**Unit II: Pointer and I/O and File management** :

Pointers in C++, Pointes and Objects, this pointer, virtual and pure virtual functions. **Streams:** Concept of streams, cin and cout objects, C++ stream classes, Unformatted and formatted I/O, manipulators, File stream, C++File stream classes, File management functions.

**Pointers in C++**

The pointer in C++ language is a variable, it is also known as locator or indicator that points to an address of a value.

The symbol of an address is represented by a pointer. In addition to creating and modifying dynamic data structures, they allow programs to emulate call-by-reference. One of the principal applications of pointers is iterating through the components of arrays or other data structures. The pointer variable that refers to the same data type as the variable you're dealing with has the address of that variable set to it (such as an int or string).

Syntax

1. datatype \*var\_name;
2. **int** \*ptr;   // ptr can point to an address which holds int data

## **How to use a pointer?**

1. Establish a pointer variable.
2. employing the unary operator (&), which yields the address of the variable, to assign a pointer to a variable's address.
3. Using the unary operator (\*), which gives the variable's value at the address provided by its argument, one can access the value stored in an address.

Since the data type knows how many bytes the information is held in, we associate it with a reference. The size of the data type to which a pointer points is added when we increment a pointer.



### Advantage of pointer

* 1. Pointer reduces the code and improves the performance, it is used to retrieving strings, trees etc. and used with arrays, structures and functions.
	2. We can return multiple values from function using pointer.
	3. It makes you able to access any memory location in the computer's memory.

## **Declaring a pointer**

The pointer in C++ language can be declared using ∗ (asterisk symbol).

1. **int** ∗   a; //pointer to int
2. **char** ∗  c; //pointer to char

## **Pointer Example**

Let's see the simple example of using pointers printing the address and value.

1. #include <iostream>
2. **using** **namespace** std;
3. **int** main()
4. {
5. **int** number=30;
6. **int** ∗   p;
7. p=&number;//stores the address of number variable
8. cout<<"Address of number variable is:"<<&number<<endl;
9. cout<<"Address of p variable is:"<<p<<endl;
10. cout<<"Value of p variable is:"<<\*p<<endl;
11. **return** 0;
12. }

**Output:**

Address of number variable is:0x7ffccc8724c4

Address of p variable is:0x7ffccc8724c4

Value of p variable is:30

## **C++ Object**

In C++, Object is a real world entity, for example, chair, car, pen, mobile, laptop etc.

In other words, object is an entity that has state and behavior. Here, state means data and behavior means functionality.

Object is a runtime entity, it is created at runtime.

Object is an instance of a class. All the members of the class can be accessed through object.

Let's see an example to create object of student class using s1 as the reference variable.

1. Student s1;  //creating an object of Student

In this example, Student is the type and s1 is the reference variable that refers to the instance of Student class.

## **C++ Class**

In C++, class is a group of similar objects. It is a template from which objects are created. It can have fields, methods, constructors etc.

Let's see an example of C++ class that has three fields only.

1. **class** Student
2. {
3. **public**:
4. **int** id;  //field or data member
5. **float** salary; //field or data member
6. String name;//field or data member
7. }

## **C++ Object and Class Example**

Let's see an example of class that has two fields: id and name. It creates instance of the class, initializes the object and prints the object value.

#include <iostream>

**using** **namespace** std;

**class** Student {

   **public**:

      **int** id;//data member (also instance variable)

      string name;//data member(also instance variable)

};

**int** main() {

    Student s1; //creating an object of Student

    s1.id = 201;

    s1.name = "Sonoo Jaiswal";

    cout<<s1.id<<endl;

    cout<<s1.name<<endl;

    **return** 0;

}

Output:

201

Sonoo Jaiswal

# **C++ this Pointer**

In C++ programming, **this** is a keyword that refers to the current instance of the class. There can be 3 main usage of this keyword in C++.

* It can be used **to pass current object as a parameter to another method.**
* It can be used **to refer current class instance variable.**
* It can be used **to declare indexers.**

## **C++ this Pointer Example**

Let's see the example of this keyword in C++ that refers to the fields of current class.

1. #include <iostream>
2. **using** **namespace** std;
3. **class** Employee {
4. **public**:
5. **int** id; //data member (also instance variable)
6. string name; //data member(also instance variable)
7. **float** salary;
8. Employee(**int** id, string name, **float** salary)
9. {
10. **this**->id = id;
11. **this**->name = name;
12. **this**->salary = salary;
13. }
14. **void** display()
15. {
16. cout<<id<<"  "<<name<<"  "<<salary<<endl;
17. }
18. };
19. **int** main(**void**) {
20. Employee e1 =Employee(101, "Sonoo", 890000); //creating an object of Employee
21. Employee e2=Employee(102, "Nakul", 59000); //creating an object of Employee
22. e1.display();
23. e2.display();
24. **return** 0;
25. }

Output:

101 Sonoo 890000

102 Nakul 59000

[**Virtual Function in C++**](https://www.geeksforgeeks.org/virtual-function-cpp/)A virtual function is a member function which is declared within a base class and is re-defined(Overridden) by a derived class. When you refer to a derived class object using a pointer or a reference to the base class, you can call a virtual function for that object and execute the derived class’s version of the function.

