

C 高级实用程序设计

王士元 编著



清华大学出版社

096037

C 高级实用程序设计

王士元 编著



清华大学出版社

(京)新登字 158 号

内 容 简 介

本书针对目前应用程序设计的热点,如中、英文菜单设计,画图,动画,中断程序,程序的驻留,屏幕图形的存取、打印,C 程序汉字显示技术,C 语言与汇编语言的混合编程,C 语言与 FoxBASE 的混合编程等进行设计示范,附有大量示例程序和注释。本书也用了部分篇幅对高级程序设计涉及的硬件及 C 中的文件、指针、内存分配、图形适配器等内容进行分析,并简单介绍了实用程序编程方法。本书适用于理工科本科生、研究生和广大计算机应用人员。

JS200/18

版权所有,翻印必究。

本书封面贴有清华大学出版社激光防伪标签,无标签者不得销售。

图书在版编目(CIP)数据

C 高级实用程序设计/王士元编著. —北京:清华大学出版社,1996.3
ISBN 7-302-02063-9

I. C… II. 王… III. C 语言-程序设计 IV. TP312C

中国版本图书馆 CIP 数据核字(95)第 23619 号

C 高级实用程序设计

出版者:清华大学出版社(北京清华大学校内,邮编 100084)

印刷者:北京大中印刷厂

发行者:新华书店总店北京科技发行所

开 本:787×1092 1/16 印张:32 字数:793 千字

版 次:1996 年 6 月第 1 版 1997 年 2 月第 2 次印刷

书 号:ISBN 7-302-02063-9/TP·959

印 数:6001—12000

定 价:32.00 元

前 言

C 语言使用愈来愈普及,愈来愈深层次,许多读者,由于对知识的渴望或实际工作的需要,希望能更深入的了解一些 C 语言与 PC 机硬件的联系,希望能对 C 语言编程时遇到的一些技术难点给予帮助,这些难点诸如对 PC 机内存的分配与管理,80x86 CPU 内部寄存器的使用,BIOS 和 DOS 的调用,指针的分类与使用,堆和栈的结构及使用,文件的操作,对 PC 机输入输出接口的编程,中断处理程序的编写,驻留程序的设计,各种图形的绘制技术,中英文菜单程序设计,屏幕图形的存取与打印技术,C 程序中的汉字使用及显示技术,C 语言与汇编语言的混合编程,C 语言与 FoxBASE(dBASE)的接口技术等,这些问题也是目前 C 程序应用设计中的一些热点,它们涉及到 PC 机的硬件结构,中英文 DOS 系统和编程时积累的实践经验。

由于目前较多地看到的是一些有关 C 语言的教科书和参考书,对于上面提到的程序设计中的热点问题(它们应用于科研、教学、工程项目和软件产品中)的书较少见,这也是目前广大读者急需的。由于这方面的程序设计往往和 PC 机的系统结构联系在一起,属于 C 程序深层次的应用,故而称为高级实用程序设计。

笔者作为一个教师,一个科研工作者,一个较早搞计算机设计与应用且目前又直接讲授语言的人,了解一些学生及搞计算机应用开发的人们急需的要求,为此将自己的一些实践、体会、心得及参考到的新内容写到这本书中,奉献给读者,以期能对进行 C 程序设计及软硬件开发人员能有所帮助。

由于 Turbo C 2.0 和 Microsoft 5.0~7.0 是目前大家较熟悉且又比较流行的 C 语言编译系统,而 Turbo C 有较良好的集成开发环境,所以本书以 Turbo C 2.0 为开发环境,书中的程序均在此环境中进行了调试,源程序集中在一张高密盘上,这些程序也可以方便地移植到别的编译系统上。

由于作者水平所限,加之时间仓促,书中错误及不妥之处在所难免,希望广大读者及专家批评指正。

作者在此对引入我出书的引路者——王家骅教授表示感谢,乔晓原、乔圆圆、王津涛、刘振实各位老师给予了不少帮助,李乐天、朱恩愈教授审阅了本书,提出了宝贵意见,在此深表谢意。

