UNIT - IV **Object Oriented Programming with C++** **CHAPTER 16**

Inheritance

1.What is inheritance? What are the advantages of inheritance?

- It is a process of creating new classes called derived classes, from existing or base classes.
- Inheritance allows us to inherit all the code (except declared as private) of one class to another class

The main advantage of inheritance is

- It represents real world relationships well
- It provides reusability of code
- It supports transitivity

2.What is base class?

- A class that is used as the **basis for inheritance** is called a superclass or base class.

3.Define derived class. Or why derived class is called power packed class?

- A class that **inherits from** a superclass is called a subclass or derived class
- The derived class is a **power packed** class, as it can add additional attributes and methods and thus **enhance its functionality**

4.What are the types of Inheritance ?

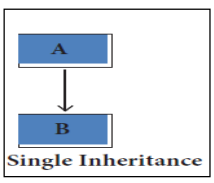
- There are different types of inheritance viz., Single Inheritance, Multiple inheritance, Multilevel inheritance, hybrid inheritance and hierarchical inheritance.

5.Explain the types of Inheritance.

There are different types of inheritance viz., Single Inheritance, Multiple inheritance, Multilevel inheritance, hybrid inheritance and hierarchical inheritance.

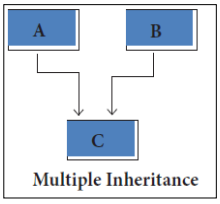
1. Single Inheritance

When **a derived class** inherits only from **one base class**, it is known as single inheritance.



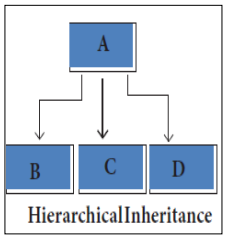
2. Multiple Inheritance

When **a derived class** inherits from **multiple base classes** it is known as multiple inheritance



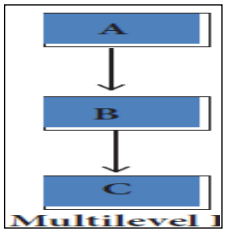
3. Hierarchical inheritance

When **more than one derived classes** are created from **a single base class ,**it is known as Hierarchical inheritance.



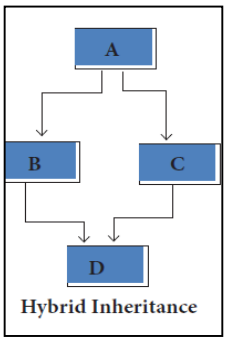
4. Multilevel Inheritance

- The **transitive nature** of inheritance.
- In multilevel inheritance a derived class itself **acts as a base class**to derive another class.



5. Hybrid inheritance

A combination of **more than one type of inheritance** is known as hybrid inheritance. It may be a combination of Multilevel and Multiple or Hierarchical and Multilevel or Hierarchical,



6.What are the points to be noted while deriving a new class?

- The keyword **class** has to be used
- The **name of the derived class** is to be given after the keyword class
- **A single colon**
- The type of derivation (the visibility mode), namely **private, public or protected**.
  Default visibility mode is **private**.
- The names of all base classes(**parent classes)** separated by **comma**.

```
class derived_class_name
:visibility_modebase_class_name
        {
        // members of derivedclass
        };
```

Ex. **Class boys : public student**

```
        {

        };
```

7.Differentiate between access specifier and visibility modes

- **Access specifiers** control the accessibility of the **class members** with in the class
- **Visibility modes** control the access of **inherited members** with in the class

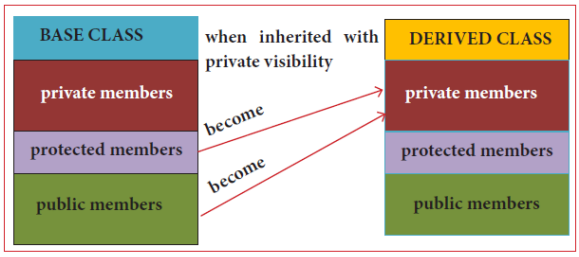
8.Explain the different visibility mode in inheritance.

