4.3文字的生成技术

- 常用的方法有：

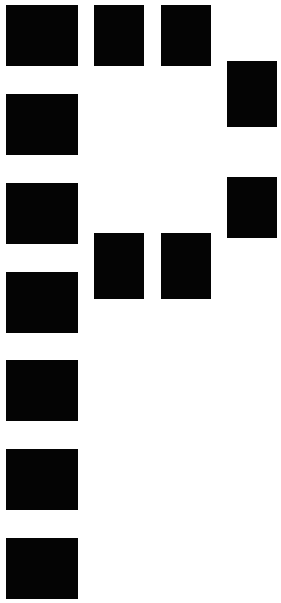
  - 点 阵 式 pDotText
  - 矢 量 式 pShiLiangText
  - 编 码 式 directText

- 点阵式字符

- DEF：每个字符都定义成一个称为掩膜的矩阵。矩阵中的元素都是一位二进制数，当该位为1时，表示字符的笔划经过此位，对应于此位的象素应置为字符颜色；当该位为0时，表示字符的笔划不经过此位，对应于此位的象素应置为背景色或不改变。

P-DotText

掩膜的矩阵(16×16)



0000000000000000

0001111111110000

0001100000011000

0001100000001100

0001100000001100

0001100000011000

0001100001100000

0001111111000000

0001100000000000

0001100000000000

0001100000000000

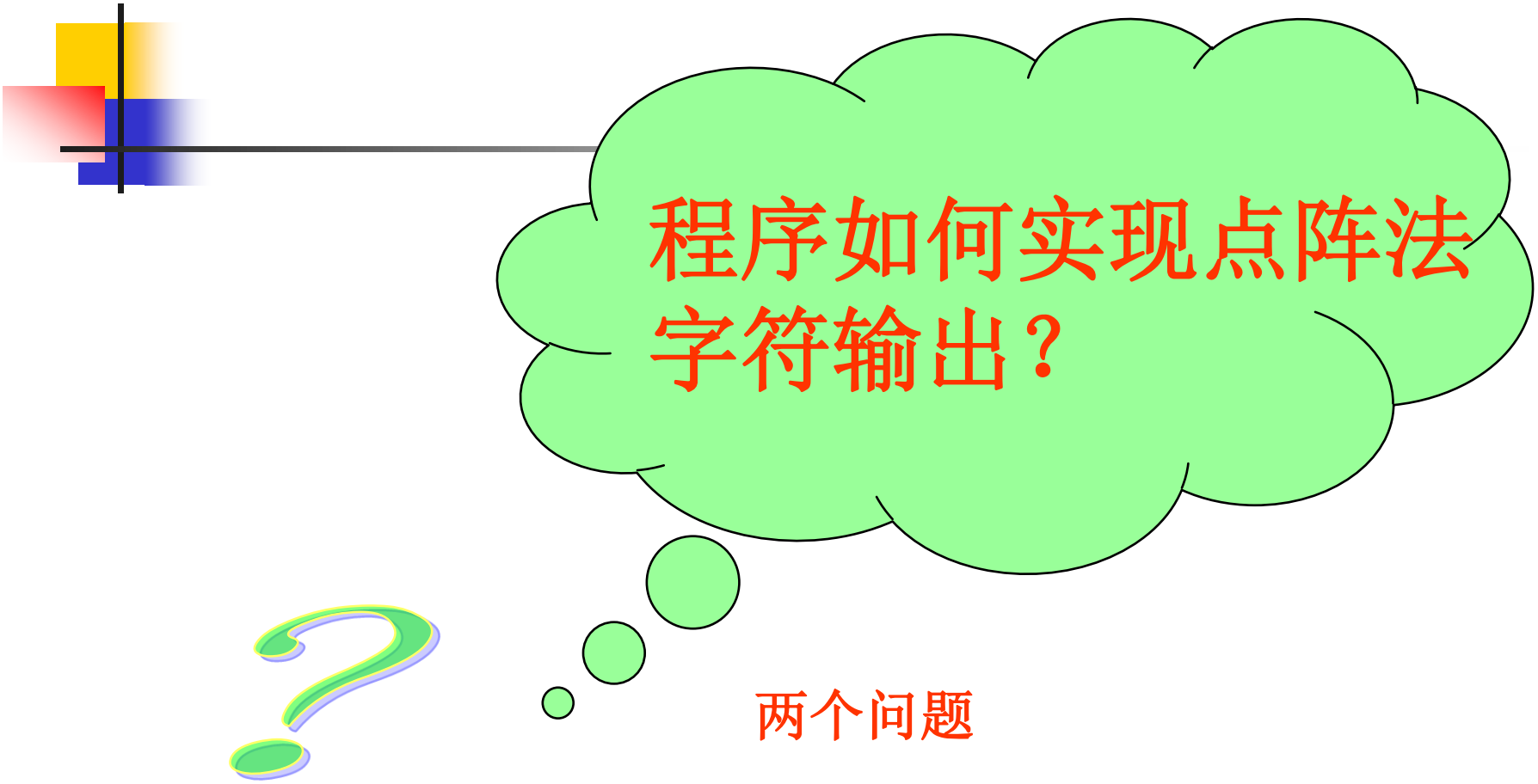
0001100000000000

0000000000000000

0000000000000000

当该位为1时，表示字符的笔划经过此位

当该位为0时，表示字符的笔划不经过此位



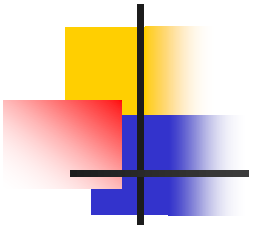
两个问题

（1）如何表示掩膜的矩阵？

（2）如何确定各个位是1还是0

- Int （16位)
- Int textP[16]={
- ox0000,
- 0x0000,
- 0x1FF0,
- 0x1818,
- ……
- 0x0000
- }

