潘建华　卫跃文

C语言实用软件界面技术

本书以C语言为工具，介绍图形模式下计算机用户界面的实现方法和编程技巧。

本书突出实用性，实例丰富，所述技术都能全面支持鼠标器操作。

西安电子科技大学出版社

C语言实用软件界面技术

潘建华　卫跃文

西安电子科技大学出版社

1995

(陕)新登字 010 号

内 容 提 要

本书以C语言为工具，介绍图形模式下计算机用户界面的实现方法和编程技巧。主要内容包括C语言图形及颜色调配、鼠标器访问、屏幕图形存取、重叠式窗口、菜单、对话框及屏幕帮助功能等。此外还介绍如图符创建、图形资源利用、字段编辑及西文环境下界面中的汉字技术等。书中所述的所有技术都能全面支持鼠标器操作。

全书以实用性为原则，实例丰富，所附源程序均可直接编译运行。语言力求简练，具有一般C语言编程经验的各类人员均可阅读。可为有关计算机使用及开发人员提供参考，也是初学C语言的一本实用型参考书籍。

C语言实用软件界面技术

潘建华　卫跃文

责任编辑　云立实

西安电子科技大学出版社出版发行
西安市长青印刷厂印刷
新华书店经销
开本 787×1092 1/16　印张 14 6/16　字数 341 千字
1995 年 10 月第 1 版　1995 年 10 月第 1 次印刷　印数 1—5 000

ISBN 7-5606-0407-2/TP·0164　　定价：15.00 元

前　言

软件界面是计算机与用户间通讯的重要综合环境。良好的用户界面对一个应用软件来说，起着关键作用。近年来，以窗口、菜单等一系列方便的软件环境为特点的图形用户界面(GUI)成为软件界面的主流，深受广大计算机用户的喜爱。有关软件界面技术的书籍也不少见，但大多数都是以文本模式为基础或者偏于概念不便实用，能支持鼠标器的有关界面技术介绍就更少。本书以功能强大而灵活的C语言为工具，介绍基于图形模式的界面程序的设计技术和开发技巧。鉴于鼠标器应用日益广泛、快捷且方便，本书中所讨论的各种界面技术都是以全面支持鼠标器操作为前提的。

基于图形模式和全面支持鼠标器是本书技术方面的重要特点。在内容上，实用性是我们追求的目标。书中的大多数内容取材于作者开发一个工程实用电路计算机辅助设计软件的编程实践及近年来的有关文献资料，而且所附源程序都经过执行验证。因此，实用性是本书的又一特点。

在内容安排上，从计算机软件界面设计的实际出发，全书力求概念明晰，语言简练，内容充实。第1、2章是用户界面及C语言的基础；第3章详细讨论有关鼠标器的技术和编程技巧，为后续章节中鼠标器的应用打下基础；第4章是有关图形模式窗口技术的讨论，包括VGA屏幕图形存取及窗口管理等；第5章开始到第9章是常用界面友好技术的讨论，包括菜单、对话框、屏幕帮助、目录列表选择、图符与字段编辑等；第10章是图形化编辑的具体应用；第11章则详细讨论界面技术中的汉字显示问题，为实现汉化软件界面提供参考；最后一章介绍几个C语言编程的技巧。

本书第2、11章的大部分内容由卫跃文执笔，其它章节由潘建华编写。全书由潘建华最后统稿。书中的一些技术得益于和同事们的讨论；单民珩阅读了本书初稿，并提出了很多有益的建议和修改意见；写作过程中航天部五〇四所的有关同志及西安电子科技大学出版社的编辑同志给予了极大的鼓励和支持，在此一并表示感谢。

成书匆匆，错误和缺点在所难免，恳请读者指正。倘若本书能对读者有所帮助和参考的话，作者则甚感欣慰。

目　录

第 1 章　计算机用户界面概述

良好的用户界面(UI)和方便的人机交互是一个成功的软件所必需的。计算机的迅速发展使软件技术发展日新月异。现代软件开发技术中，用户界面的开发占有重要地位。命令行环境的软件界面只能成为历史。

