

# namespace

[http://blog.csdn.net/yao\\_zhuang/article/details/1853625](http://blog.csdn.net/yao_zhuang/article/details/1853625)

namespace中文意思是命名空间或者叫名字空间，传统的C++只有一个全局的namespace，但是由于现在的程序的规模越来越大，程序的分工越来越细，全局作用域变得越来越拥挤，每个人都可能使用相同的名字来实现不同的库，于是程序员在合并程序的时候就会可能出现名字的冲突。namespace引入了复杂性，解决了这个问题。namespace允许像类，对象，函数聚集在一个名字下。本质上讲namespace是对全局作用域的细分。

我想大家都见过这样的程序吧：

hello\_world.c

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{  
    printf("hello world !");  
    return 0;  
}
```

我想很多人对namespace的了解也就这么多了

但是namespace远不止如此，让我们再多了解一下namespace

namespace的格式基本格式是

```
namespace identifier
```

```
{  
    entities;  
}
```

举个例子，

```
namespace exp
```

```
{  
    int a,b;  
}
```

有点类似于类，但完全是两种不同的类型。

为了在namespace外使用namespace内的变量我们使用::操作符，如下

```
exp::a
```

```
exp::b
```

使用namespace可以有效的避免重定义的问题

```
#include <iostream>
```

```
using namespace std;
```

```
namespace first
```

```
{  
    int var = 5;  
}
```

```
namespace second
```

```
{  
    double var = 3.1416;  
}
```

```
int main () {
```

```
cout << first::var << endl;
cout << second::var << endl;
return 0;
}
```

结果是

5

3.1416

两个全局变量都是名字都是var，但是他们不在同一个namespace中所以没有冲突。

关键字using可以帮助从namespace中引入名字到当前的声明区域

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

```
namespace first
```

```
{
    int x = 5;
    int y = 10;
}
```

```
namespace second
```

```
{
    double x = 3.1416;
    double y = 2.7183;
}
```

```
int main () {
    using first::x;
    using second::y;
    cout << x << endl;
    cout << y << endl;
    cout << first::y << endl;
    cout << second::x << endl;
    return 0;
}
```

输出是

5

2.7183

10

3.1416

就如我们所指定的第一个x是first::x， y是second.y

using也可以导入整个的namespace

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

```
namespace first
```

```
{
    int x = 5;
    int y = 10;
}
```

```
namespace second
```

```

{
    double x = 3.1416;
    double y = 2.7183;
}

int main () {
    using namespace first;
    cout << x << endl;
    cout << y << endl;
    cout << second::x << endl;
    cout << second::y << endl;
    return 0;
}

```

输出是

```

5
10
3.1416
2.7183

```

正如我们所预见的导入的整个的first的namespace，前一对x,y的值就是first中的x, y的值。

这里我们不能在“using namespace first;”下加一句“using namespace second;”，为什么呢？

这样做无异于直接完全的忽视namespace first和namespace second，会出现重复定义的结果，所以前面的hello\_world.c中的using指令的使用一定程度上存在问题的，只是因为我们就用了一个namespace，一旦引入了新的namespace这种做法很可能会出现重复定义的问题。

在头文件中，我们通常坚持使用显式的限定，并且仅将using指令局限在很小的作用域中，这样他们的效用就会受到限制并且易于使用。

类似的例子有

```

#include <iostream>
using namespace std;

```

```

namespace first
{
    int x = 5;
}

```

```

namespace second
{
    double x = 3.1416;
}

```

```

int main () {
    {
        using namespace first;
        cout << x << endl;
    }
    {
        using namespace second;
        cout << x << endl;
    }
    return 0;
}

```

输出是

```

5
3.1416

```

可以看到两个不同的namespace都被限制在了不同作用域中了，他们之间就没有冲突。

namespace也支持嵌套

```
#include <iostream>
```

```
namespace first
```

```
{  
    int a=10;  
    int b=20;
```

```
    namespace second
```

```
{  
    double a=1.02;  
    double b=5.002;  
    void hello();
```

```
}
```

```
void second::hello()
```

```
{  
    std::cout <<"hello world"<<std::endl;  
}
```

```
}
```

```
int main()
```

```
{  
    using namespace first;
```

```
    std::cout<<second::a<<std::endl;  
    second::hello();
```

```
}
```

输出是

1.02

hello world

在namespace first中嵌套了namespace second，second并不能直接使用，需要first来间接的使用。

namespace可以使用别名，在对一些名字比较长的namespace使用别名的话，是一件很惬意的事。但是与using相同，最好避免在头文件使用namespace的别名（f比first更容易产生冲突）。

```
namespace f = first;
```

最后，namespace提供了单独的作用域，它类似于静态全局声明的使用，可以使用未命名的namespace定义来实现：

```
namespace { int count = 0;}    //这里的count是唯一的  
                               //在程序的其它部分中count是有效的  
void chg_cnt (int i) { count = i; }
```

参考资料：

《C++必知必会》 [美] Stephen C.Dewhurst 著，荣耀译 条款23名字空间

<http://www.cplusplus.com/doc/tutorial/namespaces.html>

《C++精粹》 [美] Ira Pohl 著 王树武 陈朔鹰 等译 P15