0000000000000000
0000000000000000
0001111111110000
0001100000011000
0001100000001100
0001100000001100
0001100000011000
0001100001100000
0001111111000000
0001100000000000
0001100000000000
0001100000000000
0001100000000000
0000000000000000
0000000000000000
0000000000000000



Text[16]	mask[16]
0000000000000000	1000000000000000
0001111111110000	0100000000000000
0001100000011000	0010000000000000
0001100000001100	0001000000000000
0001100000001100	0000100000000000
0001100000011000	0000010000000000
0001100001100000	0000001000000000
0001111111000000	0000000100000000
0001100000000000	0000000010000000
0001100000000000	0000000001000000
0001100000000000	0000000000100000
0001100000000000	0000000000010000
0000000000000000	0000000000001000
0000000000000000	0000000000000100
0000000000000000	0000000000000010
	0000000000000001

如何判断字模中各个位的子呢？

```
int  text[16]={ 0x0000, 0x0000, 0x0000,
0x0000, 0x0000, 0x0000,  0xFFFF,  0xFFFF,0xFFFF,
0xFFFF, 0x0000, 0x0000,  0x0000, 0x0000,
0x0000,0x0000 };

int mask[16]={0x8000, 0x4000, 0x2000, 0x1000,
0x0800, 0x0400, 0x0200, 0x0100, 0x0080, 0x0040,
0x0020, 0x0010, 0x0008, 0x0004, 0x0002, 0x0001 }
```

```
void displayText(int x0,int y0,int Text[],int color)
{ int i,j;
   for (i=0;i<16;i++) {   //行
      for(j=0;j<16;j++) {   //列
         if ((Text[i] & mask[j])!=0)
            putpixel(x0+j,y0+i,color) ;
   }}
}
```

