LECTURE-11

STATIC DATA MEMBER:

A data member of a class can be qualified as static . The properties of a static member variable are similar to that of a static variable. A static member variable has contain special characteristics.

Variable has contain special characteristics:-

1) It is initialized to zero when the first object of its class is created.No other initialization is permitted.
2) Only one copy of that member is created for the entire class and is shared by all the objects of that class, no matter how many objects are created.
3) It is visible only with in the class but its life time is the entire program. Static variables are normally used to maintain values common to the entire class. For example a static data member can be used as a counter that records the occurrence of all the objects.

```
int item :: count; // definition of static data member
```

Note that the type and scope of each static member variable must be defined outside the class definition .This is necessary because the static data members are stored separately rather than as a part of an object.

Example :-

```
#include<iostream.h>
        class   item
        { static int count; //count is static
                int      number;
        public:
              void getdata(int a)
                  {
                        number=a;
                        count++;
                   }
               void getcount(void)
                {
                        cout<<"count:";
                        cout<<count<<endl;

        } };
int item :: count ; //count defined
        int main( )
```

```
                        {
                        item
                        a,b,c;
                        a.get_count( );
                        b.get_count( );
                        c.get_count( ):
                        a.getdata( ):
                        b.getdata( );
                        c.getdata( );
                      cout«"after reading data : "«endl;
                              a.get_count( );
                              b.gel_count( );
                              c.get count( );
                return(0);
                }
```

```
The output would
        be count:0
        count:0
        count:0
After  reading
        data
        count:
        3
        count:
        3
        count:
        3
```

The static Variable count is initialized to Zero when the objects created . The count is incremented whenever the data is read into an object. Since the data is read into objects three times the variable count is incremented three times. Because there is only one copy of count shared by all the three object, all the three output statements cause the value 3 to be displayed.

<u>**STATIC MEMBER FUNCTIONS:-**</u>

A member function that is declared static has following properties :-

1. A static function can have access to only other static members declared in the

same class.

2. A static member function can be called using the class name as follows:-
class - name :: function - name;

Example:-

```
            #include<iostream.h>
                    class test
                    { int code;
                            static  int  count; //  static  member  variable
                    public:
                           void set(void)
                           {
                           code=++count;
                           }
                           void showcode(void)
                           {
                              cout<<”object member : “<<code<<end;
                              }     static   void
                    showcount(void)
                              { cout<<”count=”<<count<<endl; }
                           };
    int          test::
    count;       int
    main()
    {
            test
            t1,t2;
            t1.setco
            de(      );
            t2.setco
            de( );
            test :: showcount ( );
            ' test t3; t3.setcode( );
            test:: showcount( );
            t1.showcode( ) ;
            t2.showcode( );
            t3.showcode( );
            return(0);
```

output:-

```
count : 2
  count:    3
object
number    1
```

object
number 2
object number 3

OBJECTS AS FUNCTION ARGUMENTS

Like any other data type, an object may be used as A function argument. This can cone in two ways 1. A copy of the entire object is passed to the function.

2. Only the address of the object is transferred to the function

The first method is called pass-by-value. Since a copy of the object is passed to the function, any change made to the object inside the function do not effect the object used to call the function. The second method is called pass-by-reference . When an address of the object is passed, the called function works directly on the actual object used in the call. This means that any changes made to the object inside the functions will reflect in the actual object .The pass by reference method is more efficient since it requires to pass only the address of the object and not the entire object.

Example:-

```
#include<iostream.h>
        class time
        { int hours;
              int minutes;
        public:
              void gettime(int h, int m)
              {
                              hours=h;
                              minutes=m;
                }
      void puttime(void)
              {
              cout<< hours<<"hours and:";

              cout<<minutes<<"minutes:"<<end;
              }
        void sum( time ,time);
}; void time :: sum (time
t1,time t2)     .
{    minutes=t1.minutes     +
t2.minutes;
hours=minutes%60;
minutes=minutes%60;
hours=hours+t
1.hours+t2.hours;
}

int main() {
time
T1,T2,T3;
T1.gettime(2
,45);
T2.gettime(3
,30);
T3.sum(T1,
T2);
cout<<”T1=
”;
T1.puttime(
);
cout<<”T2=
”;
T2.puttime(
);
cout<<”T3=
”;
T3.puttime(
); return(0);
}
```