《电脑报》实用技术丛书

Dos下的仿Windows界面設計

——C语言高级实用技术

- 仿 Windows 弹出式窗口及菜单设计方法
- 建立下拉式菜单系统原理及制作方法
- 非汉字操作系统下对汉化性目录函数的编制
- 图形方式下鼠标应用基本技术

温海燕　张　雁　编著

成都科技大学出版社

DOS 下的仿 Windows 界面设计

—— C 语言实用技术

温海燕　张　雁　编著

成都科技大学出版社

[川]新登字 015 号

责任编辑　唐　勇
封面设计　李光宇

内容简介

本书从实用的角度出发，通过大量的有用程序诠释了 C 语言的汉字处理、特殊的弹出式菜单及窗口的制作、在非汉字操作系统下的有关字段编辑、仿 Windows 界面设计等方面的技术。书中的示例性源程序，给出了注解，便于理解其设计思想和框架。学习本书，能够领会 C 语言的各种编程技巧，熟练掌握 C 语言的编程方法，使自己具有使用 C 语言编制非汉字操作系统下的汉化应用性软件等方面的能力。

DOS 下的仿 Windows 界面设计
温海燕　张　雁　编著
成都科技大学出版社出版
电脑报社照排部排版
重庆华林印务有限公司印刷
四川省新华书店经销
*
开本　787×1092 毫米　1/16　印张 18.5　字数 464 千字
版次　1996 年 6 月第一版　印次　1996 年 6 月第一次印刷
ISBN 7-5616-3156-1/TP·147

定价：22.00 元

前　言

C语言虽然流行，在市面上有关C语言应用的书籍也比比皆是，但是涉及到C语言的汉字处理技术、特殊弹出菜单及窗口制作、在非汉字操作系统下有关汉字字段编辑技术方面的书籍却很少。而这些技术，却是程序员在C语言编程中常常需要解决的问题。笔者正是针对这些情况，根据自己多年的C语言编程经验撰写了本书。在编写中，为了便于理解，笔者附上了大量的实例，读者可根据自己的需要，将这些实例加入到自己的程序中去，使编程工作更加容易。

本书共分六章，第一章主要介绍一些C语言基本的图形函数，怎样建立图形应用性程序及其图形方式下对屏幕的管理、字符的输入输出等问题，以便更好地理解后面章节的内容。

第二章实用汉字技术，从分析汉字库的基本结构、汉字处理技术的基本知识着手，介绍怎样利用C语言建立非汉字操作系统下的应用程序，较全面地阐述了汉字显示及汉字输入的技术问题。为读者提供了建立自己的汉字系统的方法和一个完整的C语言源程序，其中包括怎样建立全拼、缩拼、区位及双字节图符性文字输入技术，汉字直接写屏，汉字打印技术及具有光标前后移动、文字前删及后删等完整编辑功能的汉化字段编辑器。并附上了用C语言编制的汉字模块进行开发的较实用的应用程序，可帮助读者从认识到理解，从理解到实际应用，使读者能够在用C语言编制非汉字操作系统下的汉化性软件的过程中，轻松愉快。

第三章介绍在DOS下仿Windows的弹出式窗口及菜单的设计方法，从弹出式窗口及菜单的基本原理出发，介绍怎样建立以栈的方式存取及以文件存取方式的弹出式窗口。特别介绍了以文件存取方式的弹出式窗口设计方法，它仅需要很小内存，就能够建立无数的弹出式窗口及菜单。此外，还介绍了在图形方式下，利用画线及矩形函数去制作高效、简便的立体菜单及窗口，即仿Windows的视窗系统。对位图的基本格式和在建立仿Windows视窗中的应用，包括BMP及PCX格式的图象文件的应用也作了相应介绍。在本章的后面附上了大量的应用实例，包括建立各种弹出式菜单、完整的C源代码及其应用实例。使读者通过本章学习后，能够自行编制图形方式下含有任何菜单、窗口（包括图符性菜单，按钮型菜单及窗口）的应用性程序。

第四章介绍怎样建立下拉式菜单系统的原理及其制作方法，包括在菜单中怎样进行字符控制、光条控制的方法，附上了建立汉化下拉式菜单系统的C源程序及应用实例。

第五章介绍在非汉字操作系统下对汉化性目录函数的编制。利用前面所编制的C语言汉字技术来编制汉化文件搜索窗口（即汉化文件列表选择），阐述了建立这些应用函数的原理，并给出了C语言全部源代码和应用这些函数的示例性应用程序。