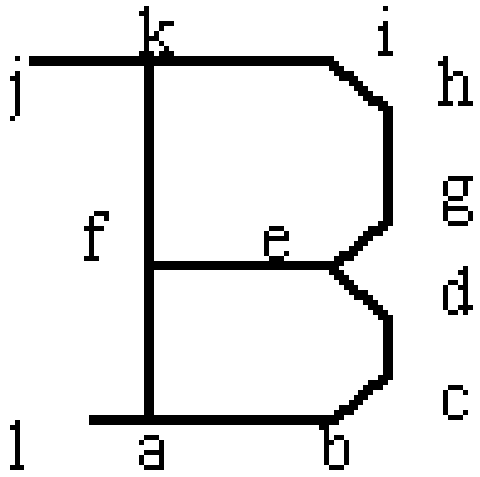
- 点阵式字符时主要的文字表示形式。
- 常用的点阵大小有

- **5 × 7、7 × 9、8 × 8、16 × 16等等**
- 当点阵变大时，字型可以做得非常漂亮。

- 优点：字形美观是字符表示的主要形式
- 缺点：旋转比较困难、占用的存储空间较大

矢量式字符

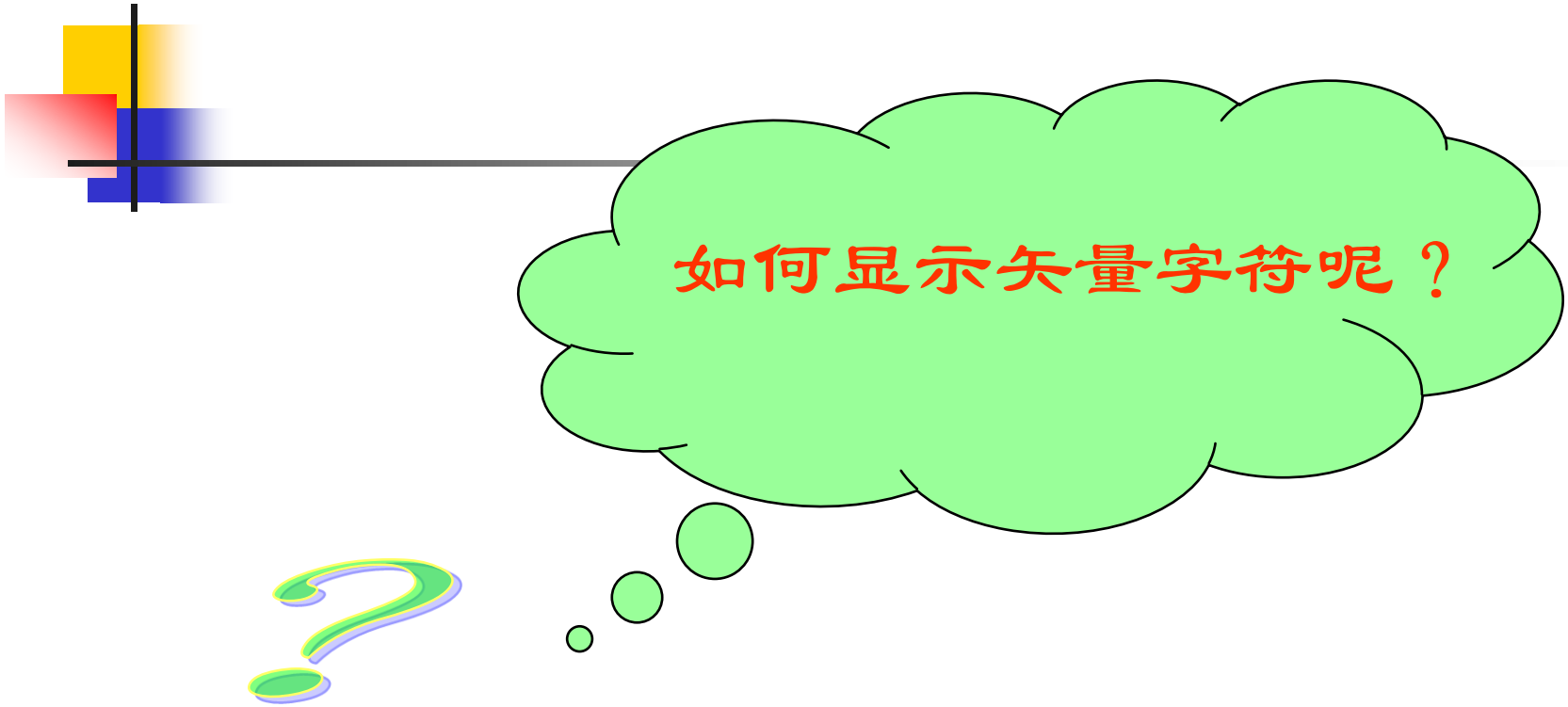
- DEF：将字符看作是一个图形，用点坐标的序列来表示一个字符，相邻两点表示一条矢量，字符的形状便由矢量序列刻划。



例：“B”是由顶点序列

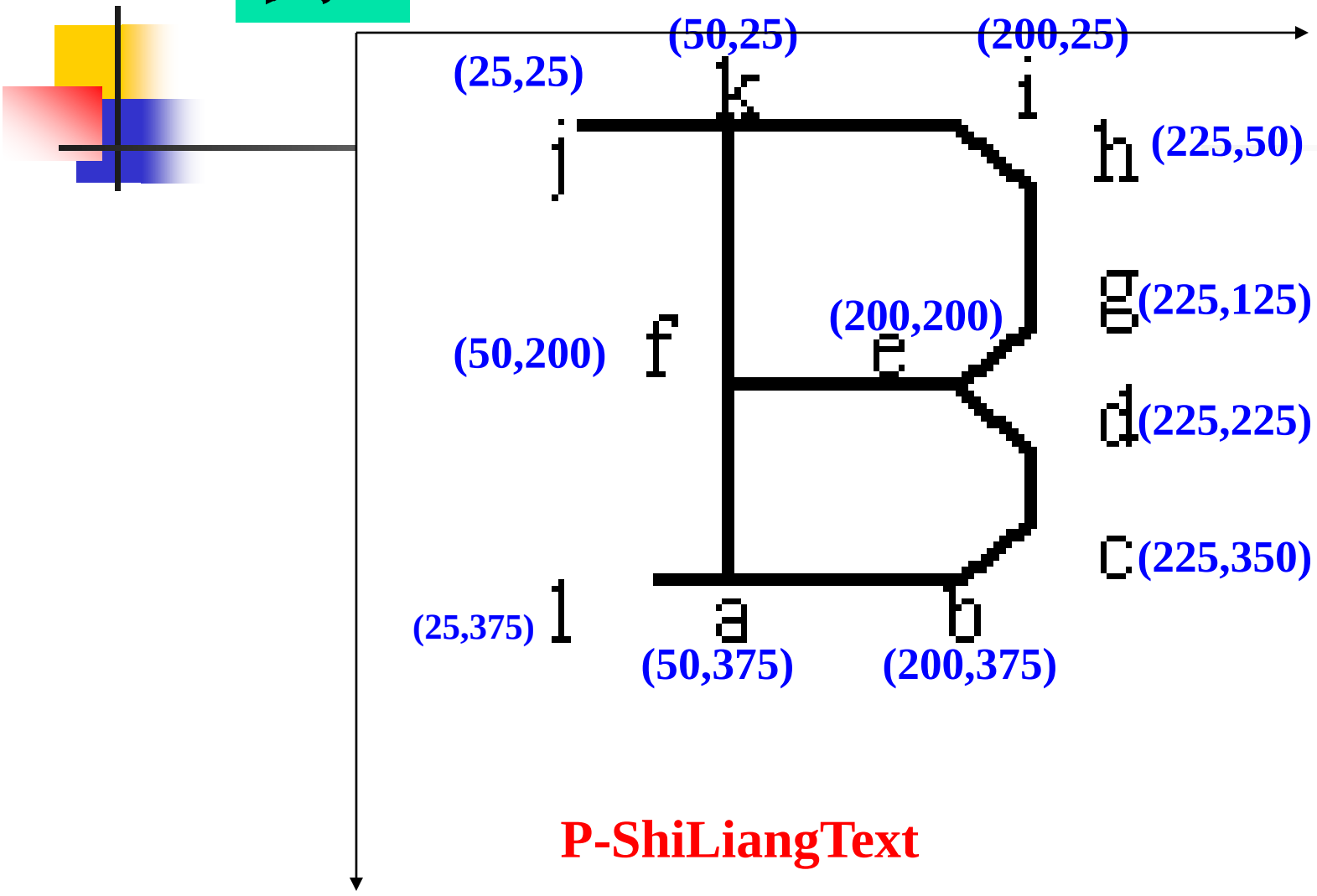
{a, b, c, d, e, f, e, g, h, I, j, k ,a, l, a}

