C++

Page No.:
Date:
YOUVA

Important Questions from previous year

Subject → OOPS through C++ (303)

15 Questions & Answers

Saurabh Kumar
BCA - 3rd Semester
Roll No - 61
L.S. College, Muzaffarpur.

INDEX

Page No.: 1
Date: YOUVA

Q1: What is C++ ? Write down the Features of OOPS. V.V.I

Ans:- C++ is a Middle level programming language based on structure and object oriented programming concept. C++ is define by Bjarne Stroustrup In 1979 at Bell Lab. U.S.A. The C++ is the extension or superset of C language. The C++ code save with .cpp extension. C++ is the superset of C. So, C++ use all C component and function all valid statement for C is also valid for C++. The C++ use compliers to run code like Turbo C++, devcpp or various IDE. The C++ language can be used to develop operating systems, browsers, games and many types of softwares. In short we can say that C++ is a General Purpose, case-sensitive, free-form programming language that supports object-oriented procedural and Generic programming.

* Program to print Hello world In C++

=> #include <iostream.h>
#include <conio.h>
void main()
{
cout<<"Hello. World";
getch();
}

* Features of OOPS

=> The Object oriented Programming is a type of programming method provide several features to enhance programming ability. The most Important features of OOPS are as:-

(i) Data Encapsulation
(ii) Data Abstraction
(iii) Data Hiding
(iv) Overloading
(v) Inheritance
(vi) Object
(vii) Class
(viii) Polymorphism

(i) Data Encapsulation
=> Encapsulation create container In which we can specify collection of data member & member functions. It is Implemented by using class.

(iii) Data Abstraction

=> Abstraction allow to extract encapsulated data and it also provide facility to create object. In C++ the Abstraction perform by using object or Inheritance.

(iii) Data Hiding

=> Hiding set rules for accessing encapsulated data. In C++ the access specifier like public private & protected is used for Data Hiding.

(iv) Overloading

=> Overloading Implement function & operator overloading. Function overloading allow to create multiply function with same name but different argument where operator overloading provide feature to use exiting operator for object we can change the properties or add new properties to the existing operator.

(v) Inheritance

=> Inheritance Implement reusability concept. It is used to achieve runtime polymorphism. Using Inheritance object of one class can aquire the properties of another class.

(vi) Object

=> The object is the instance of class. It is identifiable entity with some characteristics and behaviour. It is the basic unit of object oriented Program and the object take up space in the memory.

(vii) Class

=> The class is the user defined data type which hold its own data member and member function. A class is the blueprint of an object. It does not take any space in the memory. The memb of class can be accessed by using object.

(viii) Polymorphism.

=> The word Polymorphism means having many forms. When one task is performed by many ways that is known as Polymorp In C++ we use Function overloading & Function overrid or virtual and Pure virtual to achieve Polymorphism.

Q. V.V.I What is class & object? Write down a C++ program to enter Student name, roll no and marks. Display Students name roll no and marks using class & object.

⇒ Class :-
In object oriented programming class create container and implement data encapsulation feature. The class create container in which we can store data member & member function. We can say that the class is a user defined data type. Class use "class" keyword for declaration followed by the classname. and must be declare starting of the program and before main function. The class must be terminate with the semicolon(;). The class member function and data member is only accessible by using inheritance of object.

Syntax
```
class <classname>
{
    Data member;
    member function;
};
```

Ex:-

```
class Student
{
private:
    int a,b;
public:
    void Print()
    {
    cout<<"Hello";
    }
};
```

⇒ Object:

The object is the Instance of a class where using object Implement data abstraction feature. Object create memory block and assign its properties according to class. Using object we can access class member by using (.) operator. Where object only access the public member of the class. The object create seperate memory block for each Instance and Initialize with Garbage value. The object take space In the memory and it is created at runtime. and it is runtime enti

Page No.: 7
Date: YOUVA

Syntax

```
<classname> <objectname>;
```

ex:- Student S;
Circle C;

* Question Program

=>
```
#include<iostream.h>
#include<conio.h>
class Student
{
    private:
        char name[20];
        int roll, marks;
    public:
        void input()
        {
        cout<<"enter name";
        cin>>name;
        cout<<" enter roll no";
        cin>>roll;
        cout<<" enter marks";
        cin>>marks;
        }
        void show()
        {
        cout<<" Student name ="
            <<name<<endl;
        cout<<" Roll no. ="<<roll
            <<endl;
        cout<<" marks ="<<marks;
        }
};
void main()
{
Student S;
clrscr();
S.input();
S.show();
getch();
}
```

(3.) What is object oriented programming ? Discuss the features & application of C++

ans:- object oriented Programming is a type of programming method Provide several features to enhance Programming ability to Implement bottom-up approach on object. OOPS programming method based on class & object. OOPS Provide many concept such as Inheritance, polymorphism data binding, abstraction overloading etc. The object oriented programming is the most popular Programming paradism. The main aim of the object oriented programming is to bind together the data and the function that operate on them so, that no other part of the code can access this data except that function. These days various programming language Implemen object oriented programming concept like C++, Java, C# Python etc.

=

⇒ Features of C++

The main features of C++ are as:-

(i) Object oriented Programming

⇒ One of the most important feature because of which C++ got famous is that the C++ implement OOPS concept. The OOPS feature like class object, Inheritance, Polymorphism abstraction, overloading etc. are implemented In C++.

(ii) Simple

⇒ C++ is the one of the most simple language when it comes to programming. It is easy to understand and learn.

