微机接口报告
汇编语言图形绘制

目录

题目要求：

微机接口图形绘制，实现下面的目标

1、将 screen 设置为图形显示模式；
2、程序开始提示绘制：直线、矩形还是三角形；
3、根据 2 的选择，进一步通过键盘输入直线、矩形或三角形的参数（即直线端点或▲、■的各个顶点坐标）；
4、步骤 3 要求的图形绘制结束以后，再进入第 2 步。

提示：

1、以程序的规模，需要进行**模块化**的设计，即首先写好直线绘制的函数，而矩形和三角形分别由 4 条和 3 条直线构成；
2、直线的实现分别用直接写屏技术和中断实现。
3、该题目的实现涉及到循环、中断调用、函数的编写和调用、显示模式的设置等知识点。

第一部分　解决方案概述思路

根据题目的要求，根据题目的要求，可以将该程序分成四部分来实现。分别是：显示提示信息、画直线、画矩形、画三角形。

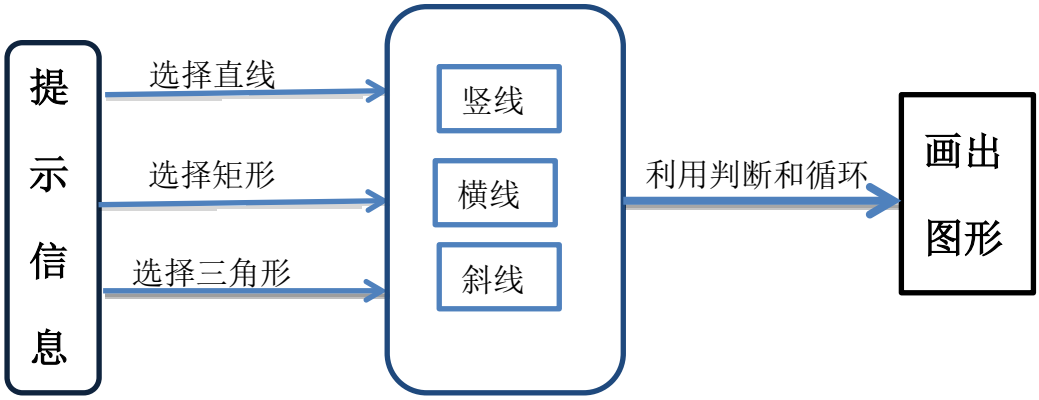
显示提示信息部分需要应用的知识主要有：

视屏模式的设置，直接写屏技术，中断，调用库函数判断输入值。

画直线、画矩形、画三角形部分需要应用的知识主要有：

将屏幕设置为图形显示模式，中断，循环和判断，布雷森汉姆直线算法，模块化和函数的嵌套

流程图如下：



第二部分　所需算法和例程的学习

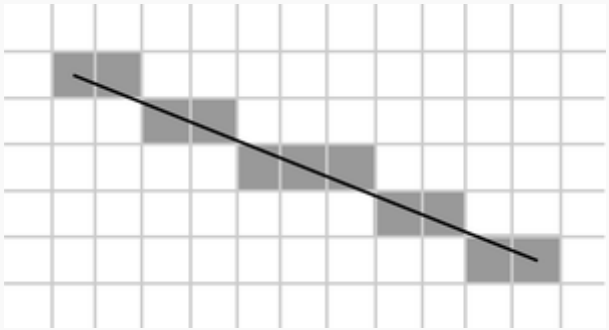
一、Bresenham 直线算法的学习

● Bresenham 直线算法描绘直线

在图形模式下画线需要用到的是布雷森汉姆直线算法。通过维基百科查阅资料了解到布雷森汉姆直线算法的基本知识。

Bresenham 直线算法是用来描绘由两点所决定的直线的算法，它会算出一条线段在 n 维光栅上最接近的点。这个算法只会用到较为快速的整数加法、减法和位元移位，常用于绘制电脑画面中的直线。是计算机图形学中最先发展出来的算法。经过少量的延伸之后，原本用来画直线的算法也可用来画圆。且同样可用较简单的算术运算来完成，避免了计算二次方程式或三角函数，或递归地分解为较简单的步骤。