第六章介绍了在图形方式下鼠标应用的基本技术及给出了在标准图形模式下的编程实例。

本书的读者对象：已基本熟悉或掌握了C语言的基本编程技术，而想在PC机DOS操作系统下，用C语言开发更完美的软件，或者想更深入地了解C语言的一些编程方式和使用技巧的大学生、研究生及软件开发人员。

本书中的程序均是用TURBO C 2.0编写。读者对本书程序中的代码，稍加修改，可移植到任何C编译器中。本书中的程序是在286及以上档次，VGA显示方式的微机上调试通过。

温海燕

1996年4月

目 录

敬告读者:为方便大家的学习和编程,本书中的所有源程序,均已制作成 1 张 5 英寸软盘。需要者请直接与电脑报社软件部联系购买,每套 20 元。联系地址:四川省重庆市人民路 236 号电脑报社软件部
邮编:630015 电话:(0811)3876722

第一章　TURBO C 基本图形函数简介

TURBO C 的基本图形函数按功能大致可分为：图形系统控制函数、画图和图形填充函数、屏幕及视口管理函数、图形方式下的本文输出函数、颜色控制函数、错误处理函数及状态查询函数七大类。在这里，我们仅讨论与本书将要涉及的内容相关的部分函数，以加深理解。

1.1　图形的初始化及关闭

使用图形函数前，必须把视频适配器设置为某种图形方式，这种设置过程称为图形的初始化。其初始化函数为：

```
void far initgraph(int far * graphdriver,int far * graphmode,char far * pathtodriver)
```

函数中的第一个变量(graphdriver)是指要连接的图形驱动程序名，该参数在 TURBO C 中，共有 11 个值，其符号与值的对应关系如下表所示，从符号名就可以知道相应的驱动程序适用于哪一种显示卡。

表 1.1

驱动程序常量	对应值
DETECT	0
CGA	1
MCGA	2
EGA	3
EGA64	4
EGAMONO	5
IBM8514	6
HERCMONO	7
ATT400	8
VGA	9
PC3270	10
CURRENT_DRIVER	−1

第二个参数，是设置图形模式，它也是一个整数指针，它说明用哪一种图形显示方式。如果在第一参数 graphdriver 指定为 DETECT 时，它则自动选择最高分辨率的显示方式。例如显示卡如果是 VGA 显示卡时，屏幕的分辨率则定为 640×480。下表列出了 TURBO C 定义的显示方式。

表 1.2

驱动程序号	显示方式	方式值	分辨率	调色板	总页数
CGA	CGAC0	0	320×200	C0	1
	CGAC1	1	320×200	C1	1
	CGAC2	2	320×200	C2	1
	CGAC3	3	320×200	C3	1
	CGAHI	4	320×200	2 色	1
MCGA	MCGAC0	0	320×200	C0	1
	MCGAC1	1	320×200	C1	1
	MCGAC2	2	320×200	C2	1
	MCGAC3	3	320×200	C3	1
	MCGAMED	4	640×200	2 色	1
	MCGAHI	5	640×200	2 色	1
EGA	EGALO	0	640×200	16 色	4
	EGAHI	1	640×350	16 色	2
EGA64	EGA64LO	0	640×200	16 色	1
	EGA64HI	1	640×350	4 色	1
EGA-MONO	EGAMONO-HI	3	640×350	2 色	1(64K)
	EGAMONO-HI	3	640×350	2 色	2(256K)
HERC	HERC-MONOHI	0	720×348	2 色	2
ATT400	ATT400C0	0	320×200	C0	1
	ATT400C1	1	320×200	C1	1
	ATT400C2	2	320×200	C2	1
	ATT400C3	3	320×200	C3	1
	ATT400CMED	4	640×200	2 色	1
	ATT400CHI	5	640×400	2 色	1
VGA	VGALO	0	640×200	16 色	2
	VGAMED	1	640×350	16 色	2
	VGAHI	2	640×480	16 色	1
PC3270	PC3270HI	0	720×350	2 色	1
IBM8514	IBM8514LO	0	640×480	256	
	IBM8514HI	1	1024×768	256	
DETECT	用于硬件测试	0			

