STREAMS

In C++ there are number of stream classes for defining various streams related with files and for doing input-output operations. All these classes are defined in the file **iostream.h**. Figure given below shows the hierarchy of these classes.

1. **ios class** is topmost class in the stream classes hierarchy. It is the base class for **istream, ostream,** and **streambuf** class.

2. **istream** and **ostream** serves the base classes for **iostream** class. The class **istream** is used for input and **ostream** for the output.

3. Class **ios** is indirectly inherited to **iostream** class using **istream** and **ostream**. To avoid the duplicity of data and member functions of **ios** class, it is declared as virtual base class when inheriting in **istream** and **ostream** as

```
class istream: virtual public ios
{
};
class ostream: virtual public ios
{
};
```

The **_withassign classes** are provided with extra functionality for the assignment operations that's why **_withassign** classes.

Facilities provided by these stream classes.

1. **The ios class:** The ios class is responsible for providing all input and output facilities to all other stream classes.

2. **The istream class:** This class is responsible for handling input stream. It provides number of function for handling chars, strings and objects such as **get, getline, read, ignore, putback** etc..
   Example:

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

int main()
{
    char x;

    // used to scan a single char
    cin.get(x);
```

```
    cout << x;
}
```

Input:

```
g
```

Output:

```
g
```

The ostream class: This class is responsible for handling output stream. It provides number of function for handling chars, strings and objects such as **write, put** etc..

Example:

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

int main()
{
    char x;

    // used to scan a single char
    cin.get(x);

    // used to put a single char onto the screen.
    cout.put(x);
}
```

1. **Input:**

```
g
```

Output:

```
g
```

What are C++ streams?

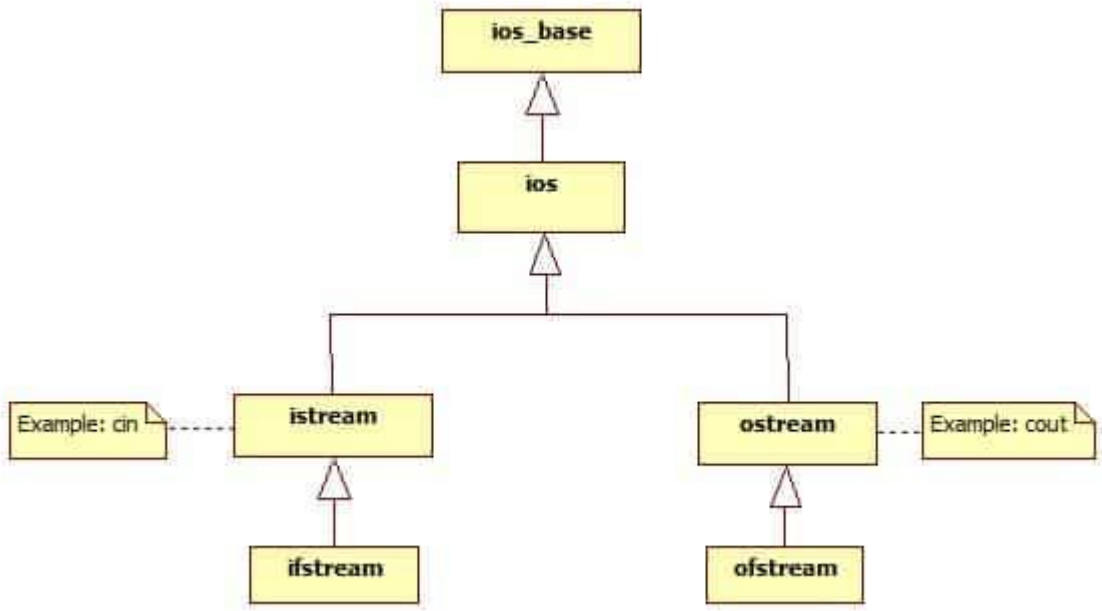
In C++, a stream refers to a sequence of characters that are transferred between the program and input/output (I/O) devices. Stream classes in C++ facilitate input and output operations on files and other I/O devices. These classes have specific features to handle program input and output, making it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

In this blog, we'll be exploring four essential C++ stream classes:

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

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.



Source

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, istream_withassign, and ostream_withassign classes.

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, such as the _withassign classes.

Let's deep dive into the four essential stream classes:

1. The 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++.

```
1    #include <iostream>
2    using namespace std;
3    int main()
4    {
```

```
5       char a;
6       cin.get(a);
7       cout << a;
8       return 0;
9   }
```

2. The ifstream class

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>

3. The 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.

```
1   #include <iostream>
2   using namespace std;
3   intmain()
4   {
5   char u;
6   cin.get(u);
7   cout.put(u);
8   }
```

4. The ofstream class

An ofstream is an output file stream, 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:

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

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

C++ streams are used in a wide variety of applications, from simple console programs to complex software systems. Some common use cases for C++ streams include:

1. Reading and writing to files – C++ streams provide a convenient way to read and write data to and from files. This can be used to create log files, read configuration files, and more.
2. Parsing input – C++ streams can be used to parse input data, such as reading data from a CSV file or parsing command-line arguments.
3. Debugging – C++ streams can be used to output debugging information, such as the value of variables, to the console or a file.
4. Network programming – C++ streams can be used to read and write data over a network connection, such as when creating a client-server application.