以上特性使其仍是一种重要的算法，并且用在绘图仪、绘图卡中的绘图芯片，以及各种图形程式库。这个算法非常的精简，使它被实作于各种装置的固件，以及绘图芯片的硬件之中。



Bresenham 直线算法描绘的直线

假设我们需要由 (x_0,y_0) 这一点，绘画一直线至右下角的另一点(x_1,y_1)，x,y 分别代表其水平及垂直座标，并且 x_1-x_0>y_1-y_0。在此我们使用电脑系统常用的座标系，即 x 座标值沿 x 轴向右增长，y 座标值沿 y 轴向下增长。

因此 x 及 y 之值分别向右及向下增加，而两点之水平距离为$x_1 - x_0$且垂直距离为 y_1-y_0 由此得之，该线的斜率必定介乎于 1 至 0 之间。而此算法之目的，就是找出在x_0与x_1之间，第 x 行相对应的第 y 列，从而得出一像素点，使得该像素点的位置最接近原本的线。

对于由(x_0,y_0)及(x_1,y_1)两点所组成之直线，公式如下：

$$y - y_0 = \frac{y_1 - y_0}{x_1 - x_0}(x - x_0)$$

因此，对于每一点的 x，其 y 的值是 $\frac{y_1 - y_0}{x_1 - x_0}(x - x_0) + y_0$

因为 x 及 y 皆为整数，但并非每一点 x 所对应的 y 皆为整数，故此没有必要去计算每一点 x 所对应之 y 值。反之由于此线之斜率介乎于 1 至 0 之间，故此我

们只需要找出当 x 到达那一个数值时，会使 y 上升 1，若 x 尚未到此值，则 y 不

变。至于如何找出相关的 x 值，则需依靠斜率。斜率之计算方法为 $m = (y_1 - y_0)/(x_1 - x_0)$。由于此值不变，故可于运算前预先计算，减少运算次数。

