

高等学校计算机教材

# C 语言高级程序设计

陈天洲 编著

人民邮电出版社

# 序 论

C 语言是介于低级语言和高级语言之间的语言，故可称之为中级语言。其功能强大，具有丰富灵活的控制与数据结构、简洁而高效的语句表达、清晰的程序结构、良好的移植性、较小的空间开销等优点，已被广泛应用于系统软件与应用软件的开发中。由于 C 语言在程序设计上的广泛用途，不仅计算机专业的工作者喜欢使用，而且非计算机专业的工作者也将 C 语言作为自己应用领域的主要编程语言。可以说，目前国内大部分程序设计语言课程的开设，都是主讲 C 语言程序设计方法，即以 C 语言为范例讲解程序设计的基本概念。

但是，很多同学在学习了 C 语言后，觉得学习到的编程知识与实际软件仍然有一定的差距。为什么呢？因为在程序设计语言课程中，没有具体介绍如何在特定的计算机硬件上编写软件，尤其是应用软件的技术。作为程序设计的后续课程之一的“数据结构”，教授了数据存储组织，链表、树、图等结构，搜索排序等算法，但没有面对具体硬件编程。

这就是为什么在“程序设计语言”课程后还需要开设“C 语言高级编程”的主要原因了。

C 语言高级编程，以深化程序设计语言 C 语言为目的，以具体常见的计算机硬件与操作系统为背景，讲述在微机上编写精巧、美观、友好的应用软件。通过本课程的学习，能够激发我们对程序设计语言的认识，并为下一步学习计算机组成、微机原理、接口与通信、控制理论和计算机安全打下坚实的基础。

C 语言，到底能够写什么程序，能够解决什么问题？这是每个学习完程序设计语言的我们都会问的一个问题。我们应该认识到，C 语言除了能够进行科学计算以外，还能够进行人机交互、多媒体展示、工业控制等应用，但是这些应用都必须建立在一定的硬件平台与开发平台上。

综合 C 语言的应用，可以有以下几个方面。

1. 文本屏幕交互：类似与 Turbo C 的编程界面，就是一种文本屏幕交互的界面方式。
2. 完全的图形界面：图形界面给使用者良好的视觉效果，将屏幕当作一张画布，程序员对屏幕各个像素进行处理，显示图形交互界面。
3. 修改操作系统的设置，让操作系统为我所用，改变原来系统的特征，如按一个特殊的键盘就可以让计算机放音乐，同时不影响前台工作。
4. 充分利用内存，突破 DOS 与 Turbo C 对内存的限制。
5. 常驻内存程序。
6. 计算机音乐发声。
7. 快速的键盘输入。
8. 使用鼠标。
9. 访问修改计算机时间。
10. 使用打印机。
11. 在非中文环境下使用汉字。

12. 混合编程：使用 C 语言与其他语言共同书写程序。
13. 使用 FoxBASE 数据库。
14. 进程管理：一个程序可以调用外部可执行程序。
15. 磁盘与文件操作：磁盘加密、文件加密、文件防拷贝、文件型病毒和引导区病毒。

针对以上计算机中 C 语言的应用，本书将主要介绍 C 语言的高级应用。其中部分内容，如微机原理、DOS 操作系统，由于本课程安排时间通常尚未具备其他知识，所以将简化或者作相应的删减。而磁盘加密、病毒等内容，由于不具备汇编语言等基础，本书不作介绍。

本书讲述的软硬件环境是：

- 主机以 IBM PC 及其兼容机为主；
- 操作系统是 DOS 3.3 或 DOS 5.0 以上，Windows 95/98 的 DOS 环境；
- Turbo C 版本为 Turbo C Version 2.0。

Turbo C 在国内流行较早，用户很多。选用 Turbo C 2.0 版本作为讲解基础的主要原因之一是该版本在国内出现较早，流行比较广（需要此版本的读者可与作者联系\*）。特别是它占用内存小（在家用电脑上都可用），灵活方便，各个层次的用户都可利用。另一个原因是该版中已有一个很好的学习 C 语言的工具，即集成环境。它集源程序编辑、编译、连接和调试于一体，尤其是单步断点符号调试以及调试表达式等，对初学者极有帮助，它使一些难以理解的概念（如指针、结构、联合、文件句柄、流和堆栈等）很容易得到直观的解释。

