计算机等级考试C语辅导:数组 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/285/2021_2022__E8_AE_A1_E7_AE_97_E6_9C_BA_E7_c97_285292.htm C语言处理数组的方式是它广受欢迎的原因之一。C语言对数组的处理是非常有效的，其原因有以下三点： 第一，除少数翻译器出于谨慎会作一些繁琐的规定外，C语言的数组下标是在一个很低的层次上处理的。但这个优点也有一个反作用，即在程序运行时你无法知道一个数组到底有多大，或者一个数组下标是否有效。ANSI/ISOC标准没有对使用越界下标的行为作出定义，因此，一个越界下标有可能导致这样几种后果： (1) 程序仍能正确运行； (2) 程序会异常终止或崩溃； (3) 程序能继续运行，但无法得出正确的结果； (4) 其它情况。 换句话说，你不知道程序此后会做出什么反应，这会带来很大的麻烦。有些人就是抓住这一点来批评C语言的，认为C语言只不过是一种高级的汇编语言。然而，尽管C程序出错时的表现有些可怕，但谁也不能否认一个经过仔细编写和调试的C程序运行起来是非常快的。 第二，数组和指针能非常和谐地在一起工作。当数组出现在一个表达式中时，它和指向数组中第一个元素的指针是等价的，因此数组和指针几乎可以互换使用。此外，使用指针要比使用数组下标快两倍(请参见9．5中的例子)。 第三，将数组作为参数传递给函数和将指向数组中第一个元素的指针传递给函数是完全 等价的。将数组作为参数传递给函数时可以采用值传递和地址传递两种方式，前者需要完整地拷贝初始数组，但比较安全；后者的速度要快得多，但编写程序时要多加小心。C 和ANSIC中都有const关键字，

利用它可以使地址传递方式和值传递方式一样安全。如果你想了解更多的细节，请参见2．4，8．6和第7章“ 指针和内存分配 ”开头部分的介绍。 数组和指针之间的这种联系会引起一些混乱，例如以下两种定义是完全相同的： void f(chara[MAX]) { /*．．． */ } void f(char *a) { /*．．． */ } 注意：MAX是一个编译时可知的值，例如用#define预处理指令定义的值。 这种情况正是前文中提到的第三个优点，也是大多数C程序员所熟知的。这也是唯一一种数组和指针完全相同的情况，在其它情况下，数组和指针并不完全相同。例如，当作如下定义 (可以出现在函数说明以外的任何地方)时： char a[MAX]. 系统将分配MAX个字符的内存空间。当作如下说明时： char *a. 系统将分配一个字符指针所需的内存空间，可能只能容纳2个或4个字符。如果你在源文件中作如下定义： char a[MAX]； 但在头文件作如下说明； extern char *a. 就会导致可怕的后果。为了避免出现这种情况，最好的办法是保证上述说明和定义的一致性，例如，如果在源文件中作如下定义： char a[MAX]； 那么在相应的头文件中就作如下说明， externchar a[]； 上述说明告诉头文件a是一个数组，不是一个指针，但它并不指示数组a中有多少个元素，这样说明的类型称为不完整类型。在程序中适当地说明一些不完整类型是很常见的，也是一种很好的编程习惯。 9．1 数组的下标总是从0开始吗? 是的，对数组a[MAX](MAX是一个编译时可知的值)来说，它的第一个和最后一个元素分别是a[o] 和aLMAX-1)。在其它一些语言中，情况可能有所不同，例如在BASIC语言中数组a[MAX]的元素是从a[1]到a[MAX]，在Pascal语言中则两种方式都可行。 注意：a[MAX]是一个有

效的地址，但该地址中的值并不是数组a的一个元素(见9。2)。 上述这种差别有时会引起混乱，因为当你说“ 数组中的第一个元素 ”时，实际上是指“ 数组中下标为。的元素 ”，这里的“ 第一个 ”的意思和“ 最后一个 ”相反。 尽管你可以假造一个下标从1开始的数组，但在实际编程中不应该这样做。下文将介绍这种技巧，并说明为什么不应该这样做的原因。因为指针和数组几乎是相同的，因此你可以定义一个指针，使它可以象一个数组一样引用另一个数组中的所有元素，但引用时前者的下标是从1开始的： /*don ’ t do this!!*/ int a0[MAX]， int *a1=a0-1. /*amp.a0[-1)完全有可能不是一个有效的地址(见9．3)。对于某些编译程序，你的程序可能根本不会出问题；在有些情况下，对于任何编译程序，你的程序可能都不会出问题；但是，谁能保证你的程序永远不会出问题呢? 第二，这种方式背离了C语言的常规风格。人们已经习惯了C语言中数组下标的工作方式，如果你的程序使用了另外一种方式，别人就很难读懂你的程序，而经过一段时间以后，连你自己都可能很难读懂这个程序了。 请参见： 9．2 可以使用数组后面第一个元素的地址吗? 9．3 为什么要小心对待位于数组后面的那些元素的地址呢? 9．2 可以使用数组后面第一个元素的地址吗? 你可以使用数组后面第一个元素的地址，但你不可以查看该地址中的值。对大多数编译程序来说，如果你写如下语句： int i，a[MAX]，j. 那么i和j都有可能存放在数组a最后一个元素后面的地址中。为了判断跟在数组a后面的是i还是j，你可以把i或j的地址和数组a后面第一个元素的地址进行比较，即判断"amp.a[MAX]"或"amp.a[MAX]"是否为真。这种方法通常可行，但不能保证。 问题的关键是：如果

你将某些数据存入a[MAX]中，往往就会破坏原来紧跟在数组a后面的数据。即使查看a[MAX]的值也是应该避免的，尽管这样做一般不会引出什么问题。 为什么在C程序中有时要用到&amp.a[MAX]呢?因为很多C程序员习惯通过指针遍历一个数组中的所有元素，即用 for(i＝0.i{ /*do something*/ } 代替 for(p＝a. p{ /*do something*/ } 这种方式在已有的C程序中是随处可见的，因此ANSIC标准规定这种方式是可行的。 100Test下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com