CSE 232

Programming with C++

Lecture 11

File Handling in C++

Prepared by

Md. Mijanur Rahman, Prof. Dr.
Dept. of Computer Science and Engineering
Jatiya Kabi Kazi Nazrul Islam University, Bangladesh
Email: mijan@jkkniu.edu.bd
Web: www.mijanrahman.com

Contents

File Handling in C++

File Handling through C++ Classes

- File handling is used to store data permanently in a computer. Using file handling we can store our data in secondary memory (Hard disk).
- **How to achieve the File Handling?**
- For achieving file handling we need to follow the following steps:-
- STEP 1-Naming a file
- STEP 2-Opening a file
- STEP 3-Writing data into the file
- STEP 4-Reading data from the file
- STEP 5-Closing a file.

File Handling through C++ Classes

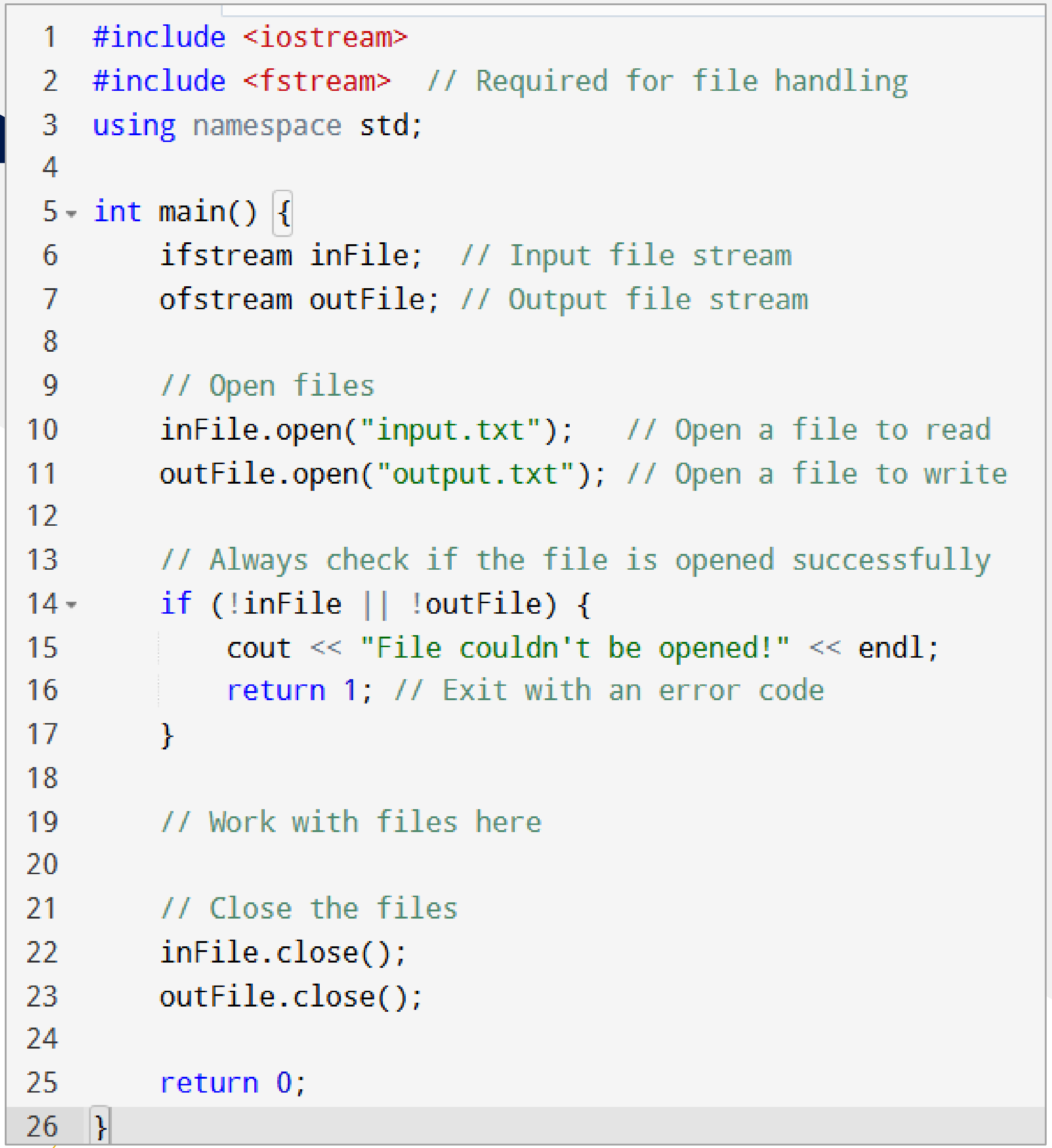
- File handling is an essential feature in programming, allowing us to store data in files and retrieve it as needed. In C++, file handling is supported by the <fstream> library, which provides classes and functions for working with files.
- File handling in C++ is based on the concept of "streams." A stream is an abstraction that represents a source or destination of data.
- There are three types of file streams:
  - Input File Stream (ifstream): Used to read data from a file.
  - Output File Stream (ofstream): Used to write data to a file.
  - File Stream (fstream): Used for both reading and writing.