作 者

1994 年 12 月于南开大学

目 录

第 1 章 概述	1	2.7.1 数据类型与存储字节数	33
1.1 C 语言的发展状况	1	2.7.2 变量的存储类型	34
1.2 C 语言的编程格式	2		
1.2.1 C 语言程序的结构特点	2	第 3 章 关于 DOS 的说明及 BIOS 和	
1.2.2 模块化的程序设计	7	DOS 调用	38
1.2.3 大程序的设计风格	9	3.1 关于 DOS 的说明	38
1.3 Turbo C 2.0 的程序设计		3.1.1 DOS 的基本组成	38
开发过程	10	3.1.2 关于 DOS 的启动	40
1.3.1 Turbo C 集成开发环境	11	3.1.3 关于 BIOS	40
1.3.2 Turbo C 的命令行		3.1.4 中断向量表	41
编译连接	11	3.2 BIOS	41
		3.2.1 ROM BIOS 的使用	42
第 2 章 PC 机存储器结构及变量存储		3.2.2 视频 BIOS	42
方式	13	3.2.3 BIOS 的调用	42
2.1 PC 机存储器结构	13	3.3 DOS 调用	43
2.1.1 系统存储器(System Memory)	13	3.4 BIOS 和 DOS 系统调用函数	44
.....	13	3.4.1 int86()函数	44
2.1.2 扩展存储器(Extended Memory)	14	3.4.2 int86x()函数	46
.....	14	3.4.3 intdos()函数	47
2.1.3 关于 HMA(高端存储器)的使用	14	3.4.4 intdosx()函数	48
.....	14	3.4.5 intr()函数	49
2.1.4 扩充存储器(Expanded			
Memory)	15	第 4 章 指针、函数	51
2.2 存储器的分段与物理地址的形成	16	4.1 指针的赋值	51
.....	16	4.1.1 指向一个简单变量的指针	52
2.3 与地址操作有关的几个宏	18		52
2.4 寄存器与伪变量	19	4.1.2 指向数组的指针	52
2.4.1 80x86 的内部寄存器	19	4.1.3 指向函数的指针	55
2.4.2 伪变量	22	4.1.4 指向结构的指针	57
2.4.3 伪变量的使用	23	4.2 指针数组	59
2.5 保护虚地址方式下的段和偏移	24	4.3 二级指针	60
2.6 扩展存储器的使用实例	28	4.4 指针型函数	62
2.6.1 程序 1	28	4.5 指针的分类	63
2.6.2 程序 2	31	4.5.1 近程指针(near 型)	64
2.7 变量的存储方式	32	4.5.2 远程指针(far 型)	64

4.5.3 巨指针(huge)	65	6.2 流与文件	128
4.6 函数	66	6.3 标准文件的输入输出操作	129
4.6.1 函数的说明	67	6.3.1 标准文件输入输出	130
4.6.2 函数的类型与定义	68	6.3.2 关于 FILE 数据结构	130
4.6.3 函数的调用	70	6.3.3 标准文件打开函数 fopen()	131
4.6.4 函数的参数传送	72	6.3.4 标准文件关闭函数 fclose()	135
4.6.5 main() 函数中的参数	84	6.3.5 标准文件的读写	136
4.7 函数的递归调用	85	6.3.6 清除和设置文件缓冲区的函数	138
4.8 函数的组织	87	6.3.7 应用例	139
第 5 章 内存模式与动态存储结构	88	6.3.8 文件的随机读写函数	141
5.1 内存模式(编译模式)	88	6.4 非标准的文件输入输出操作	147
5.1.1 微小模式(Tiny)	88	6.4.1 建立一个新文件或重写一个已 有文件的函数 creat(), _creat(), creatnew()	148
5.1.2 小模式(Small)	89	6.4.2 非标准的文件打开函数	149
5.1.3 中模式(Medium)	89	6.4.3 关于系统标准输入输出设备的 文件代号	151
5.1.4 紧凑模式(Compact)	89	6.4.4 非标准文件的关闭函数 close()	151
5.1.5 大模式(Large)	90	6.4.5 非标准文件的读写函数	151
5.1.6 巨模式(Huge)	91	6.4.6 删除文件函数 unlink()	154
5.2 各内存模式下缺省段名和堆的分配	91	6.4.7 移动文件指针的函数 lseek()	154
5.3 栈的结构	93	第 7 章 I/O 接口的输入输出	158
5.4 堆的结构	95	7.1 I/O 接口的寻址方式	160
5.5 堆管理函数	96	7.2 I/O 接口的输入输出函数	161
5.5.1 示例 1—使用 coreleft() 和 malloc() 函数例	97	7.2.1 接口输入函数	162
5.5.2 示例 2—使用 farcalloc()	98	7.2.2 接口输出函数	162
5.5.3 示例 3—堆和栈操作演示例	99	7.3 应用例	163
5.5.4 示例 4—使用堆空间指针例	100	7.3.1 发声程序	163
5.6 动态存储例——链表	102	7.3.2 信号采集程序	164
5.6.1 单链表结构	102	第 8 章 中断服务程序的编写	167
5.6.2 链表的建立	103	8.1 PC 机的中断类型	167
5.6.3 按升序排列的单链表	105	8.1.1 软中断	168
5.6.4 单链表的输出及节点删除	107	8.1.2 硬中断	168
5.6.5 单链表程序例	108	8.1.3 中断向量表	168
5.6.6 双链表	115		
5.7 关于联合的说明	123		
第 6 章 文件	126		
6.1 文本流和二进制流	127		

