C++ streams

Stream classes in C++ are used to input and output operations on files and io devices. These classes have specific features and to handle input and output of the program.

The iostream.h library holds all the stream classes in the C++ programming language.

Let's see the hierarchy and learn about them,

1. ios class − This class is the base class for all stream classes. The streams can be input or output streams.ios class is topmost class in the stream classes hierarchy.

2. istream Class − The istream class handles the input stream in c++ programming language. These input stream objects are used to read and interpret the input as a sequence of characters. The cin handles the input.

3. ostream class − The ostream class handles the output stream in c++ programming language. These output stream objects are used to write data as a sequence of characters on the screen. cout and puts handle the out streams in c++ programming language.

Example
OUT STREAM
COUT

```
#include <iostream>
using namespace std;
int main(){
  cout<<"This output is printed on screen";
}
```

Output

```
This output is printed on screen
```

Example
OUT STREAM
COUT

```
#include <iostream>
using namespace std;
int main(){
  cout<<"This output is printed on screen";
}
```

Output

```
This output is printed on screen
```

PUTS

```
#include <iostream>
using namespace std;
int main(){
  puts("This output is printed using puts");
}
```

Output

```
This output is printed using puts
```

IN STREAM

CIN

```
#include <iostream>
using namespace std;
int main(){
   int no;
   cout<<"Enter a number ";
   cin>>no;
   cout<<"Number entered using cin is "<
```

Output

```
Enter a number 3453
Number entered using cin is 3453
```

gets

```
#include <iostream>
using namespace std;
int main(){
   char ch[10];
   puts("Enter a character array");
   gets(ch);
   puts("The character array entered using gets is : ");
   puts(ch);
}
```

Output

```
Enter a character array
thdgf
The character array entered using gets is :
thdgf
```

Manipulators in C++ with Examples

Manipulators are helping functions that can modify the input/output stream. It does not mean that we change the value of a variable, it only modifies the I/O stream using insertion (<<) and extraction (>>) operators.

Manipulators are special functions that can be included in the I/O statement to alter the format parameters of a stream.
Manipulators are operators that are used to format the data display.
To access manipulators, the file iomanip.h should be included in the program.

For example, if we want to print the hexadecimal value of 100 then we can print it as:

cout<<setbase(16)<<100
Types of Manipulators There are various types of manipulators:

Manipulators without arguments: The most important manipulators defined by the IOStream library are provided below.

endl:
It is defined in ostream. It is used to enter a new line and after entering a new line.

ws: It is defined in istream and is used to ignore the whitespaces in the string sequence.

ends: It is also defined in ostream and it inserts a null character into the output stream.

flush: It is also defined in ostream and it flushes the output stream, i.e. it forces all the output written on the screen or in the file. Without flush, the output would be the same,

Manipulators with Arguments: Some of the manipulators are used with the argument like setw (20), setfill ('*'), and many more. These all are defined in the header file. If we want to use these manipulators then we must include this header file in our program.
Some important manipulators in <iomanip> are:
setw (val): It is used to set the field width in output operations.
setfill (c): It is used to fill the character 'c' on output stream.
setprecision (val): It sets val as the new value for the precision of floating-point values.
setbase(val): It is used to set the numeric base value for numeric values.
setiosflags(flag): It is used to set the format flags specified by parameter mask.
resetiosflags(m): It is used to reset the format flags specified by parameter mask.
Some important manipulators in <ios> are:
showpos: It forces to show a positive sign on positive numbers.
noshowpos: It forces not to write a positive sign on positive numbers.
showbase: It indicates the numeric base of numeric values.
uppercase: It forces uppercase letters for numeric values.
nouppercase: It forces lowercase letters for numeric values.
fixed: It uses decimal notation for floating-point values.
scientific: It uses scientific floating-point notation.
hex: Read and write hexadecimal values for integers and it works same as the setbase(16).
dec: Read and write decimal values for integers i.e. setbase(10).
oct: Read and write octal values for integers i.e. setbase(10).
left: It adjusts output to the left.
right: It adjusts output to the right.
There are two types of manipulators used generally:
1] Parameterized and
2] Non-parameterized

1] Parameterized Manipulators:-

Manipulator	->	Meaning
setw (int n)	->	To set field width to n
setprecision (int p)	->	The precision is fixed to p

setfill (Char f)	->	To set the character to be filled
setiosflags (long l)	->	Format flag is set to l
resetiosflags (long l)	->	Removes the flags indicated by l
Setbase(int b)	->	To set the base of the number to b

setw () is a function in Manipulators in C++:

The setw() function is an output manipulator that inserts whitespace between two variables. You must enter an integer value equal to the needed space.

