About the content

This C++ material taken from the following websites:

- https://www.tutorialspoint.com/cplusplus
- https://www.geeksforgeeks.org/cpp-tutorial
- https://www.javatpoint.com/cpp-tutorial
- https://www.programiz.com/cpp-programming
- https://www.cplusplus.com
- https://www.mygreatlearning.com/
-

UNIT-VI

Module-1

File handling Streams in C++

So far, we have been using the **iostream** standard library, which provides **cin** and **cout** objects for reading from standard input and writing to standard output respectively.
This tutorial will teach you how to read and write from a file. This requires another standard C++ library called **fstream**, which defines three new stream classes −

Sr.No	Streams & Description
1	**Ofstream** This stream class represents the output file stream and is used to create files and to write information to files.
2	**Ifstream** This stream class represents the input file stream and is used to read information from files.
3	**fstream** This stream class represents the file stream generally, and has the capabilities of both ofstream and ifstream which means it can create files, write information to files, and read information from files.

To perform file processing in C++, header files <iostream> and <fstream> must be included in your C++ source file.

File handling is used to store a data permanently in computer. Using file handling we can store our data in secondary memory (Hard disk).
How to achieve the File Handling
For achieving file handling we need to follow the following steps:-
STEP 1-Naming a file
STEP 2-Opening a file
STEP 3-Writing data into the file
STEP 4-Reading data from the file
STEP 5-Closing a file.

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

For e.g.

Open File by using constructor

```
    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");
```

Here, the first argument specifies the name and location of the file to be opened and the second argument of the **open()** member function defines the mode in which the file should be opened.

Sr.No	File opening Mode Flags & Description
1	**ios::in** Opens file for reading purpose.
2	**ios::out** Opens a file for writing purposes.
3	**ios::app** Append mode. All output to that file to be appended to the end.
4	**ios::ate** It opens the file and moves the read/write control at the End of the File. The basic difference between the ios::app and this one is that the former will always start writing from the end, but we can seek any particular position with this one.
5	**ios::trunc** If the file already exists, its contents will be truncated before opening the file.

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

following will be the syntax −

```
          ofstream outfile;
          outfile.open("file.txt", ios::out | ios::trunc );
```

Similar way, you can open a file for reading and writing purpose as follows −

```
          fstream  afile;
          afile.open("file.txt", ios::out | ios::in );
```

and also

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

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

Class	default mode parameter
Ofstream	ios::out
Ifstream	ios::in
Fstream	ios::in | ios::out

Closing a File

When a C++ program terminates it automatically flushes all the streams, release all the allocated memory and close all the opened files. But it is always a good practice that a programmer should close all the opened files before program termination.

Following is the standard syntax for close() function, which is a member of fstream, ifstream, and ofstream objects.

```
fileobject.close();
```

Writing to a File

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.

Example:

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

int main () {
  ofstream myfile;
  myfile.open ("example.txt");
  myfile << "Writing this to a file.\n";
  myfile.close();
  return 0;
}
```

Example2:

```
// writing on a text file
#include <iostream>
#include <fstream>
using namespace std;

int main () {
  ofstream myfile ("example.txt");
  if (myfile.is_open())
  {
    myfile << "This is a line.\n";
    myfile << "This is another line.\n";
    myfile.close();
  }
  else cout << "Unable to open file";
  return 0;
}
```

Output in file example.txt

```
This is a line.
This is another line.
```

is_open():- it is used to check if a file stream was successful opening a file, you can do it by calling to member is_open. 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:

Reading from a File

You read information from a file into your program using the stream extraction operator (>>) just as you use that operator to input information from the keyboard. The only difference is that you use an **ifstream** or **fstream** object instead of the **cin** object.

```
// reading a text file
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

int main () {
  string line;
  ifstream myfile ("example.txt");
  if (myfile.is_open())
  {
    while ( getline (myfile,line) )
    {
      cout << line << '\n';
    }
    myfile.close();
  }

  else cout << "Unable to open file";

  return 0;
}
```

Output:

```
This is a line.
This is another line.
```

Read and Write Example

Following is the C++ program which opens a file in reading and writing mode. After writing information entered by the user to a file named afile.dat, the program reads information from the file and outputs it onto the screen −

```
/* File Handling with C++ using ifstream & ofstream class object*/
/* To write the Content in File*/
/* Then to read the content of file*/
#include <iostream>
#include <fstream>
```

