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

* FileStream: Предоставляет возможности для чтения и записи байтов в файле.
* StreamReader и StreamWriter: Позволяют считывать и записывать текстовые данные в файле.
* DirectoryInfo и FileInfo: Предоставляют информацию о директориях и файлах соответственно.
* Path: Содержит методы для работы с путями к файлам и директориям.
* BinaryReader и BinaryWriter: Позволяют считывать и записывать двоичные данные в файле.
* MemoryStream: Предоставляет поток в памяти для работы с данными.

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

отличие?

Оба класса необходимы для работы с директориями. Класс DirectoryInfo, в отличие от Directory, является классом экземпляра. Он предоставляет более широкий набор методов и свойств для работы с директориями.

Directory позволяет выполнять операции с директориями без необходимости создания экземпляра класса, в то время как DirectoryInfo требует создания экземпляра.

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

содержат.

Оба необходимы для работы с файлами. Класс FileInfo, в отличие от File, является классом экземпляра. Он предоставляет более широкий набор методов и свойств для работы с файлами.

* File.Exists(path): Проверяет, существует ли файл по указанному пути.
* File.Copy(sourcePath, destinationPath): Копирует файл с исходного пути на указанный путь.
* File.Delete(path): Удаляет файл по указанному пути.
* File.Move(sourcePath, destinationPath): Перемещает файл с исходного пути на указанный путь.
* File.ReadAllLines(path): Считывает все строки из файла и возвращает их в виде массива строк.
* File.WriteAllText(path, content): Записывает указанное содержимое в файл.
* fileInfo.Exists: Возвращает значение, указывающее, существует ли файл.
* fileInfo.CopyTo(destinationPath): Копирует файл в указанный путь.
* fileInfo.Delete(): Удаляет файл.
* fileInfo.MoveTo(destinationPath): Перемещает файл в указанный путь.
* fileInfo.OpenRead(): Открывает файл для чтения в виде потока.
* fileInfo.OpenWrite(): Открывает файл для записи в виде потока.
* fileInfo.Length: Возвращает размер файла в байтах.
* fileInfo.LastWriteTime: Возвращает дату и время последней записи в файл.

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

StreamReader считывает символы из потока байтов в определенной кодировке.

StreamWriter для записи символов в поток в определенной кодировке.

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

BinaryReader Считывает примитивные типы данных как двоичные значения в заданной кодировке.

BinaryWriter Записывает примитивные типы в двоичный поток и поддерживает запись строк в заданной кодировке.

6. Как можно сжимать и восстанавливать файлы?

Это классы ZipFile, DeflateStream и GZipStream, которые находятся в пространстве имен System.IO.Compression

7. Расскажите алгоритм сжатия *GZip*.

GZip - это алгоритм сжатия данных, который использует комбинацию алгоритмов Deflate и CRC-32. Он разбивает исходный файл на блоки, сжимает их с помощью алгоритма Deflate, вычисляет контрольную сумму CRC-32 для проверки целостности данных и сохраняет их в выходной GZip-файл. При распаковке GZip-файла алгоритм восстанавливает исходные данные и проверяет их целостность.

8. Для чего служит класс *Path*?

Он предоставляет функциональность для обработки и манипулирования строками путей, включая операции соединения путей, получения имени файла, расширения файла и другие.

9. Что такое тестовый и бинарный файл. Как можно выполнить чтение

и запись в эти файлы.

* Тестовый файл содержит текстовую информацию, обычно в формате ASCII или Unicode.
* Чтение и запись в текстовый файл можно выполнить с использованием класса StreamReader для чтения текста из файла и класса StreamWriter для записи текста в файл.
* Бинарный файл содержит двоичные данные, которые не являются текстом и не могут быть прочитаны напрямую в текстовом редакторе.
* Чтение и запись в бинарный файл можно выполнить с использованием класса FileStream в сочетании с другими классами для чтения и записи байтовых данных.

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

Произвольный доступ к файлу (Random Access) означает возможность чтения или записи данных в произвольное место файла без необходимости последовательного чтения или записи от начала файла до нужного места.

11.Как применяется конструкция ***using*** (не директива) при работе с

файловыми потоками? Для чего ее используют.

Конструкция using в C# применяется для работы с файловыми потоками и другими ресурсами, которые требуют явного освобождения после использования. Она помогает упростить и безопасно управлять ресурсами в коде и способствует повышению эффективности и надежности приложений.