

**Foshan University**

**课程设计**

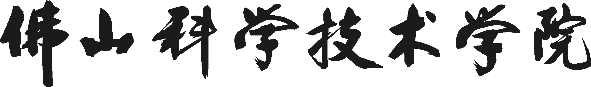
**学 院：** 电子信息工程学院

**专 业：** 16网络工程3

**学 号：** 20160390332

**学生姓名：** 蒙信亨

**2019 年 1 月**



课程设计任务书

网络工程 专业 16网络工程 级 3 班 蒙信亨

1. **设计题目：操作系统图形显示程序接口设计**
2. **主要内容：**

**本设计的目的是通过设计一些简单的中断调用接口，使学生掌握程序接口的设计方法。**

**要求在熟悉DOS操作系统的中断接口及程序接口的基础上，利用C语言设计简单的显示程序接口，最后通过程序验证全部接口的正确性。**

1. **具体要求**

利用C语言设计DOS中断中21H与图形操作相关的中断调用，具体包括：

1. 设置点阵图形的彩色
2. 画点
3. 画线
4. 画矩形
5. 画圆
6. 屏幕区域填充

7、屏幕区域保存

8、屏幕区域逻辑恢复

9、清除屏幕/区域

10、屏幕区域卷动

11、鼠标控制

12、多窗口功能

上述功能由两部分程序验证，中断驻留程序和验证程序。首先运行中断驻留程序，然后运行验证程序得到预期结果。

1. **进度安排**

1-2天：资料查找、系统分析，数据流程分析，概要设计

**1-2天：系统详细设计、功能设计**

**5-6天：模块设计、编程调试**

**1天：资料整理、课程设计说明书编写。**

**五、完成后应上交的材料**

**1、课程设计说明书（程序流程图、功能模块图、相关数据结构、核心算法等）**

**2、相关源程序文件**

**六、总评成绩**

**指导教师 签名日期 年 月 日**

**系 主 任 审核日期 年 月 日**

目 录

一、课程设计概述 ……………………………………………………………5

1.1 内容 …………………………………………………………………5

1.2 要求 …………………………………………………………………5

二、相关理论 …………………………………………………………………6

三、程序设计 …………………………………………………………………7

3.1 程序流程图 …………………………………………………………7

四、 调用函数…………………………………………………………………8

五、设计结果 …………………………………………………………………8

六、设计总结 …………………………………………………………………9

附 录……………………………………………………………………………10

1. 课程设计概述

1.1内容

题目：操作系统图形显示程序接口设计

主要内容：

本设计的目的是通过设计一些简单的中断调用接口，使学生掌握程序接口的设计方法。

要求在熟悉DOS操作系统的中断接口及程序接口的基础上，利用C语言设计简单的显示程序接口，最后通过程序验证全部接口的正确性。

1.2 要求

利用C语言设计DOS中断中21H与图形操作相关的中断调用，具体包括：

1.设置点阵图形的彩色

2.画点

3、画线

4、画矩形

5、画圆

6、屏幕区域填充

7、屏幕区域保存

8、屏幕区域逻辑恢复

9、清除屏幕/区域

10、屏幕区域卷动

11、鼠标控制

12、多窗口功能

上述功能由两部分程序验证，中断驻留程序和验证程序。首先运行中断驻留程序，然后运行验证程序得到预期结果。

1. 相关理论
2. 中断意思就是指计算机运行过程中，出现某些意外情况需主机干预时，机器能自动停止正在运行的程序并转入处理新情况的程序，处理完毕后又返回原被暂停的程序继续运行。中断是一种计算机术语。通俗点说就是停下当前的执行程序转而执行新的程序。

2、Windows提供了一个称为图形设备接口GDI（Graphics Device Interface）的抽象接口。GDI作为Windows的重要组成部分，它负责管理用户绘图操作时功能的转换。用户通过调用GDI函数与设备打交道，GDI通过不同设备提供的驱动程序将绘图语句转换为对应的绘图指令，  
避免了直接对硬件进行操作，从而实现所谓的设备无关性。编程时采用MFC方法绘制图形也很方便，MFC对GDI函数和绘图对象进行了封装

