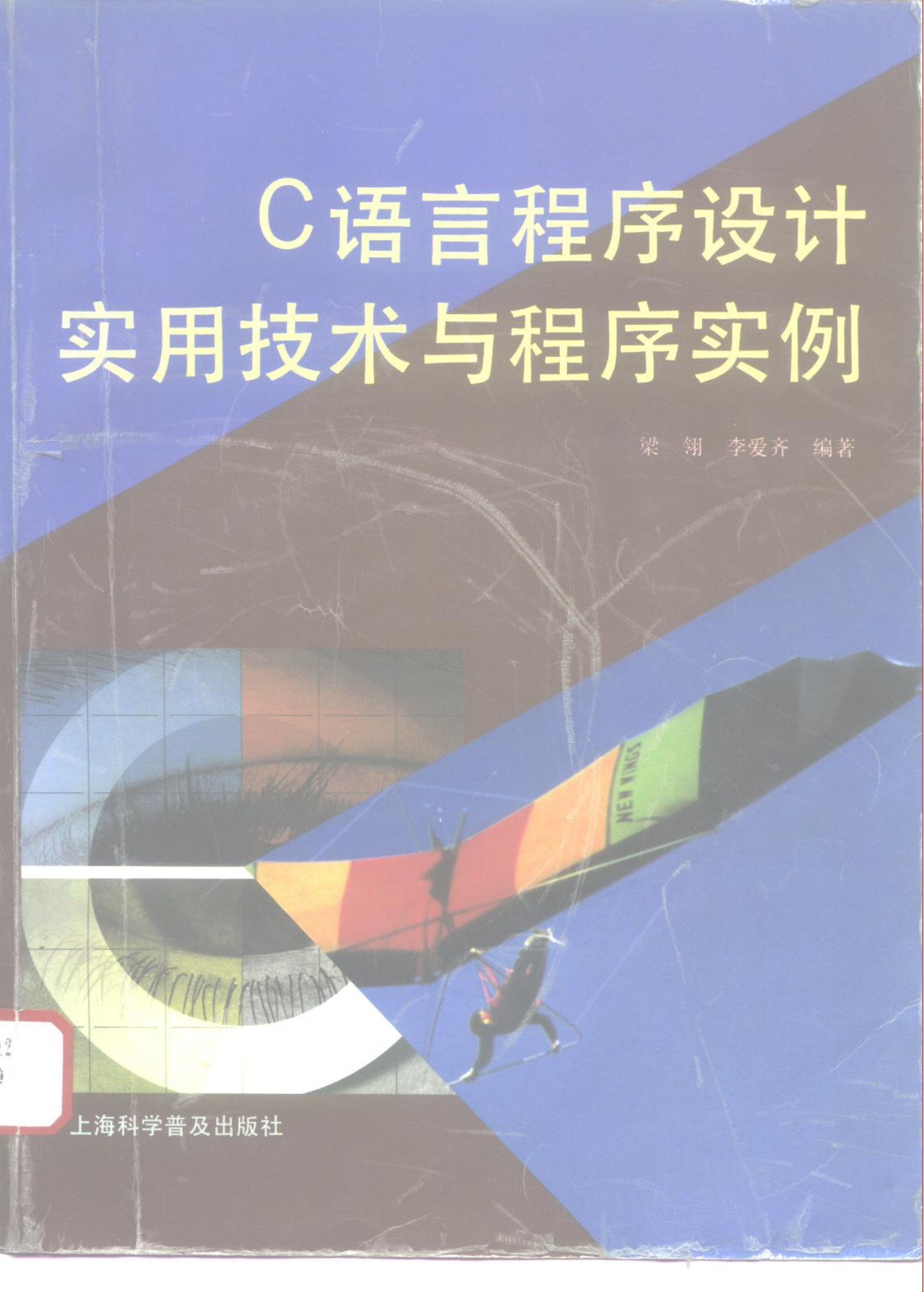


C语言程序设计 实用技术与程序实例

梁 翎 李爱齐 编著



上海科学普及出版社

385262

17
L 0.0

C 语言程序设计

实用技术与程序实例

梁 翎 李爱齐 编著



上海科学普及出版社

(沪)新登字第 305 号

责任编辑 徐丽萍

C 语言程序设计实用技术与程序实例

梁翎 李爱齐 编著

上海科学普及出版社出版

(上海曹杨路 500 号 邮政编码 200063)