- The accessibility of base class by the derived class is controlled by visibility modes.
- The three visibility modes are private, protected and public.
- The default visibility mode is private.

Private visibility mode

When a base class is inherited with **private** visibility,

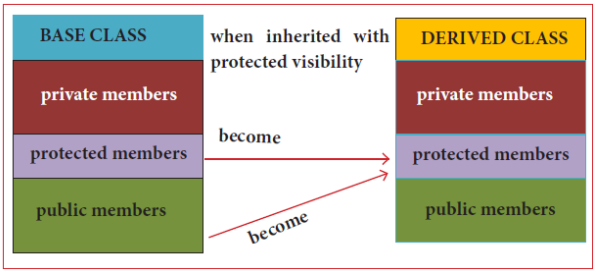
- The **public** and **protected** members of the base class become '**private**' **members** of the **derived** class



Protected visibility mode

When a base class is inherited with **protected** visibility.

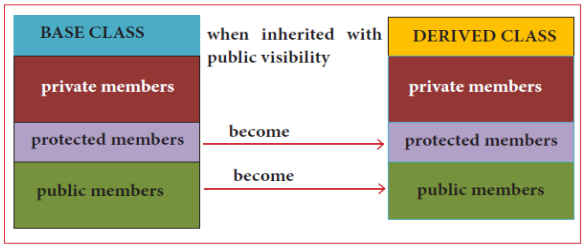
- The **protected** and **public** members of the base class become '**protected members** ' of the derived class



Public visibility mode

When a base class is inherited with **public** visibility,

- The **protected** members of the **base** class will be inherited as **protected** members of the **derived** class.
- The **public** members of the **base** class will be inherited as **public** members of the **derived** class.



- When classes are inherited the private members of the base class are **not inherited** they are **only visible**i.e continue to exist in derived classes, and cannot be accessed

9.Differentiate between private and public visibility modes.

Ref.above ans.

10.What is the size of following class?

```
classx
{
};
```

- A class without any declaration will have 1 byte size. So ,x occupies **1 byte**.

11.Explains the significance of different visibility modes from following program.

```
#include <iostream>
using namespace std;
class Shape
{
private:
int count;
protected:
int width;
int height;
public:
voidsetWidth(int w)
{
width = w;
}
voidsetHeight(int h)
{
height = h;
}
};
class Rectangle: public Shape
{
public:
intgetArea()
{
return (width * height);
}
};
int main()
{
Rectangle Rect;
Rect.setWidth(5);
Rect.setHeight(7);
cout<< "Total area: "<<Rect.getArea() <<endl;
return 0;
}
```

Output

Total area: 35

The following table contain the members defined inside each class before inheritance

MEMBERS of class	visibility modes		
	Private	Protected	Public
Shape(base class)	int count;	int width; int height;	void setWidth(int) void setHeight(int)
Rectangle (derived class only with its defined members)			intgetArea();

The following table contain the details of members defined after inheritance

In case the class rectangle is **derived** with private visibility mode

MEMBERS of class	visibility modes –private for acquiring the properties of the base class		
	Private	Protected	Public
Shape(base class)	int count;	int width; int height;	void setWidth(int) void setHeight(int)
Rectangle (derived class acquired the properties of base class with private visibility)	Private members of base classes are not directly accessible by the derived class	int width; int height; void setWidth(int) void setHeight(int)	int getArea();

Suppose the class rectangle is derived with **protected** visibility

MEMBERS of class	visibility modes –protected for acquiring the properties of the base class		
	Private	Protected	Public
Shape(base class)	int count;	int width; int height;	void setWidth(int) void setHeight(int)
Rectangle (derived class acquired the properties of base class with protected visibility)	Private members of base classes are not directly accessible by the derived class	int width; int height; void setWidth(int) void setHeight(int)	intgetArea();

Suppose the class rectangle is derived with **public**visibility

MEMBERS of class	visibility modes –public for acquiring the properties of the base class		
	Private	Protected	Public
Shape(base class)	int count;	int width; int height;	void setWidth(int) void setHeight(int)
Rectangle (derived class acquired the properties of base class with public visibility)	Private members of base classes are not directly accessible by the derived class	int width; int height;	int getArea(); void setWidth(int) void setHeight(int)

12.WriteSome Facts about the execution of constructor in inheritance.

