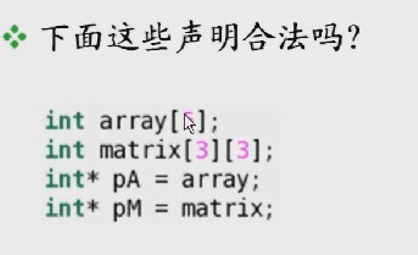




下半部分难度有所增加

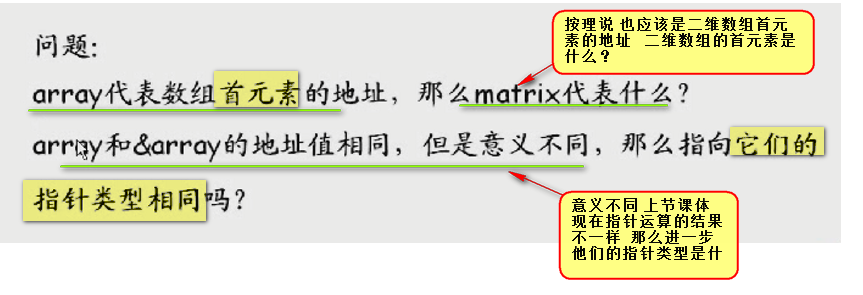
看一下下面这些声明是否合法？

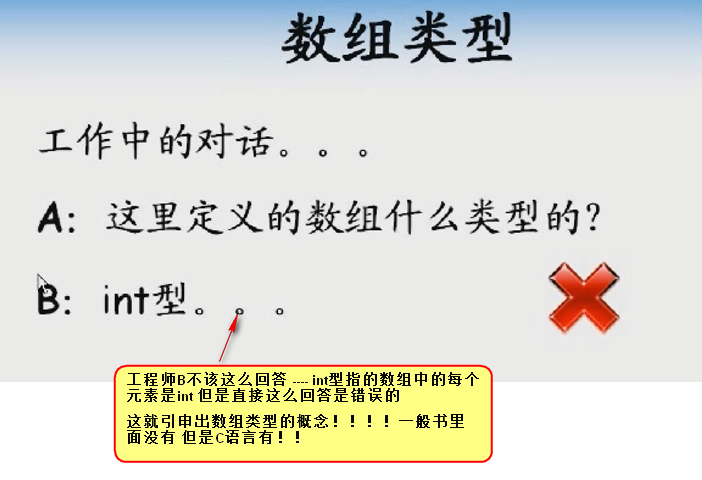


第一个和第二个都是合法的 一维数组和二维数组

pA指向了array这个数组的首地址

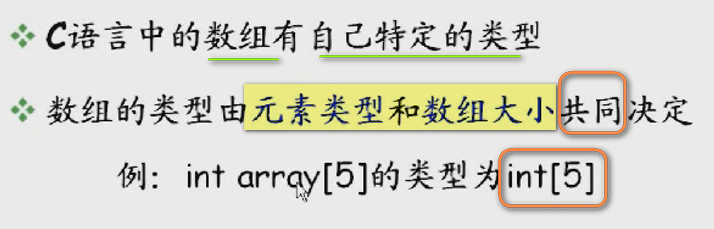
看上去都很正常





**C语言中 每个数组有自己特定的类型**

数组的类型 两个方面：数组元素的类型+数组的大小【貌似java也是这样的】

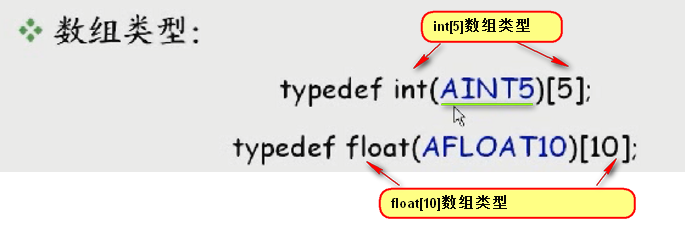


C语言中 如何定义一个数组类型？ ---- 重命名

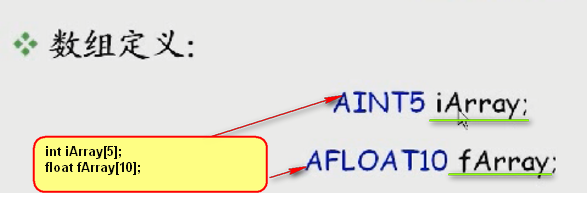


【数组的类型是 元素类型+数组大小功能决定的

那么 这里面的type就是元素类型 size就是数组大小 ---- 都体现出来了 ---- 对应的name就是别名 也就是对type[size]这种类型重命名的别名】



