polymorphism (poly -> many
| morphism -> forms)

| **|**

compile time poly. **run time poly.**

| **|**

fun. overloading **operator overloading** **virtual function**

sum(int ,int) **+ , - , / ,....**

sum(float , float) **t = a + b; // int**

t = p + q; // complex no.

OVERLOADING :-

FUN. NAME --> SAME

ARGUMENT --> DIFFERENT

OVERRIDING :-

FUN. NAME --> SAME

ARGUMENT --> SAME

// COMPILE - TIME POLYMORPHISM
OR STATIC BINDING OR EARLIER BINDING

```
#include<iostream>
using namespace std;

  class  base
  {
        protected :  int  a , b;

        public      : void  get()
                      {
                            cout<<" ENTER NO "<<endl;
                            cin>> a >> b;
                      }
```

```
                    void out()
                    {
                            cout<< " a= " << a << endl;
                            cout<< " b= " << b << endl;
                    }
};

class  derived   : public base
{
            protected  : char  name[10];
                         int    roll;

            public      :
                  void   get()
                  {
                          cout << " enter name and roll " << endl;
                          cin >>  name >> roll;
                  }

                  void   out()
                  {
                        cout<< " NAME = " << name << endl;
                         cout<< " ROLL = "  << roll     << endl;
                  }
};

int    main()
{
```

```
        base  *p;
        base    b;
        p = &b;
        p -> get();
        p -> out(); // base -> a,b

       derived  d;

       p = &d;
       p -> get();
       p -> out();  //  base -> a , b
}

//       RUN - TIME POLYMORPHISM
  OR   DYNAMIC  BINDING          OR  LATE   BINDING

//   VIRTUAL  FUNCTION

#include<iostream>
using namespace std;

  class  base // manager  // gener
  {
        protected :  int  a , b;

        public     :  virtual  void   get()
                      {
```

```
                cout<<" ENTER NO "<<endl;
                cin>> a >> b;
            }
            virtual  void    out()
            {
                cout<< " a= " << a << endl;
                cout<< " b= " << b << endl;
            }
};

class  derived   : public base
{
        protected  : char  name[10];
                     int    roll;

        public      :
             void   get()
             {
                 cout << " enter name and roll " << endl;
                 cin >>  name >> roll;
             }
            void   out()
             {
                cout<< " NAME = " << name << endl;
                 cout<< " ROLL = "  << roll     << endl;
             }
};
int    main()
```

```
 {
        base  *p;
        base    b;
        p = &b;
        p -> get();
        p -> out(); // base -> a,b

       derived  d;

       p = &d;
       p -> get();
       p -> out();  //  derived -- > name , roll
 }
```

```
#include<iostream>
using namespace std;
class A
{
      public:
            virtual void disp()
            {
                  cout<<"\n Class A";
```

```
        }
};

class B:public A
{
    public:
        void disp()
        {
            cout<<"\n Class B";
        }
};

class C:public B
{
    public:
        void disp()
        {
            cout<<"\n Class C";
        }
};

int main()
{
    A ob1,*ptr;
    B ob2;
    C ob3;

    ptr=&ob1;
```

```
        ptr->disp();

        ptr=&ob2;
        ptr->disp();

        ptr=&ob3;
        ptr->disp();


}
```

// PURE VIRTUAL FUNCTION

```
#include<iostream>
using namespace std;

 class  graphics
 {
        public :

                virtual  void  draw()
                {
                        cout<< " graphics draw called " << endl;
                }

                virtual  void  show() = 0 ; // pure  virtual function
```

```
};

class  ract   :  public  graphics
{
        public :
                    void draw()
                    {
                            cout<< " ract draw called " << endl;
                    }
};



class square : public graphics
{

       public :
              void show()
              {
                     cout<< " square show called " << endl;
              }
};

int         main()
{
             graphics *p;

             ract  r;
```

```
        p = &r;
        p -> draw(); // ract draw called

        square s;

        p = &s;

        p -> draw();    // graphics draw called

        p -> show();    // square show called
}



/*
    virtual void show() = 0 ;

    ure virtual function -> ONLY declare , not define

    abstract base class   --> no object created

*/



#include<iostream>
using namespace std;
```

```
//      POINTER TO OBJECT

// INPUT  AND PRINT  TWO NOS

class test
{
      private :  int  a , b ;

      public :
            void  get()
            {
                cout<< " ENTER TWO NOS " << endl;

                cin >> a >> b;
            }

            void  out()
            {
                cout<< a << endl << b << endl;
            }
};


     // NORMAL   OBJECT

     int main()
     {
```

```
    test t;

    t . get();

    t . out();


}


// POINTER TO OBJECT

int main()
{


    test  t ;

    test *p; // declaration

    p = &t;  // initiazation

    p -> get();

    p -> out();
```

```
}



// DYNAMIC OBJECT

int main()
{

    test *p;

    p = new test;

    p -> get();

    p -> out();

    delete p;


}
// REFERENCE OBJECT

int main()
{
```

```
        test   t;

        test  &u = t;

        u. get();

        u. out();
    }
/*
```