本书将解决上述各应用软件提出的 C 语言编程问题，勾通各部分内容，较详细地解释使用 Turbo C 来完成软件设计，从而较好地利用 Turbo C 来解决自己的实际问题，这是本书追求的主要目标。

Borland C 的 DOS 下版本，大部分的函数与 Turbo 相似，本书内容基本上适合 Borland C 的应用软件开发。

本课程以编程实验教师辅导为主，教师授课为辅，使得我们能够巩固程序设计语言，掌握应用软件的编程技术。本书不仅是授课教材，更是编程实验不可或缺的参考书。

本书所有例程可以在 <http://www.addoil.org> 或者 <http://www.cs.zju.edu.cn/lab/sys/tzchen/index.htm> 下载，作者 E-mail：tzchen@zju.edu.cn。

## 编者的话

C 语言以其丰富灵活的控制与数据结构、简洁而高效的语句表达、清晰的程序结构、良好的移植性、较小的空间开销等优点，已被广泛应用于系统软件与应用软件的开发中。由于 C 语言在程序设计上的广泛用途，不仅计算机专业的工作者喜欢使用，而且非计算机专业的工作者也将 C 语言作为自己应用领域的主要编程语言。可以说，是否掌握 C 语言编程已作为衡量程序员水平的一个尺度。

多年来，编者一直在浙江大学从事 C 语言高级编程的教学工作，对 C 语言的高级应用，学生学习 C 语言高级编程中的难点问题较为明了；同时，我们也有多年从事 C 语言编程的实践经验，清楚 C 语言与微机原理及计算机安全之间的关系。因此在编写此书的过程中，尽量体现我们在教学与编程实践中的这些经验，使学习者能够少走弯路。

信息类学生的“C 语言程序设计”课程有两个方向的后续课程：“数据结构”是介绍 C 语言软件层次技术的后续课程，“C 语言高级编程”是解决硬件问题的后续课程。“C 语言高级编程”的目的是巩固 C 语言程序设计，学习使用 C 语言解决实际软件问题，进行与计算机底层相关的编程。本课程建议学时为 1-2（每周 1 节授课，2 节实验课）。

本书旨在介绍 Turbo C 的各种高级编程技术。通过本书的学习，使读者能够进行键盘鼠标输入、文本图形输出，能够书写硬件驱动，编写常驻内存程序或中断服务程序，了解微机底层编程的基本方法，还能够了解中文输出、打印机输出方法。

本书根据作者近几年在浙江大学讲授“C 语言高级编程”课所用讲义改写而成。本书在编写大纲时，得到了浙江大学计算机学院副院长何钦铭教授的热情指导，初稿完成后又征求浙江大学计算机学院颜晖副教授的意见，并在浙江大学试用一年。在编写过程中，他们以及浙江大学全体 C 语言高级编程的教师给予了真诚鼓励并提出了许多宝贵意见。此外，本书编写过程得到了吴永康、殷萍、王瑛等同志的帮助。谨在此向他们表示衷心的感谢！

本书不当之处，敬请广大读者不吝赐教。

编者  
2002 年 10 月

## 图书在版编目(CIP)数据

C语言高级程序设计 / 陈天洲编. —北京:人民邮电出版社, 2002.12  
ISBN 7-115-10909-5

I. C... II. 陈... III. C语言—程序设计 IV. TP312

中国版本图书馆CIP数据核字(2002)第089330号

## 内 容 提 要

本书是编者根据多年从事C语言高级编程课程的教学与C语言编程经验,按照C语言本身的特点精心编写而成的。全书共分为6章,主要介绍C语言文本方式下的字符输出技术,图形显示技术,图形加速处理技术,并从微机原理的角度简单介绍了微机硬件编程技术,修改操作系统中断设置,扩展内存的方法,常驻内存程序的编程方法和计算机发声等技术。此外,本书还介绍了快速输入方法,主要包括键盘输入与鼠标输入,以及一些C语言的其他高级应用,如系统时间的操纵,打印机的使用,汉字使用,混合编程,FOXBASE数据库编程,进程管理与磁盘文件操作。最后还针对C语言的特点,较详细地介绍了大型程序的编程方法、调试方法,并给出了鼠标驱动程序及其应用的完整例程。

