Static Keyword in C++

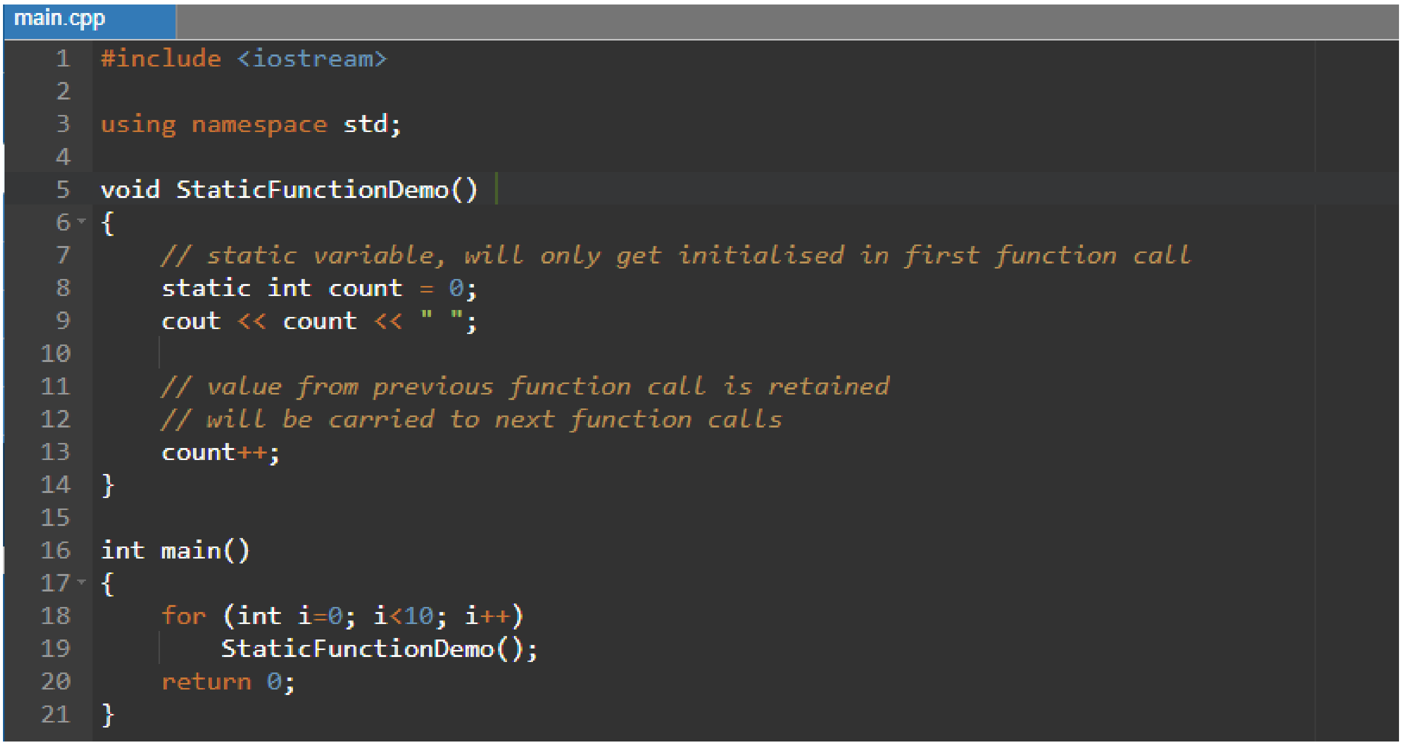
Static is a keyword in C++ used to give special characteristics to an element. Static elements are allocated storage only once in a program lifetime in static storage area. And they have a scope till the program lifetime. Static Keyword can be used with following,

1. Static variable in functions
2. Static member Variable in class
3. Static Class Objects
4. Static Methods in class

Static Variables inside Functions

If a function has a static variable (data member), then the variable gets allocated throughout the lifetime of program.

