1. （p340）一个名字空间形成一个作用域，通常的作用域规则也适用于名字空间。因此，名字空间是一个非常基础、非常简单的概念。程序规模越大，用名字空间表达程序的逻辑划分就越有用。全局作用也也是一个名字空间，可以显示地用::来引用。例如：

int f(); //全局函数

int g()

{

int f; //**局部变量；屏蔽了全局函数**

f(); //错误：不能调用一个整型变量

::f(); //正确：调用全局函数

}

**类也是名字空间（见16.2节）**

1. using声明

当我们需要名字空间外频繁使用其名字时，反复使用名字空间进行显示限定很繁琐。

std::vector<std::string> splict(const std::string& s )

{

std::vector<std::string> res;

std::istringstream iss(s);

for(std::string buf; iss>>buf)

res.push\_buck(buf);

return res;

}

我们可以使用using声明来指出这段代码中string表示std::string:

using std::string; //用string表示std::string

std::vector<string> splict(conststring& s )

{

std::vector<string> res;

std::istringstream iss(s);

for(string buf; iss>>buf)

res.push\_buck(buf);

return res;

}

using声明将一个代用名引入了作用域，最好尽量保持代用名的局部性以避免混淆。

当用于一个重载的名字时，using声明会应用于其所有重载版本。例如：

namespace N {

void f(int);

void f(string);

};

void g()

{

using N::f;

f(789); //N::f(int)

f(“Bruce”); //N::f(string)

}

有关在类层次中使用using声明的内容，请见20.3.5节。

1. using指示（p341）

作为替代方法，我们可以使用using指示，要求编译器允许我们在所在作用域中无需使用限定符即可访问某个名字空间的所有名字，例如：

using namespace std; //令来自std的每个名字都可访问

vector<string> splict( const string& s )

{

vector<string> res;

istringstream iss(s);

for(string bug, iss<<buf;)

res.push\_back(buf);

return res;

}

（p342）除了极特殊情况外（例如为了帮助代码转换），不要在头文件中将一个using指示置于全局作用域中，因为你永远也不知道头文件可能在哪里被#include

1. 参数依赖查找（p342）