下面使用这个数组类型的别名来定义



这样 从AINT5看出来知道iArray的数组元素类型是5 每个元素是int类型

**这样的一个例子 --- 在C语言中对数组类型进行重命名**

**有了一个类型 ---- 如何通过这个类型定义指针？ ----- 数组自己特定的类型 ----- 有类型 就有指向这个类型的指针**

===🡺指向数组的指针 ----- 简称数组指针

*类比 如何定义一个int型的指针 然后 通过数组类型指向一个数组 -----*

*之前的做法： int array[5];*

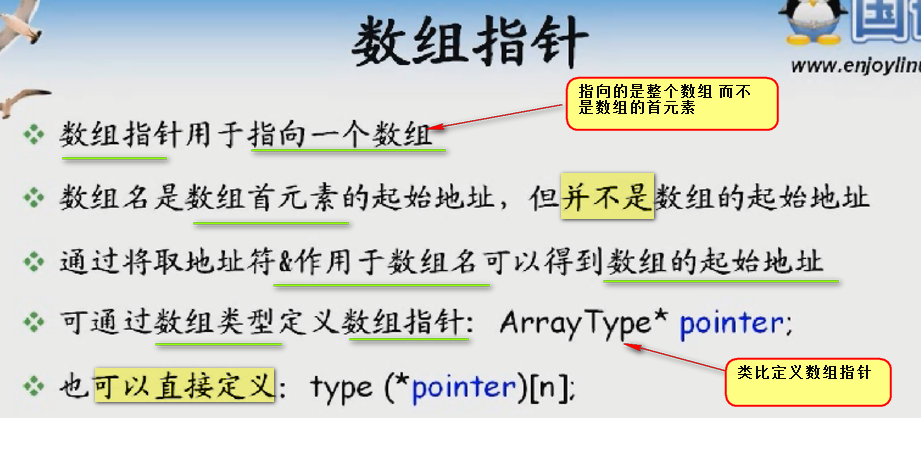
*Int\* pA = array;*

*这个pA不是数组指针么？ ---- 不是 pA指向的是int类型内存单元的地址*

*int\* pA = array; ---- array是数组首元素的地址 ---- 首元素明显是一个int类型*

*pA看上去指向array 但是完全不是数组指针*

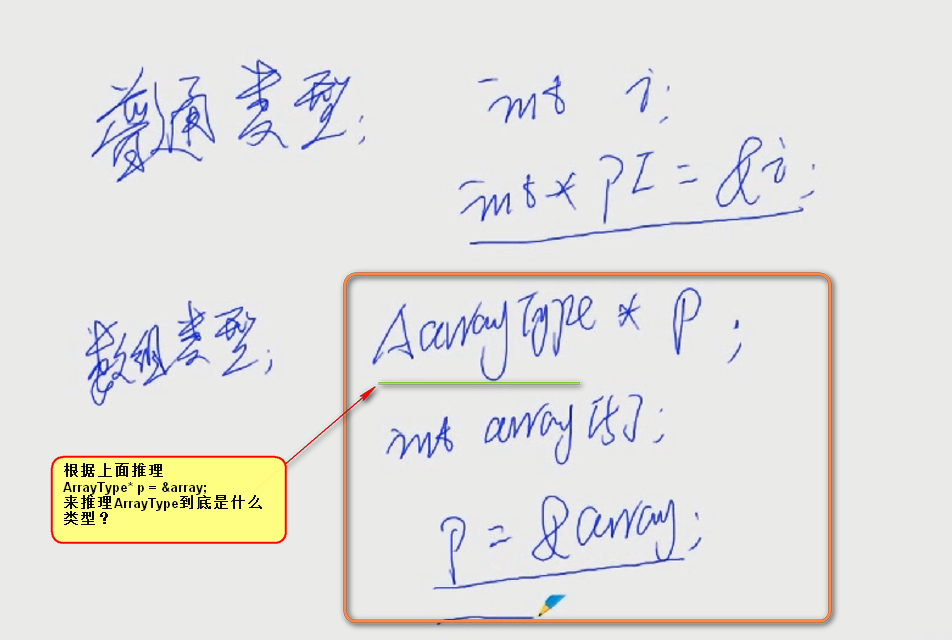
真正的数组指针在哪？



普通类型 定义 int i;

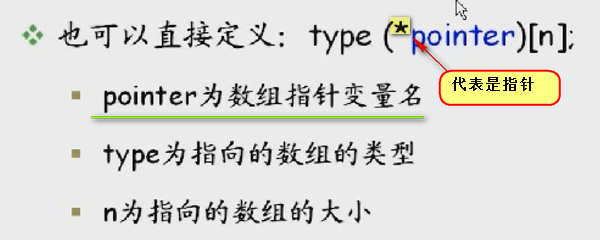
int\* pi = &i;