新华书店上海发行所发行 常熟高专印刷厂印刷

开本 787×1092 1/16 印张 28.25 字数 678000

1996 年 5 月第 1 版 1996 年 5 月第 1 次印刷

ISBN 7-5427-1041-9/TP·252 定价: 30.00 元
(附软件) 48.00 元

内 容 提 要

本书详细介绍了 C 语言程序设计技巧在应用程序中的实现,包括菜单设计、窗口设计、TSR 程序设计、图形以及游戏程序设计、计算机通信与文件传输、语言解释程序的实现、有关屏幕和扬声器的操作、鼠标编程与制图、条形图的绘制等十个方面。本书所论述的部分均有大量源程序可供参考引用。

本书内容丰富,且结合实例论述,可作为计算机程序设计人员及大专院校学生进行应用程序设计的参考书。

JS-36/08



前 言

由于 C 语言的灵活性,它已经成为当今计算机程序设计的主流语言之一。本书正是利用其优越性,向读者展示各种程序设计技巧以及在应用程序设计中的实现,读者可以从中领略到 C 语言的威力。

本书共分十章。第一章,上弹以及下拉菜单。通过对 BIOS 和视频 RAM 的介绍,展示上弹以及下拉菜单用户接口程序设计的方法。第二章,上弹窗口。在第一章的基础上,进一步将一般用户接口推至窗口设计,并介绍了三个窗口应用程序。第三章,TSR 上弹程序。TSR 是目前比较流行的技巧,如用于适当的场合,它可以增强应用程序的性能。第四章,图形。本章介绍基本图形(如直线、矩形以及圆)的绘制,从而可以在屏幕上绘制较复杂图形。第五章,视频游戏。在第四章的基础上,本章进一步展示视频游戏的实现,包括各种动画画面的制作,并提供了一个完整的捉迷藏游戏的实现。第六章,使用串行口。主要介绍通过串行口实现计算机简单通信的问题,进而介绍一个在低廉的“局域”网串行口进行文件操作。第七章,语言解释程序。本章通过对一个自行设计的 BASIC 解释程序的介绍,展示了计算机语言解释程序的工作原理,对于进一步理解计算机语言有一定帮助。第八章,有关屏幕和扬声器。介绍有关屏幕和扬声器丰富多采的操作,从而可以使应用程序充分地利用它们。第九章,鼠标接口。介绍鼠标编程的一般原理,以及它在绘图应用程序中的实现。第十章,建立商用条形图。介绍条形图的绘制,它对于数据的统计具有直接的效果。

本书所介绍的程序设计技巧内容丰富,且均有结合实例的大量源程序,并通过 Turbo C 编译实现。这些程序也适用于 Microsoft C。

本书在编写过程中得到了多方帮助。本书的顺利出版得到了徐丽萍编辑的热心支持。赵小英绘制了部分插图,唐健鹏、王兴新、祁连红等三位同志对本书的排版给予了大力支持,王蕙、谢燕等对全书的计算机录入做了大量工作,在此一并表示感谢。

编 者

目 录

第一章 上弹以及下拉菜单	(1)
1.1 上弹以及下拉菜单	(1)
1.2 视频适配器	(2)
1.3 通过 BIOS 访问屏幕	(4)
1.3.1 使用 int86()	(5)
1.3.2 保存屏幕的一部分	(6)
1.3.3 屏幕的恢复	(7)
1.4 建立上弹菜单	(7)
1.4.1 显示菜单	(8)
1.4.2 显示线框	(9)
1.4.3 输入用户响应.....	(10)
1.4.4 popup()函数	(13)
1.4.5 各部分合成.....	(15)
1.5 直接访问视频 RAM	(24)
1.5.1 确定视频 RAM 的位置	(24)
1.5.2 转换 save.video()和 restore.video()	(25)
1.6 建立下拉菜单.....	(35)
1.6.1 菜单框架.....	(36)
1.6.2 建立一个菜单框架.....	(36)
1.6.3 pulldown()函数	(38)
1.6.4 屏幕的恢复.....	(39)
1.6.5 一个使用下拉例程的示例程序.....	(39)
1.7 增加选项.....	(51)
第二章 上弹窗口	(52)
2.1 上弹窗口的机理.....	(52)
2.2 窗口框架.....	(52)
2.3 建立窗口框架.....	(53)
2.4 激活和释放一个窗口	(54)
2.5 窗口 I/O 函数.....	(56)
2.5.1 窗口光标定位函数.....	(57)
2.5.2 window.getche()函数	(57)
2.5.3 window.gets()函数	(59)
2.5.4 window.putchar()函数	(60)