8.1.4 中断向量表的填入	169	9.8 鼠标器	229
8.2 用 Turbo C 编写中断程序的方法	169	9.8.1 鼠标器工作原理简介	229
8.2.1 编写中断服务程序	170	9.8.2 鼠标器的 INT 33H 功能调用	230
8.2.2 安装中断服务程序	170	9.8.3 用鼠标器作图	233
8.2.3 中断服务程序的激活	171	9.8.4 用鼠标器热键激活 TSR 程序	236
8.3 中断服务程序例	174		
8.3.1 硬中断演示程序——秒表	174	第 10 章 Turbo C 作图	241
8.3.2 用 geninterrupt() 函数产生中断	177	10.1 图形显示的坐标和像素	241
8.3.3 扬声器唱歌程序	178	10.1.1 图形显示的坐标	241
8.3.4 采用中断方式的信号采集程序	181	10.1.2 像素	241
8.3.5 定时中断程序	185	10.2 图形显示器与适配器	242
8.3.6 能被 Turbo C 程序调用的汇编语言中断程序	189	10.3 显示器工作方式	244
第 9 章 驻留程序的设计	194	10.4 Turbo C 支持的适配器和图形模式	244
9.1 几个特殊区域	194	10.5 图形系统的初始化	246
9.1.1 程序段前缀 PSP 和 DTA	194	10.5.1 图形系统的初始化函数	246
9.1.2 DOS 环境块	196	10.5.2 图形系统检测函数	247
9.2 TSR 程序设计	197	10.5.3 清屏和恢复显示方式的函数	248
9.2.1 TSR 的中断服务部分	198	10.6 基本图形函数	249
9.2.2 程序的驻留	198	10.6.1 画点函数	249
9.2.3 关于 DOS 重入问题的解决方法	199	10.6.2 有关画图坐标位置的函数	249
9.2.4 TSR 程序设计中另外的几个问题	200	10.6.3 画线函数	250
9.2.5 几个有关的库函数说明	201	10.6.4 画矩形和条形图函数	251
9.3 用户激活驻留程序 TSR 的方法	202	10.6.5 画椭圆、圆和扇形图函数	252
9.4 键盘编码	204	10.7 颜色控制函数	253
9.5 键盘缓冲区	207	10.7.1 颜色设置函数	254
9.6 键盘操作函数 bioskey()	208	10.7.2 调色板颜色的设置	255
9.7 程序例	211	10.8 画线的线型函数	259
9.7.1 用 int5 激活驻留程序	211	10.8.1 设定线型函数	259
9.7.2 用 Ctrl-Break 激活驻留程序	215	10.8.2 得到当前画线信息的函数	261
9.7.3 一个完整的驻留程序	217	10.9 封闭图形的填色函数及有关画图函数	261
		10.9.1 填色函数	262
		10.9.2 用户自定义填充函数	262