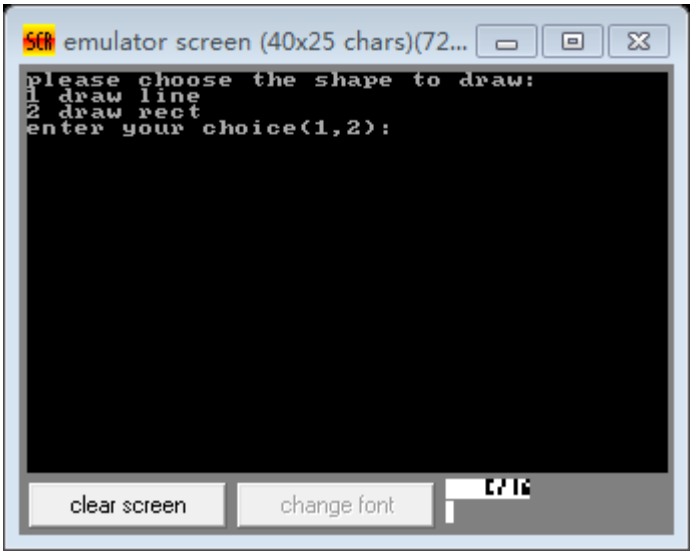
一、信息显示和调用库函数例程的学习

老师所提供的例程主要实现的功能是显示提示信息，读取输入的选择，判断后执行相应的程序。

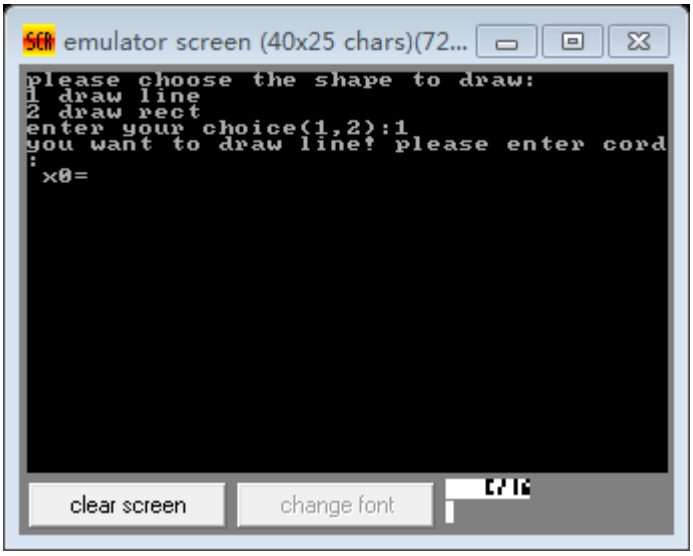
```
        include emu8086.inc
    org 100h
    lableShowTip:
        call ShowTip;
        call scan_num;
        mov ax,cx;

        cmp ax,1
        je lableDrawLine;
        cmp ax,2
       je lableDrawRect;

        print 0ah,0dh
print "please enter your choice again";
        print 0ah,0dh
        jmp lableShowTip;

lableDrawLine:
        call DrawLine;
        print 0ah,0dh
        jmp lableShowTip;
lableDrawRect:
        call DrawRect;
        print 0ah,0dh
        jmp lableShowTip;
    ret
DrawLine proc
    print 0ah,0dh
   print "you want to draw line! please
enter cord:"
     print 0ah,0dh
     print "x0="
        call scan_num
        mov [bx+si+1],cx

        print 0ah,0dh
        print "y0="
        call scan_num
        mov [bx+si+2],cx

        print 0ah,0dh
        print "x1="
        call scan_num
        mov [bx+si+3],cx

        print 0ah,0dh
        print "y1="
        call scan_num
        mov [bx+si+4],cx
        sub cx,[bx+si+2]
        ret
DrawLine endp
DrawRect proc
    print 0ah,0dh
    print    "you want to draw rect!"
    print 0ah,0dh
 print "please enter cord(x0,y0,x1,y1):"
        call get_string
         ret
    DrawRect endp
    ShowTip proc
        mov al, 13h
        mov ah, 0
        int 10h

      jmp ShowTipBeg
      tip0 db "please choose the shape
```

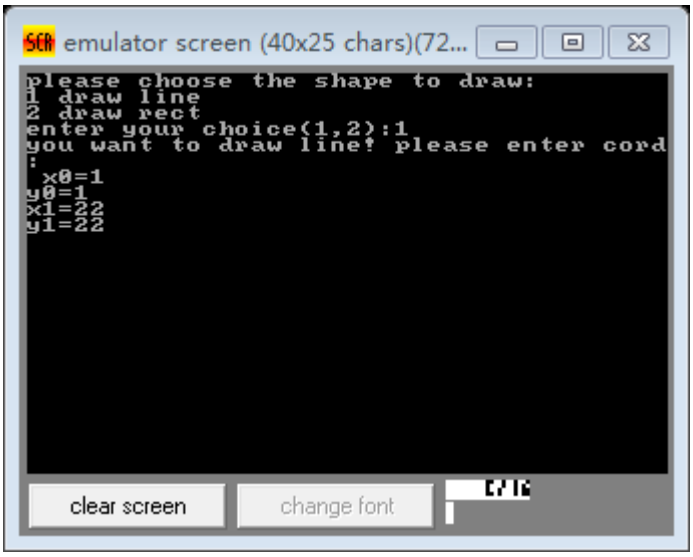
```
to draw:",0ah,0dh,'$'
    tip1 db "1 draw line",0ah, 0dh,'$'
    tip2 db "2 draw rect",0ah, 0dh,'$'
    tip3 db "enter your choice(1,2):",'$'

    ShowTipBeg:
    mov dx, offset tip0
    mov ah, 9
    int 21h

    mov dx, offset tip1
    mov ah, 9
    int 21h

    mov dx, offset tip2
    mov ah, 9
    int 21h

    mov dx, offset tip3
    mov ah, 9
    int 21h

ret
ShowTip endp

DEFINE_SCAN_NUM
DEFINE_GET_STRING
```

