What is File Handling?

File handling in C++ allows us to **store data permanently** in files (like `.txt` or `.dat`) and **read/write/update/delete** data when needed.

C++ provides file handling through the **`fstream` library**:

```
#include <fstream>
```

File Streams in C++

Class	Used For	Header File
`ofstream`	Writing to a file	`<fstream>`
`ifstream`	Reading from a file	`<fstream>`
`fstream`	Both read and write	`<fstream>`

Opening and Closing Files

Syntax:

```
fstream file;
file.open("filename.txt", ios::in | ios::out);
file.close();
```

File Modes:

Mode	Meaning
`ios::in`	Open file for reading
`ios::out`	Open file for writing
`ios::app`	Append to the file
`ios::ate`	Open and move pointer to end
`ios::binary`	Open file in binary mode
`ios::trunc`	Delete old data and write new

1. Writing to a File

Example:

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

```
int main() {
    ofstream file("example.txt");  // Create and open file
    file << "Hello, this is a C++ file handling example.\n";
    file << "We are writing data into a file.\n";
    file.close();  // Close the file
    cout << "Data written successfully!\n";
    return 0;
}
```

Output (in example.txt):

```
Hello, this is a C++ file handling example.
We are writing data into a file.
```

2. Reading from a File

Example (read line by line):

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

int main() {
    ifstream file("example.txt");
    string line;

    if (!file) {
        cout << "File not found!";
        return 0;
    }

    while (getline(file, line)) {
        cout << line << endl;
    }

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

Output:

```
Hello, this is a C++ file handling example.
We are writing data into a file.
```

3. Append (Update) Data in File

Example:

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

int main() {
    ofstream file("example.txt", ios::app);  // Append mode
    file << "New line added to the existing file.\n";
    file.close();
    cout << "File updated successfully!\n";
    return 0;
```

```
}
```

This adds new content **without deleting old data**.

4. Update Specific Content (Read + Modify + Write)

You cannot directly "edit" inside a file; instead:

1. Read file content.
2. Modify the text in memory.
3. Write back to the same file.

Example:

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

int main() {
    string line, content = "";
    ifstream fin("example.txt");

    while (getline(fin, line)) {
        if (line == "We are writing data into a file.")
            line = "This line has been updated!";
        content += line + "\n";
    }
    fin.close();

    ofstream fout("example.txt");
    fout << content;
    fout.close();

    cout << "File updated successfully!\n";
    return 0;
}
```

5. Delete a File

Example:

```
#include <iostream>
#include <cstdio> // for remove()
using namespace std;

int main() {
    if (remove("example.txt") == 0)
        cout << "File deleted successfully!\n";
    else
        cout << "Error deleting the file!\n";
    return 0;
}
```

6. Reading a File Character by Character

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

```
using namespace std;

int main() {
    ifstream file("example.txt");
    char ch;
    while (file.get(ch)) {
        cout << ch;
    }
    file.close();
    return 0;
}
```

7. Reading and Writing Using `fstream` (Both)

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

int main() {
    fstream file;
    file.open("data.txt", ios::out);
    file << "Hello from fstream!\n";
    file.close();

    file.open("data.txt", ios::in);
    string line;
    while (getline(file, line)) {
        cout << line << endl;
    }
    file.close();

    return 0;
}
```

Summary Table

Operation	**Class/Mode**	**Function Used**
Write	`ofstream`, `ios::out`	`<<`
Read	`ifstream`, `ios::in`	`getline()` or `>>`
Append	`ofstream`, `ios::app`	`<<`
Update	`fstream`	Read → Modify → Write
Delete	`<cstdio>`	`remove("file.txt")`