的坐标表达



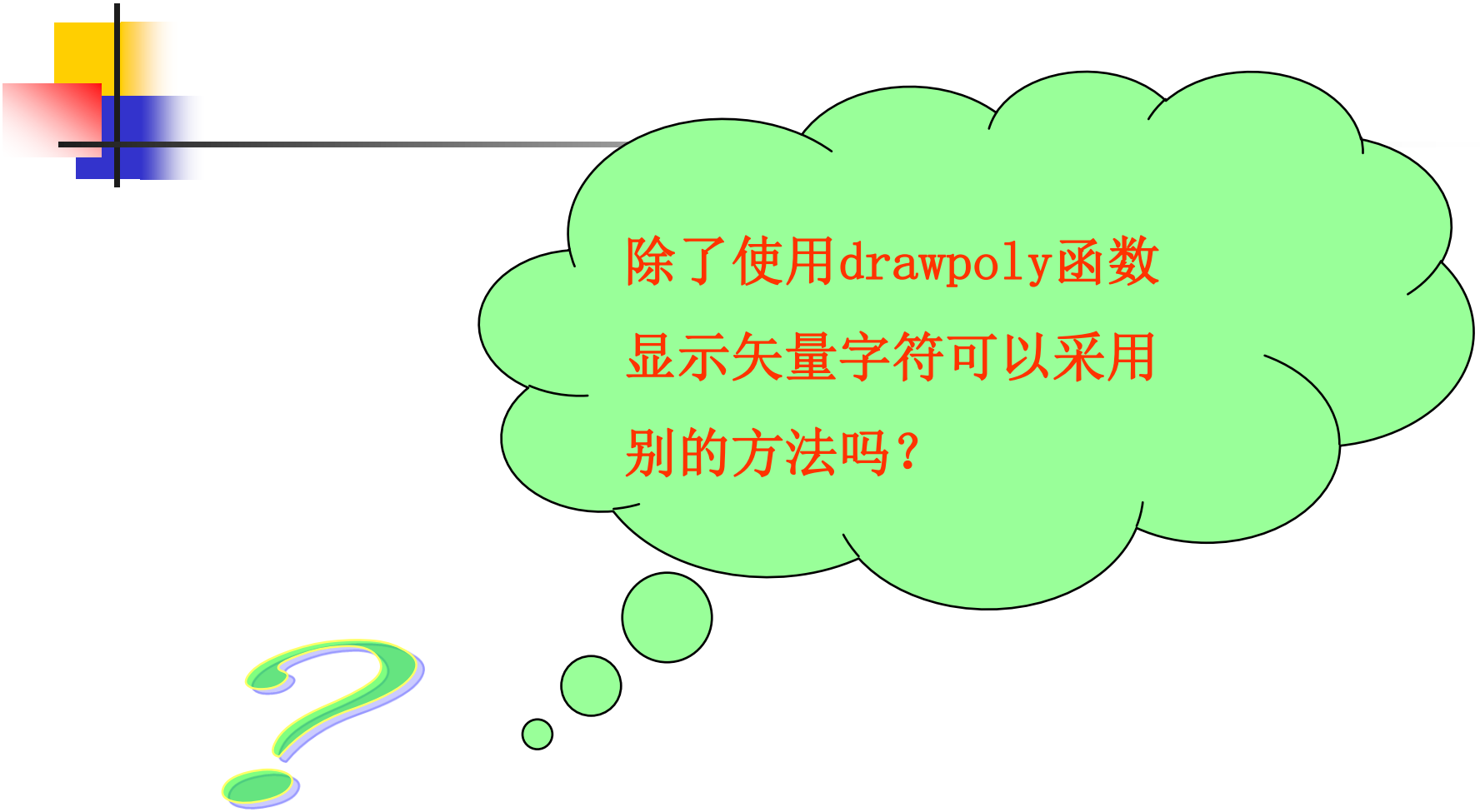
调用矢量式字符的过程相当于输出一个polylin

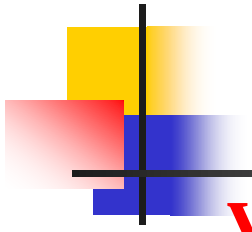
例：



P-ShiLiangText

```
int text[]= {50,375,  200,375,  225,350,225,
225,200,200,  50,200,  200,200,  225,175,  225,50,
200,25,  25,25,  50,25,  50,375,  25,375,  50,375};
void displayText(int x,int y,int *text,int len){
   int i;
   int *newText=malloc(sizeof(int)*len);
   for(i=0;i<len/2;i++)   {
      newText[2*i]=x+text[2*i]  ;
      newText[2*i+1]=y+text[2*i+1];
   }
      drawpoly(len/2,newText);
}
```


