# **Лабораторная работа № 11**

**Тема:** Разработка и тестирование программ для работы с файлами на языке С#.

**Цель работы:**

1. формирование навыков разработки и кодирования программ на языке С# для работы с файлами;
2. изучение основных приемов и функций для работы с файлами на языке C#/

**Оборудование:**

1. ПК
2. Программное обеспечение: ОС Windows, среда Visual Studio.Net

**Контрольные вопросы:**

1. Поток в языке C# представляет собой последовательность данных, которые могут быть прочитаны или записаны. Потоки используются для чтения и записи данных в различные источники, такие как файлы, сеть, память и т.д.

2. Класс Stream в C# является абстрактным базовым классом для всех потоковых операций ввода-вывода. Он предоставляет методы и свойства для чтения, записи и управления данными в потоке.

3. Из класса Stream производятся следующие потоки:

- StreamReader и StreamWriter - потоки для работы со строками (текстовые данные) в потоке.

- BinaryReader и BinaryWriter - потоки для работы с двоичными данными в потоке.

- FileStream - поток для чтения и записи данных в файл.

- NetworkStream - поток для чтения и записи данных через сетевое соединение.

- MemoryStream - поток для чтения и записи данных в память.

4. В C# имеются следующие режимы открытия файлов:

- FileMode.Create - создает новый файл или перезаписывает существующий файл.

- FileMode.Open - открывает существующий файл для чтения.

- FileMode.OpenOrCreate - открывает существующий файл для чтения или создает новый файл, если он не существует.

- FileMode.Append - открывает существующий файл для записи в конец файла или создает новый файл, если он не существует, и записывает данные в конец файла.

- FileMode.Truncate - открывает существующий файл и обрезает его размер до нуля.

5. Система при открытии файла выполняет следующие действия:

- Проверяет наличие файла в указанном пути.

- Получает доступ к файлу.

- Устанавливает указатель на начало файла.

6. В языке C# для чтения из байтового потока используются методы класса Stream:

- Read - читает последовательность байтов из потока.

- ReadByte - читает один байт из потока и возвращает его как целое число.

7. Для побайтовой записи данных в поток в C# используется метод Write класса Stream:

- Write - записывает последовательность байтов в поток.

8. Для закрытия файла в C# используется метод Close класса Stream. Этот метод освобождает ресурсы, связанные с потоком, и закрывает его.

9. Для создания символьного потока в C# можно использовать классы StreamReader и StreamWriter, которые оборачивают поток данных и предоставляют методы для работы со строками и символами.

10. Двоичные потоки используются в C# для чтения и записи двоичных данных, таких как числа, байты, структуры и т.д. Эти потоки, такие как BinaryReader и BinaryWriter, предоставляют методы для удобной работы с двоичными данными.

11. BinaryWriter и BinaryReader - это классы в C#, которые обеспечивают функциональность для записи и чтения двоичных данных в поток. BinaryWriter позволяет записывать примитивные типы данных, строки и другие объекты в двоичный формат, а BinaryReader - считывать эти данные обратно.

12. В языке C# используются следующие стандартные потоки для ввода и вывода:

- Console.In - стандартный поток ввода, который связан с командной строкой или входным устройством.

- Console.Out - стандартный поток вывода, который связан с командной строкой или выходным устройством.

- Console.Error - стандартный поток ошибок, который используется для вывода сообщений об ошибках и предупреждений.

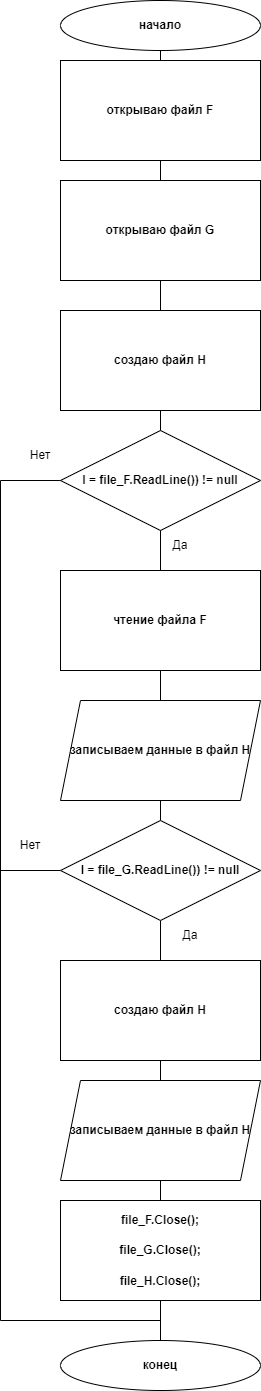
**Ход работы:**

1. Разработать алгоритм задачи и представить его в виде схемы программы
2. В Блокноте или в другом текстовом редакторе создать файл с программой
3. Используя средства Денвер выполнить тестирование и отладку программу
4. Результаты представить в виде отчета
5. Сделать вывод о проделанной работе