Calling the same function multiple times the variable will be allocated memory once only and value of the variable is retained in every function call for



```cpp
#include <iostream>

using namespace std;

void StaticFunctionDemo()
{
    // static variable, will only get initialised in first function call
    static int count = 0;
    cout << count << " ";

    // value from previous function call is retained
    // will be carried to next function calls
    count++;
}

int main()
{
    for (int i=0; i<10; i++)
        StaticFunctionDemo();
    return 0;
}
```

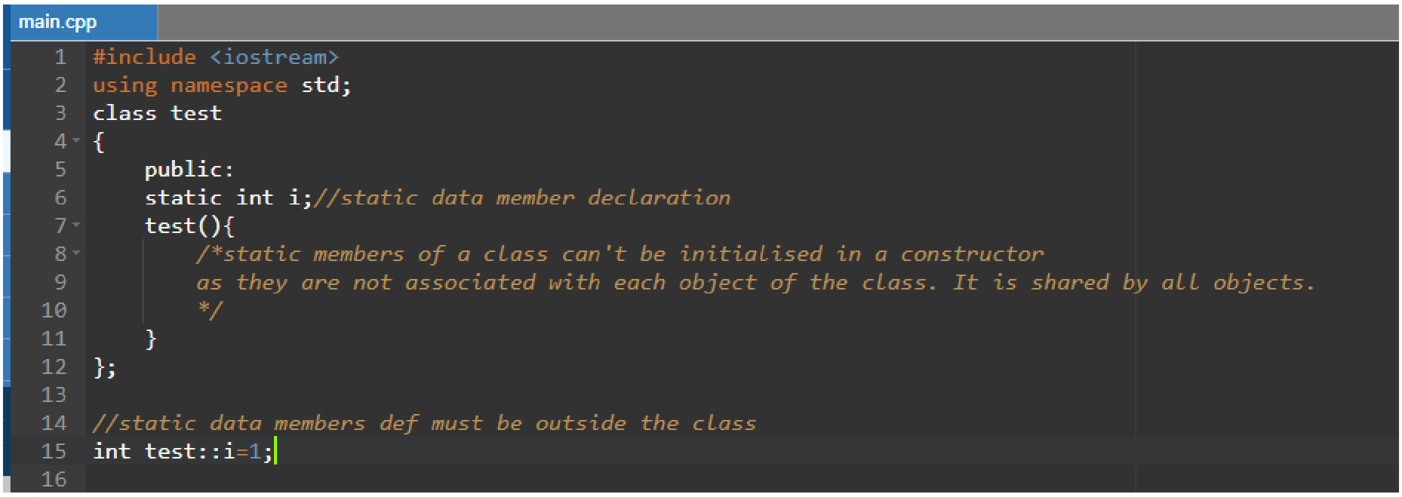
Static Data Member in Class

Generally, separate copies &memories of variables (data members) are created for each object for any given class.

However, When a variable (data member) is declared as ***static*** only a single copy of memory is created for all the objects & each object shares the same copy.

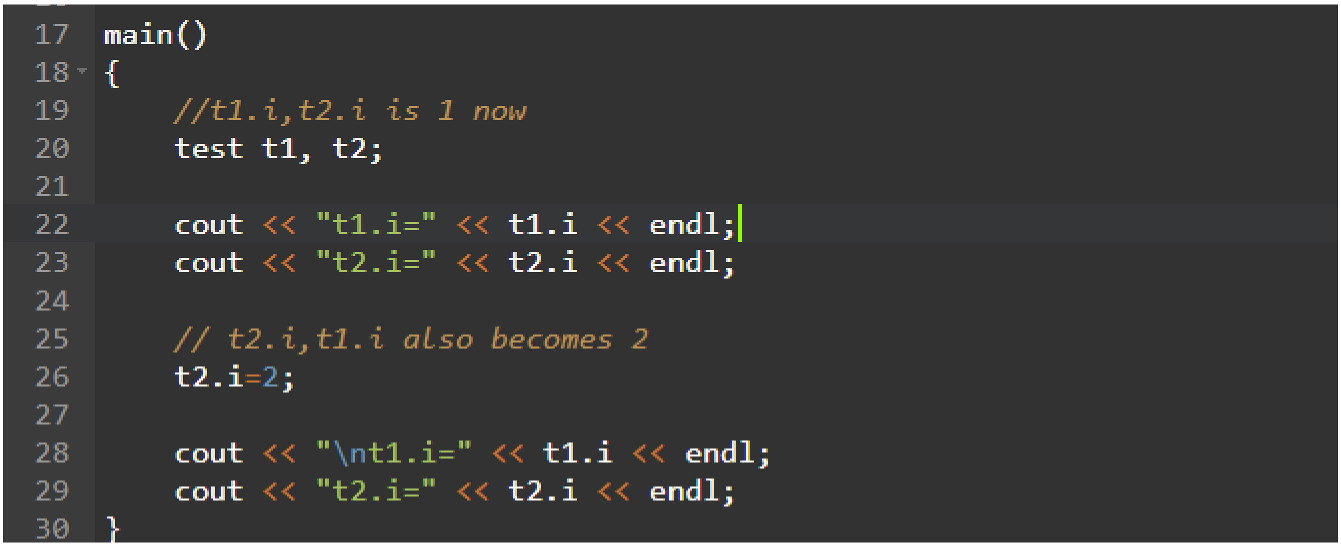
Static member variables (data members) are not initialied using constructor, because these are not dependent on object initialization.

