Week5作业：阅读C++ prime plus 第9~10 章并总结关键内容

第九章 内存模型和名称空间

1. 单独编译

为减少程序出现的问题，可以将原来的程序分成三部分：头文件（包含结构声明和使用这些结构的函数的原型）、源代码文件（包含与结构有关的函数的代码）、 源代码文件（包含调用与结构相关的函数的代码）。

注意：不要将函数定义或变量声明放到头文件中。在包含头文件时，使用“coordin.h”，而不是<coodin.h>。另外，不要使用#include来包含源代码文件，这样做将导致多重声明。

1. 存储持续性、作用域和链接性
2. 作用域和链接

作用域（scope）描述了名称在文件（翻译单元）的多大范围内可见。链接性（linkage）描述了名称如何在不同单元间共享。

1. 自动存储持续性

在两个不同函数中声明的变量只有在函数自身内部有效。代码块中定义的变量也一样。

1. 关键字register最初是由C语言引入的，它建议编译器使用CPU寄存器来存储自动变量：register int coun\_fast;
2. 静态持续变量

C++也为静态存储持续性变量提供了3种链接性：外部链接性（可在其他文件中访问）、内部链接性（只能在当前文件中访问）和无链接性（只能在当前函数或代码块中访问）。



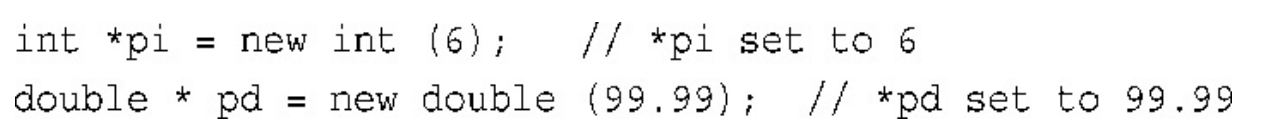
（5）函数和链接性

C++不允许在一个函数中定义另外一个函数，因此所有函数的存储持续性都自动为静态的，即在整个程序执行期间都一直存在。

1. new

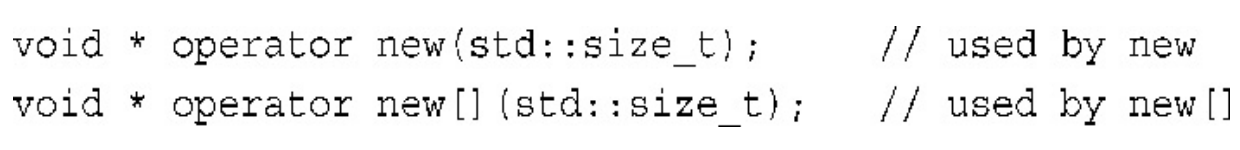
动态内存由运算符new和delete控制，而不是由作用域和链接性规则控制。

1. 使用new运算符初始化



要初始化常规结构或数组，需要使用大括号的列表初始化。

1. 运算符new和new []分别调用如下函数：



1. 定位new运算符

通常，new负责在堆（heap）中找到一个足以能够满足要求的内存块。它还有另一种变体，被称为定位（placement）new运算符，它让您能够指定要使用的位置。要使用定位new特性，首先需要包含头文件new。除需要指定参数外，句法与常规new运算符相同。

1. 名称空间
2. 传统C++名称空间

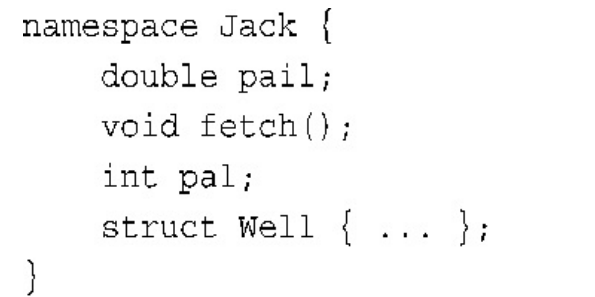
声明区域（declaration region）：是可以在其中进行声明的区域。

潜在作用域（potential scope）：变量的潜 在作用域从声明点开始，到其声明区域的结尾。因此潜在作用域比声明区域小，这是由于变量必须定义后才能使用。

1. 新的名称空间特性

通过定义一种新的声明区域来创建命名的名称空间，以提供一个声明名称的区域。一个名称空间中的名称不会与另外一个名称空间的相同名称发生冲突，同时允 许程序的其他部分使用该名称空间中声明的东西。