2.5.5	window.puts()函数	(61)
2.5.6	各种屏幕处理函数	(61)
2.6	改变窗口的大小与位置	(64)
2.7	用上弹窗口建立应用程序	(68)
2.7.1	十进制到十六进制数转换程序	(68)
2.7.2	四则运算程序	(69)
2.7.3	上弹便笺	(71)
2.8	一个完整的窗口例程	(74)
2.9	一些尝试	(99)
第三章	TSR 编程	(100)
3.1	为什么 TSR 使用如此不便	(100)
3.2	TSR 及中断	(101)
3.3	中断类型修改程序	(101)
3.4	PSP 初步	(101)
3.5	一个交互式 TSR 的基本设计	(102)
3.5.1	TSR 初始化	(102)
3.5.2	ISR	(102)
3.5.3	上弹应用程序	(103)
3.6	何时进行中断 DOS 是安全的	(103)
3.6.1	空闲中断	(104)
3.6.2	DOS 活动标志	(104)
3.7	定时器中断	(104)
3.8	TSR 和图形方式	(104)
3.9	一些特殊的 Turbo C 函数	(105)
3.10	建立一个 TSR 应用程序	(106)
3.10.1	初始化 TSR	(106)
3.10.2	tsr.keystroke()ISR	(109)
3.10.3	击键字符缓冲区	(110)
3.10.4	new.int8()中断	(111)
3.10.5	dos.idle() ISR	(111)
3.10.6	activate.tsr()函数	(112)
3.11	TSR 上弹应用程序	(114)
3.12	完整的 TSR 程序	(116)
3.13	一些其它的 TSR 考虑	(139)
3.14	建立你自己的 TSR 应用程序	(139)
第四章	图形	(140)
4.1	坐标系	(140)
4.1.1	直角坐标	(140)
4.1.2	极坐标	(141)

4.2	视频方式和调色板	(141)
4.3	写像素	(143)
4.3.1	方式 4 图形	(143)
4.3.2	建立 mempoint()函数	(144)
4.4	画线	(146)
4.5	画矩形并填充	(149)
4.6	画圆	(150)
4.7	一个示例测试程序	(154)
4.8	图像的存贮与装载	(161)
4.9	屏幕区域的复制	(164)
4.10	物体的二维旋转.....	(165)
4.10.1	指定物体的旋转.....	(167)
4.11	综合应用例程.....	(171)
第五章	视频游戏	(195)
5.1	子画面	(195)
5.2	游戏插件	(196)
5.3	屏幕级动画制作	(196)
5.4	子画面级动画制作	(204)
5.5	视频游戏数据的组织	(206)
5.5.1	边界的识别	(206)
5.5.2	颜色计数	(206)
5.6	参赛者和记分员	(207)
5.7	一个视频游戏的开发	(207)
5.7.1	游戏的定义	(207)
5.7.2	游戏的颜色编码	(207)
5.7.3	子画面的定义	(208)
5.7.4	主循环	(209)
5.7.5	计算机运动的生成	(213)
5.7.6	“捉到”检测	(215)
5.7.7	完整的捉迷藏程序	(216)
5.8	进一步的开发	(232)
第六章	串行口使用	(234)
6.1	数据的异步串行传输	(234)
6.2	异步串行通讯接口	(236)
6.2.1	硬件的数据交换	(237)
6.3	通讯问题	(237)
6.4	通过 BIOS 访问 PC 串行口	(238)
6.4.1	端口初始化	(239)
6.4.2	字符的传输	(241)

