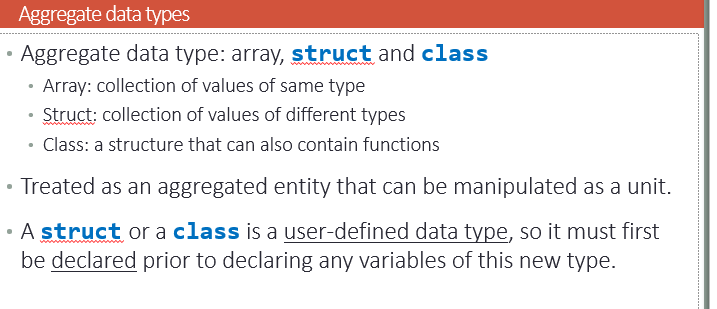
Aggregate data type聚合数据类型

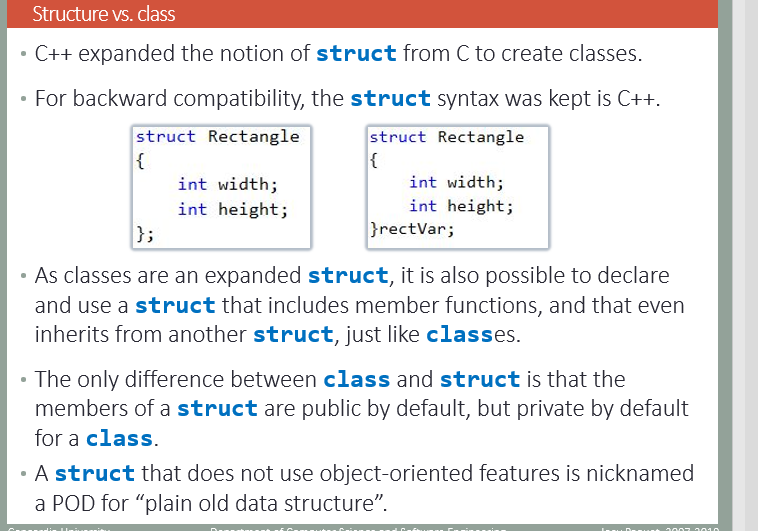


Array包含一组相同type的value

Struct:包含了一组不同type的value //struct也可以包含function，一般不这么做

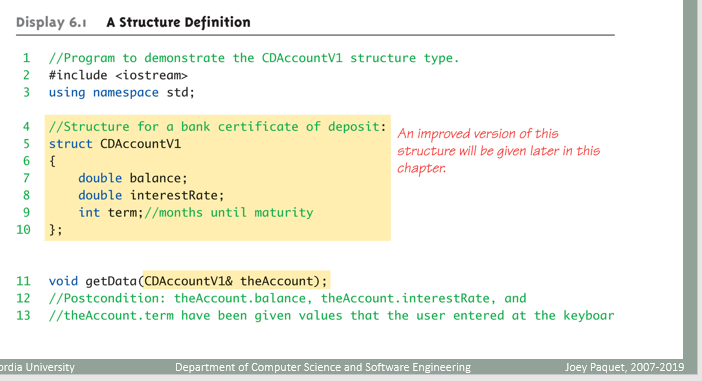
Class:一个结构可以包含function

Struct与class是user-defined数据类型，你在声明任意这个类型的variable的时候你必须先自己描述这个class/struct



C++在C上struct的概念建立了class，然后struct也是可以在C++使用的

Struct与与class的区别在于struct默认public，class默认private

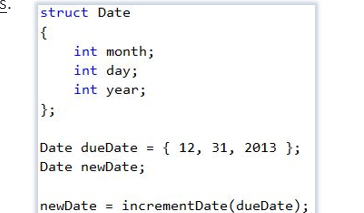


Struct用途就是将一组数据绑在一起形成新type，尽管他也可以写function,但如果写function我们更倾向于使用class

我们要用struct里的元素就直接使用.就行了，

例如 theAccount.balance

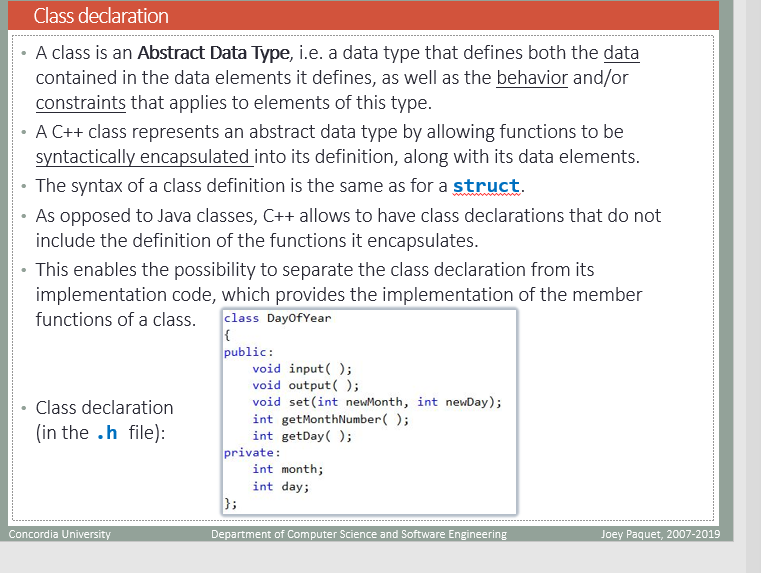
structure可以被初始化，等于，作为参数传递或者pointer



但是如果structure包含了一个pointer variable，那么你必须要你必须要为这个值额外allocation,deallocation

例如，如果你让创造了 object 1与object2， 然后让object1=object2，实际上你只是把pointer地址发送过去了，而没有copy被指的value，

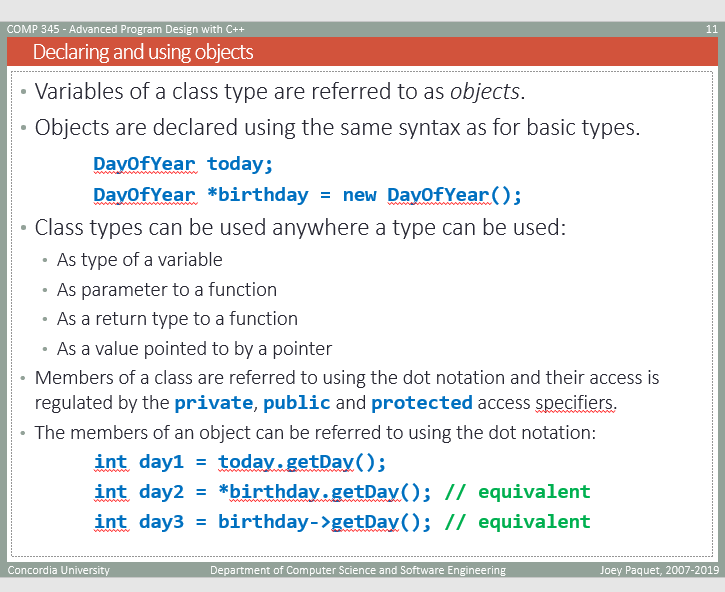
Class



Class是个抽象数据类型，

你可以写参数，behavior（method）等

Declare要写在h里，



Class形式的变量就是object

声明变量的方法

普通的就很简单

用指针声明就用Pointer

普通的变量用.method就行

指针的可以\*name. 因为\*name就是展开

或者直接对pointer使用->

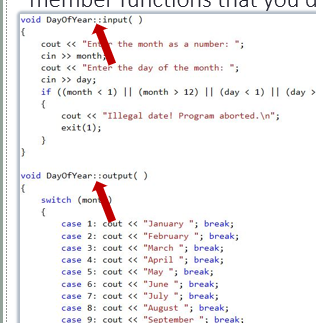
Class type可以用作variable type,一个function的参数，一个function的return type，一个被pointer所指的value

