STUDY ON INHERITANCE OF CLASSES

Pallav Thapa
Deaprtment Of Information Technology, Dronacharya College Of Engineering, Gurgaon

***Abstract-*.This paper address about the inheritance in object oriented programming. In this paper we shall be discussing about the need for inheritance, different forms of inheritance, derived and base class, inheritance and access control**

I. INTRODUCTION

When an object or class is based on another object or class, using the same implementation or specifying implementation to maintain the same behavior it is called inheritance .It is a mechanism for code reuse and to allow independent extensions of the original software via public classes and interfaces. The relationships of objects or classes through inheritance give rise to a hieracy. Inheritance was invented in 1967 for Simula. Simula is a name for two simulation programming languages Simula I and Simula 67,Developed in the 1960s at the Norwegian Computing Center in Oslo by Ole-Johan Dahl and Kristen Nygaard Syntactically, it is a fairly faithful superset of ALGOL 6 One of the most important concepts in object-oriented programming is that of inheritance. Inheritance allows us to define a class in terms of another class, which makes it easier to create and maintain an application. This also provides an opportunity to reuse the code functionality and fast implementation time. The idea of inheritance implements is a relationship. For example, mammal IS-A animal, dog IS-A mammal hence dog IS-A animal as well and so on.

II. NEED FOR INHERITANCE

Inheritances, is a concept of object-oriented programming languages. There are several reasons why inheritance was introduced into Object-oriented language. We are discussing here some major reasons behind the introduction of inheritance.

1. One major reason behind this is the capability to express the inheritance relationship which ensures the closeness with the real-world models
2. Another reason is the idea of reusability. The advantages of reusability are: faster development time, easier maintenance, and easy to extend. Inheritance allows the addition of additional features to an existing class without modifying it. One can derive a new class from an existing one and add new features to it.
3. Inheritance is transitive in nature :

Suppose we inherit class B from existing class A .The class C and D inherit from class B .Later we find that class A(base class of B) has a bug that must be corrected . After correction the bug in A, it automatically will be reflected across all classes that inherit from A, if the class A has been inherited without changes. See the reduction in the amount efforts that one would have done if each class inherited from A was to be modified separately, a gifted benefit of being transitive.

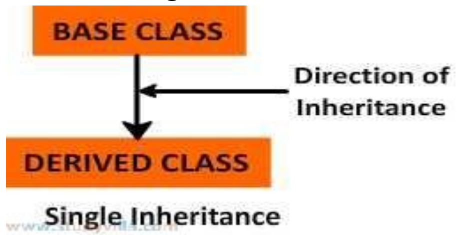
In this paper we have considered the following types of Inheritance:

- ❖ Single Level Inheritance
- ❖ Multiple Inheritances
- ❖ Hierarchical inheritance
- ❖ Multilevel Inheritance
- ❖ Hybrid Inheritance.

III. DIFFERENCE FORMS OF INHERITANCE

1. SINGLE INHERITANCE

When a subclass inherits only from one base class, it is known as single inheritance



A derived class with only one base class is called single inheritance.

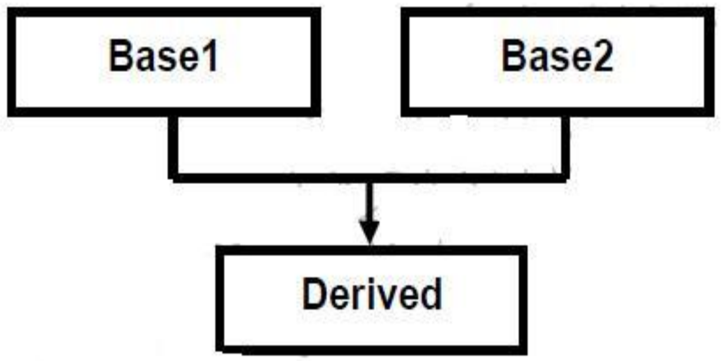
IV. EXAMPLE OF SINGLE INHERITANCE:

#include<iostream.h>

```
#include<conio.h>
class B
{
private:
int a;
public:
int b;
void set_ab()
{
a = 5;
b = 10;
}

int get_a()
{
return a;
}
};

class D : public B
{
private:
int c;
public:
void mul()
{
c = b*get_a();
}

void display()
{
cout<<"a = "<<get_a()<<"\n";
cout<<"b = "<<b<<"\n";
cout<<"c = a*b = "<<c<<"\n\n";
}
};

void main()
{
clrscr();
D d;

d.set_ab();
d.mul();
d.display();

d.b = 20;
d.mul();
d.display();
getch();
}
```

V. MULTIPLE INHERITANCES

When a sub class inherits from multiple base classes, it is known as multiple inheritances.



