C语言编程常见问题解答之编译预处理 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/135/2021_2022_C_E8_AF_AD_E8_A8_80_E7_BC_96_c97_135885.htm 本章集中讨论与预处理程序有关的问题。在编译程序对程序进行通常的编译之前，要先运行预处理程序。可能你以前没有见过这个程序，因为它通常在幕后运行，程序员是看不见它的，然而，这个程序非常有用。 预处理程序将根据源代码中的预处理指令来修改你的程序。预处理指令(如#define)为预处理程序提供特定的指令，告诉它应该如何修改你的源代码。预处理程序读入所有包含的文件和待编译的源代码，经过处理生成源代码的预处理版本。在该版本中，宏和常量标识符已用相应的代码和值代替。如果源代码中包含条件预处理指令(如#if)，预处理程序将先判断条件，然后相应地修改源代码。 预处理程序有许多非常有用的功能，例如宏定义，条件编译，在源代码中插入预定义的环境变量，打开或关闭某个编译选项，等等。对专业程序员来说，深入了解预处理程序的各种特征，是创建快速和高效的程序的关键之一。 在阅读本章时，请记住本章采用的一些技术(以及所提到的一些常见陷阱)，以便更好地利用预处理程序的各种功能。 5．1 什么是宏(macro)?怎样使用宏? 宏是一种预处理指令，它提供了一种机制，可以用来替换源代码中的字符串，宏是用“ #define"语句定义的，下面是一个宏定义的例子： #define VERSIONSTAMP "1．02" 上例中所定义的这种形式的宏通常被称为标识符。在上例中，标识符VERSION_STAMP即代表字符串"1．02"在编译预处理时，源代码中的每个VERSION_STAMP标识符都将被字符串“ 1

. 02 ” 替换掉。 以下是另一个宏定义的例子： #define CUBE(x)((x)，(x)*(x)) 上例中定义了一个名为CUBE的宏，它有一个参数x。CUBE宏有自己的宏体，即((x)*(x)*(x))在编译预处理时，源代码中的每个CUBE(x)宏都将被((x)*(x)*(x))替换掉。 使用宏有以下几点好处； (1)在输入源代码时，可省去许多键入操作。 (2)因为宏只需定义一次，但可以多次使用，所以使用宏能增强程序的易读性和可靠性。 (3)使用宏不需要额外的开销，因为宏所代表的代码只在宏出现的地方展开，因此不会引起程序中的跳转。 (4)宏的参数对类型不敏感，因此你不必考虑将何种数据类型传递给宏。 需要注意的是，在宏名和括起参数的括号之间绝对不能有空格。此外，为了避免在翻译宏时产生歧义，宏体也应该用括号括起来。例如，象下例中这样定义CUBE宏是不正确的： denne CUBE(x) x * x * x 对传递给宏的参数也要小心，例如，一种常见的错误就是将自增变量传递给宏，请看下例： #include #include CUBE(x) (x * x * x) void main (void). void main (void) { int x, y. x = 5. y = CUBE(x). printfC ’ y is %d\n" . y). } 在上例中，y究竟等于多少呢?实际上，y既不等于125(5的立方)，也不等于336(6* 7*8)，而是等于512。因为变量x被作为参数传递给宏时进行了自增运算，所以上例中的CUBE宏实际上是按以下形式展开的： y = ((x) * (x) * (x)) ； 这样，每次引用x时，x都要自增，所以你得到的结果与你预期的结果相差很远，在上例中，由于x被引用了3次，而且又使用了自增运算符，因此，在展开宏的代码时，x实际上为8，你将得到8的立方，而不5的立方。 上述错误是比较常见的，作者曾亲眼见过有多年C语言编程经验的人犯这种错误。因为在程序中检查这种错误是非常费劲的，

所以你要给予充分的注意。你最好试一下上面的例子，亲眼看一下那个令人惊讶的结果值(512)。 宏也可使用一些特殊的运算符，例如字符串化运算符“ # ”和。连接运算符“ ## ”。“ # ”运算符能将宏的参数转换为带双引号的字符串，请看下例： define DEBUG_VALUE(v) printf(#v"is equal to %d．\n"，v) 你可以在程序中用DEBUG_VALUE宏检查变量的值，请看下例: int x＝20； DEBUG_VALUE(x). 上述语句将在屏幕上打印"x is equal to 20"。这个例子说明，宏所使用的“ # ”运算符是一种非常方便的调试工具。“ ## ”运算符的作用是将两个独立的字符串连接成一个字符串，详见5．16。