6.4.3	端口状态的检测	(241)
6.4.4	字符的接收	(243)
6.5	在计算机之间传输文件	(243)
6.5.1	软件的数据交换	(244)
6.5.2	7 位数据和 8 位数据	(244)
6.5.3	发送文件	(245)
6.5.4	接收文件	(247)
6.5.5	传输程序	(249)
6.5.6	传输程序的使用	(256)
6.5.7	增强性能	(256)
6.6	局域网	(256)
6.6.1	文件服务器	(257)
6.6.2	文件的装载	(268)
6.6.3	文件的存贮	(273)
6.6.4	局域网的使用	(278)
6.6.5	局域网的改进	(278)
第七章	语言解释程序	(279)
7.1	表达式分析	(279)
7.1.1	表达式	(279)
7.1.2	标记	(280)
7.1.3	表达式的构成	(284)
7.1.4	表达式分析程序	(285)
7.1.5	分析程序如何处理变量	(293)
7.2	小型 BASIC 解释程序	(294)
7.3	主循环	(296)
7.3.1	赋值函数	(297)
7.3.2	PRINT 命令	(298)
7.3.3	INPUT 命令	(300)
7.3.4	GOTO 命令	(301)
7.3.5	IF 语句	(304)
7.3.6	FOR 循环	(306)
7.3.7	GOSUB/RETURN 语句	(309)
7.3.8	完整的解释程序文件	(311)
7.3.9	小型 BASIC 的使用	(324)
7.3.10	解释程序的增强和扩充	(325)
第八章	屏幕和扬声器	(326)
8.1	在文本方式下使用色彩	(326)
8.1.1	文本方式中的属性字节	(326)
8.1.2	使用彩色写一个字串	(327)

8.1.3	色彩的使用	(329)
8.2	改变光标的大小	(329)
8.3	屏幕部分的滚动	(330)
8.4	一个演示程序	(331)
8.5	将屏幕作为磁盘文件保存	(335)
8.6	引入音响	(337)
8.6.1	可编程定时器的使用	(337)
8.6.2	一个测听示例程序	(338)
8.6.3	建立警报器和“激光枪”	(340)
8.6.4	编制“天堂音乐”	(343)
第九章	鼠标接口的编程	(346)
9.1	鼠标器初步	(346)
9.2	实屏和虚屏	(347)
9.3	鼠标设备驱动程序的访问	(348)
9.3.1	复位与读状态	(349)
9.3.2	显示游标	(349)
9.3.3	关闭游标	(350)
9.3.4	读取按键状态和游标位置	(350)
9.3.5	设置游标位置	(350)
9.3.6	运动指示	(350)
9.4	高级鼠标函数	(351)
9.4.1	鼠标的重置	(351)
9.4.2	鼠标游标的显示与关闭	(351)
9.4.3	确定某键是否按下	(352)
9.4.4	运动检测	(352)
9.4.5	游标位置的读取与设置	(353)
9.4.6	一个演示程序	(354)
9.5	将鼠标输入集成到绘图程序中	(359)
9.5.1	主循环	(361)
9.5.2	用鼠标定义物体	(368)
9.5.3	完整版的绘图程序	(371)
9.6	其它一些增强	(398)
第十章	建立商用条形图	(400)
10.1	数据标准化	(400)
10.2	条形图函数的开发	(400)
10.2.1	画坐标网	(402)
10.2.2	值的分类	(402)
10.2.3	画基准线	(403)
10.2.4	显示图例	(403)

10.2.5 一个演示程序.....	(404)
10.3 一个绘图程序.....	(413)
10.3.1 main()函数.....	(413)
10.3.2 enter()函数.....	(415)
10.3.3 min.max()函数.....	(416)
10.3.4 完整的条形图程序.....	(417)
10.4 图形显示.....	(430)
10.5 一些有趣的实验.....	(432)
附录 键盘扫描值.....	(433)

第一章 上弹以及下拉菜单

专业程序中最明显的标志之一是上弹以及下拉菜单的使用。只要正确地操作,这些菜单使程序显得活泼,从而满足用户的期望。虽然其概念很简单,但上弹以及下拉菜单的建立对于一些基本程序设计方法提出了挑战。

上弹以及下拉菜单的建立需要对屏幕的直接控制。虽然现实菜单例程相当适合,但访问屏幕的例程是与硬件及操作系统内在相关的,而且必须通过 C 的标准控制台 I/O 函数。本书提供的视频访问例程适用于任何使用 DOS 并有一个 IBM 兼容的 BIOS 为其操作系统的计算机。之所以选择 DOS,是因为它是目前广泛使用的操作系统,但你也可以把这些基本概念推广到其它系统上。