数组类型 定义 ArrayType 那么 定义一个数组指针



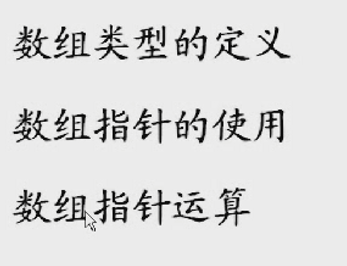
array代表数组首元素的地址 ---- &array 数组的地址赋值给一个数组指针 就是天经地义

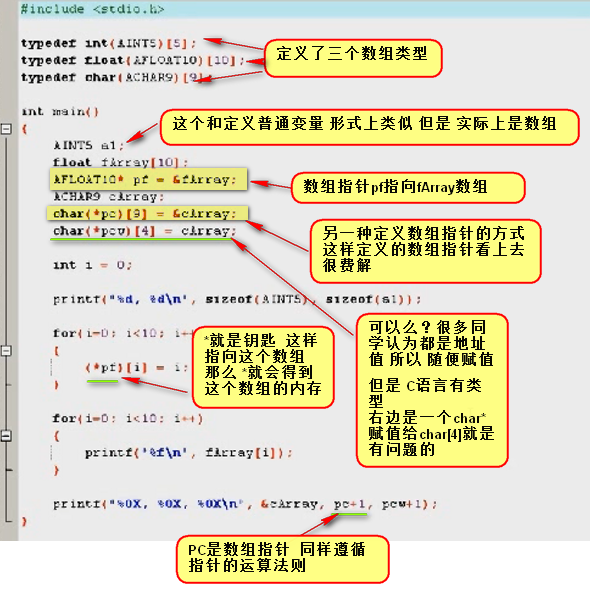
这样&array 代表数组指针的类型



通过这一行 把指针 和 数组类型都定义出来

以后工作中 喜欢type(\*pointer)[n]; 不推荐

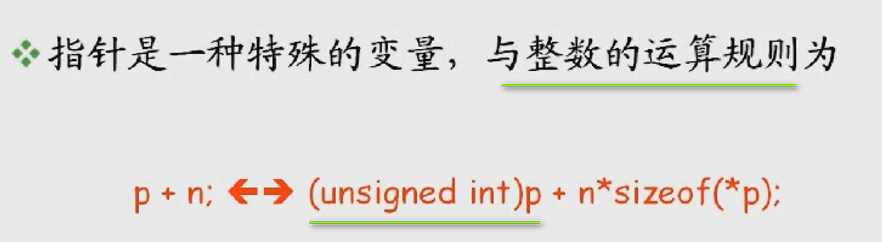




对pc+1进行展开



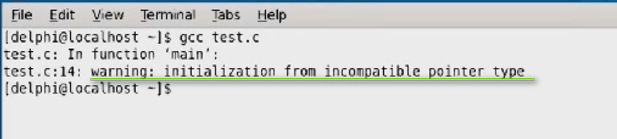
那么sizeof(\*pc) ---- 也就是pc 【

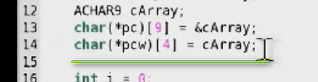
 --- \*p就是p指向的元素 ---- 那么 sizeof(\*p) 就是pc指向元素的大小

P的值本身是地址 被转换为无符号的整形

这里面pc+1 sizeof(\*pc) ---- pc指向元素是 char[9] ---- 那么sizeof(\*pc)=9B

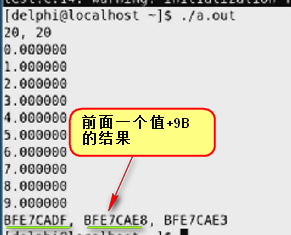
pc指向到数组最后一个元素后面去了】





但是 编译过去了

 注意 前面说过了 数组有自己的特定的类型 是数组中元素的类型+数组大小共同决定 ---- AINT5是Int[5]的别名 那么 4\*5=20个字节 sizeof(a1)值也是20B



虽然是数组指针 遵循前面的公式 数组指针遵循前面的公式

Pcw+1



1\*sizeof(\*pcw) ----- pcw指向的元素的大小 -----  正好相差5（这个地方赋值类型有问题）--- 这里面通过定义知道 pcw指向的是char[4] 大小是4

Pc指向的是char[9]