运行结果如下：

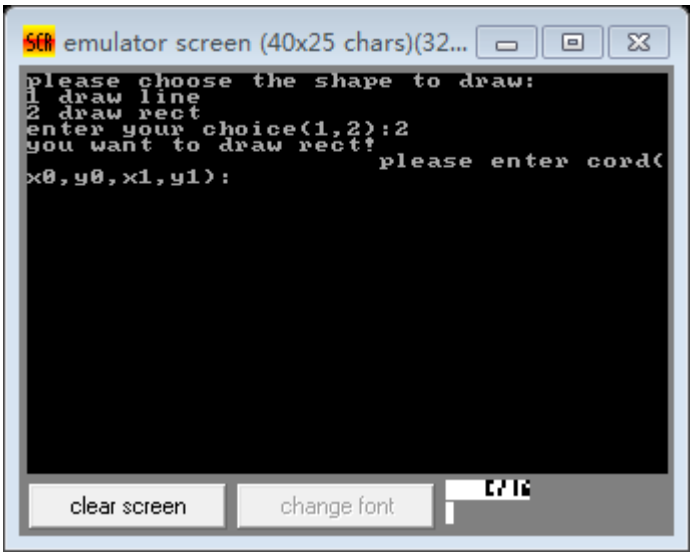


选择 1 后结果为：

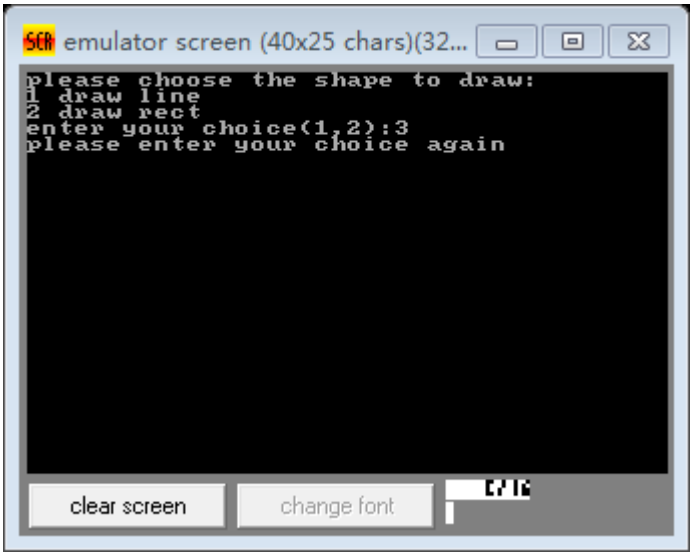




选择 2 后结果为：



选择错误的结果



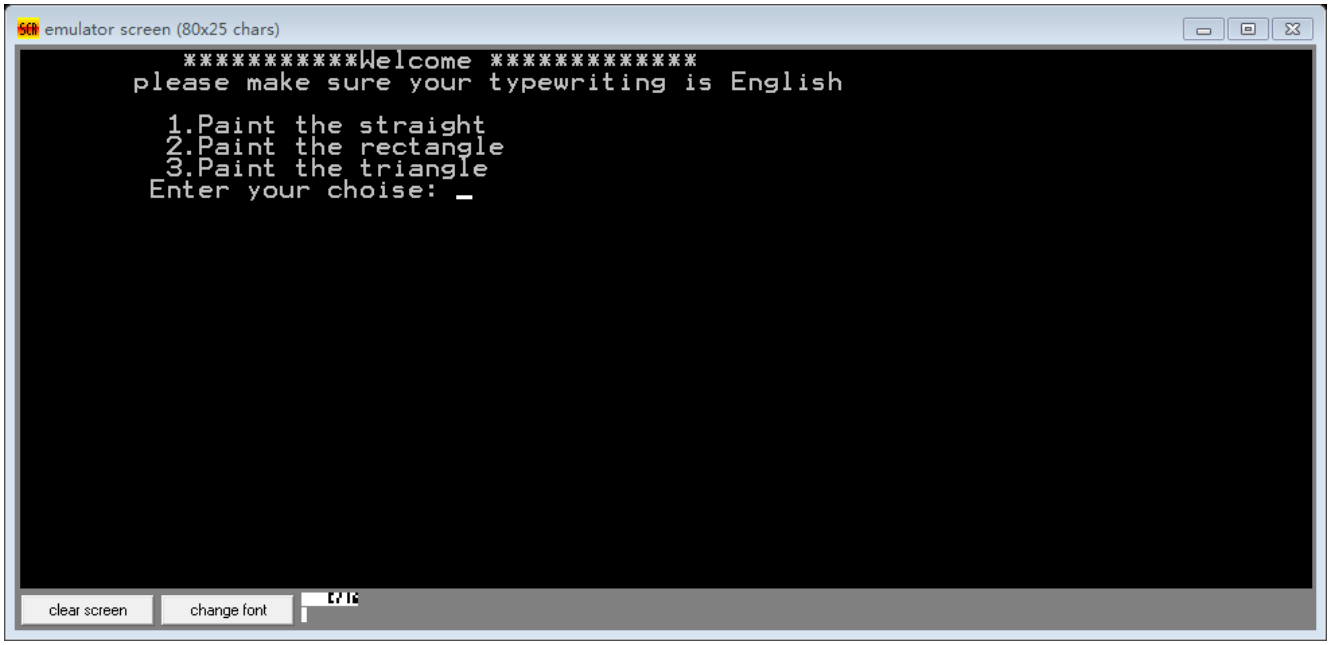
第三部分　汇编语言图形绘制过程：

一、 提示信息的显示：

```
include emu8086.inc
org 100h
start:
mov al,03h
mov ah,0
int 10h

 ;显示  msg0
GOTOXY    10,0
mov dx, offset msg0
mov ah, 9
int 21h

;显示  msg1
GOTOXY    7,1
mov dx, offset msg1
mov ah, 9
int 21h

;显示 msg2
GOTOXY    9,3
mov dx, offset msg2
mov ah, 9
int 21h

;显示 msg3
GOTOXY    9,4
mov dx, offset msg3
mov ah, 9
int 21h

;显示 msg4
GOTOXY    9,5
mov dx, offset msg4
mov ah, 9
int 21h

;显示 msg5
GOTOXY    8,6
mov dx, offset msg5
mov ah, 9
int 21h

CALL     SCAN_NUM
MOV       AX, CX
```

```
msg0    DB   "***********Welcome! *************$"
msg1    DB   "please make sure your typewriting is English$"
              ;确定用英文输入法输入数字
msg2    DB   "1.Paint the straight$"    ;画出直线
msg3    DB   "2.Paint the rectangle$" ;画出矩形
msg4    DB   "3.Paint the triangle$" ; 画出三角形
msg5    DB   "Enter your choise: $" ;选择（按数字+回车）
msg6    DB   "Enter the Starting point coordinate: $"   ;直线起点坐标
msg7    DB   "Enter the Ending point coordinate:$ "     ; 直线终点坐标
msg8    DB   "Enter the upper-left corner coordinate: $" ;左上角坐标
msg9    DB   "Enter the width and length: $" ;宽和长
msg10    DB   "Enter 1 to exit: $" ;
msg11    DB   "Enter the PointA coordinate: $" ;顶点坐标