**Содержание отчета:**

**Задание на лабораторную работу:** Даны символьные файлы f и g. Записать в файл h сначала компоненты файла f, затем — компоненты файла g с сохранением порядка.

**Схема программы**



Листинг с исходным кодом

using System.IO;

using System;

class Program

{

static void Main()

{

StreamReader file\_F = new StreamReader("G:\\f.txt");

StreamReader file\_G = new StreamReader("G:\\g.txt");

// Создаем файл h для записи

StreamWriter file\_H = new StreamWriter("G:\\h.txt");

// Записываем компоненты файла f в файл h

string l;

while ((l = file\_F.ReadLine()) != null)

{

file\_H.WriteLine(l);

}

// Записываем компоненты файла g в файл h

while ((l = file\_G.ReadLine()) != null)

{

file\_H.WriteLine(l);

}

// Закрываем все файлы

file\_F.Close();

file\_G.Close();

file\_H.Close();

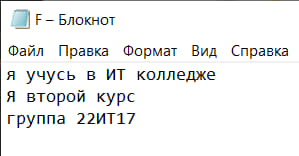
Console.WriteLine("Файлы успешно объединены.");

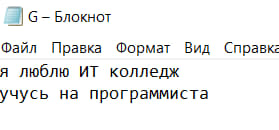
Console.ReadKey();

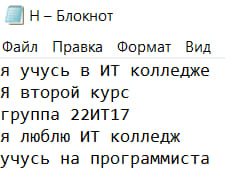
}

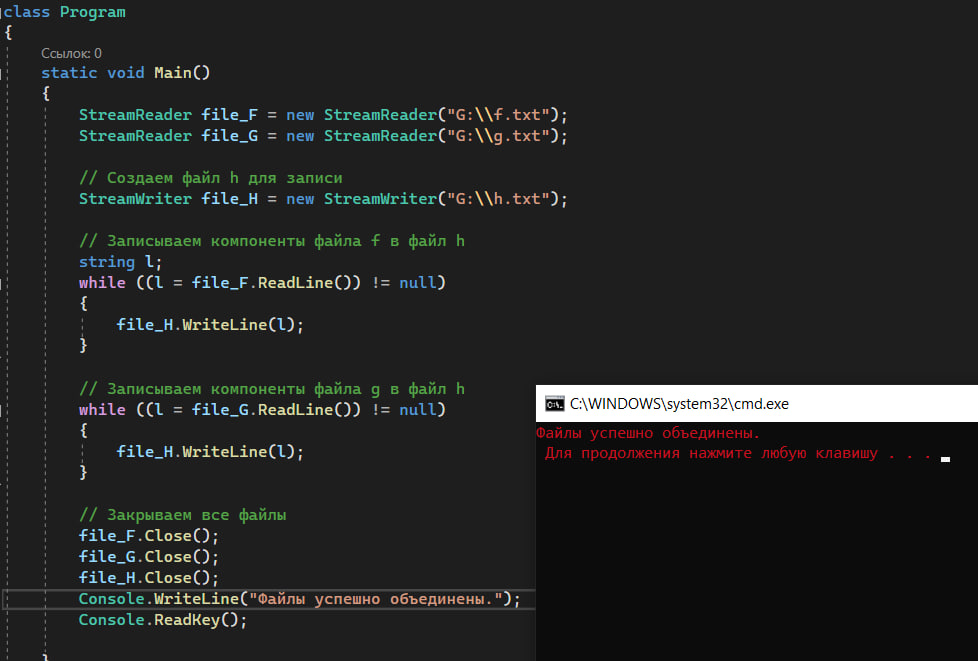
}

**Результаты тестирования**









**Вывод по работе.** Работа с потоками важна, потому что позволяет программам эффективно обрабатывать большие объемы данных без необходимости загрузки всей информации в память одновременно.