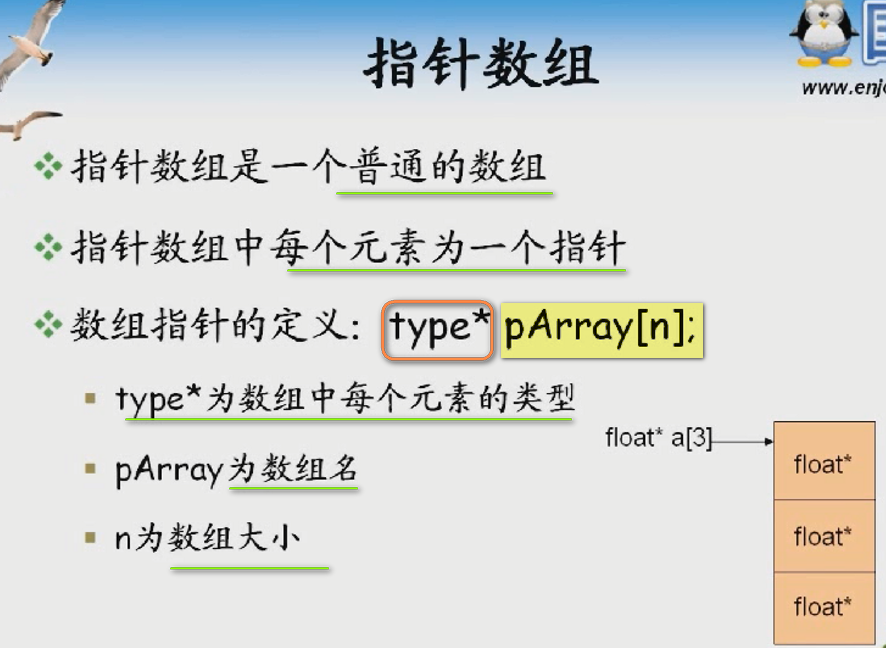
那么pcw就应该在pc前面 相差5

所以  后面两个相差5

再次证明 指针运算公式 适用于数组指针

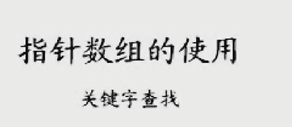
--------------------------

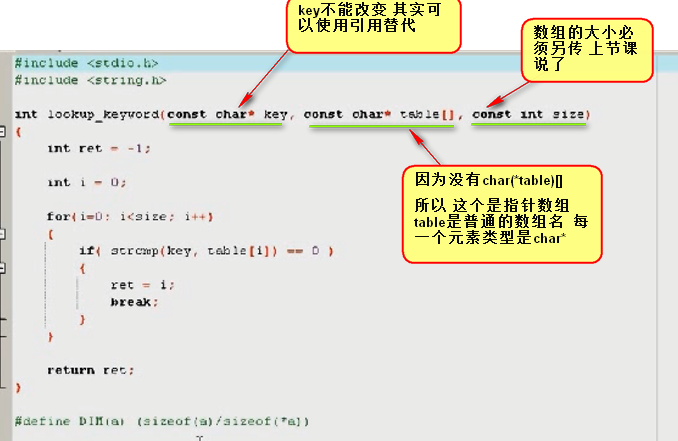
指针数组



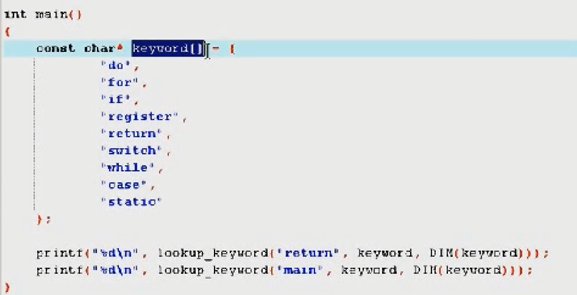
int(\*pc)[9] ---- pc指向int[9]数组

int\* pc[9] ----- pc就是一个普通数组名 里面每一个元素都是int\*类型的指针





 Main函数中





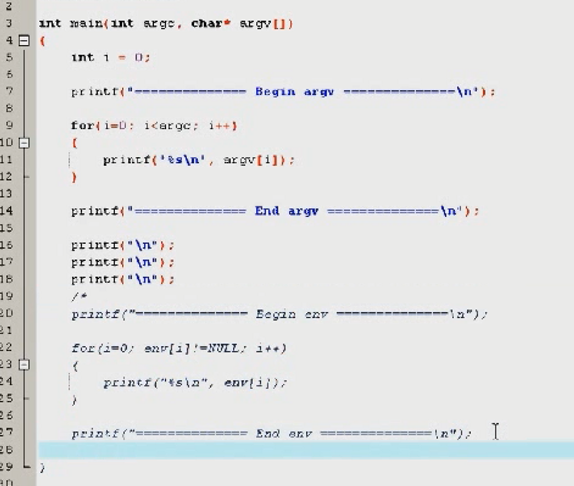
找不到 返回-1

=============== main函数的参数=============

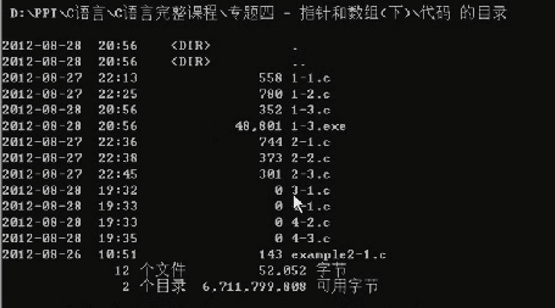