```
using namespace std;
// Driver Code
int main()
{
     // Creation of ofstream class object
     ofstream fout;
    char ch;
     string line;

    // by default ios::out mode, automatically deletes the content of file.
    //To append the content, open in ios:app like fout.open("sample.txt", ios::app)
   fout.open("sample.txt");

   // Execute a loop If file successfully opened otherwise it shows error
   do {
                       cout<< "enter a line"<<endl;
                       getline(cin, line);   // Read a Line from standard input

                       fout << line << endl;   // Write line in file
                       cout<< "enter y to continue n to exit"<<endl;
                        cin>>ch;
                       getchar(); //or we can use cin.ignore();
      }while(ch=='y'||ch=='Y');

   fout.close();    // Close the File

   ifstream fin;    // Creation of ifstream class object to read the file

   // by default mode is ios::in for ifstream class
   fin.open("sample.txt");

   // Execute a loop until EOF (End of File)
   // or u can write !fin.eof()instead of fin
   while (fin) {

       getline(fin, line);      // Read a Line from File
        cout << line << endl;       // Print line in Console

   }

   fin.close();      // Close the file
   return 0;
}
```

Output:

```
enter a line
this is rgukt
enter y to continue n to exit
y
enter a line
this is nuzvid
enter y to continue n to exit
n
this is rgukt
this is nuzvid
```

Below is the implementation by using fstream class.

```
/* File Handling with C++ using fstream class object */
/* To write the Content in File */
/* Then to read the content of file*/
```

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

using namespace std;

// Driver Code
int main()
{
    // Creation of fstream class object
    fstream fio;

    string line;

  // by default openmode = ios::in|ios::out mode
  // Automatically overwrites the content of file, To append
  // the content, open in ios:app
  // fio.open("sample.txt", ios::in|ios::out|ios::app)
  // ios::trunc mode delete all content before open a file if it is already existed

    fio.open("sample.txt", ios::trunc | ios::out | ios::in);

    // Execute a loop If file successfully Opened
        while (fio) {
                        cout<< "enter a line to continue or -1 to exit"<<endl;
                        // Read a Line from standard input
                        getline(cin, line);
                        if (line == "-1")        // Press -1 to exit
                                  break;
                       fio << line << endl;    // Write line in file
                     }

        // point read pointer at beginning of file
       fio.seekg(0, ios::beg);

         // Execute a loop until EOF (End of File)
         // we can write !fio.eof()instead of fio
         while (fio)
                {
                         getline(fio, line);    // Read a Line from File
                         cout << line << endl;   // Print line in Console
                 }
         fio.close();     // Close the file
         return 0;
}
```

Output:

```
enter a line to continue or -1 to exit
this is rgukt
enter a line to continue or -1 to exit
this is nuzvid
enter a line to continue or -1 to exit
-1
this is rgukt
```

this is nuzvid

Example to write a file handling program to store and retrieve students records

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

using namespace std;
main()
{
     char ch;
     int rno;
     float fee;
     string name;
     ofstream fout("d:/student.doc");  //calling parameterized constructor in ofstream
     do{
            cout<< "enter rno student name fee"<<endl;
            cin>> rno >>name>>fee;

            fout<<rno<<"\t"<<name<<"\t"<<fee<<endl;   //write data to the file student
            cout<< "enter y to continue…."<<endl;
             cin>>ch;
            }while(ch=='y'||ch=='Y');

     fout.close();

     ifstream fin("d:/student.doc"); //calling parameterized constructor in ifstream

/*in while !fin.eof() can use but it will print   last line two times. if we write the reading from file
statement directly in while, if  it read some thing from file it becomes true otherwise false. so
repeating last line problem also solved*/
    while (fin>>rno>>name>>fee)
     {
            cout<<endl<<rno<<"\t"<<name<<"\t"<<fee;
     }
     fin.close();

   return 0;
}
```

Append data in to a file:

Appending is nothing but adding some more data / information into already existing file. Append data into a file can be possible only if they file is already existed.

Example

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

using namespace std;
int main()
{
   // Creation of fstream class object
   fstream file;
   char ch;
   string line;
   // opening the file is appending mode and also read mode
   file.open("sample.txt",ios::app|ios::in);
   if (file.is_open())
```

```
        {
                // Execute a loop If file successfully opened
                do {
                        cout<< "enter a line"<<endl;
                        getline(cin, line);// Read a Line from standard input
                        file << line << endl; // Write line in file
                        cout<< "enter y to continue n to exit"<<endl;
                        cin>>ch;
                        //we can use the fallowing function istead of getchar();
                        cin.ignore();
                }while(ch=='y'||ch=='Y');

                 // changing the posission of a cursor pointer to begining of a file
                file.seekg(0,ios::beg);

                while (file)
                {
                        getline(file, line);    // Read a Line from File

                        cout << line << endl;   // Print line in Console
                }

                file.close();    // Close the file
         }
         else
            {
                cout << "Unable to open file";
            }
   return 0;
}
```