Syntax:

```
setw ( int n)
As an example,
int a=15;  int b=20;
cout << setw(10) << a << setw(10) << b << endl;
```

setfill() is a function in Manipulators in C++:
It replaces setw(whitespaces)'s with a different character. It's similar to setw() in that it manipulates output, but the only parameter required is a single character.
Syntax:

```
setfill(char ch)
```

Example:

```
int a,b;
a=15;   b=20;
cout<< setfill('*') << endl;
        cout << setw(5) << a << setw(5) << a << endl;
```

setprecision() is a function in Manipulators in C++:
It is an output manipulator that controls the number of digits to display after the decimal for a floating point integer.
Syntax:

```
setprecision (int p)
```

Example:

```
float A = 1.34255;

cout <<fixed<< setprecision(3) << A << endl;
```

setbase() is a function in Manipulators in C++:
The setbase() manipulator is used to change the base of a number to a different value. The following base values are supported by the C++ language:
• hex (Hexadecimal = 16)
• oct (Octal = 8)
• dec (Decimal = 10)

The manipulators hex, oct, and dec can change the basis of input and output numbers.

```
// Example:

#include <iostream>
```

```
#include <iomanip>


using namespace std;
main()
{


int number = 100;


cout << "Hex Value =" << " " << hex << number << endl;


cout << "Octal Value=" << " " << oct << number << endl;


cout << "Setbase Value=" << " " << setbase(8) << number << endl;


cout << "Setbase Value=" << " " << setbase(16) << number << endl;



return 0;

}
```

Output

```
Hex Value = 64
Octal Value= 144
Setbase Value= 144
Setbase Value= 64
```

2] Non-parameterized

Examples are endl, fixed, showpoint and flush.

- endl – Gives a new line
- ends – Adds null character to close an output string
- flush – Flushes the buffer stream
- ws – Omits the leading white spaces present before the first field
- hex, oct, dec – Displays the number in hexadecimal or octal or in decimal format

What is a C++ Exception?

An exception is an unexpected problem that arises during the execution of a program our program terminates suddenly with some errors/issues. Exception occurs during the running of the program (runtime).

Types of C++ Exception

There are two types of exceptions in C++

Synchronous: Exceptions that happen when something goes wrong because of a mistake in the input data or when the program is not equipped to handle the current type of data it's working with, such as dividing a number by zero.
Asynchronous: Exceptions that are beyond the program's control, such as disc failure, keyboard interrupts, etc.
C++ try and catch
C++ provides an inbuilt feature for Exception Handling. It can be done using the following specialized keywords: try, catch, and throw with each having a different purpose.

Syntax of try-catch in C++
```
try {
    // Code that might throw an exception
    throw SomeExceptionType("Error message");
 }
catch( ExceptionName e1 )  {
    // catch block catches the exception that is thrown from try block
 }
```
1. try in C++
The try keyword represents a block of code that may throw an exception placed inside the try block. It's followed by one or more catch blocks. If an exception occurs, try block throws that exception.

2. catch in C++
The catch statement represents a block of code that is executed when a particular exception is thrown from the try block. The code to handle the exception is written inside the catch block.

3. throw in C++
An exception in C++ can be thrown using the throw keyword. When a program encounters a throw statement, then it immediately terminates the current function and starts finding a matching catch block to handle the thrown exception.

Note: Multiple catch statements can be used to catch different type of exceptions thrown by try block.

MemoryAllocation
Reserving or providing space to a variable is called memory allocation. For storing the data, memory allocation can be done in two ways -

Static allocation or compile-time allocation - Static memory allocation means providing space for the variable. The size and data type of the variable is known, and it remains constant throughout the program.

Dynamic allocation or run-time allocation - The allocation in which memory is allocated dynamically. In this type of allocation, the exact size of the variable is not known in advance. Pointers play a major role in dynamic memory allocation.