第三个参数 pathtodriver 是说明驱动程序的路径。如果该项参数为 NULL，则说明在当

前目录下寻找图形驱动程序。

在 DOS 环境下，一般缺省为文本方式，故在运行了图形方式的程序后，需要关闭图形状态。在 TURBO C 中确定该项工作由下面函数完成。

void far closegraph(void);

closegraph() 函数释放图形系统分配的存储区，恢复调用 initgraph() 之前的模式。另外，下面函数也可返回调用 initgraph() 函数前的文本模式：

void far restoremode(void);

但是，这两个函数有本质性的区别，前者关闭图形状态的同时，还释放了图形驱动程序、字体占用的内存，而后者并不释放这些内存。后者常用于不同图形模式的转换，或图形软件的子进程中。

下面的例子将说明这几个函数的用法：

```
/* ------------- WHY1_1.C ------------- */
#include <graphics.h>
main()
{
   int gdriver=DETECT, gmode;
   /* 图形初始化 */
   initgraph(&gdriver,&gmode,"c:\lc\lib");
   setbkcolor(BLUE); /* 设置背景色 */
   setcolor(RED);    /* 设置画线颜色 */
   bar3d(100,100,400,300,0,0); /* 画一个长方形 */
   getch();
   closegraph();     /* 关闭图形模式 */
}
```

1.2 建立独立的图形运行程序

TURBO C 对用 initgraph() 函数直接进行图形初式化的程序，在编译及连接过程中并没有将相应的图形驱动程序(*.BGI)装入到运行的程序中去，在程序运行到 initgraph() 函数时，首先是从该函数的第三个参数 pathtodriver 中所规定的路径去找相应的驱动程序，如果没有在规定的路径及当前目录中找到驱动程序，则在执行程序运行中将会出现如下错误信息：

BGI Error: Graphics not initialized (use 'initgraph')

因此，对于建立一个不需要驱动程序，可独立运行的可执行图形程序是有必要的。其具体方法如下(这里以 EGA 及 VGA 显示卡为例)：

首先，在目录中找到 BGIOBJ.EXE 及 EGAVGA.BGI 两文件，输入如下命令：

C:>BGIOBJ EGAVGA

此命令是将 EGAVGA.BGI 转换为 EGAVGA.OBJ 目标文件。之后将目标文件加入到你的图形库中(如下所示)，便于连接及编译。

C:GRAPHICS.LIB+EGAVGA.OBJ

最后还需要在所编译的程序中 initgraph() 函数之前加上一句

registerbgidriver(EGAVGA_driver);

语句，以此告诉连接程序时，把 EGAVGA 的驱动程序装入用户的执行程序中去。

如果我们把前面的例子程序 WHY1－1.C 改为可独立运行的图形执行程序时，其修改后的情况如下：

```
/*－－－－－－－－－－－－－－ WHY1_2.C －－－－－－－－－－－－－－－－*/
#include <graphics.h>
main()
{
   int gdriver=VGA,gmode=VGAHI;
   registerbgidriver(EGAVGA_driver); /* 建立独立的图形运行程序 */
   initgraph(&gdriver,&gmode,"");
   setbkcolor(BLUE);
   setcolor(RED);
   bar3d(100,100,400,300,0,0);
   getch();
   closegraph();
}
```

1.3 图形状态下的色彩选择及设置

对于图形模式的屏幕颜色设置，可分为背景色与前景色的设置。TURBO C 中分别由下列函数完成其功能。

void far setbkcolor(int color)；设置屏幕背景色

void far setcolor(int color)；设置屏幕前景色

两个函数在不同的图形驱动器下，它的参数变量(color)的取值范围是不同的。在 CGA 显示卡上只能有 4 种选择(参见有关 TURBO C 使用手册)，在 VGA 上有 16 种色彩选择，见下表所示：

表 1.3

颜色名	对应值
BLACK	0
BLUE	1
GREEN	2
CYAN	3
RED	4
MAGENTA	5
BROWN	6

LIGHTGRAY	7
DARKGRAY	8
LIGHTBLUE	9
LIGHTGREEN	10
LIGHTCYAN	11
LIGHTRED	12
LIGHTMAGENTA	13
YELLOW	14
WHITE	15

各种模式，其背景色缺省时为黑色，可以通过 setpalette 来改变。

函数 setbkcolor() 及 setcolor() 在前面的 WHY1—1.C 及 WHY1—2.C 这两个例子中已体现了它的用法，这里也就不再举例说明。

在图形色彩设置中，另外有几个相关的函数，也需要提一提，它们是：

int far getmaxcolor(void) 获取最高可用的颜色值

int far getbkcolor(void) 获取背景的颜色值

int far getcolor(void) 获取现行作图的颜色值

这三个函数的具体用法，在后面章节中将会了解到。

1.4 基本图形函数

基本图形函数包括画点、线、矩形以及其它一些基本图形的函数，本节对这些函数将作一简要的介绍。

画点函数：

画点函数的调用格式为

void far putpixel(int x,int y,int color);