3.中断驻留程序，指的是当加载进内存的程序执行完毕后，依然能够保存产

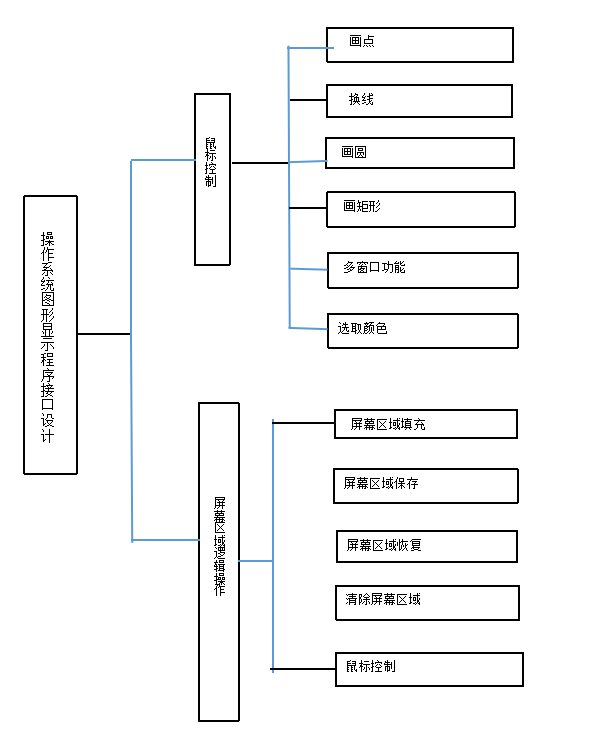
生的临时数据跟临时状态，而在下一次调用时继续执行

4、系统调用的实现原理：计算机系统的各种硬件资源是有限的，在现代多任务操作系统上同时运行的多个进程都需要访问这些资源，为了更好的管理这些资源进程是不允许直接操作的，所有对这些资源的访问都必须有操作系统控制。也就是说操作系统是使用这些资源的唯一入口，而这个入口就是操作系统提供的系统调用（System Call）。系统调用是属于操作系统内核的一部分的，必须以某种方式提供给进程让它们去调用。CPU可以在不同的特权级别下运行，而相应的操作系统也有不同的运行级别，用户态和内核态。运行在内核态的进程可以毫无限制的访问各种资源，而在用户态下的用户进程的各种操作都有着限制，比如不能随意的访问内存、不能开闭中断以及切换运行的特权级别。显然，属于内核的系统调用一定是运行在内核态下，但是如何切换到内核态呢？

答案是中断。操作系统一般是通过中断从用户态切换到内核态。中断就是一个硬件或软件请求，要求CPU暂停当前的工作，去处理更重要的事情。比如，在x86机器上可以通过int指令进行软件中断，而在磁盘完成读写操作后会向CPU发起硬件中断。

三、程序设计

3.1、程序设计流程图



3.2 操作方法

1. 调用函数

initgraph( ); 初始化绘图窗口，创建绘图环境

setfillcolor(WHITE);填充颜色

GetCursorPos(&point); 获取鼠标指针位置（屏幕坐标）

ScreenToClient(hwnd, &point); 将屏幕鼠标指针位置转换为相对当前窗口坐标

GetAsyncKeyState(VK\_LBUTTON))是一个用来判断函数调用时指定虚拟键的状态，确定用户当前是否按下了键盘上的一个键的函数。这里的键指鼠标左键

setcolor(getpixel(point.x,point.y));设置颜色(检索到的指定坐标点的像素的RGB颜色值)

