**Inheritance**

1. Inheritance is another important aspect of object oriented programming. C++ supports this concept.
2. The relationships can be expressed through derivations.
3. Using inheritance, a new class can be derived by reusing the properties of existing class.
4. The derived class inherits all features from a base class and it can have additional features of its own.
5. Inheritance is the concept through which one class acquires the properties of another class.
6. The class from which the properties are inherited is called the base class.
7. The class that inherits the properties from the base class is called as derived class.
8. Code reusability is the powerful feature of inheritance. i.e., once a base class is written, tested and saved, it need not be modified again, but reused in a different situation. This is done by creating derived class from the existing classes.

Advantages:

* Reusing existing code
* Faster development time
* Easy to maintain
* Easy to extend
* Memory utilization

The following example illustrates the advantages of inheritance.

**Defining derived classes**

First the base class is defined. Next the derived class is defined by specifying its relationship with the base class.

class derived\_class\_name : visibility-mode base\_class\_name

{

//Members of the derived class

}

Here colon indicates the derivation from the base class

visibility\_mode Type of derivation. i.e., private, public or protected

The body of the derived class contains its own data members and member functions. The class definition should be terminated by the semicolon. If no visibility\_mode is specified, the default visibility mode is private.

Note that all the members of the class except private can be inherited

**TYPES OF INHERITANCE:**

* Single inheritance
* Multiple inheritance
* Multilevel inheritance
* Hybrid inheritance
* Hierarchical inheritance

Single inheritance:

The class which draws the properties from only one existing class.

Multiple inheritance:

The class which draws the properties from more than one existing classes.

Mutlilevel inheritance:

It is represented in chain form.

Hierarchical inheritance:

It contains many sub levels. It specifies tree structure.

Hybrid inheritance:

It is combination of multiple and multilevel inheritance.

SINGLE INHERITANCE:

SYNTAX:

class a ---------------> BASE OR PARENT CLASS

{

};

class b:public a --------> b is a derived or child class

{

};

public : it is mode of derivity

Note: object should be created for derived class.

PROTECTED:

Protected members are available only to the derived class .

Under inheritance, the data member be declared under protected category.

Non inheritance , the data member be declared under private category.

Member functions always be declared under public category.

// SINGLE INHERITANCE

#include<iostream.h>

#include<conio.h>

class first

{

protected:

int x;

public:

void getdata()

{

cout<<"enter the value for x"<<endl;

cin>>x;

}

};

class second : public first

{

public:

void display()

{

cout<<x<<endl;

}

};

void main()

{

second s;

clrscr();

s.getdata();

s.display();

getch();

}

AMBIGUITY IN SINGLE INHERITANCE:

When both the base and derived class contain the same function name , then the Object of the derived invokes derived class function not the base.

This is ambiguity.

// SINGLE INHERITANCE

#include<iostream.h>

#include<conio.h>

class first

{

public:

void display()

{

cout<<"base class function"<<endl;

}

};

class second : public first

{

public:

void display()

{

cout<<"derived class function"<<endl;

}

};

void main()

{

second s;

clrscr();

s.display(); // calling the derived class function (overriding)

getch();

}

**OVERRIDING:**

Intentional hiding of base class function by using derived class function.

Provided both the base and derived class contain same function name with same signature .

Object of the derived class invokes the derived class function is called method overriding.

MULTIPLE INHERITANCE:

class first

{

};

class second

{

};

class three:public first,public second

{

};

first , second-------> BASE CLASSES

three-----------> DERIVED CLASS.

Gross or earnings = basic+ hra+da+ca+ma+fa

Deduction = lic + it + pf

Netsalary = gross - deduction

class first

{

};

class second:public first

{

}

// MULTIPLE INHERITANCE

#include<iostream.h>

#include<conio.h>

class earnings

{

protected:

int basic,hra,da,ca,ma,fa;

};

class deduction

{

protected:

int lic,it,pf;

};

class net:public earnings,public deduction

{

protected:

int total,gross,ded;

public:

void getdetails();

void calcdetails();

void putdetails();

};

void net::getdetails()

{

cout<<"enter the earnings details"<<endl;

cin>>basic>>hra>>da>>ca>>ma>>fa;

cout<<"enter deduction details"<<endl;

cin>>lic>>it>>pf;

}

void net::calcdetails()

{

gross = basic + hra + da + ca + ma +fa;

ded = lic + it +pf;

total = gross - ded;

}

void net::putdetails()

{

cout<<"gross="<<gross<<" "<<"deduction ="<<ded<<"net="<<total<<endl;

}

void main()

{

net obj;

clrscr();

obj.getdetails();

obj.calcdetails();

obj.putdetails();

getch();

}

MULTILEVEL INHERITANCE

#include<iostream.h>

#include<conio.h>

class first

{

protected:

int x;

};

class second:public first

{

public:

void input()

{

cin>>x;

}

};

class three:public second

{

public:

void output()

{

cout<<x<<endl;

}

};

void main()

{

three t;

clrscr();

t.input();

t.output();

getch();

}

// MULTILEVEL INHERITANCE

#include<iostream.h>

#include<conio.h>

class first

{

protected:

int x;

};

class second:public first

{

public:

void input()

{

cin>>x;

}

};

class three:protected second

{

public:

void output()

{

cout<<x<<endl;

}

};

void main()

{

three t;

clrscr();

t.input();

t.output();

getch();

}

// HYBRID INHERITANCE

#include<iostream.h>

#include<conio.h>

class exam

{

protected:

char name[20];

public:

void getname()

{

cout<<"enter the name"<<endl;

cin>>name;

}

};

class exam1

{

protected:

int rollno;

public:

void getrollno()

{

cout<<"enter the rollno"<<endl;

cin>>rollno;

}

};

class evaluator:public exam1

{

protected:

int marks;

public:

void getmarks()

{

cout<<"enter the marks"<<endl;

cin>>marks;

}

};

class result:public evaluator,public exam

{

public:

void output()

{

cout<<name<<" "<<rollno<<" "<<marks;

}

};

void main()

{

result obj;

clrscr();

obj.getname();

obj.getrollno();

obj.getmarks();

obj.output();

getch();

}

Single Inheritance Multiple Inheritence

A

B

A

C

B

MULTILEVEL HIERARCHIAL

A



C

B

A

B

C

Hybrid inheritence

C

D

A

B