



Without practice, your knowledge is poison.

# Wollo University, Kombolcha

## Institute of Technology

### College of Informatics

#### Department of Information System

By Daniel G.

May/2021 G.C



INSY2031

# Chapter Four

## File Operations (File Input/output)



# Contents

- ◇ Introduction
- ◇ Stream classes
  - ◇ Writing and reading modes
  - ◇ Writing to and reading from files
  - ◇ Types of files (Text and Binary)
  - ◇ File access methods (sequential and random access files)





# Basic concept of File Operations

1

A file is a collection of data stored in one unit, identified by a filename.

Files are used to store data in a storage device permanently.

File handling provides a mechanism to store the output of a program in a file and to perform various operations on it.



# Stream classes

In C++, files are mainly dealt by using three classes `fstream`, `ifstream`, `ofstream`.

1

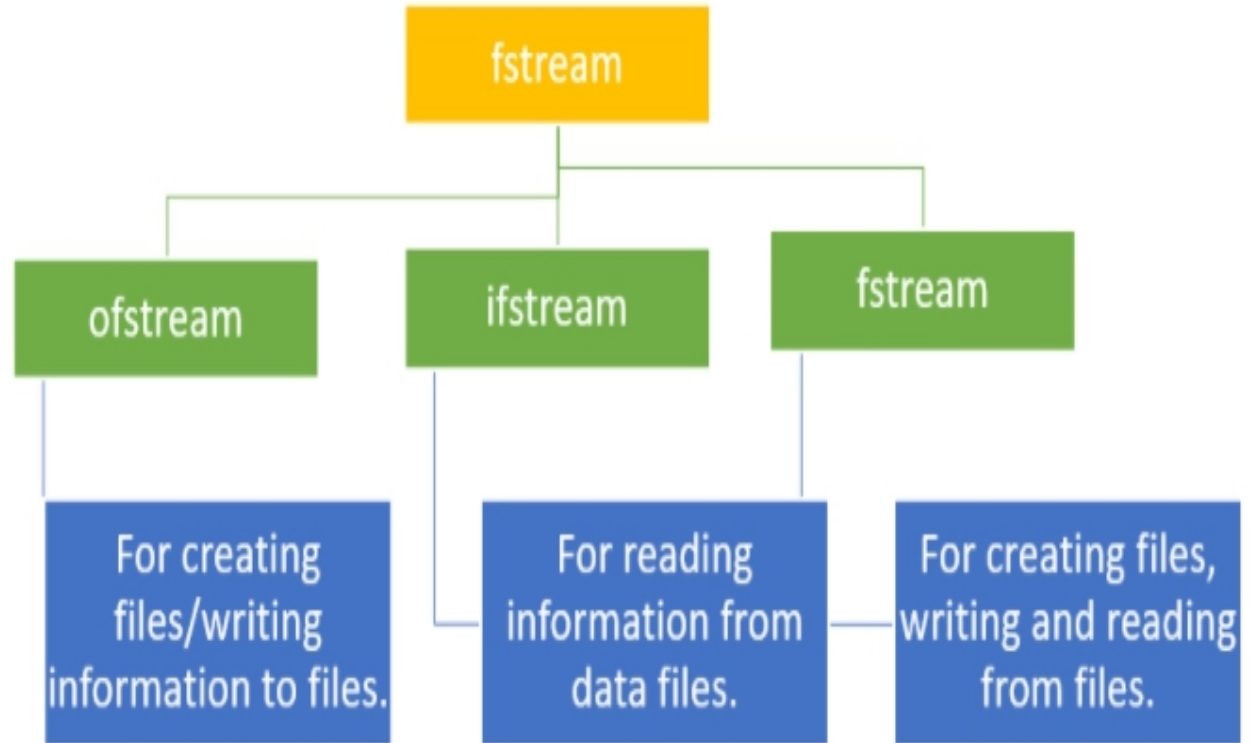
**Ofstream:** This Stream class signifies the output file stream and is applied to create files for writing information to files

**Ifstream:** This Stream class signifies the input file stream and is applied for reading information from files

**Fstream:** This Stream class can be used for both read and write from/to files.

# Conti...

1





## Conti....

C++ provides us with the following operations in File Handling:

1

- ❑ Creating a file: `open()`
- ❑ Reading data: `read()`
- ❑ Writing new data: `write()`
- ❑ Closing a file: `close()`