Note – Static data members are ***class members not object member***

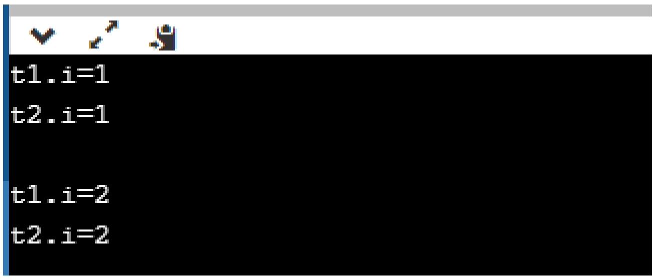


main.cpp

```
#include <iostream>
using namespace std;
class test
{
    public:
    static int i;//static data member declaration
    test(){
        /*static members of a class can't be initialised in a constructor
        as they are not associated with each object of the class. It is shared by all objects.
        */
    }
};

//static data members def must be outside the class
int test::i=1;

```



```
main()
{
    //t1.i,t2.i is 1 now
    test t1, t2;

    cout << "t1.i=" << t1.i << endl;
    cout << "t2.i=" << t2.i << endl;

    // t2.i,t1.i also becomes 2
    t2.i=2;

    cout << "\nt1.i=" << t1.i << endl;
    cout << "t2.i=" << t2.i << endl;
}
```



```
t1.i=1
t2.i=1

t1.i=2
t2.i=2
```

