**1. Какие классы содержаться с System.IO?**

StreamWriter

StreamReader

DriveInfo

FileInfo

File

Directory

DirectoryInfo

Path

FileStream **2. Для чего используются классы Directory и DirectoryInfo? В чем  
отличие?**

Класс **Directory**. Статический класс Directory предоставляет ряд методов для управления каталогами. Если надо совершить одну-две операции с одним каталогом, выполняют дополнительные проверки безопасности

Класс **DirectoryInfo**. Данный класс предоставляет функциональность для создания, удаления, перемещения и других операций с каталогами. Во многом он похож на Directory, но не является статическим. Если необходимо выполнить последовательность операций с одним и тем же каталогом.

**3. Для чего используются классы File и FileInfo? Какие методы они  
содержат.**

**File** - это статический класс, предоставляющий методы для выполнения операций с файлами без необходимости создания экземпляра класса.

* Exists(path): Проверяет, существует ли файл по указанному пути.
* ReadAllText(path): Считывает содержимое текстового файла и возвращает его в виде строки.
* WriteAllLines(path, lines): Записывает строки в текстовый файл.
* Copy(sourcePath, destinationPath): Копирует файл из одного места в другое.
* Delete(path): Удаляет файл.

**FileInfo** - это класс, представляющий файл и предоставляющий экземплярные методы для работы с этим файлом.

* Exists: Возвращает true, если файл существует.
* OpenRead(): Открывает файл для чтения в виде потока FileStream.
* OpenWrite(): Открывает файл для записи в виде потока FileStream.
* CopyTo(destinationPath): Копирует файл в указанное место.
* Delete(): Удаляет файл.

**4. Для чего используются классы StreamReader и StreamWriter?**

**StreamWriter** в C# предназначен для удобной записи текстовых данных в поток в текстовой форме.

Класс **StreamReader** позволяет нам легко считывать весь текст или отдельные строки из текстового файла.

**5. Для чего используются классы BinaryWriter и BinaryReader?**

Для работы с бинарными файлами предназначена пара классов BinaryWriter и BinaryReader. Эти классы позволяют читать и записывать данные в двоичном формате.

**6. Как можно сжимать и восстанавливать файлы?**

Для сжатия и восстановления файлов в C# вы можете использовать классы из пространства имен System.IO.Compression. Например класс ZipFile и его методы CreateFromDirectory и ExtractToDirector.

**7. Расскажите алгоритм сжатия GZip.**

Использует алгоритм LZ77 для поиска и замены повторяющихся последовательностей. Применяет Huffman coding для кодирования символов с целью уменьшения размера данных.Добавляет заголовок с метаданными о сжатии и хвост с CRC-32 контрольной суммой. Сжатые данные записываются в выходной поток.

**8. Для чего служит класс Path?**

Класс Path в C# из пространства имен System.IO предоставляет методы для работы с путями к файлам и каталогам. Он предоставляет удобные средства для создания, объединения и анализа путей файлов.

**9. Что такое тестовый и бинарный файл. Как можно выполнить чтение  
и запись в эти файлы.**

**Текстовый файл** содержит данные в текстовой форме, обычно в виде читаемого человеком текста. Текстовые файлы могут содержать символы, цифры и другие символы, предназначенные для человеческого восприятия.

Текстовые файлы можно читать и записывать с использованием класса StreamReader для чтения текста из файла и StreamWriter для записи текста в файл.

**Бинарный файл** содержит данные в бинарной (двоичной) форме, что означает, что данные представлены в виде последовательности байтов. Он может содержать произвольные данные, включая изображения, аудио, видео и другие двоичные форматы. Бинарные файлы можно читать и записывать с использованием классов BinaryReader и BinaryWriter.

**10.Что такое произвольный доступ к файлу? Приведите пример.**

Произвольный доступ к файлу — это возможность читать и записывать данные в файл не последовательно, а произвольно, используя адресацию. Это означает, что можно осуществлять операции чтения и записи в произвольных местах файла, а не только с начала и до конца. Произвольный доступ обеспечивается с использованием класса FileStream

**11.Как применяется конструкция** using **(не директива) при работе с  
файловыми потоками? Для чего ее используют.**

В контексте работы с файловыми потоками, использование using с классами, реализующими интерфейс IDisposable, обеспечивает автоматическое закрытие файлового потока после выхода из блока using.IDisposable - это интерфейс в языке программирования C#, предназначенный для реализации управления ресурсами и освобождения их после использования.