Course Code: CL-1004	Course : Object Oriented Programming Lab

File I/O

Introduction to filing

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.

Streams

C++ I/O occurs in **streams**, which are sequences of bytes.

In **input operations**, the bytes flow from a device (e.g., a keyboard, a disk drive, a network connection, etc.) to main memory.

In **output operations**, bytes flow from main memory to a device (e.g., a display screen, a printer, a disk drive, a network connection, etc.).

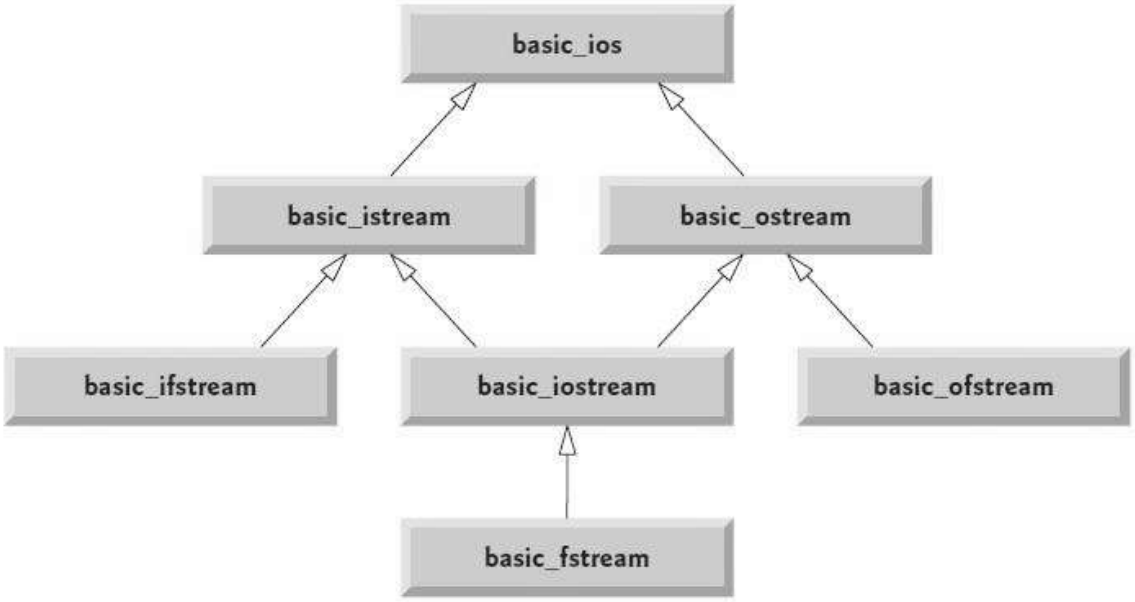
So far, we have been using the **iostream** standard library, which provides **cin** and **cout** methods for reading from standard input and writing to standard output respectively.

<iostream.h>: Contains cin & cout objects

Stream Input/Output Classes and Objects

Sr.No	Data Type & Description
1	**ofstream** This data type represents the output file stream and is used to create files and to write information to files.
2	**ifstream** This data type represents the input file stream and is used to read information from files.
3	**fstream**

Stream-I/O template hierarchy



Opening a File

Now the first step to open the particular file for read or write operation. We can open file by

1. passing file name in constructor at the time of **object creation**
2. using the open method

Open File by using constructor

ifstream (const char filename, ios_base::openmode mode = ios_base::in);*

ifstream fin(filename, openmode) by default openmode = ios::in

ifstream fin("filename");

Open File by using open method

Calling of default constructor

ifstream fin;

fin.open(filename, openmode)

fin.open("filename");

File open using constructor method

```
#include <iostream>
#include <fstream>
using namespace std;
int main(){
    ofstream my_file("XYZ");
    if (!my_file) {
        cout << "File not created!";
    }
    else {
        cout << "File created successfully!";
        my_file.close();
    }
    my_file.close();
}
```

