function是一个有名字的block，通常来说，接受某个Input，进行一项任务，然后return一个结果，但也有可能不接受Input或不产生结果

一个C里的function的标准形式是



其中parameter-list的格式是



如果parameter list前面没有指定type，那么默认就是int

如果我们进入function的地方在描述function之前，那么我们必须要让compiler知道这个function的存在 BY 用以下方式来描述 function的原型prototype



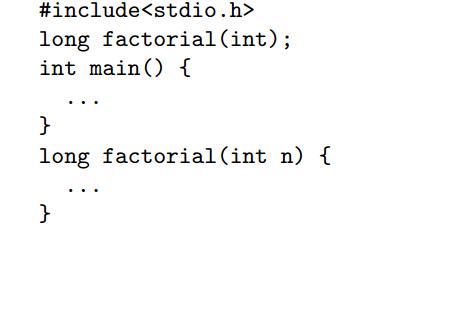
期中return type与function name必须与后面的 function描述一一对应

function main()不需要prototype

recursion: c支持call it self

recursion例子：

基本结构：

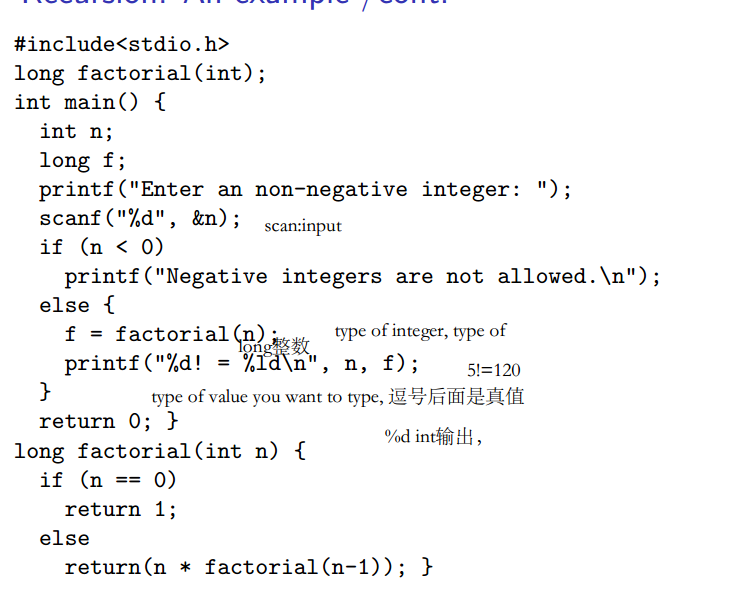


注意我们第一个 long factorial(int)就是 原型prototype

int main() 不需要prototype

后面才是正式描述

完整版



printf: 输出

scanf 输入

scanf(“%d”,&n)



%d是 我们接受的格式， n是变量，scanf就是我们输入一个十进制d并且 把他赋值给n

printf自己是没法换行的，所以我们要在最后打\n

f=fractorial(n)我们在main里可以使用还没没定义的fractorial，因为我们已经用了prototype，因此main能意识到这个fractorial存在

printf

%d意思是接受一个十进制整数，

%ld 输出long整数

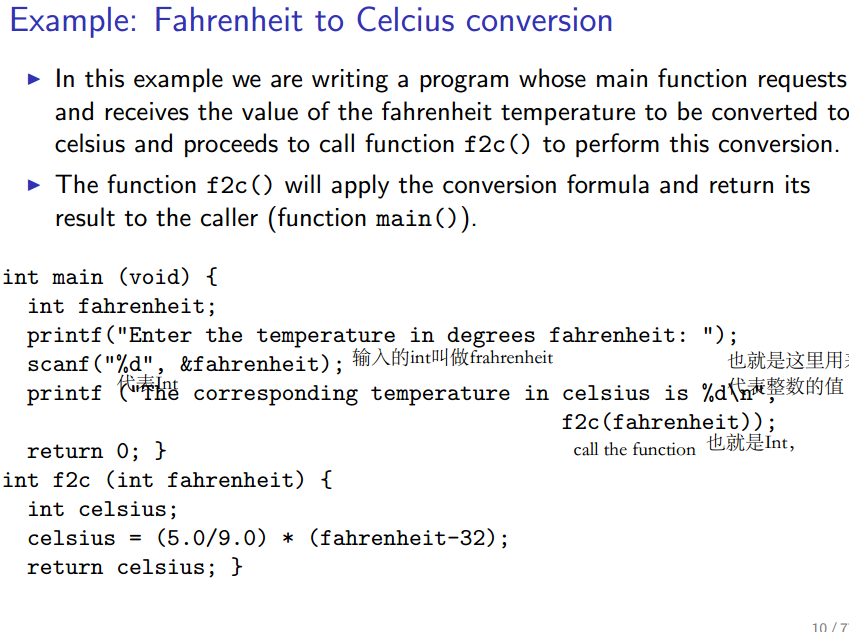
\n换行，

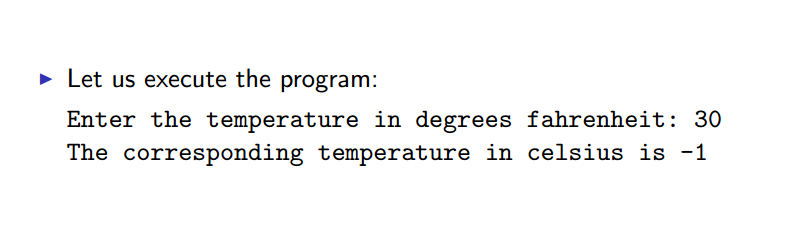
然后n对应的是%d， f对应的是%ld

scanf 是把接收到的%d 赋给& n

printf 是吧已有的n 赋给%d的位置打印出来

在C中，程序的进行永远是从main()开始的，





标准的应该是



原来的没有prototype



Global and local variable

我们要区别global与local variable

global variable:

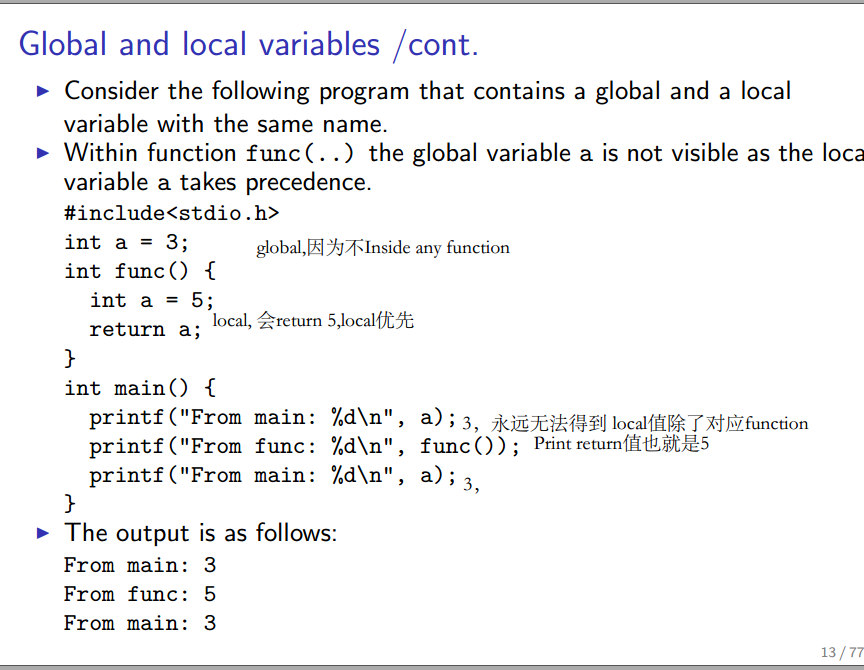
在program file的最上方被定义，并且所有的function都可以access他