- **Base** class constructors are executed **first** ,before the derived class constructors execution
- Derived class **can not** inherit the base class constructor.
- If there are multiple base classes ,then its start executing from the **left** most base class
- In multilevel inheritance, the constructors will be executed in **the order of inheritance**

13.Find the output of the following program.

```
#include<iostream>
using namespace std;
classbase
{
public:
base()
{
cout<<"\nConstructor of base class...";
}
~base()
{
cout<<"\nDestructor of base class.... ";
}
};
classderived:public base
{
public :
derived()
{
cout<< "\nConstructor of derived ...";
}
~derived()
{
cout<< "\nDestructor of derived ...";
}
};
class derived1 :public derived
{
public :
derived1()
{
cout<< "\nConstructor of derived1 ...";
}
~derived1()
{
cout<< "\nDestructor of derived1 ...";
}
};
int main()
{
derived1 x;
return 0;
}
```

Output:

Constructor of base class...
Constructor of derived ...
Constructor of derived1 ...
Destructor of derived1 ...
Destructor of derived ...
Destructor of base class....

14.How the size of derived class object is calculated?

Size of derived class object **=**
Size of all **base class** data members **+** size of all **derived class** data members.

15.What do you mean by overriding /Shadowing?

- When **a derived class member** function has the **same name** as that of its **base class member** function ,
- The **derived class member** function **shadows/hides** the **base class's** inherited function .
- This situation is called **function overriding /Shadowing .**
- This can be resolved by giving the **base class name** followed by :: and the **member function name**

Ex.**employee :: display();**

16.Define"this" pointer

- **this** pointer used to refer the **current objects members**
- 'this' pointer is a **constant pointer** that holds the **memory address** of the current object. .
- It identifies the currently calling object.
- It is useful when the **argument variable name** in the member function and the **data member name** are same. Syntax: **this->data member name**

```
#include<iostream>
using namespace std;
class T
{
        public:
        int x;
void foo()
  {
    x = 6;      // same as this->x = 6;
this->x = 5; // explicit use of this->
cout<<endl<<x<<"  "<<this->x;
  }
void foo(int x)
  {
this->x = x;            // unqualified x refers to the
parameter.'this->' required for disambiguation
cout<<endl<<x<<"  "<<this->x;
  }};
```

17.Answer the following questions based on the given program.

```
#include<iostream>
#include<string.h>
#include<stdio.h>
usingnamespacestd;
class publisher
{
charpname[15];
charhoffice[15];
char address[25];
double turnover;
protected:
char phone[3][1O];
void register();
public:
publisher();
~publisher();
void enter data();
voiddisp data();
};

class branch
{
charbcity[15];
charbaddress[25];
protected:
intno_of_emp;
public:
charbphone[2][10];
branch();
~branch();
void have data();
void give data();
};

class author: public branch, publisher
{
Intaut_code;
Charaname[2O];
float income;
public:
author();
~author();
voidgetdata();
voidputdata();
};
```

a.Which type of Inheritance is shown in the program?

Multiple inheritance

b.Specify the visibility mode of base classes.

public:

c. Give the sequence of Constructor/Destructor Invocation when object of class author is created.

Constructor of publisher
Constructor of branch
Constructor of author
Destructor of author
Destructor ofbranch
Destructor of publisher

d. Name the base class(/es) and derived class (/es).