[**Pure Virtual Functions in C++**](https://www.geeksforgeeks.org/pure-virtual-functions-and-abstract-classes/)A pure virtual function (or abstract function) in C++ is a virtual function for which we don’t have an implementation, we only declare it. A pure virtual function is declared by assigning 0 in the declaration.

**Similarities between virtual function and pure virtual function**

1. These are the concepts of Run-time polymorphism.
2. Prototype i.e. Declaration of both the functions remains the same throughout the program.
3. These functions can’t be global or static.

**Difference between virtual function and pure virtual function in C++**

| **Virtual function** | **Pure virtual function** |
| --- | --- |
| A virtual function is a member function of base class which can be redefined by derived class. | A pure virtual function is a member function of base class whose only declaration is provided in base class and should be defined in derived class otherwise derived class also becomes abstract. |
| Classes having virtual functions are not abstract. | Base class containing pure virtual function becomes abstract. |
| Syntax:

|  |
| --- |
| **virtual**<func\_type><func\_name>(){    // code} |

 | Syntax:

|  |
| --- |
| **virtual**<func\_type><func\_name>()    = 0; |

 |
| Definition is given in base class. | No definition is given in base class. |
| Base class having virtual function can be instantiated i.e. its object can be made. | Base class having pure virtual function becomes abstract i.e. it cannot be instantiated. |
| If derived class do not redefine virtual function of base class, then it does not affect compilation. | If derived class do not redefine virtual function of base class, then no compilation error but derived class also becomes abstract just like the base class. |
| All derived class may or may not redefine virtual function of base class. | All derived class must redefine pure virtual function of base class otherwise derived class also becomes abstract just like base class. |

**C++ stream**

In C++ stream refers to the stream of characters that are transferred between the program thread and i/o.

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,



Now, let’s learn about the classes of the iostream library.

**ios class** − This class is the base class for all stream classes. The streams can be input or output streams. This class defines members that are independent of how the templates of the class are defined.

**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.

**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

**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

# **C++ Files and Streams**

In [C++ programming](https://www.javatpoint.com/cpp-tutorial) we are using the **iostream** standard library, it provides **cin** and **cout** methods for reading from input and writing to output respectively.

To read and write from a file we are using the standard C++ library called **fstream**.

 Let us see the data types define in fstream library is:

|  |  |
| --- | --- |
| **Data Type** | **Description** |
| fstream | It is used to create files, write information to files, and read information from files. |
| ifstream | It is used to read information from files. |
| ofstream | It is used to create files and write information to the files. |

## **C++ File Stream example: writing to a file**

Let's see the simple example of writing to a text file **testout.txt** using C++ FileStream programming.

1. #include <iostream>
2. #include <fstream>
3. **using** **namespace** std;
4. **int** main () {
5. ofstream filestream("testout.txt");
6. **if** (filestream.is\_open())
7. {
8. filestream << "Welcome to javaTpoint.\n";
9. filestream << "C++ Tutorial.\n";
10. filestream.close();
11. }
12. **else** cout <<"File opening is fail.";
13. **return** 0;
14. }

**Output:**

## **C++ FileStream example: reading from a file**

Let's see the simple example of reading from a text file **testout.txt** using C++ FileStream programming.

1. #include <iostream>
2. #include <fstream>
3. **using** **namespace** std;
4. **int** main () {
5. string srg;
6. ifstream filestream("testout.txt");
7. **if** (filestream.is\_open())
8. {
9. **while** ( getline (filestream,srg) )
10. {
11. cout << srg <<endl;
12. }
13. filestream.close();
14. }
15. **else** {
16. cout << "File opening is fail."<<endl;
17. }
18. **return** 0;
19. }

**Output:**

Welcome to javaTpoint.

C++ Tutorial.

## **C++ Read and Write Example**

Let's see the simple example of writing the data to a text file **testout.txt** and then reading the data from the file using C++ FileStream programming.