local variable

在function内部被定义，只能被此function使用

shadowing:虽然不是一个好习惯，但是如果有global variable与local variable有相同的名字， 那么在这个function内，local variable优先级更高

global variable有默认初始值，local variable没有

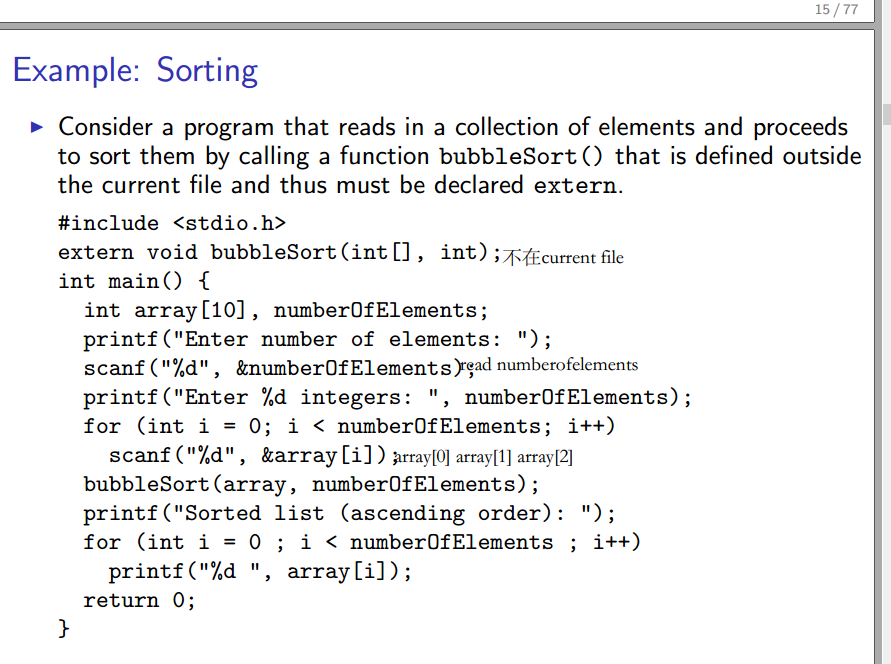


三个前缀

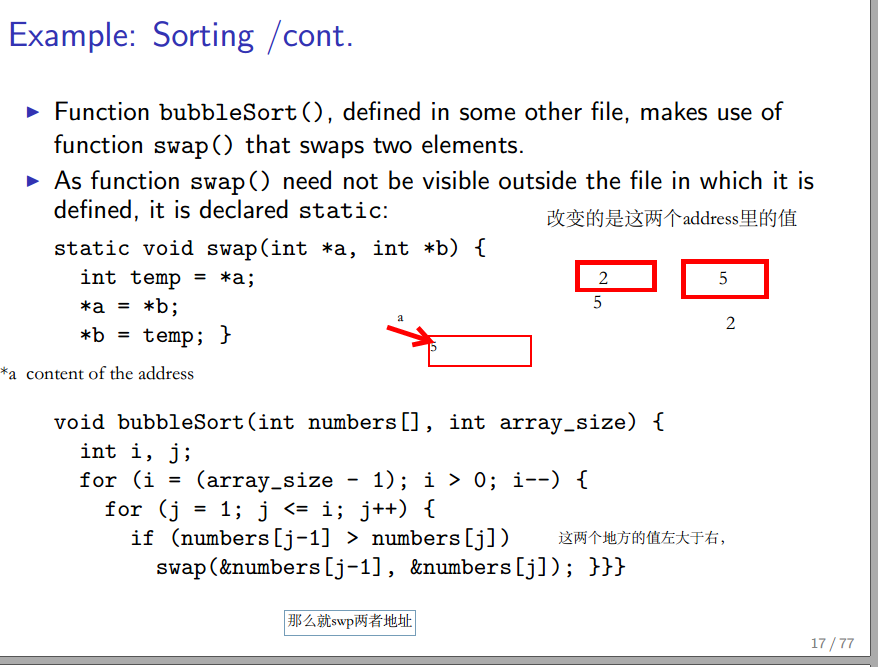
extern:这个variable/function可以在当前file以外被描述定义

static：这个variable/function仅在当前file可见

（blank）默认前缀：可以在当前file被描述定义，并且可以被以外的file可见



bubbleSort是在另外一个文件被具体定义的，这里用了extern，所以我们在当前文件也可以用bubbleSort



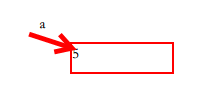
bubblesort在别的文件被定义，因为是默认前缀，所以对其他文件来说是可见的

