

xLib.h

```
1  #ifndef XLIB_H
2  #define XLIB_H
3
4  // introducing xLib as a namespace
5  namespace xLib {
6
7      int doSomething(int);
8      const int minimum = 100;
9  }
10
11 // ...
12
13 // expanding xLib again
14 namespace xLib {
15     class Foo
16     {
17     private: int a;
18     public: Foo(int);
19             void print() const;
20     };
21 }
22 #endif
```

xLib.cpp

```
23 #include<iostream>
24 using std::cout;
25 using std::endl;
26 #include "xlib.h"
27
28 // expanding xLib again
29 namespace xLib {
30     int doSomething(int m) {
31         cout << "doSomething in xLib" << endl;
32         return minimum - m;
33     }
34     Foo::Foo(int a) : a(a) { }
35 }
36
37 // ...
38
39 // expanding xLib again
40 namespace xLib {
41     void Foo::print() const { cout << "xLib::Foo::a " << a << endl; }
42 }
```

yLib.h

```
43 #ifndef YLIB_H
44 #define YLIB_H
45
46 // introducing yLib as a namespace
47 namespace yLib {
48     int doSomething(int);
49 }
50 // ...
51 // expanding yLib again
52 namespace yLib {
53     const int minimum = 200;
54 }
55 // ...
56 // expanding yLib again
57 namespace yLib {
58     class Foo
59     {
60     private: int a;
61     public: Foo(int);
62             void print() const;
63     };
64 }
65 #endif
```

yLib.cpp

```
66 #include<iostream>
67 using std::cout;
68 using std::endl;
69 #include "ylib.h"
70
71 // expanding yLib again
72 namespace yLib {
73     int doSomething(int m) {
74         cout << "doSomething in yLib" << endl;
75         return minimum - m;
76     }
77 }
78
79 // ...
80 // expanding yLib again
81 namespace yLib {
82     Foo::Foo(int a) : a(a) { }
83     void Foo::print() const { cout << "yLib::Foo::a " << a << endl; }
84 }
```

xyLib_Test_Driver.cpp

```
85 #include<iostream>
86 #include "ylib.h"
87 #include "xlib.h"
88
89 int main() {
90
91     {// a local scope
92         using namespace xLib; // the using directive makes all the names available.
93         std::cout << doSomething(10) << std::endl;
94         std::cout << minimum << std::endl;
95     }
96
97     {// another local scope
98         using yLib::doSomething; // A using declaration, makes a single name available.
99         std::cout << doSomething(20) << std::endl;
100
101         using yLib::minimum; // A using declaration, makes a single name available.
102         std::cout << minimum << std::endl;
103     }
104
105     xLib::Foo f1(1000); // no ambiguity when you use the scope-resolution operator
106     f1.print();
107
108     yLib::Foo f2(2000); // no ambiguity when you use the scope-resolution operator
109     f2.print();
110
111     return 0;
112 }
```

Output

```
113 doSomething in xLib
114 90
115 100
116 doSomething in yLib
117 180
118 200
119 xLib::Foo::a 1000
120 yLib::Foo::a 2000
```