如果你declare的是today，那么你就可以直接getday

而birthday是一个Pointer,他point一个new的东西

所以要用\*.getday来指代实 际的东西

用双冒号scope resolution operator来指定你这个member function(作为一个class里的一个member)属于哪个class



一个设计良好的program：

.h :Header file : 用来declare class

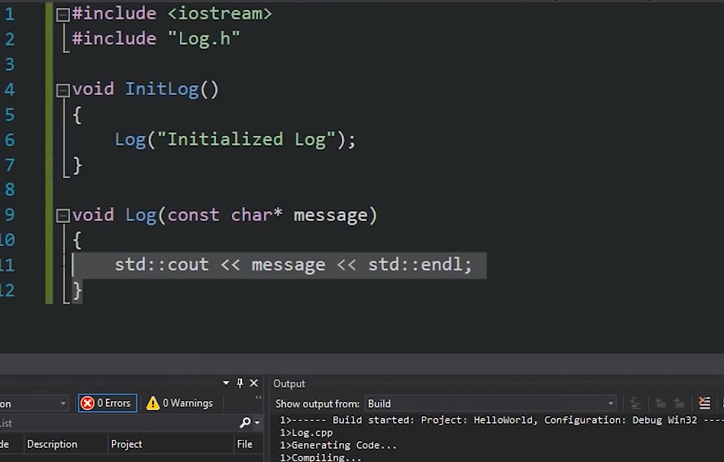
.cpp : Implementation file：用来Implement所有的member function

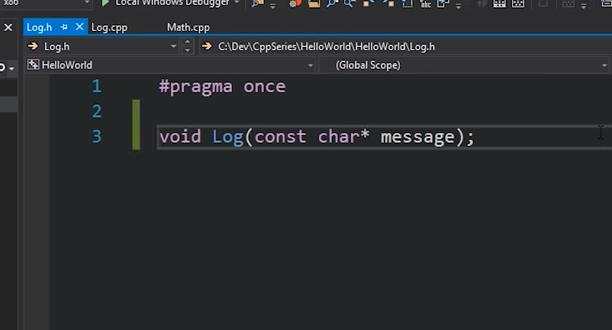
Compilation unit:使用 #include来包含对应的header file，他不应该包含class declaration（f辅助function还是可以的）

如果你的program用了free functions或free operators

那么free function与free operator headers应该去header file

Free function与free operator implementation 应该去implementation file

Log.cpp

Log.h

可以看见没有class，只有free function

Class 的method可以分为以下几类

1. accessor：get

2. mutator: set

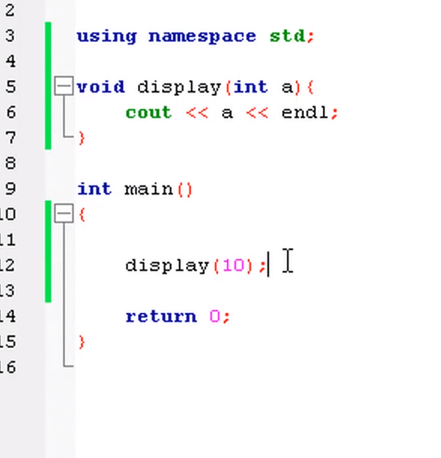
3, service method: 就是绝大多数method

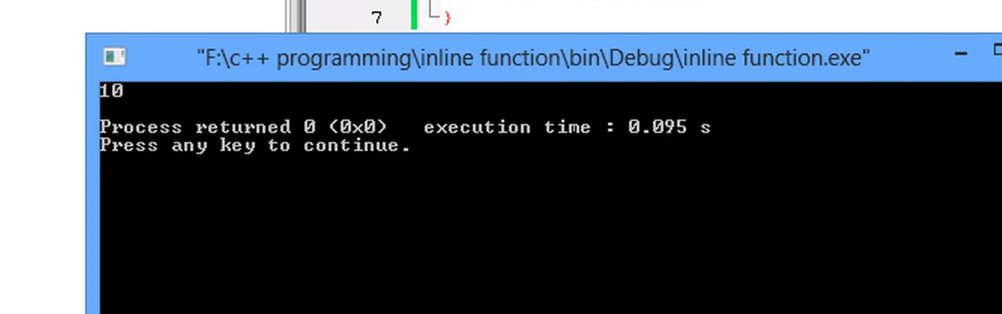
4.internal behavior methods :自用method,通常private

