В C++ работа с файлами может осуществляться двумя основными способами: **последовательный доступ** и **прямой доступ**.

1. **Последовательный доступ**

Последовательный доступ предполагает чтение данных в порядке их записи, начиная с начала файла и продолжая до конца.

Для последовательного доступа используются потоки: `ifstream` (чтение из файла) и `ofstream` (запись в файл).

Пример чтения и записи с последовательным доступом:

```
#include <iostream>
#include <fstream>
#include <string>

int main() {
    // Запись в файл
    std::ofstream outFile("example.txt");
    if (outFile.is_open()) {
        outFile << "Hello, World!" << std::endl;
        outFile << "C++ File Handling" << std::endl;
        outFile.close();
    } else {
        std::cout << "Unable to open file for writing" << std::endl;
    }

    // Чтение из файла
    std::ifstream inFile("example.txt");
    std::string line;
    if (inFile.is_open()) {
        while (getline(inFile, line)) {
            std::cout << line << std::endl;
        }
        inFile.close();
    } else {
        std::cout << "Unable to open file for reading" << std::endl;
    }

    return 0;
}
```

Этот пример демонстрирует, как поочередно записывать строки в файл и затем читать их.

2. **Прямой доступ**

Прямой доступ позволяет обращаться к конкретным байтам или блокам данных в произвольной части файла. Для этого часто используют функции работы с файлами на уровне байтов: `seekg` и `seekp` для

перемещения указателей чтения и записи.

Пример прямого доступа к файлу:

```
#include <iostream>
#include <fstream>

int main() {
    // Открываем файл в режиме побайтовой записи
    std::ofstream outFile("example.dat", std::ios::binary);
    if (outFile.is_open()) {
        int data[5] = {10, 20, 30, 40, 50};
        outFile.write(reinterpret_cast<char*>(data), sizeof(data));
        outFile.close();
    } else {
        std::cout << "Unable to open file for writing" << std::endl;
    }

    // Открываем файл для побайтового чтения
    std::ifstream inFile("example.dat", std::ios::binary);
    if (inFile.is_open()) {
        int value;
        // Перемещаем указатель на третий элемент
        inFile.seekg(2 * sizeof(int), std::ios::beg);
        inFile.read(reinterpret_cast<char*>(&value), sizeof(value));
        std::cout << "The third value is: " << value << std::endl;
        inFile.close();
    } else {
        std::cout << "Unable to open file for reading" << std::endl;
    }

    return 0;
}
```

В этом примере сначала в бинарный файл записывается массив целых чисел, а затем происходит чтение третьего элемента напрямую с помощью функции `seekg`, которая перемещает указатель на нужную позицию в файле.

Основные функции для работы с файлами:

- `open()`: Открывает файл.
- `close()`: Закрывает файл.
- `write()`: Записывает данные в файл (используется с бинарными файлами).
- `read()`: Читает данные из файла (используется с бинарными файлами).
- `seekg()`: Перемещает указатель чтения.
- `seekp()`: Перемещает указатель записи.
- `tellg()`: Возвращает текущую позицию указателя чтения.
- `tellp()`: Возвращает текущую позицию указателя записи.

Режимы открытия файлов:

- `std::ios::in`: Открытие файла для чтения.
- `std::ios::out`: Открытие файла для записи.
- `std::ios::app`: Открытие файла для добавления данных в конец.
- `std::ios::binary`: Открытие файла в бинарном режиме.

Таким образом, последовательный доступ удобен для работы с текстовыми файлами, где нужно читать или писать данные последовательно, тогда как прямой доступ применим, когда требуется работать с файлами произвольно, например в случае работы с бинарными файлами.