10.9.3 得到填充模式和颜色的函数	264	11.5 一个动画例子	323
10.9.4 与填充函数有关的作图函数	265	第 12 章 文本的屏幕输出	327
10.9.5 可对任意封闭图形填充的函数	266	12.1 文本方式的控制	327
10.10 屏幕操作函数	267	12.1.1 文本方式控制函数	327
10.10.1 屏幕图象存储和显示函数	267	12.1.2 文本方式颜色控制函数	328
10.10.2 设置显示页函数	269	12.1.3 字符显示亮度控制函数	329
10.11 图视口操作函数	271	12.2 窗口设置和文本输出函数	330
10.11.1 图视口设置函数	271	12.2.1 窗口设置函数	331
10.11.2 图视口清除与取信息函数	271	12.2.2 控制台文本输出函数	331
10.12 图形方式下的文本输出函数	274	12.3 清屏和光标操作函数	331
10.12.1 文本输出函数	274	12.3.1 清屏函数	331
10.12.2 定义文本字型函数	277	12.3.2 光标操作函数	332
10.12.3 文本输出字符串函数	278	12.4 程序例	333
10.13 综合应用例	281	12.5 屏幕文本移动与存取函数	334
10.13.1 用户自定义图模填充长方框 图象	281	12.5.1 屏幕文本移动函数	334
10.13.2 画圆饼图程序	283	12.5.2 屏幕文本存取函数	334
10.13.3 画条形图程序	285	12.6 状态查询函数	336
10.13.4 画函数曲线	287	12.6.1 得到屏幕文本显示有关信息 的函数	336
10.14 图形程序运行的条件	289	12.6.2 得到当前光标位置的函数	338
10.15 图形方式下字型输出的条件	290	12.7 综合应用例	338
第 11 章 菜单设计与动画技术	292	12.7.1 一个弹出式菜单	338
11.1 菜单	293	12.7.2 一个下拉式菜单	342
11.2 菜单设计要点	293	第 13 章 屏幕图形的存取	347
11.3 两个菜单程序	294	13.1 屏幕图形与 VRAM 地址的关系	347
11.3.1 菜单程序 1	294	13.2 存取屏幕图象时地址指针的设置	348
11.3.2 菜单程序 2	302	13.3 VRAM 的位面结构和对它的读写 操作	349
11.4 动画技术	320	13.3.1 VRAM 的位面结构	349
11.4.1 利用动态开辟图视口方法	320	13.3.2 将 VRAM 位面信息存入 文件	350
11.4.2 利用显示页和编辑页交替 变化	320	13.3.3 将文件图象信息写入 VRAM 位面	351
11.4.3 利用画面存储再重放的方法	321	13.4 程序例	352
11.4.4 直接对图象动态存储器进行 操作	323	13.4.1 将屏幕图形存入文件的 程序	352
		13.4.2 将图形文件显示到屏幕上的 程序	353

13.4.3 存多幅图形的程序	354	点阵显示字模	403
13.4.4 显示图象程序例	357	14.4.5 程序例	404
13.5 屏幕图形和图形文件的打印		14.5 汉字的任意倍数放大	407
输出	359	14.6 利用 BIOS 中断调用显示汉字	
13.5.1 打印机适配器及其寄存器			411
结构	359	14.6.1 文本方式下显示汉字的	
13.5.2 点阵式打印机打印图形的		原理	411
原理	360	14.6.2 中文菜单	414
13.6 屏幕输出打印编程要求	363	14.7 建立一个小型专用汉字库	419
13.6.1 打印屏幕图形例	363	14.7.1 库的建立方法	420
13.6.2 打印图形文件例	366	14.7.2 建立小型汉字库的程	
13.7 用单色打印机打印彩色图象的		序例	420
方法	368	14.7.3 利用小型汉字库显示汉字程	
13.7.1 模式法	369	序例	426
13.7.2 抖动法	369		
13.7.3 用模式法打印图象例	370	第 15 章 C 语言与汇编语言的混合编程	
13.7.4 用 32 级灰度抖动法打印图象			429
例	373	15.1 汇编语言子程序使用的场合	429
13.7.5 用 16 级灰度抖动法打印图象		15.2 汇编语言程序的结构	430
例	378	15.2.1 常规的汇编语言程序	
		结构	430
第 14 章 C 程序中汉字显示技术		15.2.2 用简化段定义的程序结构	
.....	382		435
14.1 可在中文 DOS 下显示汉字的程序		15.2.3 简化段定义的伪指令	436
编制	382	15.2.4 段组定义伪指令	437
14.2 在西文 DOS 下 C 程序显示汉字		15.2.5 定义内存模式伪指令	438
技术	385	15.2.6 段名的缺省名	438
14.2.1 国标汉字字符集与区位码		15.2.7 定义段次序	439
.....	385	15.2.8 一个用简化段定义的汇编	
14.2.2 汉字的内码	386	程序标准框架	440
14.2.3 内码转换为区位码与字模		15.3 能被 C 程序调用的一个汇编子	
显示技术	387	程序框架	441
14.2.4 程序例	388	15.3.1 可被 C 调用的一般程序	
14.3 在西文 DOS 下, 24×24 点阵汉字		结构	441
的显示与放大	391	15.3.2 按 C 编译要求段序的一个	
14.4 用直接写显示存储器 VRAM 的		汇编子程序框架	442
方法显示汉字	394	15.4 由 Turbo C 自动产生的一种汇编语言	
14.4.1 将汉字字模装入 VRAM 的		程序结构框架	443
方法一	396	15.5 用简化段定义的汇编语言子程序	
14.4.2 程序例	399		445
14.4.3 将汉字字模装入 VRAM 的		15.6 编写汇编语言子程序的几个问题	
方法二	401		446
14.4.4 24×24 打印字模转换为 24×24		15.6.1 变量和函数的相互调用	446

