**Unit-III: Inheritance, Virtual Functions and polymorphism**

Inheritance and the access specifies, Constructors and Destructors in derived classes, Multiple Inheritance, Passing parameters to a basic class, Pointers and references to derived types, Virtual Functions, Why virtual functions?, Pure virtual functions and abstract types, Early Vs Late binding.

**Unit-IV: Static & Dynamic memory allocation**

Static & Dynamic allocation using new and delete, static class members, Virtual base classes, Copy constructors,

//Multilevel inheritance

#include<iostream>

using namespace std;

class Student{

protected:

int roll\_no;

public:

void get\_number(int x)

{

roll\_no=x;

}

void put\_number()

{

cout<<"\n Rollno "<<roll\_no;

}

};

class Test:public Student

{

protected:

float sub1,sub2;

public:

void get\_marks(float a,float b)

{

sub1=a;

sub2=b;

}

void put\_marks()

{

cout<<"\n Subject1: "<<sub1;

cout<<"\n Subject2: "<<sub2;

}

};

class Result:public Test

{

protected:

float total,percent;

public:

void display()

{

total=sub1+sub2;

percent=total/2;

put\_number();

put\_marks();

cout<<"\n Total marks: "<<total<<"\t percent: "<<percent;

}

};

int main()

{

Result r;

r.get\_number(102);

r.get\_marks(68,80);

r.display() ;

}

//Multilevel inheritance

#include<iostream>

using namespace std;

class Student{

protected:

int roll\_no;

char name[30],branch[10],sec[5];

public:

void get\_data()

{

cout<<"\n Enter roll: ";

cin>>roll\_no;

cout<<"\n Enter name:";

cin>>name;

cout<<"\n Enter Branch: ";

cin>>branch;

cout<<"\n Enter section: ";

cin>>sec;

}

void put\_data()

{

cout<<"\n Roll: "<<roll\_no;

cout<<"\n Name: "<<name;

cout<<"\n Branch: "<<branch;

cout<<"\n Section: "<<sec;

}

};

class Test:public Student

{

protected:

float sub1,sub2,sub3,sub4,sub5;

public:

void get\_marks()

{

cout<<"\n Enter Maths marks:";

cin>>sub1;

cout<<"\n Enter OOPS marks:";

cin>>sub2;

cout<<"\n Enter DSC marks:";

cin>>sub3;

cout<<"\n Enter HWM marks:";

cin>>sub4;

cout<<"\n Enter POM marks:";

cin>>sub5;

}

void put\_marks()

{

cout<<"\n MATHS: "<<sub1;

cout<<"\n OOPS: "<<sub2;

cout<<"\n DSC: "<<sub3;

cout<<"\n HWM: "<<sub4;

cout<<"\n POM: "<<sub5;

}

};

class Sports

{

protected:

int sp\_marks;

public:

void sports\_marks()

{

cout<<"\n Enter sports marks: ";

cin>>sp\_marks;

}

void display\_sports()

{

cout<<"\n Sports marks: "<<sp\_marks;

}

};

class Result:public Test,public Sports

{

protected:

float total,percent;

public:

void display()

{

total=sub1+sub2+sub3+sub4+sub5+sp\_marks;

percent=(total)/6;

put\_data();

put\_marks();

cout<<"\n Total marks: "<<total<<"\t percent: "<<percent;

}

};

int main()

{

Result r;

r.get\_data();

r.get\_marks();

r.display\_sports();

r.display() ;

}

//multiple inheritance

#include<iostream>

using namespace std;

class M

{

protected:

int m;

public:

void get\_m(int x)

{

m=x;

}

};

class N

{

protected:

int n;

public:

void get\_n(int y)

{

n=y;

}

};

class P:public M,public N

{

public:

void display()

{

cout<<"\n M = "<<m;

cout<<"\n N = "<<n;

cout<<"\n Result = "<<m+n;

}

};

int main()

{

P obj;

obj.get\_m(20);

obj.get\_n(10);

obj.display();

}

//multiple inheritance

#include<iostream>

using namespace std;

class M

{

protected:

int m;

public:

void display(int x)

{

m=x;

}

};

class N

{

protected:

int n;

public:

void display(int y)

{

n=y;

}

};

class P:public M,public N

{

public:

int o;

void display(int z)

{

o=z;

cout<<"\n M = "<<m;

cout<<"\n N = "<<n;

cout<<"\n O = "<<o;

cout<<"\n Result "<<m\*n\*o;

}

};

int main()

{

P obj;

obj.M::display(8);

obj.N::display(10);

obj.display(2);

