Paper VI: Object Oriented Programming Using C++

Unit I: Concept of OOP

Difference between OOP and POP

POP	OOP
POP means procedure oriented programming. Language are C,PASCAL etc.	OOP means object oriented programming. Language are C++, Java etc.
In POP program divide into small parts call function	In POP program divide into parts called object.
In POP, important is not given to data but to function as well as sequence of actions (procedure) to be done.	In OOP, important is given to data but not to function or procedure.
POP is top down approach.	OOP is bottom up approach.
POP don't have any access specifier.	OOP have access specifier i.e. private, public and protected
In OOP, overloading is not possible.	In OOP, overloading is possible in the form of function and operator overloading.

History of C++

C++ is an object oriented programming language. It was developed by Bjarne Strousrup at AT & T Bell lab in USA in early 1980s. It combines the object oriented features of language called simula 67 and power of C, the result was C++. Therefore, C++ is an extension of C with major addition of the class constructed feature of simula 67. Bjarne Straustrup initially called new language C with classes. However in letter 1993 the name was changed to C++ because of some new added feature.

C++ is a super set of C. Almost all C program are also C++ program. The most important facilities that C++ adds on to C are

classes, inheritance, functions overloading and operator overloading. The object oriented feature in C++ allow to programme to build large programmes with clearly efficiency and easy for maintenance.

Basic Concept of OOP(Object Oriented Program)

1) Objects –

In oop the programs consists of different objects. Objects are basic run time entities. In object oriented system they may represent a person, place, bank account or any item. Programming problem is analysing in terms of object and nature communication between them. When programs executed the objects interact by sending message to another object.

For example, consider library system which has different objects as book and member. Member object can pass message to book object to request for a book. Each object contains data and function to manipulate data. Different system can have different objects.

- Item in inventory system.
- Customer in Bank
- Book in library System
- GUI element like menu, icon
- Employee in Payroll System.

Book
Book Code Book Name Book Author
Accept() Issue()

Class –

Class is a user defined data type, which consist of data member and member function. Classes are declared by using the keyword class, followed by class name and objects are instance of class.

Once a class has been define, we can create no. of object belonging to that class, each object is associated with the data of type class, which are created.

```
class Book
{
        private:
                int code;
                char author;
                char name;
        public:
                accept();
                issue();
};
```

FEATURE OF OBJECT ORIENTED PROGRAMMING

1) Data Encapsulation:

The binding of data and function into single unit is called data encapsulation. Encapsulation is the most striking feature of the class. The data can't be accessible to the outside the world and only there function which is present in the class can access it. This function provides the interface between the objects data and program. This insulation of the data from direct access by the function(program) is called as data hiding. The internal data of an object is hidden from rest of program. To hide a data we have to put it in class and make it private.

Syntax –

```
        class class_name
        {
```

```
private:
variable declaration;
function declaration;
public:
variable declaration;
function declared;
};
```

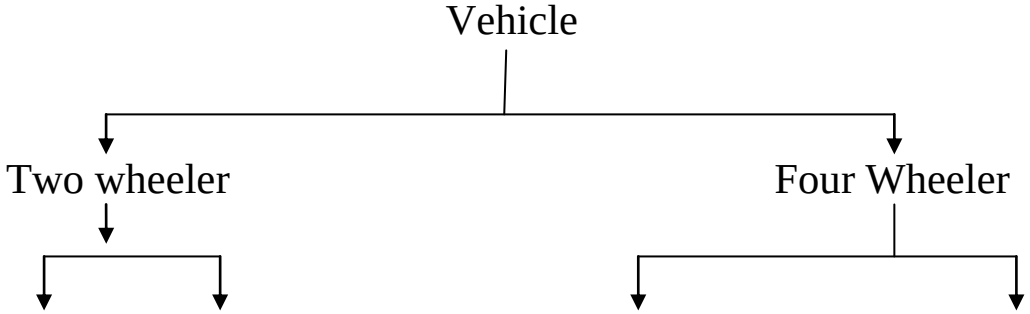
2) Data Abstraction –

Abstraction refers to act of representing essential features without including the background detail or explanation. The process of define data type is called an abstract data type. The definition of an abstract data type involves specifying the internal representation of the abstract data type as well as the function to be used to manipulate the abstract data type.

3) Inheritance –

Inheritance is the ability to derive new class from an existing one. The child class is subset of the parents. It supports the concept of hierarchical classification.