# Open Files

Before performing any operation on a file, you must first open it. If you need to write to the file, open it using `fstream` or `ofstream` objects

`open` (`file_name`, `mode`);

- ❑ The `file_name` parameter denotes the name of the file to open.
- ❑ The `mode` parameter is optional. It can take any of the following values:



# 1

<i><b>Modes</b></i>	<i><b>Description</b></i>
in	Opens the file to read(default for ifstream)
out	Opens the file to write(default for ofstream)
binary	Opens the file in binary mode
app	Opens the file and appends all the outputs at the end
ate	Opens the file and moves the control to the end of the file
trunc	Removes the data in the existing file
nocreate	Opens the file only if it already exists
noreplace	Opens the file only if it does not already exist

It is possible to use two modes at the same time. You combine them using the | (OR) operator.



## Conti..

1

Example of file opened for writing:

```
st.open("E:\studytonight.txt",ios::out);
```

Example of file opened for reading:

```
st.open("E:\studytonight.txt",ios::in);
```

Example of file opened for appending:

```
st.open("E:\studytonight.txt",ios::app);
```

Example of file opened for truncating:

```
st.open("E:\studytonight.txt",ios::trunc);
```

# Example

```
#include<iostream>
#include <fstream>
using namespace std;
int main(){
    fstream st; // Step 1: Creating object of fstream class
    st.open("E:/studytonight.txt",ios::out); // Step 2:
    Creating new file
    if(!st) // Step 3: Checking whether file exist{
        cout<<"File creation failed";}
    else{
        cout<<"New file created";
        st.close(); // Step 4: Closing file}
    return 0;
}
```

1



## Writing to a File

1

While doing C++ programming, you write information to a file from your program using the stream insertion operator (<<) just as you use that operator to output information to the screen. The only difference is that you use an ofstream or fstream object instead of the cout object.



1

```
#include <iostream>
#include <fstream>
using namespace std;
int main(){
    fstream new_file;
    new_file.open("D:/write.txt",ios::out);
    if(!new_file) {
        cout<<"File creation failed";
    }
    else{
        cout<<"New file created";
        new_file<<"Welcome to write file";    //Writing to file
        new_file.close();
    }
    return 0;
}
```

# Example



# Read from Files

1

You can read information from files into your C++ program. This is possible using stream extraction operator (`>>`). You use the operator in the same way you use it to read user input from the keyboard. However, instead of using the `cin` object, you use the `ifstream/ fstream` object.

# Example

```
#include <iostream>
#include <fstream>
using namespace std;
int main(){
    fstream st; // step 1: Creating object of fstream class
    st.open("D:/myfile.txt",ios::in); // Step 2: Creating new file
    if(!st) // Step 3: Checking whether file exist
    {
        cout<<"No such file";}
    else{
        string ch;
        while (!st.eof()){
            st >>ch; // Step 4: Reading from file
            cout << ch<<endl; // Message Read from file
        }
        st.close(); // Step 5: Closing file
    }
    return 0;
}
```

1



# Types of files (Text and Binary)

1

A text file stores data in the form of alphabets, digits and other special symbols by storing their ASCII values and are in a human-readable format. ... whereas binary file contains a sequence or a collection of bytes which are not in a human-readable format.





## Conti...

The C++ language supports two types of files:

- Text files
- Binary files

1

The basic difference between text files and binary files is that in text files various character translations are performed such as “\r+\f” is converted into “\n”, whereas in binary files no such translations are performed.

By default, C++ opens the files in text mode.

## Conti...

- For writing data

### Text Files

```
ofstream out  
("myfile.txt");  
  
or  
  
ofstream out;  
out.open("myfile.txt");
```

### Binary Files

```
ofstream out  
("myfile.txt",ios::binary);  
  
or  
  
ofstream out;  
out.open("myfile.txt", ios::binary);
```

- For Appending (adding text at the end of the existing file)

### Text Files

```
ofstream  
out("myfile.txt",ios::app);  
  
or  
  
ofstream out;  
out.open("myfile.txt",  
ios::app);
```

### Binary Files

```
ofstream out  
("myfile.txt",ios::app|ios::binary);  
  
or  
  
ofstream out;  
out.open("myfile.txt", ios::app |  
ios::binary);
```

1



# Thanks!

## Any questions?

Feel free!