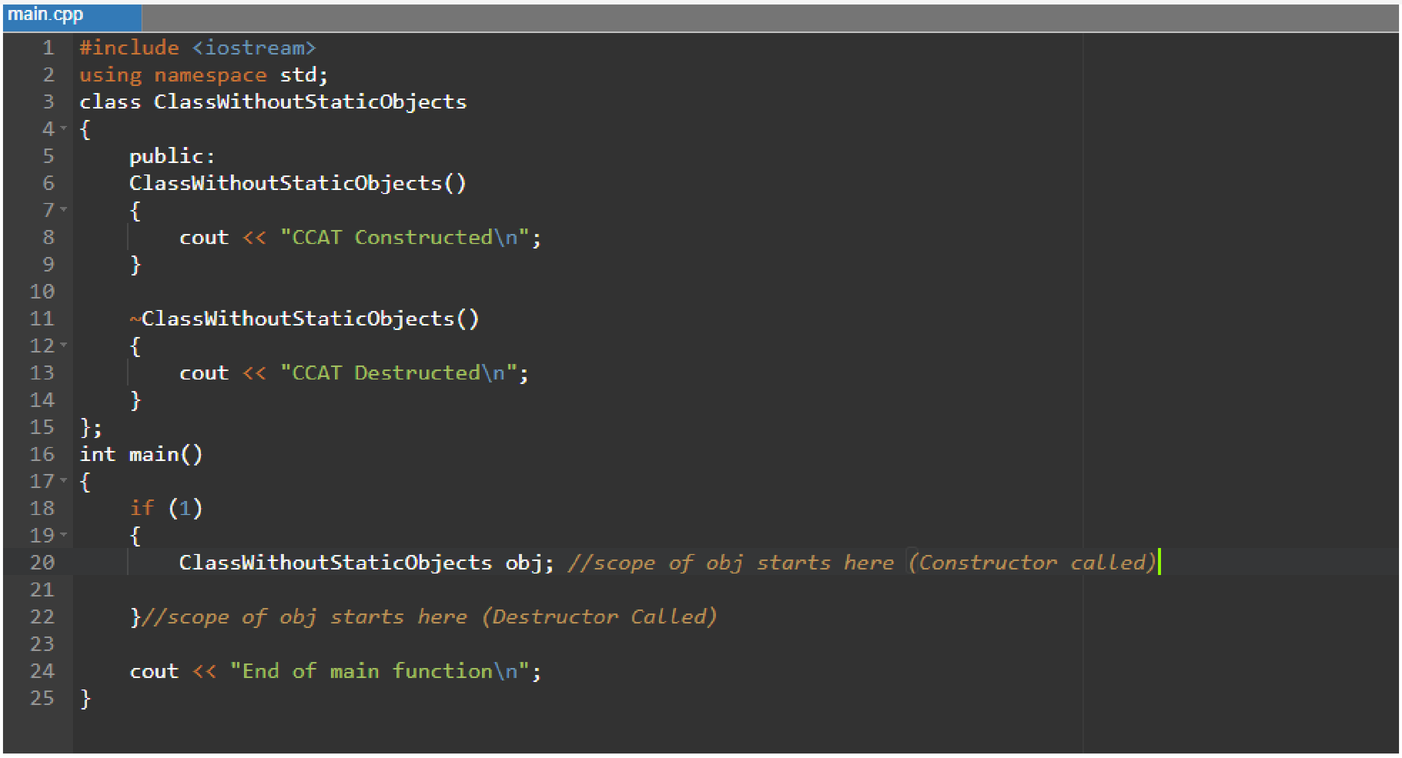
Basic Properties of Static data members

1. Static data members are ***initialized with zero automatically***
2. Static data members can be **called directly using the class name**

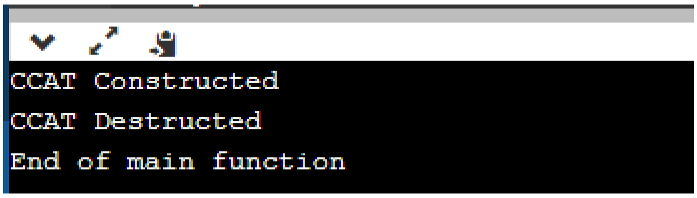
Static Class Objects

Very similar to static variables whose scope becomes till lifetime of the program, similarly, for static objects their scope becomes till just before the program terminates.