即使你目前对上弹以及下拉菜单不感兴趣,你仍应该读读本章讨论视频适配器的有关部分,所涉及的许多基本概念将在以后有关窗口以及图形的章节中用到。

1.1 上弹以及下拉菜单

了解什么是上弹以及下拉菜单,以及它们与标准菜单有什么不同是重要的。当使用一个标准菜单时,屏幕不是被清除就是被向上卷动,接着显示菜单。当做了适当选择后,屏幕再次被清除或向上卷动,程序继续下去。选择操作或是用数字或是用每个选项的头一个字母。

在启动上弹或下拉菜单时,它暂时覆盖当前屏幕上的内容。当从菜单中作了选择之后,屏幕又恢复到先前的状态。在上弹以及下拉菜单中,你可以用以下两种方式之一选择一个选项:(1)通过按热键(hot key),这是一个与各个菜单选项相关的一个字母或数字;(2)使用箭头键移动光条到你所要的选项上,并按回车(ENTER)键。通常地,被置高亮的选项是以反相视频或不同颜色显示的。

标准菜单与上弹或下拉菜单的主要不同是标准菜单的启动即终止了程序。然而,上弹或下拉菜单好像是“中止了”目前程序的活动。从用户的角度来看,标准菜单使用户注意力产生中断,而上弹或下拉菜单仅仅是打扰了用户视线;而用户的注意力没有受到影响。

上弹以及下拉菜单的不同也是明显的,任何时刻只有一个上弹菜单可以出现在屏幕上。它用于只有一级深度的菜单,也就是说,菜单中的选项不再有子选项。而对于下拉菜单则可以同时有几个被激活。它们适用于菜单的选择需要使用另一个菜单来确定特定的选项。例如:如果你正在设计一个订购水果的程序,你可能需要使用下拉菜单。如果用户选择“apple”,接下来的菜单提示苹果的颜色,并用第三个菜单显示满足先前的确定特性的苹果。

你可以简单地把上弹菜单想象成一个没有任何子菜单的下拉菜单,但是为这两种类型的菜单开发独立的例程肯定具有优越性,因为下拉菜单在程序中需要比简单上弹菜单更多的附加。

虽然在屏幕上实现一个菜单有许多种方法,但在本章中所提供的函数是最通用的形式。该方法把每个菜单项目作为一行。以下是这种方法的示例。

Load file
Save file
Print
Enter addresses
Delete addresses
Modify
Quit

1.2 视频适配器

由于上弹以及下拉菜单的建立需要对屏幕的直接控制,所以了解视频显示适配器是很重要的。四种最常见的适配器类型是:单显适配器、彩色/图形适配器(CGA)、增强图形适配器(EGA)以及视频图形阵列(VGA)。表 1.1 为各种视频适配器的内存结构。

表 1.1 各种视频适配器的内存结构

类型	工作方式	显示段地址	缓冲区长
MDA	文本	B000H	1
CGA	文本	B000H	4/8
	图形	B000H	1
EGA	单色	B000H	可变
	文本	B000H	可变
	CGA 图形	B000H	可变
	图形	A000H	可变
VGA	单色	B000H	8
	文本	B000H	8
	CGA 图形	B000H	1
	图形	A000H	1/2/4/8

CGA、EGA 以及 VGA 又各有多种操作方式,包括 40 或 80 列的文本或图形操作。其方式如表 1.2 所示。本章所开发的菜单例程被设计成使用 80 列的文本方式。也就是说系统的视频方式必须是 2、3 或 7。无论使用哪种方式,屏幕左上角均为 0,0。

显示在屏幕上的字符被保留在显示适配器上的 RAM 中。单显内存的地址是 B000:0000H,CGA/EGA/VGA 视频 RAM 起址在 B800:0000H,之所以不同是为了区分图形和文本屏幕,但在实际应用中很少这样做。虽然 CGA 和 EGA 在某些方式下功能不同,但是它在

