Размер матрицы: " + m + " , " + n);  // Перебираем эллементы матрицы  if (matrix.Length > 0)  {  Console.WriteLine(" Чётные строки матрицы:\n");  // Строки считаються с 1, а не с 0.  // Т.е. 0 строка в матрице это 1 строка в представлении.  // => т.к. 0 строка это 1, то она нечётна, значит начинаем считать со строки 1(2).  for (int i = 1; i < m; i+=2)  {  for (int j = 0; j < n; j++)  {  Console.Write(matrix[i, j].ToString("E3") + " ");  S += matrix[i, j];  }  Console.WriteLine();  }  Console.WriteLine();  return S;  }  else  {  Console.WriteLine("Нет данных");  return 0;  }  }  // Сумма двух сумм от двух матриц и вывод на экран  public void Print\_Summ(float A, float B)  {  Console.WriteLine("Сумма элементов чётных строк матрицы 1: " + A);  Console.WriteLine("Сумма элементов чётных строк матрицы 2: " + B);  S = A + B; // Сумма  Console.WriteLine("\nИтоговая сумма: " + S);  }  }  } |

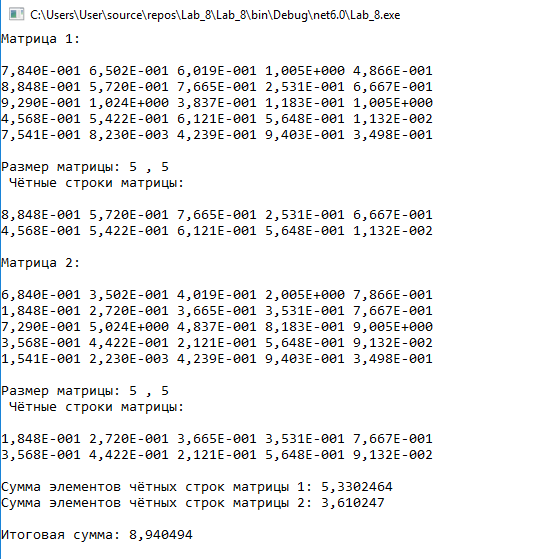


Рисунок 2 - Результат выполнения программы

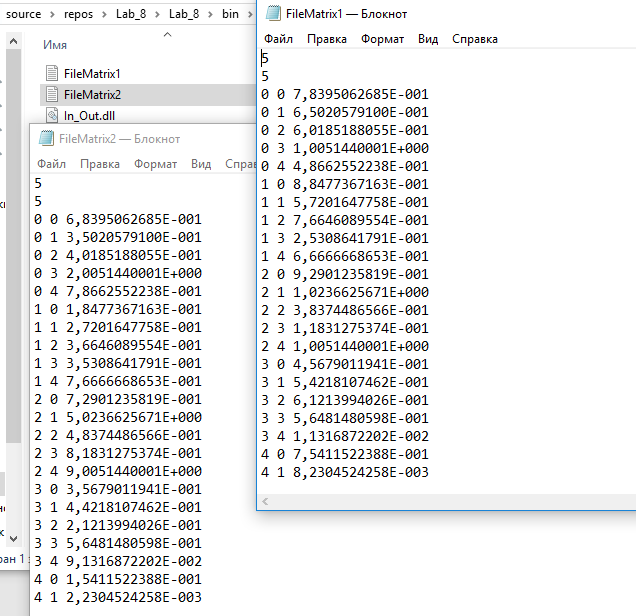


Рисунок 3 — Входные данные 1

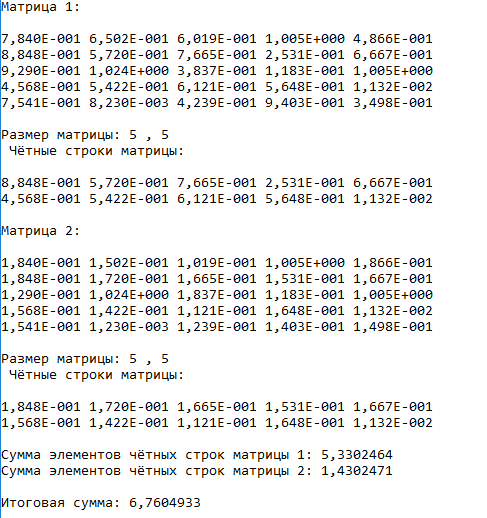


Рисунок 3 – Второй результат выполнения программы

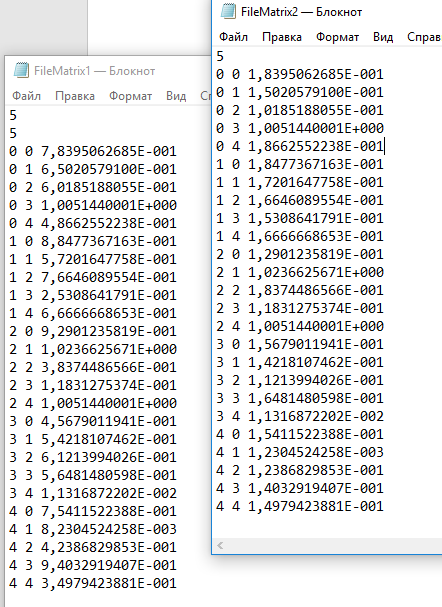


Рисунок — Входные данные 2

1. Каталоги

|  |
| --- |
| *Листинг 4 — Отдельный расматриваемый код работы с директориями* |
| DirectoryInfo dir1 = new DirectoryInfo(@"C:\Users\User\source\repos\Lab\_8\Lab\_8\bin\Debug\net6.0");  if (dir1.Exists)  {  Console.WriteLine("\nDirectoryInfo dir1:");  Console.WriteLine("FullName: {0}", dir1.FullName);  Console.WriteLine("Name: {0}", dir1.Name);  Console.WriteLine("Parent: {0}", dir1.Parent);  