Scope resolution operator in C++

In C++, scope resolution operator is **::**. It is used for following purposes.

**1) To access a global variable when there is a local variable with same name:**

|  |
| --- |
| // C++ program to show that we can access a global variable  // using scope resolution operator :: when there is a local  // variable with same name  #include<iostream>  using namespace std;    int x;  // 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 0

Value of local x is 10

**2) To define a function outside a class.**

|  |
| --- |
| // C++ program to show that scope resolution operator :: is used  // to define a function outside a class  #include<iostream>  using namespace std;    class A  {  public:       // Only declaration     void fun();  };    // Definition outside class using ::  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.**

|  |
| --- |
| // C++ program to show that :: can be used to access static  // members when there is a local variable with same name  #include<iostream>  using namespace std;    class Test  {      static int x;  public:      static int y;        // Local parameter 'a' hides class member      // 'a', but we can access it using ::      void func(int x)      {         // We can access class's static variable         // even if there is a local variable         cout << "Value of static x is " << Test::x;           cout << "\nValue of local x is " << x;      }  };    // In C++, static members must be explicitly defined  // like this  int Test::x = 1;  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:**  
If same variable name exists in two ancestor classes, we can use scope resolution operator to distinguish.

|  |
| --- |
| // Use of scope resolution operator in multiple inheritance.  #include<iostream>  using namespace std;    class A  {  protected:      int x;  public:      A() { x = 10; }  };    class B  {  protected:      int x;  public:      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

**5) For namespace**  
If a class having the same name exists inside two namespace we can use the namespace name with the scope resolution operator to refer that class without any conflicts

|  |
| --- |
| // Use of scope resolution operator for namespace.  #include<iostream>      int main(){      std::cout << "Hello" << std::endl;    } |

Here, cout and endl belong to the std namespace.

**6) Refer to a class inside another class:**  
If a class exists inside another class we can use the nesting class to refer the nested class using the scope resolution operator

|  |
| --- |
| // Use of scope resolution class inside another class.  #include<iostream>  using namespace std;    class outside  {  public:        int x;        class inside        {        public:              int x;              static int y;              int foo();          };  };  int outside::inside::y = 5;    int main(){      outside A;      outside::inside B;    } |

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above