Why Dynamic memory allocation?

Dynamically we can allocate storage while the program is in a running state, but variables cannot be created "on the fly". Thus, there are two criteria for dynamic memory allocation -

A dynamic space in the memory is needed.
Storing the address to access the variable from the memory
Similarly, we do memory de-allocation for the variables in the memory.

In C++, memory is divided into two parts -

Stack - All the variables that are declared inside any function take memory from the stack.
Heap - It is unused memory in the program that is generally used for dynamic memory allocation.

Dynamic memory allocation using the new operator

To allocate the space dynamically, the operator new is used. It means creating a request for memory allocation on the free store. If memory is available, memory is initialized, and the address of that space is returned to a pointer variable.

Syntax,
Pointer_variable = new data_type;

The pointer_varible is of pointer data_type. The data type can be int, float, string, char, etc.

Delete operator

We delete the allocated space in C++ using the delete operator.

Syntax

delete pointer_variable_name

What are files and streams?

Files are used to store data permanently. A stream is an abstraction that represents a device on which input and output operations are performed. A stream can basically be represented as a source or destination of characters of indefinite length.

C++ file handling provides a mechanism to store output of a program in a file and read from a file on the disk. So far, we have been using <iostream> header file which provide functions cin and cout to take input from console and write output to a console respectively. Now, we introduce one more header file <fstream> which provides data types or classes (ifstream , ofstream , fstream) to read from a file and write to a file.

In this tutorial chapter we will understand the file handling mechanism in C++. We will understand:

What are files and streams

Opening a file
File Opening Modes
Writing to a file
Reading from a file
Closing a file
Error Handling Functionsm
File Pointers & their Manipulation
What are files and streams?
Files are used to store data permanently. A stream is an abstraction that represents a device on which input and output operations are performed. A stream can basically be represented as a source or destination of characters of indefinite length.

C++ file handling provides a mechanism to store output of a program in a file and read from a file on the disk. So far, we have been using <iostream> header file which provide functions cin and cout to take input from console and write output to a console respectively. Now, we introduce one more header file <fstream> which provides data types or classes (ifstream , ofstream , fstream) to read from a file and write to a file.

file handling in C++ programming

Data Type	Description
ofstream	This data type represents the output file stream and is used to create files and to write information to files.
ifstream	This data type represents the input file stream and is used to read information from files.
fstream	This data type 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.

Simple Snippets
HomepageProgramming Languages

Tanmay Sakpal
6 years ago
File Handling in C++

In this tutorial chapter we will understand the file handling mechanism in C++. We will understand:

What are files and streams
Opening a file
File Opening Modes
Writing to a file
Reading from a file

Closing a file
Error Handling Functions
Binary Files
File Pointers & their Manipulation

Opening a File
A file must be opened before you can read from it or write to it. Either the ofstream or fstream object may be used to open a file for writing and ifstream object is used to open a file for reading purpose only.
Following is the standard syntax for open() function, which is a member of fstream, ifstream, and ofstream objects.

```
void open(const char *filename, ios::openmode mode);
```

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.

Write to a File
In order to write to a file we use the insertion operator (<<). The only difference is that you use an ofstream or fstream object instead of the cout object.

Read from a File
In order to read from a file in C++, we use the extraction operator (>>). The only difference is that you use an ifstream or fstream object instead of the cin object.

Closing a File
When C++ program terminates, it automatically closes any opened files and streams but it is always a good practice to close opened files explicitly. Following is the syntax of closing a file in C++ which can be used with objects of fstream, ifstream, and ofstream objects.

```
void close();
```

File Pointer & their Manipulation
The read operation from a file involves get pointer. It points to a specific location in the file and reading starts from that location. Then, the get pointer keeps moving forward which lets us read the entire file. Similarly, we can start writing to a location where put pointer is currently pointing. The get and put are known as file position pointers and these pointers can be manipulated or repositioned to allow random access of the file. The functions which manipulate file pointers are as follows :

Function	Description
seekg( )	Moves the get pointer to a specific location in the file
seekp( )	Moves the put pointer to a specific location in the file
tellg( )	Returns the position of get pointer
tellp( )	Returns the position of put pointer

pointer