15.6.2 参数的传递原则	448	第 16 章 C 与 FoxBASE (dBASE) 的	
15.6.3 汇编语言子程序的返回值	450	接口技术	470
15.6.4 汇编语言子程序中寄存器的使用	451	16.1 C 程序直接读取 FoxBASE 数据库中的数据	470
15.7 混合编程的编译和连接	452	16.1.1 FoxBASE (dBASE) 数据库文件结构	470
15.7.1 在 Turbo C 集成环境下进行编译和连接	452	16.1.2 对 FoxBASE 数据库中数据的读取	473
15.7.2 用 Turbo C 命令行编译程序 TCC 进行编译连接	452	16.1.3 程序例	474
15.8 混合编程实例	453	16.1.4 C 程序读取数据库中 MEMO 字段	477
15.8.1 程序 1——同为小内存模式的混合编程	453	16.1.5 自定义的几个对 FoxBASE 操作的函数	479
15.8.2 程序 2——C 程序和汇编子程序是不同的内存模式	454	16.2 通过 FOXBASE 索引文件读取数据	483
15.8.3 程序 3——中内存模式下的混合编程	455	16.3 从数据库的 .MEM 文件中读取数据	489
15.9 Turbo C 程序中内嵌汇编指令行	458	16.4 C 程序间接读取数据库的 .DBF 文件	492
15.10 内嵌汇编指令的 C 程序编译连接方法	461	16.4.1 用 COPY TYPE 命令将 .DBF 转换成文本文件	492
15.11 嵌入汇编指令行的程序例	462	16.4.2 C 程序间接传送数据给 .DBF 文件	494
15.12 汇编语言程序调用 C 函数	465	16.5 关于 C 和 FoxBASE 的交替使用问题	495
15.13 汇编语言调用 C 函数例	466	参考文献	499
15.13.1 程序 1——无参调用	466		
15.13.2 程序 2——有参调用	467		

第 1 章

概 述

C 语言是当前最流行的程序设计语言,它像其它高级语言一样,面向用户,面向解题的过程,编程者不必熟悉具体的计算机内部结构和指令;C 语言又像汇编语言一样,可以对机器硬件进行操作,如进行端口 I/O 操作、位操作、地址操作,并可内嵌汇编指令,将汇编指令当作它的语句一样。我们知道,汇编语言将涉及计算机硬件,所以 C 语言又像低级语言一样,可以对计算机硬件进行控制,因此人们把它称为介于高级语言与低级语言之间一种中级语言。

由于 C 语言的这种特点,因而它不但用于编一般应用程序,而且许多大的操作系统、编译系统也是由 C 语言来写的,甚至可以说 C 原来就是写系统软件的,因为它是和 UNIX 操作系统同时发展起来的,它最初用来写 UNIX 操作系统,由于 UNIX 的不断移植和推广,C 语言也就不断得到发展和普及,像后来的 PC-DOS,WORDSTAR,DBASE III,PLUS 等都是由 C 语言和汇编语言相结合写成的。

1.1 C 语言的发展状况

1970 年美国贝尔实验室的 Ken Thompson 和 Dennis Ritchie 完成了 UNIX 的初版,与此同时,他们还改写了由 Martin Richards 开发的 BCPC 语言,形成了一种称为 B 的语言,此后 B 语言又进一步被进行了改进和完善,形成了称之为 C 的语言。

C 语言大约形成于 1972 年,1973 年 Dennis Ritchie 把 UNIX 系统中的 90% 又用 C 语言进行了改写,并在 PDP-11 小型机上完成了用 C 语言及汇编语言编写的 UNIX 操作系统的调试,并将其投入运行,因而随着 UNIX 的移植、推广,C 语言也得到移植和推广,DOS 支持下的,甚至 Windows 支持下的 C 语言都相继出现,如我们目前广泛使用的在 PC 微机上的 MS-DOS 支持下的 Turbo C 和 Microsoft C 就是典型的 C 语言版本。

由于众多的适用于不同机种、不同字长的 C 编译系统的出现(达几十种),给 C 语言的统一性和兼容性带来了困难,虽然它们基本遵循在 UNIX V 操作系统上配备的 C 语言标准,但仍有许多各自的特点,使得相互移植困难,不利于推广,为此美国国家标准局(ANSI)语言标准化委员会公布了一个 C 语言标准草案(83 ANSI C),该委员会又进一步修改完善该草案,终于在 1987 年公布了 C 语言标准,这就是目前投入商业运营的各种 C 编译系统所遵循的 87 ANSI C 语言标准,Turbo C、Microsoft C 均遵守这个标准。由于 PC 机的快速发展,原来的 C 标准已不能满足对 C 的功能要求,因而在不同的 C 语言版本中又增加了许多新的功能,它们可看作是对 C 标准的扩充和增强,如最明显的是 C 中增加了对屏幕分辨率的定义和一些图形功能的实现。