(iii) Rich library

⇒ C++ library is full of inbuilt function that save a huge amount of time in the software development process. It also speed up the software development by using inbuilt functions.

(iv) Case Sensitive

⇒ The C++ is case sensitive that means lowercase and uppercase character written in code will have different meaning and will be treated differently

(v) mid level language

⇒ The c++ is collection of special feature of low level and high level language. It can be use to develop application based on the required level of programming language that is high or low.

(vi)

* application of c++.

⇒ The c++ can be use to develop high as well as low level software. The main application of c++ are as:-

(i) It can be used in Game development (xbox, PlayStation)

(ii) The c++ can be used to develop operating System (windows, macos)

(iii) The c++ is used to develop database softwares like MySQL

(iv) It is used to develop browser like Mozila Firefox

(v) It can be used to develop GUI software like adobe, AMP media Player

(vi) many Application of google is develop by using c++ language.

V.V.I

(5 question → from any type)

4. What is Inheritance ? Discuss the types of Inheritance with suitable example.

⇒ The Inheritance Implement reusability feature so, using Inheritance we can Inherit one class properties by another class. The Inheritance provide another form of data abstraction. The Inheritance stabish a relation-ship between two or more classes. In simple word we can say that Inheritance is a machanism of designing a new class from the old class. In such a way that the new class aquire or Inherit all the member (data member & member function) of the old class. The new class has properties & behaviour of the existing class as well as its own unique properties & behaviours.

The class that is Inherit the new class is called base, Super or parent class and the class which is Inherited by the old class is called child, sub or derived class.

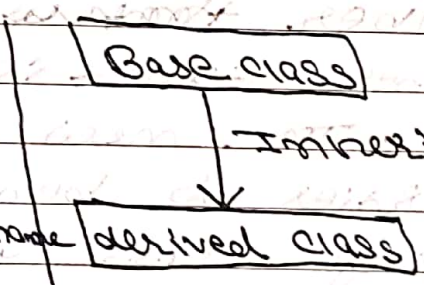
Types of Inheritance.

⇒ In C++ Inheritance is divided into 5 categories.

(i) Single Inheritance
(ii) multiple Inheritance
(iii) multilevel Inheritance
(iv) Hierarchical Inheritance
(v) Hybrid Inheritance.

(i) Single Inheritance

⇒ It is a type of Inheritance in a derived class is inherit from a single base class. The derived class aquire all the properties and behaviour of the single base class and it has also its own properties and behaviour.



Syntax:

```
class base
{
};
class derivedClassname :
    mode BaseClassname
{
};
```

Example:-

```
#include<iostream.h>
#include<conio.h>
class base
{
    public:
        int x;
        get ()
        {
        cout<<" enter any
            value";
        cin>>x;
        }
};
```

```
class derive : public base
{
public:
    int y;
void input()
{
cout<<"enter another
    number";
cin>>y;
}
void product()
{
cout<<" product ="
    <<x*y;
}
};
```

```
void main()
{
derive a;
clrscr();
a.get();
a.input();
a.product();
getch();
}
```

O/P -> enter any number
5 ↵
enter another num
10 ↵
product = 50

(iii) multiple Inheritance

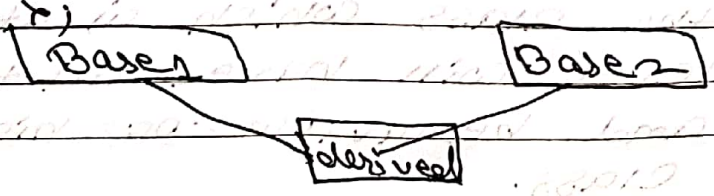
=) It is a type of inheritance in which a derived class is inherited from more than one class. The derived class acquire all the properties and behaviour of all the parent class. And it has also its own properties and behaviour.

Syntax :-

```
class base1
{
};
```

```
class base2
{
};
```

class derivedclassname : mode baseclass1, mode baseclass2 ;



Example:

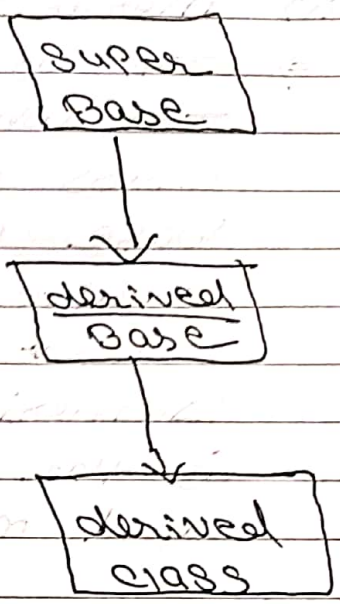
```
#include<iostream.h>
#include<conio.h>
class A
{
  public:
    int x;
    void get1()
    {
    cout<<"enter no";
    cin>>x;
    }
};
class B
  {
    public:
      int y;
    void get2()
    {
     cout<<"enter
      another number";
      cin>>y;
    }
};
class C:public A,
        public B
{
  public
  void sum()
  {
  cout<<"sum="
    <<x+y;
  }
};
void main()
{
  C obj;
  obj.get1();
  obj.get2();
  obj.sum();
  getch();
}
```

(iii) multilevel Inheritance

⇒ It is a type of Inheritance in which one class is inherited by another class and it also inherit the other class. So that the class which is at the lower level will have all the properties and behaviour of higher level class.

