# Лабораторная работа № 4.1 Работа с текстовыми файлами

**Цель лабораторной работы**

Изучить способы считывания-записи данных в текстовые файлы, закрепить навыки работы с файловой системой.

**Постановка задачи**

Работу с файловой системой начнем с самого верхнего уровня - дисков. Для представления диска в пространстве имен System.IO имеется класс **DriveInfo**.

Этот класс имеет статический метод GetDrives, который возвращает имена всех логических дисков компьютера. Также он предоставляет ряд полезных свойств:

* **AvailableFreeSpace**: указывает на объем доступного свободного места на диске в байтах
* **DriveFormat**: получает имя файловой системы
* **DriveType**: представляет тип диска
* **IsReady**: готов ли диск (например, DVD-диск может быть не вставлен в дисковод)
* **Name**: получает имя диска
* **TotalFreeSpace**: получает общий объем свободного места на диске в байтах
* **TotalSize**: общий размер диска в байтах
* **VolumeLabel**: получает или устанавливает метку тома

Для работы с каталогами в пространстве имен System.IO предназначены сразу два класса: **Directory** и **DirectoryInfo**.

**Класс Directory**

Класс Directory предоставляет ряд статических методов для управления каталогами. Некоторые из этих методов:

* **CreateDirectory(path)**: создает каталог по указанному пути path
* **Delete(path)**: удаляет каталог по указанному пути path
* **Exists(path)**: определяет, существует ли каталог по указанному пути path. Если существует, возвращается true, если не существует, то false
* **GetDirectories(path)**: получает список каталогов в каталоге path
* **GetFiles(path)**: получает список файлов в каталоге path
* **Move(sourceDirName, destDirName)**: перемещает каталог
* **GetParent(path)**: получение родительского каталога

**Класс DirectoryInfo**

Данный класс предоставляет функциональность для создания, удаления, перемещения и других операций с каталогами. Во многом он похож на Directory. Некоторые из его свойств и методов:

* **Create()**: создает каталог
* **CreateSubdirectory(path)**: создает подкаталог по указанному пути path
* **Delete()**: удаляет каталог
* Свойство **Exists**: определяет, существует ли каталог
* **GetDirectories()**: получает список каталогов
* **GetFiles()**: получает список файлов
* **MoveTo(destDirName)**: перемещает каталог
* Свойство **Parent**: получение родительского каталога
* Свойство **Root**: получение корневого каталога
* Свойство **Exists**: указывает, существует ли файл
* Свойство **Length**: получает размер файла
* Свойство **Extension**: получает расширение файла
* Свойство **Name**: получает имя файла
* Свойство **FullName**: получает полное имя файла

Класс File реализует похожую функциональность с помощью статических методов:

* **Copy()**: копирует файл в новое место
* **Create()**: создает файл
* **Delete()**: удаляет файл
* **Move**: перемещает файл в новое место
* **Exists(file)**: определяет, существует ли файл

**Получение информации о файле**

string path = @"C:\apache\hta.txt";

FileInfo fileInf = new FileInfo(path);

if (fileInf.Exists)

{

Console.WriteLine("Имя файла: {0}", fileInf.Name);

Console.WriteLine("Время создания: {0}", fileInf.CreationTime);

Console.WriteLine("Размер: {0}", fileInf.Length);

}

**Удаление файла**

string path = @"C:\apache\hta.txt";

FileInfo fileInf = new FileInfo(path);

if (fileInf.Exists)

{

fileInf.Delete();

// альтернатива с помощью класса File

// File.Delete(path);

}

**Перемещение файла**

string path = @"C:\apache\hta.txt";

string newPath = @"C:\SomeDir\hta.txt";

FileInfo fileInf = new FileInfo(path);

if (fileInf.Exists)

{

fileInf.MoveTo(newPath);

// альтернатива с помощью класса File

// File.Move(path, newPath);

}

**Копирование файла**

string path = @"C:\apache\hta.txt";

string newPath = @"C:\SomeDir\hta.txt";

FileInfo fileInf = new FileInfo(path);

if (fileInf.Exists)

{

fileInf.CopyTo(newPath, true);

// альтернатива с помощью класса File

// File.Copy(path, newPath, true);

}

Метод CopyTo класса FileInfo принимает два параметра: путь, по которому файл будет копироваться, и булевое значение, которое указывает, надо ли при копировании перезаписывать файл (если true, как в случае выше, файл при копировании перезаписывается). Если же в качестве последнего параметра передать значение false, то если такой файл уже существует, приложение выдаст ошибку.