该函数是在指定的象元画一个按参数 color 所指定颜色的点，而 x,y 是指图形象元的坐标。与 putpixel() 相应另一个关于点的函数是

int far getpixel(int x,int y);

它是获取当前点 (x,y) 的颜色值。

这两个函数的使用技巧或具体用法，在后面的章节中将要详细介绍。

画线函数：

画线函数主要有两个函数，即：

void far line(int x1, int y1, int x2, int y2);

void far lineto(int x, int y);

前者是从点 (x1, y1) 到点 (x2, y2) 画一条直线，后者是从当前坐标处画一条到点 (x, y) 处的直线。这两个函数画线的颜色是由 setcolor() 函数确定。

另外与画线函数相关的还有如下三个函数：

void far setlinestyle(int linestyle,unsigned upattern,int thickness);

设定线形函数，它包括设置线型的宽度和形状，其参数值参见有关手册。

void far getlinesettings(struct linesettingstype far * line info);

用来获取当前的线型和宽度。

void far setwritemode(int mode);

该函数用来规定画线的方式。如果 mode=0，则表示画线时将所画位置的原来信息覆盖，如果 mode=1，则表示画线时用现在特性的线与所画之处原有的线进行异或(XOR)操作，也就是指画出的线是原有线与现在规定的线进行了异或后的结果。

画矩形及填充函数：

画矩形函数常用的有以下三个

void far bar(int ux,int uy,int dx,int dy);

void far bar3d(int ux,int uy,int dx,int dy,int depth,int topflag);

void far rectangle(int ux,int uy,int dx,int dy);

这三个函数中，后两个函数较常用，本书后面的章节中主要用于画图形窗口及菜单。在应用中，它常与填充函数配合应用。与本书有关的填充函数主要有如下三个

void far setfillstyle(int pattern, int color);

该函数是用于设置填充的式样和颜色。color 必须是当前屏幕模式的有效值，pattern 值及其等价宏如下所示：

表 1.4

宏	值	含义
EMPTY_FILL	0	用背景色填充
SOLID_FILL	1	实填充
LINE_FILL	2	线“—”填充
LTSLASH_FILL	3	斜杠填充(阴影线)
SLASH_FILL	4	粗斜杠填充(粗阴影线)
BKSLASH_FILL	5	粗反斜杠填充(粗阴影线)
LTBKSLASH_FILL	6	反斜杠填充(阴影线)
HATCH_FILL	7	网格线填充
XHATCH_FILL	8	斜网格线填充
INTERLEAVE_FILL	9	间隔点填充
WIDE_DOT_FILL	10	稀疏点填充
CLOSE_DOT_FILL	11	密集点填充
USER_FILL	12	用户定义的模式

void far setfillpattern(char far * upattern, int color); 用于设置填充类型

void far floodfill(int x,int y,int border);是指在点(x,y)处开始填充，并且指定其填充的边界由颜色为 border 所定的矩形区域。

在下面的例子中将看到上述函数的一些用法：

```
/* ---------------- WHY1_3.C ---------------- */
#include <graphics.h>
main()
{
   int gdriver=VGA,gmode=VGAHI,k;
   registerbgidriver(EGAVGA_driver); /* 建立独立的图形运行程序 */
   initgraph(&gdriver,&gmode,"");
   setbkcolor(BLUE);   /* 设置背景色 */
   setcolor(RED);      /* 设置画线颜色 */
   setfillstyle(1,GREEN); /* 设置填充模式为实填充，填充颜色为绿色 */
   bar3d(100,100,400,300,0,0); /* 在屏幕左上坐标为(100,100),右下坐标为(400,
                                  300)的区域画一个矩形，矩形的边框颜色为红色
                               */
   setcolor(WHITE);
   line(10,10,300,10);        /* 在屏幕上以起始点为(10,10)处，向点 (300,10)
                                 处画一直线 */
   for(k=0;k<10;k++)           /* 在以 X 坐标 100，Y 坐标 320 为起点，*/
     putpixel(100+8*k,320,WHITE);/* 水平向有每间隔 8 个象素共画 10 个白色的
点 */
   getch();
   closegraph();
}
```