Console.WriteLine("CreationTime: {0}", dir1.CreationTime);  Console.WriteLine("Attributes: {0}", dir1.Attributes);  Console.WriteLine("Root: {0}", dir1.Root);  Console.WriteLine();  }  // Создаём подкаталог  dir1.CreateSubdirectory("In");  // Создаём вложенный каталог  dir1.CreateSubdirectory(@"Out\3"); |

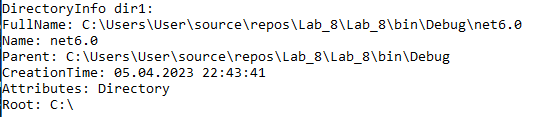


Рисунок — Результат выполнения программы

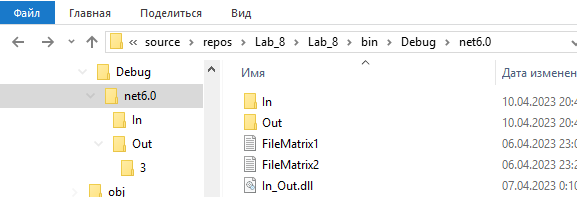


Рисунок — Результат создания каталогов

1. Ответы на вопросы

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

Stream, StreamWriter, FileStream, DriveInfo, Directory, DirectoryInfo, File, FileInfo и т.д.

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

Поток (stream) – это абстракция, которая генерирует и принимает данные. С помощью потока можно читать данные из различных источников (клавиатура, файл, память) и записывать в различные источники (принтер, экран, файл, память).

Центральную часть потоков системы C# занимает абстрактный класс Stream, производный от него класс FileStream – байтовый поток, разработанный для файлового ввода-вывода; BufferedStream - заключает в оболочку байтовый поток и добавляет буферизацию, которая во многих случаях увеличивает производительность программы; MemoryStream - байтовый поток, который использует память для хранения данных.

***3. Что такое байтовый поток?***

Поток представляет порцию данных, протекающих от источника к цели. Потоки представляют собой общий способ взаимодействия с последовательностью байтов, независимо от того, какого рода устройство (файл, сеть, соединение, принтер и т.п.) хранит или отображает эти байты.

