Scope Resolution Operator

Branch: S6 ECE

Faculty: SREEDIVYA

Local and global variable

- The scope of the variable extends from the point of its declaration till the end of the block containing the declaration.
- **Local** variable- a variable declared inside the block
- **Global** variable – a variable declared outside the block

SCOPE RESOLUTION OPERATOR

- **Scope resolution operator** is used to uncover a hidden variable
- This operator allows access to a **global** variable when there is a local variable with the same name
- Used to define a function outside a class
- It can be used to access **static members** when there is a local variable with same name
- Used in the case of multiple inheritance that is, if same variable name exists in the ancestor classes

Syntax:

::variable_name;

Simple example showing the usage of ::

```
#include<iostream>
using namespace std;
int m=10;

int main()
{
  int m = 20;
        {
                int k=m;
                int m=30;
                cout<<"\nINNER BLOCK\n";
                cout<<"k="<<k<<"\n";
                cout<<"m="<<m<<"\n";
                cout<<"::m="<<::m<<"\n";
        }
  cout<<"\nOUTER BLOCK\n";
  cout<<"m="<<m<<"\n";
  cout<<"::m="<<::m<<"\n";

  return 0;
}
```

OUTPUT

INNER BLOCK

k=20

m=30

::m=10

OUTER BLOCK

m=20

::m=10

1. To access a global variable when there is a local variable with same name

```
#include<iostream>
using namespace std;
int x=5; // Global x

int main()
{
int x = 10; // Local x
cout << "Value of global x is " << ::x;
cout << "\nValue of local x is " << x;
return 0;
}
```

Output

```
Value of global x is 5
Value of local x is 10
```

2. To define a function outside a class

```
#include <iostream>
using namespace std;

class A
{
    public:
       void fun();
};

void A::fun()
{
    cout << "fun() called";
}

int main()
{
    A a;
    a.fun();
    return 0;
}
```

OUTPUT

```
fun() called
```

3. To access a class's static variables

```
#include<iostream>
using namespace std;
class Test
{
    static int x;
public:
    static int y;

    void func(int x)
    {
    cout << "Value of static x is " << Test::x;
    cout << "\nValue of local x is " << x;
    }
};
int Test::x = 1; //In C++, static members are explicitly defined like this
int Test::y = 2;
int main()
{
    Test obj;
    int x = 3 ;
    obj.func(x);
    cout << "\nTest::y = " << Test::y;
    return 0;
}
```

OUTPUT

Value of static x is 1

Value of local x is 3

Test ::y = 2

4. In case of Multiple Inheritance

```
#include<iostream>
using namespace std;

class A
{
protected:
    int x;
public:
    void A()
      {
        x = 10;
      }
};

class B
{
protected:
    int x;
public:
    void B()
    {
      x = 20;
    }
};
```

```
class C: public A, public B
{
public:
void fun()
{
    cout << "A's x is " << A::x;
    cout << "\nB's x is " << B::x;
}
};

int main()
{
    C c;
    c.fun();
    return 0;
}
```

OUTPUT

```
A's x is 10
B's x is 20
```