File Handling through C++ Classes

- The <fstream> library provides these classes:
  - ifstream (for input)
  - ofstream (for output)
  - fstream (for both input and output)

Opening and Closing Files

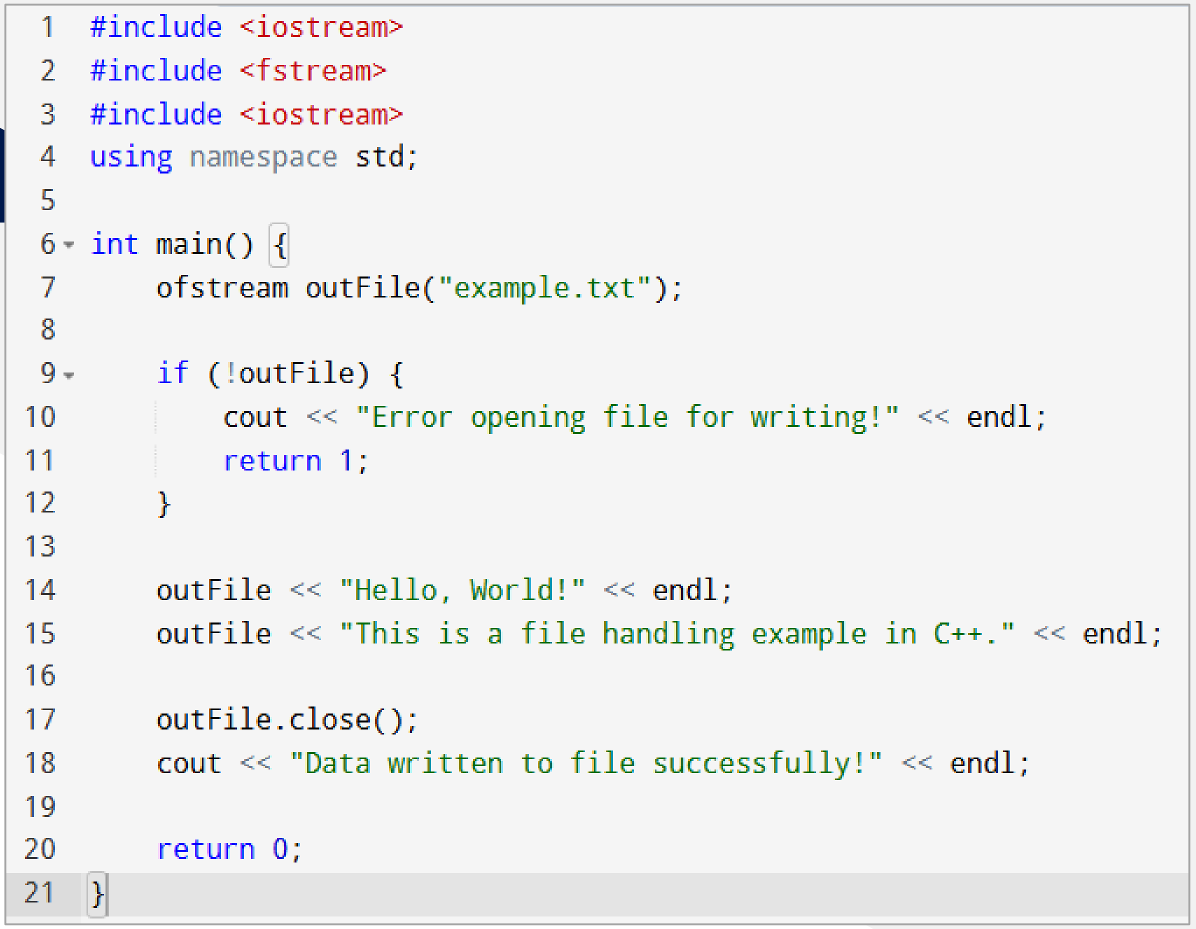
- To work with files in C++, we need to open them.
- The open() function is used to open a file, and files can be closed using the close() function.
- In this example, input.txt is opened for reading and output.txt for writing.



```cpp
#include <iostream>
#include <fstream>  // Required for file handling
using namespace std;

int main() {
    ifstream inFile;  // Input file stream
    ofstream outFile; // Output file stream

    // Open files
    inFile.open("input.txt");   // Open a file to read
    outFile.open("output.txt"); // Open a file to write

    // Always check if the file is opened successfully
    if (!inFile || !outFile) {
        cout << "File couldn't be opened!" << endl;
        return 1; // Exit with an error code
    }

    // Work with files here

    // Close the files
    inFile.close();
    outFile.close();

    return 0;
}
```

