菜单技术和面向对象的程序设计

for Windows 95/98

史斌星　余加莉　编著

国防工业出版社

菜单技术和面向对象的程序设计 for Windows 95 / 98

史斌星　余加莉　编著

国防工业出版社

·北京·

图书在版编目(CIP)数据

菜单技术和面向对象的程序设计 for Windows 95/98/史斌星,余加莉编著. -北京:国防工业出版社,1999.1
ISBN 7-118-02009-5

Ⅰ.菜… Ⅱ.①史… ②余… Ⅲ.①电子计算机-界面②面向对象语言-程序设计③窗口软件,Windows Ⅳ.TP311.11

中国版本图书馆 CIP 数据核字(98)第 28868 号

国防工业出版社出版发行
(北京市海淀区紫竹院南路 23 号)
(邮政编码 100044)
河北三河市腾飞胶印厂印刷
新华书店经售
*
开本 787×1092 1/16 印张 17¼ 403 千字
1999 年 1 月第 1 版 1999 年 1 月北京第 1 次印刷
印数:1—4000 册 定价:24.00 元

序

在当前蓬勃发展的计算机应用软件中，菜单(Menu)技术显得格外绚丽多彩。它之所以受到人们喜爱，一方面，它以生动的图像界面代替了纯文字界面，使人感到清晰悦目；另一方面，为实现所见即所得的操作方式创造了条件，在鼠标或触摸屏的配合下，可以轻松地在图中选取菜单栏目而无需在键盘输入操作命令，这可使不懂计算机的人更容易学习使用计算机。从这个意义上讲，它把计算机的使用“傻瓜化”了。许多技术领域的发展都已表明，“傻瓜化”是现代技术得以推广的有效手段，可以说这是一种时代的潮流。

从下面一些例子可以看到，菜单技术已在一个相当广泛的领域得到应用。近几年来，视窗技术 (Windows)有了很大发展。人们预言用 Windows 系统代替 DOS 系统已经指日可待。Windows95/98 的应用已使 DOS 系统的应用缩小到了一个极小的领域。菜单技术是 Windows 的基础，在 Windows 中得到了充分的发挥，从开始进入 Windows 起，用户就会被一系列色彩缤纷的漂亮菜单所吸引，使用户进入了一个令人愉快的图像境界，从而使操纵计算机不再是一件枯燥乏味的事了。在我国，销售量很大、在软件市场中十分看好的财务软件也是菜单技术大显身手的领域。一些有实力的公司纷纷推出自己的财务软件。可以毫不夸张地说，除了与财务有关的专业内容之外，在这场竞争中能否取胜，菜单技术的水平具有举足轻重的分量。电子游戏的编制在当前也是软件市场的一个广阔天地。在这些软件中，大量地应用菜单技术。有时为了一个漂亮的菜单按钮，设计者也愿为之付出大量的心血。电子读物更是方兴未艾，如雨后春笋般地在市场出现，各种各样的电子出版物都以菜单的美观和使用方便来打扮和推销自己。

广大计算机爱好者，都有自己设计软件的愿望，当然也希望将自己的软件菜单化，使作品的界面更为美观、使用更加方便。作为起步，也可以先为家里设计一个记录经济收支的帐目软件，您可以设计成一打开电脑，出现在人们眼前的就是一个记帐用的菜单，这样，家中即使不熟悉计算机的人也可以利用它了。学生可为自己设计一个学习计划或者设计一个记录外语词汇的菜单软件；经商的人可能需要一个专用的记录营销商品项目的菜单软件；有股票生意的也许需要为自己设计一个记录和研究股票涨落规律的菜单软件等。这种业余的菜单设计往往是许多人从业余走向专业的一个途径。

如何才能让读者更容易地学习菜单技术，这是编写本书的原动力。目前市场上已拥有大量有关计算机软件的书籍，但是专门论述菜单技术的书籍却不多，将基本计算机语言与菜单技术结合起来写的则更少。一些读者苦于找不到既能帮助他们入门又能引导他们达到一定高度的书籍。菜单技术有它自己的规律和编程技巧。从掌握基本语言到编写菜单还有一段相当的距离，仅仅依靠自己的摸索就会走弯路。本书能够帮助读者少走弯路和缩短学习过程。