msg12    DB   "Enter the PointB coordinate: $" ;顶点坐标
msg13    DB   "Enter the PointC coordinate: $";顶点坐标

;声明各使用的库函数
DEFINE_SCAN_NUM
```

运行后的显示结果为


emulator screen (80x25 chars)
**********Welcome **********
please make sure your typewriting is English
1.Paint the straight
2.Paint the rectangle
3.Paint the triangle
Enter your choise:
clear screen
change font

一、 直线的绘制

```
001  include emu8086.inc
002 org 100h
003 start:
004 mov al,03h
005 mov ah,0
006 int 10h
007
008 Straiht:
009          GOTOXY  8,10
010          mov dx, offset msg6;输入起始点坐标
011          mov ah, 9
012          int 21h
013          CALL    SCAN_NUM
014
015          ;将起始点坐标放入栈内保存
016          mov ax,cx
017          push ax
018
019          GOTOXY  50,10
020          CALL    SCAN_NUM
021          mov bx,cx
022          push bx
023
024          GOTOXY  8,11
025
026          mov dx, offset msg7;输入终点坐标
027          mov ah, 9
028          int 21h
029          CALL    SCAN_NUM
030
031          ;将终点坐标放入栈内保存
032          mov ax,cx
033
034          GOTOXY  50,11
035          CALL    SCAN_NUM
036          mov bx,cx
037
038           pop dx
039           pop cx;调出栈供使用
040
041           cmp ax,cx
042           je shuxian ;若横坐标相等， 则跳转到画竖线
043           cmp bx,dx
044           je hengxian;若纵坐标相等， 则跳转到画横线
045           jmp xiexian;其他情况跳转到画斜线
```

