C语言程序设计(第4章函数)3 PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/134/2021_2022_C_E8_AF_AD_E8_A8_80_E7_A8_8B_c97_134850.htm 4.3 函数的调用与参数 如果一个函数要使用参数，它就必须定义接受参数值的变量。 4.3.1 形式参数与实际参数 函数定义时填入的参数我们称之为形式参数，简称形参，它们同函数内部的局部变量作用相同。形参的定义是在函数名之后和函数开始的花括号之前。调用时填入的参数，我们称之为实际参数，简称实参。 必须确认所定义的形参与调用函数的实际参数类型一致，同时还要保证在调用时形参与实参的个数出现的次序也要一一对应。如果不一致，将产生意料不到的结果。与许多其它高级语言不同，（是健壮的，它总要做一些甚至你不希望的事情，几乎没有运行时错误检查，完全没有范围检测。作为程序员，必须小心行事以保证不发生错误，安全运行。 4.3.2 赋值调用与引用调用 一般说来，有两种方法可以把参数传递给函数。第一种叫做“ 赋值调用 ”（call by value），这种方法是把参数的值复制到函数的形式参数中。这样，函数中的形式参数的任何变化不会影响到调用时所使用的变量。 把参数传递给函数的第二种方法是“ 引用调用 ”（call by reference）。这种方法是把参数的地址复制给形式参数，在函数中，这个地址用来访问调用中所使用的实际参数。这意味着，形式参数的变化会影响调用时所使用的那个变量(详细内容请参见后续章节)。 除少数情况外，C语言使用赋值调用来传递参数。这意味着，一般不能改变调用时所用变量的值。请看例4 - 9。 [例4 - 9］ main ( ) { int t =10 ； printf("%d %d ",sqr(t),t). /* sqr(t)是

函数调用，t是实参* / } int sqr(x) /* 函数定义，x是形式参数* / int x；{ x = x * x. return (x). } 在这个例子里，传递给函数sqr( )的参数值是复制给形式参数x的，当赋值语句x = x * x执行时，仅修改局部变量x。用于调用sqr( )的变量t，仍然保持着值10。 执行程序： 100 10 切记，传给函数的只是参数值的复制品。所有发生在函数内部的变化均无法影响调用时使用的变量。 4.4 递归 C语言函数可以自我调用。如果函数内部一个语句调用了函数自己，则称这个函数是“ 递归 ”。递归是以自身定义的过程。也可称为“ 循环定义 ”。 递归的例子很多。例如定义整数的递归方法是用数字1，2，3，4，5，6，7，8，9加上或减去一个整数。例如，数字1 5是7 8；数字2 1是9 1 2； 数字1 2是9 3。 一种可递归的计算机语言，它的函数能够自己调用自己。一个简单的例子就是计算整数阶乘的函数factor( )数N的阶乘是1到N之间所有数字的乘积。例如3的阶乘是1×2×3，即是６。 factor( )和其等效函数fact( )如例4 - 1 0所示。 [例4 - 1 0] factor(n) /* 递归调用方法* / int n；{ int answer. if (n==1) return (1). answer=factor(n-1) * n； /* 函数自身调用* / return(answer). } [例4 - 11] fact(n) /* 非递归方法* / int n. { int t，a n s w e r. answer = 1. for (t=1. t answer = answer * t. return(answer). } 非递归函数fact( )的执行应该是易于理解的。它应用一个从1开始到指定数值结束的循环。 在循环中，用“ 变化 ”的乘积依次去乘每个数。 factor( )的递归执行比fact( )稍复杂。当用参数1调用factor( )时，函数返回1；除此之外的其它值调用将返回factor(n-1) * n这个乘积。为了求出这个表达式的值，用（ n - 1）调用factor( )一直到n等于1，调用开始返回。 计算2的阶乘时对factor( )的首次调用引起了以参数1

对factor( )的第二次调用。这次调用返回1，然后被2乘（n的初始值），答案是2（把printf( )语句插入到factor ( )中，察看各级调用及其中间答案，是很有趣的）。