Writing and reading data in file using class objects

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

using namespace std;
class Test
{
        int rno;
        float fee;
        string name;

      public:
        void getdet()
        {
                cout<< "enter rno student name fee"<<endl;
                 cin>> rno >>name>>fee;
        }

                void display()
        {
                cout<<endl<<rno<<"\t"<<name<<"\t"<<fee;
        }
};

int main()
{
    Test t;   //creating object for Test class
    char ch;
```

```
        fstream file;    //creating object for fstream class

        //ios::out|ios::in is default mode in fstream class
        //ios::trunc is the mode to delete all the content in a file before opening if it is already existed
        file.open("objtest.txt",ios::out|ios::in|ios::trunc);

        do{
                t.getdet();
                file.write((char*)&t,sizeof(t));   //write data to the file
                cout<< "enter y to continue of n to exit"<<endl;
                cin>>ch;
             }while(ch=='y'||ch=='Y');

          // changing the posission of a cursor pointer to begining of a file
         file.seekg(0,ios::beg);

  /*in while !file.eof() can use but it will print   last line two times. if we write the reading from file statement directly in while, if  it read some thing from file it becomes true otherwise false. so repeating last line problem also solved*/
       while (file.read((char*)&t,sizeof(t)))
        {
                t.display();
        }
        file.close();

      return 0;
  }
```

Output:

```
enter rno student name fee
22 venky 200.230
enter y to continue of n to exit
y
enter rno student name fee
33 subbu 3000.55
enter y to continue of n to exit
y
enter rno student name fee
11 azeez 123.32
enter y to continue of n to exit
n

22      venky   200.23
33      subbu   3000.55
11      azeez   123.32
--------------------------------
Process exited after 150.9 seconds with return value 0
Press any key to continue . . .
```

File Position Pointers

Both **istream** and **ostream** provide member functions for repositioning the file-position pointer. These member functions are **seekg** ("seek get") for istream and **seekp** ("seek put") for ostream.
The argument to seekg and seekp normally is a long integer. A second argument can be specified to indicate the seek direction. The seek direction can be **ios::beg** (the default) for positioning relative to

the beginning of a stream, **ios::cur** for positioning relative to the current position in a stream or **ios::end** for positioning relative to the end of a stream.
The file-position pointer is an integer value that specifies the location in the file as a number of bytes from the file's starting location. Some examples of positioning the "get" file-position pointer are −

```
// position to the nth byte of fileObject (assumes ios::beg)
fileObject.seekg( n );

// position n bytes forward in fileObject
fileObject.seekg( n, ios::cur );

// position n bytes back from end of fileObject
fileObject.seekg( n, ios::end );