C 语言是一种结构程序设计的好语言,但随着软件处理对象从简单的数字和字符串发展到目前的图、文、声、象,信息量愈来愈大,愈来愈复杂,因而对程序的设计方法提出了更高的要求,随之在 80 年代出现了一种崭新的程序设计方法——面向对象的程序设计方法,结构程序设计的基本单位是模块,而面向对象的程序设计基本单位则是对象,因此 C 又进一步发展和充实,又出现了支持面向对象的程序设计语言 C++,它实际上是 C 的超级集,其基本核心仍是 C。

例如美国 BorLand 公司 1989 年推出了 Turbo C 2.0,它又在继承和发扬 Turbo C 2.0 的集成开发环境的基础上,推出了面向对象的程序设计语言 Turbo C++,它实际上是 Turbo C 的一个超集,Turbo C++包含了所有 Turbo C 的内容,因而学好、用好 C,仍是面向对象程序设计的基础和前提,它甚至也可体现出面向对象设计的一些方法。

1.2 C 语言的编程格式

C 语言程序一般用小写字母,而仅在一些宏定义中,将常量名用大写字母表示,对一些有特殊含义的变量,偶而也用大写表示,C 语言中对大小写字母是视作两个不同的量。

在 C 语言程序中没有程序行的概念,即在一行中可以任意书写多个语句,只要每个语句用分号作为结尾即可,多个语句还可用大括号{}括起来,形成一个如同单独语句一样的复合语句。一般情况下,为了层次清楚,一行只写一个语句,对复合语句也是按组成的语句依次写在不同的行中。

1.2.1 C 语言程序的结构特点

1. 一个 C 语言程序,通常由带有 # 号的编译预处理语句开始,如 #include <文件名> 它表示,在编译源程序前,用指明的文件取代该预处理语句,通常文件名是指带有后缀为 .h 的磁盘文件,程序编译时,它将从磁盘中读出并插入到原来的预处理语句处,即预处理语句被所指明的包含文件(头文件)代替。

头文件通常是在程序中被调用函数的说明语句和该函数用到的一些符号常量的宏定义等,如在程序中要调用一些标准库函数时,系统提供了相应的头文件,它们其中的一些内容是对该函数的说明及该函数用到的符号常量的宏定义等,如对 fgetc() 的说明放在 stdio.h 头文件中,在该文件中,包含了对 fgetc() 函数的说明:

```
char * fgetc(char *,int,file *)
```

对符号常量 NULL 的宏定义:

```
#define NULL 0
```

当然还包含对其它一些标准 I/O 函数的说明和宏定义等。

用户也可根据自己需要建立自己的头文件,如将程序中用到的符号常量组成一个单独的头文件,在程序开头用 #include 进行包含预处理,使得程序简捷方便。如在进行按键判别时,常要对键值(字符键值对应于 ASCII 码,功能键则为扩充的 ASCII 码)进行分析判别,以确定何键按下,键值为两字节的整型数,若高字节为 0,则表示低字节是代表相应的普通键的 ASCII 码;若低字节为 0,则高字节代表相应的功能键的扩充 ASCII 码。如在菜单程序设计中常用到一些功能键,为此可将它们的宏定义集成一个文本文件,起名为 menu.h 头文

件:

```
#define    INSERT    0x5200
#define    ESC        0x0016
#define    TAB        0x0f09
#define    RETURN    0x000d
#define    RIGHT    0x4d00
#define    LEFT    0x4b00
#define    UP        0x4800
#define    DOWN    0x5000
#define    B        0x0e08
#define    HOME    0x4700
#define    END    0x4f00
#define    PGUP    0x4900
#define    PGDN    0x5100
#define    DEL    0x5300
#define    F1        0x3b00
#define    F2        0x3c00
#define    F3        0x3d00
#define    F4        0x3e00
#define    F5        0x3f00
#define    F6        0x4000
#define    F7        0x4100
#define    F8        0x4200
#define    F9        0x4300
#define    F10    0x4400
```

在程序开头用 `#include "menu.h"` 进行包含预处理后,在程序中就可用这些大写的键名,以代替难懂的键值,这样程序便显得明快易懂。不易因键值写错而导致程序有错。