§1.1　界面技术及其发展

在计算机普遍应用的今天，人们的工作乃至日常生活愈来愈借助甚至依赖计算机。在计算机应用中，用户与计算机之间经常进行一系列的信息传递与交换，这种过程称为人机交互。人机交互是通过一定的接口关系进行的，这种用户接口又被称之为界面(interface)。用于信息传递的界面不仅存在于人和计算机之间，而且还存在于人与一切软件之间。总之，用户界面(UI)可以理解为人在使用、操作计算机或软件时，所面对的一切信息源与信息接口的总和。

在计算机的硬件设备中，监视器、键盘、鼠标器、数字化仪、打印机、绘图仪等，都是现代用户界面的重要组成部分，也是用户界面的最基础部分。用户界面另一个重要的方面是软件界面，它提供硬件界面的实现方法，也是建立在硬件界面的基础上的用户界面。它是实现良好用户界面的关键。界面是否友好，除先进的硬件设备之外，优秀的软件实现与控制是至关重要的。只有软件才能实现真正意义的友好人机界面，使界面在外观与感觉上都具有吸引力。操作一个具有良好人机界面的优秀软件，在一定意义上说是一种享受，它使用户在轻松、愉快的环境中从事计算机的应用和开发。

用户界面的发展是随着计算机技术的发展而发展的。事实上，界面技术本身就是计算机技术的组成部分。早期的计算机由于在硬件功能方面尚有较大的不足，计算机研究的重心是计算机硬件性能及硬件界面(接口)，用户界面操作起来复杂而繁琐，效率很低。当计算机硬件技术有了一定程度的发展之后，人机界面变得容易一些。用户可以通过一定的键盘命令及监视器与计算机进行信息传递和交流。在用户界面的友好技术的发展史上，有突破性的是美国 Xerox 公司基于图形监视器及鼠标器等设备的重叠窗口、弹出式菜单技术，该界面一举吸引了计算机研究人员的广泛注意。随后的几年，界面友好软件得到了迅速发展。目前，以 Windows 为代表的窗口、菜单、对话框以及图形模式的其它用户接口方式成为友好界面的主流。图形用户界面(GUI)取得了很大的进展，键盘不再是唯一的信息输入工具，鼠标器、数字化仪等非键盘输入工具得到广泛的应用。与此同时，包括文字、图形、声音、图像等多种传播媒介的多媒体技术在用户界面的发展中也愈来愈受到人们的关注。用户界面将向多元化的方向发展，“使计算机适应人”的推动，必将使界面技术得到飞速的发展。

现代微机系统的典型配置(主机、监视器、鼠标等)为用户界面的发展提供了条件。在这种意义上说，界面技术重点将在软件方面。本书的重点也是基于键盘、鼠标器及监视器

的软件界面开发技术，提供若干软件界面开发技术的参考。

§1.2　用户界面技术的实现

界面包括硬件界面及软件界面两个方面。与之对应，界面技术的实现也有硬件和软件两个方面。硬件支持是界面技术的基础，是软件界面的前提条件，软件界面是界面友好的关键，也是界面友好技术中最引人注意的。

从硬件方面来看，显示器是基本界面设备之一，早期的显示器是单色、低分辨率的，其适配器上显示缓冲区也很小，只能作一般的字符显示。后来出现的 CGA、EGA 及第三代的视频图形适配器 VGA、SVGA 等无论是在色彩、分辨率及显示缓冲区大小等方面都有了很大的提高。监视器性能的提高使基于图形的软件接口成为可能。传统的键盘是另一个重要的硬件界面设备，非键盘的输入设备鼠标器、数字化仪等随着高性能图形显示器的出现，也成为重要的计算机输入设备，其应用日益广泛。类似 Windows 一样，多数优秀软件愈来愈依赖于鼠标器等非键盘输入设备。

从软件方面来看，先进的硬件设备配置只有在优秀软件的管理和控制下其性能才能得到充分的发挥。因此，软件方面的支持是界面友好技术的关键。以软件方式实现的重叠窗口、菜单、鼠标器控制、对话框等技术是实现界面友好的重要手段。