本书内容丰富、概念清晰、深入浅出、侧重实用,是高等学校信息类专业C语言编程的教材,也可以作为C语言程序设计员的工具书。

### 高等学校计算机教材 C语言高级程序设计

---

编 著 陈天洲

责任编辑 潘春燕

执行编辑 赵慧君

人民邮电出版社出版发行 北京市崇文区夕照寺街14号

邮编 100061 电子函件 315@ptpress.com.cn

网址 <http://www.ptpress.com.cn>

读者热线 010-67180876

北京汉魂图文设计有限公司制作

北京 印刷厂印刷

新华书店总店北京发行所经销

开本: 787×1092 1/16

印张: 16.25

字数: 387千字 2002年12月第1版

印数: 1—000册 2002年12月北京第1次印刷

ISBN 7-115-10909-5/TP·3228

---

定价: 22.00元

本书如有印装质量问题,请与本社联系 电话:(010)67129223

# 目 录

序论	1
第 1 章 文本屏幕界面设计	3
1.1 文本方式的控制	3
1.1.1 文本方式控制函数	3
1.1.2 文本方式颜色控制函数	4
1.1.3 字符显示亮度控制函数	5
1.2 窗口设置和文本输出函数	6
1.2.1 窗口设置函数	6
1.2.2 控制台文本输出函数	6
1.3 清屏和光标操作函数	7
1.3.1 清屏函数	7
1.3.2 光标操作函数	7
1.4 屏幕文本移动与存取函数	8
1.4.1 屏幕文本移动函数	8
1.4.2 屏幕文本存取函数	8
1.5 状态查询函数	10
1.6 综合应用实例	11
1.6.1 一个弹出式菜单	11
1.6.2 一个下拉式菜单	15
1.7 directvideo 变量	19
1.8 光标编程	20
小结	21
习题	21
第 2 章 图形程序设计	22
2.1 图形显示的坐标和像素	22
2.1.1 图形显示的坐标	22
2.1.2 像素	22
2.2 图形显示器与适配器	23
2.3 显示器工作方式	25
2.4 Turbo C 支持的适配器和图形模式	26
2.5 图形系统的初始化	28
2.5.1 图形系统的初始化函数	28
2.5.2 图形系统检测函数	29
2.5.3 清屏和恢复显示方式的函数	29

2.6	基本图形函数	30
2.6.1	画点函数	30
2.6.2	有关画图坐标位置的函数	31
2.6.3	画线函数	31
2.6.4	画矩形和条形图函数	32
2.6.5	画椭圆、圆和扇形图函数	33
2.7	颜色控制函数	34
2.7.1	颜色设置函数	35
2.7.2	调色板颜色的设置	37
2.8	画线的线型函数	40
2.8.1	设定线型函数	40
2.8.2	得到当前画线信息的函数	42
2.9	封闭图形的填色函数及有关画图函数	43
2.9.1	填色函数	43
2.9.2	用户自定义填充函数	44
2.9.3	得到填充模式和颜色的函数	46
2.9.4	与填充函数有关的作图函数	46
2.9.5	可对任意封闭图形填充的函数	47
2.10	屏幕操作函数	48
2.10.1	屏幕图像存储和显示函数	49
2.10.2	设置显示页函数	50
2.11	图视口操作函数	52
2.11.1	图视口设置函数	52
2.11.2	图视口清除与取信息函数	52
2.12	图形方式下的文本输出函数	54
2.12.1	文本输出函数	55
2.12.2	定义文本字型函数	57
2.12.3	文本输出字符串函数	59
2.13	动画技术	60
2.13.1	利用动态开辟图视口的方法	60
2.13.2	利用显示页与编辑页交替变化的方法	60
2.13.3	利用画面存储再重放的方法	60
2.13.4	直接对图像动态存储器进行操作的方法	61
2.14	菜单生成	61
2.15	图形程序使用环境	62
2.15.1	BGI 使用	62
2.15.2	图形方式下字型输出的条件	64
2.15.3	BGI 图形驱动	64
2.16	直接存储存取	65

