C++ BASIC INPUT/OUTPUT

The C++ standard libraries provide an extensive set of input/output capabilities which we will see in subsequent chapters. This chapter will discuss very basic and most common I/O operations required for C++ programming.

C++ I/O occurs in streams, which are sequences of bytes. If bytes flow from a device like a keyboard, a disk drive, or a network connection etc. to main memory, this is called **input operation** and if bytes flow from main memory to a device like a display screen, a printer, a disk drive, or a network connection, etc, this is called **output operation**.

I/O Library Header Files:

There are following header files important to C++ programs:

Header File	Function and Description
<iostream>	This file defines the **cin, cout, cerr** and **clog** objects, which correspond to the standard input stream, the standard output stream, the un-buffered standard error stream and the buffered standard error stream, respectively.
<iomanip>	This file declares services useful for performing formatted I/O with so-called parameterized stream manipulators, such as **setw** and **setprecision**.
<fstream>	This file declares services for user-controlled file processing. We will discuss about it in detail in File and Stream related chapter.

The standard output stream (cout):

The predefined object **cout** is an instance of **ostream** class. The cout object is said to be "connected to" the standard output device, which usually is the display screen. The **cout** is used in conjunction with the stream insertion operator, which is written as << which are two less than signs as shown in the following example.

```
#include <iostream>

using namespace std;

int main( )
{
   char str[] = "Hello C++";

   cout << "Value of str is : " << str << endl;
}
```

When the above code is compiled and executed, it produces the following result:

```
Value of str is : Hello C++
```

The C++ compiler also determines the data type of variable to be output and selects the appropriate stream insertion operator to display the value. The << operator is overloaded to output data items of built-in types integer, float, double, strings and pointer values.

The insertion operator << may be used more than once in a single statement as shown above and **endl** is used to add a new-line at the end of the line.

The standard input stream (cin):

The predefined object **cin** is an instance of **istream** class. The cin object is said to be attached to the standard input device, which usually is the keyboard. The **cin** is used in conjunction with the stream extraction operator, which is written as >> which are two greater than signs as shown in the following example.

```
#include <iostream>

using namespace std;

int main( )
{
   char name[50];

   cout << "Please enter your name: ";
   cin >> name;
   cout << "Your name is: " << name << endl;

}
```

When the above code is compiled and executed, it will prompt you to enter a name. You enter a value and then hit enter to see the result something as follows:

```
Please enter your name: cplusplus
Your name is: cplusplus
```

The C++ compiler also determines the data type of the entered value and selects the appropriate stream extraction operator to extract the value and store it in the given variables.

The stream extraction operator >> may be used more than once in a single statement. To request more than one datum you can use the following:

```
cin >> name >> age;
```

This will be equivalent to the following two statements:

```
cin >> name;
cin >> age;
```

The standard error stream (cerr):

The predefined object **cerr** is an instance of **ostream** class. The cerr object is said to be attached to the standard error device, which is also a display screen but the object **cerr** is un-buffered and each stream insertion to cerr causes its output to appear immediately.

The **cerr** is also used in conjunction with the stream insertion operator as shown in the following example.

```
#include <iostream>

using namespace std;

int main( )
{
   char str[] = "Unable to read....";

   cerr << "Error message : " << str << endl;
}
```

When the above code is compiled and executed, it produces the following result:

```
Error message : Unable to read....
```

The standard log stream (clog):

The predefined object **clog** is an instance of **ostream** class. The clog object is said to be attached to the standard error device, which is also a display screen but the object **clog** is buffered. This means that each insertion to clog could cause its output to be held in a buffer until the buffer is filled or until the buffer is flushed.

The **clog** is also used in conjunction with the stream insertion operator as shown in the following example.

```
#include <iostream>

using namespace std;

int main( )
{
   char str[] = "Unable to read....";

   clog << "Error message : " << str << endl;
}
```

When the above code is compiled and executed, it produces the following result:

```
Error message : Unable to read....
```

You would not be able to see any difference in cout, cerr and clog with these small examples, but while writing and executing big programs then difference becomes obvious. So this is good practice to display error messages using cerr stream and while displaying other log messages then clog should be used.