cleardevice();清除图形屏幕

saveimage( );保存图片函数

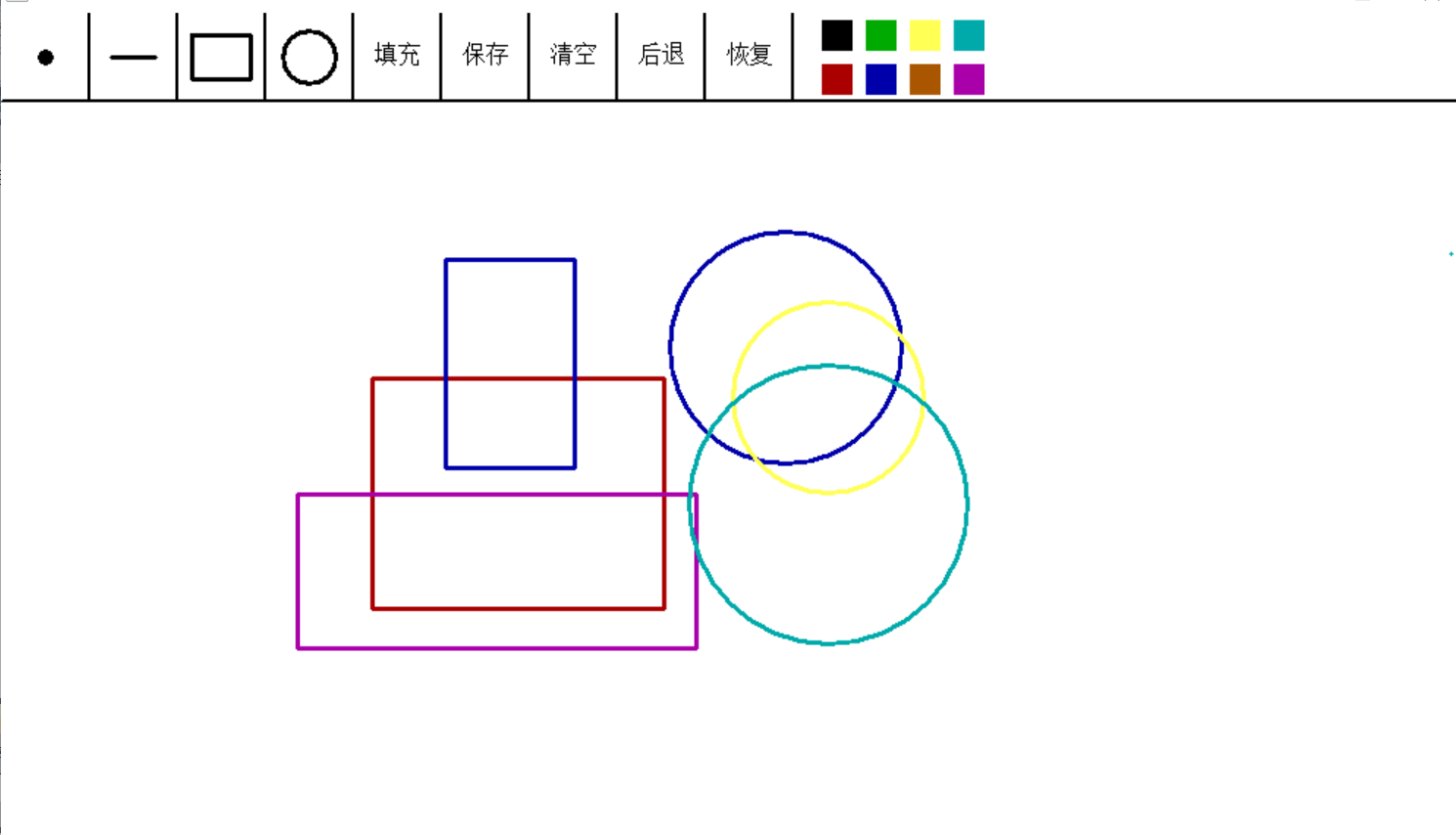
MessageBox( ) ; 消息框函数

bar( );使用对角线坐标画矩形

line( );画直线

circle( );画圆

五、设计结果



5.1、图形绘制界面如上所示，上面工具栏功能分别为：

1.画点

2.画直线

3.画矩形

4.画圆

5.颜色填充

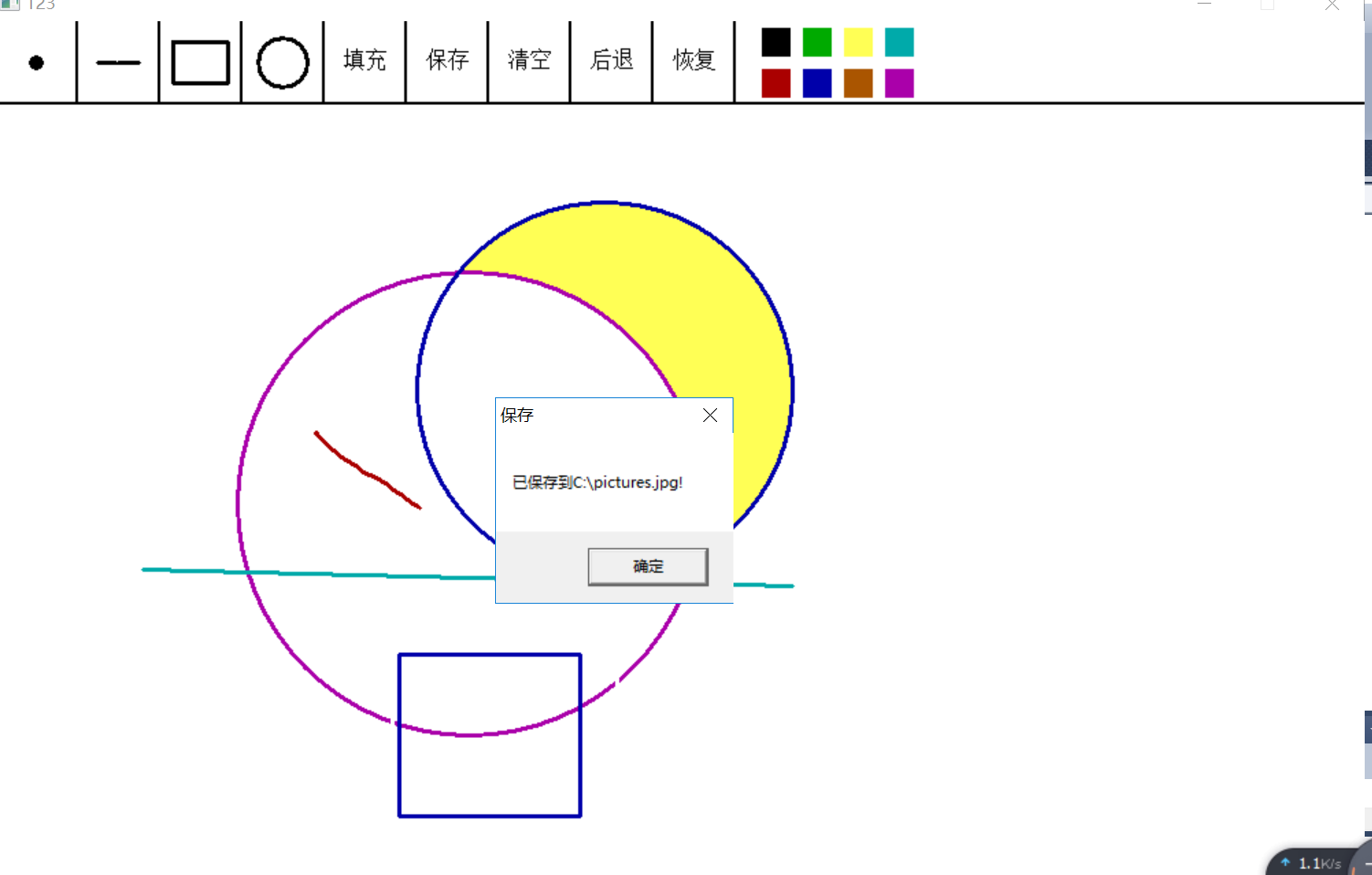
6.图片保存

7.清空图片

8.撤销/后退

9.载入已保存的图片

10.选取颜色



六、设计总结

通过两周的钻研顺利完成了课程设计，过程中遇到了一些困难，但经过和同学的交流讨论，还有自己查阅资料，多尝试，逐步克服困难。本次课程设计让我对C语言有了更深刻的认识，C语言的功能可谓强大，能实现多种功能，很多时候想要实现的功能只要查阅并调用相关的函数就可以了，迅速便捷，大大提高了课程设计的效率。对C语言的学习还会继续深入。此外，还运用到了数据结构和操作系统的知识，所以完成此次课程设计对我们有综合性的提高。设计过程中也多次遇到了一些生涩难懂的概念，比如句柄，中断程序等等，但经过自己查找资料，有了一定的了解，通过课程设计让我接触到了新知识，扩充了自己的知识储备。还有查阅资料，思考很重要，在设计填充功能时。由于对填充函数 floodfill（）不是很了解，所以填充颜色的代码写了好多条，显得累赘。后面经过查阅资料，使用了getpixel（）函数，只要一条填充颜色代码floodfill(point.x,point.y,getpixel(point.x,point.y),1)就可以实现填充功能。当然课程设计的结果还能进一步完善，好多功能还能继续添加，程序仍有待改进。通过两周的课程设计学习，让我对操作系统功能有了进一步的认识，也提高了自己的编程能力，在今后的学习和工作我还会继续努力学习相关知识，提升自己的知识技能。