1.5 屏幕及视口管理函数

在文本方式下可以开窗口，而图形模式下对应的窗口被称之为视口，许多图形函数的操作均是相对当前视口而言的。确定视口的函数主要有以下三个：

```
void far setveiwport(int x1,int y1,int x2,int y2,int clip);
void far clearviewport(void);
void far cleardevice(void);
```

第一个函数是用于建立一个相对于屏幕整屏位置的操作窗口，它的坐标是相对于屏幕的左上角和右下角的绝对坐标。参数 clip 为非零时，画的图形如超出视区的边缘，则超出的部分将被截断。

第二个函数是清除当前视区的，并把当前位置恢复到 (0,0)。

最后一个函数是清除整个屏幕，并把当前位置恢复到屏幕的 (0,0)。

下面的例子将说明或看到它的具体用法。

```
/* ---------------- WHY1_4.C ---------------- */
#include <stdio.h>
#include <stdlib.h>
```

```
#include <conio.h>
#include <graphics.h>
main()
{
   int gdriver=VGA,gmode=VGAHI;
   registerbgidriver(EGAVGA_driver); /* 建立独立的图形运行程序 */
   initgraph(&gdriver,&gmode,"");
   setbkcolor(BLUE);   /* 设置背景色 */
   setcolor(RED);      /* 设置画线颜色 */
   setfillstyle(1,GREEN); /* 设置填充模式为实填充，填充颜色为绿色 */
   bar3d(10,10,80,60,0,0);/*在屏幕左上坐标为(10,10),右下坐标为(80,60)的区域
                            画一个矩形，矩形的边框颜色为红色 */
   /* 建立一个视区 */
   setviewport(15,15,115,115,0);
   /* 在视区内再画一个矩形 */
   setfillstyle(1,GREEN); /* 本矩形与前一个矩形大小及坐标都一样， */
   bar3d(10,10,80,60,0,0); /*只是坐标前者是相对与屏幕，而后者是相对于刚刚才
                            建立的视口 */
   /* 按任一键清除视口 */
   getch();
   clearviewport();
   /* 按任一键清除屏幕内容 */
   getch();
   cleardevice();
   /* 按任一键退出程序 */
   getch();
   closegraph();
}
```

另外，有关这几个函数的应用，在第三章弹出式窗口及菜单的设计中，将会更进一步说明它们的用法。

对屏幕及视口管理，以下几个函数是有必要了解的，它们是：

void far setactivepage(int page);

函数 setactivepage() 是用来确定接受 TURBO C 图形函数所输出的屏幕页。

void far getimage(int left,int top,int right,int bottom,void far * bitmap);

函数 getimage() 把屏幕图形部分拷贝到由 bitmap 所指向的内存区域，图形的右上角用坐标 left，top 来表示，右下角坐标为 right 及 bottom。用 imagesize()来确定存储图象所需要的字节数。

unsigend far imagesize(int left,int top,int right,int bottom);

函数 imagesize() 是用来返回存储一块屏幕图象所需要的存储器字节数。该函数一般与 getimage() 联用。

void far putimage(int left,int top,void far * bitmap,int op);

putimage() 函数是把一个事先存储(由 getimage())在 bitmap 所指向的内存中的图象拷贝到启始位置为 x,y 的屏幕上。op 的值将决定图象以何种方式写到屏幕上。其有效值如下表所示：

表 1.5

名字	值	含义
COPY_PUT	0	复制
XOR_PUT	1	与屏幕图象取异或后复制
OR_PUT	2	与屏幕图象取或后复制
AND_PUT	3	与屏幕图象取与后复制
NOT_PUT	4	复制源图象的逆

这几个函数的用法，将在后面的一些例程中介绍，特别是在菜单选择条的制作章节中看到它的使用方法，这里就不再举例说明。

1.6 图形状态下的坐标及字符输出和输入

图象方式的坐标与文本方式不同，前者是以象素为单位，也就是说它的坐标取决于屏幕的显示分辨率，即在 VGA 最高显示模式 (640×480) 下，它的最大 x 坐标是 639，最大 y 坐标是 479。而后者，整屏的最大 x 坐标是 79，最大的 y 坐标是 24。在 TURBO C 中关于图形状态下的坐标相关函数主要有以下五个，即：

```
int far getmaxx(void); /* 获取显示屏幕的最大 x 坐标 */
int far getmaxy(void); /* 获取显示屏幕的最大 y 坐标 */
int far getx(void); /* 获取当前点的 x 坐标 */
int far gety(void); /* 获取当前点的 y 坐标 */
void far moveto(void); /* 把当前位置移到当前视区的 (x,y) 坐标处 */
```

