Module03-06
C++ 语言基础：名字空间

语言基础 - 名字空间

- 名字空间 (namespace)
  - 关于名字空间
  - 名字的限定
  - using 指令
  - 名字空间的多处定义
  - 避免名字冲突
  - 名字空间的别名
  - 名字空间的组合

名字空间 - 关于名字空间

- 关于名字空间
  - 广义上讲，所有的作用域：全局作用域或任何局部作用域，都可以描述为名字空间
  - 一个名字空间就是一个命名或匿名的作用域，通过名字空间将一些相关的名字逻辑上归为一组，这样一来这些名字处于同一个作用域之下
  - 名字空间主要用于减少名字冲突，另一个作用是将项目模块化表示

名字空间 - 关于名字空间

- 示例:

```
namespace ns1 {
class Matrix;
Matrix operator+(const Matrix& m1, const Matrix& m2);
}

namespace ns2 { // 尽管下面声明的名字与上面相同，但没有冲突
class Matrix;
Matrix operator+(const Matrix& m1, const Matrix& m2);
}

int main() {
    ns1::Matrix m1, m2, m3;
    // ...
    m3 = m1 + m2;
}
```

名字空间 - 名字的限定

- 使用名字空间中的名字
  - 要访问不同名字空间下的名字，需冠以该名字空间的名称：
    - 如访问 namespace ns1 下的 Matrix： ns1::Matrix

名字空间 - using 指令

- using 的常用方式：
  - 打开名字空间
    - `using namespace space-name`
  - 使用名字空间下指定的名字：
    - `using space-name::name`
- using 指令的问题：
  - 在一个大的作用域中使用 using 指令，会把该名字或该名字空间下的所有名字暴露，削弱名字保护的功能

名字空间 - using 指令

- 示例：

```
namespace ns1 {
const int SIZE = 64;
class A;
void func1();
}

namespace ns2 {
enum { SIZE = 512 };
const int A = 8;
void func1();
}

int main() {
    using ns2::SIZE;
    int a[SIZE];

    using ns1::A;
    A a;
    ns1::func1();
}
```

```
namespace ns1 {
const int SIZE = 64;
class A;
void func1();
}

namespace ns2 {
enum { SIZE = 512 };
const int A = 8;
void func1();
}

int main() {
    using namespace ns1;
    using namespace ns2;

    int a[SIZE]; // ?? ambiguous
    A a;  // ?? ambiguous
    func1(); // ?? ambiguous
}
```

名字空间 - 名字空间的多处定义

- 名字空间的定义可以出现多次：
  - 在同一个文件中定义多次名称相同的名字空间
  - 或在不同文件中各自定义相同名称的名字空间
  - 注意：同一处定义一样：分开定义的同名名字空间中不能有相同名字的成员
- 示例

```
// file: a.h
namespace ns1 {
const int SIZE = 64;
class A;
}

// file: b.h
namespace ns1 {
void func2();
}
```

using

```
// c.cpp
#include "a.h"
#include "b.h"
// other codes...
```

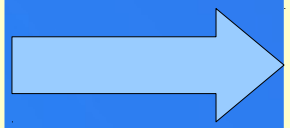
merged

```
namespace ns1 {
const int SIZE = 64;
class A;
void func2();
}
```

名字空间 - 避免名字冲突

- 将名字纳入不同的名字空间，减少冲突的几率

```
// a.h
const int SIZE = 64;
class A;
void func1();

// b.h
enum { SIZE = 512 };
const int A = 8;
void func1();

// c.cpp
// 试想以下情形:
#include "a.h"
#include "b.h"
// other codes;
```



```
// a.h
namespace nsa {
const int SIZE = 64;
class A;
void func1();
}

// b.h
namespace nsb {
enum { SIZE = 512 };
const int A = 8;
void func1();
}

// c.cpp
#include "a.h"
#include "b.h"
// other codes;
```

名字空间 - 避免名字冲突

- 使用无名名字空间
  - 无名名字空间：就是定义一个名字空间，但该名字空间没有指定名字，如： `namespace { const int M = 128; }`
  - 无名名字空间将名字的作用域局限于各个编译单元 (如 .cpp 文件)，所以任意两个编译单元的无名名字空间都是不同的
  - 无名名字空间可以替代不同编译单元中全局 static 名字的方案
  - 示例 (DEMO)

名字空间 - 名字空间的别名

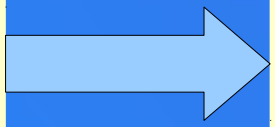
- 别名的定义

```
namespace graphic {
class GraphicObject;

namespace g3d {
class Curve3d;
}
}

int main() {
    namespace g3d = graphic::g3d;  // 定义 graphic::g3d 的别名
    g3d::Curve3d c3d;
}
```

名字空间 - 名字空间的组合

- 在新名字空间中组合已有的名字空间

```
namespace ns1 {
int i;
int n;
}

namespace ns2 {
using ns1::n;
int k;
int i;
}
```



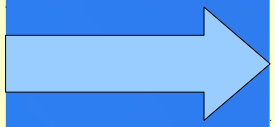
```
namespace ns3 {
using namespace ns1;
using namespace ns2;

class A;
}
```

```
int main() {
    cout << ns3::n << endl; // OK
    cout << ns1::n << endl; // OK
    cout << ns2::n << endl; // OK
    cout << ns3::i << endl; // ??
}
```

名字空间 - 名字空间的组合

- 解决组合后可能的名字使用中的二义性

```
namespace ns1 {
int i;
int n;
}

namespace ns2 {
using ns1::n;
int k;
int i;
}
```



```
namespace ns3 {
using namespace ns1;
using namespace ns2;

using ns2::i;

class A;
}
```

```
int main() {
    cout << ns3::n << endl; // OK
    cout << ns1::n << endl; // OK
    cout << ns2::n << endl; // OK
    cout << ns3::i << endl; // OK
    cout << ns1::i << endl;
}
```

名字空间 - Bjarne's Advices

- Bjarne's Advices
  - 使用名字空间表示逻辑上的结构
  - 除 main() 函数外，将所有非局部名字纳入合适的名字空间
  - 设计一个名字空间以方便访问，且不会意外的访问到其它无关的名字空间
  - 避免名字很短小的名字空间
  - 避免给使用名字空间的用户平添沉重的记法上的负担
  - 必要的情况下，使用名字空间别名来缩短很长的名字空间的名字
  - 记得使用 Namespace::member 的方式定义成员 member
  - 仅在临时使用或局部作用域使用 using namespace xxx 的方式