Tutorial 62 - File I/O in C++ – open() and eof() Functions

Introduction

- In this tutorial, we will learn about the `open()` and `eof()` functions used in file handling.
- We have previously learned **two ways to open a file**:
  a. **Using a constructor** → `ofstream out("file.txt");`
  b. **Using the `open()` function** → `out.open("file.txt");` (covered in this tutorial)

Using the `open()` Function

📌 **Purpose:**

- The `open()` function is used to **connect a file** to a C++ program.
- Unlike the constructor method, we first **declare the file object** and then use `open()`.

📌 **Code Example: Writing to a File Using `open()`**

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

using namespace std;

int main() {
    // Declaring an object of ofstream
    ofstream out;

    // Connecting the object to the file using open()
    out.open("sample60.txt");

    // Writing to the file
    out << "This is me\n";
    out << "This is also me";

    // Closing the file connection
    out.close();

    return 0;
}

```

Explanation:

1. **Declare an `ofstream` object** → `ofstream out;`
2. **Open the file using `open()`** → `out.open("sample60.txt");`
3. **Write content to the file** using `out << "text";`
4. **Close the file** using `out.close();`

📌 **File Content After Writing:**

```
This is me
This is also me

```

✅ **Closing the file is important** to ensure that data is properly saved and the file is free for other operations.

Using the `eof()` Function

📌 **Purpose:**

- `eof()` stands for **End-of-File**.
- It returns `true (1)` when the file **reaches the end**.
- It returns `false (0)` if there is still content left to read.

📌 **Code Example: Reading a File Using `eof()`**

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

using namespace std;

int main() {
    // Declaring an object of ifstream
    ifstream in;

    // Declaring string variable st
    string st;

    // Opening the file using open()
    in.open("sample60.txt");

    // Reading until end-of-file (eof)
    while (!in.eof()) {
        // Using getline to read full lines
        getline(in, st);
        cout << st << endl;
    }

    // Closing the file (Good Practice)
    in.close();

    return 0;
}

```

Explanation:

1. **Declare an `ifstream` object** → `ifstream in;`
2. **Open the file using `open()`** → `in.open("sample60.txt");`
3. **Use a `while` loop** to read until the **end of the file** (`eof() == 1`).
4. **Use `getline()`** to read each full line.
5. **Print the content** line by line.
6. **Close the file** using `in.close();`

📌 **Output (If File Contains):**

```
This is me
This is also me

```

✅ `eof()` ensures that the **entire file is read**, not just the first word.

Summary Table

Function	Purpose
`open("file.txt")`	Opens a file for reading/writing
`close()`	Closes the file after operation
`eof()`	Returns `true` if the end of the file is reached
`getline()`	Reads an entire line from the file

Short Notes

Opening a File: Two Methods

1. **Using Constructor:** `ofstream out("file.txt");`
2. **Using `open()` Function:**

```
ofstream out;
out.open("file.txt");

```

Closing a File

- Always close files after use:

```
out.close();  // For writing
in.close();   // For reading

```

Reading a File Until End

- **Using `eof()` in a loop**:

```
while (!in.eof()) {
    getline(in, st);
    cout << st << endl;
}

```

- Ensures **complete file reading**.

Final Thoughts

✅ We learned:

- How to **open files using** `open()` instead of constructors.
- How to **read entire files using** `eof()`.
- The importance of **closing files** after operations.

📌 **Next Topic: C++ Templates** (important for competitive programming). 🚀
Let me know if you need further improvements! 🎯