附录

#include<ctype.h>

#include <graphics.h>//图形模式的初始化

#include <stdio.h>

#include<math.h>

void main()

{

void show();

IMAGE img;//创建空白画纸

initgraph(1000, 680); // 初始化绘图窗口以下局部代码创建一个尺寸为1000x680 的绘图环境：

HWND hwnd = GetHWnd(); // 获取绘图窗口句柄

POINT point;// point 是POINT 定义的结构体

setfillcolor(WHITE);//填充颜色

show();

int x1,y1,a=0,p,color;

char s[]="保存";

char seban[]=" ";

char t[]="填充";

char c[]="清空";

char b[]="后退";

char r[]="恢复";

outtextxy(255,20,t);//填充面板位置

outtextxy(10,635,seban);//调色板位置

outtextxy(315,20,s);//保存面板位置

outtextxy(375,20,c);// 清空面板位置

outtextxy(435,20,b);//后退面板位置

outtextxy(495,20,r);//恢复面板位置

while(true)

{

GetCursorPos(&point); // 获取鼠标指针位置（屏幕坐标）

ScreenToClient(hwnd, &point); // 将屏幕鼠标指针位置转换为相对当前窗口坐标

if(GetAsyncKeyState(VK\_LBUTTON))// 是一个用来判断函数调用时指定虚拟键的状态，确定用户当前是否按下了键盘上的一个键的函数。这里的键指鼠标左键

{

if(point.y<60)

{

setcolor(getpixel(point.x,point.y));//设置颜色(检索到的指定坐标点的像素的RGB颜色值)

color=getpixel(point.x,point.y);}

if(point.y<60)

{

if(point.x>0&&point.x<=60)// 曲线

a=0;

if(point.x>60&&point.x<=120)// 直线

a=1;

if(point.x>120&&point.x<=180)// 矩形

a=2;

if(point.x>180&&point.x<=240)// 圆

a=3;

if(point.x>240&&point.x<=300)//填充

a=4;

if(point.x>360&&point.x<=420)// 刷新屏幕，重新输出所有

{

getimage(&img,0,60,1000,620); //可以用来保存左上角与右下角所定义的屏幕上像素图形到指定的内存区域

cleardevice();//清除图形屏幕

show();

outtextxy(375,20,c);

outtextxy(315,20,s);

outtextxy(255,20,t);

outtextxy(10,635,seban);

outtextxy(435,20,b);

outtextxy(495,20,r);

}

if(point.x>300&&point.x<=360)// 保存

{

getimage(&img,0,60,1000,620);

saveimage("E:\\pictures.jpg");//保存图片

HWND hwnd = GetHWnd();//获取窗口句柄

MessageBox(hwnd, \_T("已保存到E:\\pictures.jpg!"),\_T("保存"), MB\_OK) ; //消息框（对话框父窗口句柄，欲显示的信息字符串，对话框标题字符串，确定按钮）

a=5;

}

if(point.x>420&&point.x<=480)// 后退

a=5;

if(point.x>480&&point.x<=560)// 恢复到文件内容

a=6;

}

while(1)

{

x1=point.x;

y1=point.y;

if(a==0) // 曲线

{

getimage(&img,0,60,1000,620);

GetCursorPos(&point); // 获取鼠标指针位置（屏幕坐标）

ScreenToClient(hwnd, &point); // 将鼠标指针位置转换为窗口坐标

line(x1,y1,point.x,point.y);

if(!GetAsyncKeyState(VK\_LBUTTON))

break;