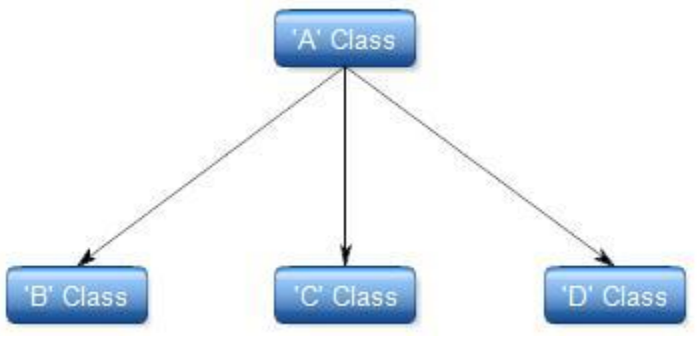
The derived class “stuaddressinfo” contains the data members: address only.

```
#include<iostream.h>
#include<conio.h>
Class Stuinfo
{
Private:
Char name [25];
Int roll;
Public:
Void getdata ()
{
Cout<<”\n Enter name”;
Cin>>name;
Cout<<”\n Enter roll”;
Cin>> roll;
}
Void display ()
{Cout<<”\n name=”<<name;
Cout<<”\n Roll number”<<roll;
}
};
Class Stuacademicinfo
{
Private:
Char course [25];
Char semester [15];
Public:
Void getdata ()
{
Cout<<”\n Enter course name”;
```

```
Cin>>course;
Cout<<"\n Enter semester";
Cin>>semester;
}
Void display ()
{
Cout<<"\n Course "<<course;
Cout<<"\n semester"<<semester;
};
Class stuaddressinfo: public Stuinfo, public Stuacademicinfo
{
Private:
Char address [25];
Public:
Void getdata ()
{
Stuinfo:: getdata ();
Stuacademicinfo:: getdata ();
Cout<<"\n Enter the address";
Cin>>address;
}
Void display ()
{
Stuinfo:: display ();
Stuacademic:: display ();
Cout <<"\n Address"<<address;
};
Void main ()
{
Stuaddressinfo obj;
Obj .getdata ();
Obj.dispay ();
getch ();
}
```

VI. HIERARCHICAL INHERITANCE

When many sub class inherit from a single base class, it is known as hierarchical inheritance.

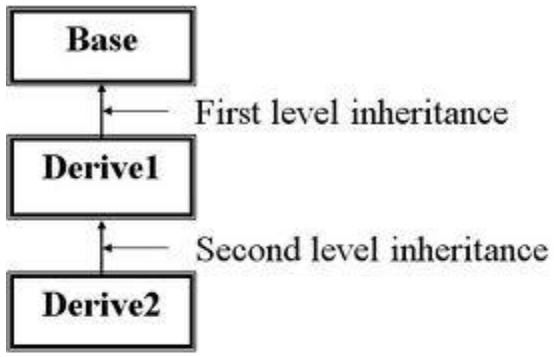


VII. EXAMPLE OF HIERARCHICAL INHERITANCE

```
#include<iostream.h>
#include<conio.h>
class polygon
{
protected:
int width, height;
public:
void input(int x, int y)
{
width = x;
height = y;
}
};

class rectangle : public polygon
{
public:
int areaR ()
{
return (width * height);
}
};

class triangle : public polygon
{
public:
int areaT ()
{
return (width * height / 2);
}
};

void main ()
{
clrscr();
rectangle rect;
triangle tri;
rect.input(6,8);
tri.input(6,10);
cout <<"Area of Rectangle: "<<rect.areaR()<< endl;
cout <<"Area of Triangle: "<<tri.areaT()<< endl;
getch();
}
```

VIII. MULTILEVEL INHERITANCE

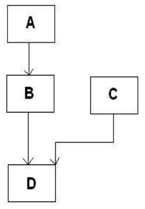
The transitive nature of inheritance is reflected by this form of inheritance. When a subclass inherits from that itself inherits from another class, it is known as multilevel inheritance.



```
#include<iostream.h>
#include<conio.h>
class top                    //base class
{
public :
int a;
void getdata()
{
cout<<"\n\nEnter first Number :::\t";
cin>>a;
}
void putdata()
{
cout<<"\nFirst Number Is :::\t"<<a;
}
};

//First level inheritance
class middle :public top      // class middle is derived_1
{
public:
int b;
void square()
{
getdata();
b=a*a;
cout<<"\n\nSquare Is :::"<<b;
}
};

//Second level inheritance
class bottom :public middle   // class bottom is derived_2
{
public:
int c;
void cube()
{
square();
c=b*a;
cout<<"\n\nCube :::\t"<<c;
}
};

int main()
{
clrscr();
bottom b1;
b1.cube();
getch();
}
```

IX. HYBRID INHERITANCE

Hybrid inheritance combines two or more forms of inheritance, e.g, when a sub class inherits from multiple base classes and all of its base classes inherit from a single base class, this form of inheritance is known as hybrid inheritance



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

```
class A     //Base class
{
public:
int l;
void len()
{
cout<<"\n\nLenght :::\t";
cin>>l;                  //Lenght is enter by user
}
};
class B :public A   //Inherits property of class A
{
public:
int b,c;
void l_into_b()
{
len();
cout<<"\n\nBreadth :::\t";
cin>>b;                    //Breadth is enter by user
c=b*l;                     //c stores value of lenght *
Breadth i.e. (l*b) .
}
};

class C
{
public:
int h;
void height()
{
cout<<"\n\nHeight :::\t";
cin>>h;                  //Height is enter by user
}
};

//Hybrid Inheritance Level
class D:public B,public C
{
public:
int res;
void result()
{
l_into_b();
height();
res=h*c;                        //res stores value of c*h
where c=l*b and h is height which is enter by user
cout<<"\n\nResult (l*b*h) :::\t"<<res;
}
};
int main()
{
clrscr();
D d1;
d1.result();
getch();
}
```

X. CONCLUSION

The mechanism of deriving a new class from an old class is called inheritance, it provides the concept of Reusability that is
the most important concept in C++. All types of inheritance with its own features and its use to provide users to reusability
concepts strongly, to give save time and reduce the complexity.
Here, in this paper we have to study the above five types of inheritance. We have to find that inheritance is central
concepts in C++ that allows deriving a class from multiple classes at a time.

REFERENCE

Books:
1. E Balagurusamy, Object oriented Programming with C++, 6
th Edition, New Delhi: Tata McGraw-Hill
Publishing Company Limited.
2. Yashavant Kanetkar, Test your C++ Skills, 1st Edition, BPB Publication.