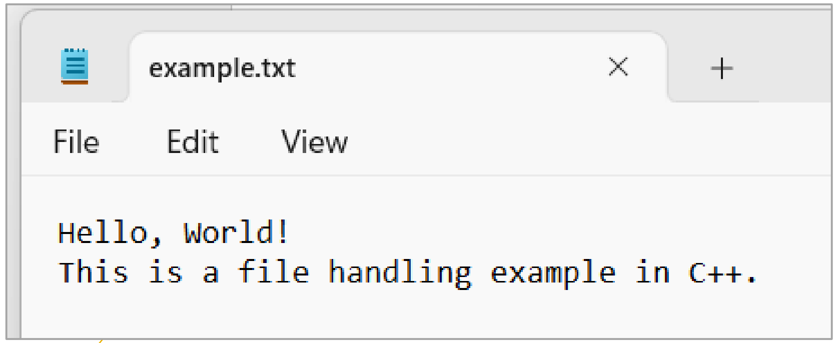
Writing to a File

- The ofstream object is used to write data to a file. Once the file is open, we can use the insertion operator (<<) to write data.



```cpp
#include <iostream>
#include <fstream>
#include <iostream>
using namespace std;

int main() {
    ofstream outFile("example.txt");

    if (!outFile) {
        cout << "Error opening file for writing!" << endl;
        return 1;
    }

    outFile << "Hello, World!" << endl;
    outFile << "This is a file handling example in C++." << endl;

    outFile.close();
    cout << "Data written to file successfully!" << endl;

    return 0;
}
```

