函数的可变参数详谈 PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/459/2021_2022__E5_87_BD_E6_95_B0_E7_9A_84_E5_c97_459818.htm 可变参数的英文表示为:variable argument.它在函数的定义时,用三个点号.表示,用逗号与其它参数分隔.可变参数的特点:不像固定参数那样一一对应,也不像固定参数有固定的参数类型和参数名称.可变参数中个数不定可是传入的是一个参数也可以是多个.可变参数中的每个参数的类型可以不同,也可以相同.可变参数的每个参数并没有实际的名称与之相对应.由此可见,可变参数的形式非常自由而富有弹生.因些,它给那些天才程序员有更大地想象和发挥空间.然而,更多地自由,同样也加大操作上的难度.以下就对可变参数的几个方面作一定的介绍. 1)可变参数的存储形式.大家都知道,一般函数的形参属于局部变量.而局部变量就是存储在内存的栈区(所谓的栈区:由编译器自动分配释放,存放函数的参数值,局部变量的值等。其操作方式类似于数据结构中的栈。).可变参数也是存储在内存栈区.在对函数的形参存储的时侯,编译器是从函数的形参的右边到左边逐一地压栈,这样保证了栈顶是函数的形参的第一个参数(从左到右数).而80x86平台下的内存分配顺序是从高地址内存到低地址内存.因此,函数的形参在内存的存储形式如下图(以fun(int var1,int var2,...,int var3,int var4)为例):栈区:|栈顶 低地址|第一个固定参数var1|可变参数前的第一个固定参数var2|可变参数的第一个参数|...|可变参数的最后一个参数|函数的倒数第二个固定参数var3|函数的最后一个固定参数var4|...|函数的返回地址|...|栈底 高地址2)使用可变参数所用到头文件和相关宏说明在此,以TC2.0编译

器为参考对象来说明.可变参数的相关定义在TC2.0的名为"STDARG.H"的头文件中.此文件为:/* stdarg.hDefinitions for ACCESSing parameters in functions that accepta variable number of arguments.Copyright (c) Borland International 1987,1988All Rights Reserved.*/#if __STDC__#define _Cdecl#else#define _Cdecl cdecl#endif#if !defined(__STDARG)#define __STDARGtypedef void *va_list.#define va_start(ap, parmN) (ap = ...)#define va_arg(ap, type) (*((type *)(ap)))#define va_end(ap)#define _va_ptr (...)#endif以上为"STDARG.H"的内容.该文件定义了使用可变参数所用到的数据类型:typedef void *va_list.va_start(ap,parmN)起到初始化,使用得ap指向可变参数的第一个参数.ap的类型为va_list,parmN为可变参数的前面一个固定参数.va_arg(ap,type)获得当前ap所指向的参数,并使ap指向可变参数的下一个参数,type为需要获得的参数的类型.va_end(ap) 结束可变参数获取.3)可变参数的使用实例实例目的:用可变参数来实现个数不定的字符串的传递,并显示传递过来的字符串.#include#include#includevoid tVarArg(int num,...)./*num为可变参数的个数*/int main(void){clrscr().tVarArg(5,"Hello! ","My ","name ","is ","neverTheSame.\n").tVarArg(8,"This ","is ","an ","example ","about ","variable-argument ","in ","funtion").getch().return 0.}void tVarArg(int num,...){va_list argp. /*定义一个指向可变参数的变量*/va_start(argp,num). /*初始化,使用argp指向可变参数的第一个参数*/while(--num>=0) printf("%s",(va_arg(argp,char*))./*va_arg(argp,char*)获得argp所指向的参数, 并使用argp指向下一个参数,char*使用所获得的参

数的类型转换为char*型.*/va_end(argp). /*结束可变参数获取*/return .}4)可变参数的使用需要注意的问题1.每个函数的可变参数至多有一个.2.va_start(ap,parmN)中parmN为可变参数前的一个固定参数.3.可变参数的个数不确定,完全由程序约定.4.可变参数的类型不确定,完全由va_arg(ap,type)中的type指定,然后就把参数的类型强制转换.而printf()中不是实现了识别参数吗?那是因为函数 printf()是从固定参数format字符串来分析出参数的类型,再调用va_arg 的来获取可变参数的.也就是说,你想实现智能识别可变参数的话是要通 过在自己的程序里作判断来实现的. 5.编译器对可变参数的函数的原型检查不够严格,对编程人员要求很高.