1. #include <fstream>
2. #include <iostream>
3. **using** **namespace** std;
4. **int** main () {
5. **char** input[75];
6. ofstream os;
7. os.open("testout.txt");
8. cout <<"Writing to a text file:" << endl;
9. cout << "Please Enter your name: ";
10. cin.getline(input, 100);
11. os << input << endl;
12. cout << "Please Enter your age: ";
13. cin >> input;
14. cin.ignore();
15. os << input << endl;
16. os.close();
17. ifstream is;
18. string line;
19. is.open("testout.txt");
20. cout << "Reading from a text file:" << endl;
21. **while** (getline (is,line))
22. {
23. cout << line << endl;
24. }
25. is.close();
26. **return** 0;
27. }

**Output:**

Writing to a text file:

 Please Enter your name: Nakul Jain

Please Enter your age: 22

 Reading from a text file: Nakul Jain

 22

# **Formatted I/O in C++**

C++ helps you to format the I/O operations like determining the number of digits to be displayed after the decimal point, specifying number base etc.

**Example:**

* If we want to add + sign as the prefix of out output, we can use the formatting to do so:
* stream.setf(ios::showpos)

If input=100, output will be +100

* If we want to add trailing zeros in out output to be shown when needed using the formatting:
* stream.setf(ios::showpoint)

If input=100.0, output will be 100.000

**Note:** Here, stream is referred to the streams defined in c++ like cin, cout, cerr, clog.

There are two ways to do so:

1. Using the ios class or various ios member functions.
2. Using manipulators(special functions)
3. **Formatting using the ios members:**

The stream has the format flags that control the way of formatting it means Using this setf function, we can set the flags, which allow us to display a value in a particular format. The ios class declares a bitmask enumeration called fmtflags in which the values(showbase, showpoint, oct, hex etc) are defined. These values are used to set or clear the format flags.

Few standard ios class functions are:

1. **width():** The width method is used to set the required field width. The output will be displayed in the given width
2. **precision():** The precision method is used to set the number of the decimal point to a float value
3. **fill():** The fill method is used to set a character to fill in the blank space of a field
4. **setf():** The setf method is used to set various flags for formatting output
5. **unsetf():** The unsetf method is used To remove the flag setting

**Formatting using Manipulators**
The second way you can alter the format parameters of a stream is through the use of special functions called manipulators that can be included in an I/O expression.
The standard manipulators are shown below:

1. [boolalpha:](https://www.geeksforgeeks.org/ios-manipulators-boolalpha-function-in-c/) The boolalpha manipulator of stream manipulators in C++ is used to turn on bool alpha flag
2. **dec:** The dec manipulator of stream manipulators in C++ is used to turn on the dec flag
3. **endl:** The endl manipulator of stream manipulators in C++ is used to Output a newline character.
4. **and:** The and manipulator of stream manipulators in C++ is used to Flush the stream
5. **ends:** The ends manipulator of stream manipulators in C++ is used to Output a null
6. **fixed:** The fixed manipulator of stream manipulators in C++ is used to Turns on the fixed flag
7. **flush:**The flush manipulator of stream manipulators in C++ is used to Flush a stream
8. **hex:** The hex manipulator of stream manipulators in C++ is used to Turns on hex flag
9. **internal**: The internal manipulator of stream manipulators in C++ is used to Turns on internal flag
10. **left**: The left manipulator of stream manipulators in C++ is used to Turns on the left flag
11. [**noboolalpha**:](https://www.geeksforgeeks.org/ios-manipulators-noboolapha-function-in-c/) The noboolalpha manipulator of stream manipulators in C++ is used to Turns off bool alpha flag
12. [**noshowbase**:](https://www.geeksforgeeks.org/ios-manipulators-noboolapha-function-in-c/) The noshowbase manipulator of stream manipulators in C++ is used to Turns off showcase flag
13. [**noshowpoint**](https://www.geeksforgeeks.org/ios-manipulators-noshowpoint-function-in-c/): The noshowpoint manipulator of stream manipulators in C++ is used to Turns off show point flag
14. **noshowpos**: The noshowpos manipulator of stream manipulators in C++ is used to Turns off showpos flag
15. [**noskipws**](https://www.geeksforgeeks.org/ios-manipulators-showpos-function-in-c/): The noskipws manipulator of stream manipulators in C++ is used to Turns off skipws flag
16. [nounitbuf](https://www.geeksforgeeks.org/ios-manipulators-nounitbuf-function-in-c/): The nounitbuf manipulator of stream manipulators in C++ is used to Turns off the unit buff flag
17. [nouppercase](https://www.geeksforgeeks.org/ios-manipulators-nouppercase-function-in-c/): The nouppercase manipulator of stream manipulators in C++ is used to Turns off the uppercase flag
18. **oct**: The oct manipulator of stream manipulators in C++ is used to Turns on oct flag
19. **resetiosflags(fmtflags f)**: The resetiosflags manipulator of stream manipulators in C++ is used to Turns off the flag specified in f
20. **right**: The right manipulator of stream manipulators in C++ is used to Turns on the right flag
21. **scientific**: The scientific manipulator of stream manipulators in C++ is used to Turns on scientific flag
22. **setbase(int base)**: The setbase manipulator of stream manipulators in C++ is used to Set the number base to base
23. **setfill(int ch)**: The setfill manipulator of stream manipulators in C++ is used to Set the fill character to ch
24. **setiosflags(fmtflags f):** The setiosflags manipulator of stream manipulators in C++ is used to Turns on the flag specified in f
25. **setprecision(int p):** The setprecision manipulator of stream manipulators in C++ is used to Set the number of digits of precision
26. **setw(int w):** The setw manipulator of stream manipulators in C++ is used to Set the field width to w
27. [**showbase**:](https://www.geeksforgeeks.org/ios-manipulators-showbase-function-in-c/) The showbase manipulator of stream manipulators in C++ is used to Turns on showbase flag
28. [**showpoint**:](https://www.geeksforgeeks.org/ios-manipulators-noboolapha-function-in-c/) The showpoint manipulator of stream manipulators in C++ is used to Turns on show point flag
29. [showpos](https://www.geeksforgeeks.org/ios-manipulators-showpos-function-in-c/): The showpos manipulator of stream manipulators in C++ is used to Turns on showpos flag
30. [skipws](https://www.geeksforgeeks.org/ios-manipulators-skipws-function-in-c/): The skipws manipulator of stream manipulators in C++ is used to Turns on skipws flag
31. [unitbuf](https://www.geeksforgeeks.org/ios-manipulators-unitbuf-function-in-c/): The unitbuf manipulator of stream manipulators in C++ is used to turn on unitbuf flag
32. [uppercase](https://www.geeksforgeeks.org/ios-manipulators-uppercase-function-in-c/): The uppercase manipulator of stream manipulators in C++ is used to turn on the uppercase flag
33. **ws**: The ws manipulator of stream manipulators in C++ is used to skip leading white space