---

2.16.1	BIOS 中断在显示中的应用	66
2.16.2	VGA 寄存器	68
2.16.3	屏幕图形与 VRAM 地址的关系	71
2.16.4	VRAM 的位面结构	71
2.16.5	将 VRAM 位面信息存入文件	72
2.16.6	将文件图像信息写入 VRAM 位面	73
2.16.7	VGA 标准图形模式 12H 编程	74
2.16.8	VGA 标准图形模式 13H 编程	76
2.17	SVGA 编程简述	76
2.17.1	SVGA 显卡的检测	77
2.17.2	SVGA 模式信息的获取与模式操作	79
2.17.3	SVGA 的直接存储显存与内存控制	81
	小结	81
	习题	81
第 3 章	微机硬件驱动	83
3.1	I/O 接口的输入输出简介	83
3.1.1	I/O 接口的寻址方式	83
3.1.2	I/O 接口的输入输出函数	84
3.2	中断服务程序的编写	85
3.2.1	PC 机的中断类型	86
3.2.2	用 Turbo C 编写中断程序的方法	88
3.2.3	中断服务程序例子	91
3.3	BIOS 与 DOS 调用	96
3.3.1	关于 DOS 与 BIOS 的说明	96
3.3.2	BIOS 调用	97
3.3.3	DOS 调用	98
3.3.4	BIOS 和 DOS 系统调用函数	99
3.4	驻留程序的设计	104
3.4.1	TSR 程序设计	104
3.4.2	用户激活驻留程序 TSR 的方法	109
3.4.3	TSR 唱歌程序例子	111
3.5	扩充存储器编程	113
3.5.1	PC 存储器结构	113
3.5.2	存储器的分段与物理地址的形成	116
3.5.3	与地址操作有关的几个宏	118
3.5.4	指针的分类	119
3.5.5	寄存器与伪变量	121
3.5.6	内存模式	125
3.5.7	保护虚地址方式下的段和偏移	129

3.5.8	扩展存储器的使用实例	130
3.5.9	扩展内存	133
3.6	计算机发声	134
3.6.1	发声原理	134
3.6.2	声音函数	135
3.6.3	计算机乐谱	136
3.7	使用串口通信	136
	小结	140
	习题	140
第 4 章	输入方法编程	141
4.1	键盘输入	141
4.1.1	键盘编码	141
4.1.2	键盘缓冲区	144
4.1.3	键盘操作函数 bioskey ( )	146
4.2	鼠标输入	147
4.2.1	鼠标简介	147
4.2.2	鼠标的 INT 33H 功能调用	148
4.2.3	鼠标主要功能函数	150
4.2.4	用鼠标作图	153
4.2.5	用鼠标热键激活 TSR 程序	156
	小结	156
	习题	156
第 5 章	其他高级编程技术	158
5.1	目录时间函数编程	158
5.1.1	目录文件函数编程	158
5.1.2	时间函数编程	158
5.2	汉字技术	160
5.2.1	汉字库	160
5.2.2	显示	161
5.3	打印驱动	162
5.4	混合编程简介	163
5.4.1	C 语言外部接口约定原则	163
5.4.2	C 语言与汇编语言程序接口	164
5.4.3	C 语言与 Pascal 语言程序接口	165
5.4.4	Turbo C 行间直接嵌入汇编	167
5.5	FoxBASE 数据库编程	168
5.6	进程管理	169
5.6.1	exec 函数组	170
5.6.2	spawn 函数组	171

---

5.6.3 system 函数	171
5.7 磁盘与文件操作	172
小结	176
习题	176
第 6 章 大型综合程序开发	177
6.1 大程序的设计风格	177
6.2 Turbo C 调试器	178
6.3 鼠标驱动程序例程	182
6.4 魔方程序	192
小结	220
习题	221
附录 Turbo C 2.0 函数说明	223
1. 字符分类函数 (ctype.h)	223
2. 数学函数 (math.h、stdlib.h、string.h、float.h)	223
3. 进程函数 (stdlib.h、process.h)	226
4. 转换子程序 (math.h、stdlib.h、ctype.h、float.h)	228
5. 诊断函数 (assert.h、math.h)	228
6. 输入输出子程序 (io.h、conio.h、stat.h、dos.h、stdio.h、signal.h)	229
7. 接口子程序 (dos.h、bios.h)	235
8. 字符串、内存操作函数 (string.h、mem.h)	242
9. 存储分配子程序 (dos.h、alloc.h、malloc.h、stdlib.h、process.h)	245
10. 时间日期函数 (time.h、dos.h)	246

