**1. Что такое бинарный файл?**  
Бинарный файл – это файл, в котором данные хранятся в виде последовательности байтов, без преобразования в человекочитаемый текст. При работе с бинарными файлами запись и чтение происходит «как есть», что позволяет сохранить точное представление данных (например, чисел, структур) без форматирования.

**2. Как изменить текущую позицию в файле?**  
Для изменения позиции в файле используются методы перемещения указателя потока:

* **В потоках ввода:** seekg(offset, dir)
* **В потоках вывода:** seekp(offset, dir)  
  Здесь offset – смещение, а dir – начальная позиция (например, ios::beg, ios::cur или ios::end).

**3. Как узнать текущую позицию в файле?**  
Для определения текущей позиции в потоке применяют:

* **Для чтения:** tellg()
* **Для записи:** tellp()  
  Оба метода возвращают позицию в виде значения типа streampos.

**4. Для чего используется метод clear()?**  
Метод clear() применяется для сброса флагов состояния потока (например, ошибок или флага достижения конца файла). Это позволяет «очистить» поток и продолжить работу, если до этого произошла ошибка или был установлен флаг eof.

**5. Объясните понятие прямой доступ к файлу.**  
Прямой (или произвольный, random) доступ означает возможность перемещаться к любой позиции файла без необходимости последовательного (пошагового) чтения или записи от начала файла. Это позволяет эффективно читать или записывать данные в определённом месте файла.

**6. Перечислите команды, обеспечивающие прямой доступ к файлу.**  
В C++ для реализации произвольного доступа применяются методы:

* seekg() – для установки позиции чтения,
* seekp() – для установки позиции записи,  
  а также их соответствующие методы определения текущей позиции (tellg(), tellp()).

**7. Перечислите команды, обеспечивающие считывание и запись двоичных блоков.**  
Для работы с двоичными блоками применяются методы:

* read(char\* buffer, streamsize count) – для чтения блока данных,
* write(const char\* buffer, streamsize count) – для записи блока данных.  
  При открытии файла в бинарном режиме (с флагом ios::binary) эти методы работают без преобразований данных.

**8. Какая команда обеспечивает проверку достижения конца файла.**  
Для проверки конца файла используется метод eof(). Он возвращает логическое значение true, если достигнут конец файла, и false в противном случае.

**9. Какие режимы открытия файлов существуют?**  
При открытии файлов с помощью классов fstream, ifstream и ofstream доступны следующие режимы (флаги):

* ios::in – открытие для чтения,
* ios::out – открытие для записи,
* ios::app – открытие с добавлением данных в конец файла,
* ios::ate – открытие с установкой указателя в конец файла сразу после открытия,
* ios::trunc – открытие с удалением содержимого файла (если он существует),
* ios::binary – открытие в бинарном режиме (без преобразования данных).

**10. Какие значения возвращает метод открытия файла в зависимости от результата?**  
Метод open() сам по себе не возвращает значение, но после попытки открытия можно проверить состояние потока с помощью методов:

* is\_open() – возвращает true, если файл успешно открыт,
* fail() или bad() – возвращают true, если произошла ошибка при открытии файла.  
  Также можно использовать приведение потока к логическому значению (например, if(stream)).

**11. Какие значения возвращает метод проверки достижения конца файла?**  
Метод eof() возвращает булево значение:

* true – если достигнут конец файла,
* false – если конец файла ещё не достигнут.

**12. Функции ввода-вывода данных файла: символьный ввод-вывод, строковый ввод-вывод, блоковый ввод-вывод.**

* **Символьный ввод-вывод:** Работа с отдельными символами (например, метод get() или оператор >> для ввода символа).
* **Строковый ввод-вывод:** Работа со строками; для чтения используется getline(), для записи – операторы или метод <<.
* **Блоковый ввод-вывод:** Позволяет считывать или записывать большие блоки данных (например, методами read() и write()), что особенно полезно при работе с бинарными файлами.

**13. Основные алгоритмы работы с файлами: запись данных в файл, чтение данных из файла.**

* **Запись в файл:**
  1. Открыть файл в нужном режиме (например, ios::out или ios::app).
  2. Проверить состояние потока после открытия.
  3. Использовать операторы или методы (<<, write()) для записи данных.
  4. Закрыть файл методом close().
* **Чтение из файла:**
  1. Открыть файл в режиме чтения (например, ios::in).
  2. Проверить, что файл открыт корректно.
  3. Считать данные с помощью операторов или методов (>>, getline(), read()).
  4. При необходимости обрабатывать ошибки или конец файла (eof()).
  5. Закрыть файл.

**14. В чем основные отличия в организации символьного, строкового, блокового и форматированного ввода-ввода в файлы.**

* **Символьный ввод-вывод:**  
  Операции производятся с отдельными символами, что позволяет детально обрабатывать данные, но может быть менее эффективным при работе с большими объемами информации.
* **Строковый ввод-вывод:**  
  Ориентирован на работу со строками (последовательностями символов) – удобен для обработки текстовых данных, но требует обработки разделителей (например, пробелов или переводов строк).
* **Блоковый ввод-вывод:**  
  Позволяет считывать и записывать данные блоками (массивы байтов), что эффективно для бинарных файлов и больших объемов данных, поскольку уменьшается число операций ввода-вывода.
* **Форматированный ввод-вывод:**  
  Осуществляется с преобразованием данных (например, числа в строковое представление и наоборот) с использованием манипуляторов и спецификаторов формата. Это удобно для представления данных в заданном виде, но может быть менее производительным, чем небформатированные операции.