Syntax

```
class A
{
};
class B: mode A
{
};
class C: mode B
{
};
```



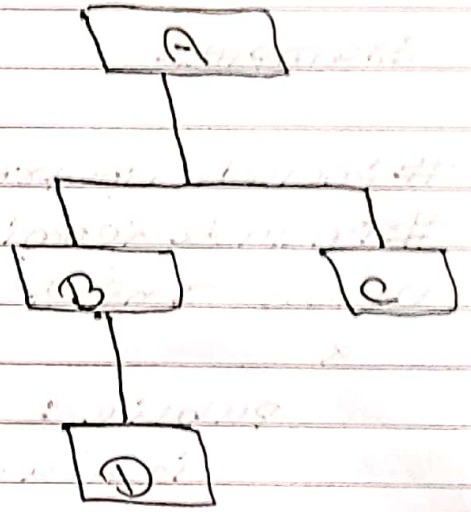
Example:-

```
#include<iostream.h>
#include<conio.h>
class Input()
{
    public:
        int a,b,c;
    void get()
    {
    cout<<"\n enter two
          number";
    cin>>a>>b;
    }
};
class Process: public Input
{
public:
    void sum()
    {
    c=a+b;
    }
};
class Show: public
{          Process
public:
    void output()
    {
    cout<<"\n Sum="
        <<c;
    }
};
void main()
{
  Show S;
  S.get();
  S.sum();
  S.output();
  getch();
}
```

(iv) Hierarchical Inheritance

⇒ It is a type of inheritance in which multiple derived class is inherited from single base class and all derived class acquire the properties of base class.

It is the combination of single and multilevel inheritance.



Syntax:-

```
class A
{
  ;
};
class B: public A
{
};
class C: public A;
{
};
class D: public B;
{
};
```

Example:-

```
#include<iostream.h>
#include<conio.h>
class input
{
  public:
    int a,b;
    void get();
    {
      cout<<"enter two number"
      cin>>a>>b;
    }
};
class sum: publi
    input
{
  public:
```

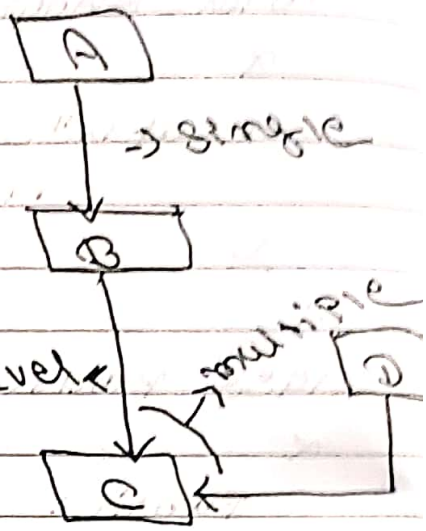
```
void Show2();
{
cout<<"Sum = "<<a+b
    <<endl;
}
};
class Product:Public Input
{
public:
    void Show2()
    {
    cout<<"Product = "
    <<a*b<<endl;
    }
};
class Sub:Public Input
{
public:
    void Show3()
    {
    cout<<"Substract = "
    <<a-b;
    }
};
void main()
{
Sum S;
Product P;
Sub C;
S.get();
P.get();
C.get();
S.Show2();
P.Show2();
C.Show3();
getch();
}
```

(v) Hybrid Inheritance

⇒ According to the user requirement the various types of Inheritance is combined in a single program then such Inheritance is called Hybrid Inheritance. It is the combination of Single Inheritance multiple Inheritance and multilevel Inheritance.

Syntax:-

```
class A
  {
  };
class B: public A
  {
  };
class D:
  {
  };
class C: public B,
         public D
  {
  };
```



example:-

```
#include<iostream.h>
#include<conio.h>
class A
  {
    public:
      int x;
  };
class B: public A
  {
    public:
      void get1()
        {
        cout<<" enter
          number";
        cin>> x;
```

```
class D
  {
  public:
    int y;
    void get2();
      {
      cout<<"enter
        another num
      cin>>y;
      }
  };
class C: public B
         public D.
  {
  public:
    void sum()
      {
      cout<<" Sum
        <<x+y;
      }
  };
```

```
void main()
{
    C obj;
    clrscr();
    obj.get1();
    obj.get2();
    obj.sum();
    getch();
}
```

5. What is constructor? Discuss the different types of constructor with suitable example.

=> Constructor is the special member function for class which must be define publically. The Intiallization of the class member is done with the help of constructor. Constructor use same name as class name for declaration and it do not have any return type even void. In the class we can define multipe constructor (except default constructor). The constructor is invoked automatically whenever an object of class is created. The constructor may have accept the parameters. The constructor are divided into 3 categories :-

(i) default constructor
(ii) parametrized constructor
(iii) copy constructor.

(i) default constructor

⇒ It is also called non-Parameter constructor. It is a type of constructor which do not accept any parameter. it has empty parameter list. In a class we can define only one default constructor because it has no return type and no parameter so it can not be overloaded. It called automatically when the object of class is created.

Syntax

```
Public:
    Classname ()
    {
        Statement;
    }
```

declaratio

calling

```
Classname obj;
```

Example

```
#include<iostream.h>
#include<conio.h>
Class Sum
{
private:
    int x,y;
public:
    Sum()          -> default constru
    {
        x=10;
        y=20;
    }
    void S1()
    {
        cout<<"Sum = "<<(x+
    }
};
void main()
{
    Sum S;          -> calling
    S.S1();
}
```

(iii) Parameterized Constructor :

⇒ When different object needs to be Initialize with different value, a parameterized constructor can be declared. The parameterized constructor is a type of constructor that accepts one or more parameter at the time of declaration of the object and Initialize the data member of the object with these parameters. In a class we can define multiple parameterized constructor by overloading. It is also know as argument constructor.