static swap是用来辅助bubbleSort的，所以仅在当前文件可见

int \*a

a是指针

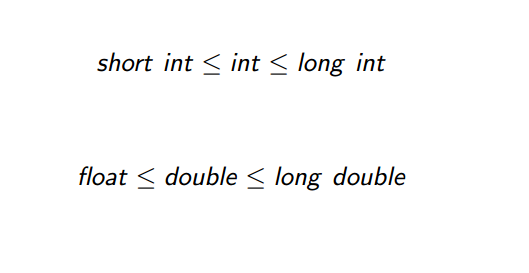
\*a是指针所指的值

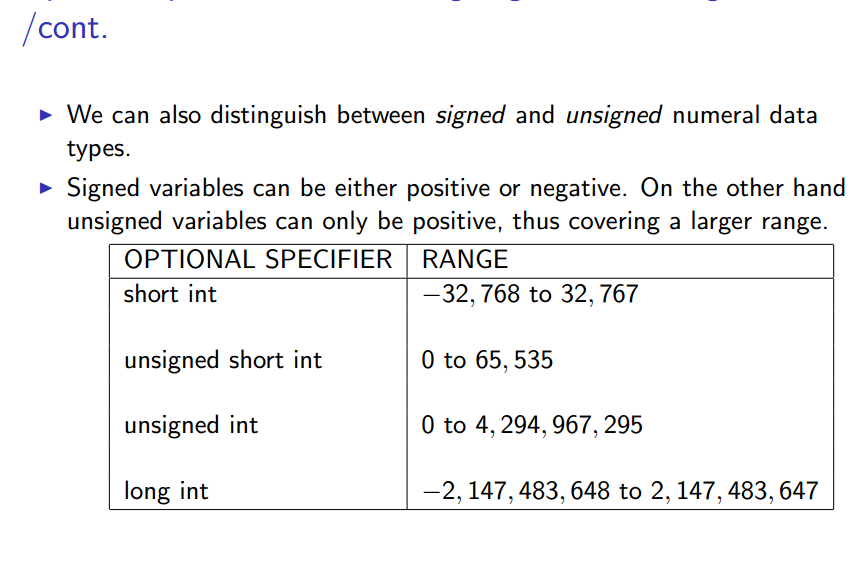


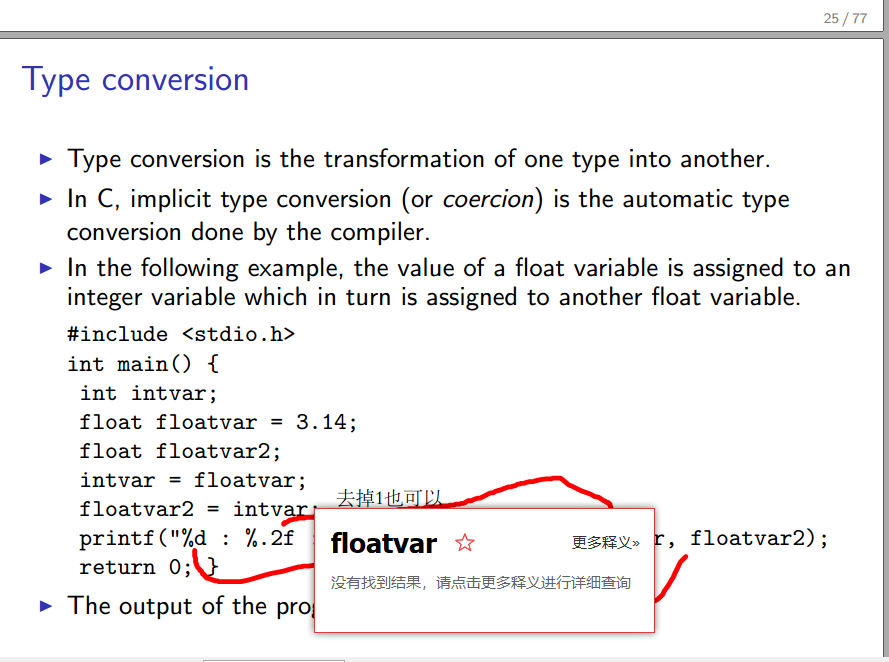
a是指针本身，\*a才是对应的值

我们又假设a是一个值，那么&a就是这个值所对应的地址







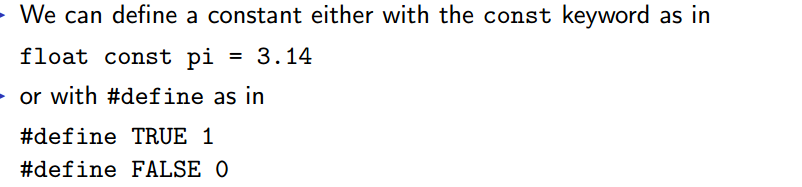


floatvar是float，2是float，intvar被floatvar赋值，floatvar2倍intvar赋值，

第一个indvar是3，因为被float赋值，float转换成int少了两位数

floatvar2是3.00，因为他是3赋值出来的float

定义常量，



类似JAVA的final，一旦定义就不能更改

composite data type

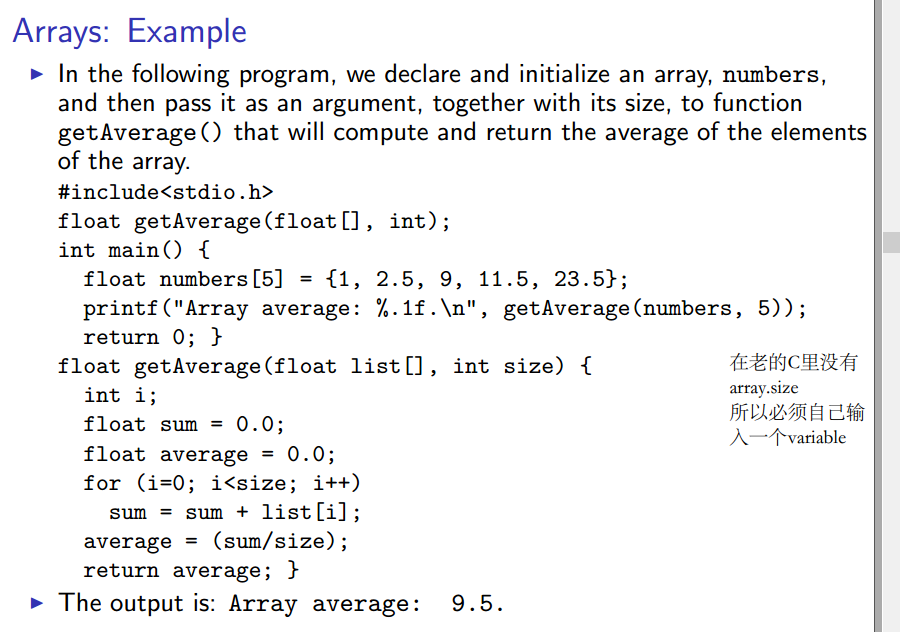
由其他primitive types或者composite types组合而成的数据结构

一般来讲它被称为数据结构：一种来组织，存储数据的方式，让这些数据可以被有效进入管理

基本的composite type有array 和record

array:

array可以是static(长度固定)或者dynamic(可以拓展的),



Pointer

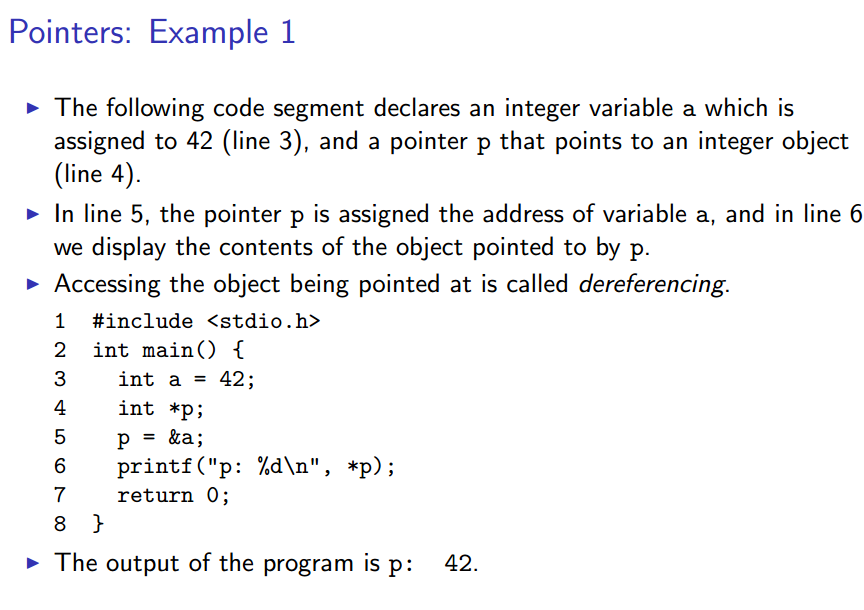
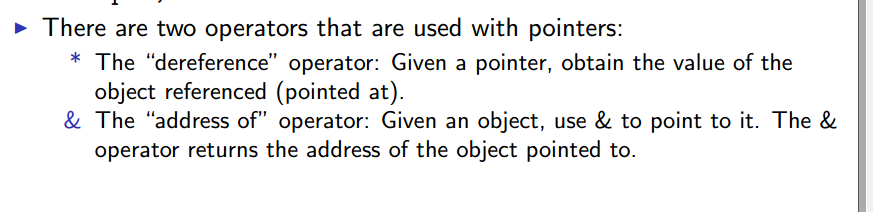
pointer是一种type：reference指向了另一个 value，通过存储另一个value的address

