INHERITANCE & ITS TYPES

Jyoti & sonu yadav

Abstract- **Modern object oriented programming languages such as C++ provide convenient abstractions and data encapsulation mechanism for software developers . C++ strongly supports the concept of reusability. The mechanism of driving a new class from an old one is called inheritance . The old class is referred to as the base class and the new one is called the derived class or subclass. In particular , the C++ language exhibits features such as multiple inheritance , static and dynamic typecasting that makes static analyzers for C++ quite hard to implement .We have implemented our ideas in a framework for C++ called CILpp that is analogus to the popular C intermediate language (CIL) framework.**

Index Terms- **Reusability , base class-sub class**

I. INTRODUCTION

Inheritance is the process by which objects of one class acquire the properties of objects of another class in the hierarchy.
Such mechanisms include function and operator overloading, constructors and destructors, multiple class inheritance,
dynamic virtual-function dispatch, templates, exceptions, functors, standard libraries such as STL and BOOST

For example, the scooter is a type of the class two-wheelers, which is again a type of (or kind of) the class motor vehicles. As shown in the below diagram the principle behind it is that the derived class shares common characteristics with the class from which it is derived.
In this paper we have considered the following types

II. SINGLE LEVEL INHERITANCE

A derived class with only one base class is called single inheritance. Consider a simple example – in this program show a base class B and derived class D. The class B contain one private data member , one public data member and three .

<u>Example of single level inheritance:</u>

```
#include<iostream.h
Class value
{
Protected:
Int val;
Public:
Void set value (int a)
{val=a;}
};
Class cube: public value
{
Public:
Int cube()
{ return (val*val*val); }
};
Int main()
{
Cube cub;
Cub.set values (5);
Cout<<"The cube of 5 is :: "<<cub.cube() <<end1;
Return 0;
}
```

III. MULTIPLE INHERITANCE

A class can inherit properties from more than one class which is known as multiple inheritance. It allows us to combine the features of several existing classes as a starting point for defining a new classes.

<u>Example of multiple inheritance:</u>

```
#include<iostream.h>
Class M
{
Protected:
Int m;
Public:
Void get m(int);
};
Class N
{
Protected:
Int n;
Public:
Void get n(int);
};

Class P :public M,public N
{
Public:
Void display();
};
Void M:: get m(int x)
{
M=x;
}
Void N:: get n(mt y)
{
N=Y;
}
void P:: display ()
{
Count<<"m="<<m<<"\n";
Count<<"n="<<n<<"\n";
Count<<"m*n="<<m*n<<"\n";}
Int main()
{
Pp1
P1.get m(10);
P1.get n(20);
P1.display();
Return 0;
}
```

IV. HIERARCHICAL INHERITANCE

When the properties of one class are inherited by more than one class, it is called hierarchical inheritance.

For example : Bank accounts

```
Class first
{
Int x=10, y=20;
Void display()
{
System.out.println("This is the method in class
one");
System.out.println("value of X="+x);
System.out.println("value of Y="+y);
}

Class two extends first
{
Void add()
{
System.out.println("This is the method in class
two");
System.out.println(x+y="+(x+y));
}
}
Class three extends first
{
Void mul()
{
System.out.println("This is the method in class
three");
System.out.println("x*y="+(x*y));
}
}

class Hier
{
public static void main(string args[])
{
two t1=new two();
three t2=new three();
t1.display();
t1.add();

}
}
```

V. MULTI LEVEL INHERITANCE

A class can be derieved from another derieved class which is known as multilevel inheritance.
For example : if we take a case of multilevel inheritance , where class B inherits from class A , and class C inherits from class B , which shows the order of constructor calling.

Example of multilevel inheritance:

```
Class A
{
A0
{
System.out.println("construrctor of class A has been
called");
}
}
Class B extends A
```

```
{
B0
{
Super();
System.out.println("constructor of class B has been
called");
}
}

Class C extends B
{
C0
{
Super();
System.out.println("constructor of class C has been
called");
}
}
Class constructor call
{
public static void main (string[] args)
{
System.out.println("-------WELCOME TO
CONSTRUCTOR CALL DEMO-----");
C objc = newC();
}
}
```

VI. HYBRID INHERITANCE

There could be situations where we need to apply two or more types of inheritance to design one inheritance called hybrid inheritance.
For instance , consider the case of processing the student results,the weight age for support is stored in seperate classes.

Example of hybrid inheritance :

```
#include<iostream.h>
Class mm
{
Protected:
Int roll no;
Public:
Void get num(int a)
{ roll no = a;  }
Void put num ()
{ cout <<"Roll number is :"<<roll no <<"\n";  }
};
Class marks : public mm
{
Protected:
Int sub1;
Int sub2;
Public:
Void get marks (int x,int y)
{
Sub1 = x;
Sub2 = y;
}
Void put marks(void )
{
Cout<<"subject 1:" << sub1 <<"\n";
Cout<<"subject 2:" << sub 2<<"\n";
}
};
class extra
{
protected:
float e;
public:
void get extra(float s)
{ e=s;}
Void put extra (void)
{ cout << "Extra score::" << e<<"\n";}
};
class res : public marks, public extra
{
Protected:
Float tot;
Public:
Void disp(void)
{
tot = sub1+sub2+e;
put num();
put marks();
put extra();
cout << "total;"<<tot;
}
int main ()
{
res std1;
std1 . get_num(10);
```

```
std1.get_marks(10,20);
std1.get_marks(33.12);
std1.disp();
return 0
```

VII. CONCLUSION

Here in this paper we have to study the above five types of inheritance. 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.

REFERENCES

Books

[1] E Balagurusamy, Object oriented Programming with C++, 6th Edition, New Delhi: Tata McGraw-Hill Publishing Company Limited.

[2] Yashavant Kanetkar, Test your C++ Skills, 1st Edition, BPB Publication.

[3] Salivahanan S, Arivazhagan S – Digital Circuits and Design 4th Edition, Vikas Publishing House Pvt Ltd.

[4] Yashwant Kanetkar, Let Us C++ Solutions – 2010 , BPB Publication.

[5] S. Geetha, D. Jeya Mala – Object Oriented Analysis and Design Using UML, 1st Edition, McGraw-Hill Education.

[6] Yashwant Kanetkar, Data Structures Through C++ 1st Edition, BPB Publication

[7] ISRD Group, Data Structures Through C++ 1st Edition, McGraw-Hill Education.

[8] Bjarne Stroustrup, The C++ Programming Language, Person Publication.

[9] Kyle Loudon, C++ Pocket Reference 1st Edition, O’reilly Publication.

[10] Nell B. Dale, Studyguide for C++ Plus Data Structure, Academic Internet Publishers