5.constructor ;构造器

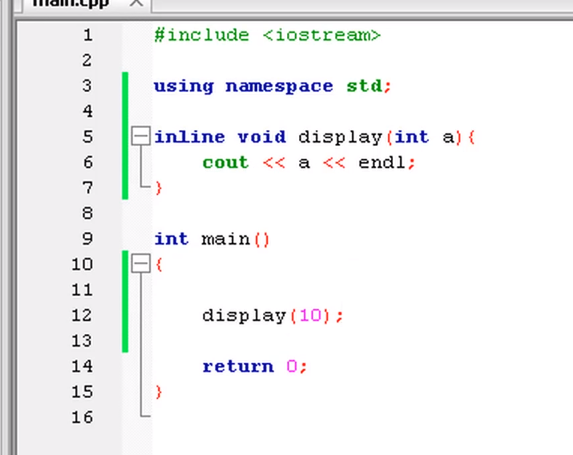
6.destructor：结构器，用来destroy object

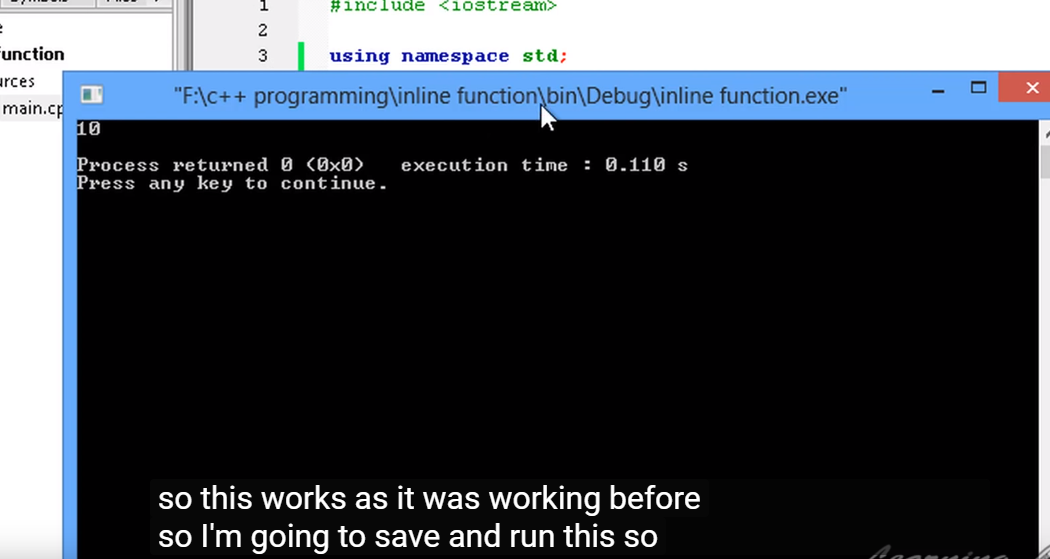
Inline functions and methods





这种是regular

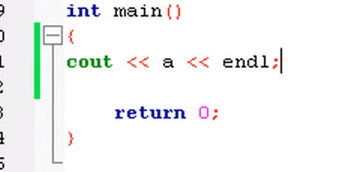




这种是Inline

regular的方式call function，要把function放到slack啥的分配内存，所以较慢

inline就有点像.h，直接把内容复制到下面

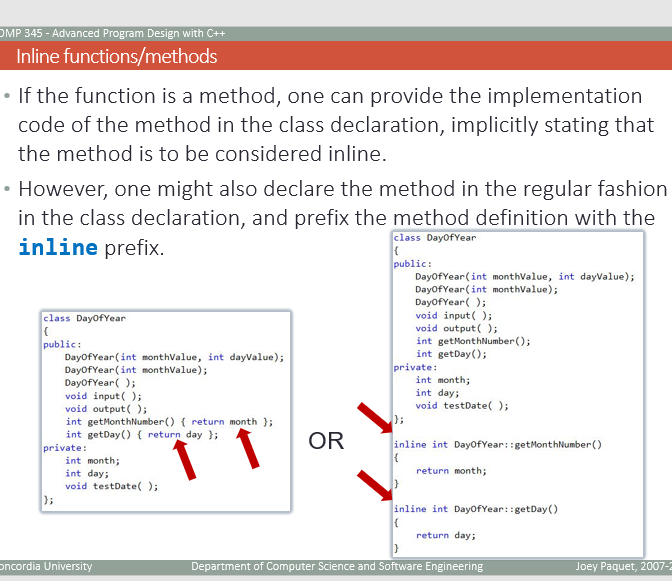


就相当于这样，不用放slack，较快

但是如果一个function如果要重复多次，就不合适，相当于多加了很多代码

如果一个function很大很复杂也不合适

两者都会让exe变大



method可以加inline前缀//h class外面，像cpp一种写法

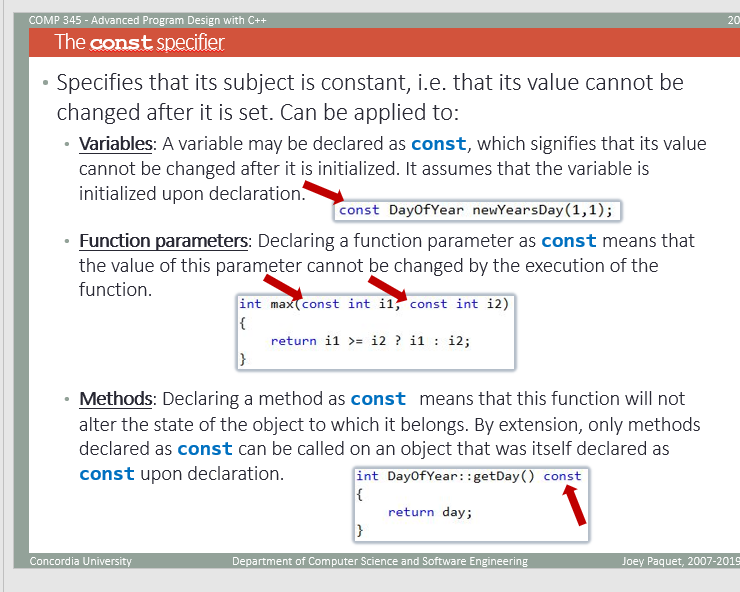
但还是要在内部declare

注意Inline只是一个提示符号对于compiler来说，如果他发现compile不了，compiler就会忽视inline，把它当做常规function//例如recursive function

简单来说短的用Inline,

get function用inline就挺好的

const specifier



指出这个subject 是constant的，也就是说，在设定完以后，他的值无法再改动

给variable设置const，代表着初始化以后无法在变动，而且初始化一定要在declare那步就开始

给function parameter， 那个给const的值，让你告诉在execution的function进行过程中，这个parameter的值不能被改变

‘

给method加const值，代表这个方法不会改变 object的状态，如果一个class object在declare的时候本身就加了const限制 //const variable， 那么他就只能 call constant method

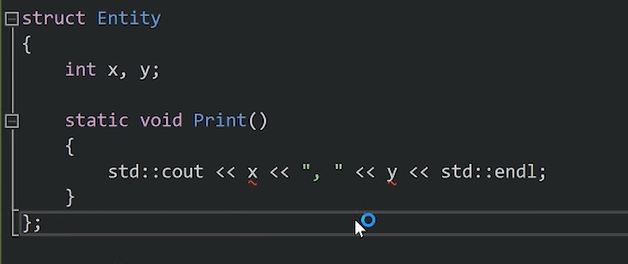
别的都写在前面，method写在后面

Static

对于class里的member加上static关键词，

如果是一个class variable，意思是这是一个所有object共享的一个数据

如果一个class 的function，那么它只能使用static variable



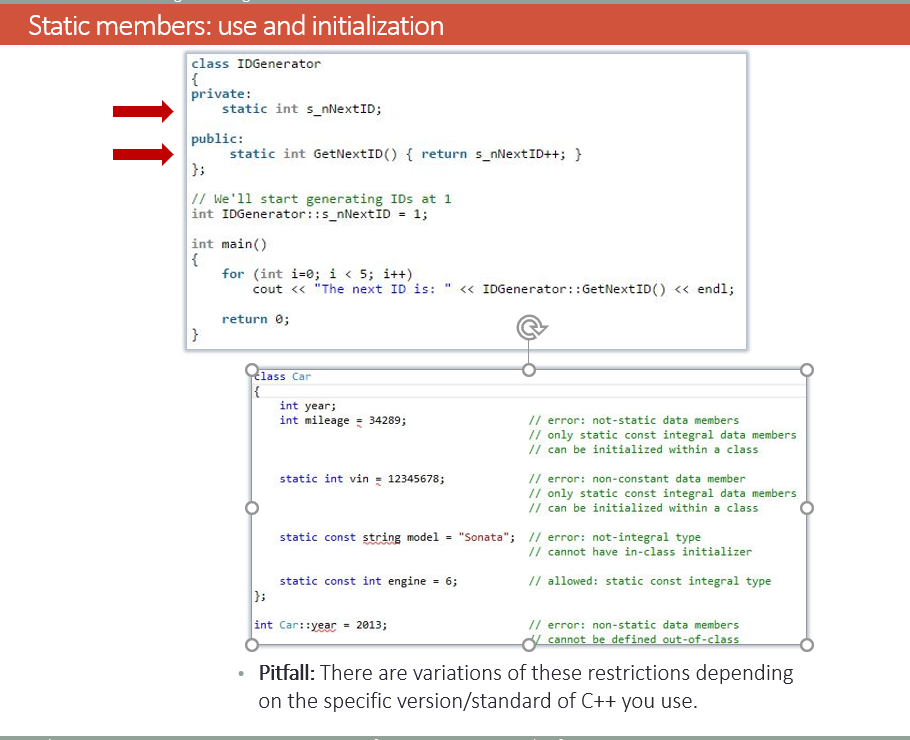
会发现有error，因为static 方法无法进入非static variable

Xy是 非static variable

Static function 通常用于描述那些独立于object 本身状态，（与个体object 无关）的behavior

Static 变量通常在class declaration以外被赋值， 只有constant static int可以在class declaration以内被赋值

C++ 不支持static Class



上面这个例子你可以看出给static赋值是在class外面的 //但是要declare

下面的例子：

你给class内给普通Int 赋值，错误//只有static const integral 才能在class内赋值

你在class内给static赋值，错误，//没有const

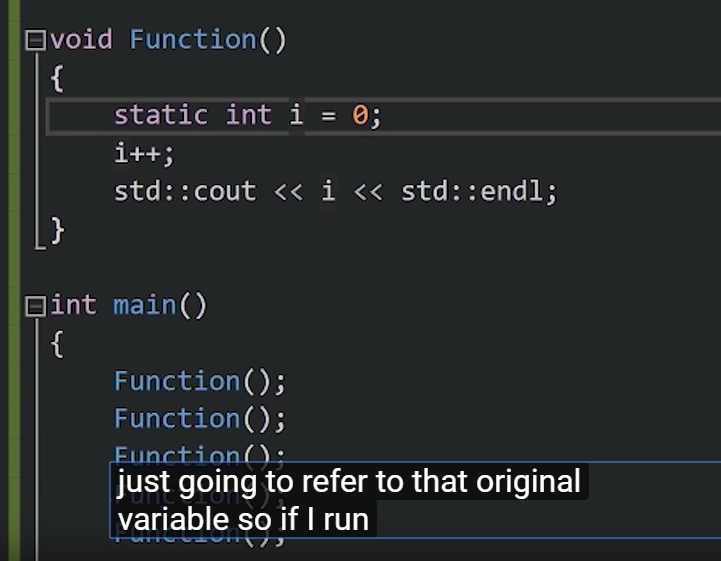
第三个，仍然错误//不是int

只有第四个才可以

第五个，错误，非static data member不能再class外被描述

Class外的variable后free function 加上个static

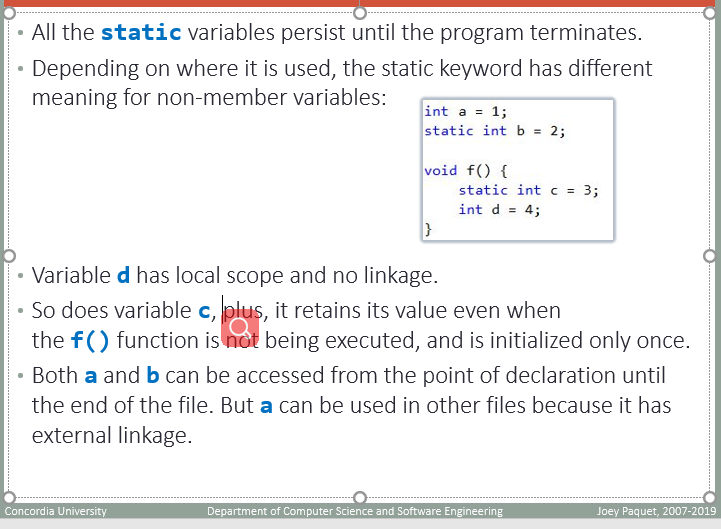
让这个变量或function的范围变成 只在这个文件内global， 其他文件不能访问，而