除了使用drawpoly函数
显示矢量字符可以采用
别的方法吗？



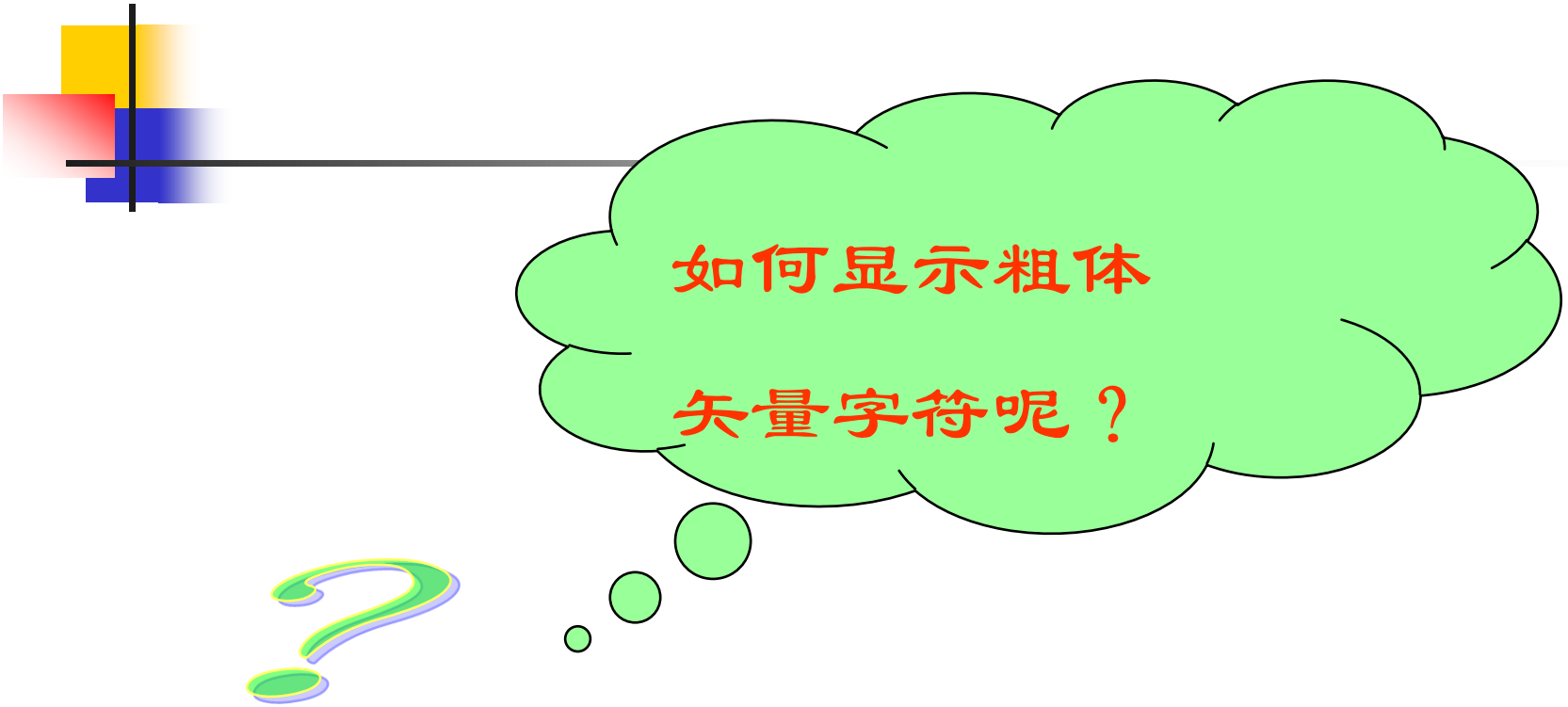
```
void displayText2(int x,int y,int *text,int len)
    {   int i;
        moveto(x+text[0],y+text[1]);
        for(i=1;i<len/2;i++)
            lineto(x+text[2*i],y+text[2*i+1]);
}
```

- 矢量式字符的优点：
- 矢量式字符具有和图形相一致的数据结构，因而可以接受任何对于图形的操作，如放大、旋转，平移等

a
b
c
d
e
f
e
g
h
I
j
k
a
l
a

$$\begin{pmatrix} 50 & 375 & 1 \\ 200 & 375 & 1 \\ 225 & 350 & 1 \\ 225 & 225 & 1 \\ 200 & 200 & 1 \\ 50 & 200 & 1 \\ 200 & 200 & 1 \\ 225 & 175 & 1 \\ 225 & 50 & 1 \\ 200 & 25 & 1 \\ 25 & 25 & 1 \\ 50 & 25 & 1 \\ 50 & 375 & 1 \\ 25 & 375 & 1 \\ 50 & 375 & 1 \end{pmatrix} * \begin{pmatrix} a & b & p \\ c & d & q \\ L & m & s \end{pmatrix} = \begin{pmatrix} x' & y' & 1 \end{pmatrix}$$