Syntax

declaration
```
Public :
    classname (argument)
    {
        Statement;
    }
```

Calling

```
classname obj (parameters);
```

Example :-

⇒
```
#include <iostream.h>
#include <conio.h>
class area
{
    private :
```

```
float r, ar;
public:
area (float x)
{
r = x;
}
void show()
{
ar = 3.14 * r * r;
cout << "area =" << ar;
}
};
```

```
void main()
{
float a;
cout << "enter radius";
cin >> a
area p(a); // call
p.show();
getch();
}
```

(iii) Copy Constructor

⇒ A copy constructor is a type of constructor which is used to Initialize an object from another object. The process of Initializing through a copy constructor is known as copy Initialization. The copy constructor call whenever an object of class is created and pass the object as argument. A copy constructor a reference to an object of the same class as an argument.

The copy constructor also use as same as classname and It does not have any return type

Syntax

```
classname ( classname &object)
{
    statement;
    statement;
}
```

Calling -> classname obj. (another obj.);
or
classname obj = obj;

Example:-

```
#include <iostream.h>
#include <conio.h>
class area
{
  private:
    float x;
  public:
    area()                          // default
    {
      x = 10;
    }
    area ( area &p)                 // copy constructor
    {
      x = p.x;
    }
    void show()
    {
      cout << "area = "
           << 3.14*x*x;
    }
};
```

```
void main()
{
  area a;
  area b(a);
  b.show();
  getch();
}
```

6. What is function overloading? Write a c++ program to demonstrate the concept of function overloading

=> Function overloading is defined as the process of having two or more functions with same name but different. In parameters. The function is redefine with same name by using either different no. of argument or different type of argument and also by the different sequence of the argument. Whenever these functio are called the compiler determin and call the most appropriate function which match the parameter type. The function overloading feature is used to improve the readability of the code and It help the programmer because they do not need to remember various function name. The function overloading is used when a similar operation is performed on different type of argument or different numbers of argument.

Example -

```
#include <iostream.h>
#include <conio.h>

void Sum (int, int);
void Sum (int, int, int);
void Sum (float, float);
void Sum (int, float);
void main()
{
    int a,b,c;
    float d,e;
    clrscr();

    cout<<" enter 3 Int value";
    cin>>a>>b>>c;
    cout<<" enter 2 float value";
    cin>>d>>e;
    Sum (a,b);
    Sum (a,b,c);
    Sum (d,e);
    Sum (a,d);
    getch();
}
void Sum (int x,int y)
{
    cout<<" Sum = "
        <<x+y<<endl;
}
```

```
void Sum (int x, int y
          int z)
{
    cout<<" Sum = "
        <<x+y+z<<endl;
}
void Sum (float p,
          float q)
{
    cout<<" Sum = "
        <<p+q<<endl;
}
void Sum (int m
          float n)
{
    cout<<" Sum = "
        <<m+n;
}
```

7. What is operator overloading. Write program to overload (++ unary), (=+ short hand), (- binary), (= assignment)

ans:- Operator overloading is a compile time polymorphism. In which the operator is overloaded to provide the special meaning to the user defined data type. In C++ operator overloading use operator keyword for declaration. We can define or overload most of the built in operator like arithmatic operator, relational operator inc/dec. operator & but we can not overload some operators like scope resolution operator (::), dot (.) operator) conditional operator (?: etc. These operators can be overloaded globally or on class by class-basis. The operator overloading function accept the reference of object as an argument using & (address). and the function have return type even void. Operator overloading function is call automatically when object is use with these operators.

Syntax

```
(Return type) (operator) (Symbol) (Argument type)
{
    Statement;
    Statement;
}
```

Example

(i) ++ Unary

```
#include <iostream.h>
#include <conio.h>
class Incr
{
    private:
        int a;
    public:
        void get()
        {
        cout << " enter
        any number";
        cin >> a;
        }
    void show()
        {
    cout << " after
    Increment the
    value is =" << a;
    }
void operator ++(int)
{
    a
    ++ ;
};
void main()
{
    Incr I;
    I.get();
    I ++;
    I.show();
    getch();
}
```

(ii) (-)(binary)

```
=> #include<iostream.h>
#include<conio.h>
class complex
{
  private:
  int real, img;
  public:
  void get()
  {
    cout<<" enter
    real & Img
    value";
    cin>>real>>img;
  }
  void show()
  {
    cout<<"real <<"+"
    <<img<<"i"<<endl;
  }
complex operator -
  (complex &obj)
  {
    complex t;
t.real=real-obj.
      real;
t.img=img-obj.
      Img;
  return(t);
  }
};
```

```
void main()
{
  complex c1,c2,c3
  c1.set();
  c1.get();
  c2.get();
  c3=c1-c2;
  cout<<" after
  Subtract value
  c3.show();
  getch();
}
```

(iii) = assignment.

```
#include<iostream.h>
#include<conio.h>
class equal
{
    private:
        int a,b;
    public:
        void get()
        {
        cout<<"enter
        two value";
        cin>>a>>b;
        }
    void show()
    {
    cout<<"a="<<a
    <<endl;
    cout<<"b="<<b
    <<endl;
    }
void operator =(
        equal &obj)
    {
    a=obj.a;
    b=obj.b;
    }
};
void main()
{
    equal e,f;
    e.get();
    f=e;
    f.show();
    getch();
}
```