# **Manipulators in C++**

**Manipulators** are helping functions that can modify the [input/output](https://www.geeksforgeeks.org/basic-input-output-c/) 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:

1. **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 it flushes (i.e. it forces all the output written on the screen or in the file) the output stream.
	* **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. It typically works with std::ostrstream, when the associated output buffer needs to be null-terminated to be processed as a C string.
	* **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, but may not appear in real-time.

**File management Functions:**

File handling is used to store data permanently in a 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.

### Streams in C++ :-

We give input to the executing program and the execution program gives back the output. The sequence of bytes given as input to the executing program and the sequence of bytes that comes as output from the executing program are called stream. In other words, streams are nothing but the flow of data in a sequence.

The input and output operation between the executing program and the devices like keyboard and monitor are known as “console I/O operation”. The input and output operation between the executing program and files are known as “disk I/O operation”.

### Classes for File stream operations :-

The I/O system of C++ contains a set of classes which define the file handling methods. These include ifstream, ofstream and fstream classes. These classes are derived from fstream and from the corresponding iostream class. These classes, designed to manage the disk files, are declared in fstream and therefore we must include this file in any program that uses files.
1. ios:-

* ios stands for input output stream.
* This class is the base class for other classes in this class hierarchy.
* This class contains the necessary facilities that are used by all the other derived classes for input and output operations.

2. istream:-

* istream stands for input stream.
* This class is derived from the class ‘ios’.
* This class handle input stream.
* The extraction operator(>>) is overloaded in this class to handle input streams from files to the program execution.
* This class declares input functions such as get(), getline() and read().

3. ostream:-

* ostream stands for output stream.
* This class is derived from the class ‘ios’.
* This class handle output stream.
* The insertion operator(<<) is overloaded in this class to handle output streams to files from the program execution.
* This class declares output functions such as put() and write().

4. streambuf:-

* This class contains a pointer which points to the buffer which is used to manage the input and output streams.

5. fstreambase:-

* This class provides operations common to the file streams. Serves as a base for fstream, ifstream and ofstream class.
* This class contains open() and close() function.

6. ifstream:-

* This class provides input operations.
* It contains open() function with default input mode.
* Inherits the functions get(), getline(), read(), seekg() and tellg() functions from the istream.

7. ofstream:-

* This class provides output operations.
* It contains open() function with default output mode.
* Inherits the functions put(),  write(), seekp() and tellp() functions from the ostream.

8. fstream:-

* This class provides support for simultaneous input and output operations.
* Inherits all the functions from istream and ostream classes through iostream.

9. filebuf:-

* Its purpose is to set the file buffers to read and write.
* We can also use file buffer member function to determine the length of the file.

In C++, files are mainly dealt by using three classes fstream, ifstream, ofstream available in fstream headerfile.
**ofstream:** Stream class to write on files
**ifstream:** Stream class to read from files
**fstream:** Stream class to both read and write from/to files.