Global属性也很重要哦

这里不加static，每次都打印1

然而加了static，打印1 2 3 4 5，因为是global



A 与b都是 global，所以都能在这个文件内使用，但是a还可以在文件外使用因为他不是static，

D是local 变量，没有Linkage

C 是理论local，但他有internal linkage，所以他是 file global

//理论local的意义：除了你调用这个function 可以改变他的值，你不可以直接改变，因为你没有权限（他是local）

Class内，非static function可以进入static variable，static function不可以进入非static variable(只可以进入static variable，只可以call其他 static function)

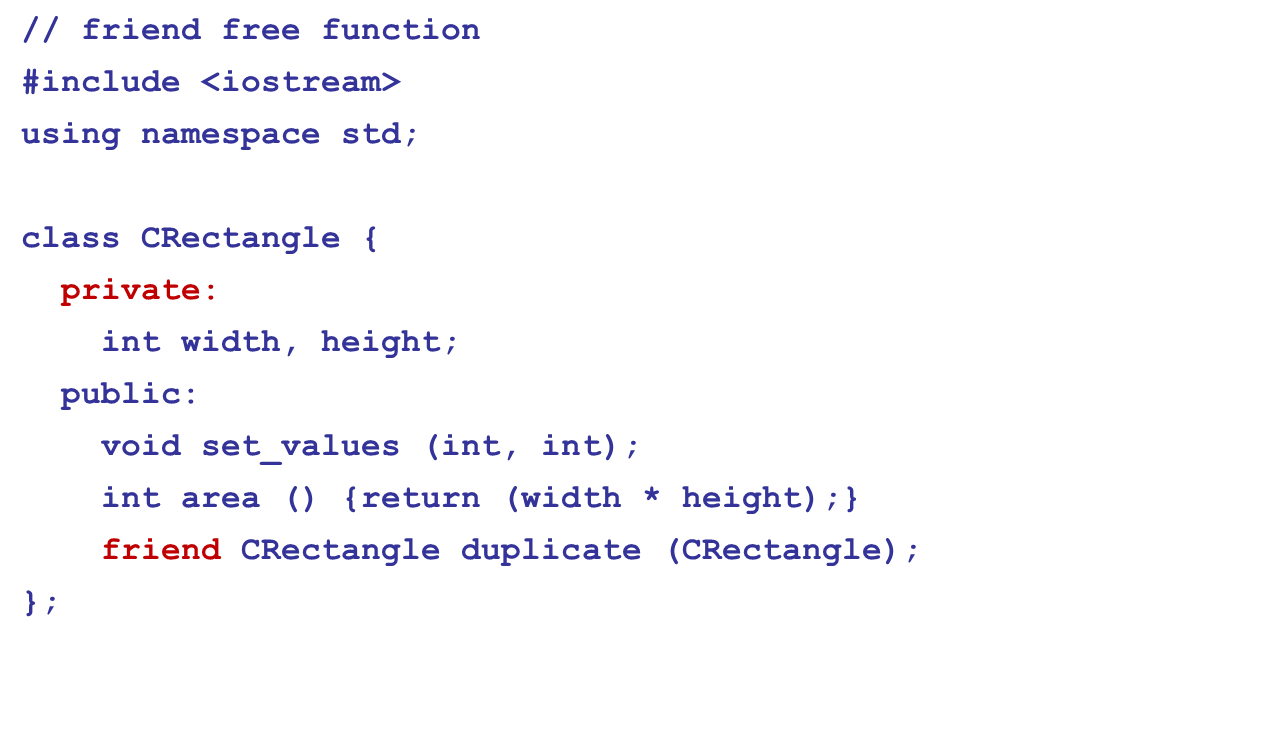
Friends

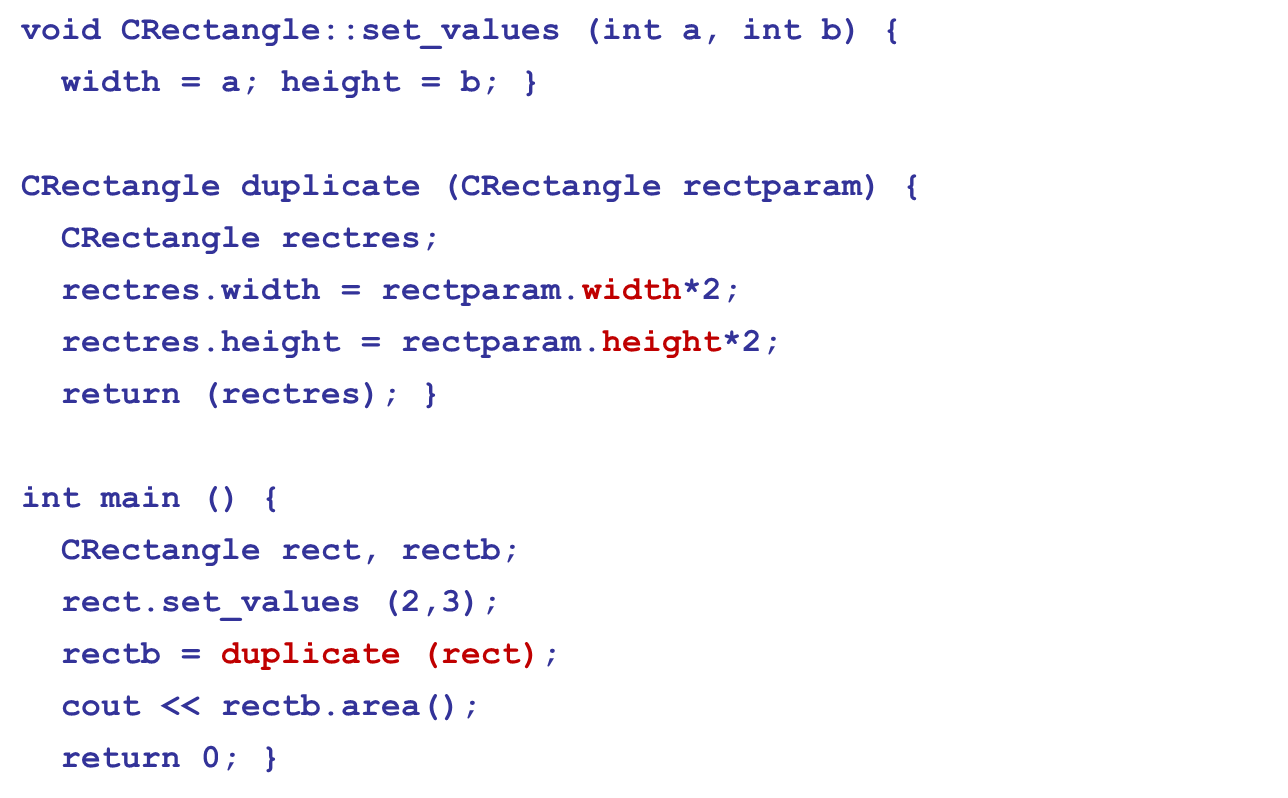
原则上 private或者protected function或variable 是不能在class以外的地方access的。