***4. Что такое символьный поток?***

Чтобы создать символьный поток нужно поместить объект класса Stream внутрь объекта класса StreamWriter или объекта класса StreamReader. В этом случае байтовый поток будет автоматически преобразовываться в символьный. Грубо говоря, символьный поток- это, фактически, поток байтов, который был обернут логикой, которая позволяет ему выводить символы из определенной кодировки.

***5. С какими потоками вы работали в данной работы?***

Символьный поток.

***6. Какие статические классы для работы с файлами и каталогами вы знаете.***

File и Directory.

***7. Какие способы для очистки памяти от объектов вы знаете. Примеры.***

Все потоки имеют интерфейс IDisposable, поэтому имеют метод Dispose для освобождения неуправляемых ресурсов. При вызове метода Dispose, автоматически вызывается метод Flush() класса Stream для того, чтобы завершить работу с остатками данных, хранящимися во внутреннем буфере потока.

Существуют два способа написания кода, которые предусматривают автоматическое удаление объектов из памяти.

1. Использование конструкции using – в ней мы объявляем переменную, которую следует удалить после исполнения кода в блоке, который принадлежит using.
2. Использование конструкции try/catch/finaly – в finaly мы должны сами вызвать метод s.Dispose(), который описан в интерфейсе IDisposable.

Примеры:



Рисунок - Использование метода Dispose

Конструкция using оформляет блок кода и создает объект некоторого типа, который реализует интерфейс IDisposable, в частности, его метод Dispose. При завершении блока кода у объекта tom, вызывается метод Dispose. Важно, что данная конструкция применяется только для типов, которые реализуют интерфейс IDisposable.

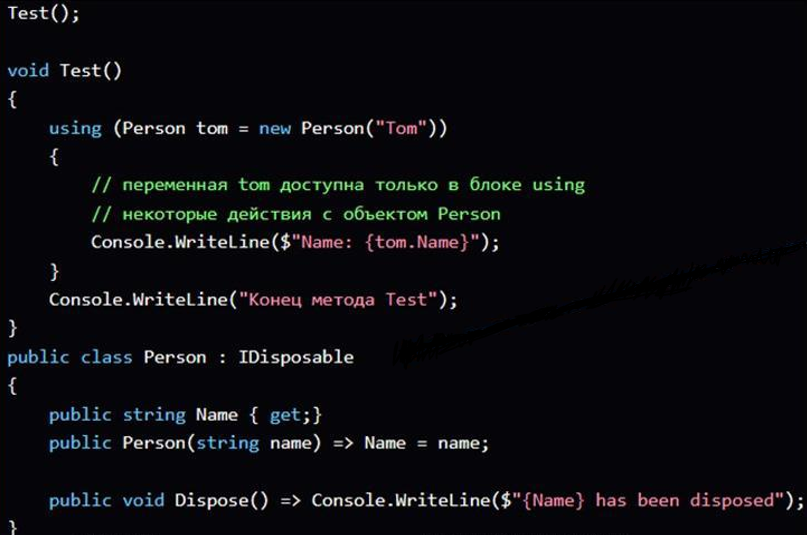


Рисунок - Использование конструктора using

***8. Использовали ли вы в вашей программе очистку памяти?***

Нет.

***9. Приведите пример использования не статического класса для работы с файлом.***

|  |
| --- |
| *Листинг 5 — Использование класса FileInfo* |
| using System;  using System.IO;  namespace FileInfoClass1  {  class Program  {  [Foo<string>("Hello")]  static void Main(string[] args)  {  FileInfo fi = new FileInfo($@"C:\Test\TestFile\_Dpk1.txt");  using StreamWriter str = fi.CreateText();  Console.WriteLine("File has been created");  Console.ReadLine();  }  }  public class FooAttribute<T> : Attribute  {  public T FooProperty { get; set; }  public FooAttribute(T fooValue) => FooProperty = fooValue;  }  } |