File open using open() method

The three objects, that is, fstream, ofstream, and ifstream, have the open() function defined in them. The function takes this syntax:

```
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:

Mode Flag & Description

Mode	Description
iso::in	File opened in reading mode
iso::out	File opened in write mode
iso::app	File opened in append mode
iso::ate	File opened in append mode but read and write performed at the end of the file.
iso::binary	File opened in binary mode
iso::trunc	File opened in truncate mode

iso::nocreate	The file opens only if it exists
iso::noreplace	The file opens only if it doesn't exist

Difference between ios::app and ios::ate

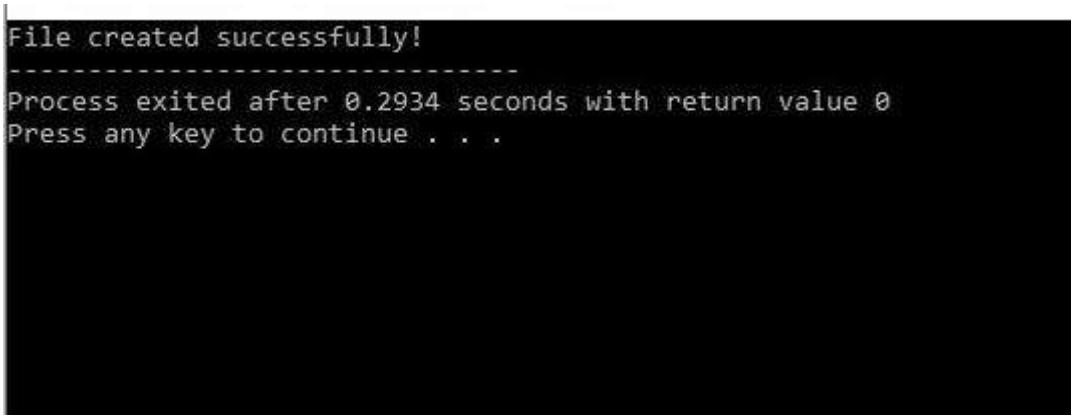
When ios::ate (also called update) is set, the initial position will be the end of the file, but you are free to seek thereafter. When ios::app is set, all output operations are performed at the end of the file. Since all writes are implicitly preceded by seeks, there is no way to write elsewhere.

File Open Mode

If you want to set more than one open mode, just use the **OR** operator- **|**. This way:

```
ios::ate | ios::binary
```

```cpp
#include <iostream>
#include <fstream>
using namespace std;
int main() {
    fstream my_file;
    my_file.open("my_file", ios::out);
    if (!my_file) {
        cout << "File not created!";
    }
    else {
        cout << "File created successfully!";
        my_file.close();
    }
    return 0;
}
```



Open()

- Opening a file associates a file stream variable declared in the program with a physical file at the source, such as a disk.
- In the case of an input file:
- the file must exist before the open statement executes.
- If the file does not exist, the open statement fails and the input stream enters the fail state
- An output file does not have to exist before it is opened;
- if the output file does not exist, the computer prepares an empty file for output.

- If the designated output file already exists, by default, the old contents are erased when the file is opened.

Validate the file before trying to access

Method 1:

```
By checking the stream variable;
If ( ! Mystream)
{
Cout << "Cannot open file.\n ";
}
```

Method 2:

```
By using bool is_open() function.
If ( ! Mystream.is_open()) {
Cout << "File is not open.\n ";
}|
```

File Processing Function

- open(): To create a file
- close(): To close an existing file
- get(): to read a single character from the file
- put(): to write a single character in the file
- read(): to read data from a file..>>
- write(): to write data into a file.. <<

How to Close Files

Once a C++ program terminates, it automatically

- flushes the streams
- releases the allocated memory
- closes opened files.

However, as a programmer, you should learn to close open files before the program terminates.

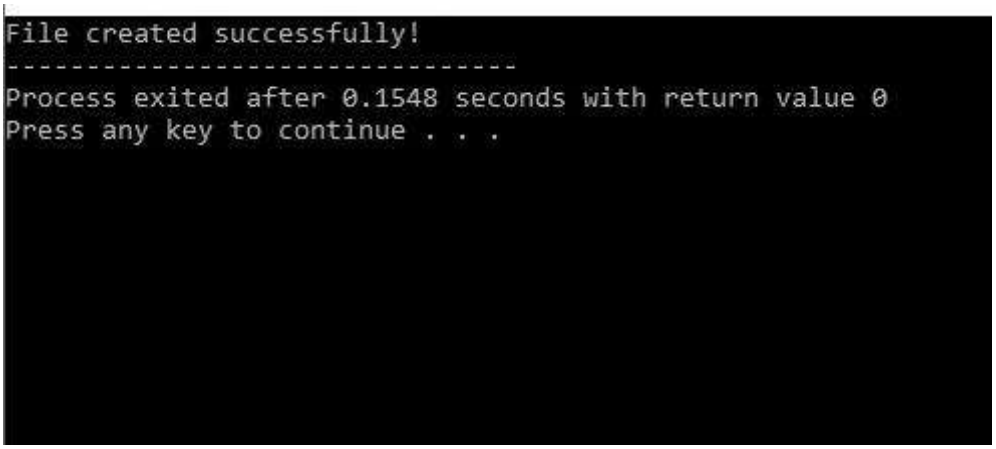
The fstream, ofstream, and ifstream objects have the close() function for closing files. The function takes this syntax:

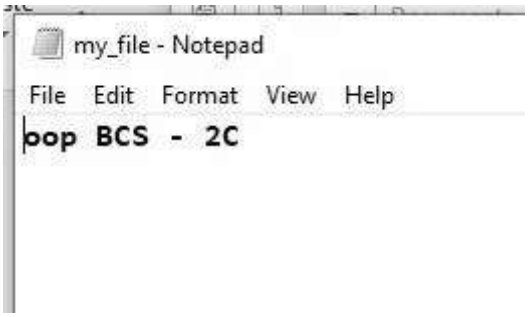
```
void close();
```

How to Write to Files

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.