图 1－1 是一个典型的用户软件界面，它是为一个计算机辅助电路分析软件而设计的。从功能上看，它能实现电路图的编辑、电路特性曲线显示、参数输出、文件存取及工作环境设置等，该界面还可提供实时帮助、错误操作警告等，是一个功能较为完善的用户接口。

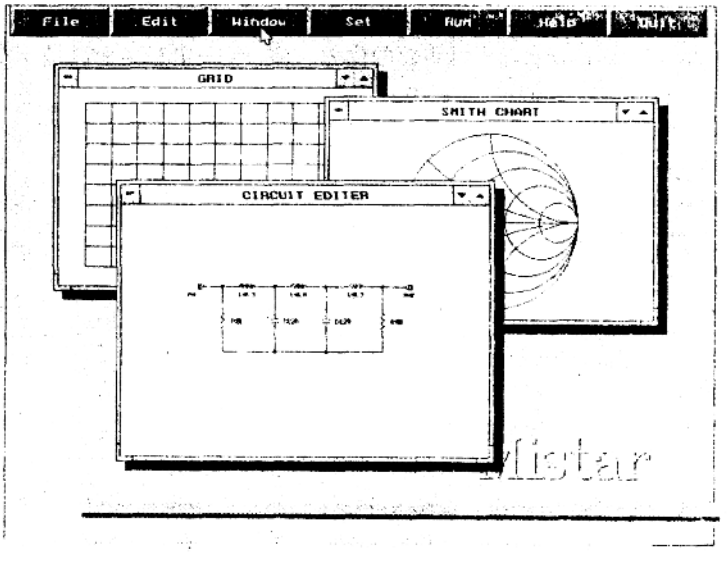

图 1－1　用户软件界面实例

从特点上看，该界面提供完整的基于鼠标器操作的重叠窗口、菜单系统。在鼠标器的控制下，用户可方便地进行菜单选择、窗口移动、窗口放大与缩小以及窗口打开与关闭等。电路编辑功能是以图形编辑的方式实现的。实现电路编辑功能的菜单也很有特点，当需要做出选择时，菜单自动弹出，并具有下拉菜单的功能。对话框及文本处理也是在鼠标器的操作中实现的。本书将以此界面为实例，讨论其界面技术以及其它相关界面友好手段的实现方法和编程技巧。

原则上讲，任何计算机语言都可以实现窗口和菜单等友好界面。基本的计算机语言 BASIC、适合于科学计算的 FORTRAN 等都可以用来实现窗口及菜单等许多界面友好技术，只是其方便程度与功能实现的难易不同而已。功能强大的 C 语言及 C++应该是进行界面技术开发最具吸引力的语言之一。本书主要讨论用 Turbo C 实现界面友好技术。

§1.3　软件界面程序设计风格

写一个计算机程序，尤如构建一座大厦，坚固的基础是至关重要的。程序开发策略、程序设计风格都是一些重要的基础问题。掌握并运用一些基本的技巧，就能构造出适用面广泛的软件产品，以适应实际的需要，并能为以后的维护和修改提供条件。

图文并茂的用户界面并不仅仅是把漂亮的图像放在屏幕上，图形及其构造仅仅是一种手段，界面程序的设计必须有一定的程序开发策略。无论是小型程序界面还是一个庞大系统的运行环境，在正式的程序开发之前都必须有一个管理软件项目的全盘计划。准备工作是最重要的因素之一。首先必须清楚自己究竟是要做什么，要达到的目标是什么，对即将开发的软件有一个清晰的规范；其次，必须了解自己所开发的软件界面是建立在什么样的平台之上的，能够提供怎样的硬件支持，或者期望由哪些硬件设备来支持。另外一点，必须熟悉计划开发的软件界面程序所要用到的软件工具及可以利用的软件资源。在一定的软件资源环境下，开发自己的程序，可以大大提高开发产品的效率。