为了适应不同层次读者的需要，我们采取了以下编写方式：全书分成两大部分：第

一部分为上篇,即基础篇,叙述与菜单技术有关的C语言基本原理和应用;第二部分为下篇,即提高篇,论述一些较为深入的问题。即使您已熟悉C语言,我们相信读一下本书也是会有收获的,因为本书采用了特定的叙述方法,是从菜单设计的角度来组织这一部分内容的,注重对基本概念的叙述。对于有了一些C语言基础的读者,如果从头读起,您将会不知不觉地跟着书中的叙述逐渐深入。对于毫无C语言基础的读者,最好先读一点有关C语言的初级教材。

C语言是当前人们广泛选用的编程语言,可以方便地解决用其它高级语言难以处理的问题。此外,由于C语言被广泛采用,人们编写了大量功能齐备的专用函数,它有一个庞大的函数库,在需要时可以方便地进行调用,这为程序编制节省大量的工作量。本书采用目前广为流行的C++作为基本语言,并以Windows95和Windows98下的Borland C++ 5.0版本为基本软件。DOS下的程序则采用Borland C++ 3.1版本。书中将给出的实例与其它版本(例如Borland C++ 3.1、4.0或4.5)的同一程序进行比较,指出它们的共同和不同之处。事实上对于许多程序它们是十分相近的,只要稍加修改便可将5.0版本的程序改为低版本的程序。本书共给出了50多个程序实例,每一个都是在机器上通过的可以实际执行的程序。

本书虽然不是论述C语言的专著,但是在C语言的基本语句上仍然花了不少精力进行分析和说明。

可视化的编程方法已有了极大的发展,Visual Basic、Visual C++等编程工具的出现大大便利了程序设计人员。但从学习角度来看,传统的C语言程序仍然是学习的一个重要平台。学好它们有助于掌握C语言的基本规律和精髓。

本书作者长期在大学从事教育工作,作者之一曾在美国多年从事软件的开发工作,我们希望能将自己的教学经验和计算机语言的实践经验融合到这本书中,为计算机语言的推广作出一点贡献。由于受到作者学识所限制,可能会有许多缺点和不妥之处,恳求批评指正。

编者衷心地感谢本书合作者。沈宏博士审阅了本书内容,史佳先生审核了书中的程序,方歆、庄茜女士完成了文字输入工作,黄阳朝先生、黄素娴女士校对了全部文字和图片。

史斌星　余加莉

一九九八年八月于清华大学

内 容 简 介

菜单技术和面向对象的程序设计(OOP)是当前乃至今后很长时期内计算机软件的两个重要话题,菜单是 GUI 的重要组成部分,OOP 技术则是当今编程人员之必备。本书着重揭示菜单技术与 OOP 的有机联系。突出面向对象程序设计的概念和方法。使读者对当今最重要的编程语言——C + +,与编程方法——面向对象编程有深入了解。

目　录

上篇　菜单设计基础

下篇　Windows95／98中的菜单设计

上篇　菜单设计基础

第一章　计算机的图文显示

所有菜单都是由图像和文字组成的。不言而喻，菜单技术的一个重要方面是图像和文字的显示。本书将它作为叙述的开端。

PC 机显示器的显示是通过视频适配器实现的。视频适配器又称为图形卡或显示卡，它由硬件电路构成，具有控制显示器的功能，作为一个插件被安装在计算机中。显示卡的种类很多，早期的都是单色图形适配器，现在多为彩色图形适配器。例如有 CGA、EGA、VGA、TVGA、PVGA、SVGA、EVGA 和 XGA 卡等。它们具有的不同性能，将在 §1.2 中介绍。

§1.1　文本方式和程序编译

计算机有两种显示方式，分别称为文本方式和图像方式。其中文本方式是一种以字符为基本单元的输出方式；图像方式适用于图像显示，但也可以进行字符显示。对于 PC 机而言，一般情况下文本方式是缺省方式，也称作默认方式，即开机后机器自动进入这种显示方式。如果在 DOS 操作系统下工作，开机后首先需要键入 DOS 命令。DOS 命令是由字符组成的，因此，这时文本方式比较方便。当然，也可以设置图像方式为缺省方式。如果要将某一种方式转换为另一种方式则需要重新设置。在图像方式下也可以显示字符，但在文本方式下却不能显示图像，这是不是说只要图像方式就可以了呢？不是这样的。在图像方式下显示字符与文本方式下显示字符是不同的，它把字符也当做一个图像来显示，也就是说一个字符相当于一个图像。在文本方式下西文字符由字符发生器生成。字符发生器是显示电路的一部分，因此是硬件产生的，这时有很快的显示速度。图形方式下，字符由软件生成，它将构成字符的像素一个个地画出来，相对来说，显示速度慢一些。因此，两种显示方式各有特点。