C语言中的main() 可以理解为操作系统调用的 ---- 很多时候说main是C的入口点 是函数 就被调用 --- main函数可以有参数 可以理解为操作系统调用的main函数 所以 可以给main传递参数

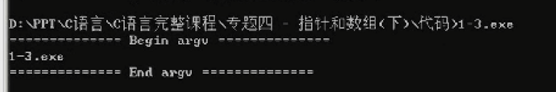


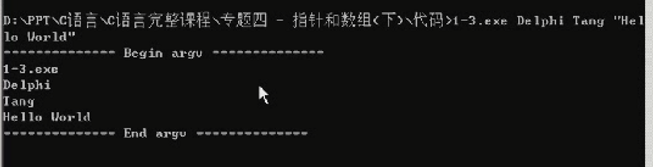
看看命令行参数的使用



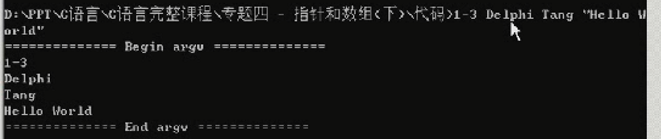
这个里面打印的是通过main方法传递的指针数组



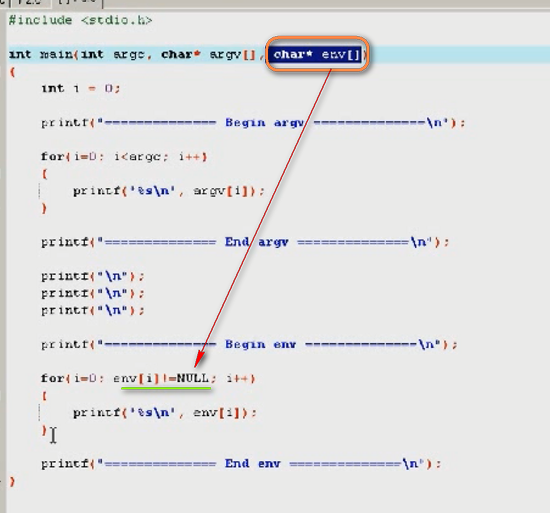




命令行全部打印出来 【argc是被自动注入】



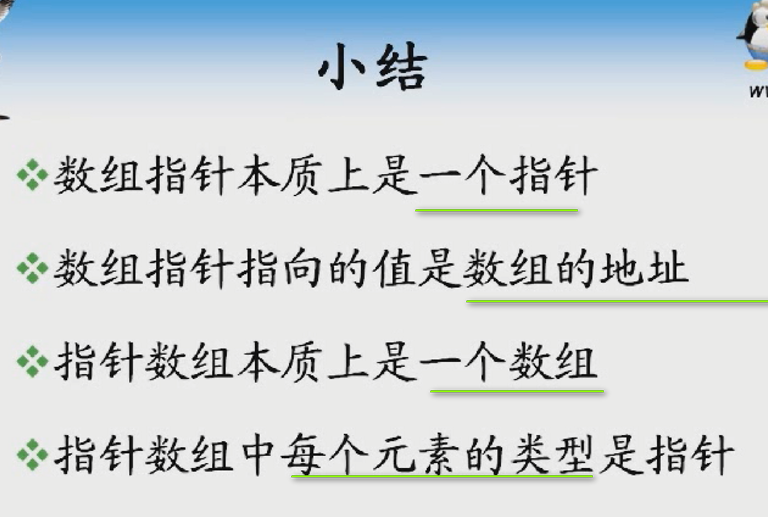
实现自己的命令

 这个代表当前os的环境变量 ----- 因为main是Os调用的 所以 os可以把环境变量全部注入进来



这个就是windows操作系统的环境变量 ---- 这个是OS层面的环境变量

============= 总结 ================



非常简单