Base class: **publisher, branch**Derived class**:author**

e. Give number of bytes to be occupied by the object of the following class: (a) publisher (b) branch (c) author

publisher:93 bytes branch:64 bytes author: 28 bytes

f. Write the names of data members accessible from the object of class author.
Phone,no_of_emp, bphone,aut_code,aname,income
g. Write the names of all member functions accessible from the object of class author.
getdata(),putdata(),give data(),have data(),disp data(), enter data();
h. Write the names of all members accessible from member functions of class author
getdata(),putdata(),give data(),have data(),disp data(), enter data(),register()
Phone,no_of_emp, bphone,aut_code,aname,income

18.Consider the following c++ code and answer the questions

```
class Personal
{
Int Class,Rno;
char Section;
protected:
char Name[20];
public:
personal();
void pentry();
void Pdisplay();
};
Class Marks:private Personal
{
float M{5};
protected:
char Grade[5];
public:
Marks();
void M entry();
void M display();
};
classResult:public Marks
{
floatTotal,Agg;
public:
charFinalGrade, Commence[20];
Result();
void R calculate();
void R display();
}:
```

a. Which type of Inheritance is shown in the program?
Multilevel Inheritance
b. Specify the visibility mode of base classes.
private , public
c Give the sequence of Constructor/Destructor Invocation when object of class Result is created.
Constructor
Personal()
Marks()
Result()
Destructor
Result()
Marks()
Personal()
d. Name the base class(/es) and derived class (/es).
Base :**Personal,Marks**Derived **: Marks , Result**
e.Give number of bytes to be occupied by the object of the following class:
(a) Personal turbo 25bytes,devc++ 29bytes
(b) Marks 25 bytes(c) Result 29bytes

f. Write the names of data members accessible from the object of class Result.
Name,Grade,total,Agg,FinalGrade.Commence
g. Write the names of all member functions accessible from the object of class Result.
R calculate(),R display(),M entry(),M display(),personal();void pentry(); voidPdisplay();
h. Write the names of all members accessible from member functionsof class Result.
R calculate(),R display(),M entry(),M display(),personal();void pentry(); voidPdisplay();

19.Write the output of the following program

```
#include<iostream>
using namespace std;
class A
{
protected:
int x;
public:
void show()
{
cout<<"x = "<<x<<endl;
}
A()
{
cout<<endl<<" I am class A "<<endl;
}
~A()
{
cout<<endl<<" Bye ";
}
};
class B : public A
{
{
protected:
int y;
public:
B(int x, int y)

{
this->x = x;
this->y = y;
```

```
}
B()
{
cout<<endl<<" I am class B "<<endl;
}
~B()
{
cout<<endl<<" Bye ";
}
void show()
{
cout<<"x = "<<x<<endl;
cout<<"y = "<<y<<endl;
}
};
int main()
{
AobjA;
B objB(30, 20);
objB.show();
return 0;
}
```

Output:

I am class A
I am class B
X=30
Y=20
Bye
Bye

20.Debug the following program

Output

15
14
13

Program :

```
%include(iostream.h)
#include<conio.h>
Class A
{
public;
int a1,a2:a3;
Void getdata[]
{
a1=15;
a2=13;a3=13;
}
}
Class B:: public A()
{
PUBLIC
voidfunc()
{
int b1:b2:b3;
A::getdata[];
b1=a1;
b2=a2;
a3=a3;
cout<<b1<<'\t'<<b2<<'t\'<<b3;
}
void main()
{
clrscr()
B der;
der1:func();
getch();
}
```

To Know Yourself

Define Pointer variable

- A pointer is a variable that hold a memory address of other variable
- In Pointer ,the memory location of a variable can be directly accessed.
- The address of (&) and the value at operator (*)are deals with pointer.

To declare

Syntax :**data type *Variable name;**

Ex. **Int *abc;**- Here The variable abc can only store addresses.

Initialization of pointer variable:

```
Int *abc , n;
n=10;
abc = &n;
```

- Pointer variable can store the address of other variables
- Pointer variable and assigning variable should have same data type.

For example

```
Int *abc ; float n;
n=10.5;
abc = &n;
```

Compiler shows an error because,
Pointer variable and assigning variable should have same data type

==========