

**LAB 14**

# Summary

|  |  |
| --- | --- |
| **Items** | **Description** |
| Course Title | Programming Fundamentals |
| Lab Title | File I/O |
| Duration | 3 Hours |
| Operating  System/Tool/Language | Ubuntu/ g++/ C++ |
| Objective | To understand the concept of saving data to files on disk |

## The C++ programming skills that are expected to be acquired in this lab:

* To understand the concept of saving data to files on disk
* To understand the syntax of opening and closing a file.
* To learn data I/O using **text** files.

**File**

A file is a collection of data that is usually stored on a computer’s disk. Data can be saved to files and then later reused. The information/data stored under a specific name on a storage device, is called a file. Files store data permanently in a storage device. With file handling, the output from a program can be stored in a file. Various operations can be performed on the data while in the file.

**File I/O**

C++ provides the following classes to perform output and input of characters to/from files:

* **fstream (File Stream):** Stream class to both read and write from/to files.
* **ofstream (Output File Stream):** Stream class to write on files
* **ifstream (Input File Stream):** Stream class to read from files

These classes are derived directly or indirectly from the classes istream, and ostream. We have already used objects whose types were these classes: cin is an object of class istream



and cout is an object of class ostream. Therfore, we have already been using classes that are related to our file streams. And in fact, we can use our file streams the same way we are already used to use cin and cout, with the only difference that we have to associate these streams with physical files.

## Creating a File Example 14.1

// basic file operations #include <iostream> #include <fstream> using namespace std;

int main () { ofstream myfile;

myfile.open ("example.txt");

//writing to a file

myfile << "Writing this to a file.\n";

//closing the file myfile.close();

return 0;

}

[file example.txt] Writing this to a file.

This code creates a file called example.txt and inserts a sentence into it in the same way we are used to do with cout, but using the file stream myfile instead.

## Opening a File

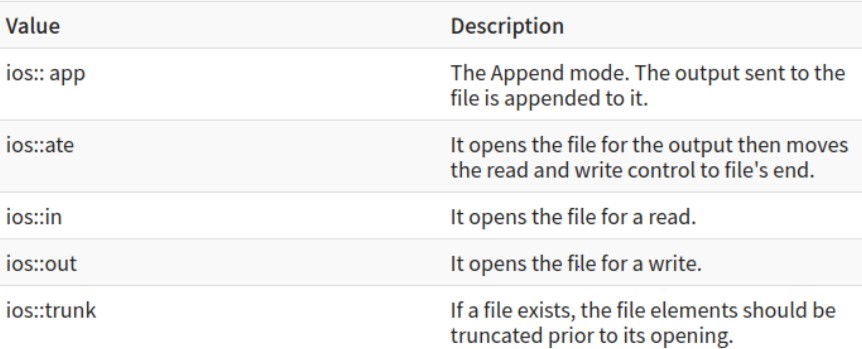
The first operation generally performed on an object of one of these classes is to associate it to a real file. This procedure is known as to *open a file*. An open file is represented within a



program by a stream object (an instantiation of one of these classes, in the previous example this was myfile) and any input or output operation performed on this stream object will be applied to the physical file associated to it.

In order to open a file with a stream object we use its member function open(): open (filename, mode);

Where filename is a null-terminated character sequence of type const char \* (the same type that string literals have) representing the name of the file to be opened, and mode is an optional parameter with a combination of the following flags:

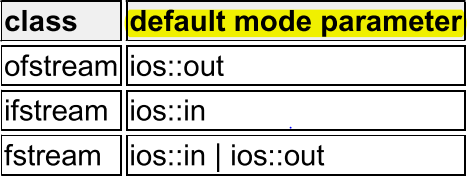


All these flags can be combined using the bitwise operator OR (|). For example, if we want to open the file example.txt to add data we could do it by the following call to member function open():

ofstram myfile;

myfile.open ("example.bin", ios::out | ios::app );

Each one of the open() member functions of the classes ofstream, ifstream and fstream has a default mode that is used if the file is opened without a second argument:





Since the first task that is performed on a file stream object is generally to open a file, these three classes include a constructor that automatically calls the open() member function and has the exact same parameters as this member. Therefore, we could also have declared the previous myfile object and conducted the same opening operation in our previous example by writing:

ofstream myfile ("example.bin", ios::out | ios::app);

Combining object construction and stream opening in a single statement. Both forms to open a file are valid and equivalent.

To check if a file stream was successful opening a file, you can do it by calling to member is\_open() with no arguments. This member function returns a bool value of true in the case that indeed the stream object is associated with an open file, or false otherwise:

if (myfile.is\_open()) { /\* ok, proceed with output \*/ }

* 1. **Writing to a File**

You can write to file right from your C++ program. You use stream insertion operator (<<) for this. The text to be written to the file should be enclosed within double-quotes.

