10 Name Control

Hu Sikang
skhu@163.com

School of Computer
Beijing Institute of Technology

Contents

- Static variables
- **Namespace**
- Static member

Namespaces

- Although names can be nested inside classes, the names of global functions, global variables, and classes are still in a single global name space.
- The static keyword gives you some control over this by allowing you to give variables and functions internal linkage. But in a large project, lack of control over the global name space can cause problems.
- You can subdivide the global name space into more manageable pieces using the *namespace* feature of C++.

1. Creating a namespace

```
//MyLib.cpp
namespace MyLib
{
   // members
}

void main()
{ }
```

Differences from class:

- It can only appear at global scope, or nested within another namespace.
- **“;”** is not necessary after the closing brace.
- The name MyLib can be used in multiple header.
- The name can be *aliased* to another name:

  namespace **Lib** = **MyLib;**
- You cannot create an instance of a namespace.

2. Scope resolution

```
//ScopeResolution.cpp
namespace DB
{
    class SQL
    {
     static int i;
    public:
     void g(int) { }
    };
    class EXCEL;
    void GetDBType( );
}

int DB::SQL::i = 9;
```

```
class DB:: EXCEL
{
    int u, v, w;
public:
    EXCEL (int i);
    int g();
};
DB::EXCEL::EXCEL(int i) { u=v=w=i; }
int DB::EXCEL::g() { return w; }
void DB::GetDBType()
{
     DB::SQL  a;     // object
     a.g(1);
}
void  main()  {  DB::GetDBType();  }
```

3. Using directive

```
namespace calculator  {
        double Add(double x, double y) { return x + y; }
        void Print(double x) { cout << x << endl; }
        class Shape  {   };
}
calculator :: Shape   S1;            // Define object with namespace
using namespace calculator;     // Using Directive
void main( )  {
        Shape   S2;
        double a, b;
        cin >> a >> b;
        double = Add(a, b));
}
```