一个Pointer变量，（也叫做地址变量）被declared的时候通常在他的名字面前 加一个\*



\* “dereference”符号： 给你一个Pointer,\*号可以从pointer中得到被reference的object的值

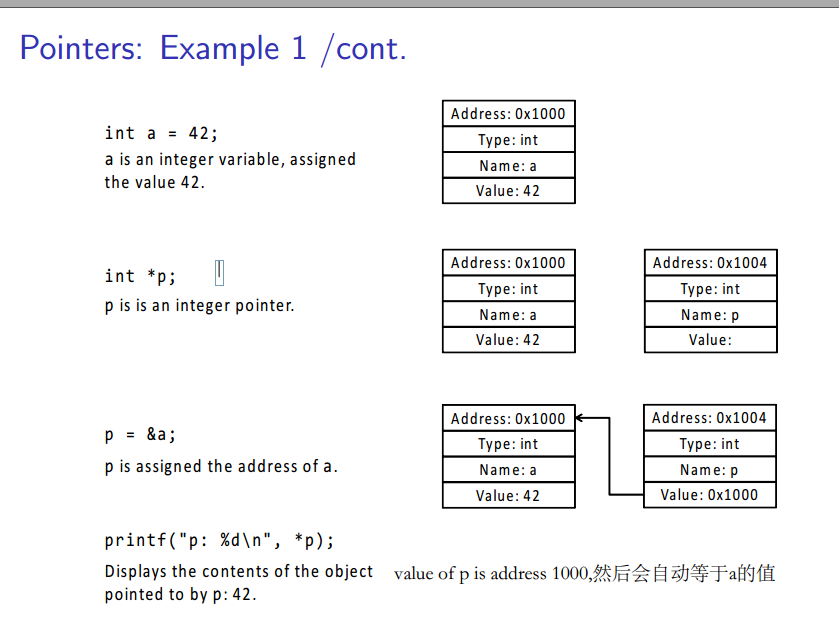
&object: address of 符号：给你一个被指向的object，通过&可以让你指向他， &符号return被指向的object的地址



int\*p p是指针，\*p是指针所指的值

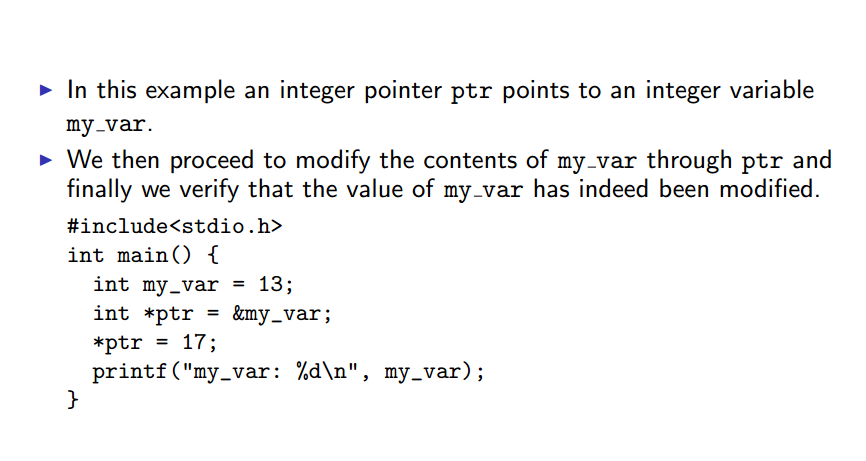
p=&a， &a，a是被指向的object值，&a就是指向a的地址，p=&a，我们让pointer等于a的地址，，p此时成功指向a

\*p就是对p这个指针提取他所对应的值，也就是a



p=&a p的值等于a的地址

\*p等于p所指的值，等于a的值



my\_var定义了一个变量

\*ptr=&my\_var

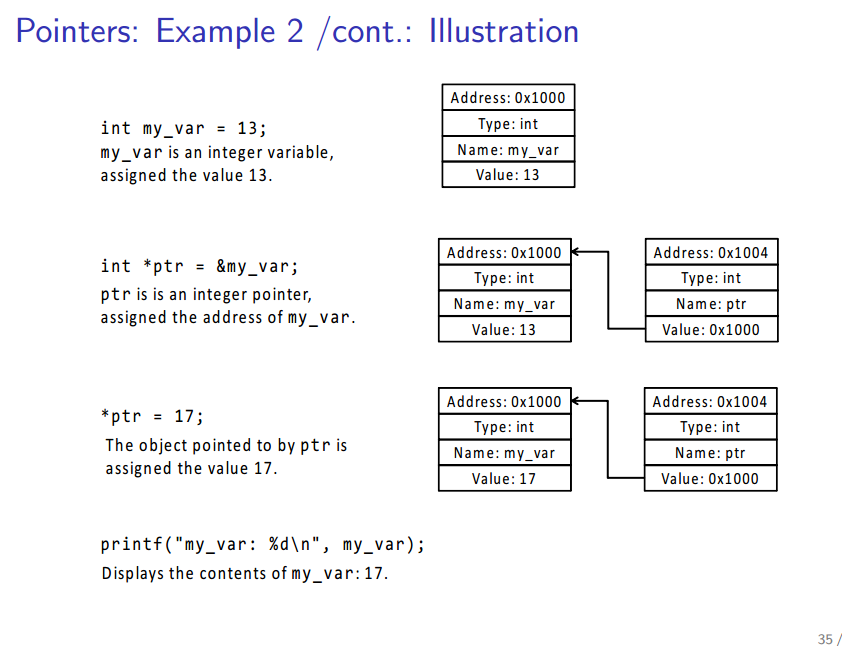
\*ptr就是ptr所指的object的值， &my\_var就是my\_var的address