菜单中有图像、也有字符，因此菜单设计都采用图像方式。

§1.1.1　文本方式

我们只讨论西文字符。文本方式的基本输出单元是字符，数字计算机只和数字打交道，因此每个西文字符都用一个数码表示，称为该字符的 ASCII 码或该字符的代码。总共有 256 个西文字符，其中包括 26 个英文字母，考虑大小写共 52 个，还有 0 至 9 共 10 个数字字符，剩下的便是包括标点符号在内的各种符号字符。用两位 16 进制数正好可以表

示 256 个字符的 ASCII 码，例如英文小写字母 a 的 ASCII 码是 61、大写字母 A 的 ASCII 码是 41。其中 61 和 41 都是 16 进制数，换算成十进制数分别是 97 和 65。图 1.1.1 中给出了全部 256 个字符，其中有的字符是很少用到的，看起来有点奇形怪状。和这 256 个字符对应的 ASCII 码也可以从图中得到，行的值是高位数，列是低位数。注意都是从 0 行 0 列开始。例如第一行第一列的 ASCII 码为 00，字符 A 在第 5 行第 2 列，所以 ASCII 码为 41，依此类推，不难得到所有字符的 ASCII 码。通常只用上半区的 ASCII 码，往往将下半区的 ASCII 码留作它用。下半区在第 9 行开始，ASCII 码的高位从 8 开始，也就是说高位在 8 以后的 ASCII 码可改作它用。

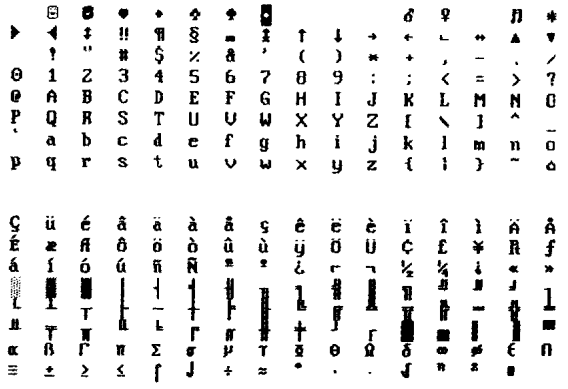

图 1.1.1　256 个 ASCII 字符

文本方式下有各种不同模式的显示方式，它们具有不同的分辨能力和不同的色彩。表 1.1.1 给出了各种模式的特征。其中“色彩”栏目内的数字表示该模式提供色彩的种类。“列 × 行”栏目内的数字表示该模式的分辨能力，即在整个屏幕上能够显示的字符的列数和行数。列数和行数越多表示该模式的分辨能力越大。“模式名”栏目内的英文名称称作符号常量，顾名思义它们代表一些常量。符号常量和相应的数字等价。将常量用符号表示的一个好处就是，可以从符号常量看出数字的含义。表 1.1.1 中的符号常量表示各自右边“数值”栏目中的数值。C 语言中，通常是在以 h 为扩展名的头文件中说明符号常量所代表的数值。例如表 1.1.1 中的符号常量是在头文件 conio.h 中说明的，因此，在选用文本方式的显示模式时必须同时包括头文件 conio.h，只有这样，机器才能了解这些符号常量所代表的数值。归根到底，计算机只能识别数字，如果符号常量不和它们所代表的数值联系起来，对计算机来说就毫无意义。有了头文件中的说明，在选用模式时既可采用符号常量也可直接采用它代表的数值，因为两者是等价的。建议读者尽量选用符号常量。符号常量比数值有更好的可读性，可以一目了然地判断是选用了那一种模式。而数值的含义是不明显的。表 1.1.1 中模式 C4350 的分辨率与机器配备的图形卡有关，对于 VGA 卡是 80 × 50，而 CGA 卡则是 80 × 43。