Метод Copy класса File принимает три параметра: путь к исходному файлу, путь, по которому файл будет копироваться, и булевое значение, указывающее, будет ли файл перезаписываться.

Класс **FileStream** представляет возможности по считыванию из файла и записи в файл. Он позволяет работать как с текстовыми файлами, так и с бинарными.

Рассмотрим наиболее важные его свойства и методы:

* Свойство **Length**: возвращает длину потока в байтах
* Свойство **Position**: возвращает текущую позицию в потоке
* Метод **Read**: считывает данные из файла в массив байтов. Принимает три параметра: int Read(byte[] array, int offset, int count) и возвращает количество успешно считанных байтов. Здесь используются следующие параметры:
  + array - массив байтов, куда будут помещены считываемые из файла данные
  + offset представляет смещение в байтах в массиве array, в который считанные байты будут помещены
  + count - максимальное число байтов, предназначенных для чтения. Если в файле находится меньшее количество байтов, то все они будут считаны.
* Метод **long Seek(long offset, SeekOrigin origin)**: устанавливает позицию в потоке со смещением на количество байт, указанных в параметре offset.
* Метод **Write**: записывает в файл данные из массива байтов. Принимает три параметра: Write(byte[] array, int offset, int count)
  + array - массив байтов, откуда данные будут записываться в файла
  + offset - смещение в байтах в массиве array, откуда начинается запись байтов в поток
  + count - максимальное число байтов, предназначенных для записи

FileStream представляет доступ к файлам на уровне байтов, поэтому, например, если вам надо считать или записать одну или несколько строк в текстовый файл, то массив байтов надо преобразовать в строки, используя специальные методы. Поэтому для работы с текстовыми файлами применяются другие классы.

В то же время при работе с различными бинарными файлами, имеющими определенную структуру FileStream может быть очень даже полезен для извлечения определенных порций информации и ее обработки.

**Задание на лабораторную работу**

Сформировать файл с входными данными. Создать приложение, реализующее чтение/запись файла. Предусмотреть меню выбора файла из списка.

**Варианты заданий на лабораторную работу**

1. **Анализ данных о погоде**: Считайте данные о температуре и влажности из CSV файла. Рассчитайте среднюю температуру и влажность за неделю и запишите результаты в новый файл.
2. **Обработка данных о продажах**: Прочитайте файл с данными о продажах (товар, количество, цена). Рассчитайте общую выручку для каждого товара и запишите результаты в новый CSV файл.
3. **Статистический анализ оценок**: Считайте файл с оценками студентов. Рассчитайте среднее, медиану и стандартное отклонение для каждого студента и запишите результаты в новый файл.
4. **Анализ данных о трафике**: Прочитайте лог-файл с данными о посещениях веб-сайта (время, IP-адрес). Определите количество уникальных посетителей и запишите это значение в новый файл.
5. **Обработка данных о финансах**: Считайте данные о транзакциях (дата, сумма, категория). Рассчитайте общие расходы по каждой категории и запишите результаты в новый файл.
6. **Анализ временных рядов**: Прочитайте файл с временными рядами (время, значение). Выполните скользящее среднее и запишите сглаженные данные в новый файл.