# 第 1 章 文本屏幕界面设计

在程序设计中，我们已经学习了基本输入输出方法，主要是 `printf` 与 `scanf`，`putchar` 与 `getchar` 等。注意到基本输入输出都是行卷动的，光标只能从左到右，从上到下移动，而不能向上向左移动。同时我们又注意到 Turbo C 这个应用程序界面美观，不仅光标可以随意移动，而且有各种热键和菜单。C 语言是否可以编写这样的应用软件呢？答案是肯定的，而且是这本书中最简单的界面设计——文本屏幕界面。其界面由字符文本构成。我们看到的 Turbo C 的边框，就是由一些特殊的符号排列而成的。本章就介绍文本屏幕界面的设计。

显示器显示方式有文本方式和图形方式两种，本章讲述文本方式下如何控制屏幕的输出，并介绍一些有关屏幕处理的函数。在 Turbo C 集成环境下，我们进行的源程序编辑、编译、连接，均是在文本方式下，它们都是以文本方式进行工作，即每屏 80 列 25 行蓝底白字，输出为整个屏幕。Turbo C 提供了一些屏幕处理函数，它们可以改变屏幕输出的方式，可以开设窗口，使文本输出在该窗口中进行，这些函数的有关信息（如宏定义等）均包含在 `conio.h` 头文件中，因此在用户程序中使用这些函数时，必须用 `include` 将 `conio.h` 包含到程序中。

## 1.1 文本方式的控制

### 1.1.1 文本方式控制函数

顾名思义，文本方式就是显示文本的模式，它的显示单位是字符而不是图形方式下的像素，因而在屏幕上显示字符的位置坐标就用行和列表示。如在缺省方式下，每屏为 80 列 25 行，Turbo C 规定屏的左上角为 1 行 1 列，屏的右下角为 25 行 80 列，如图 1.1 所示。Turbo C 支持的文本显示方式有 5 种（见表 1.1 中的第 1~5 行），它们可以用文本显示方式设置函数来进行设置，该函数原型为：

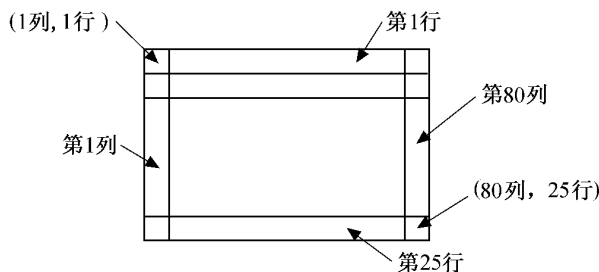


图 1.1 屏幕文本显示坐标

```
void textmode(int newmode)
```

该函数中的 `newmode` 参数可以选用如表 1.1 所示的任一种方式，可以用表中指出的方式代码，也可以用大写的方式名，该函数将清除屏幕，以整个屏幕为当前窗口，并移光标到屏的左上角。

表 1.1 文本显示方式

方式代码	方式名	显示列 × 行数和颜色
0	BW40	40 × 25 黑白显示
1	C40	40 × 25 彩色显示
2	BW80	80 × 25 黑白显示
3	C80	80 × 25 彩色显示
7	MONO	80 × 25 单色显示
- 1	LASTMODE	上一次的显示方式

表 1.1 中的 LASTMODE 方式指上一次设置的文本显示方式，它常用于图形方式到文本方式的切换。关于颜色，将在文本颜色设置函数中介绍。MONO 方式用在 MGA 显示器上。

上述的文本显示方式仅是 CGA 显示器所特有的，它并不支持 EGA、VGA 显示器的 80 列 43 行文本方式，更不支持 TVGA 的 132 列 60 行的文本方式，所以文本显示方式时，EGA、VGA、TVGA 均工作在 CGA 仿真方式下。