当在模块化程序设计中,一个大的程序由多个文件组成,若这多个文件共同使用一些函数或常量时,可将这些共用的函数说明和符号常量的定义写在一个头文件中,然后在每个文件的开头用 `#include` 进行包含说明,这样使得程序简捷、可靠,可维护性能提高。一般包含预处理有两种形式,即:

```
#include <文件名>
```

和 `#include "文件名"`

它们的区别在于对包含文件的查找路径不同,前者表示在安装编译系统时,在系统设定的标准目录中去查找,由于由系统提供的头文件一般放在 `include` 子目录下,所以常使用这种形式。

第二种形式表示在当前目录下去查找包含文件,若找不到,再到系统设定的目录中去查找。这种形式常用于用户自己写的包含文件,因为它们通常放在和用户程序同一个目录下。当然在这种形式中,文件名也可用文件路径名代替,这时,将在指定的文件路径中去查找文件。

文件包含也可用在程序中,如:

```

main ( )
{
    :
    #include "con.c"
    :
}

```

其中 con.c 是一个文本文件,编译时,它将从磁盘中调出,插入到该位置处。

一些在程序中用到的符号常量也常用编译预处理命令 #define 来进行定义,如在进行真假判断时,常用符号常量 TURE 和 FALSE 表示真假,这时可以定义:

```

#define TURE 1
#define FALSE 0

```

写在程序开头,这样在编译时在程序中出现 TURE 的地方,就表示为 1,即用 1 代替,出现 FALSE 的地方则用 0 代替。

当程序中经常要用到同一形式的表达式时,也可将该表达式定义为一个带参数的宏,使用时,如同调用函数一样,宏中写上参数即可。例如

求 x 和 y 中较大的一个:

```

#define max(x,y) (x>y ? x : y)

```

程序中经常要用到输出语句 printf("%d\n",x),可定义为:

```

#define PRINTX printf("%d\n",x)

```

这样,只要在程序中需要两个变量中的大者(如 a 和 b),或需要上述格式的输出语句的地方,写上 max(a,b)或 PRINTX 即可。

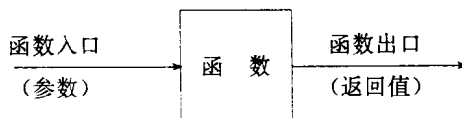
2. 一个完整的 C 程序,总是由 main()函数开始,它像一个大型乐曲的引子,由此引出许多乐章——执行不同功能的函数;main()函数又像一个大型建筑的框架,它显示了要完成这个建筑的轮廓,这些轮廓就是由一个个的函数调用勾画出来的。因此可以说一个 C 程序是由一个个模块堆砌起来的,这个模块的最小单位是一个函数。当然模块也可以是一个源程序,它又由许多函数组成。所以 C 程序的设计,是一种模块化的设计,是许多函数的堆砌。因此在应用程序设计中,应将一个个的功能用一个个的函数来实现。

3. 函数的使用

Turbo C 提供了 400 多个标准库函数,每个函数都完成一定的功能,当程序需要执行这些功能时,只要调用这些函数即可,用户不用自己再去定义。这些函数包括输入输出函数、数学函数、字符串处理函数、内存函数、与 BIOS 和 DOS 调用有关的函数、屏幕和图形功能函数、过程控制函数、目录函数等。

当标准库函数中没有用户要用的功能函数时,就必须自己设计,设计函数的好方法是:

- ① 一个函数不宜处理太多的功能,要保持函数的小型化,功能单一化。
- ② 一个函数要保持自己的独立性,如同一个黑匣子一样,单进单出,即如下图所示:



这样一个函数编译后,其内部定义的程序代码与数据和另一函数的程序代码和数据不会相互影响,为此在函数中要使用局部变量,即它的生存期只维持在调用该函数并执行时,也就是说函数被激活时。要尽量少用或不用全局变量,它将破坏函数的独立性。函数的这种设计方法类似于面向对象设计方法中的封装性。

③ 可以在函数中多使用复合语句,使得函数也具有结构化,且可提高运行效率和节省存储空间。如假设一个函数可以划分成若干个逻辑上独立的程序段,将每个独立的程序段用复合语句表示之,然后用开关语句来控制它们的执行,即:

```
{
    :
    switch (x) {
        case A : { ...,break; }
        case B : { ...,break; }
        :
        default : { ..., }
    }
```

使程序显得清晰明快。

又如一个函数要对三个整型数组 a[300]、b[400]、c[500]分别进行同一处理后进行存盘。为了节省空间,可用复合语句来设计函数结构:

```
array() {
    {int a[300];...;} /* 分程序 1 */
    {int b[400];...;} /* 分程序 2 */
    {int c[500];...;} /* 分程序 3 */
}
```