Constructor functioning without static objects

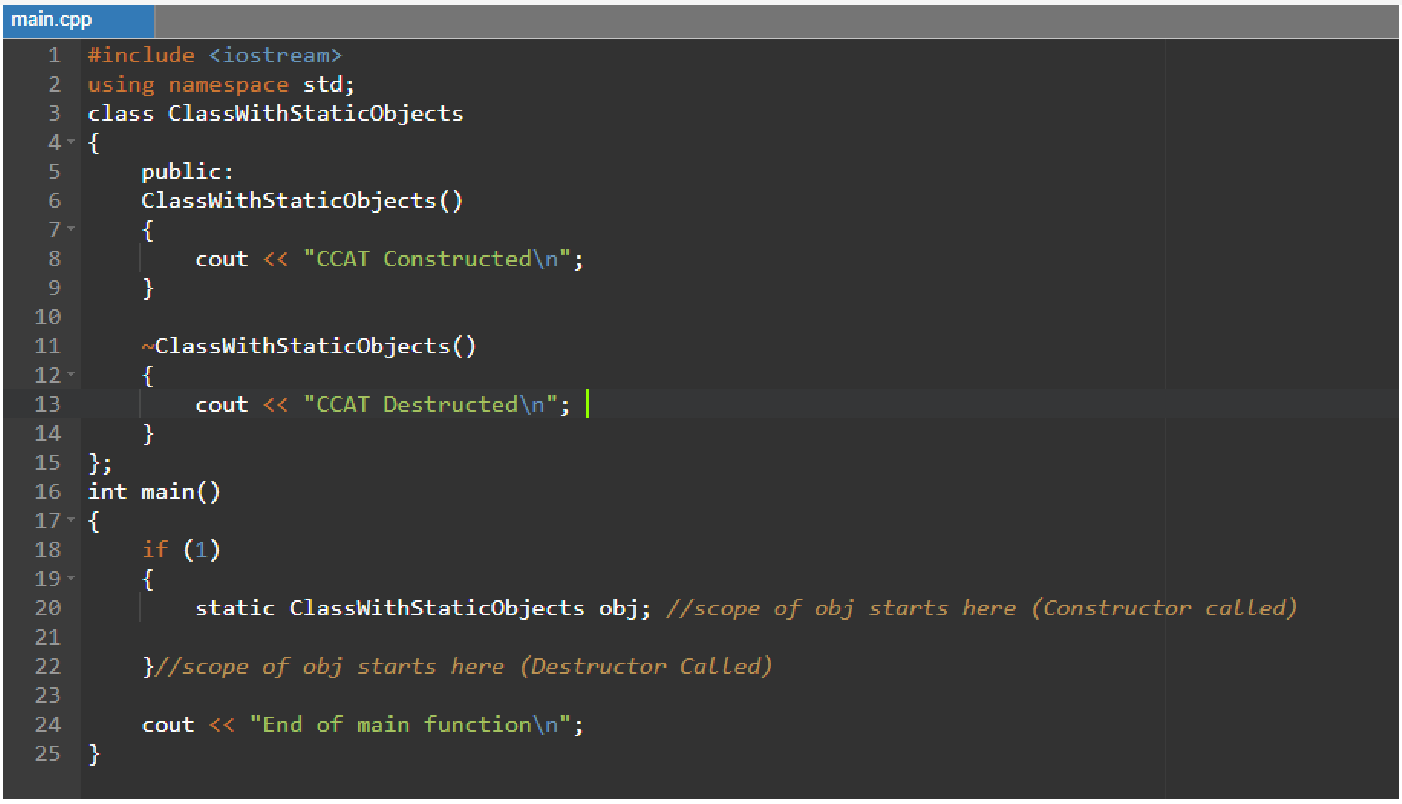


```cpp
#include <iostream>
using namespace std;
class ClassWithoutStaticObjects
{
    public:
    ClassWithoutStaticObjects()
    {
        cout << "CCAT Constructed\n";
    }

    ~ClassWithoutStaticObjects()
    {
        cout << "CCAT Destructed\n";
    }
};
int main()
{
    if (1)
    {
        ClassWithoutStaticObjects obj; //scope of obj starts here (Constructor called)

    }//scope of obj starts here (Destructor Called)

    cout << "End of main function\n";
}
```



```
CCAT Constructed
CCAT Destructed
End of main function
```

Constructor functioning with static objects



main.cpp

```cpp
#include <iostream>
using namespace std;
class ClassWithStaticObjects
{
    public:
    ClassWithStaticObjects()
    {
        cout << "CCAT Constructed\n";
    }

    ~ClassWithStaticObjects()
    {
        cout << "CCAT Destructed\n";
    }
};
int main()
{
    if (1)
    {
        static ClassWithStaticObjects obj; //scope of obj starts here (Constructor called)

    }//scope of obj starts here (Destructor Called)

    cout << "End of main function\n";
}
```