感觉是有问题的

应该是

\*ptr=my\_var，ptr所指的值等于my\_var

ptr=&my\_var，ptr的值等于my\_var的地址

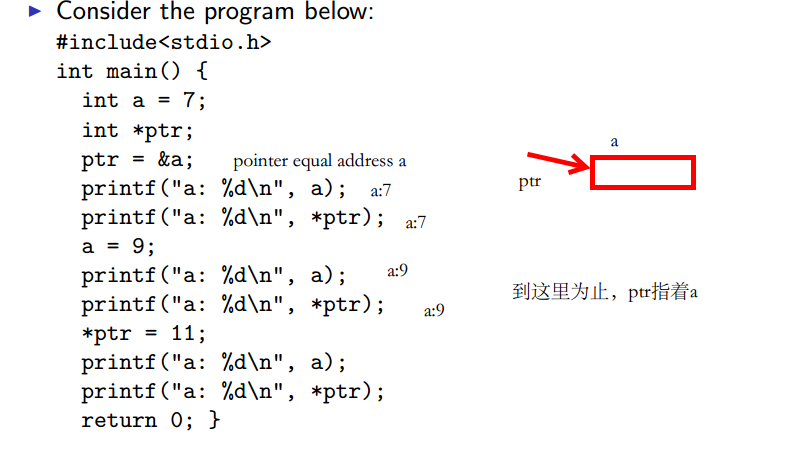


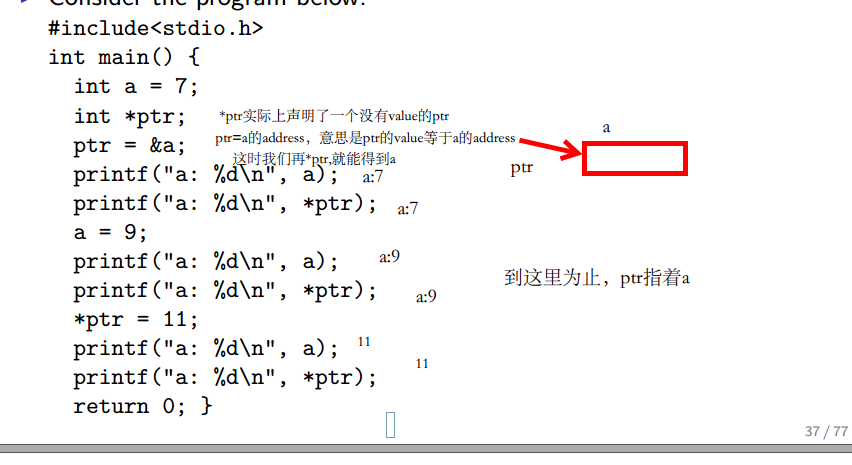
但是实际上是my\_var 被改变

这是简写法\*ptr=&a

aliasing

aliasing指的是一个memory location可以被不同的变量进入



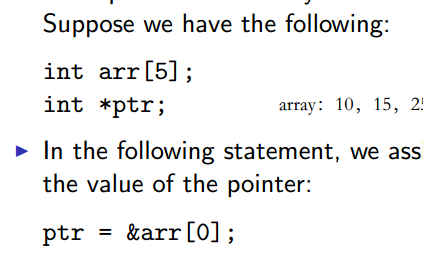


a与\*ptr是绑定关系，我们改变\*ptr，就会改变a‘

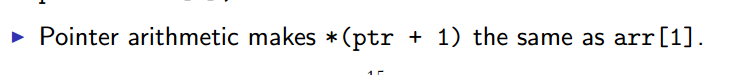
pointer arithmetic

假设有一个array arr[5]

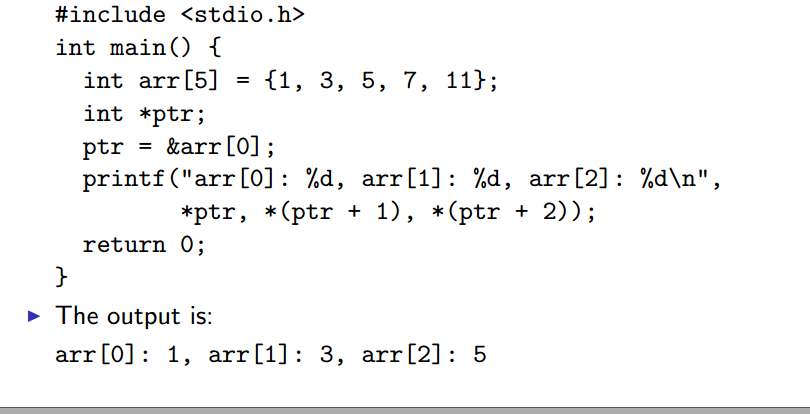
让ptr指向arr[0]



这时，通过pointer arithmetic

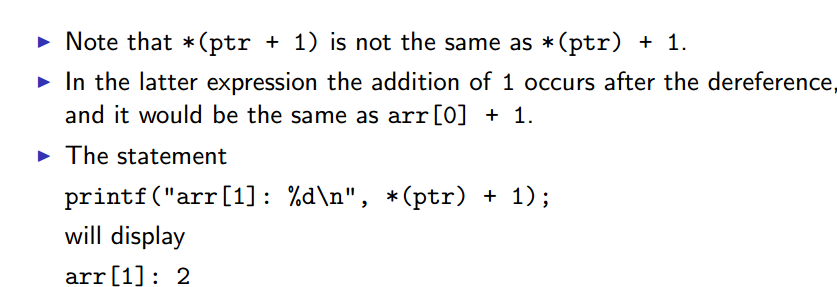


\*(ptr+1)实际上等于arr[1]



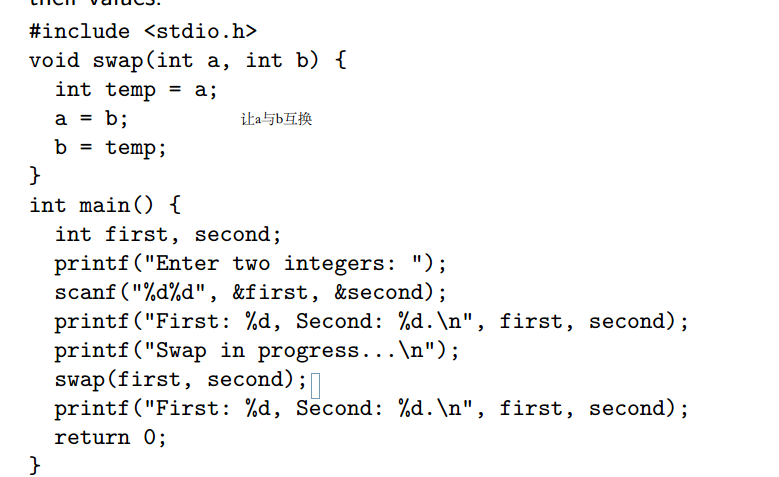
\*(ptr+1)与\*ptr+1的区别，

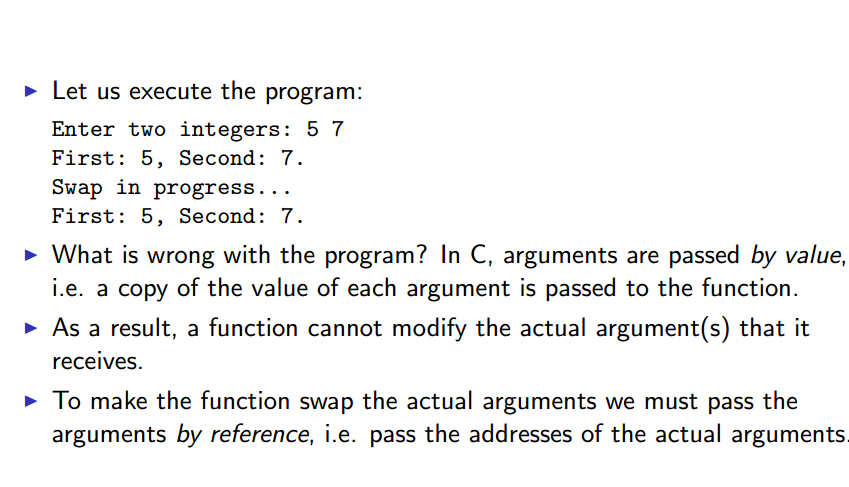
前者是a[1]后者是a[0]的值加1



pointer作为function参数

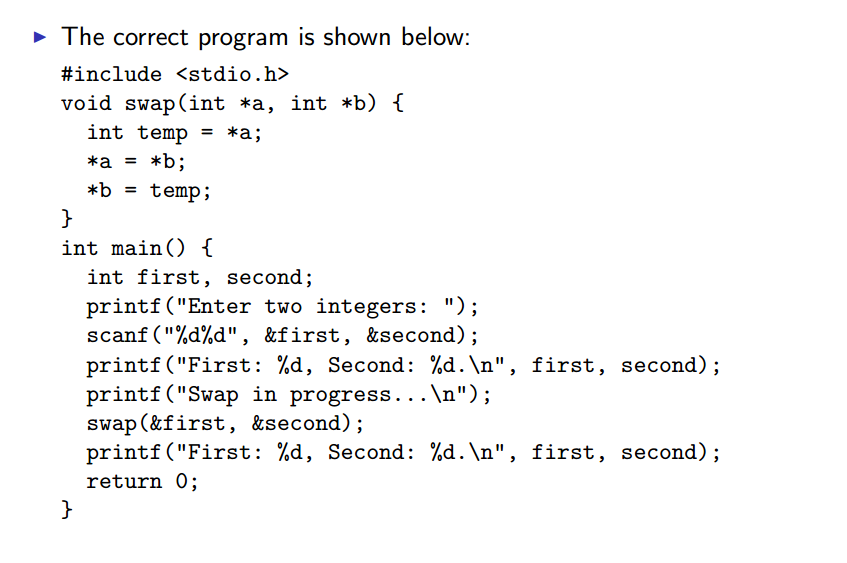
错误版本





错误原因，在C中，参数是值传递的， 每个argument的值被复制一份，并且传送给function，因此，function实际上是改变的复制的参数，而不是实际参数，

