C语言之指针、数组和函数 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/448/2021_2022_C_E8_AF_AD_E8_A8_80_E4_B9_8B_c97_448761.htm 基本解释 1、指针的本质是一个与地址相关的复合类型，它的值是数据存放的位置（地址）；数组的本质则是一系列的变量。 2、数组名对应着（而不是指向）一块内存，其地址与容量在生命期内保持不变，只有数组的内容可以改变。指针可以随时指向任意类型的内存块，它的特征是“ 可变 ”，所以我们常用指针来操作动态内存。 3、当数组作为函数的参数进行传递时，该数组自动退化为同类型的指针。 问题：指针与数组 听说char a[]与char *a是一致的，是不是这样呢？ 答案与分析： 指针和数组存在着一些本质的区别。当然，在某种情况下，比如数组作为函数的参数进行传递时，由于该数组自动退化为同类型的指针，所以在函数内部，作为函数参数传递进来的指针与数组确实具有一定的一致性，但这只是一种比较特殊的情况而已，在本质上，两者是有区别的。请看以下的例子：char a[] = "Hi, pig!".char *p = "Hi, pig!".上述两个变量的内存布局分别如下： 数组a需要在内存中占用8个字节的空间，这段内存区通过名字a来标志。指针p则需要4个字节的空间来存放地址，这4个字节用名字p来标志。其中存放的地址几乎可以指向任何地方，也可以哪里都不指，即空指针。目前这个p指向某地连续的8个字节，即字符串“ Hi, pig! ”。 另外，例如：对于a[2]和p[2]，二者都返回字符‘ i ’，但是编译器产生的执行代码却不一样。对于a[2]，执行代码是从a的位置开始，向后移 动2两个字节，然后取出其中的字符。对于p[2]，执行

代码是从p的位置取出一个地址，在其上加2，然后取出对应内存中的字符。 问题：数组指针 为什么在有些时候我们需要定义指向数组而不是指向数组元素的指针？如何定义？ 答案与分析： 使用指针，目的是用来保存某个元素的地址，从而来利用指针独有的优点，那么在元素需要是数组的情况下，就理所当然要用到指向数组的指针，比如在高维需要动态生成情况下的多维数组。 定义例子如下： int (*pElement)[2]。 下面是一个例子： int array[2][3] = {{1，2，3}，{4，5，6}}.int (*pa)[3]. //定义一个指向数组的指针 pa = amp.符号能够体现pa的含义，表示是指向数组的指针printf ("%d", (*pa)[0]). //将打印array[0][0]，即1 pa ； // 猜一猜，它指向谁？array[1]？对了！printf ("%d", (*pa)[0]). // 将打印array[1][0]，即4上述这个例子充分说明了数组指针一种指向整个数组的指针的定义和使用。 需要说明的是，按照我们在第四篇讨论过的，指针的步进是参照其所指对象的大小的，因此，pa 将整个向后移 动一个数组的尺寸，而不是仅仅向后移 动一个数组元素的尺寸。 问题：指针数组 有如下定义：struct UT_TEST_STRUCT *pTo[2][MAX_NUM]. 请分析这个定义的意义，并尝试说明这样的定义可能有哪些好处？ 答案与分析： 前面我们谈了数组指针，现在又提到了指针数组，两者形式很相似，那么，如何区分两者的定义呢？分析如下： 数组指针是：指向数组的指针，比如 int (*pA)[5]。 指针数组是：指针构成的数组，比如int *pA[5]。 至于上述指针数组的好处，大致有如下两个很普遍的原因： a)、各个指针内容可以按需要动态生成，避免了空间浪费。 b)、各个指针呈数组形式排列，索引起来非常方便。 在实际编程中，选择使用指针数组大多都是想要获得

如上两个好处。 问题：指向指针的指针 在做一个文本处理程序的时候，有这样一个问题：什么样的数据结构适合于按行存储文本？ 答案与分析： 首先，我们来分析文本的特点，文本的主要特征是具有很强的动态性，一行文本的字符个数或多或少不确定，整个文本所拥有的文本行数也是不确定的。这样的特征决定了用固定的二维数组存放文本行必然限制多多，缺乏灵活性。这种场合，使用指向指针的指针有很大的优越性。 现实中我们尝试用动态二维数组（本质就是指向指针的指针）来解决此问题： 图示是一个指针数组。所谓动态性指横向（对应每行文本的字符个数）和纵向（对应整个文本的行数）两个方向都可以变化。 就横向而言，因为指针的灵活性，它可以指向随意大小的字符数组，实现了横向动态性。 就竖向而言，可以动态生成及扩展需要的指针数组的大小。 下面的代码演示了这种动态数组的用途：// 用于从文件中读取以 \0结尾的字符串的函数extern char *getline(FILE *pFile).FILE *pFile.char **ppText = NULL. // 二维动态数组指针 char *pCurrText = NULL； // 指向当前输入字符串的指针ULONG ulCurrLines = 0；ULONG ulAllocedLines = 0； while（p = getline(pFile)）{ if (ulCurrLines >= ulAllocedLines) { // * 当前竖向空间已经不够了，通过realloc对其进行扩展。 ulAllocedLines = 50. // 每次扩展50行。 ppText = realloc (ppText, ulAllocedLines * (char *)). if (NULL == ppText) { return. // 内存分配失败，返回 } } ppText[ulCurrLines] = p. // 横向 “ 扩展 ”，指向不定长字符串 } 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