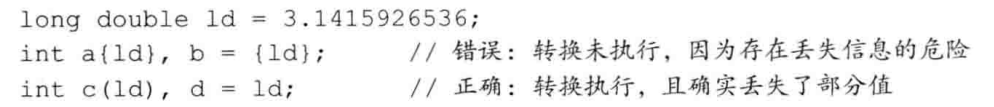
# C++11新特性

## 1 long long

## 2列表初始化。

用花括号初始化变量。列表初始化可以防止初始值存在丢失的情况。



## 3 nullptr

原本NULL被定义为((void\*)0)会混淆0和空指针。

void func(int a);

void func(int\* a);

func(NULL);

nullptr 的类型为nullptr\_t，能够隐式 的转换为任何指针或成员指针的类型，也能和他们进⾏相等或者不等的⽐较。

## 4 constexpr

用于确保用常量初始化一个常量。防止出现不确定因素。下面的会报错。

const int b = a + 1;

constexpr int c = a + 1;

该关键字也可用于修饰函数。要求函数形参、返回值类型必须是字面值类型且函数有且只有一条return语句。

特殊的，该关键字常用于用于修饰字面值常量类的构造函数。函数体为空。

## 5 using别名

using起别名包含了typdef的所有功能。

当typedef复合类型时，可能会和预期不一样。typedef int \*ip;

typedef不能定义一个模板的别名。往往需要新增一个包裹类。

template <typename T>

struct alias\_map // 包裹类

{

typedef std::map<std::string, T> map;

};

alias\_map<int>::map map\_t;

alias\_map<int>::map map\_str;

## 6 auto

auto会忽略顶层const。

## 7 decltype

会保留引用和顶层const。可以用作函数的返回值。

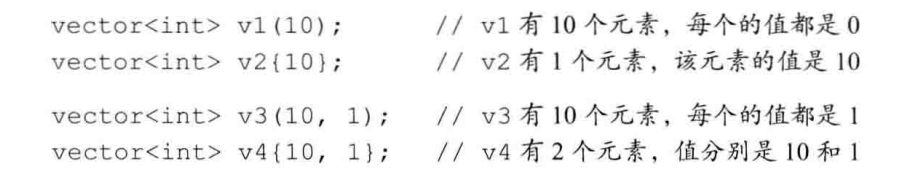
## 8 类内初始化

减少构造函数的代码量。

## 9 基于范围的for

分为两部分：第一部分是范围内用于迭代的变量，第二部分则表示被迭代的范围。

## 10 vector列表初始化和元素数量



## 11 cbegin

begin返回的具体类型由对象是否是常量决定。与begin不同的是cbegin返回的始终是const\_iterator可以保证不被修改。

## 12 标准库的begin

int arr[3]{ 1,2,3 };

vector<int>vec(begin(arr), end(arr));

c++数组属于内置类型，不属于容器。

## 13 除法

c++早期版本，允许结果为负值的商向上或者向下取整。c++11一律采取切除小数部分的方案。

## 14 sizeof用于类成员

获取某个类成员大小，不需要真的创建一个对象才可以知道。

## 15 尾置返回类型

用于函数返回类型比较复杂的情况。

int(\*func(int i))[10];

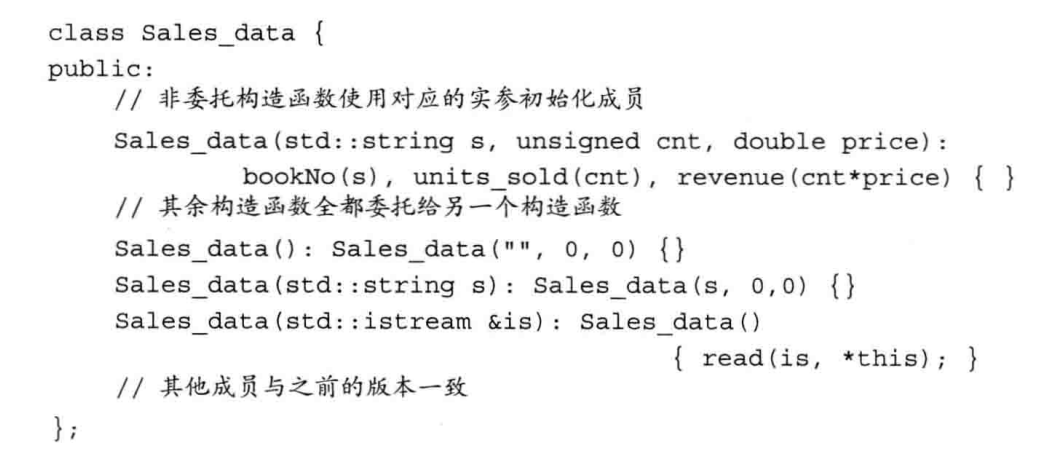
auto func(int i)->int(\*)[10];

## 16 利用=default生成默认构造函数

前提是类内已经对成员变量初始化。

## 17 委托构造函数

使用它所属类的其他构造函数执行初始化过程。可以传递委托。



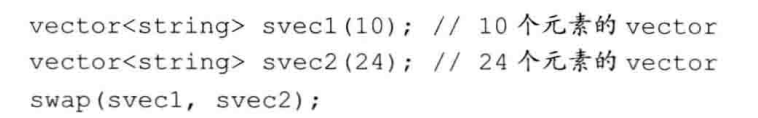
## 18 forward\_list和array

forward\_list是单项链表，list是双向链表。某些情况下前者性能更优。

array。是容器，对应于C语言数组，大小固定。与内置数组相比，优点是更安全、更容易使用。

## 19 容器的swap操作

交换两个相同类型容器的内部数据结构。除array外，不对任何元素进行拷贝、插入、删除操作。因此可以保证常数时间内完成。



## 20 使用insert的返回值

通过使用insert的返回值，可以在容器某一特定位置反复插入元素。