例如，使用namespace创建名称空间的做法：



using声明由被限定的名称和它前面的关键字using组成：

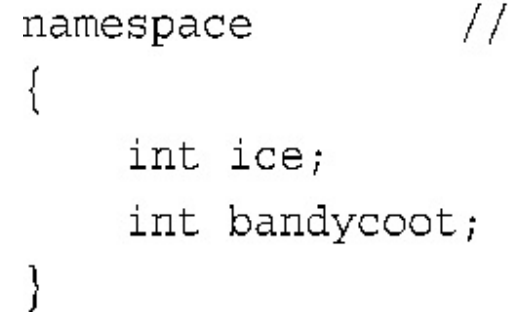
如using Jill::fetch;

1. 名称空间的其他特性

可以将名称空间声明进行嵌套，也可以在名称空间中使用using编译指令和using声明。

1. 未命名的名称空间

可以通过省略名称空间的名称来创建未命名的名称空间：



1. 对象和类

面向对象编程（OOP）是一种特殊的、设计程序的概念性方法。

最重要的OOP特性：抽象、封装和数据隐藏、多态、继承、代码的可重用性。

采用OOP方法时，首先从用户的角度考虑对象。

1. 抽象

抽象是通往用户定义类型的捷径，用户定义类型指的是实现抽象接口的类设计。

1. 指定基本类型完成了三项工作： 决定数据对象需要的内存数量；决定如何解释内存中的位；决定可使用数据对象执行的操作或方法。
2. C++中的类

类是一种将抽象转换为用户定义类型的C++工具，它将数据表示和操纵数据的方法组合成一个整洁的包。

1. 类规范由两个部分组成。

类声明：以数据成员的方式描述数据部分，以成员函数（方法）的方式描述公有接口。

类方法定义：描述如何实现类成员函数。

类定义示例：（详见week7/practice1）

class name

{

private://私有成员，不能直接访问

int x;

public://公有部分，可以直接访问

int y;

void show();//也可以定义函数

};//别忘了分号

使用类对象的程序都可以直接访问公有部分，但只能通过公有成员函数（或友元函数，参见第11章）来访问对象的私有成员。通常，数据成员放在私有部分中，成员函数放在公有部分中。

1. 实现类成员函数

成员函数定义与常规函数定义很相似，都有函数头和函数体，也可以有返回类型和参数。但是除此之外还有两个特殊的特征：定义成员函数时，使用作用域解析运算符（::）来标识函数所属的类；类方法可以访问类的private组件。

1. 类的构造函数和析构函数

在创建对象时对它进行初始化，例如：

Stock gift;

Gift.buy(10, 24.75);

1. 定义构造函数

示例：

Stock::Stock(const string&co,long n,double pr)

{

...

}

构造函数的参数表示的不是类成员，而是赋给类成员的值,所以参数名不能与类成员相同。

1. 使用构造函数

显式地调用构造函数：

Stock food=Stock(“World Cabbage”,250,1.25);//将food对象的company成员设置为字符串“World Cabbage”，将 shares成员设置为250，依此类推。

隐式地调用：

Stock garment(“Furry Mason”,50,2.5);

1. 默认构造函数

当且仅当没有定义任何构造函数时，编译器才会提供默认构造函数。但通常应初始化所有的对象，以确保所有成员一开始就有已知的合理值。

1. 析构函数

析构函数的名称：在类名前加上~，比如Stock类的析构函数为~Stock( )。与构造函数不同的是，析构函数没有参数，因此Stock析构函数的原型必须是：~Stock();

Q：何时调用析构函数？

A：这由编译器决定，通常不应在代码中显式地调用析构函数。

\*头文件定义格式例：

#ifndef STOCK10\_H\_

#define STOCK01\_H\_

#include<string>

Class Stock

{

...

} ;

#endif

1. this指针

指向用来调用成员函数的对象。每个成员函数都有一个this指针。this指针指向调用对象。如果方法需要引用整个调用对象，则可以使用表达式\*this。在函数的括号后可用const限定符将this限定为const，这样将不能使用this来修改对象的值。

1. 对象数组

声明对象数组的方法（与声明标准类型数组相同）：

Stock mystuff[4];

1. 类作用域

在类中定义的名称（如类数据成员名和类成员函数名）的作用域都为整个类，只在该类中是已知的，在类外是不可知的。

1. 作用域为类的常量

在类中定义常量的方式：使用关键字static

例如 class Bakery

{

private:

static const int Months =12;

double const[Months];

...

};

1. 作用域内枚举

C++11提供了一种新枚举，其枚举量的作用域为类。举例如下：

enum class egg{Small,Medium,Large,Jumbo};

enum class t\_shirt{Small,Medium,Large,Jumbo};//也可以用struct代替class。

这样他们就不会因为位于相同作用域而冲突。