### 1.1.2 文本方式颜色控制函数

为了控制文本显示的前景和背景颜色，Turbo C 提供控制颜色的函数。

#### 1. 文本颜色设置函数

文本颜色设置函数如下：

```
void textcolor(int color);
```

该函数将设置显示的前景色，即字符显示的颜色，其颜色 color 可用表 1.2 中所列的值或大写的颜色名表示，该函数只能用在彩色显示的方式下。

表 1.2 颜色表

颜色名	值	显示色	用 处
BLACK	0	黑	前景、背景色
BLUE	1	蓝	前景、背景色
GREEN	2	绿	前景、背景色
CYAN	3	青	前景、背景色
RED	4	红	前景、背景色
MAGENTA	5	洋红	前景、背景色
BROWN	6	棕	前景、背景色
LIGHTGRAY	7	淡灰	前景色
DARKGRAY	8	深灰	前景色
LIGHTBLUE	9	淡蓝	前景色

续表

颜色名	值	显示色	用 处
LIGHTGREEN	10	淡绿	前景色
LIGHTCYAN	11	淡青	前景色
LIGHTRED	12	淡红	前景色
LIGHTMAGENTA	13	淡洋红	前景色
YELLOW	14	黄	前景色
WHITE	15	白	前景色
BLINK	16	闪烁	前景色

## 2. 文本背景颜色设置函数

文本背景颜色设置函数如下：

```
void textbackground(int color);
```

该函数将设置文本显示的背景颜色，其参数 color 仅能选择表 1.2 中的前 8 种颜色，即值为 0~7。

## 3. 文本属性设置函数

文本属性设置函数如下：

```
void textattr(int attr);
```

该函数将设置文本显示的属性，所谓显示的属性是指字符显示的颜色、背景色，字符显示是否闪烁。显示属性参数 attr 可用一个字节即 8 位来描述，各位含义如图 1.2 所示。

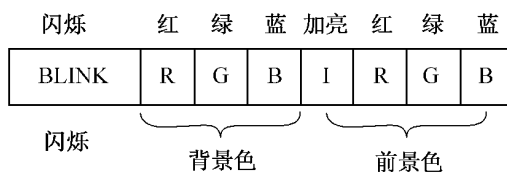


图 1.2 属性字节各位含义

其中低 4 位 0~3 用来设置字符显示的颜色（对应颜色值 0~15），4~6 位用来设置显示背景色（颜色值为 0~7），第 7 位最高位用来设置显示的字符是否闪烁。

如要求在蓝色背景下显示红色的字符，可用此函数设置：

```
textattr(RED + (BLUE << 4));
```

要求在白色背景下显示闪烁的蓝色字符，可设置为：

```
textattr((WHITE << 4) + BLUE + BLINK); 或者 textattr(128 + 1 + (15 << 4));
```

其中  $WHITE \ll 4$  表示左移 4 位，即其对应的二进制值左移 4 位，变成 4~6 位，这恰是设置背景色的位，也就是把 15 的二进制数左移 4 位。用十六进制表示可写成 `textattr(0xF1)`。

### 1.1.3 字符显示亮度控制函数

字符显示亮度控制函数有如下几个。

```
void highvideo(void);
```

该函数将设置用高亮度显示字符。

```
void lowvideo(void);
```

该函数将设置用低亮度显示字符。

```
void normvideo(void);
```

该函数将设置通常亮度显示字符。

## 1.2 窗口设置和文本输出函数

在文本方式下,没进行窗口设置时,即窗口设置缺省时,认为整个屏幕为显示窗口。Turbo C 也提供了窗口设置函数 `window()`,此函数可由用户根据自己的需要来重新设定显示窗口。当设定后,以后的控制台 I/O 操作(即文本输入输出),就可在此窗口中进行。现对文本方式下的窗口设置函数和控制台输出函数(即文本输出函数)作以下介绍。

### 1.2.1 窗口设置函数

窗口设置函数如下:

```
void window(int x1, int y1, int x2, int y2);
```

其中 $(x1, y1)$ 为窗口的左上角坐标, $(x2, y2)$ 为窗口的右下角坐标,这些坐标是以整个屏幕为参考坐标系,如图 1.1 所示。当定义窗口时,若坐标超出屏幕坐标界限,则该窗口将不会建立。

利用窗口函数可以在屏幕上定义多个不同窗口,以显示不同的信息,但最后一次定义的窗口为当前窗口,与窗口有关的操作函数,均在此窗口内进行。如定义了一个窗口后,前面讲过的函数 `textcolor`, `textbackground`, `textattr` 将仅对此窗口起作用,窗口外不会受到影响。