方式 2 和 3 下是相同的。每个被显示在屏幕上的字符需要 2 字节的视频内存。第一个字节存放实际字符,第二个字节存放其屏幕属性。

表 1.2 IBM 系列微型计算机视频方式

方式	象素框	类 型	分辨率	适配器
0	8x8,8x14,9x16	文本,黑白	40x25	CGA,EGA,VGA
1	8x8,8x14,9x16	文本,16 色	40x26	CGA,EGA,VGA
2	8x8,8x14,9x16	文本,黑白	80x25	CGA,EGA,VGA
3	8x8,8x14,9x14	文本,16 色	80x25	CGA,EGA,VGA
4	8x8	图形,4 色	320x200	CGA,EGA,VGA
5	8x8	图形,4 级灰度	320x200	CGA,EGA,VGA
6	8x8	图形,黑白	640x200	CGA,EGA,VGA
7	9x14,9x16	文本,黑白	80x25	单色
8	8x8	图形,16 色	160x200	PCjr
9	8x8	图形,16 色	320x200	PCjr
10	8x8	图形,4 色	640x200	PCjr
11		保留		
12		保留		
13	8x8	图形,16 色	320x200	EGA,VGA
14	8x8	图形,16 色	640x200	EGA,VGA
15	8x14	图形,4 色	640x350	EGA,VGA
16	8x14	图形,16 色	640x350	VGA
17	8x16	图形,2 色	640x480	VGA
18	8x16	图形,16 色	640x480	VGA
19	8x8	图形,256 色	640x200	VGA

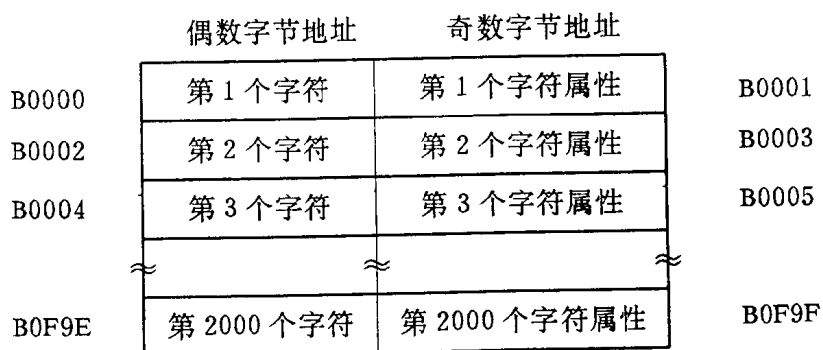


图 1.1 视频内存布局

对于彩色适配器,属性字节由表 1.3 所示。如果你有一个 CGA、EGA 或 VGA,缺省视频方式是 3,字符以属性字节值 7 显示。则结果为 3 个前景色打开,产生白色。要产生相反视频效果,则关闭背景位,同时 3 个前景位被打开,产生一个 0x70 值。

表 1.3 视频属性字节

位	二进制值	置位意义
0	1	蓝色前景
1	2	绿色前景
2	4	红色前景
3	8	高亮度
4	16	蓝色背景
5	32	绿色背景
6	64	红色背景
7	128	字符闪烁

单色适配器能够识别闪烁及亮度位。幸而,单色适配器能把 7 属性解释为普通视频,并把 0x70 释为反视频。而且,值 1 产生下划线字符。

每种适配器实际有 4 倍于在 80 列方式下显示文本所需的内存量。其原因有二。首先,对于图形需要额外内存(除单色适配器以外)。其次,要允许在 RAM 中保留多屏,并在需要时简单地加以切换。内存的每个区域被称做一个视频页(video page),而且活动视频页切换可以产生戏剧性的效果。缺省情况下,DOS 初始化时使用 0 页,而且实际上所有的应用程序都使用 0 页。基于这个原因,在本章例程中也使用 0 页。然而,如果你需要则也可以使用其它视频页。

要访问视频适配器有三种方法。第一个是通过 DOS 调用,这对于上弹或下拉菜单来说显得太慢。第二个是通过 BIOS 例程,如果是在一个高速机器上而且菜单量较小,这种方法相对快些。第三种方法是对视频 RAM 的直接读写,该方法速度非常快,但它需要做许多编程工作。本章提供两个独立的视频例程方法,一个是使用 BIOS,另一个是直接访问视频 RAM。

1.3 通过 BIOS 访问屏幕

由于上弹以及下拉菜单函数必须保存屏幕的部分内容,并在做完选择后恢复之,所以你必须要有例程来保存和装入屏幕部分的内容。本节中所提供的保存和恢复屏幕内容的方法,是依靠调用两个内部 BIOS 函数来完成在屏幕上读写字符。