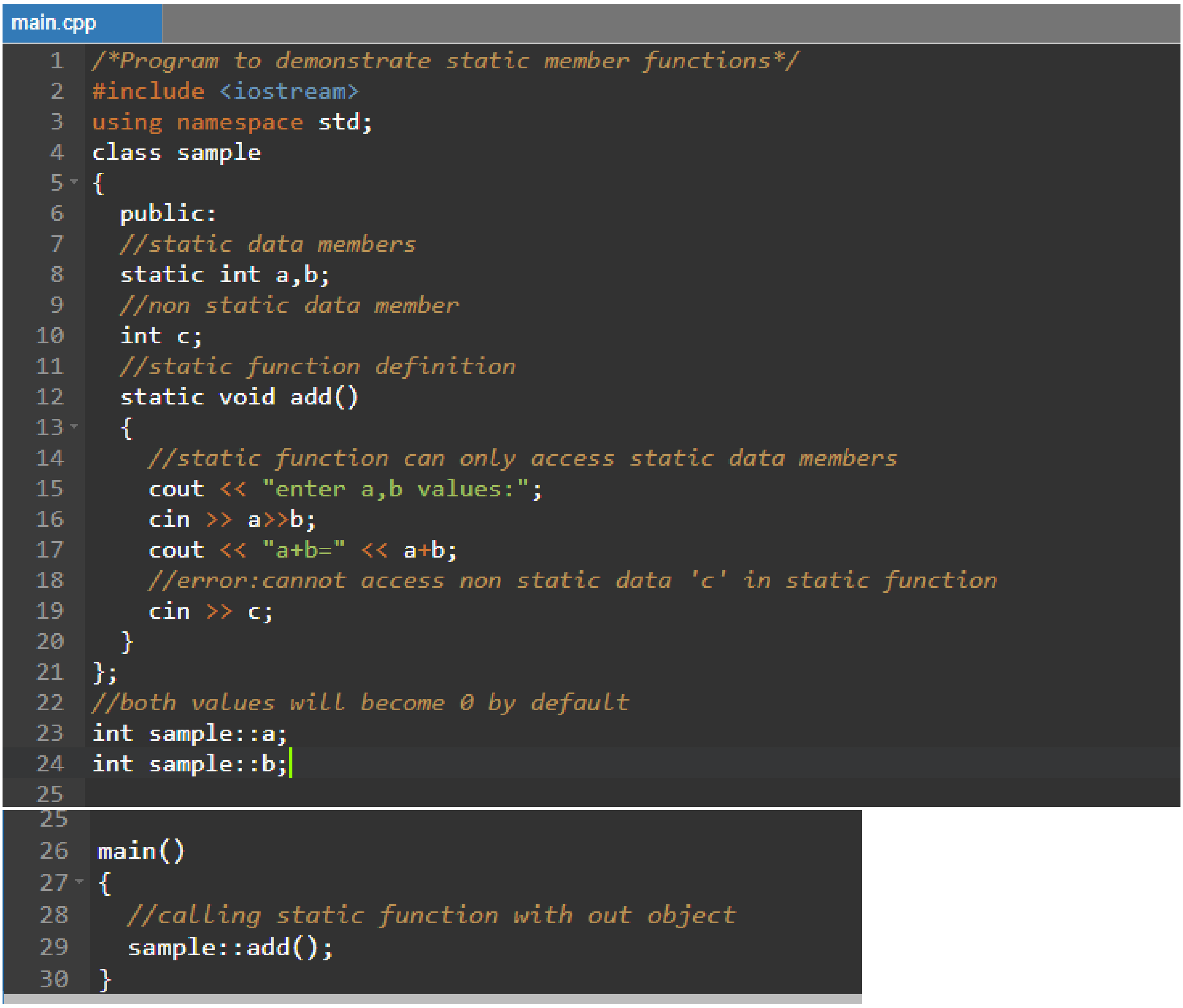
When static keyword is added to object, then rather than the scope of object being limited to the enclosed brackets it is in, the scope becomes till the program terminates.

Static Member functions

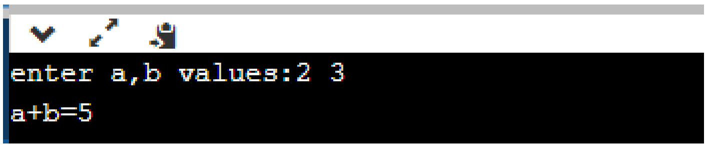
Definition: A static member function is **designed only for accessing static data members**,

other static member functions and any other functions from outside the class.

- A static member function can be called ***even if no objects of the class exist***
- Static member functions have a class scope and do not have access to this pointer of the class.
- Static function can only access static members of class



```cpp
/*Program to demonstrate static member functions*/
#include <iostream>
using namespace std;
class sample
{
  public:
  //static data members
  static int a,b;
  //non static data member
  int c;
  //static function definition
  static void add()
  {
    //static function can only access static data members
    cout << "enter a,b values:";
    cin >> a>>b;
    cout << "a+b=" << a+b;
    //error:cannot access non static data 'c' in static function
    cin >> c;
  }
};
//both values will become 0 by default
int sample::a;
int sample::b;

main()
{
  //calling static function with out object
  sample::add();
}
```



```
enter a,b values:2 3
a+b=5
```

Facts about static members / methods

- Static data members can be accessed by other methods too
- Overloading static methods is possible
- Non-static data members can not be accessed by static methods.
- Static methods can only access static members (data and methods)