有关在图形方式下，字符输入及输出问题，在 TURBO C 中定义了两个专用函数如下：

```
void far outtext(char far * textstring);
void far outtextxy(int x,int y,char far * textstring);
```

这两个函数中参数 textstring 是指所要输出的字符串，在第二个函数中的 x 及 y 参数是指在当前视区指定的 (x,y)坐标位置处输出字符串。另外这两个函数在视区内显示长度如超过了当前视区则被截断。

除上述两个函数外，还有以下几个与图形方式下的文本输出密切相关的函数，它们是：

```
void far gettexttings(struct textseettingstype far * texttypeinfo);
```

该函数是把有关当前图形文字设置信息放入由 info 所指向的结构中，这个结构定义在 graphics.h。如下所示：

```
struct textsettingstype{
    int font;        /* font type */
    int direction;   /* horizontal or vertical */
    int charsize;    /* size of characction */
    int horiz;       /* horizontal justification */
    int vert; /* vertival justification */
    };
```

其中 font 元素为以下各值之一：

表 1.6

值	字型(Font)
0	8×8 点阵字形(缺省值)
1	三倍笔划字形
2	小号笔划字形
3	无衬线笔划字形
4	黑体笔划字形

方向元素(direction)必须设置为水平文字 HORIZ－DIR（缺省)或垂直文字 VERT－DIR。元素 charsize 是一个确定字符输出大小的系数。horiz 和 vert 元素是指明文字如何在当前位置(CP)上排列。其值如下：

表 1.7

宏	值	含义
LEFT_TEXT	0	CP 在左边
CENTER_TEXT	1	CP 在中心
RIGHT_TEXT	2	CP 在右边
BOTTOM_TEXT	3	CP 在底部
TOP_TEXT	4	CP 在顶部

```
void far registerbgifont(void( * font)(void));
```

该函数用于告诉图形系统，字体已被连接，并且不必再寻找相应的磁盘文件。

```
void far settextjustify(int horiz,int vert);
```

settextjustify()原形在 graphics. h 中。它设置与 CP 有关的字符排列的方式。其值见表 1.7。

```
void far settextstyle(int font,int direction,int charsize);
```

该函数是用于为图形字符函数设置当前字体，同时设置方向和字符大小。其 Font 值见表 1.6。

```
void far setusercharsize(int multx,int divx,int multy,int divy);
```

是确定图形字符大小等级的因子及除数。调用该函数后，每个显示在屏幕上的字符都以其缺省值大小乘以 multx/divx 为其字符的宽，乘以 muly/divy 为其字符的高。在以后章节的汉字显示的程序中将用到此函数。

void far textheight(char far * textstring);

textheight() 函数是以象素为单位返回由 textstring 所指向的字符串高度，它是针对当前字符的字体及大小的。

void far textwidth(char far * textstring);

textwidth() 函数是以象素为单位返回由 textstring 所指向的字符串长度，它是针对当前字符的字体及大小。

在图形方式中，一般不利用基本的输出函数 printf，因为它要破坏画面美观，特别是在图形菜单更不适宜。对于字符输入，C 语言基本的输出函数 scanf 一般也不采用，因为它不回显在屏幕，除非在它之前用了 clrscr() 这个函数。因此，在图形方式下，如果要保留屏幕画面时的文本输出均采用前面所提的图象状态下文本输出的专用函数(outtext 等)，对于图形方式下的文本输入问题将在本书后面的章节中专题讨论。

下面的例子将看到本节介绍的函数的具体用法。

1.7 实例一 图形封面的制作

本节列举一个在实际编程中，应用图形函数的例子，希望能够对你的编程工作有所启发或帮助。

```
/*--------------WHY1_8.c--------------------*/
/*              用 TURBO C 库函数制作程序的图形封面              */
/*--------------------------------------------*/
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <graphics.h>
main()
{
    int k,maxx,maxy,ux,uy,y,x;
    unsigned char *str[]={"Main Menu",
                          "Auther: Wenhaiyan",
                          "First",
                          "Second",
                          "Third",
                          "Quit"};
    void *buf;
    int test=0;
```