STATIC MEMBERS OF A C++ CLASS

We can define class members static using **static** keyword. When we declare a member of a class as static it means no matter how many objects of the class are created, there is only one copy of the static member.

A static member is shared by all objects of the class. All static data is initialized to zero when the first object is created, if no other initialization is present. We can't put it in the class definition but it can be initialized outside the class as done in the following example by redeclaring the static variable, using the scope resolution operator **::** to identify which class it belongs to.

Let us try the following example to understand the concept of static data members:

```
#include <iostream>

using namespace std;  class Box
{
   public:
      static int objectCount;
      // Constructor definition
      Box(double l=2.0, double b=2.0, double h=2.0)
      {
         cout <<"Constructor called." << endl;  length = l;
         breadth = b;
         height = h;
         // Increase every time object is created  objectCount++;
      }
      double Volume()
      {
         return length * breadth * height;
                                              }
                                          private:
      double length;     // Length of a box  double breadth; // Breadth of a box
      double height;     // Height of a box
};

// Initialize static member of class Box  int Box::objectCount = 0;

int main(void)
{
   Box Box1(3.3, 1.2, 1.5);    // Declare box1  Box Box2(8.5, 6.0, 2.0);  // Declare
   box2

   // Print total number of objects.
   cout << "Total objects: " << Box::objectCount << endl;

   return 0;
}
```

When the above code is compiled and executed, it produces the following result:

```
Constructor called.  Constructor called.  Total objects: 2
```

Static Function Members:

By declaring a function member as static, you make it independent of any particular object of the class. A static member function can be called even if no objects of the class exist and the **static** functions are accessed using only the class name and the scope resolution operator **::**.

A static member function can only access static data member, other static member functions and any other functions from outside the class.

Static member functions have a class scope and they do not have access to the **this** pointer of the class. You could use a static member function to determine whether some objects of the class have been created or not.

Let us try the following example to understand the concept of static function members:

```
#include <iostream>  using namespace std;  class Box
{
   public:
      static int objectCount;
      // Constructor definition
      Box(double l=2.0, double b=2.0, double  h=2.0)
      {
         cout <<"Constructor called." << endl;  length = l;
         breadth = b;
         height = h;
         // Increase every time object is created  objectCount++;
      }
      double Volume()
      {
         return length * breadth * height;
      }
      static int getCount()
      {
         return objectCount;
                                                  }
                                               private:




      double length;     // Length of a box
      double breadth;    // Breadth of a box
      double height;     // Height of a box
};

// Initialize static member of class Box  int
Box::objectCount = 0;

int main(void)
{

   // Print total number of objects before creating object.
   cout << "Inital Stage Count: " << Box::getCount() << endl;

   Box Box1(3.3, 1.2, 1.5);     // Declare box1
   Box Box2(8.5, 6.0, 2.0);     // Declare box2

   // Print total number of objects after creating object.  cout << "Final
   Stage Count: " << Box::getCount() << endl;

   return 0;
}
```

When the above code is compiled and executed, it produces the following result:

```
Inital Stage Count: 0  Constructor called.  Constructor called.
```

```
Final Stage Count: 2
```

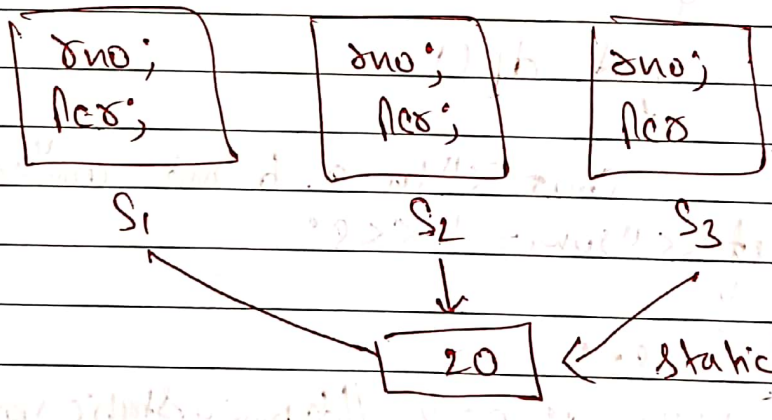
"Static Member And Static function"

Static means same for all objects.

Eg: In 70 object if all of them have one same information than we use static function.

```
class student
{
    int rno;        ] non-static
    float per;      ]
static int scc;     ] static.

    void ip()
    void op()
} s1, s2, s3;
    int student :: scc = 20;  { all student will have same section }.
```



A static function can have access to only static data member declared in same class. Can be called using class name instead of objects.

Note: Static Member function can be declared in class but not access in class. It can access outside the class

No.1 = Class :: function-name
No.2 = obj.f() <- by object

1)
```
#include<iostream>
using namespace std;
class addition
{
  public: int a,b,s;
  static int c;  // c is same for both s1 and s2
  void ip()
  {
    cout<<" Enter a,b ";
    cin>>a>>b;
  }
  void logic()
  {
    s= a+b+c;
  }
  void dp()
  {
    cout<<"In a, b and c are"<<" "<<b<<"
cout<<"sum= "<<s;
  }
} s1, s2;
int addition::c=2 //defining static variable c=2
  main()
{
  s1.ip(); s1.ip();
  s1.logic(); s2.logic()
  s1.dp(); s2.dp()
  }
```

```
11) class test
    {
      int code;
      static int count;
      public:
      void setcode()
        {
          code = ++count;
        }
      void showcode()
        {
          cout<<" Code : "<< code <<endl;
        }
      static void showcount()
        {
          cout<< " Count: "<< count <<endl;
        }
      int test:: count=10;
      int main()
      {
        test t1,&t2;
          t1.setcode();
          t2.setcode();
          test:: showcount();
          test t3;
          t3.setdata();
        test:: showCount();
          t1.showcode();
          t2.showcode();
          t3.showcode();
        }
```