正像你所知道的,调用 BIOS 可能速度相当慢,然而,它可以保证工作在任何与 IBM 兼容的 BIOS 计算机上,即使实际屏幕硬件不相同。因此,BIOS 菜单例程适用于需要有广泛兼容性的应用系统中。实际上当今生产的所有基于 DOS 的计算机都百分之百地与 IBM 兼容,但也有许多老系统不兼容。

1.3.1 使用 int86()

通过使用软件中断实现对 BIOS 的调用。BIOS 有几个不同的中断用于不同用途。这里用于访问屏幕的是中断 0x10, 它用于访问视频显示器(如果你还不熟悉 BIOS 的访问, 可参看有关书籍; 如《C 语言大全》)。像许多 BIOS 中断一样, 中断 0x10 有几个基于 AH 寄存器值的选项。如果函数返回一个值, 其通常是在 AL 中。然而, 如果有几个值返回, 则可以使用一些其它寄存器。要访问 BIOS 中断, 你需要使用被称为 int86() 的函数。一些编译程序通过不同的名字来调用这个函数, 如 Lattic C 中的 sysint()。但 Microsoft C 和 Turbo C 则用 int86() 调用。

int86() 函数用以下通用形式:

```
int int86(int num,          /* 中断号 */
          union REGS *inregs, /* 输入寄存器值 */
          union REGS *outregs) /* 输出寄存器值 */
```

int86() 的返回值是 AX 寄存器中的值。类型 REGS 在头文件 BIOS.H 中指明。此外, 类型 REGS 是 Turbo C 所定义的; 然而, 它与 Microsoft C 以及其它编译程序所定义的相似。

```
/*
```

```
Copyright (c) Borland International 1987, 1988, 1990, 1991
```

```
All Right Reserved.
```

```
*/
```

```
struct WORDREGS {
    unsigned int ax, bx, cx, dx, si, di, cflag, flags;
};
```

```
struct BYTEREGS {
    unsigned char al, ah, bl, bh, cl, ch, dl, dh;
};
```

```
union REGS {
    struct WORDREGS x;
    struct BYTEREGS h;
};
```

正如你所看到的, REGS 是两个结构的联合。使用 WORDREGS 结构允许你以 16 位形式访问 CPU 寄存器。BYTEREGS 使你能够访问单个 8 位寄存器。例如, 要访问中断 0x10, 功能 5, 你可能使用以下代码:

```
union REGS in, out;
```



```
in.h.ah = 5;
int86(0x10,&in,&out);
```

1.3.2 保存屏幕的一部分

要保存屏幕上的信息,必须读取并存储每个屏幕位置的当前值。要从一个特定屏幕位置读取一个字符,使用中断 0x10,功能 8,它返回在当前光标位置处的字符及其属性。因此,要从屏幕的一个指定部分读取字符,你必须有一个定位光标的方法。虽然大多数 C 编译程序提供这类函数,但有一些可能没有。因此,可以使用这里提供的 goto.xy()。它使用中断 0x10,功能 2,列坐标在 DL 中,行坐标在 DH 中,视频页在 BH 中指定(使用缺省 0 页)。

```
/* Send the cursor to specified x,y coordinates. */
void goto.xy(int x, int y)
{
    union REGS r;
    r.h.ah = 2;      /* cursor addressing function */
    r.h.dl = x;     /* column coordinate */
    r.h.dh = y;     /* row coordinate */
    r.h.bh = 0;     /* video page */
    int86(0x10, &r, &r);
}
```

读字符中断 0x10,功能 8,需要调用时,视频页在 BH 中,而返回当前光标处的字符在 AL 中,其属性在 AH 中。函数 save.video()读取屏幕的一部分,保存信息到一个缓冲区中,并清除屏幕的那个部分:

```
/* Save a portion of the screen. */
void save.video(int startx, int endx, int starty, int endy,
               unsigned int *buf_ptr)
{
    union REGS r;
    register int i,j;
    for(i=startx; i<endx; i++)
        for(j=starty; j<endy; j++) {
            goto.xy(i, j);
            r.h.ah = 8;          /* read character */
            r.h.bh = 0;         /* assume active display page is 0 */
            *buf_ptr++ = int86(0x10, &r, &r); /* save in buffer */
            putchar(' ');      /* clear the screen */
        }
}
```