可以通过一个程序测定机器设置的缺省模式，我们将这个程序的源文件命名为 pg111.c。

本书中所给程序都是用 C 语言编写的，对于尚不熟悉 C 语言的读者，请先阅读有关书籍。例如，在阅读程序 pg111.c 之前最好先学习一下程序结构、程序编译、定义、说明、引用、调用、头文件、数据类型、数据的输出和输入、函数、指针以及有关的句法等。如果您

表 1.1.1 文本方式下的显示模式

模式名	数值	色彩	列×行
BW40	0	黑白	40×25
C40	1	16	40×25
BW80	2	黑白	80×25
C80	3	16	80×25
MONO	7	黑白	80×25
C4350	64	16	CGA 80×43
			VGA 80×50

觉得阅读此书没有困难就说明您的预备知识已经够了。

pg111.c 中的行号是为了便于对程序进行说明而加入的,在对该程序进行编译前必须将它们除去,否则不能通过编译。pg111.c 是一个很简单的程序。其中第一行是一个说明语句,说明 text _ info 是一个结构数据类型。关键字 struct 是结构的意思。C 语言要求变量、函数等在引用之前必须预先进行说明,否则不能通过编译,这时编译器会告诉您某某变量或函数没有定义。因此第一行的说明语句是不可缺少的。结构是 C 语言的一种常用的数据类型,在头文件 conio.h 中定义了这个结构:

```
struct text _ info{
        unsigned char currmode;
        unsigned char screenwidth;
        unsigned char screenheight;
        unsigned char winleft;
        unsigned char wintop;
        unsigned char winright;
        unsigned char winbottom;
        unsigned char attribute;
        unsigned char normattr;
        unsigned char curx;
        unsigned char cury;}ti;
```

text _ info 是结构的名称,末尾的 ti 是该结构的变量,圆括号内各参量称为该结构的数据成员。同一个结构可以包含许多变量,所有这些变量都具有相同的数据成员。本结构只定义了一个变量 ti,ti 的数据成员表示的是显示模式的各种参量。其中 currmode 是机器当前所选用的显示模式;screenwidth 和 screenheight 分别是显示屏幕的宽度和高度;winleft、winright、wintop、winbottom 分别为屏幕的左部、右部、顶部和底部边界的位置;attribute 和 normattr 分别称为文本属性和正常属性,暂时可不必管它;curx 和 cury 为当前光标在窗口中的位置。程序 pg111.c 的全部清单为:

pg111.c

```
# include < conio.h >
# include < stdio.h >
main( ){
1 struct text _ info ti;
2 gettextinfo( &ti);
3 printf("mode          %2d \ n",ti.currmode);
4 printf("win _ width   %2d \ n",ti.screenwidth);
5 printf("win _ height  %2d \ n",ti.screenheight);
6 printf("win _ left    %2d \ n",ti.winleft);
7 printf("win _ top     %2d \ n",ti.wintop);
8 printf("win _ right   %2d \ n",ti.winright);
9 printf("win _ bottom  %2d \ n",ti.winbottom);
10 printf("attribute    %2d \ n",ti.attribute);
11 printf("normattr     %2d \ n",ti.normattr);
12 printf("currentx     %2d \ n",ti.curx);
13 printf("currenty     %2d \ n",ti.cury);
14 getch( );
15 return;}
```

C 语言有强大的功能,上述数据可以通过第 2 行中调用函数 gettextinfo( )一次性地测得。函数 gettextinfo(&ti)是本程序的核心部分,它是 C 语言的一个库函数,可在库函数手册或 Borland C + + 3.1 版本的 Help 文件中查到有关说明。下面让我们来看看它的调用方法。该函数的原型为:

```
# include < conio.h >
void gettextinfo(struct text _ info * r);
```

本书第一次介绍某个函数时,都先写出它的原型。# include < conio.h > 表示如果要调用 gettextinfo( )函数,则程序中必须加上这一语句,以将头文件 conio.h 包含在程序中。因为结构 text _ info 是在这个文件中定义的,函数 gettextinfo( )也在这个文件中进行说明。void 表示函数 gettextinfo( )无返回值,r 为结构 text _ info 的一个指针变量(即定义的结构变量是一个指针),应该是一个地址,所以在 pg111.c 中调用该函数时采用了 &ti。符号 & 表示地址,&ti 就是结构变量 ti 的首地址。通过这个调用便测得了结构 text _ info 中的各项数据。从程序的第 3 至第 13 行都是调用格式化的输出函数 printf( ),将所测得的结果在屏幕上显示出来。stdio.h 是调用函数 printf( )时必须引用的头文件。14 行使得程序有一个暂时停顿,让读者有时间阅读屏幕上显示的内容,按任意键后便继续往下执行程序,即退出程序。