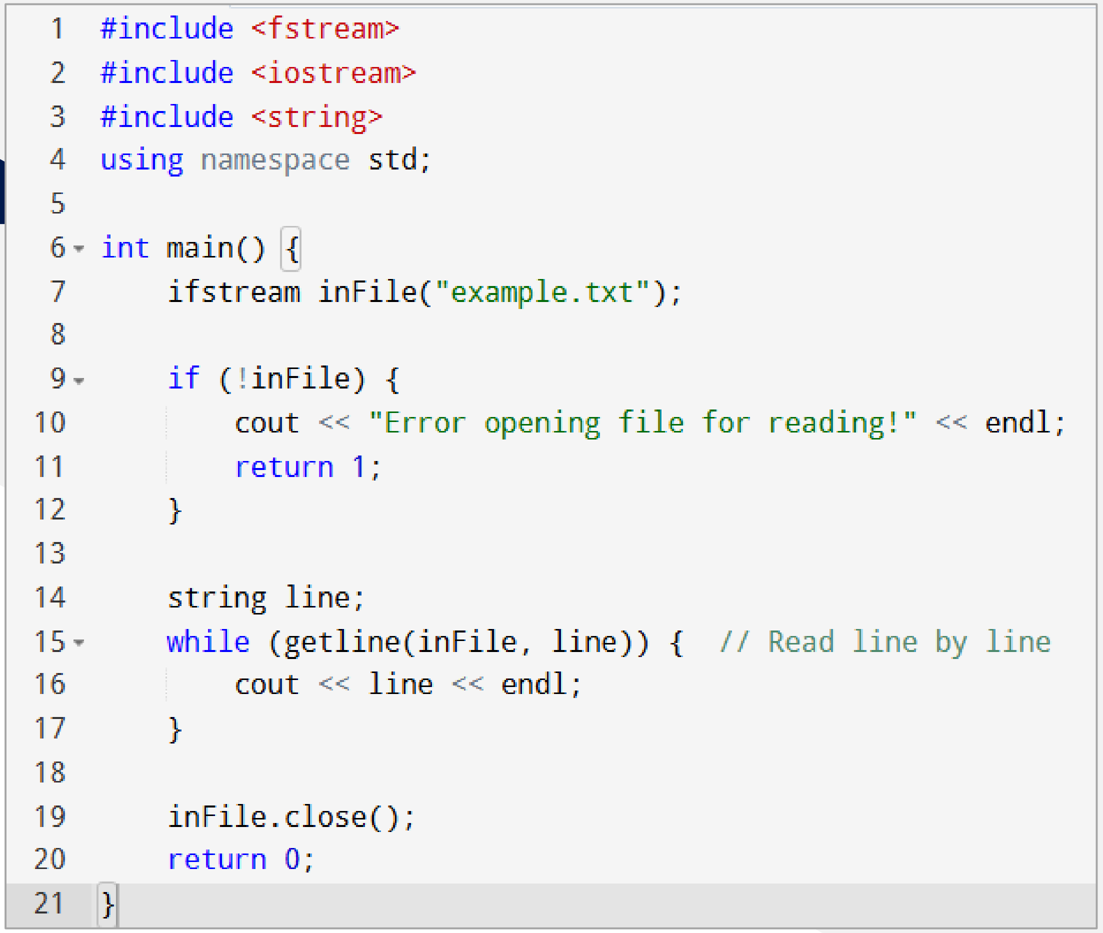
- This code will create a file called example.txt (or overwrite it if it exists) and write two lines of text to it.



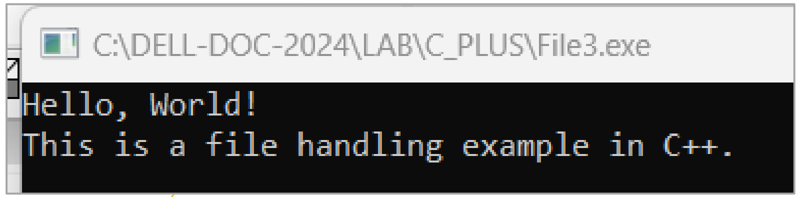
```
Hello, World!
This is a file handling example in C++.
```

Reading from a File

- The ifstream object is used to read data from a file. The extraction operator (>>) or getline() function can be used for reading.



```cpp
#include <fstream>
#include <iostream>
#include <string>
using namespace std;

int main() {
    ifstream inFile("example.txt");

    if (!inFile) {
        cout << "Error opening file for reading!" << endl;
        return 1;
    }

    string line;
    while (getline(inFile, line)) {  // Read line by line
        cout << line << endl;
    }

    inFile.close();
    return 0;
}
```



C:\DELL-DOC-2024\LAB\C_PLUS\File3.exe

```
Hello, World!
This is a file handling example in C++.
```

- In this example, each line from example.txt is read and printed to the console.