但是一个friend(function或class) 可以进入private 或protected members

Friends是在class里面declare的function 或class

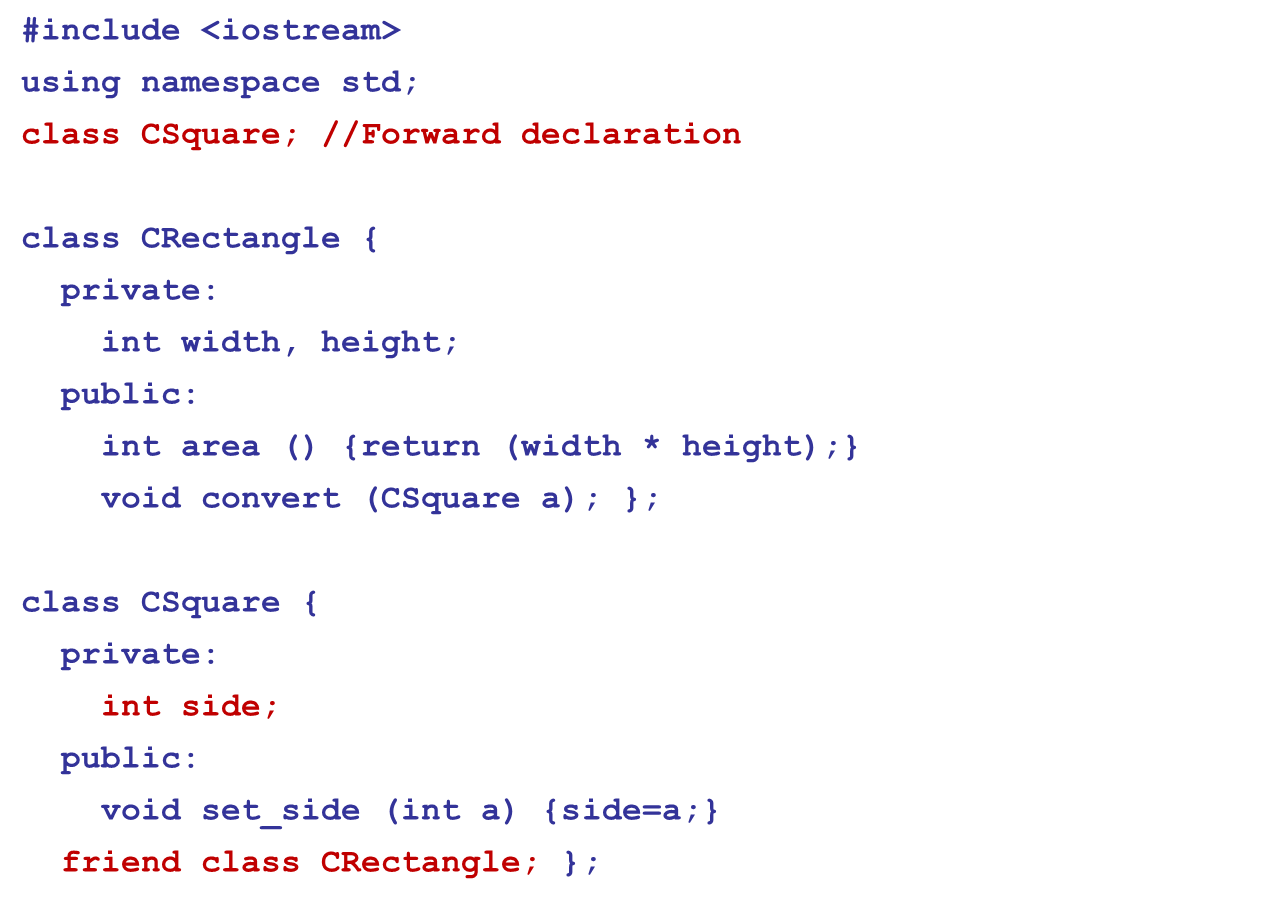
Friends 不是member，所以他们不能从superclass里继承，也就是说子类不会有这些friend