字符B的旋转、平移、缩放，和斜体(x方向错切15)

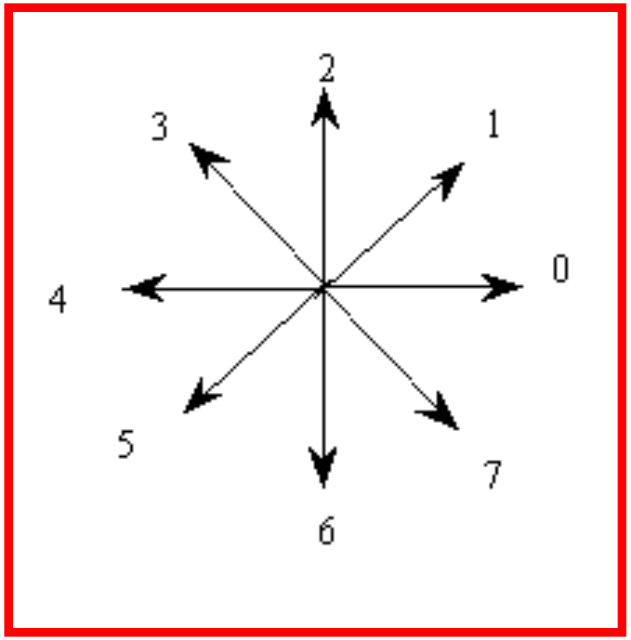

如何显示粗体
矢量字符呢？

```
void displayText2(int x,int y,int *text,int len)
{   int i;
 setlinestyle(SOLID_LINE,0,3);
 moveto(x+text[0],y+text[1]);
  for(i=1;i<len/2;i++)
      lineto(x+text[2*i],y+text[2*i+1]);
}
```

方向编码

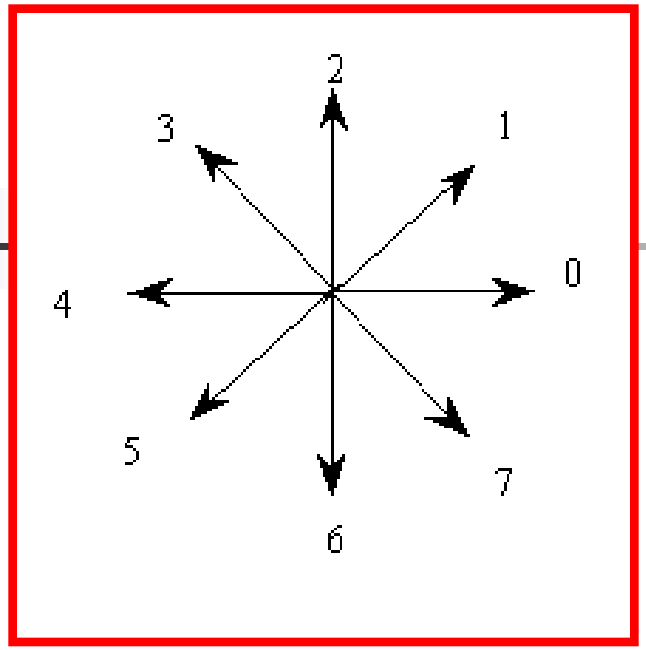
DEF:方向编码式字符用有限的若干种方向编码来表达一个字符.