为了改变真实参数，我们们必须通过reference的方式传递argument



first此时使我们输入的值1,second是我们输入的值2



swap &first &second 实际上是把first与second的地址作为参数

\*a=&first,也就是说a是指向first这个数的pointer

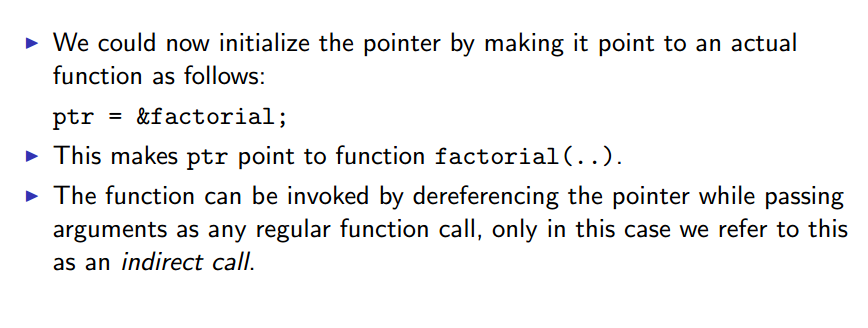
我们让a所指的数变为second，b所指的数变为first

也就是\*a=&second,\*b=&first

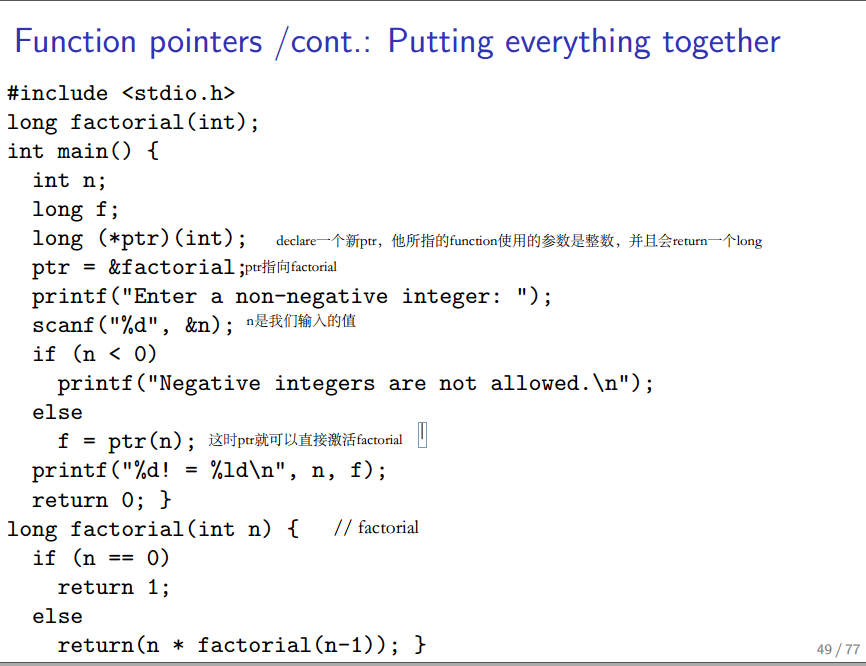


意思是让ptr指向一个int并且return一个Long的function

我们可以让ptr指向一个真实存在的function



这样这个fuction可以通过dereferencing这个pointer来激活 （\*ptr），叫做indirect call

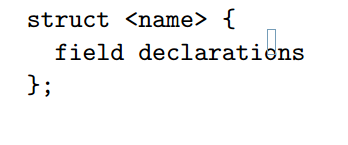