这样相当于三个分程序,当执行完第一个分程序后,原分配的存储空间 300×2 字节将被收回,依次类推。实际上该函数仅用到了 500×2 个字节的空间,它比用形如 `array() {int a[300], b[400], c[500], ...}` 定义的结构节省了更多的空间。

在复合语句中定义一个局部变量的方法也可推广应用在其它程序的设计中。由于这种局部变量的生存期仅存在于复合语句的生命期中,复合语句执行完,它即失去作用,其占用的存储空间被系统收回,因而可节约存储空间,使程序层次清楚。

④ 在主函数前,要罗列出所有使用的自定义函数的原型说明,这有利于在大程序设计中追踪要调用的函数设置是否正确,如参数传递对否? 返回值一致吗? 在函数定义时,采用现代定义风格。例如:一函数说明为

```
int average (int x,int y)
main()
{
    int x
    char c;
    register result
    x=20;
    c='0'
```

```

    result=average (x,c);
    printf ("\nThe average is %d",result);
}
int average (int x,int y)
{
    return ((x+y)/2);
}

```

这样编译时,将会告诉你函数参数类型不匹配,并将参数c(字符型)转换为2字节的整数放入堆栈中。若没有程序前的函数说明,它将按一个字节的字符放入堆栈中。所以使用函数说明,可保证参数类型的正确性。如在主函数中用如下调用:

```
ch=average (x,r,p);
```

其中ch定义为字符,r为一实数,检查该函数时马上会发现返回值和传递的参数个数和函数原说明不一致,立即会发现错误,编译时,也会立即报错。

当编制大型程序,不同的源程序模块共用一些函数时,可将这些函数的说明放在一个头文件中,每个源程序用#include进行包含说明,这是程序设计中常用的方法。

⑤ 在程序中适当的地方用/*...*/加入注释,这便于程序的阅读和调试。一个较大程序,经过数月后,再读它,可能许多关键的地方已记不起了,加注释可帮助你恢复记忆,调试时,也易于找错。

⑥ 采用层次的书写程序格式,按程序的不同功能分层,对for、while、do_while、if_else、switch_case等一类控制语句或它们的多重嵌套,采用缩格方式,可用TAB键进行控制,如设TAB键为每按一次为后移4个格(缺省方式为8个格,当要改变此值时,可在Turbo C集成环境下选择Options项下的环境子菜单Environment,然后用光标再移到该子菜单的Tab Size项下,选择缩格的格数)。例如:

```

# include <...>
# define M 450
void fun1(int a,int b)
int fun2(char *p,int c)
void main()
{
    int x,y;
    :
    for (i=0;i<M;i++)
    {
        while (...)
        {
            :
            if (x>100)
            {
                :
            }
            else
            {
                :
            }
        }
    }
}

```

```

    }
}
:
printf(...);
}

```

1.2.2 模块化的程序设计

C 语言作为结构化的程序设计语言,易采用自顶向下的设计方法,即开始暂不涉及问题的实质和具体解决的步骤,而只是从问题的全局出发,给出一个概括性的抽象的描述。例如编写一个信号处理程序,它要求对信号数据经过数字处理后进行图形显示并存盘,因而程序大轮廓应是:

- ① 信号数据的输入
- ② 信号预处理
- ③ 信号进行数字处理
- ④ 进行显示
- ⑤ 进行存盘。

接着对各功能进行细分,例如对于信号的输入,又可分为:

- ① 通过 COM1 口由 RS-232 接口输入
- ② 由磁盘数据文件输入

对信号预处理又可分为:

- ① 对信号输入进行反序排列
- ② 用窗函数预处理

对数字处理又可分为:

- ① 求快速付立叶变换
- ② 求功率谱

对用窗函数预处理,又可分为:

- ① 海明窗处理
- ② 汉宁窗处理
- ③ 布拉格曼窗处理

其它功能类推。

在此细化的基础上再进行细化,以至于成为一个个单独的功能,因而可用一个个函数来实现。

设计一个个具体函数,就进入实质性阶段。要定义变量,要选取标准函数,要确定算法,这就是构造程序的基本单元。当一个个函数都设计完后,便可将这些函数按照调用的先后次序在主函数中堆砌起来,并用主函数作为总控程序,完成对它们的参数传递,控制选择对这些函数的调用,形成一个完整的实用的信号处理程序。

如要求数组 $a[]$ 的各元素和,奇数下标的元素的和,各元素中的最大值,各元素的平均值。设计程序时,可将要求实现的各项功能用函数实现,每个函数单独完成一个功能,数组传