成功的软件界面程序设计与其它的成功的程序设计原则并没有什么不同。坚持不懈地进行自己的程序开发是成功的要素之一，完美的程序往往需要较长的和较复杂的调试过程。成功的程序设计的第二个要素是结构化程序设计，这个要素即使在面向对象的程序设计技术中，也有着同样重要的意义。成功的程序开发的另一要素是模块突破，任何问题都可以分为若干部分，独立地解决了每一个问题，在正确的协调和组织下，整个问题便可迎刃而解。C 语言的程序设计也是基于这种原理来设计软件界面程序的。在 C 语言中，可以或多或少地将程序分解为若干独立的函数，进行分别调试和检测。这种方法是自顶向下的程序设计方法的基础。

在软件开发的早期，建立一些哑元原型是一种很好的办法。这样，可以将大部分的函数设置为空操作，以检验程序中的关键部分。比如说，要在程序中使用覆盖等特殊的技术，就可以建立一个快速原型来测试计划使用的覆盖联接程序，保证所采取的步骤和方法满足覆盖链接程序的要求，这种手段为以后的程序设计提供了必要的技术保证。用同样的方法还可以检验计划开发的庞大软件中的若干重要技术，在软件开发的每一个阶段，对阶段性成果也有必要进行同样的测试，这是提高软件开发效率的重要手段。

无论是整体框架设计还是模块函数编程，预见性编程思想在程序设计中是非常重要

的。模块函数所能适应的整体框架或者是整体框架所满足的系统要求并非总是一成不变的。一般来说，软件界面所支持的整个软件并非只用在开发它的计算机上，计划设计的软件界面也应该具有适应这种变化的能力。单个模块的设计更是这样。这种特性使得在程序开发的过程中，不得不采取若干手段来对所开发的程序进行全面的检测。比如，对不同图形适配器的检测。一个成功的软件界面绝不能只适用于单一的某种监视设备。图形输入设备也是这样，在开发过程中，应尽可能多地用多个鼠标器驱动程序来验证程序的正确性。

多数时候，经过努力和认真设计成功的软件界面，在一定的时候，总是需要修改或增加新的功能，这是预见性编程的另一重要方面，即代码文档化。显示在屏幕上的程序代码的编写方式与其能否在今后被人轻松地读懂有很大的关系。即使是现在编写的程序，很可能要回头对它修改，因此要保持良好的编程风格，改善代码的外观和提高其阅读的清晰度。代码文档化是一个良好的习惯，在自己的程序中，应尽可能地加上注释以解释它做什么或者说明它如何去做。注释看起来会使源文件臃肿，但它们往往价值极大。除了给源程序加注之外，做一些笔记同样可以理清自己的思路。这种做法会给未来的工作提供极大的帮助。在程序开发中，发展并坚持自己的风格是成功的软件开发中十分重要的一点。

§1.4　优秀的程序设计语言——C

C 语言是当前最流行的程序设计语言，其丰富的数据类型、灵活精炼的语句、高效率、达能力强及可移植性好等优点，倍受程序员的青睐，目前绝大多数程序员在开发应用软，都采用 C 来编程。

语言的产生与 UNIX 系统的发展密切相关，早期由汇编语言编制的 UNIX 系统后来 C 语言改写一遍。虽然在发展初期，C 语言附属于 UNIX 系统，它的产生也主要是为了更好地描述 UNIX 系统，但现在的 C 已独立于 UNIX 系统，是近年来在计算机程序设计中做出了重大贡献并在各类计算机上共同使用的一种语言。

C 语 的基本特点是简洁、灵活、表达能力强、产生代码质量高、可读性强、可移植性好。具体地说，在 C 语言程序中使用一些简单、规整的方法可以构成相当复杂的数据类型、语句和程序结构；C 语言有多种运算符、多种表述问题的途径，并有多种求表达式值的方法，使程序设计者有很大的主动性；C 语言还有与汇编语言很相近的功能和描述方法，可对硬件端口直接操作，充分利用计算机系统资源；C 语言提供了与地址密切相关的指针及有关运算符；C 语言为字符、字符串、集合和表的处理提供了良好的基础，具有预处理程序和预处理语句。一般的 C 系统均可提供方便、功能极强的多种库函数，从而使 C 的优点更为突出。