在画横线和数线时，可以直接用循环操作

在斜线时，应当将斜率求出，在球斜率的过程中要逐一判断 y1y2 和 x1x2 的大小关系，从而保证所化直线的准确性。

（一） 水平直线

```
090 hengxian:;画横线代码
091         push bx
092         push ax
093         mov al,13h
094         mov ah,0
095         int 10h
096         pop bx
097         pheng:
098             mov al,1100b
099             mov ah,0ch
100             inc cx
101             cmp cx,bx
102             int 10h
103         jne loop pheng
104 
105         ;按1返回到start
106         GOTOXY  10,1
107         mov dx, offset msg10
108         mov ah, 9
109         int 21h
110         CALL    SCAN_NUM
111         MOV     AX,1
112         cmp     al,cl
113         je start
114
```

（二）垂直直线

```
067
068 shuxian:;画竖线代码
069         mov al,13h
070         mov ah,0
071         int 10h
072         pshu:
073             mov al,1100b
074             mov ah,0ch
075             inc dx
076             cmp dx,bx
077             int 10h
078         jne loop pshu
079
080         ;按1返回到start
081         GOTOXY  10,1
082         mov dx, offset msg10
083         mov ah, 9
084         int 21h
085         CALL    SCAN_NUM
086         MOV     AX,1
087         cmp     al,cl
088         je start
089
090 hengxian:;画横线代码
091         push bx
092         push ax
093         mov al,13h
094         mov ah,0
```

（三）倾斜的直线

```
xiexian:;画斜线代码
        mov x1,cx
        mov y1,dx
        mov x2,ax
        mov y2,bx
        mov al,13h
        mov ah,0
        int 10h
        call oblique

        ;按1返回到start
        GOTOXY  10,1
        mov dx, offset msg10
        mov ah, 9
        int 21h
        CALL    SCAN_NUM
        MOV     AX,1
        cmp     al,cl
        je start
```

```
;oblique是由各输入值求斜率、近视画直线的函数
oblique proc
        mov ax,x2
        mov bx,x1
        cmp ax,bx ;比较两点的横坐标
        jge  x2po
        jl   x2ne
    x2po:  ;x2>=x1时
        mov s1,1
        sub ax,bx
        mov xd,ax
        jmp y

    x2ne:  ;x2<x1时
        mov s1,-1
        sub bx,ax
        mov xd,bx
        jmp y
    y:
        mov ax,y2
        mov bx,y1
        cmp ax,bx;比较两点的纵坐标
        jge  y2po
        jl   y2ne
    y2po:  ;y2>=y1时
        mov s2,1
        sub ax,bx
        mov yd,ax
        jmp subyx

    y2ne:  ;y2<y1时
        mov s2,-1
        sub bx,ax
        mov yd,bx
        jmp subyx
```

```
subyx:    ;xd yd比较
          mov ax,xd
          mov bx,yd
          cmp bx,ax
          jge  interchange;yd>=xd
          jl   nointerchange;yd<xd
interchange:  ;yd>=xd时的代码
    mov yd,ax ;其中yd代表纵坐标之差
    mov xd,bx ;其中xd代表横坐标之差
    mov interflag,1
    jmp  cal
nointerchange:
    mov interflag,0
    jmp cal
cal:
   mov ax,yd
   add ax,ax
   mov bx,xd
   sub ax,bx;2*yd-xd
   mov p,ax

   mov dx,y1
   mov cx,x1
paint:
        mov al,1100b;设置颜色
        mov ah,0ch;change color for a single pixel
        int 10h

        cmp p,0
        jge ppos;2*yd-xd>=0
        jl  pneg;2*yd-xd<0

    ppos:
         cmp interflag,0
         je interflagpos;yd<xd
         jne interflagneg;yd>=xd
```

```
interflagpos:
     mov ax,s2
     mov bx,y1
     add bx,ax
     mov y1,bx
     mov ax,p
     mov bx,xd
     add bx,bx
     sub ax,bx
     mov p,ax
    jmp  plot
interflagneg:
     mov ax,s1
     mov bx,x1
     add bx,ax
     mov x1,bx
     mov ax,p
     mov bx,xd
     add bx,bx
     sub ax,bx
     mov p,ax
    jmp  plot
```