(iv) Shorthand +=

```
#include<iostream.h>
#include<conio.h>
class short
{
    private:
        int x,y;
    public:
        void get()
        {
    cout<<"enter two
        values";
    cin>>x>>y;
        }
    void show()
    {
    cout<<"x="<<x
    <<endl<<"y="<<y;
    }
void operator +=
    (short &obj)
    {
    x=x+obj.x;
    y=y+obj.y;
    }
};
```

```
void main()
{
    short s,t;
    clrscr();
    s.get();
    t.get();
    t+=s;
    t.show();
    getch();
}
```

8. What is Friend Function? write c++ program to overload operator using Friend Function (i) - binary (ii) += Shorthand also write a program to demonstrate the concept of Friend Function.

ans:- The Friend function is the one of the most important feature of c++. It break the data hiding and Incapsulation rule. The Friend function is a type of function which is used to access the class member directly without using scope resolution operator or object. The Friend function can be called direcly and its scope available for the whole progra

Friend function use "Friend" keyword for declaration where "Friend" keyword use before return type and it must be declare in the class for which it is friend. The friend function define outside of the class as a normal function.

The friend function can be declare publically or privately within the class for which it is friend. The friend function can be came friend to more than one class in which it is declared. The friend function is also used for operator overloading. We can pass an object in friend function while calling of the function.

Syntax

declaration within class

friend <return type> <fn. name> (argument);

define outside class

return type <fn. name> <argument>
{
Statement;
}

Example

(i)

```
#include<conio.h>
#include<iostream.h>
class Sum
{
  private:
    int a,b;
  public:
    void get()
    {
    cout<<"enter
    two no";
    cin>>a>>b;
    }
friend void Fun(Sum);
};

void Fun (Sum S)
{
cout<<" Sum ="
<<S.a+ S.b;
}
void main()
{
  clrscr();
  Sum x;
  x.get();
  Fun (x);
  getch();
}
```

(ii) – binary

```
#include<iostream.h>
#include<conio.h>
class Sub
{
  Private:
    int x
  Public:
  void get()
  {
  cout<<" enter val
  cin>>x;
  void show()
  {
  cout<<" value:
  <<x;
  }
friend Sub operator
(Sub , Sub);
};

Sub operator -
(Sub &a, Sub b)
{
  Sub c;
  c.x = a.x-b
  return (c);
}
void main()
{
  clrscr();
```

```
Sub S1,S2,S3;
S1.get();
S2.get();
S3 = S1 - S2;
S3.show();
getch();
}
```

(iii) += Short hand

```
#include <conio.h>
#include <iostream.h>
class Short
{
private:
    int a;
public:
    void get()
    {
    cout<<" enter value";
    cin>>a;
```

* For Short hand use & for 1st argument

address

```
void show()
{
cout<<"value ="<<a;
}
friend Short operator
+=(Short ,Short );
};
Short operator +=
(Short x, Short y)
{ x.a += y.a
}
void main()
{
Short p,q;
p.get();
q.get();
p += q;
p.show();
getch();
}
```

9. Write the C++ program to demonstrate the concept of constructor and destructor In derived class or Inheritance.

Constructor

```
#include<iostream.h>
#include<conio.h>
class Base
{
    public:
    Base()
    {
    cout<<"Base class"<<endl;
    }
};
class derived:public base
{
    public:
    derived()
    {
    cout<<"derived class";
    }
};
void main()
{
    Derived obj;
}
```

O/P → Base class
derived class

In Inheritance the when the obj. of derived class is created then it call the Base class constructor first then derived class constructor is called.

Destructor

```
#include<iostream.h>
#include<conio.h>
class Base
{
    public:
    ~Base()
    {
    cout<<"Base cla
};
class derived:Publi
{    Base
    Public:
    ~derived()
    {
    cout<<"deri
    class"<<endl
};
void main()
{
    derived obj;
}
```

O/P -> derived class
Base class.

In Inheritance when the object of derived class goes out of the scope then by default derived class call the its own derived class destructor first and then at last it call the base class destructor.

10. Write short note on the following :-

(a) Destructor (b) this pointer
(c) abstract class (d) Access Specifier
(e) Ambugity error (f) inline function.

a) Destructor

=> Destructor is a special member function of class, use for destroy object memory block. Destructor call automatically when object goes out of scope. Destructor use classname begin with ~ (tilde) Symbol for declaration. It must be declare publically within the class. The destructor can not use any arguments so, it cannot be overloaded. There is one and only one destructor exist in a class.

Syntax

```
~ classname ()
{
    statement;
    statement;
};
```

Example:-

```
#include <iostream.h>
#include <conio.h>
class test
{   public:
    ~ test ()
    {
        cout << " destructor ";
    }
};
void main ()
{
    test t
    getch();
}
```

destructor

=> calling.

(b) this Pointer

=> This is a local object Pointer in every instance member function containing address of the caller object. In C++ this

this keyword. It is mainly use to refer calling object. In member function of a class if name the parameter of any member function of class as same as data member of the class then by using this keyword the data member of the object is represented any without the this keyword the parameter is represented in the member function.