// position at end of fileObject
fileObject.seekg( 0, ios::end );
```

C++ Streams

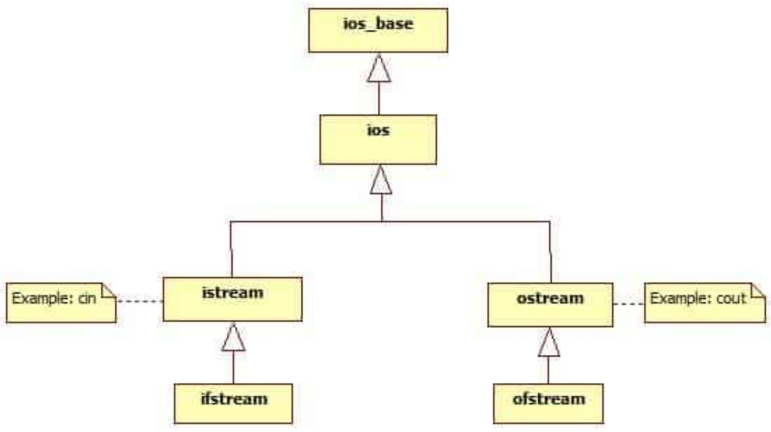
In C++, a stream refers to a sequence of characters that are transferred between the program and input/output (I/O) devices. Stream classes in C++ facilitate input and output operations on files and other I/O devices. These classes have specific features to handle program input and output, making it easier to write portable code that can be used across multiple platforms.

To use streams in C++, you need to include the appropriate header file. For instance, to use input/output streams, you would include the iostream header file. This library provides the necessary functions and classes to work with streams, enabling you to read and write data to and from files and other I/O devices.

we'll be exploring five essential C++ stream classes:

1. istream
2. ostream
3. ifstream
4. ofstream
5. fstream

Object-oriented programming, such as C++, relies heavily on the concept of inheritance.
Inheritance allows a class to inherit properties from another class that has already been defined.
Descendant classes can then add their own unique properties, making them specializations of their parent class.
Take the stream class hierarchy as an example. In the diagram below (only a portion is shown), we can see that ifstream is a specialization of istream. This means that an ifstream object is an istream object and inherits all the properties of the istream class. Additionally, it adds some additional properties of its own.



The C++ stream class hierarchy consists of a number of classes that define and provide different flows for objects in the class. The hierarchy is structured in a way that starts with the top class, which is the ios class, followed by other classes such as the istream, ostream, iostream,

The ios class is the parent class in the hierarchy and both the istream and ostream classes inherit from it. These two classes together form the ios class, which is the highest level of the entire C++ stream class hierarchy.
Other classes in the hierarchy provide functions for various operations, including assignment operations.
Let’s deep dive into the four essential stream classes:

1. istream class

This class is a general-purpose input stream and is used to read input from various sources, including the console and files. One example of an istream is cin, which is a commonly used input stream for reading input from the console. The istream class provides a range of functions for handling characters, strings, and objects, including get, getline, read, ignore, putback, and cin.

These functions enable developers to manipulate input in various ways. For example, the get function allows developers to read a single character from the input stream, while the getline function reads an entire line of text and stores it in a string object. The read function is used to read a specific number of characters from the input stream and store them in a buffer. Overall, the istream class is an essential component of input stream handling in C++.

```
#include <iostream>
using namespace std;
int main()
{
    char a;
    a=cin.get();
    cout << a;
    return 0;
}
```

2. ostream class

The ostream class is responsible for working with the output stream and provides all the necessary functions for managing chars, strings, and objects such as put, write, cout etc.

```
#include <iostream>
using namespace std;
int main()
{
```

```
char u;
u=cin.get();
cout.put(u);
return 0;
}
```

3. ifstream class

This stream class represents the input file stream and is used to read information from files. When working with the ifstream class in your code, you may come across situations where you need to read data from a file to proceed with your program. This is where file handling comes into play, and it involves using stream classes to accomplish this task.

An ifstream object represents an input file stream, which is used to read data from a file. Since an ifstream is a type of istream, any operations that can be performed on an istream can also be performed on an ifstream.

One common example of an istream is cin, which is used for standard input. Therefore, any operations that you can perform with cin can also be performed with an ifstream object.

To use ifstream (and ofstream) in your code, you need to include the fstream header by adding the following line at the beginning of your program:
#include <fstream>

4. ofstream class

This stream class represents the output file stream and is used to create files and to write information to files. and works exactly like ifstreams, except for output instead of input.
Once an ofstream is created, opened, and checked for no failures, you use it just like cout:

```
ofstream  fout( "outputFile.txt" );

fout << "The minimum oxygen percentage is " << minO2 << endl;
```

5. fstream

This stream class represents the file stream generally, and has the capabilities of both ofstream and ifstream which means it can create files, write information to files, and read information from files.

Conclusion

C++ streams are a powerful feature of the C++ programming language and provide a standardized way to handle input and output. You can read and write data to and from a variety of sources, including files, input/output devices, and even other programs. By understanding the syntax and functionality, you can write more efficient and portable code that can be used across multiple platforms.

This C++ material taken from the following websites:

- https://www.tutorialspoint.com/cplusplus
- https://www.w3schools.com/CPP
- https://www.geeksforgeeks.org/cpp-tutorial
- https://www.javatpoint.com/cpp-tutorial
- https://www.programiz.com/cpp-programming
- https://www.cplusplus.com
- https://www.mygreatlearning.com/