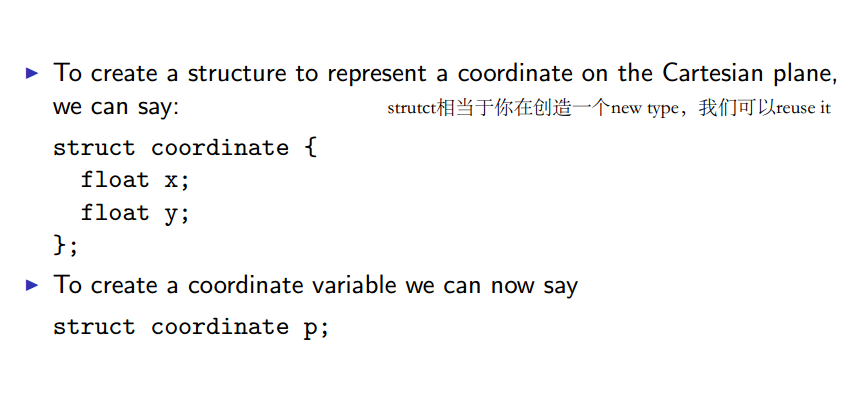
注意激活factorial的时候ptr不需要加\*

record

record，或者也叫structure，是一组element的集合，fields或者叫members,element，可以是不同的type，

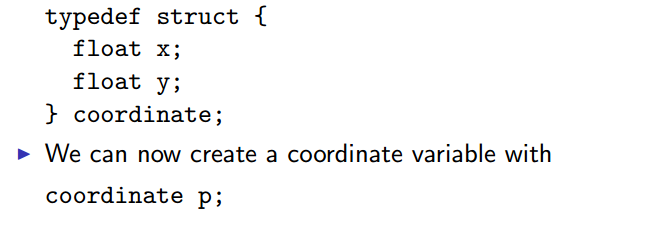
定义structure的方法

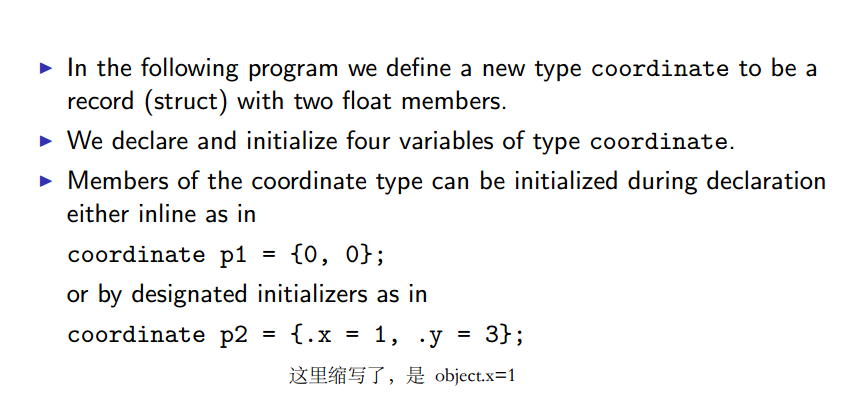


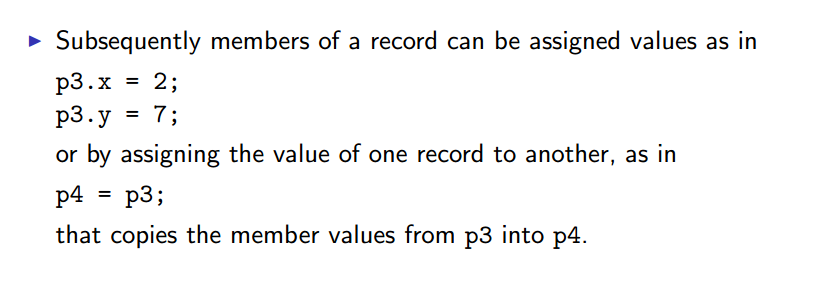


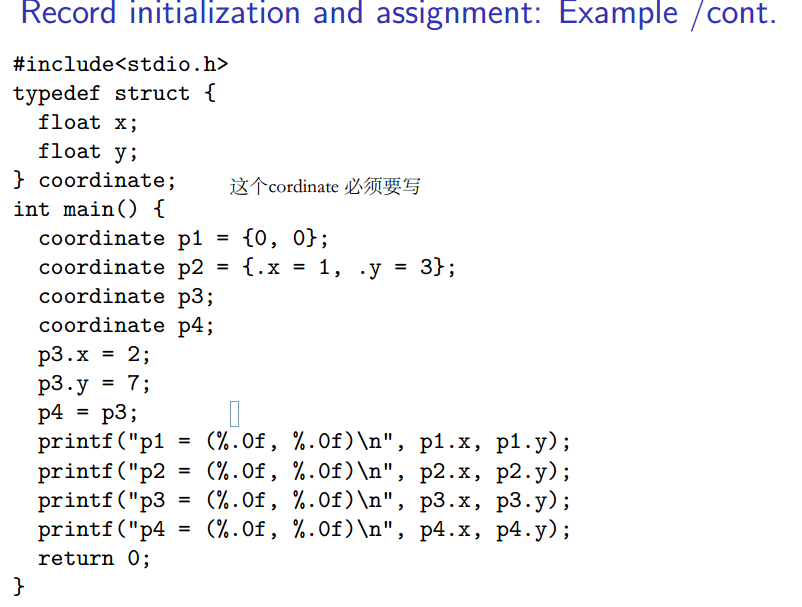
我们也可以不用每次在declare coordinate的时候加struct

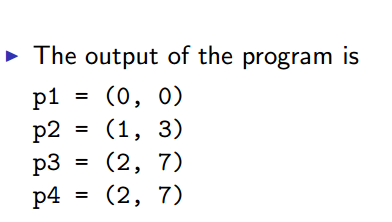
当我们使用typedef，实际上这是一种更好的方法



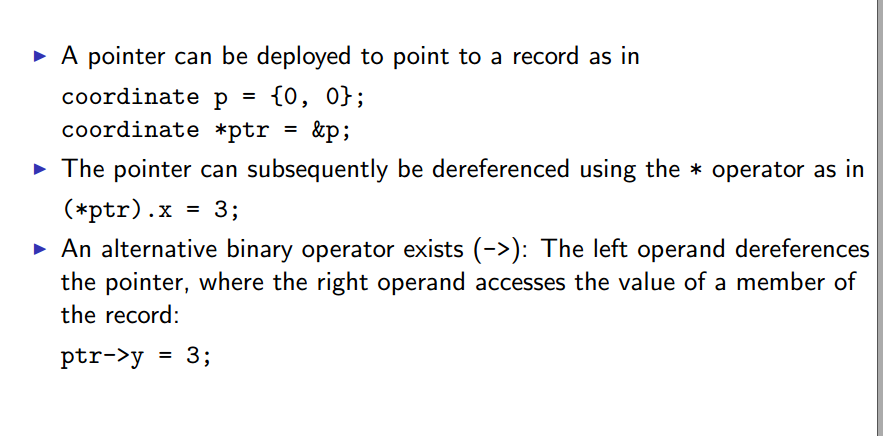