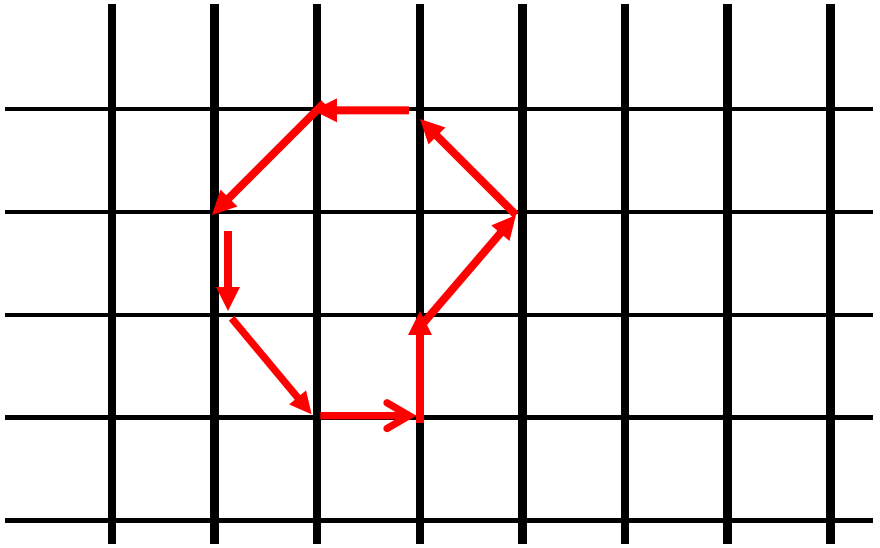
8个方向的编码为0 ~ 7



一个字符就可以表示

为一连串方向码

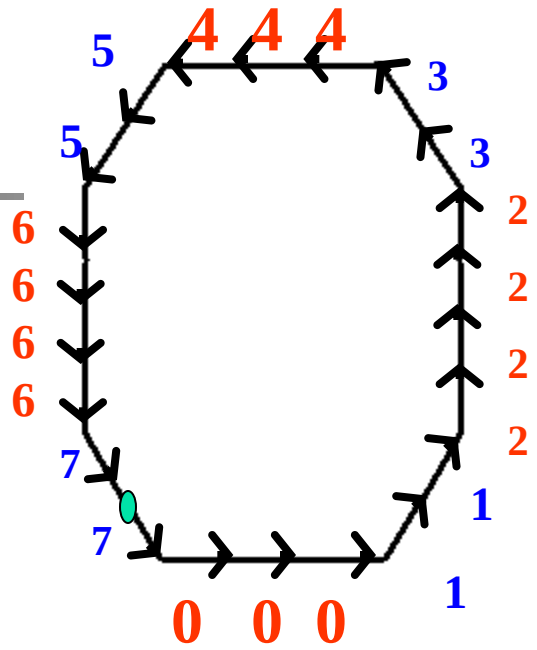




其中编码为

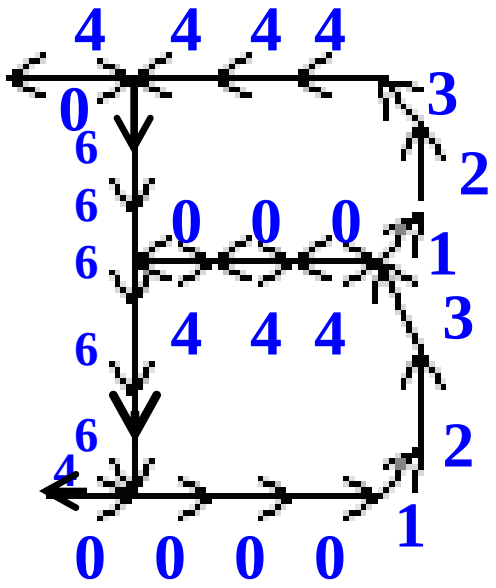
偶数方向的增量为固定长度1，

