Static Class Members

* The class members (data member and member functions) can be defined as static using the keyword 'static'.

(i) Static data members

- When a data member of a class is declared as static, it is shared by all objects of the class.
- All static data member is initialized to zero when the first object is created (if no other initialization is done).
- The static variable is redeclared and initialized outside the class using the scope resolution(::) operator to identify which class it belongs to.

Syntax: for defining static data member

```
Class classname
{
 .....
 static datatype var;
};
datatype classname :: var = Initial value ;
                            optional
```

- The static data member can be declared as private, public (or) protected inside the class. However, the static data members are global data.

Note: Usual method is to declare static data member as private in order to achieve effective data hiding.
↳ here, data member is accessed only through member functions.

(Eg)
```
#include <iostream.h>

class item
{
 static int count;
 int number;
 public:
  void getdata (int a)
   {
    number = a;
    count++;
   }
  void getcount ()
   {
    cout << "count : " << count ;
   }
};
int item:: count ; // initializes to 0 by default
void main()
{
 item a,b ; // count is initialized to zero
 a.getcount () ; // count : 0
 b.getcount () ; // count : 0
 a.getdata (10) ;
 a.getcount () ; //count : 1
 b.getdata (20) ;
 b.getcount () ; // count : 2
}
```

* In the above example, the static variable count is accessed by both objects a and b

* Public static data member.

- The public static data member can be accessed using
  - scope resolution operator (or)
  - objects with member access operator.

(Eg)

```
class test
{
 private:
   static int a;
 public:
   static int b;
};
 int test :: a;  //initializes to 0 by default.
 int test :: b=1;

void main( )
{
 test t1;
 cout << test :: a; // invalid  }  b'coz a is private member.
 cout << t1.a;      // invalid  }
 cout << test :: b;  // prints 1
 cout << t1.b; // prints 1
}
```

(ii) Static member functions

- When a member function of a class is declared as static, it is independent of any particular object of the class.
- A static member function can only access static data member and other static member functions.
- The static functions are accessed only using the class name and the scope resolution (::) operator (instead of its objects).

Syntax:

$\boxed{\text{Classname :: functionname();}}$

$\Downarrow$

fn.call

(Eg)

```
   class box
   {
   float length, breadth, height;
   public:
    static int count;
    static int getcount()
     {
      return count;
     }
    float volume(float l, float b, float h)
     {
     length= l; breadth =b; height =h;
     count ++;
     return length * breadth * height;
    }
   };
   int box :: count; // intializes to 0 by default.
void main()
{
 box b;
 cout<< "Value of count :"<< box :: count << box :: getcount(); //Prints 0 0
 cout<< b.volume(7.9, 20.8, 11.9) <<endl; // prints volume.
 cout<< "Value of count :"<< box :: getcount(); // Prints 1.
}
```