volatile

`volatile`： 易变的，被修饰的变量禁止编译器做任何的优化
应用场景：主要用于**多线程**环境

namespace命名空间

解决命名问题

前缀：解决命名冲突，但也会使得名字冗长且难记

命名空间

- 定义命名空间

```
// 定义my_space命名空间
namespace my_space {
    int a = 10;
    void foo()
    {
        ...
    }
    ...
}
```

- 使用命名空间

```
// 使用my_space命名空间中的成员
// 方案一
my_space::a = ...;
my_space::foo();

// 方案二
using namaspace my_space;   // 会受作用域影响
a = ...;
foo();

// 方案三
using my_space::foo;
using my_space::a;
foo();
a = ...;
```

- 全局命名空间

```
// 以::标识，前面不加任何东西
::CreateFile(...);
```

- 命名空间取别名

```
namsapce mysp = my_space;
mysp::foo();
mysp::a = ...;
```

C++11 —— auto与decltype

auto

`auto`为自动类型推导，在**C++14**中废除了自动变量的功能
允许定义一个变量，不指定变量类型，由编译器根据初始值推断变量的类型

```
int num = 10;
auto a = num;

double foo(double num)
{
    num = ...;
    return num;
}
auto b = foo(3.14);
```

decltype(expression)

表达式推导，主要用于推导函数的返回值类型，且表达式不会被执行

```
int a = 1;
double b = 3.14;
decltype(a + b) c = a + b;
```

C++14开始`auto`可以推导函数返回值