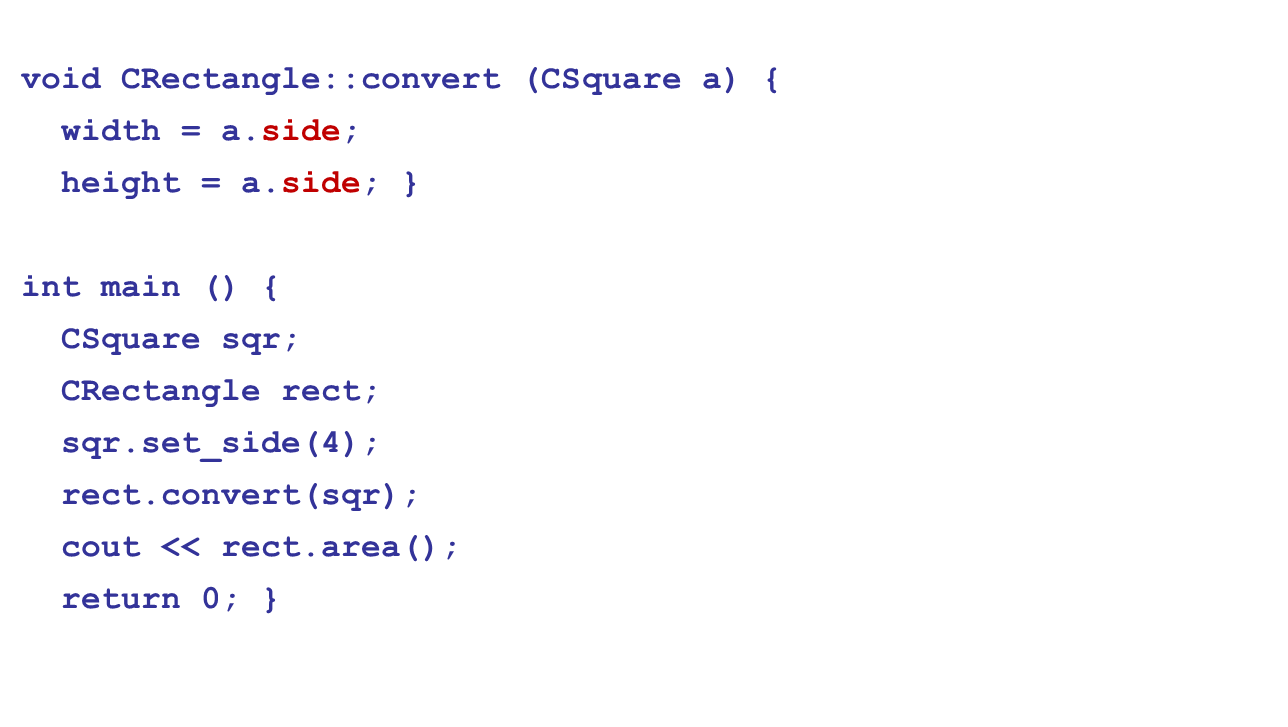
可以看见在 .h文件里声明了一个friend

然后我们就可以在class外定义使用这个method,然后我们就可以与private width ,height接触了



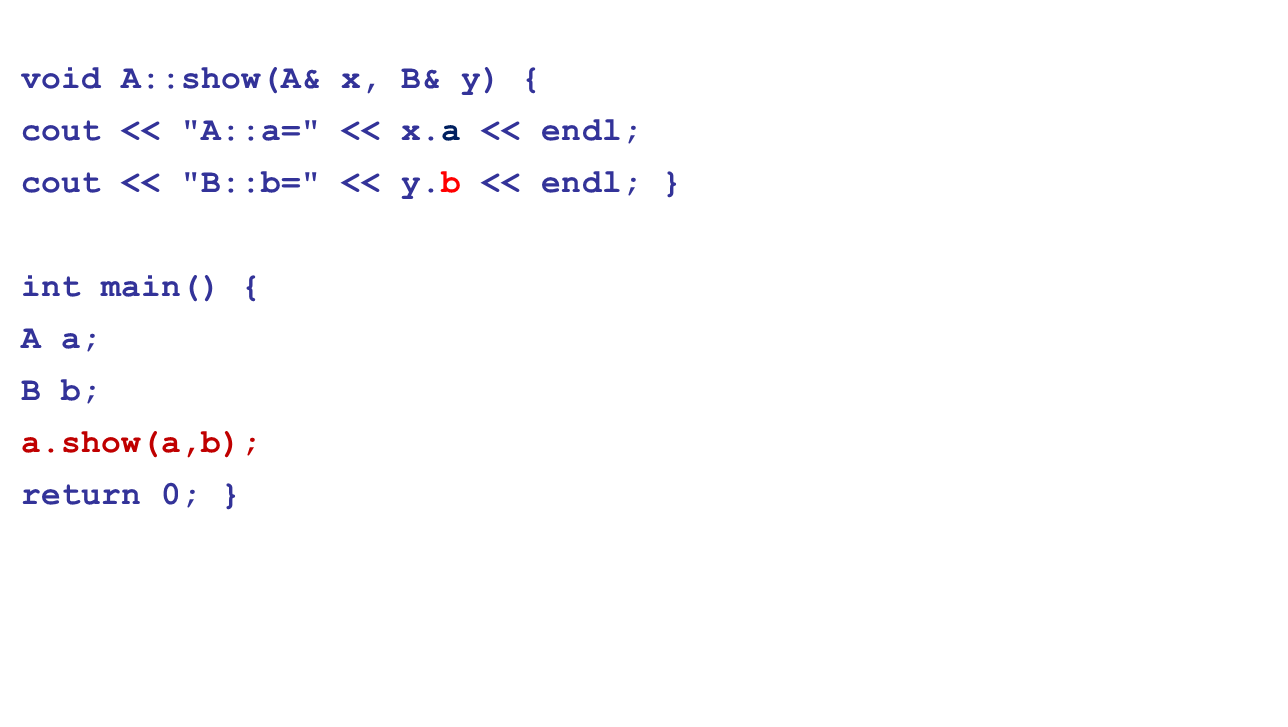
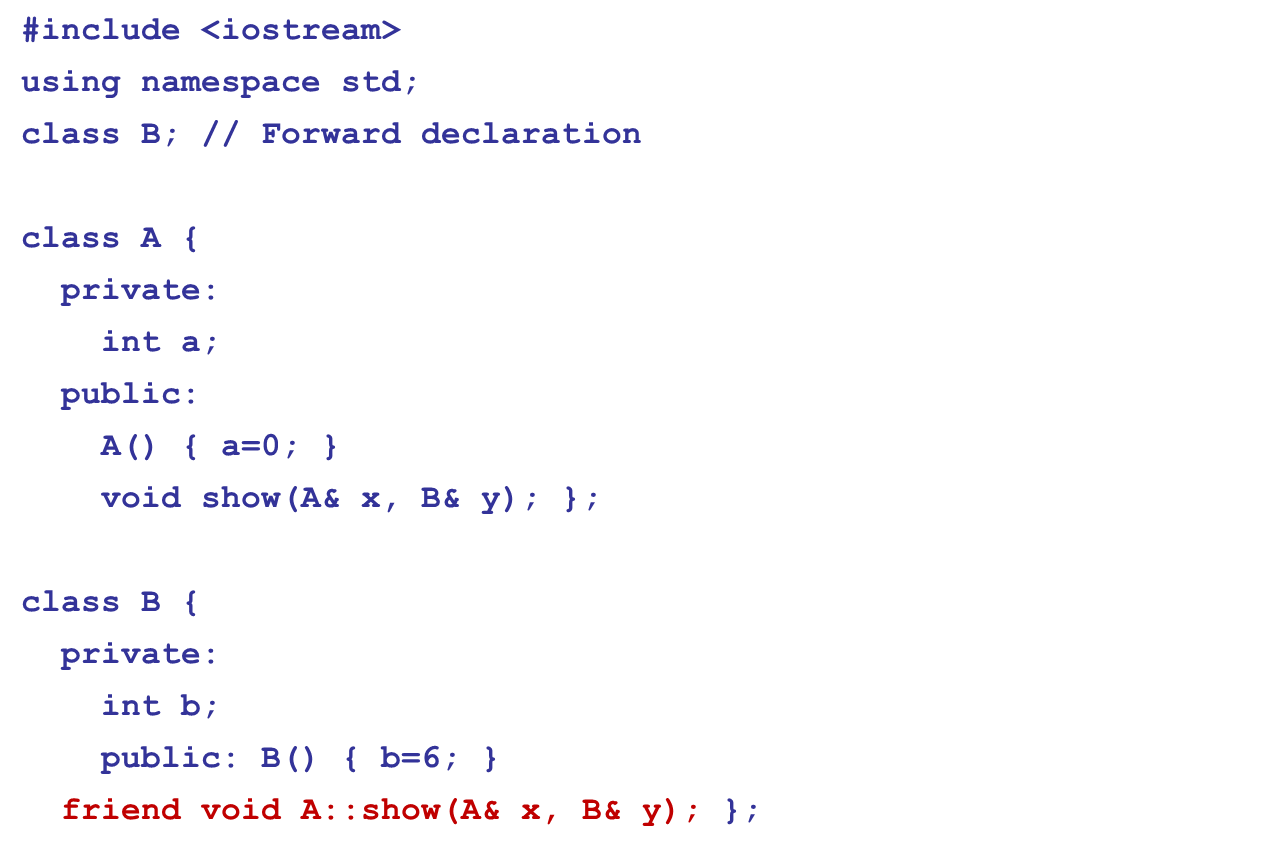
Friend一个class

注意这是两个class, Csquare给CRectangle查看他的权限，这样convert就可以直接a.side



这里Rectangle是Square的friend，

所以可以直接得到a.side



Friend其他class里的一个function

这里让A::的show作为B的friend，

所以A的show有了进入B里的private member b的权限

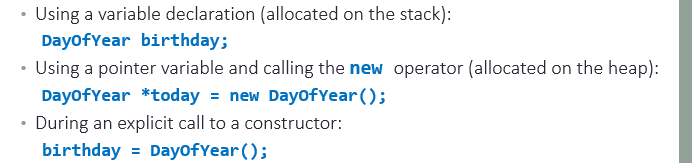
基本逻辑是

A 是请求FRIEND方， B说我允许你做我的friend //可以指定你一个function是我的friend

然后A的friend权限部分就可以得到B的private protected信息

Constructor and destructor

C++用constructor于destructor来创造or destroy objects



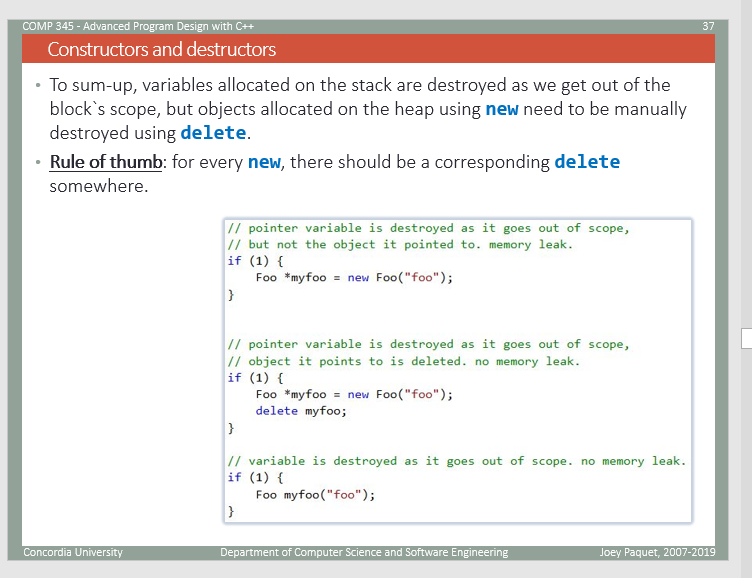
创造的三种方法，区别在于第一个被放在slack上，第二个Pointer在slack，new的object再heap

第三种是explicit call

Destructor一般来说自动使用

当超出function范围，所有在function内declare的variable会自动叫destructor，然后stack frame会把这个function Pop out 出stack