}

**#virtual class**

#include<iostream>

using namespace std;

class student

{

protected:

int roll\_no;

public:

void get\_number(int n)

{

roll\_no=n;

}

void put\_number()

{

cout<<"\n Roll no is "<<roll\_no;

}

};

class test:virtual public student

{

protected:

float sub1,sub2;

public:

void get\_marks(float s1,float s2)

{

sub1=s1;

sub2=s2;

}

void put\_marks()

{

cout<<"\n Marks obtained ";

cout<<"\n Subject1: "<<sub1;

cout<<"\n Subject2: "<<sub2;

}

};

class sports:virtual public student

{

protected:

int score;

public:

void get\_score(int s)

{

score=s;

}

void put\_score()

{

cout<<"\n Sportd score: "<<score;

}

};

class result:public test,public sports

{

protected:

float total;

public:

void display()

{

put\_number();

put\_marks();

put\_score();

total=sub1+sub2+score;

cout<<"\n Total marks "<<total;

}

};

int main()

{

result r;

r.get\_number(21);

r.get\_marks(60,70);

r.get\_score(50);

r.display();

}

**#abstract class**

#include<iostream>

using namespace std;

class base{

public:

int a;

virtual void show()=0;

};

class derived:public base

{

public:

void show()

{

cout<<"\n Implementing virtual function ";

}

};

int main()

{

base\*b1;

derived d;

b1=&d;

b1->show();

}

**#polymorphism**

#include<iostream>

using namespace std;

int main()

{

int a,\*ptr1,\*\*ptr2;

ptr1=&a;

ptr2=&ptr1;

cout<<"\n address of a "<<ptr1;

cout<<"\n address of ptr1 "<<ptr2;

cout<<"\n After incrementing the address value ";

ptr1+=2;

cout<<"\n Adress of a "<<ptr1;

ptr2+=2;

cout<<"\n Address of ptr1 "<<ptr2;

}

**#poly**

#include<iostream>

using namespace std;

int main()

{

int a=10,\*ptr;

ptr=&a;

cout<<"Value of a "<<a;

\*ptr=(\*ptr)/2;

cout<<"\n The value of a is "<<(\*ptr);

}

## #inheritance

#include<iostream>  
using namespace std;  
class B  
{  
    private:  
        int a;  
    public:  
        int b;  
        void get\_ab()  
        {  
            cout<<"enter a"<<endl;  
            cin>>a;  
            cout<<"enter b"<<endl;  
            cin>>b;  
        }  
        int get\_a()  
        {  
            return a;  
        }  
        void show\_a(void)  
        {  
            cout<<"enter a"<<endl;  
            cin>>a;  
        }  
};  
class D:public B  
{  
    private:  
        int c;  
    public:  
        void mul(void)  
        {  
            c=b\*get\_a();  
        }  
        void display(void)  
        {  
            cout<<"multiplication of a&b is:"<<c<<endl;  
        }  
};  
int main()  
{  
    D obj;  
    obj.get\_ab();  
    obj.show\_a();  
    obj.mul();  
    obj.display();  
    obj.b=20;  
    obj.mul();  
    obj.display();  
    return 0;  
}

#include<iostream>

using namespace std;

class b{

int a;

public:

int b;

void set\_ab();

int get\_a();

void show\_a();

};

class d:public b{

int c;

public:

void mul();

void display();

};

void b::set\_ab()

{

a=5;

b=20;

}

int b:: get\_a()

{

return a;

}

void b::show\_a()

{

cout<<"a="<<a;

}

void d::mul()

{

c=b\*get\_a();

}

void d::display()

{

cout<<"a="<<get\_a();

cout<<"b="<<b;

cout<<"c="<<c;

}

int main()

{

d d1;

d1.set\_ab();

d1.mul();

d1.show\_a();

d1.display();

d1.b=20;

d1.mul();

d1.display();

return 0;

}

#include<iostream>

using namespace std;

class b{

public:

int a;

int b;

void set\_ab();

int get\_a();

void show\_a();

};

class d:private b{

int c;

public:

void mul();

void display();

};

void b::set\_ab()

{

a=5;

b=20;

}

int b:: get\_a()

{

return a;

}

void b::show\_a()

{

cout<<"a="<<a;

}

void d::mul()

{

set\_ab();

c=b\*a;

}

void d::display()

{

cout<<"\n a="<<a;

cout<<"\n b="<<b;

cout<<"\n c="<<c;

}

int main()

{

b b1;

d d1;

b1.get\_a();

b1.set\_ab();

d1.mul();

b1.show\_a();

d1.display();

b1.b=20;

d1.mul();

d1.display();

return 0;

}