UNIT-II

POINTERS

Pointers are symbolic representations of addresses. They enable programs to simulate call-by-reference as well as to create and manipulate dynamic data structures. Iterating over elements in arrays or other data structures is one of the main use of pointers.

The address of the variable you're working with is assigned to the pointer variable that points to the same data type (such as an int or string).

Syntax:
datatype *var_name;

int *ptr; // ptr can point to an address which holds int data

How to use a pointer?

- Define a pointer variable
- Assigning the address of a variable to a pointer using the unary operator (&) which returns the address of that variable.
- Accessing the value stored in the address using unary operator (*) which returns the value of the variable located at the address specified by its operand.

The reason we associate data type with a pointer is **that it knows how many bytes the data is stored in**. When we increment a pointer, we increase the pointer by the size of the data type to which it points.

POINTERS TO CLASS

Pointer to object in c++ is defined as the pointer that is used for accessing objects. A pointer is a variable that stores the memory address of another variable. The types of the pointer are Null pointer, Void pointer, Wild pointer, and Dangling pointer. An object is defined as the instance of a class.

C++ Pointers and Objects

C++ allows you to have pointers to objects. The pointers pointing to objects are referred to as "object pointers."

C++ Declaration and Use of Object Pointers

Object pointers, like other pointers, are declared by placing in front of the object pointer's name. It takes the following general form:

class-name ∗object-pointer;

where class-name is the name of an already-defined class, and object-pointer is the pointer to an object of this class type. For example, to declare optr as an object pointer of the Sample class type, we shall write

Sample ∗optr;

where Sample is a predefined class. When using an object pointer to access members of a class, the arrow operator (->) is used instead of the dot operator.

C++ "THIS" POINTER

When a member function is called, it is automatically passed an implicit (in-built) argument that is a pointer to the object that invoked the function. This pointer is called "this." That is, if object1 is invoking member function3, then an implicit argument is passed to member function3 that points to object1, i.e., "this" pointer now points to object1. The "this" pointer can be thought of as analogous to the ATM card.

For instance, in a bank, there are many accounts. The account holders can withdraw money or view their bank statements through automatic teller machines. Now, these ATMs can withdraw money from any account in the bank, but which account are they supposed to work on? This is resolved by the ATM card, which gives the identification of the user and his accounts, from which the amount is withdrawn.

Similarly, the "this" pointer is the ATM card for objects, which identifies the currently-calling object. The "this" pointer stores the address of the currently-called object.

VIRTUAL AND PURE VIRTUAL FUNCTIONS IN C++

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.

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.

STREAMS

CONCEPT OF STREAMS IN C++

C++ stream is a flow of data into or out of a program, such as the data written to cout or read from cin.

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.

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

A C++ stream is a flow of data into or out of a program, such as the data written to cout or read from cin.

cin and cout objects in c++

cin is an object of the input stream and is used to take input from input streams like files, console, etc. cout is an object of the output stream that is used to show output. Basically, cin is an input statement while cout is an output statement.

The cin object in C++ is an object of class iostream. It is used **to accept the input from the standard input device i.e. keyboard**. The cout object in C++ is an object of class iostream. It is used **to display output in screen i.e monitor**.

They also use different operators. cin uses the insertion operator(>>) while cout uses the extraction operator(<<).

```
#include<iostream>
int main() {
  int age;
  cin >> age;
  cout << age;
  return 0;
}
```

iostream: iostream stands for standard input-output stream. This header file contains definitions of objects like cin, cout,etc.,

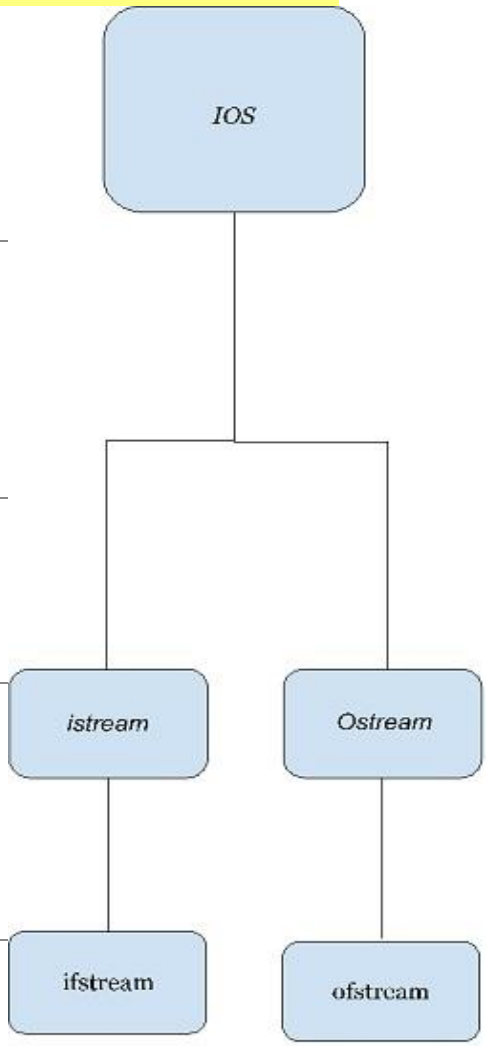
C++ STREAM CLASSES

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.

For this class we are currently interested in FOUR different classes:

- **istream** is a general purpose input stream. cin is an example of an istream.
- **ostream** is a general purpose output stream. cout and cerr are both examples of ostreams.
- **ifstream** is an input file stream. It is a special kind of an istream that reads in data from a data file.
- **ofstream** is an output file stream. It is a special kind of ostream that writes data out to a data file.