}

if(a==1)// 直线

{

if(!GetAsyncKeyState(VK\_LBUTTON))

{

getimage(&img,0,60,1000,620);

GetCursorPos(&point); // 获取鼠标指针位置（屏幕坐标）

ScreenToClient(hwnd, &point); // 将鼠标指针位置转换为窗口坐标

line(x1,y1,point.x,point.y);

break;

}

}

if(a==2)// 矩形

{

if(!GetAsyncKeyState(VK\_LBUTTON))

{

getimage(&img,0,60,1000,620);

GetCursorPos(&point); // 获取鼠标指针位置（屏幕坐标）

ScreenToClient(hwnd, &point); // 将鼠标指针位置转换为窗口坐标

line(x1,y1,point.x,y1);

line(x1,y1,x1,point.y);

line(x1,point.y,point.x,point.y);

line(point.x,y1,point.x,point.y);

break;

}

}

if(a==3)// 圆

{

if(!GetAsyncKeyState(VK\_LBUTTON))

{

getimage(&img,0,60,1000,620);

GetCursorPos(&point); // 获取鼠标指针位置（屏幕坐标）

ScreenToClient(hwnd, &point); // 将鼠标指针位置转换为窗口坐标

far circle(x1, y1,(int)sqrt((point.x-x1)\*(point.x-x1)+(point.y-y1)\*(point.y-y1)));

break;

}

}

if(a==4)//full

{

if(!GetAsyncKeyState(VK\_LBUTTON))

{

getimage(&img,0,60,1000,620);

GetCursorPos(&point); // 获取鼠标指针位置（屏幕坐标）

ScreenToClient(hwnd, &point); // 将鼠标指针位置转换为窗口坐标

setfillcolor(color);

setfillstyle(BS\_SOLID); //设置填充模式和颜色

floodfill(point.x,point.y, getpixel(point.x,point.y),1);// 用指定颜色填充一个密闭区域

break;

}

}

if(a==5)// 后退

{

putimage(0, 60, &img);

break;

}

if(a==6)// 文件恢复

{

loadimage(&img, "E:\\pictures.jpg");

putimage(0,0, &img);

break;

}

}

}

}

}

void show()

{

far floodfill(1,1,BLUE);

setbkmode(TRANSPARENT);//背景模式。透明

setcolor(BLACK);

setfillcolor(BLACK);

setlinestyle(0,2);

line(0,60,1000,60);//边线

line(60,0,60,60);

line(120,0,120,60);

line(180,0,180,60);

line(240,0,240,60);

line(300,0,300,60);

line(360,0,360,60);

line(420,0,420,60);

line(480,0,480,60);

line(540,0,540,60);

setlinestyle(0,3);//按钮

fillcircle(30,30,4);

line(75,30,105,30);//直线

rectangle(130,15,170,45);//矩形

circle(210,30,18);//圆

setcolor(BLACK);

setfillcolor(BLACK);

setfillstyle(BS\_SOLID);

bar(560,5,580,25);//使用对角线坐标画矩形

setcolor(RED);

setfillcolor(RED);

setfillstyle(BS\_SOLID);

bar(560,35,580,55);//使用对角线坐标画矩形

setcolor(GREEN);

setfillcolor(GREEN);

setfillstyle(BS\_SOLID);

bar(590,5,610,25);//使用对角线坐标画矩形

setcolor(BLUE);

setfillcolor(BLUE);

setfillstyle(BS\_SOLID);

bar(590,35,610,55);//使用对角线坐标画矩形

setcolor(YELLOW);

setfillcolor(YELLOW);

setfillstyle(BS\_SOLID);

bar(620,5,640,25);//使用对角线坐标画矩形

setcolor(BROWN);

setfillcolor(BROWN);

setfillstyle(BS\_SOLID);

bar(620,35,640,55);//使用对角线坐标画矩形

setcolor(CYAN);

setfillcolor(CYAN);

setfillstyle(BS\_SOLID);

bar(650,5,670,25);//使用对角线坐标画矩形

setcolor(MAGENTA);

setfillcolor(MAGENTA);

setfillstyle(BS\_SOLID);

bar(650,35,670,55);//调色板输出完成

setcolor(BLACK);

setfillcolor(WHITE);

}