Syntax: this -> data member

Example:-

```
#include <iostream.h>
#include <conio.h>
class Box
{
    private:
    int l,b,h;
    public:
    void set(int l,
        int b, int h)
    {
    this->l = l;
    this->b = b;
    this->h = h;
    }
    void show()
    {
    cout<<"l="<<l
    <<endl<<"b="<<b
    <<endl<<"h="<<h;
    }
};
void main();
clrscr();
Box B;
B.set(12,10,18);
B.show();
getch();
}
```

(Q) abstract class

⇒ In C++ abstract class is a class which contains at least one pure virtual function in it. or we can also create abstract class by using abstract keyword before class (abstract class classname) : The abstract class is only accessible by using Inheritance and it is used for provide an Interface for its sub classes. The purpose of an abstract class is to provide an appropriate base class from which other classes can inherit. We can not create the object of an abstract class.

Syntax

```
class classname
{
    Data member
    public:
    virtual returntype fn.name = 0;
};
```

Example :-

```
#include <iostream.h>
#include <conio.h>
class shape
{
  public:
    virtual void fun()
    =0;
};
class rect : public shape
{
  public:
    void fun()
    {
      cout<<"rectangle";
    }
};
void main()
{
  rect r;
  r.fun();
  getch();
}
```

o/p - rectangle.

=

(d) Access Specifier:
The Access Specifier are used to set boundries for availability of member of class beyond the class. It is important aspect of object oriented programming as by the help of access specifier we implement data hiding feature. There are 3 types of access specifier

(i) Public :- The public keyword allows to specify public data member and member fn. which can be access from outside of the class using object.

(ii) Private :- It is used to specify the private data member and member fn. The private member is only accessible with in the class. By default the class member are

private. So if the access specifier are missing they by default all the class member are private.

protected:- Protected keyword specify protected data member and member function where protected member only accessible by private member of the class. The protect member is not accessible in public or outside of the class.

Access specifier is also known as access modifier or visiablity mode or visibility labels.

Syntax:

```
class classname
{
    access specifier:
    data member;
    member function()
    {
    }
};
```

Example:-

```
#include <conio.h>
#include <iostream.h>
class test
{
  protected:
    int p;
  private:
    int x;
  void change()
  {
    p = 10;
  }
  public:
  void setx()
  {
    x = 20;
    change();
  }
void main()
{
  test t;
  t.setx();
  getch();
}
```

Here p is protected access in private. x is private access in public. setx() is public access from outside of class using object.

(e) Ambiguity error

=> Ambiguity error is a type of error which occur when the base and driven class contain the common member function with same argument or same name or same data member. In this suiation the compiler do not decide which member is to be call and it generate an error called ambiguity error.

ex:-

```
class test
{
  public:
    void get()
    {
    }
};
class driven:public test
{
  public:
    void get()
    {
    }
};
```

```
void main()
{
    driven d;
    d.get();
    getch();
}
```

Ambiguity error

In C++ the ambiguity error : avoid by using Pure virtual or using different name or overloading or using classname with (::) Scope resolution operator

Ex:- avoid by using Scope resolution (::)

```
#include <conio.h>
#include <iostream.h>
class test
{
    public:
    ① ← void get()
    {
    }
};
class driven:public test
{
    public:
    ⑪ → void get()
    {
    }
};
void main()
{
    driven d;
    ① - d.test::get();
    ⑪ - d.driven::get();
    getch()
}
```

(8) Inline Function

⇒ The inline Fun. is the one of the most important feature of C++. If any function is inline then the compiler place a copy of the code of that function at each point where the function is called at compile time. To create an inline function we use the "inline" keyword before return type of the function. Where using inline function the program execution

Syntax, inline return type fname (Parameter)
{
Statement;
}

Page No.:
Date: 4-3
YOUVA

Speed will increase because the Inline function jump the calling Statement after the execution of the function. All the member function define inside the class definition are by default declare as Inline function. The Inline function use maximum 5 Statement so if any function within the class we do not want to make Inline then we define the function outside of the class using Scope resolution operator (::).

* Criteria where Inline function are not used.

(i) The Inline function can not contain many Statement. We must keep Inline functions small

(ii) If a function contain loop the the compiler may not perform Inlining

(iii) If a function contain Static variable

(iv) If a function is recursive

(v) If a function contain Switch or goto Statement.

Example :-
To find maximum of given two numbers using inline function.

```
=> #include <iostream.h>
   #include <conio.h>

        inline int max(int x, int y)
        {
            if (x > y)
                return x;
            else
                return y;
        }

   void main()
   {
        int a, b;
        clrscr();
        cout << "enter two numbers";
        cin >> a >> b;
        cout << "The maximum number
             is = " << max(a, b);
        getch();
   }
```

O/P →

```
enter two number 5 ↵
8 ↵
The maximum number is = 8
```

11. Write the difference b/w constructor and destructor.

The main differences between constructor & destructor are as:-

constructor	destructor
(i) The constructor can use argument	The destructor can not use any argument
(ii) It execute atomatically when object of class is created.	It execute automatically when object of class goes out of scope.
(iii) It support overloading by difference in argument	It do not support overloading because It cannot use argument.
(iv) Constructor only use class name for declaration	destructor use ~ (tilde) symbol before class name for declaration.
(v) The constructor is use for Initialization	It is use to destroy object memory block.
(vi) We can define more than one constructor in a class b.	We can define only one destructor in a class.