C 语言是一种结构化、模块化的程序设计语言，完整的 C 软件由一系列的函数组成。函数是 C 程序的基本单位，每个 C 程序有且只有一个主函数，函数定义可以嵌套。C 的函数是功能模块，每个函数可以只完成其一种功能，从而支持传统的自顶向下、逐步求精的程序设计步骤。C 的函数支持递归调用，而且可以采用多个源文件编译、链接，为模块化、结构化设计提供了条件。

鉴于 C 语言强大的功能及灵活方便的特点，本书采用 Turbo C 2.0 来介绍软件界面的程序设计。

第 2 章　C 语言图形程序设计

在软件界面设计中，丰富的色彩与漂亮的图形是必不可少的，这样不仅能使用户感到赏心悦目，而且还能给用户醒目的提示。C 语言具有丰富的图形功能，作为软件界面设计的基础，本章将介绍 Turbo C 的图形函数及在图形方式下的编程技巧。

§2.1　显 示 模 式

利用 Turbo C 进行图形设计，主要是对其图形功能库函数进行调用。Turbo C 提供了包含有 70 多个图形函数的独立图形库，其范围包括从高级调用到面向于位的各种图形函数。这个图形库支持多种填充及线段格式，并提供若干正文字体，可选择字体大小、对齐方式等。这些函数都包含在图形库 GRAPHICS. LIB 里，并嵌入在头文件 GRAPHICS. H 中。除这两个文件外，Turbo C 图形系统还包括图形设备驱动程序和笔划字符字体文件，并都以文件形式存于磁盘，前者扩展名为 BGI，后者扩展名为 CHR。

图形的显示与计算机上所配的图形显示适配器有关，按其显示能力可分为多种显示模式。显示模式决定显示器的分辨率、可同时显示颜色的多少、调色板的设置等。在图形模式下，整个屏幕是由点组成的阵列，每一点在屏幕上产生一个有颜色的光点，这些光点称之为像素。例如，VGA 卡在 VGAHI 模式下，有 640×480 个像素。一个屏幕上像素的个数，称之为分辨率。对于特定屏幕，分辨率越高，像素点就越小。

使用 Turbo C 的任何图形功能之前，都必须把计算机设置为正确的图形模式。在未设置的情况下，绝大多数计算机系统都使用 80 列字符模式。由于这不是图形模式，图形函数不能工作。Turbo C 将屏幕设置为图形模式，有一个专门的函数：

```
void far initgraph(int *gdriver,int *gmode,char *path)
```

该函数初始化图形系统，并把硬件设置为指定的图形模式。参数 gdriver 指定图形设备卡，gmode 指定图形模式，path 为指定寻找图形设备驱动程序的路径，如果没有指定，则在当前目录下寻找。

如果已经知道所使用的图形适配器的类型和要使用的图形模式，那么初始化就很简单：

```
int gdriver, gmode
gdriver = VGA;                    /* 显示器类型为 VGA */
gmode = VGAHI;                    /* 选用高分辨率模式 */
initgraph(&gdriver,&gmode,"");    /* 初始化图形系统 */
...
```

这样编制的程序只能适用于 VGA 图形适配器，不具有通用性。其实 Turbo C 中有一个对图形显示硬件测试的函数：

```
void far detectgraph(int *gdriver,int *gmode);
```

该函数检查硬件以决定使用何种图形驱动程序，并推荐一种模式。其中 gdriver 的意义与 initgraph()中的意义一样。调用该函数后，有关图形显示器和图形模式的状况就返回给 gdriver 与 gmode。

```
int gdriver,gmode;
detectgraph(&gdriver,&gmode);
if(gdriver<0){
    printf("\n There is no graphics driver");
    exit(1);
    }
initgraph(&gdriver,&gmode," ");
...
```