```
1   #include <iostream>
2   #include <fstream>
3   using namespace std;
4   int main() {
5       fstream my_file;
6       my_file.open("my_file.txt", ios::out);
7       if (!my_file) {
8           cout << "File not created!";
9       }
10      else {
11          cout << "File created successfully!";
12          my_file << "oop BCS - 2C";
13          my_file.close();
14      }
15      return 0;
16  }
```





How to Read from Files

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.

```
#include <iostream>
#include <fstream>
using namespace std;
int main() {
    fstream my_file;
    my_file.open("my_file.txt", ios::in);
    if (!my_file) {
        cout << "No such file"; }
    else {
        char ch;
while (1) {
            my_file >> ch;
            if (my_file.eof())
                break;
cout << ch;
        }}
    my_file.close();
    return 0;
}
```

1. an else statement to state what to do if the file is found.
2. Create a char variable named ch.
3. Create a while loop for iterating over the file contents.
4. Write/store contents of the file in the variable ch.
5. Use an if condition and eof() function that is, end of the file, to ensure the compiler keeps on reading from the file if the end is not reached.
6. Use a break statement to stop reading from the file once the end is reached.
7. Print the contents of variable ch on the console.
8. End of the while body.

Using Member Function getline

as it name states, read a whole line, or at least till a delimiter that can be specified.

Syntax:

```
istream& getline(istream& is, string& str, char delim);
```

Parameters:

- **is:** It is an object of istream class and tells the function about the stream from where to read the input from.
- **str:** It is a string object, the input is stored in this object after being read from the stream.
- **delim:** It is the delimitation character which tells the function to stop reading further input after reaching this character.

Syntax: `istream& getline (istream& is, string& str);`

The second declaration is almost the same as that of the first one. The only difference is, the latter have a delimitation character which is by default new line(\n) character.

Read a line

```
1   #include <iostream>
2   #include <fstream>
3   using namespace std;
4   int main()
5   {
6   //Declare and open a text file
7   ifstream openFile("my_file.txt");
8   string line;
9   while(!openFile.eof())
10  {
11  //fetch line from my_file.txt and put it in a string
12  getline(openFile, line);
13  cout << line;
14  }
15  openFile.close(); // close the file
16  return 0; }
```

```
oop BCS - 2C
--------------------------------
Process exited after 0.5344 seconds with return value 0
Press any key to continue . . .
```

Read character by character

```
1  #include <iostream>
2  #include <fstream>
3  using namespace std;
4  int main()
5  {//Declare and open a text file
6  ifstream openFile("my_file.txt");
7  char ch;
8  if (!openFile) {
9          cout << "No such file";}
10     else {
11 while( ! openFile.eof() )
12 {
13 openFile.get(ch); // get one character
14 cout << ch; // display the character
15 }
16 }
17 openFile.close(); // close the file
18 return 0;
19 }
```

```
oop BCS  2F and 2C


--------------------------------
Process exited after 0.2954 seconds with return value 0
Press any key to continue . . .
```

Example

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

int main () {
   char data[100];

   // open a file in write mode.
   ofstream outfile;
   outfile.open("afile.dat");

   cout << "Writing to the file" << endl;
   cout << "Enter your name: ";
   cin.getline(data, 100);

   // write inputted data into the file.
   outfile << data << endl;

   cout << "Enter your age: ";
   cin >> data;
   cin.ignore();

   // again write inputted data into the file.
   outfile << data << endl;

   // close the opened file.
   outfile.close();

   // open a file in read mode.
   ifstream infile;
   infile.open("afile.dat");

   cout << "Reading from the file" << endl;
   infile >> data;

   // write the data at the screen.
   cout << data << endl;

   // again read the data from the file and display it.
   infile >> data;
   cout << data << endl;

   // close the opened file.
   infile.close();

   return 0;
}
```

```
Writing to the file
Enter your name: Zara
Enter your age: 9
Reading from the file
Zara
9
```

Examples make use of additional functions from cin object, like getline() function to read the line from outside and ignore() function to ignore the extra characters left by previous read statement

write() function

The write() function is used to write object or record (sequence of bytes) to the file. A record may be an array, structure or class. Syntax of write() function

```
fstream fout;
        fout.write( (char *) &obj, sizeof(obj) );
```

The write() function takes two arguments. **&obj :** Initial byte of an object stored in memory. **sizeof(obj) :** size of object represents the total number of bytes to be written from initial byte.