### 1.2.2 控制台文本输出函数

我们以前介绍过的 `printf()`, `putc()`, `puts()`, `putchar()` 和输出函数是以整个屏幕为窗口的,它们不受 `window` 设置的窗口所限制,也无法用函数控制它们输出的位置。但 Turbo C 提供了 3 个文本输出函数,它们受窗口的控制,窗口内显示光标的位置,就是它开始输出的位置。当输出行右边超过窗口右边界时,自动移到窗口内的下一行开始输出;当输出到窗口底部边界时,窗口内的内容将自动产生上卷,直到完全输出完为止。这 3 个函数均受当前光标的控制,每输出一个字符,光标后移一个字符位置。这 3 个输出函数是:

```
int cprintf(char *format, . . .);
```

```
int cputs(char *str);
```

```
int putch(int ch);
```

它们的使用格式同 `printf()`, `puts()` 和 `putc()`,其中 `cprintf()` 是将按格式化串定义的字符串或数据输出到定义的窗口中,其输出格式串同 `printf()` 函数,不过它的输出受当前光标控制,且输出特点如上所述,`cputs()` 同 `puts()`,是在定义的窗口中输出一个字符串,而 `putch()` 则是输出一个字符到窗口,它实际上是函数 `putc()` 的一个宏定义,即将输出定向到屏幕。

## 1.3 清屏和光标操作函数

### 1.3.1 清屏函数

清屏函数有如下几个。

```
void clrscr(void);
```

该函数将清除窗口中的文本，并将光标移到当前窗口的左上角，即 (1, 1) 处。

```
void clreol(void);
```

该函数将清除当前窗口中从光标位置开始到本行结尾的所有字符，但不改变光标原来的位置。

```
void delline(void);
```

该函数将删除一行字符，该行是光标所在行。

### 1.3.2 光标操作函数

光标操作函数如下：

```
void gotoxy(int x, int y);
```

该函数将把光标移到窗口内的 (x, y) 处，x, y 坐标是相对窗口而言。它多和 cprintf 函数配合，以指定输出开始位置。

下面的程序演示了上述函数的作用，本程序由于没有用 window 函数设置窗口，因而用缺省值，即全屏幕为一个窗口。程序开始设置 40 列 25 行文本显示方式 (C40)，背景色为蓝色，前景色为红色，因此屏上显示出蓝底红字的 press any key to continue。可以看出整个屏幕 (即窗口) 并没有显示蓝色，当按任一键后，经 clrscr 函数清屏后，设置的背景色才使屏幕背景变蓝。说明只有用 clrscr 清屏后，其前面的设置背景色函数才起作用。gotoxy (10, 10) 使光标移到第 10 行 10 列，因而接着 cprintf 函数将 welcome your 字符串从此位置开始输出，而下一个 cprintf 的输出则被定位在 14 行 10 列位置输出。接着 gotoxy (17, 10) 将光标移到 10 行 17 列位置，当按任一键后，则由 clreol 函数将该行从 17 列位置开始到行尾的所有字符删去，即将 your 删掉了。接着 gotoxy (17, 14) 又将光标移至 14 行 17 列位置，按任一键后，则显示在该行的 Let's study Turbo C 全被删除。说明 delline 函数仅受光标所在行的控制，列号不起作用，即删除光标所在行的一行字符。

程序清单如下：

```
#include <conio.h>
main()
{
    int i;
    textmode(C40);           /* 设置文本显示方式为 C40 */
    textbackground(BLUE);   /* 设置背景色为蓝色 */
    textcolor(RED);        /* 设置前景色为红色 */
    cprintf("%s", "press any key to continue. ");
    getch();
    clrscr();
}
```

```

gotoxy(10,10);          /* 光标移到(10, 10)处 */
printf("%s","welcome Your"); /* 在(10, 10)处开始显示字符串 */
gotoxy(10,14);
printf("%s","Let's study Turbo C.");
gotoxy(17,10);
getch();
clrEOF();              /* 删去该行从第 17 列开始到行尾的字符 */
gotoxy(17,14);
getch();
delline();             /* 删去第 14 行 */
getch();
}

```

## 1.4 屏幕文本移动与存取函数

### 1.4.1 屏幕文本移动函数

屏幕文本移动函数如下：

```
void movetext(int x1, int y1, int x2, int y2, int x3, int y3);
```