更简单的初始化图形系统的方法是将 DETECT 赋给 gdriver，再使用 initgraph()就可以了。当 gdriver 取 DETECT 时，系统自动调用 detectgraph()函数检测设备。

```
int gdriver=DETECT,gmode;
initgraph(&gdriver,&gmode," ");
...
```

关闭图形模式，可用 closegraph()或 restorecrtmode()，它们的原型分别为：

```
void far closegraph();
void far restorecrtmode();
```

如果程序只在文本模式下执行，可使用 closegraph()函数。该函数释放所有内存，复位视频模式，使之为调用 initgraph()之前的状态；如果程序要在文本和图形模式之间进行转换，用 restorecrtmode()，它使视频适配器复位为第一次调用 initgraph()之前的模式，但并不释放内存；如果程序到了最后，那么使用哪个函数都可以。

下面这段程序说明如何暂时离开图形模式进入文本模式，进行文本输出，然后不经过 initgraph()再返回图形模式的方法。

C 语言图形模式设置示例源程序

```
#include <stdio.h>
#include <graphics.h>
main()
{
    int gdriver=DETECT,gmode;
    initgraph(&gdriver,&gmode," ");      /* 初始化图形模式 */
    circle(200,100,50);
    getch();
    gmode=getgraphmode();                /* 获得现行模式 */
    restorecrtmode();                    /* 返回初始化前的模式 */
    printf("\nNow is text mode,press any key return graphic mode...");
    getch();
    setgraphmode(gmode);                 /* 返回图形模式 */
    rectangle(100,50,200,200);
    getch();
```

```
    closegraph();
}
```

本书以后章节中，除非特别说明，都是针对 VGA 的 640×480 高分辨率模式来进行程序设计的。

§2.2　常用绘图函数

Turbo C 提供了许多绘图函数，用来画点、线、圆、椭圆、长方形以及任意多边形等，可以设置画线的颜色、线型及粗细，还可以对一个有界区域进行填充等等，利用这些功能可以方便地绘制各种图案。

1. 坐标定义

在图形模式下，以屏幕左上角为坐标原点(0,0)，水平方向向右为 x 轴正方向，垂直方向向下为 y 轴正方向。

例如，对于 640×480 的图形模式，x、y 的取值范围分别为 0 到 639、0 到 479。(0,0)点在屏幕左上角，(639,479)在屏幕的右下角。如图 2—1 所示。

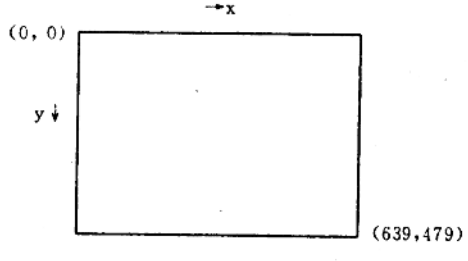

图 2-1　VGA 高分辨率屏幕坐标

相同显示器在不同的显示模式下，x、y 坐标的最大值是不相同的，可用下列函数获得这些信息。

- int far getmaxx(void)；返回 x 的最大坐标值
- int far getmaxy(void)；返回 y 的最大坐标值

在图形模式下，光标是不显示的，但用下列两个函数可以获得当前光标的位置。

- int far getx(void)；返回光标的 x 坐标
- int far gety(void)；返回光标的 y 坐标

还可通过以下两个函数来改变光标的位置。

- void far moveto(int x,int y)；将光标移动至(x,y)点
- void far moverel(int dx,int dy)；从当前位置，将光标相对移动到 dx、dy

2. 绘图函数

Turbo C 常用的绘图函数有：

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

在(x,y)点画一个由 color 指定颜色的点。

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

在(x1,y1)、(x2,y2)之间以当前颜色和线型画一条直线。

• void circle(int x,int y,int radius);

以(x,y)为圆心，redius 为半径，用当前颜色和线型画一个圆。

• void far arc(int x,int y,int angle1,int angle2,int radius);

以(x,y)为圆心，radius 为半径，angle1 为起始角，angle2 为终止角，用当前颜色和线型画一圆弧。

• void far ellipse(int x,int y, int angle1,int angle2,int xradius,int yradius);