	Const.	Des.
(vii)	In Inheritance the object of derived class call the base class constructor first.	In Inheritance the object call its own destructor first.
(viii)	In Inheritance the object of derived class call its own constructor after Base class constructor.	In Inheritance the object call the base class destructor after its own class destructor.
(ix)	There are 3 types of constructor	There are no types of destructor

(x)	Syntax
	public:
	classname(argument)
	{
	Statement;
	}
		Syntax
		public
		~classname()
		{
		Statement;
		}

Q. What is Static data member and Static member Function? Write a C++ Program to demonstrate the concept of Static data member and Static member Function.

Ans:- Static is a keyword which is used to create static data member or static member function. The static data member are create by using static keyword before data type and static data member is create by using static keyword before return type.

Static data member create common memory block for all class object. Static data member use" static keyword before data type and it initialize only one time for the entire program. The static variable can not initialize again and again it means it's value can not be further initialize but it can be change by incr/decr etc. By default the value of static data member is zero.

Syntax

declaration:- Static datatype name;

Initialisation:- datatype classname :: name = value;

Static member function is a type of member function which is created by using static keyword before return type. The static member function can be call even no object of the class is exist

because the static member function is call by using class name and :: (scope resolution operator. A static member function can only access the static data member or static member function or any other function from outside of the class.

declaration :- static return type fnname ( )
{
statement;
}

call -> classname :: fnname();

Example :-

```
#include<iostream.h>
#include<conio.h>
class test
{
  private:
    static int x;
    int y;
  public:
    void sety()
    {
      y=10;
    }
    void showy()
    {
      cout<<"y="<<y;
    }
    static void showx()
    {
      x++;
      cout<<"x="<<x;
    }
};
int test::x=50;
void main()
{
  test t;
  t.sety();
  t.showy();
  test::showx();
  getch();
}
```

O/P -> y=10
x=51

13. What is virtual & pure virtual function? Write the differences between virtual & pure virtual function

ans:- The virtual function is a member function which is declared and define in base class and is redefine by a driven class. It is not necessary that all driven class redefine the virtual fn. Lets assume if all member fn. of base class is useful and same for driven class but one member fn. is different in the same driven class then we make that fn. as virtual and redefine that fn. In that driven class in which their meaning is not as same as the fn. In the base class. If the fn. is not redefine the statement of fn. of base class will run

Syntax

In both base or driven class

```
virtual fn.name (parameter)
{
    statement;
}
```

Pure virtual ⇒ A pure virtual is a virtual function that has no definition within the base class. It can one declare in the base class and define in all the driven class.

The Pure virtual fn. also use virtual keyword for declaration and the fn. must be Initialize with 0. A class having Pure virtual fn. cannot create Objects. The class having Pure virtual fn. is known as abstract class.

Syntax → In difference Point (vi) ✓

• Example

```
virtual
#include<iostream.h>
#include<conio.h>
class Base
{
  public:
  virtual void show()
  {
    cout<<" Base ";
  }
};
class driven:public
{          Base
  public:
  virtual void show()
  {
    cout<<"driven"
  }
};
void main()
{
  Base B
  driven d
  B.show();
  d.show();
} getch();
```

O/P → Base driven

```
Pure virtual
#include<iostream.h>
#include<conio.h>
class Base
{
  public:
  virtual void show()=0;
};
class driven:public base
{
  public:
  virtual void show()
  {
    cout<<"driven";
  }
};
void main()
{
  driven d;
  d.show();
  getch();
}
```

O/P - driven

* class having virtual fn. can create object but the class having pure virtual cannot create object.