File Modes

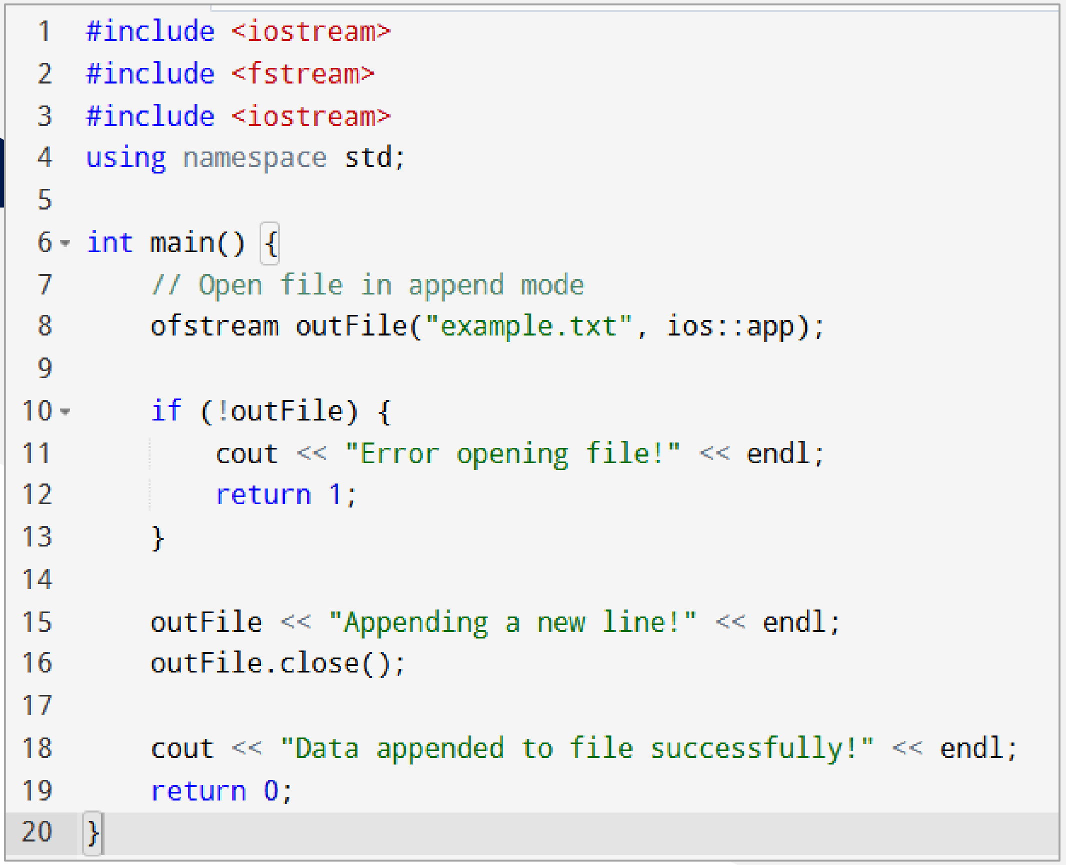
- When opening files, we can specify various modes to control how the file is accessed:
  - ios::in – Open for reading.
  - ios::out – Open for writing.
  - ios::app – Append to the end of the file.
  - ios::trunc – Truncate the file (delete content if the file exists).
  - ios::binary – Open the file in binary mode.

File Modes

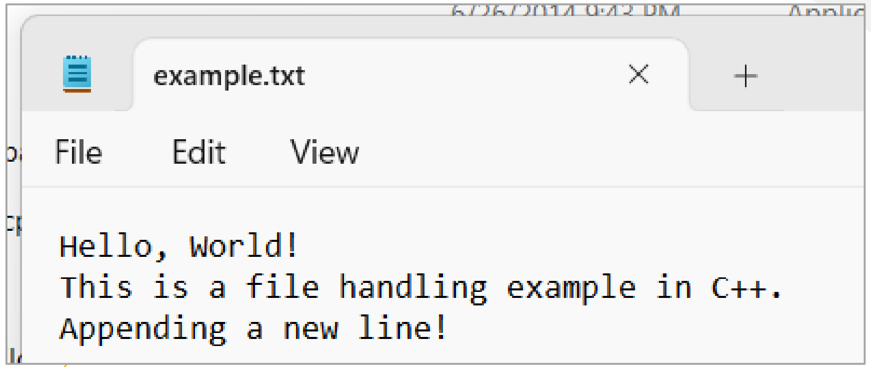
- When opening files, we can specify various modes to control how the file is accessed:
  - ios::in – Open for reading.
  - ios::out – Open for writing.
  - ios::app – Append to the end of the file.
  - ios::trunc – Truncate the file (delete content if the file exists).
  - ios::binary – Open the file in binary mode.

File Modes

- In this example, the file example.txt is opened in append mode using ios::app, so new content will be added at the end without overwriting existing content.



