## Scope Resolution Operator

The scope resolution operator is a unique operator that is used to **access variables and functions** that exist within a **specific scope**. In C++, the scope resolution operator is represented by two colons (::) and is placed **in front of the name** of the **variable or function** that is being accessed.

The scope resolution operator helps us in understanding the scope of variables and functions.

```
#include <iostream>
using namespace std;
int x = 20; // Global variable
int main() {
   int x = 10; // Local variable
   cout << "Local x: " << x <<  endl;
    cout << "Global x: " << ::x <<   endl;    // Accessing global 'x'
   return 0;    }
```

Output:

Local x: 10

Global x: 20

The (dot, .) is used to access members of an object, the (double colon, ::) is used to access members of a namespace or a class.

The :: (scope resolution) operator is used to **qualify hidden names** so that we can still use them. We can use the unary scope operator if a namespace scope or global scope name is hidden by an explicit declaration of the same name in a block or class.

We can also use the **class scope operator** to qualify **class names** or **class member names**. If a class member name is **hidden**, we can use it by qualifying it with its **class name** and the **class scope** operator.

```
#include <iostream>
using namespace std;
class X    {
public:
  static int count;    };
int X::count = 10;              // define static data member
int main ()   {
  int X = 0;       // hides class type X
  cout << X::count << endl;     // use static member of class X   }
```

**Namespaces**

A namespace is an optionally named scope. We declare names inside a namespace as we would for a class or an enumeration.

A namespace is a declarative region that provides a scope to the identifiers (the names of types, functions, variables, etc) inside it. Namespaces are used to organize code into logical groups and to prevent name collisions that can occur especially when your code base includes multiple libraries. All identifiers at namespace scope are visible to one another without qualification.

The following example shows a namespace declaration and three ways that code outside the namespace can access its members (microsoft.com).

```
namespace ContosoData    {
  class ObjectManager   {
  public:    void DoSomething() {}
                                  };
  void Func(ObjectManager) {   }
                                   } // end of namespace !!
```

**Use the fully qualified name:**

```
ContosoData::ObjectManager mgr;
mgr.DoSomething();
ContosoData::Func(mgr);
```

**Use a using declaration to bring one identifier into scope:**

```
using ContosoData::ObjectManager;
ObjectManager mgr;
mgr.DoSomething();
```

**Use a using directive to bring everything in the namespace into scope:**

```
using namespace ContosoData;
ObjectManager mgr;
mgr.DoSomething();
Func(mgr);
```