**** .

1. **Обработка данных о фитнесе**: Считайте данные из файла с информацией о тренировках (дата, продолжительность, калории). Рассчитайте общие калории за месяц и запишите результаты в новый файл.
2. **Анализ данных о клиентах**: Прочитайте файл с данными о клиентах (имя, возраст, покупки). Определите средний возраст клиентов и запишите результат в новый файл.
3. **Обработка данных о транспорте**: Считайте данные о поездках (начало, конец, расстояние). Рассчитайте общее расстояние, пройденное за день, и запишите в новый файл.
4. **Анализ данных о продуктах**: Прочитайте файл с данными о продуктах (название, цена, количество на складе). Рассчитайте общую стоимость запасов и запишите результат в новый файл.
5. **Анализ данных о сотрудниках**: Считайте данные о сотрудниках (имя, отдел, зарплата). Рассчитайте среднюю зарплату по каждому отделу и запишите результаты в новый файл.
6. **Обработка данных о заказах**: Прочитайте файл с данными о заказах (клиент, товар, количество). Определите наиболее популярный товар и запишите его в новый файл.
7. **Анализ данных о посещаемости**: Считайте данные о посещаемости студентов (имя, дата, статус). Определите процент посещаемости для каждого студента и запишите в новый файл.
8. **Обработка данных о расходах**: Прочитайте файл с данными о расходах (дата, сумма, категория). Найдите день с наибольшими расходами и запишите его в новый файл.
9. **Анализ данных о производительности**: Считайте данные о производительности сотрудников (имя, задачи, время выполнения). Рассчитайте среднее время выполнения задач и запишите результаты в новый файл.
10. **Обработка данных о транспортных средствах**: Прочитайте данные о транспортных средствах (тип, год выпуска, пробег). Определите средний пробег для каждого типа и запишите в новый файл.
11. **Анализ данных о потреблении энергии**: Считайте данные о потреблении энергии (дата, количество). Рассчитайте среднее потребление за месяц и запишите результаты в новый файл.
12. **Обработка данных о клиентах**: Прочитайте файл с данными о клиентах (имя, покупки, сумма). Определите клиента с наибольшими покупками и запишите его в новый файл.
13. **Анализ данных о производстве**: Считайте данные о производстве (дата, количество, дефекты). Рассчитайте процент дефектов и запишите результаты в новый файл.
14. **Обработка данных о здоровье**: Прочитайте файл с данными о здоровье (имя, вес, рост). Рассчитайте индекс массы тела (ИМТ) для каждого человека и запишите в новый файл.
15. **Анализ данных о продажах недвижимости**: Считайте данные о продажах (адрес, цена, площадь). Рассчитайте среднюю цену за квадратный метр и запишите результаты в новый файл.
16. **Обработка данных о бронированиях**: Прочитайте файл с данными о бронированиях (гость, дата, номер). Определите наиболее популярные даты и запишите их в новый файл.
17. **Анализ данных о доходах**: Считайте данные о доходах (источник, сумма, дата). Рассчитайте общий доход за год и запишите результаты в новый файл.
18. **Обработка данных о поставках**: Прочитайте файл с данными о поставках (товар, количество, дата). Определите товар с наибольшими поставками и запишите его в новый файл.
19. **Анализ данных о занятиях спортом**: Считайте данные о занятиях (вид спорта, время, калории). Рассчитайте общее количество сожжённых калорий и запишите результаты в новый файл.
20. **Обработка данных о путешествиях**: Прочитайте файл с данными о путешествиях (место, дата, расходы). Определите самое дорогое путешествие и запишите его в новый файл.
21. **Анализ данных о потребителях**: Считайте данные о потребителях (возраст, покупки, предпочтения). Определите наиболее популярную категорию товаров и запишите её в новый файл.
22. **Обработка данных о транспорте**: Прочитайте файл с данными о рейсах (номер рейса, время, задержка). Рассчитайте среднюю задержку и запишите результаты в новый файл.
23. **Анализ данных о производительности сайта**: Считайте данные о времени загрузки страниц (URL, время). Определите страницы с наибольшим временем загрузки и запишите их в новый файл.
24. **Обработка данных о звонках**: Прочитайте файл с данными о звонках (номер, длительность, дата). Определите самый долгий звонок и запишите его в новый файл.