- ***Using the read() and write() function for binary I/O.***

```
//Writing a class object to a file using ofstream class and mode ios::out
#include<iostream>
#include<fstream>
using namespace std;
const int size = 3;
class A
{
private:
char name[40];
int age;
float height;
```

```
char gender;
char newline_chr;
public:
void putdata();
void getdata();
};

//Defining the function putdata() to allow user to enter the data member of a object.
void A :: putdata()
{
        cout<<"Enter the name : ";
        cin.getline(name,40);
        cout<<"Enter the age : ";
        cin>>age;
        cout<<"Enter the height : ";
        cin>>height;
        cout<<"Enter the gender : ";
        cin>>gender;

        //This one captures the enter key(newline character) pressed after entering the gender.
        cin.get(newline_chr);

}

void A :: getdata()
{
        cout<<"The name is : " << name << "\n";
        cout<<"The age is : " << age << "\n";
        cout<<"The height is : " << height << "\n";
```

```
        cout<<"The gender is : " << gender << "\n";
}

int main()
{

//Creating an output stream
ofstream fstream_ob;

//Calling the open function to read and write an object to/from a file
fstream_ob.open("File1.txt", ios::out| ios:: app);

//Creating an array of objects of class A
A ob1[size];

//Calling the putdata() function
for(int i=0;i<size;i++)
{
        //Calling putdata() to let user enter the values for data member of an object.
        ob1[i].putdata();
}


//Calling the write() function to write an array of objects to a file in a binary form.
fstream_ob.write( (char *) & ob1, sizeof(ob1));


cout<<"Congrats! Your array of objects is successfully written to the file \n";
```

```
//Closing the output stream
fstream_ob.close();

//Creating an intput stream
ifstream ifstream_ob;

//Calling the open function to read and write an object to/from a file
ifstream_ob.open("File1.txt", ios::in);

cout<<"\nReading an array of objects from a file : \n";

//Calling the read() function to read an array of objects from a file and transfer its content to an
empty object
ifstream_ob.read( (char *) & ob1, sizeof(ob1));

for(int i=0;i<size;i++)
{
        //Calling getdata() to read the values of an object just read
        ob1[i].getdata();
}

//Closing the input stream
ifstream_ob.close();

return 0;
}
```

Another Example of write() function

```
#include <iostream>

#include <fstream>

 using namespace std;

class student{

        int roll;

        char name[25];

        float marks;

public:

                void getdata(){

                        cout << "enter roll no" << endl;

                        cin >> roll;


                        cout << "enter  name" << endl;

                        cin >> name;


                        cout << "enter marks" << endl;

                        cin >> marks;

                }


                void addRecord()

                {
```

```
                fstream f;
                student s;
                f.open("studen.dat",ios::app | ios::binary );
                s.getdata();
                f.write((char*)&s, sizeof(s));
                f.close();

        }
};

int main(){
                student s;
                char c = 'n';
                do{
                        s.addRecord();
                        cout << "do you want to add another record";
                        cin >> c;
                } while (c == 'y' || c == 'Y');
        cout << "data written successfully";

        }
```

read() function

The read() function is used to read object (sequence of bytes) to the file. Syntax of read() function

```
        fstream fin;
    fin.read( (char *) &obj, sizeof(obj) );
```

The read() function takes two arguments. **&obj** : Initial byte of an object stored in file. **sizeof(obj) :** size of object represents the total number of bytes to be read from initial byte.

The read() function returns NULL if no data read.

Example of read() function

```
#include <iostream>

#include <fstream>

using namespace std;

 class student{

        int roll;

        char name[25];

        float marks;

        public:

        void displayStudent(){

        cout<<"Roll no: "<<roll<<endl <<"NAME: "<<name<<endl  <<"Marks: "<<marks<<endl;


        }


                void Readdata()

                {

                        fstream f;

                student s;

                        f.open("studen.dat",ios::in | ios::binary);

                        if(f.read((char*)&s,sizeof(s))){

                        cout<<endl<<endl;

                        s.displayStudent();
```

```
        }
        else{
                cout<<"Error in reading data from file...\n";
         }
}
};

int main(){
                        student s;
                         s.Readdata();
                         return 0;
                }
```