以(x,y)为中心，angle1、angle2 分别为起始角和终止角，xradius、yradius 分别为 x、y 方向的半径，用当前颜色和线型画一椭圆。

• void rectabgle(int x1,int y1,int x2,int y2);

以(x1,y1)、(x2,y2)为两对角，用当前颜色、线型画矩形。

3. 线型

在以上介绍的几个绘图函数中，画线所用线的特性都默认是一个像素宽的实线。Turbo C 提供了一个函数可以用来改变画线的特性：线宽和线型。线型共有五种，如表 2－1 所示；而线宽只有两种：一个像素宽和三个像素宽，如表 2－2 所示。

表 2－1　线的类型

名　称	值	含　义
SOLID_LINE	0	实线
DOTTED_LINE	1	点画线
CENTER_LINE	2	中心线
DASHED_LINE	3	虚线
USERBIT_LINE	4	自定义线型

表 2－2　线　宽

名　称	值	含　义
NORMAL_WIDTH	1	一个像素宽
THICK_WIDTH	3	三个像素宽

在软件界面设计中适当使用不同的线型和线宽，不仅可使软件界面更加丰富多彩，而且还可使某些选项醒目，以达到提示用户或警告用户的作用。

画线类型和线宽可用 setlinestyle()函数来设置，其格式为：

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

其中，linestyle 说明以何种线型来画以后的线，其有效值见表 2－1；upattern 是一个仅当 linestyle 是 USERBIT_LINE 才起作用的 16 位模式，该参数在哪种情况下都是必需的，但当 linestyle 为其它数值时，它不起作用。thickness 表示以后所画线的线宽，其有效值见表 2－2。

下面通过一个例子，说明基本绘图函数及线型设置函数的使用。

C 语言图形函数应用示例源程序

```
#include <stdlib.h>
#include <math.h>
#include <graphics.h>
main()
{
    int gdriver=DETECT,gmode;
    int maxx,maxy;
    int polypoint[12]={30,50,85,64,200,35,65,176,290,76,30,50};

    initgraph(&gdriver,&gmode,"");              /* 初始化图形系统 */
    setbkcolor(BLACK);                          /* 设置背景颜色 */
    setcolor(BLUE);                             /* 设置画笔颜色 */
    cleardevice();                              /* 清屏 */
    maxx=getmaxx();                             /* 获取 x,y 最大值 */
    maxy=getmaxy();
    setlinestyle(SOLID_LINE,0,THICK_WIDTH);     /* 设置线型为粗实线 */
    rectangle(0,0,maxx,maxy);                   /* 以最大边界画一矩形 */
    line(0,maxy-50,maxx,maxy-50);               /* 再画一直线 */
    getch();
    setlinestyle(DASHED_LINE,0,NORM_WIDTH);     /* 设置线型为细虚线 */
    setcolor(RED);                              /* 设置画笔颜色 */
    circle(maxx/2,maxy/2,150);                  /* 在屏幕中间画圆 */
    getch();
    arc(maxx-100,100,0,120,50);                 /* 再画一圆弧 */
    getch();
    setcolor(GREEN);
    ellipse(maxx/2,maxy/2,0,360,100,50);        /* 画一椭圆 */
    getch();
    rectangle(maxx/2-100,maxy/2+50,maxx/2+100,maxy/2-50);
                                                /* 画椭圆的外接矩形 */
    getch();
    setlinestyle(CENTER_LINE,0,NORM_WIDTH);
    setcolor(RED);
    moveto(maxx/2-100-10,maxy/2);               /* 移动光标 */
    linerel(220,0);                             /* 从当前光标处画线 */
    moveto(maxx/2,maxy/2+50+10);
    linerel(0,-120);
    drawpoly(6,polypoint);                      /* 画一多边形 */
    getch();
    closegraph();                               /* 关闭图形系统 */
}
```

另外，函数 setwritemode()用于设置画线的输出模式。其格式为：

void far setwritemode(int mode);