From the figure given below (vehicle) is the base i.e. parent class, which has its own property. Two classes derived from parent are Two Wheeler and Four Vehicle. From that again four derived classes namely are class Bajaj, Hero Honda and Santro, Indica. The important thing behind this is that each derived class shared common characteristics with the class from which it is derived.



Bajaj Hero Honda Santro Indica

4) Polymorphism –

Object Oriented Programming language (like C++) support polymorphism which implies one interface multiple forms (methods.) Poly means “many” and morph means “form”

Multi Coloured Ball Pen
Write

From the above figure consider a multi coloured ball pen which will have different coloured refills and you can select the desired colour for writing.

So depending on what argument is passed, the right function will get called. Thus you have same function name write() which behaves differently depending on the types of argument (refill), is known as function over loading.

The behaviour depends upon the types of data used in the operation e.g. Consider the operation of addition of two numbers. The operation will generate its sum. If the operation on string then, the operation will produce 3rd string by concatenation. This process is known as operator overloading.

Reusability –

The term refers to the ability for multiple programmers to use same written and debugged existing data. This is time, effort and money saving concept.

Application of OOP

Application of OOP provides gain in many areas. The most popular application of oop is in the area of user interface designing such as windows (Operating System).

Real business systems are more complex and contain many more objects with complicated attributes and method. Oop is useful in this type of application.

1) Real time system 2) Simulation 3) OO data Base

4) A.I. (Artificial Intelligence) and expert system

5) Neural Network 6) CAD (Computer Added Design) and CAM (Computer Added Manufacturer)

Structure of C++ Program

Comment line Link section Global/Symbolic Declaration,
Class Declaration
Member function Definition
Main() Function Definition

main() Function – A C++ program must have one and only one function with the name main. The main() function should have return type of value either int or void. Since, main is entry point to the program from calling process. Object of class is declared inside main function.

Member Function –

Member function is used manipulate data member. The name of function should indicate what task or purpose that function is to be performing. Function name can be composed of letter, digits and underscore. Function must have a pair of curly braces { }. This shows beginning and end of function. The statement written in function is terminated with a semi colon(;), this called the statement terminator.

Class of Declaration –

Class is user between data type. It is almost like a structure in a C program. But there is only a difference between structure and class, in a class data member & member function are declared but in structure only data member are declared. Declared function data

variable collectively called class member. They are grouped under three section namely private, public and protected.

Comment line, link section and Global/Symbolic Declaration – This section consists of comment, link section, definition section (global/symbolic declaration), where comment gives abstract of program, Link section gives instruction to the function to link library files and Definition section declare all symbolic constants and global declaration.

Input & Output Statement

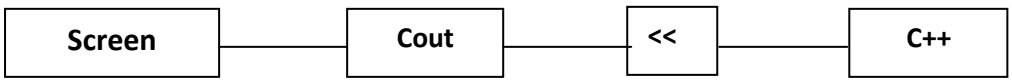
Output Operator

This statement,

cout<< "C++ is better than C";

the string in quotation mark to be displayed on the screen. This statement introduce to new C++ feature, cout and <<. The identifier cout is predefine object, that represents the standard output in C++. Here the standard output represent the screen. The operator << is called the insertion or put operator, it inserts (sends) the content of the variable to its left.

e.g. 1) cout << "x= "; 2) cout<< "Enter two number";



Input Operator

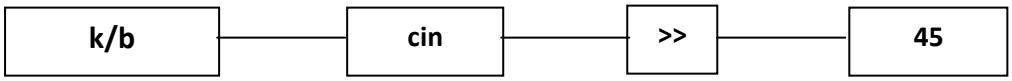
The statement

cin >> number1;

is an input statement & causes the program to wait user to type (enter) a number. The entered is a placed in the variable number1. The cin

predefine object in C++ that corresponds to the standard input string. The operator >> is known as extraction or get from operator. It extracts (takes) value from the key board and assigns it to the scanf operation.

e.g. 1) cin >> i;
2) cin >> a >> b;



Managing output with manipulators

The headers file <iomanip.h> provides set of function called manipulator, which can be used to manipulate the o/p format. Manipulators are special function that changes certain characteristic of the o/p.

1) End1 – The endl is an o/p manipulator which ends the line (to generate line feed character). The endl may be used several times in a C++ statement. It has same effect as using the new line ("\n").

e.g.- cout << "a" = "<< a << endl;
cout << "b" = "<< b<< endl;"

o/p - a = 20;
b = 30;

2) setbase() : The setbase() manipulate is used to convert the base of one numeric value into another base. Following are the common base converter in C++.