该函数将把屏幕上左上角为  $(x_1, y_1)$ ，右下角为  $(x_2, y_2)$  的矩形内文本拷贝到左上角为  $(x_3, y_3)$  的一个新矩形区内。这里  $x, y$  坐标是以整个屏幕为窗口坐标系，即屏幕左上角为  $(1, 1)$ 。该函数与开设的窗口无关，且原矩形区文本不变。

### 1.4.2 屏幕文本存取函数

#### 1. 存文本函数

存文本函数如下：

```
void gettext(int x1, int y1, int x2, int y2, void *buffer);
```

该函数将把左上角为  $(x_1, y_1)$ ，右下角为  $(x_2, y_2)$  的矩形区内的文本存到由指针 `buffer` 指向的一个内存缓冲区内。这个缓冲区大小可以这样来计算，一个在屏幕上显示的字符需占显示存储器 (VRAM) 的两个字节，第一个字节是该字符的 ASCII 码，第二个字节为属性字节，即表示其显示的前景、背景色及是否闪烁，如图 1.2 所示。所以矩形区内的文本所占字节总数可以这样来计算：

字节总数=矩形内行数 × 每行列数 × 2

其中：

矩形内行数= $y_2 - y_1 + 1$

每行列数= $x_2 - x_1 + 1$

每行列数是指矩形内每行的列数。

矩形内文本字符在缓冲区内存放的次序是从左到右，从上到下，每个字符占连续两个字节并依次存放。

#### 2. 取文本函数

取文本函数如下：

```
void puttext(int x1, int y1, int x2, int y2, void *buffer);
```

该函数将把由 `buffer` 指针指向的缓冲区内所存文本复制到屏幕上一矩形区内, 该矩形区左上角为  $(x1, y1)$ , 右下角为  $(x2, y2)$ 。

下面的程序首先定义了一个字符数组, 下标为 64, 表示用来存 4 行 8 列的文本, 在  $(10, 10)$  开始位置显示 `L:load`, 接着在下面 3 行相同的列位置显示另外 3 条信息, 13 行 10 列显示的 `E:exit` 后面带有回车换行符, 为的是将光标移到下一行开始处, 以便显示 `press any key to continue`。当按任一键后, `gettext` 函数将  $(10, 10, 18, 13)$  矩形区的内容存到 `ch` 缓存区内。`ch` 即上述的 4 行 8 列信息, 接着设置一个窗口, 并纵向写上 1, 2, 3, 4, 然后用 `movetext` (`ch`), 将此窗口内容复制到另一区域, 由于此区域包括背景色和显示的字符, 所以被复制到另一区域的内容也是相同的背景色和文本。当按任一键后, 又出现提示信息, 再按键, 则存在 `ch` 缓冲区内的文本由 `puttext` (`ch`) 又复制到开设的窗口内。注意上述的函数 `movetext` (`ch`), `gettext` (`ch`), `puttext` (`ch`) 均与开设的窗口内坐标无关, 而是以整个屏幕为参考系的。

程序清单如下:

```
#include <conio.h>
main()
{
    int i;
    char ch[4*8*2];          /* 定义 ch 字符串数组作为缓存区 */
    textmode(C80);
    textbackground(0);
    textcolor(RED);
    clrscr();
    gotoxy(10,10);
    cprintf("L:load");
    gotoxy(10,11);
    cprintf("S:save");
    gotoxy(10,12);
    cprintf("D:delete");
    gotoxy(10,13);
    cprintf("E:exit\r\n");
    cprintf("Press any key to continue");
    getch();
    gettext(10,10,18,13,ch); /* 将矩形区文本存到 ch 缓存区 */
    clrscr();
    textbackground(1);
    textcolor(3);
    window(20,9,34,14);    /* 开一个窗口 */
    clrscr();
    cprintf("1.\r\n2.\r\n3.\r\n4.\r\n"); /* 纵向写 1, 2, 3, 4 */
    movetext(20,9,34,14,40,10); /* 将矩形区文本复制到另一区域 */
    puts("hit any key");
    getch();
    clrscr();
    cprintf("press any key to put text");
    getch();
    clrscr();
    puttext(23,10,31,13,ch); /* 将 ch 缓存区所存文本在屏上显示 */
}
```