前面短短几行说明包含了不少重要概念,例如头文件、函数返回值、指针变量、结构变量的首地址以及格式化输出函数等,如果对它们还不熟悉请阅读有关书籍。这里需要特别强调,指针是一个尤为重要的概念,它是整个 C 语言的基础,因此如果不清楚,一定要

花时间把它搞清楚。

假设我们在某一台 PC 机上运行程序 pg111.c 后得到了下面一组数据，看看从中可以得到了哪些有用的信息。

mode	3
win _ width	80
win _ height	25
win _ left	1
win _ top	1
win _ right	80
win _ bottom	25
attribute	7
normattr	7
currentx	1
currenty	25

这个结果告诉我们该机器的缺省文本格式是 C80(mode = 3)，表 1.1.1 告诉我们模式 3 共有 16 种色彩，分辨率为 80 列和 25 行。上述结果也指出，win _ width = 80，win _ height = 25，两者结果相同。win _ left = 1，win _ top = 1，win _ right = 80，win _ bottom = 25，即窗口的最小横坐标为 1 在左端，最小纵坐标也为 1 在上端，最大横坐标为 80 在右端，最大纵坐标为 25 在下端。所以，横向坐标自左至右增加，纵向坐标自上至下增加。横向和纵向坐标值也就是字符的列和行的数值，也就是说，在文本格式中坐标值与字符所在的行和列的位置一致，例如，某字符的坐标为(40,12)表示该字符在第 40 列 12 行的位置上。currentx 和 currenty 的值表示光标所在的位置，它们分别为 1 和 25 表示光标在屏幕的左下角上。建议读者用这个程序对自己机器的缺省模式作一测定，一方面可以加深对本节的理解，另一方面可以更好的了解自己的机器。

pg112.c 给出了显示全部 256 个西文字符的程序，这个程序并不难理解。其中 stdio.h 是格式化的输出语句 printf( )所需要的头文件，所以，在程序的开头应有语句 #include <stdio.h>；头文件 conio.h 是第 9 行中 getch( )函数所需要的，执行 getch( )后程序暂停运行等待键盘输入，击任意键便退出程序。函数 printf( )中参数"%c"要求将 ASCII 码以字符形式输出，而不是以数值形式输出。第 2 和第 6 行的两个 for 循环是为了将 256 个字符分成两组输出，以便于阅读，每个组中包含 128 个字符。图 1.1.1 所给出的便是由这个程序得到的结果。

pg112.c

```
#include<stdio.h>
#include<conio.h>
main( ){
1   int i,j;
2   for(j=0;j<8;j++){
3       for(i=0;i<16;i++) printf(" %c",(j*16+i));
```

```
4       printf("\n");}
5   printf("\n"); printf("\n");
6   for(j=8;j<16;j++){
7       for(i=0;i<16;i++) printf("%c",(j*16+i));
8       printf("\n");}
9   getch();
10  return;}
```

C 语言的软件包提供了一系列为在文本模式下编程用的库函数，例如，前面用过的 gettextinfo( )函数就是其中之一，此外还包括具有模式选择、窗口管理、属性控制、光标定位以及文本输出、输入等功能的各种函数。下面再介绍几个与本书有关的基本函数，其余函数读者可在 C 语言的库函数手册或 C 语言的 Help 文件中找到。

前面介绍了六种文本模式，如果需要将机器设置成其中的某一种模式，便可调用函数 textmode，该函数的原型为：

```
#include <conio.h>
void textmode(int mode);
```

conio.h 为调用这个函数所需的头文件。void 表示此函数无返回值。int mode 表示变量 mode 应是一个整形变量，与六种文本模式对应，mode 的值分别为 0、1、2、3、7、64；也可以'符号常量'表示，分别为 BW40、C40、BW80、C80、MONO、C4350。