Dec = Decimal base(10)
Oct = Octal base (8)
Hexa = hexadecimal base (16)

This setbase() manipulator is also used to define the base of numeric value of the variable. The prototypes of the set base() manipulator is define in the iomanip.h header file.

e.g. #include<iostream.h>

```
  #include<iomanip.h>
main()
 {
  int value;
  cout<<"Enter No."<<endl;
  cin>>value;
  cout<<"Decimal Base= "<<setbase(10)<<value<<endl;
  cout<<"Octal Base= "<<setbase(8)<<value<<endl;
  cout<<"Hexa Base= "<<setbase(16)<<value<<endl;
}
```

3) setw(): The setw() stands for set width. The setw() manipulator is used to set width of a variable (Specify minimum no. of character position) on the o/p field a variable will consume.

Syntax – setw(int w);
The default field width is zero.

```
#include<iostream.h>
#include<iomanip.h>
main()
 {
  int a,b;
  a=200;
  b=300;
  cout<<a<<b<<endl;
  cout<<setw(5)<<a<<setw(5)<<b<<endl;
  cout<<setw(6)<<a<<setw(6)<<b<<endl;
  cout<<setw(7)<<a<<setw(7)<<b<<endl;
   }
```

4) setfill(); - The setfill() manipulator is used to specify a different character to fill the unused field width of the value.
Syntax –

e.g. – setfill(char f);
The default fill character is space.

```
#include<iostream.h>
#include<iomanip.h>
main()
 {
  int a,b;
  a=200;
  b=300;
  cout<<setfill('*');
  cout<<a<<b<<endl;
  cout<<setw(5)<<a<<setw(5)<<b<<endl;
  cout<<setw(6)<<a<<setw(6)<<b<<endl;
  cout<<setw(7)<<a<<setw(7)<<b<<endl;
   }
```

5) setprecision();

setprecision () is used to control the no. of digits of an o/p string, display the floating point value.
Syntax – setprecision(int p)
The default precision is 6

```
#include<iostream.h>
#include<iomanip.h>
main()
 {
  float a=5,b=6,c=a/b;
  cout<<c<<endl;
  cout<<setprecision(1)<<c<<endl;
  cout<<setprecision(2)<<c<<endl;
  cout<<setprecision(3)<<c<<endl;
 }
```

6) flush()–

The flush member function is used to cause the string associated with the o/p to be completely empted. This function takes no I/P parameter whenever it is invoked.

```
#include<iostream.h>
#include<iomanip.h>
main()
{
      cout<<" My name is Computer";
      cout.flush();
}
```

****

Program for practice:

```
//Class that combine both struct and function
      #include<stdio.h>
            class stud
            {  private:
                  int rollno,sub1,sub2;
                  float fee;
                       public:
                  void getdata()
                   {
                         printf("\nEnter rollno of student ");
                         scanf("%d",&rollno);
                         printf("\nEnter marks of student : ");
                         scanf("%d%d",&sub1,&sub2);
                         printf("\nEnter fee of student  ");
                         scanf("\n%f",&fee);
                   }
                   void showdata()
                   {
                               printf("\n%d",rollno);
                               printf("\n%d%d",sub1,sub2);
```

```
                printf("\n%f",fee);
        }
};
        main()
        {       stud s1;
                s1.getdata();
                s1.showdata();
        }



//sum of number
#include<iostream.h>
class sum
        {
        int num;
        public:
         void getdata()
                {
                 cout<<"\n\t Enter the number";
                 cin>>num;
                }
         void showdata()
                {  int i=1,result=0;
                 while(i<=num)
                        {
                         result=result+i;
                         i++;
                        }
                cout<<"\n\t Addition of number is = "<<result;
                }
 };
 main()
        {
        sum s;
```

```
                    s.getdata();
                    s.showdata();
                   }
```

```
//example of reusability
#include<iostream.h>
        class add
            {
             public:
                int a,b;
                void getdata()
                    {
                        cout<<"\n\t Enter value for a and b";
                        cin>>a>>b;
                    }
                void display()
                    {
                        cout<<"\t\t"<<a+b;
                    }
              };


//main program of reusability
    #include<iostream.h>
    #include<c:\tcwin45\c++\use.cpp>
        class sub : public add
            {
                public:
                    void getinfo()
                        {
                            add::getdata();
                        }
```

```
                void display1()
                        {
                                cout<<"\n\t\t"<<a-b;
                        }
};
void main()
        {
                sub a;
                a.getinfo();
                a.display();
                a.getinfo();
                a.display1();
        }
```