Tutorial 61 - File I/O in C++: Read/Write in the Same Program & Closing Files

Introduction

- In this tutorial, we will learn how to **read from** and **write to** a file in the same program using a constructor.
- We will also explore how to properly **close files** using `.close()`.

Quick Revision of File I/O Classes

C++ provides **three main classes** for file handling from the `<fstream>` library:

1. `fstreambase` → Base class for file operations.
2. `ifstream` → Used for reading from files.
3. `ofstream` → Used for writing to files.

📌 Including the Required Header File:

```
#include <fstream>

```

Writing to a File in C++

📌 Code Example: Writing to a File

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

using namespace std;

int main() {
    // Connecting file with hout stream
    ofstream hout("sample60.txt");  // Write operation

    // Taking input from user
    string name;
    cout << "Enter your name: ";
    cin >> name;

    // Writing the input to the file
    hout << name + " is my name";

    // Closing the file
    hout.close();

    return 0;
}

```

Explanation:

1. The **ofstream object** `hout` establishes a connection with `"sample60.txt"`.
2. The user **enters a name**, which is stored in the **string** `name`.

3. The **name is written into the file** using `hout << name + " is my name";`.
4. The file is **closed** using `hout.close();`.

📌 **File Content After Writing:**

```
Harry is my name

```

Reading from a File in C++

📌 **Code Example: Reading from a File**

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

using namespace std;

int main() {
    // Connecting file with hin stream
    ifstream hin("sample60.txt");  // Read operation

    // Creating a content variable
    string content;
    hin >> content;  // Reading first word only

    // Displaying file content
    cout << "The content of the file is: " << content;

    // Closing the file
    hin.close();

    return 0;
}

```

Explanation:

1. The **ifstream object** `hin` establishes a connection with `"sample60.txt"`.
2. The **content of the file is read into** `content` using `hin >> content;`.
3. The content is **printed** using `cout`.
4. The file is **closed** using `hin.close();`.

📌 **Output (If File Contains `"Harry is my name"`):**

```
The content of the file is: Harry

```

👉 Since `hin >> content;` reads only **one word**, it stops at the first space.

Solution: Reading the Whole Line

To read the **entire content**, use `getline()`:

```
getline(hin, content);

```

Combining Read & Write in the Same Program

📌 Full Program: Reading and Writing in the Same Program

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

using namespace std;

int main() {
    // Step 1: Writing to File
    ofstream hout("sample60.txt");  // Connecting file with hout
    string name;
    cout << "Enter your name: ";
    cin >> name;
    hout << name + " is my name";
    hout.close();  // Closing file after writing

    // Step 2: Reading from File
    ifstream hin("sample60.txt");  // Connecting file with hin
    string content;
    hin >> content;  // Reads only first word
    cout << "The content of the file is: " << content;
    hin.close();  // Closing file after reading

    return 0;
}

```

Workflow:

1. **File is created & opened** (`sample60.txt`).
2. **User input is taken** and written into the file.
3. **File is closed** after writing.
4. **File is reopened for reading**, and the first word is read.
5. **File is closed again** after reading.

📌 Output Example:

```
Enter your name: Harry
The content of the file is: Harry

```

Short Notes

Key File I/O Classes

Class Name	Purpose
`ofstream`	Writing to files
`ifstream`	Reading from files
`fstream`	Both reading & writing

Opening & Closing Files

- **Write Mode:** `ofstream hout("file.txt");`
- **Read Mode:** `ifstream hin("file.txt");`
- **Closing File:** `<object_name>.close();`

Key Functions

Function	Purpose
`<<`	Writes to a file
`>>`	Reads only one word from a file
`getline()`	Reads a full line from a file
`.close()`	Disconnects file from program

👉 **Tip:** Always **close the file** after reading or writing to prevent errors!