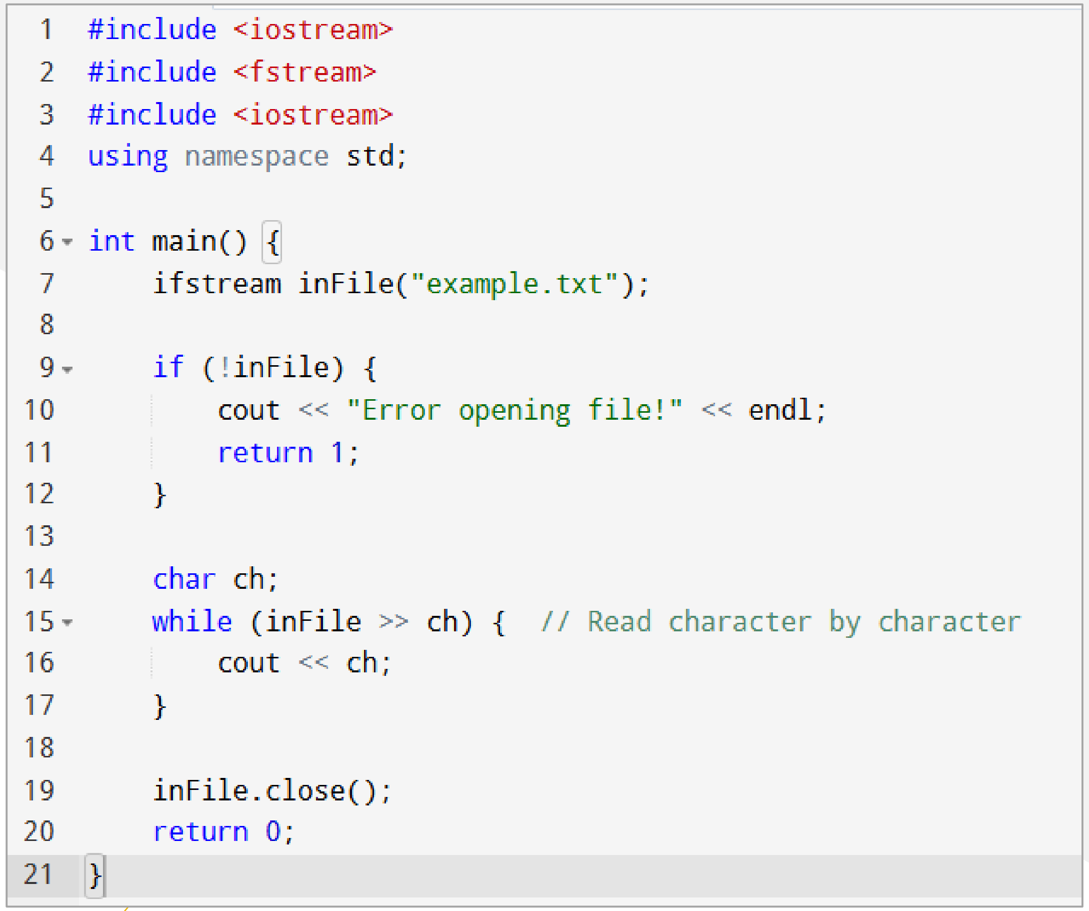
```cpp
#include <iostream>
#include <fstream>
#include <iostream>
using namespace std;

int main() {
    // Open file in append mode
    ofstream outFile("example.txt", ios::app);

    if (!outFile) {
        cout << "Error opening file!" << endl;
        return 1;
    }

    outFile << "Appending a new line!" << endl;
    outFile.close();

    cout << "Data appended to file successfully!" << endl;
    return 0;
}
```



```
Hello, World!
This is a file handling example in C++.
Appending a new line!
```

Checking End of File (EOF)