当你delete一个pointer variable的时候，也会呼叫destructor



简单来说，放在stack的variable再超出block 范围以后自动destroy，但是用new 创造的heap里的内容需要人工的使用delete

所以对于所有new，我们需要找个地方delete

Constructor是一个member function,与class名字相同，

默认constructor不用任何参数，就加一个~就行

如果你没有class内不包含constructor， compiler会提供一个自动生成的默认constructor

来进行基本的memory allocation of the non-pointer members only，

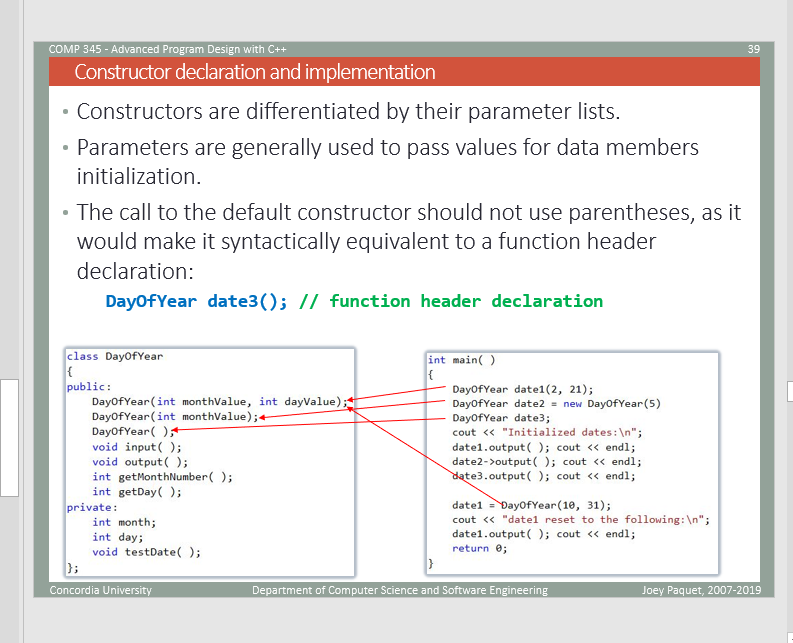
这个默认constructor只会生成没Pointer的量，没法使用 xx\* xx=new xx();

如果你有了至少一个constructor，就不会生成default constructor，所以你可以光定义一个参数型constructor而没有任何default constructor//default constructor就是没有参数的constructor，自动生成人工定义的都算，这句话讲的是我们有参数型，就不会自动生成，又没人工定义，就没default类型

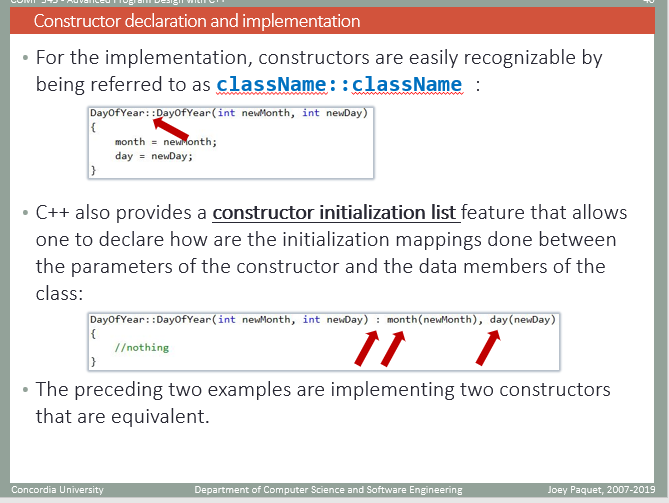
但是最好还是如果你定义了一个自己的constructor，也应该提供一个default constructor

Constructor的区别在于parameter list参数列表,

呼叫default constructor不应该使用parentheses 圆括号，因为这会让他与function header看起来一样



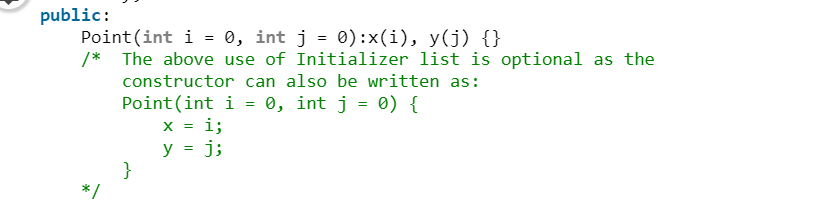
也就是说用default来创建object的时候 ，不应该使用圆括号，这里date2没加\*号，应该添加，因为有new



在CPP里砸实现

className::className

C++也提供了constructor initialization list,允许你



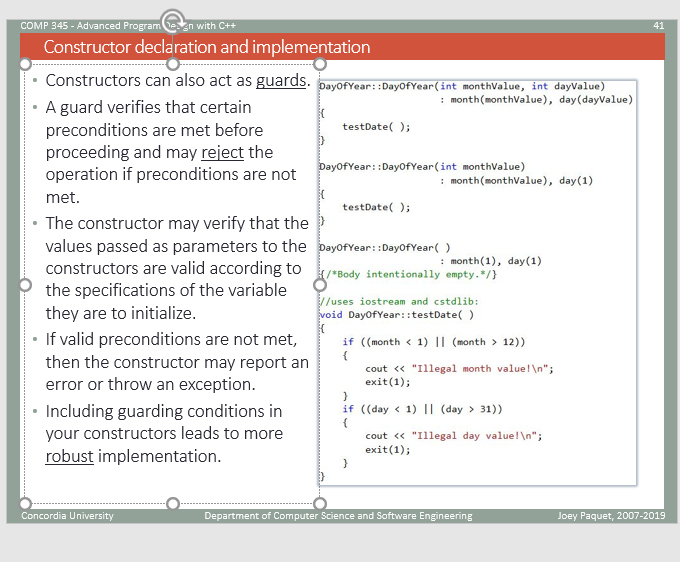
就是相当于直接赋值而不写下面的东西，左边是值，右边括号内是参数

Constructor也能作为guards使用//守卫

Guard先预设好preCondition， 有可能拒绝 生成object如果没有满足precondition,

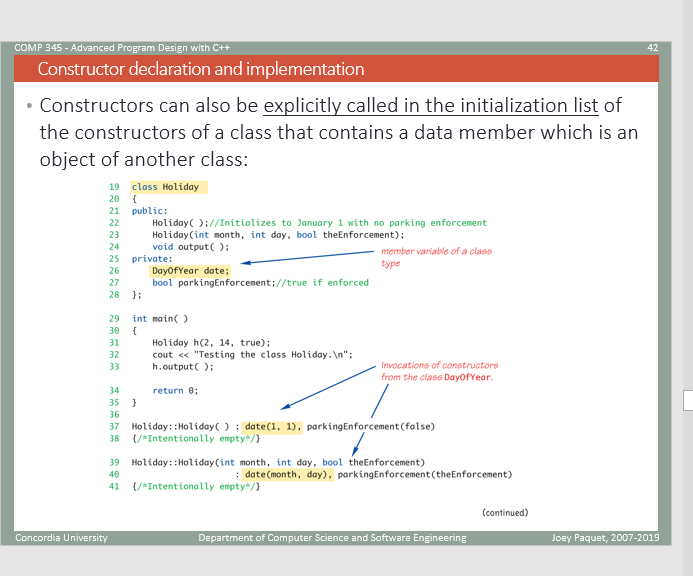
如果valid，成功initialize初始化

如果invalid，那么consttructor可能report an error或者 throw an exception



在参数后面接冒号：month(值)，

然后里面能改test



Constructor 也可以自己额外call一个其他class的object 作为member， 并且可以用在initialization list里

Inheritance:继承

用来refinement使.。。精确化一个existing class，增加supplemental data member或 function

同时redefine已有function来满足新需求

C++支持multiple inheritance，他可以同时从两个以上的父类那里继承

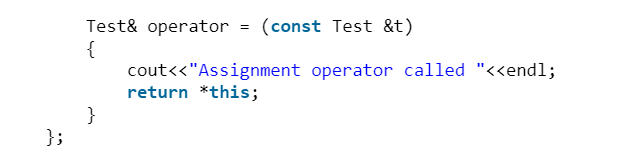
Abstract class:一个class 有着抽象member or function,这些抽象暂时没有具体描述，所以也不能实例化abstract class 的object，我们注定要用子类来让他完整