除此之外与在屏幕上进行文本处理有关的函数还有 window( )、clrscr( )、gotoxy( )、textbackground( )、textcolo( )等，它们的原型分别为：

```
#include <conio.h>
void window(int left,int top,int right,int bottom);
void clrscr(void);
void gotoxy(int x,int y);
void textbackground(int color);
void textcolor(int color);
```

conio.h 是所有这些函数的共同的头文件。函数 window( )的功能是建立一个窗口，该窗口的范围由 left(左)、top(顶)、right(右)、bottom(底)等四个参量给出。缺省值为整个屏幕大小，即如果不给任何数据，将建立一个与显示器屏幕一样大小的窗口。需要注意的是，执行 window( )的结果只是在屏幕中指定了一块窗口面积，暂时在屏幕上看不见这个窗口；只有当要在屏幕上显示字符时或改变屏幕颜色时，才能发现这一窗口的存在，所有上述显示字符和改变颜色的操作被限制在 window( )所规定的窗口范围内。程序 pg113.c 演示了这个性质。函数 clrscr( ) 的功能是清除整个屏幕上的所有字符，gotoxy( )是将光标移动到 (x,y)处，textbackground( )用来选择屏幕的背景颜色，textcolor( )用以确定字符的颜色。其中整形变量 int color 可取的数值与所选显示模式有关，例如，对于具有 16 种颜色的模式，它们的值如表 1.1.2 所示：

表 1.1.2　16 种颜色的模式

符号常量	数　值	颜　色
BLACK	0	黑
BLUE	1	蓝
GREEN	2	绿
CYAN	3	青
RED	4	红
MAGENTA	5	品红
BROWN	6	棕
LIGHTGRAY	7	浅灰
DARKGRAY	8	深灰
LIGHTBLUE	9	浅蓝
LIGHTGREEN	10	浅绿
LIGHTCYAN	11	浅青
LIGHTRED	12	浅红
LIGHTMAGENTA	13	浅品红
YELLOW	14	黄
WHITE	15	白

color 虽同是函数 textbackground 和 textcolor 的变量，但可取的数是不同的。对于背景颜色，color 只能取表 1.1.2 中的前八个数，总共只有八种不同背景色。对于字符，color 可以选取上述十六个数中的任何一个数，字符有十六种不同的颜色。

pg113.c 将缺省的文本模式改变成为 C40 模式，并将该模式全部（八种）背景色显示一遍。程序清单如下：

pg113.c

```
#include <conio.h>
main( ){
1   int backcolo;
2   textmode(C40);
3   clrscr( );
4   window(1,1,20,25);
5   for(backcolo = 1;backcolo <  = 8;backcolo + + ){
6       textbackground(backcolo);
7       getch( ); clrscr( );}
8   textbackground(BLACK);
9   textmode(3);
10  return;}
```

程序的第 1 行对常数 backcolo 进行说明。第 2 行调用函数 textmode( )建立一个 40 列 25 行的模式 C40。第 3 行是清除原来的屏幕。第 4 行在屏幕上开了一个 20 列 25 行的窗口,第 5、6、7 行不断调用 textbackground( )函数,每按一次任意一个键,都要清一次屏幕,随着参量 backcolo 的变化背景色也不断改变,共出现八种不同的颜色。注意,所有被显示颜色均在函数 window( )所定义的窗口范围。最后,第 8、9 行在退出程序前将屏幕颜色和模式恢复为原先的状态,我们假设机器的缺省模式为 mode 3,屏幕的缺省颜色为黑色。

pg 114.C 是改变字符颜色的程序,通过不断调用函数 textcolor( )实现。前面指出,共有 16 种字符颜色,它们可用四个二进制数加以区分,这四位数由参量 color 的最低四位数(即 0、1、2、3 位)表示。程序在开始时已经说明 color 是一个字符(char)型变量,char 型量由八个二进制数(即一个字节)组成,16 种颜色只需 4 个低位数表示,占用了 color 字节的 0、1、2、3 位,尚有 4 个高位数未被占用。此外,还有八种背景色,应由 3 个二进制数加以区分,它们占据了 color 字节 的 4、5、6 位。只有最高位(7 位)尚未被占用。如每个二进制数以'x'表示,则 color 应表示为 0xxxxxxx 或 1xxxxxxx(见表 1.1.3);128 的二进制数为 10000000,0 的二进制数为 00000000,这两个数的差别在最高位。textcolor( )函数中以参数 color 的最高位表示字符的闪烁属性,如果为 1 ,则字符进行闪烁;如果为 0 ,则不闪烁。符号 color|128 表示 color 和 128 两个数的"位或"运算,无论 color 是什么值,该运算的结果恒为 1xxxxxxx,是一个最高位为 1 的闪烁字符;另一方面,"位或"运算 color|0 的结果恒为 0xxxxxxx,是一个最高位为 0 的不闪烁字符。这两种运算都不改变 color 的低七位数,即不改变背景色和字符色。有了上述认识就不难对程序 pg 114.C 进行分析了。