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

C++ program

The character array entered using gets is :

C++ program

FORMATTED AND UNFORMATTED I/O OPERATIONS IN C++

C++ provides both the formatted and unformatted IO functions. In formatted or high-level IO, bytes are grouped and converted to types such as int , double , string or user-defined types. In unformatted or low-level IO, bytes are treated as raw bytes and unconverted.

Formatted I/O the data is formatted or transformed.

Unformatted I/O transfers data in its raw form or binary representation without any conversions.

Unformatted I/O is the most basic form of I/O and it is simple, efficient and compact.

getchar() and putchar()

getchar() function will read a single character from the standard input. The return value of getchar() is the first character in the standard input.The input is read until the Enter key is pressed, but only the first character in the input will be returned.

putchar() function will print a single character on standard output. The character to be printed is passed to putchar() function as an argument. The return value of putchar() is the character which was written to the output.

getchar() and putchar() functions are part of the standard C library header stdio.h

Example

```
#include <stdio.h>
 int main()
{
char ch;
printf("Input some character and finish by pressing the Enter key.\n");
ch = getchar();
printf("The input character is ");
putchar(ch);
return 0;
}
```

getch(), getche() and putch()

These functions are similar to getchar() and putchar(), but they come from the library header conio.h. The header file conio.h is not a standard C library header and it is not supported by compilers that target Unix. However it is supported in DOS like environments.

getch() reads a single character from the standard input. The input character is not displayed (echoed) on the screen. Unlike the getchar() function, getch() returns when the first character is entered and does not wait for the Enter key to be pressed.

getche() is same as getch() except that it echoes the input character. putch() writes a character that is passed as an argument to the console.

Formatted input and output statements

printf()

This function is used to print text as well as value of the variables on the standard output device (monitor), printf is very basic library function in c language that is declared in stdio.h header file.

Syntax :

printf(“message”);

printf(“message + format-specifier”,variable-list);

First printf() style printf the simple text on the monitor, while second printf() prints the message with values of the variable list.

scanf()

This function is used to get (input) value from the keyboard. We pass format specifiers, in which format we want to take input.

Syntax:

scanf(“format-specifier”, &var-name);

scanf(“fromat-specifier-list”, &var-name-list);

First type of scanf() takes the single value for the variable and second type of scanf() will take the multiple values for the variable list.

MANIPULATORS

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 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. **Examples:**

FILE STREAM

C++ Files and Streams

In C++ programming 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++ FileStream example: writing to a file

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

EX:

```
int main () {
  ofstream filestream("testout.txt");
  if (filestream.is_open())
  {
    filestream << "Welcome to javaTpoint.\n";
    filestream << "C++ Tutorial.\n";
    filestream.close();
  }
  else cout <<"File opening is fail.";
}
```

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.

Example:

```
int main () {
 string srg;
 ifstream filestream("testout.txt");
 if (filestream.is_open())
 {
  while ( getline (filestream,srg) )
  {
   cout << srg <<endl;
  }
  filestream.close();
 }
 else {
    cout << "File opening is fail."<<endl;
  }
```

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.

```
#include <fstream>
#include <iostream>
using namespace std;
int main () {
  char input[75];
  ofstream os;
  os.open("testout.txt");
  cout <<"Writing to a text file:" << endl;
  cout << "Please Enter your name: ";
  cin.getline(input, 100);
  os << input << endl;
  cout << "Please Enter your age: ";
  cin >> input;
  cin.ignore();
  os << input << endl;
```

```
  os.close();
  ifstream is;
  string line;
  is.open("testout.txt");
  cout << "Reading from a text file:" << endl;
  while (getline (is,line))
  {
  cout << line << endl;
  }
  is.close();
  return 0;
}
```

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
```

C++ FILE STREAM CLASS

A file stream can be described by using the ifstream, ofstream, and fstream classes that are contained in the header file fstream.

The class to be used depends upon the purpose whether the write data or read data operation is to be executed on the file.

By Using iostream standard library, for reading from input and writing to output it provides cin and cout methods.

Operations in File Handling:

- Creating a file: open()
- Reading data: read()
- Writing new data: write()
- Closing a file: close()

To read and write from a file we are using the standard C++ library known as fstream

Fstream: It is used to create files, writes information to files, and read information from files.

This class provides maintenance for both input and output operations on the files at the same time. It inherits all the member functions of the istream and ostream classes via iostream class. when a file is opened in default mode, it also contains the open() function.

Ifstream: It is used to read information from files. This class supports input operations on the files

It inherits the features get( ), getline( ), read( ), seekg( ) and tellg( ) from istream class. When the file is opened in default mode, it contains open( ) function.

FILE MANAGEMENT FUNCTIONS

File Handling In C++

Files are used to store data in a storage device permanently. File handling provides a mechanism to store the output of a program in a file and to perform various operations on it.

A stream is an abstraction that represents a device on which operations of input and output are performed. A stream can be represented as a source or destination of characters of indefinite length depending on its usage.

In C++ we have a set of file handling methods. These include ifstream, ofstream, and fstream. These classes are derived from fstrembase and from the corresponding iostream class. These classes, designed to manage the disk files, are declared in fstream and therefore we must include fstream and therefore we must include this file in any program that uses files.

In C++, files are mainly dealt by using three classes fstream, ifstream, ofstream.

- ofstream: This Stream class signifies the output file stream and is applied to create files for writing information to files
- ifstream: This Stream class signifies the input file stream and is applied for reading information from files
- fstream: This Stream class can be used for both read and write from/to files.

All the above three classes are derived from fstreambase and from the corresponding iostream class and they are designed specifically to manage disk files.
C++ provides us with the following operations in File Handling:

- Creating a file: open()
- Reading data: read()
- Writing new data: write()
- Closing a file: close()