一个子类class会继承所有data member， member function

并不会继承的东西有

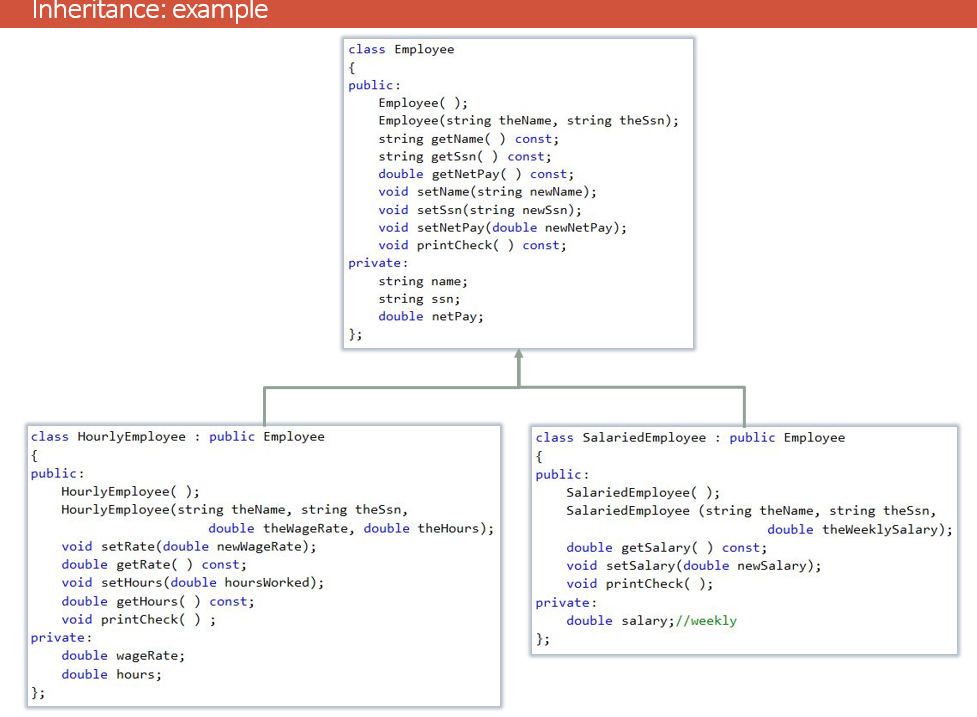
Constructor, Copy constructor,

Assignment operator



就是用来改变=定义的

Destructor,



实例，继承就是 class XX: public XX

然后就可以直接使用原有的东西，

一个derived Class可以直接对继承的function增加新的behavior 或者提供不同的behavior，

通过overriding or overloading

Overriding

子类function与父类function拥有完全相同signature：

啥叫signature:1.function name , 2. Sequence of types in parameter list, including order ,number types // 就是 variable name可以不同， 但是顺序一定要相同， 如果是 INT A ,INT B,STRING C ,那么overriding 也要是这个顺序，不是ABC没关系

Signature 不要保证：return type, const key word, reference specifiers used for the parameters//就是参数有的事 int& xx.没有这个&也是signature

Overloading，

两个或多个function名字相同，且都在同一个scope里//范围，但是参数parameter list不同

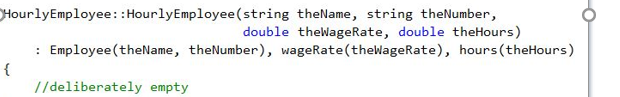
例如constructor 就是overload,

而上图的printCheck() 是overriden，因为const不考虑，有着相同signature

父类的constructor不会继承到子类

Run-time系统会第一步:call 所有的父类的default constructor，然后call他自己的default constructor

如果父类拥有额外描述的constructor，子类应该人工使用这些constructor

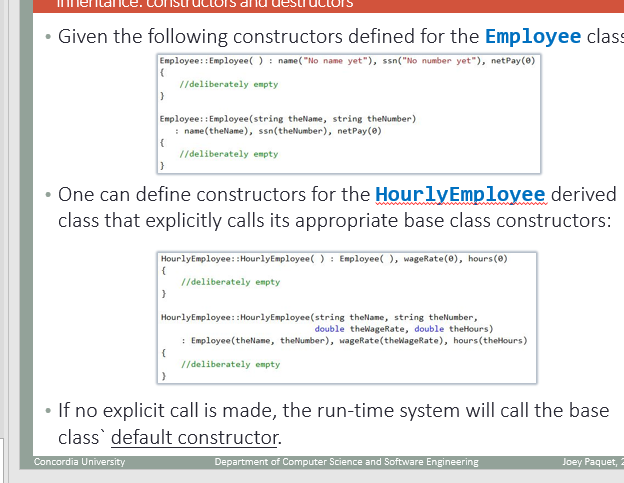


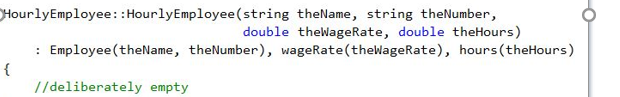
可以看见子类call 了Emplyee(theName,theNumber)

父类的constructor应该对他们的variable初始化//Initialize

因为他们将会被继承到子类，而子类没有继承他们

所以子类基本上就是call父类的constructor来得到要继承的member





看看这最后一步，如果没有选择要用哪个人工constructor，就会使用默认constructor

Assignment operator与copy constructor不会被继承

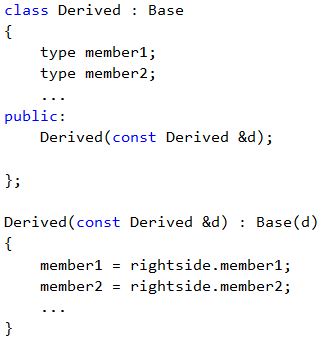
但是在子类class描述的时候我们可以使用它们

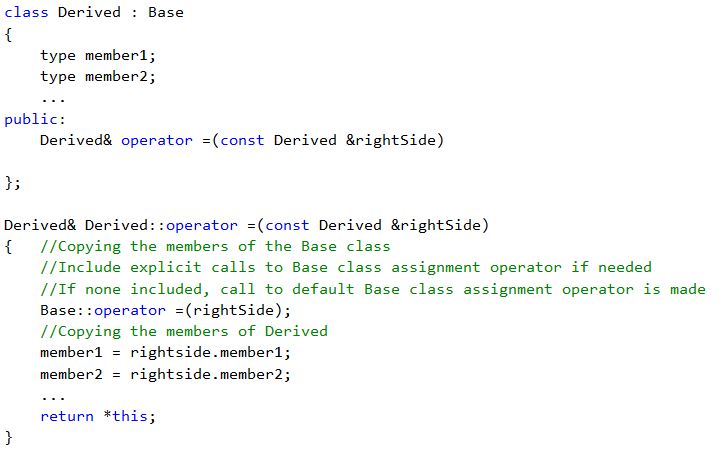
当我们用"="来用一个一个object对另外一个object赋值的时候，实际上会偷偷使用一个copy constructor先创造object的一个copy，然后用assignment operator来对新object进行赋值

如一个class继承了base class，那么为了完成上面这个操作，他会先让父类隐式implictly call这些操作，注意复制没法继承只能在当地进行，然后把这些值copy下来

compiler会自动生成assignment operator与copy constructor， 会shalow copy这个object

如果用户自己显式explictly定义， 那么与显示constructor相同，使用这个显式copy constructor//assignment需要 子类自己 额外指定





你可以看见子类声明copy constructor的时候 使用了Base(d)

一级 描述等于的时候使用了

BASE的等于

不会偷偷call copy constructor的情况



我们要知道，copy constructor也是constructor

所以如果你事先已经用default constructor创造了一个object，

那么就只会有assignment operator

啥时候回偷偷call一个copy constructor



你事先没定义，同时将赋值与declare进行，会先call copy 在赋值

额外定义copy constructor与assignment operator是绝对有必要的，如果你拥有Pointer data member, 因为默认的是shallow copy，他只copy pointer value，指向的还是一个variable，想要dep copy你必须copy所指的值，然后让他放在新的Memory里，并用一个Pointer来Point他

多重继承

在C++中，子类可以继承多个base



加逗号就行，

那么问题来了，如果base1 与base2 有相同的名字的member呢

有的人认为不应该使用多重继承

任何涉及到拥有equivalent variable /method的设计都不应该使用多重继承

Diamond problem，一个父类继承出两个子类，然后一个子类继承这两个子类

