C语言程序设计(第4章函数)2 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_134860.htm 4.2 函数的作用域规则 “ 语言的作用域规则 ” 是一组确定一部分代码是否 “ 可见 ” 或可访问另一部分代码和数据的规则。 C语言中的每一个函数都是一个独立的代码块。一个函数的代码块是隐藏于函数内部的，不能被任何其它函数中的任何语句（除调用它的语句之外）所访问（例如，用go to语句跳转到另一个函数内部是不可能的）。构成一个函数体的代码对程序的其它部分来说是隐蔽的，它既不能影响程序其它部分，也不受其它部分的影响。换言之，由于两个函数有不同的作用域，定义在一个函数内部的代码数据无法与定义在另一个函数内部的代码和数据相互作用。 C语言中所有的函数都处于同一作用域级别上。这就是说，把一个函数定义于另一个函数内部是不可能的。 4.2.1 局部变量 在函数内部定义的变量成为局部变量。在某些C语言教材中，局部变量称为自动变量，这就与使用可选关键字auto定义局部变量这一作法保持一致。局部变量仅由其被定义的模块内部的语句所访问。换言之，局部变量在自己的代码模块之外是不可知的。切记：模块以左花括号开始，以右花括号结束。 对于局部变量，要了解的最重要的东西是：它们仅存在于被定义的当前执行代码块中，即局部变量在进入模块时生成，在退出模块时消亡。 定义局部变量的最常见的代码块是函数。例如，考虑下面两个函数。 [例4 - 5] func1( ) { int x. /* 可定义为auto int x. */ x = 10 . } func2( ) { int x. /* 可定义为auto int x. */ x = -1999. } 整数变量x被定义了两次

，一次在func1( )中，一次在func2( )中。func1( )和func2( )中的x互不相关。其原因是每个x作为局部变量仅在被定义的块内可知。 语言中包括了关键字auto，它可用于定义局部变量。但自从所有的非全局变量的缺省值假定为auto以来，auto就几乎很少使用了，因此在本书所有的例子中，均见不到这一关键字。 在每一函数模块内的开始处定义所有需要的变量，是最常见的作法。这样做使得任何人读此函数时都很容易，了解用到的变量。但并非必须这样做不可，因为局部变量可以在任何模块中定义。为了解其工作原理，请看下面函数。 [例4 - 6] f( ) { int t. scanf("%d", &amp.t). if(t == 1) { char s[80]. /* 此变量仅在这个块中起作用* / printf("enter name:"). gets(s). /* 输入字符串* / process(s). /* 函数调用* / } } 这里的局部变量s就是在i f块入口处建立，并在其出口处消亡的。因此s仅在i f块中可知，而在其它地方均不可访问，甚至在包含它的函数内部的其它部分也不行。 在一个条件块内定义局部变量的主要优点是仅在需要时才为之分配内存。这是因为局部变量仅在控制转到它们被定义的块内时才进入生存期。虽然大多数情况下这并不十分重要，但当代码用于专用控制器（如识别数字安全码的车库门控制器）时，这就变得十分重要了，因为这时随机存储器（ＲＡＭ）极其短缺。 由于局部变量随着它们被定义的模块的进出口而建立或释放，它们存储的信息在块工作结束后也就丢失了。切记，这点对有关函数的访问特别重要。当访问一函数时，它的局部变量被建立，当函数返回时，局部变量被销毁。这就是说，局部变量的值不能在两次调用之间保持。 4.2.2 全局变量 与局部变量不同，全局变量贯穿整个程序，并且可被任何一个模块使用。它们在整个程

序执行期间保持有效。全局变量定义在所有函数之外，可由函数内的任何表达式访问。在下面的程序中可以看到，变量count定义在所有函数之，函数main( )之前。但其实它可以放置在任何第一次被使用之前的地方，只要不在函数内就可以。实践表明，定义全局 变量的最佳位置是在程序的顶部。[例4 - 7] int count. /*count 是全局变量* / main( ) { count = 100.