表 1.1.3 字符型变量 char color 的结构

位 序	7	6,5,4	3,2,1,0
数 值	0 或 1	xxx	xxxx
含 义	闪烁控制	背景颜色	字符颜色

pg114.c

```
#include<conio.h>
#include<stdio.h>
void main( ){
1   char color;
2   textmode(C40);
3   for(color=0;color<=7;color++){
4       textcolor(color|128);
5       clrscr( );
6       printf("This Is The Experiment of Color Test \n");
7       printf("color=%d \n",color);
8       getch( );}
9   for(color=8;color<=15;color++){
```

```
10        textcolor(color|0);
11        clrscr( );
12        printf("This Is The Experiment of Color Test \ n");
13        printf("color = %d \ n",color);
14        getch( );}
15   textmode(3);}
```

第 1 行说明 color 是字符型变量,第 2 行把机器设置为文本模式 C40,这时应有十六种字符颜色,第 3 至 8 行显示字符的前八种颜色,并且要求字符闪烁,9 至 14 行显示字符的后八种颜色,字符不闪烁。第 15 行在退出程序前将字符颜色和模式恢复为原先的缺省状态,我们假设机器的缺省模式为 mode 3,字符的缺省颜色为白色,程序结束时 color 的值正好是 15(白色),无需再作改变。当执行这个程序时,字符"This Is The Experiment of Color Test"显示了十六种颜色,前八种是闪烁的,后八种不闪烁。

§1.1.2 Borland C + + 的集成开发环境

完成了对程序 pg111.c 至 pg114.c 的编写以后,还不能运行,运行程序之前还必须将它们编译成扩展名为 .exe 的执行文件。完成程序的编译工作的途径很多,但是最方便的要算利用集成开发环境的方法了。这种工具已为愈来愈多的人所喜爱,下面就其最基本的以及和本书有关的部分进行介绍。

通常将集成开发环境简称为 IDE,它是英文 Integrate Development Environment 的缩写。所谓集成开发环境,是指集文本编辑(Edit)、源程序编译(Coinpile)、文件链接(Link)、项目管理(Project)、程序调试(Debug)和运行(Run)等功能于一体的程序开发软件包。

本书上篇介绍 DOS 操作系统中的程序设计,Borland C + + 的各种版本中,Borland C + + 3.1 是能在 DOS 操作系统中运行集成开发环境的一个版本,它以后的版本都只能在 Windows 下运行集成开发环境。本书下篇介绍 Windows95 和 Windows98 中的程序设计,目前情况下, Borland C + + 的各种版本中,Borland C + + 5.0 是直接支持 Windows95/98 的版本。在它之前的版本例如 Borland C + + 4.5 虽然也能在 Windows 95 下安装和运行,但它是在 Windows95 以前的版本下开发的,只有 Borland C + + 5.0 最能应用 Windows 95 和 Windows98 的新特色。在上篇中我们应用 Borland C + + 3.1 的集成开发环境,在下篇中我们应用 Borland C + + 5.0 的集成开发环境(见第 8 章)。

下面按照编译一个程序的操作过程一步一步骤地进行介绍。

(1) 进入 IDE

进入 Borland C + + 3.1 的 IDE 最常用的方法是:首先进入 Borland C + + 3.1 的 bin 子目录,然后在键盘键入指令:bc,接着单击 Enter(回车)键。将上述步骤采用以下链式表示方法:

"cd c: \ borlandc \ bin" / bc / Enter

这里假设 Borland C + + 3.1 的软件被放在 C 盘的 borlandc 子目录中,cd c: \ borlandc \ bin 是 DOS 操作命令,表示进入 Borland C + + 3.1 的 bin 子目录。bc 是打开 IDE 的命令,Enter 表示按击 Enter(回车)键。采用这种简洁的方法表示一连串的操作是很方