如果 mode=0，画线时采用重画方式，即所画之处原来的信息将被重写；如果 mode=1，画线时，新线像素点与旧线像素点之间先进行异或操作，然后再向屏幕输出。利用这一特性可用来擦除某些线条。

4. 填充

填充就是在一个封闭的区域内填满某种颜色或某种图案。Turbo C 共有五个填充函数。

• void far bar(int x1,int y1,int x2,int y2);

以(x1,y1)、(x2,y2)为一对对角点，画一矩形，再以当前填充颜色和图案进行填充。

• void far bar3d(int x1,int y1,int x2,int y2,int depth,int topflag);

以(x1,y1)、(x2,y2)为一对对角点画一矩形，depth 给出第三维的深度，topflag 为 0 画出三维的顶部，非 0 则不画三维的顶部。

• void far floodfill(int x,int y,int border);

以(x,y)为参考点，填充一块由 border(颜色值)指定的封闭区域。当参考点在封闭区域内时，则填充该区域；当参考点在封闭区域外时，则填充除了封闭区域外所有的区域。

此外还有扇形和椭圆扇形填充。

向封闭区域内填充，默认以当前颜色填充。Turbo C 提供了 13 种填充方式，如表 2—3 所示。

表 2－3　填充方式

名　称	值	含　义
EMPTY_FILL	0	用背景色填充
SOLID_FILL	1	全部单色填充
LINE_FILL	2	用水平线填充
LTSLASH_FILL	3	用较细左斜线填充
SLASH_FILL	4	用较粗左斜线填充
BKSLASH_FILL	5	用较粗右斜线填充
LTBKSLASH_FILL	6	用较细右斜线填充
HATCH_FILL	7	用网格填充
XHATCH_FILL	8	用粗交叉网格填充
INTERLEAVE_FILL	9	用交叉线填充
WIDE_DOT_FILL	10	用稀空心点填充
CLOSE_DOT_FILL	11	用密空心点填充
USER_FILL	12	用户自定义填充方式

填充方式和颜色可用 setfillstyle()来设置，其格式为：

void setfillstyle(int pattern,int color);

其中，color 为当前填充颜色；pattern 为填充方式，可取表 2—3 中的任何一种。如果 pattern

为 USER_FILL，则使用用户自定义的方式填充，这时需调用 setfillpattern() 函数来定义填充图案，其格式为：

void far setfillpattern(char far *upattern,int color);

其中，upattern 指向一个 8 字节的存贮区，在这个存贮区中定义一个 8×8 点阵的填充图案；color 为当前填充颜色。

下面通过绘制一面红旗的图案，来说明填充函数的使用方法。

C 语言图形填充示例

```
#include <graphics.h>
#include <math.h>
main()
{
        int gdriver=DETECT,gmode;
        int x,y,i;
        initgraph(&gdriver,&gmode,"");          /* 初始化图形系统 */
        setbkcolor(BLUE);
        setfillstyle(SOLID_FILL,RED);           /* 设置实心红色填充 */
        bar(15,10,250,150);                     /* 画一红旗 */
        setfillstyle(SOLID_FILL,BROWN);
        setcolor(LIGHTGRAY);
        bar3d(9,6,15,getmaxy(),0,0);            /* 画旗杆 */
        pieslice(12,7,0,180,5);
        setcolor(YELLOW);
        setfillstyle(SOLID_FILL,YELLOW);
        fivestar(59,35,30);                     /* 画一五角星并填充 */
        circle(74,40,16);                       /* 画一圆 */
        getch();
        closegraph();                           /* 关闭图形系统 */
}

int fivestar(int x,int y,int a)
{
        moveto(x,y);                            /* 画一五角形 */
        lineto(x+0.38*a,y);
        lineto(x+0.5*a,y-0.36*a);
        lineto(x+0.61*a,y);
        lineto(x+a,y);
        lineto(x+0.70*a,y+0.22*a);
        lineto(x+0.81*a,y+0.59*a);
        lineto(x+0.50*a,y+0.36*a);
        lineto(x+0.19*a,y+0.59*a);
        lineto(x+0.30*a,y+0.22*a);
        lineto(x,y);
```