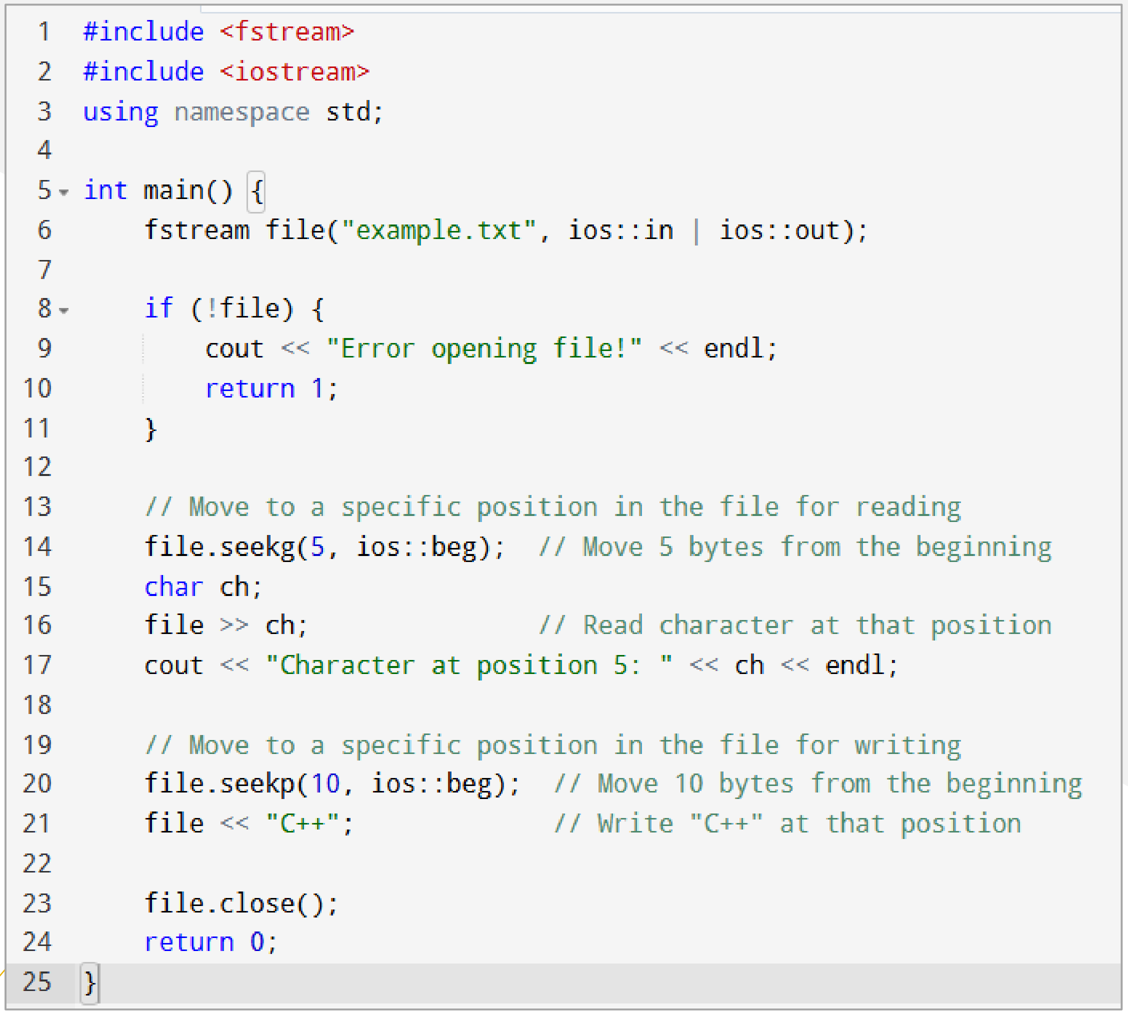
- While reading from a file, it’s essential to check if we’ve reached the end.
- The eof() function returns true when the end of the file is reached.
- This example reads and prints each character in the file until the end.



```cpp
#include <iostream>
#include <fstream>
#include <iostream>
using namespace std;

int main() {
    ifstream inFile("example.txt");

    if (!inFile) {
        cout << "Error opening file!" << endl;
        return 1;
    }

    char ch;
    while (inFile >> ch) {  // Read character by character
        cout << ch;
    }

    inFile.close();
    return 0;
}
```

File Pointers and Random Access

- File pointers allow for random access within files. There are two pointers:
  - tellg() and seekg() for ifstream to get and set the read position.
  - tellp() and seekp() for ofstream to get and set the write position.



```cpp
#include <fstream>
#include <iostream>
using namespace std;

int main() {
    fstream file("example.txt", ios::in | ios::out);

    if (!file) {
        cout << "Error opening file!" << endl;
        return 1;
    }

    // Move to a specific position in the file for reading
    file.seekg(5, ios::beg);  // Move 5 bytes from the beginning
    char ch;
    file >> ch;               // Read character at that position
    cout << "Character at position 5: " << ch << endl;

    // Move to a specific position in the file for writing
    file.seekp(10, ios::beg);  // Move 10 bytes from the beginning
    file << "C++";             // Write "C++" at that position

    file.close();
    return 0;
}
```

Lecture 11

File Handling in C++

? **THE END**