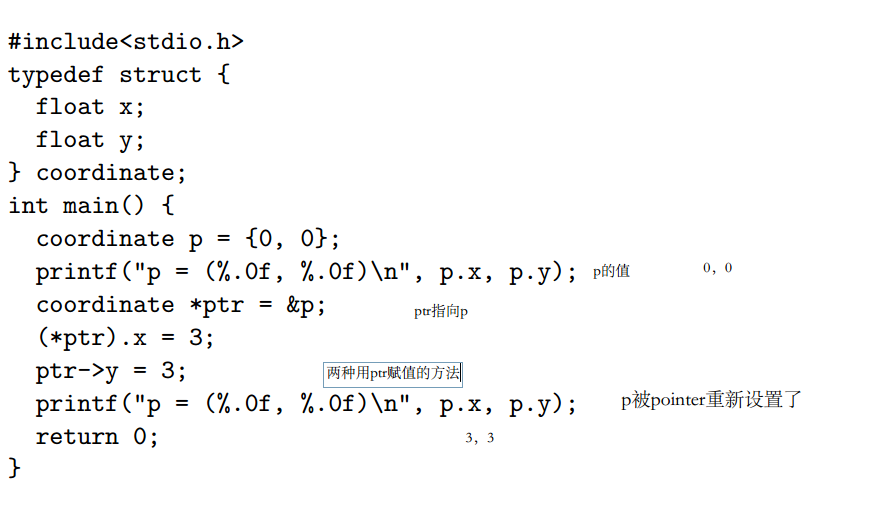


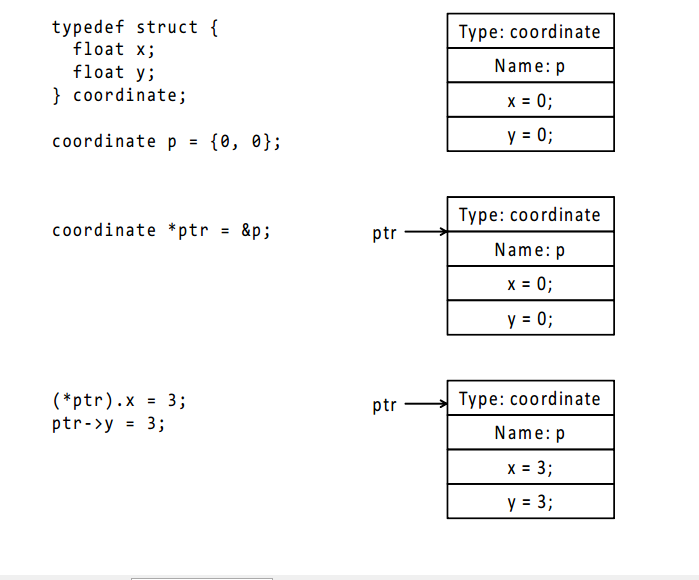




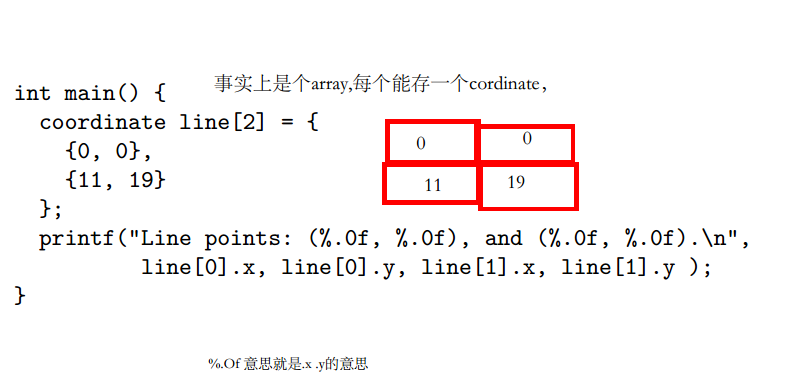
我们也可以对我们define出来的新record用pointer



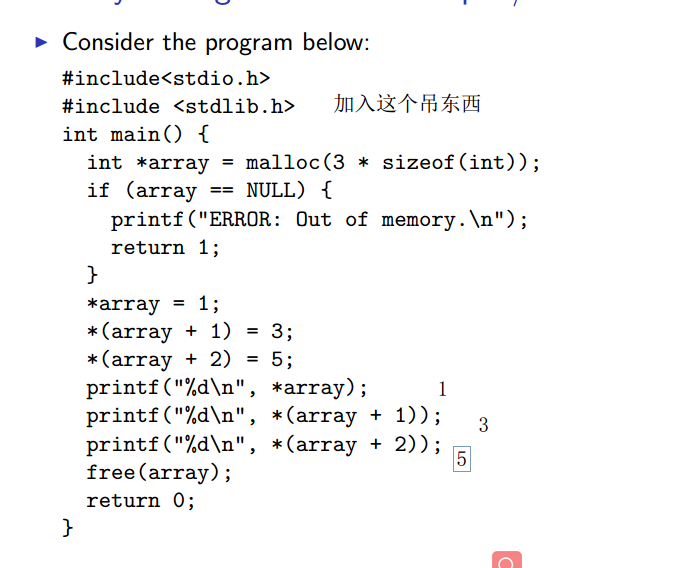


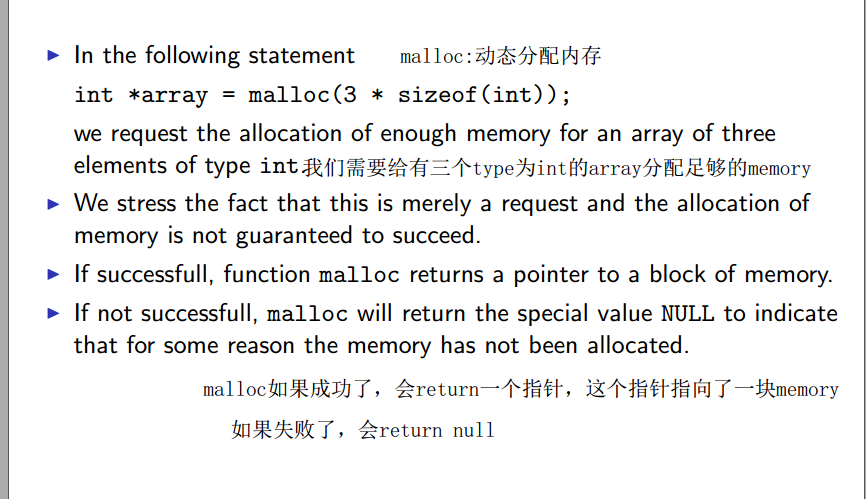


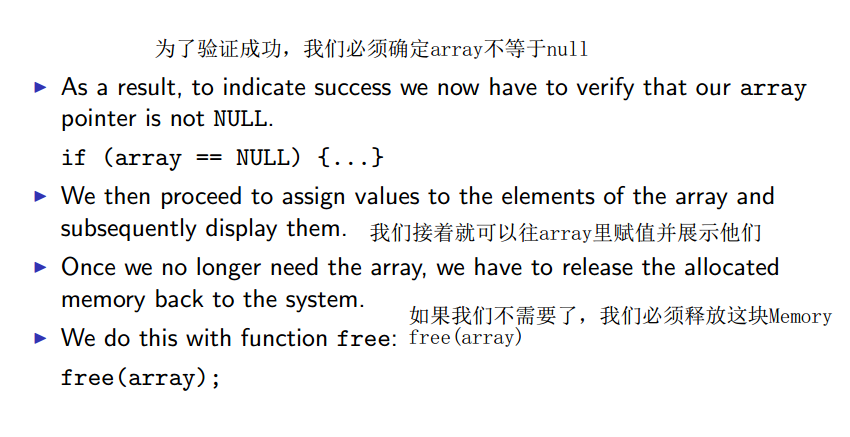
我们也可以coordinate 出一个array

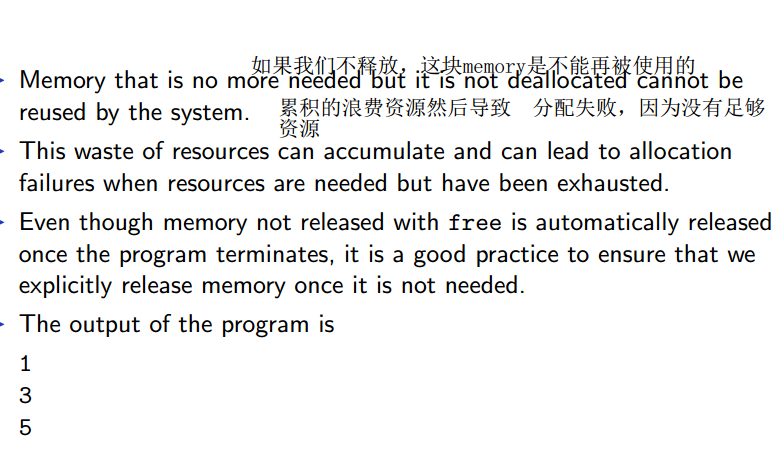


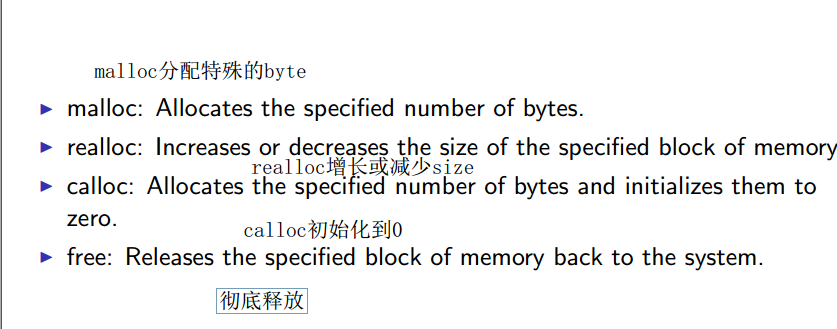
资源分配



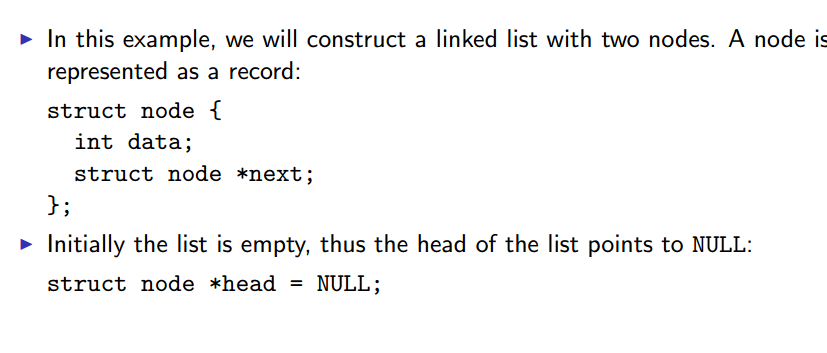








linked list



创造node

当list是空·的时候，head为空