Page 51

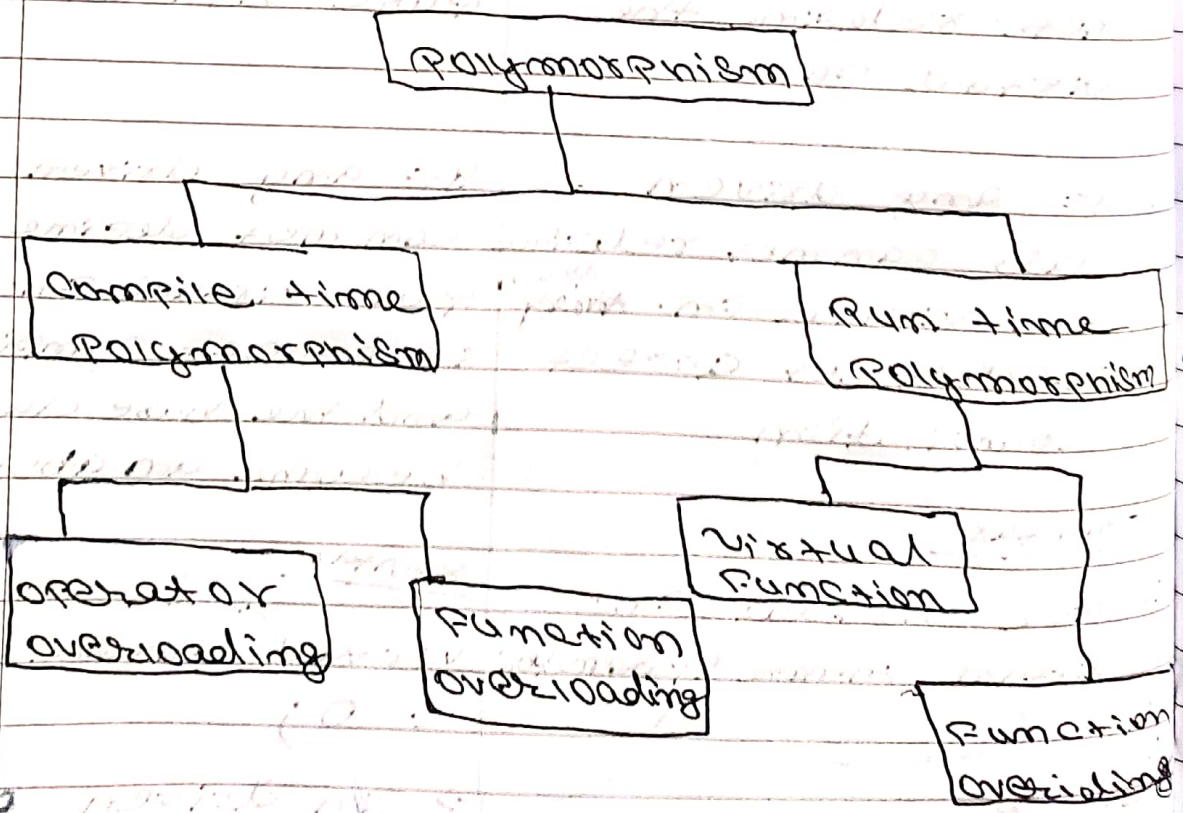
* Differences

	virtual	Pure virtual
(i)	It will be define in base class	It will one declare in the base class
(ii)	It can be redefine by the driven class	It should be define in the driven class
(iii)	The class which contain virtual fn. are not abstract class.	The class which contain pure virtual fn. are the abstract class
(iv)	All the derived class may or may not redefine the virtual fun.	All the driven class must define the pure virtual function.
(v)	If any driven class can not redefine the virtual fn. then it does not affect compilation	If any driven class can not define the pure virtual fn. then it affect compilation and the base class also become an abstract class.
	Syntax	Syntax
(vi)	virtual fn-name (parameter) { } → In base class	virtual fn-name (parameter) = 0; → In driven class

Q.14. What is Polymorphism ? Discuss different types of polymorphism with suitable example.

Ans: The Polymorphism is a greek word. The term "Polymorphism" is the combination of two words "Poly" & "morphs" which means many forms. The Polymorphism is the one of the most important feature of object oriented Programming. When the same entity function or operator behaves differently in different scenarios then it is known as Polymorphism in C++.

Types of Polymorphism



(1) Compile time Polymorphism

⇒ It is a type of polymorphism which is achieve during the compile time. It is achieve by function overloading and operator overloading. It is also known as static binding or early binding. There are two type of compile time polymorphism.

(i) operator overloading → See page no - with example

(ii) Function overloading → See page no - with example

(2) Run-time Polymorphism

It is a type of polymorphism which is achieve during the Run time. It is achieve by function overriding and virtual function. It is also known as late binding or dynamic binding. There are two type of Run time polymorphism:-

(i) virtual function → See page no - with example

(ii) Function overriding. The function overriding is a type of Run time Polymorphism which is achieve during the runtime. If a derived class define same function as defined in its base class then it is known as function overriding. but as we know it can create an

* If we do not use scope resolution the object of derived class call the fn. of derive class and vice-versa ambiguity error so, we use the scope resolution operator (::) with classname for the fn. of class you want to call.

Example :-

```
#include <iostream.h>
#include <conio.h>
class Base
{
    public:
①   void show()
    {
    cout << "Base";
    }
};
class derived:public
    base
    public
②   void show()
    {
    cout << "derived.";
    }
};
void main()
{
    derived d,e;
d.base::show();
e.derived::show();
    getch();
}
    or
void main()
{base b;
 derived d;
①  b.show();
②  d.show();
    getch();
}
```

15. What is File Stream or file handling. Write a c++ progrogram to read & write Information in binary & text file.

=

ans:- In C++ the File Stream class offers a facility to create file and store on hard disk permanently. It is also known as file handling. The file stream use fstream header file to create, open, save and do various operations over the file. The fstream library define 3 new data types:-

(i) ofstream:- It is used to create files & write information to files

(ii) ifstream:- It is used to create file & read information from files.

(iii) fstream:- It is used to create files, write information to the file and also read information from the file. It is the combination of ofstream and ifstream.

* The file stream class provide various function to do operation on the file. Some of these fn. are:-

(i) open() → It is used to create a file.

syntax → pointer.open("file name");

(ii) close() → It is used to close an existing file

(ii) get() -> It is used to read a single character from the file.

(iv) put() -> It is used to write a single character into the file.

Syntax for all

file pointer . fnname();
or
object of fstream class

* There are different modes of opening a file. Some of these are:-

(i) IOS :: in -> Input (read.

(ii) IOS :: out -> output (write.

(iii) IOS :: app -> append mode. write new data without erasing previous data of the file

(iv) IOS :: binary -> open the file in binary mode for read or write. By default opening mode is text mode.

Syntax

```
<pointername> . open ("filename",
                      opening mode);
```

ex:-
```
fstream fptr;
fptr.open("my.txt", ios::in,
          ios::binary);
```

It open the file reading in binary mode

Example:- read and write information in binary & text file.

Text mode.

```
#include <iostream.h>
#include <conio.h>
#include <fstream.h>
void main()
{
  fstream fptr;
  char ch;
  fptr.open("my.txt",
      ios::in,ios::out);
  fptr << "hello";
  while (!fptr.eof())
  {
  ch = fptr.get();
  cout << ch;
  }
  fptr.close();
  getch();
}
```

Binary mod only change in (1)

```
fptr.open("my.txt",
ios::in, ios::out,
ios::binary);
```

C++ Completed

Saurabh Kumar