// Write to the file

myfile << "This is a line.\n";

myfile << "This is another line.\n";

* 1. **Reading From a File**

You can read information from files into your C++ program 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.

//read the file line by line getline (myfile,line);

// Output the text from the file cout << line << endl;



## Closing a File

When we are finished with our input and output operations on a file we shall close it so that its resources become available again. In order to do that we have to call the stream's member function close(). This member function takes no parameters, and what it does is to flush the associated buffers and close the file:

myfile.close();

Once this member function is called, the stream object can be used to open another file, and the file is available again to be opened by other processes.

In case that an object is destructed while still associated with an open file, the destructor automatically calls the member function close().

# Text Files

These files are designed to store text and thus all values that we input or output from/to them can suffer some formatting transformations, which do not necessarily correspond to their literal binary value.

Data output operations on text files are performed in the same way we operated with cout.

## Example 14.2

// writing on a text file #include <iostream> #include <fstream> using namespace std; int main () {

// Create and open a text file ofstream myfile ("example.txt"); if (myfile.is\_open())

{

// Write to the file

myfile << "This is a line.\n";

myfile << "This is another line.\n"; myfile.close();

}

else

cout << "Unable to open file"; return 0;

}



[file example.txt] This is a line.

This is another line

Data input from a file can also be performed in the same way that we did with cin:

## Example 14.3

// reading a text file #include <iostream> #include <fstream> using namespace std; int main() {

fstream myfile; myfile.open("example.txt", ios::in); if (!myfile) {

cout << " file not found";

}

else {

char ch;

while (1) {

myfile >> ch;

if (myfile.eof())

break;

cout << ch;

}

}

myfile.close(); return 0;

}



This last example reads a text file and prints out its content on the screen. Notice how we have used a new member function, called good() that returns true in the case that the stream is ready for input/output operations. We have created a while loop that finishes when indeed myfile.good() is no longer true, which will happen either if the end of the file has been reached or if some other error occurred.

## Checking State Flags

***eof***()

Returns true if a file open for reading has reached the end.

In order to reset the state flags checked by any of member functions we have just seen we can use the member function clear(), which takes no parameters.

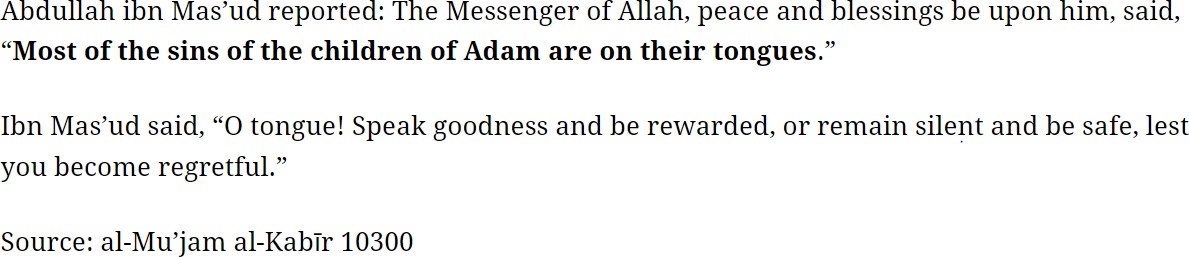
++++++++++++



**Lab Tasks**

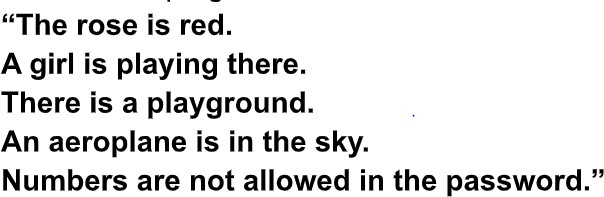
Task#01

Create a file named File1 and write the following to the file:



Task#02

Create another file File2 and write following string to it:



Use a user defined function to:

* display the Contents of text file * count number of alphabets

* count number of words

Task#03

Write a C++ Program to Merge Two Files you have created earlier (File1, File2) into a Single file (File3) using File Handling.



Task#04

Write a C++ Program to Encrypt File2 and write in another text file (File4) using File Handling.

**Hint:** (char+100)

Task#05

Write a C++ Program to Decrypt File4 and write in another text file (File5) and display the contents of File5.

Good Luck 

**Submission Instructions:**

1. Save all .cpp files with your roll no and task number

e.g. i22XXXX\_Task01.cpp

2. Now create a new folder with name ROLLNO\_LAB014 e.g. i23XXXX\_LAB02

3. Move all your .cpp files to this newly created directory and compress it into

.zip file.

4. Now you must submit this zipped file on Google Classroom.