奇数方向的增量为固定长度 $\sqrt{2}$



方向编码是：

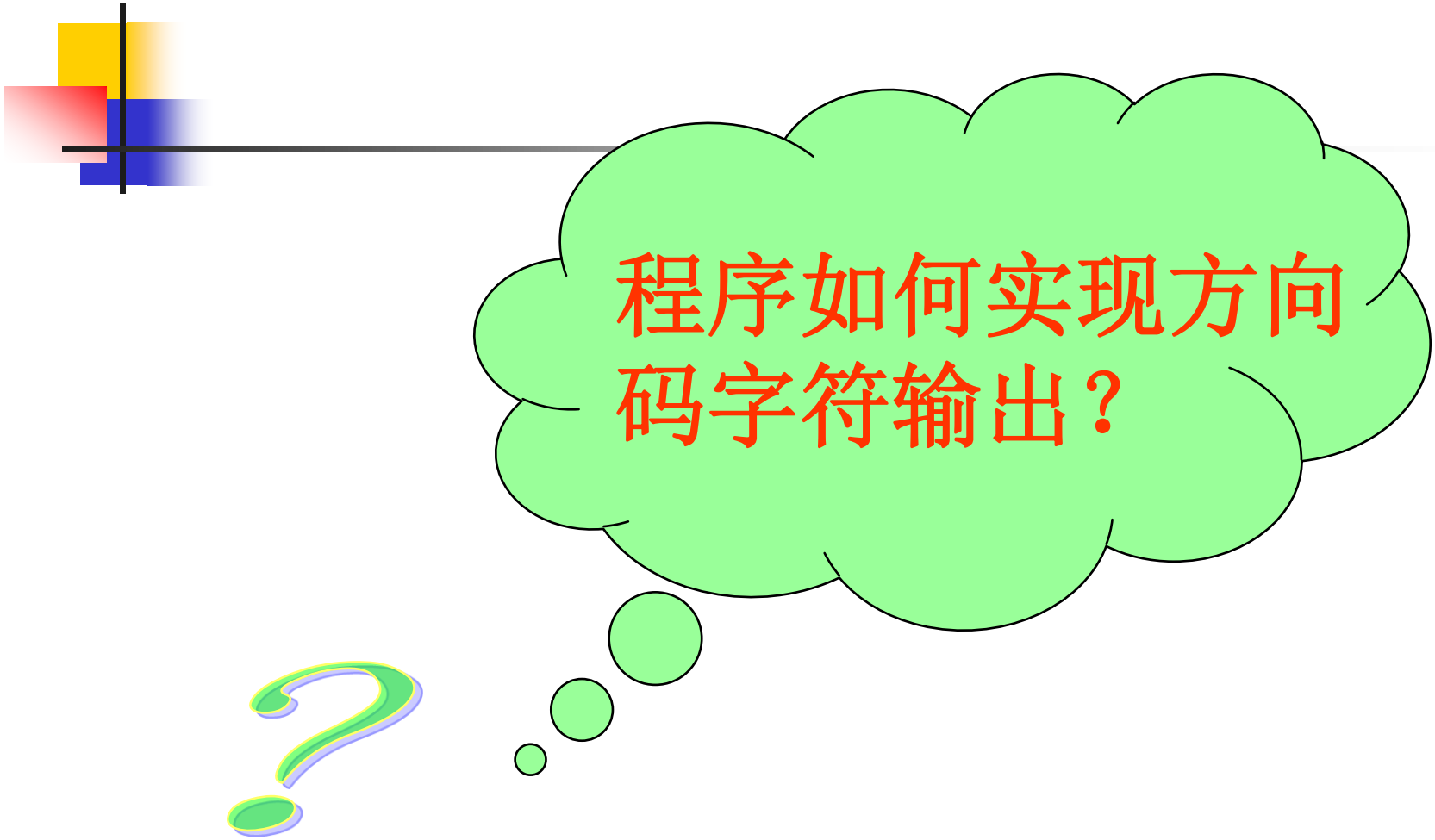
{ 000 11 2222 33 444 55 6666 77 }



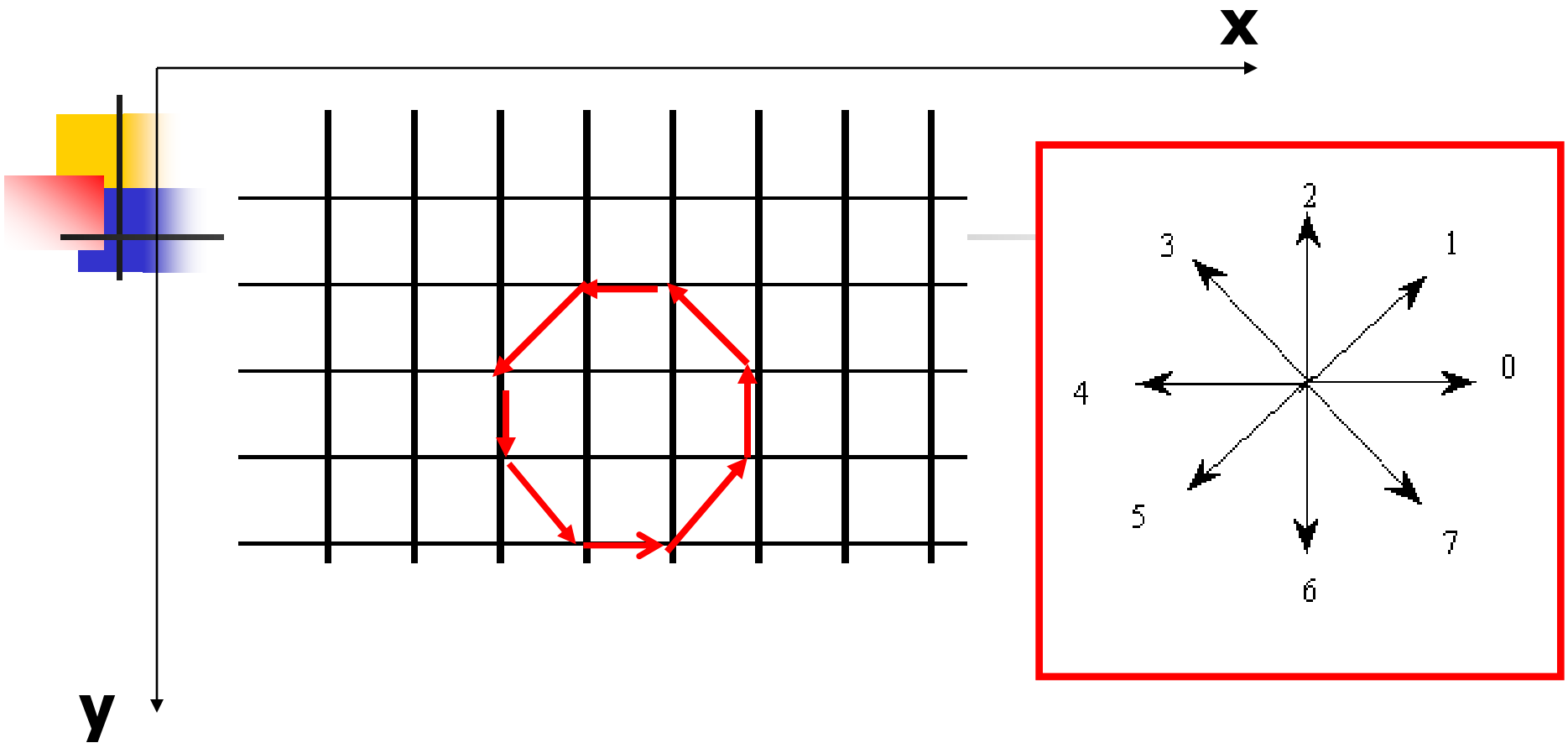
(a)

“B”就表示为8方向编码：

{0000 123 444 000 123 4444 0 66666640}

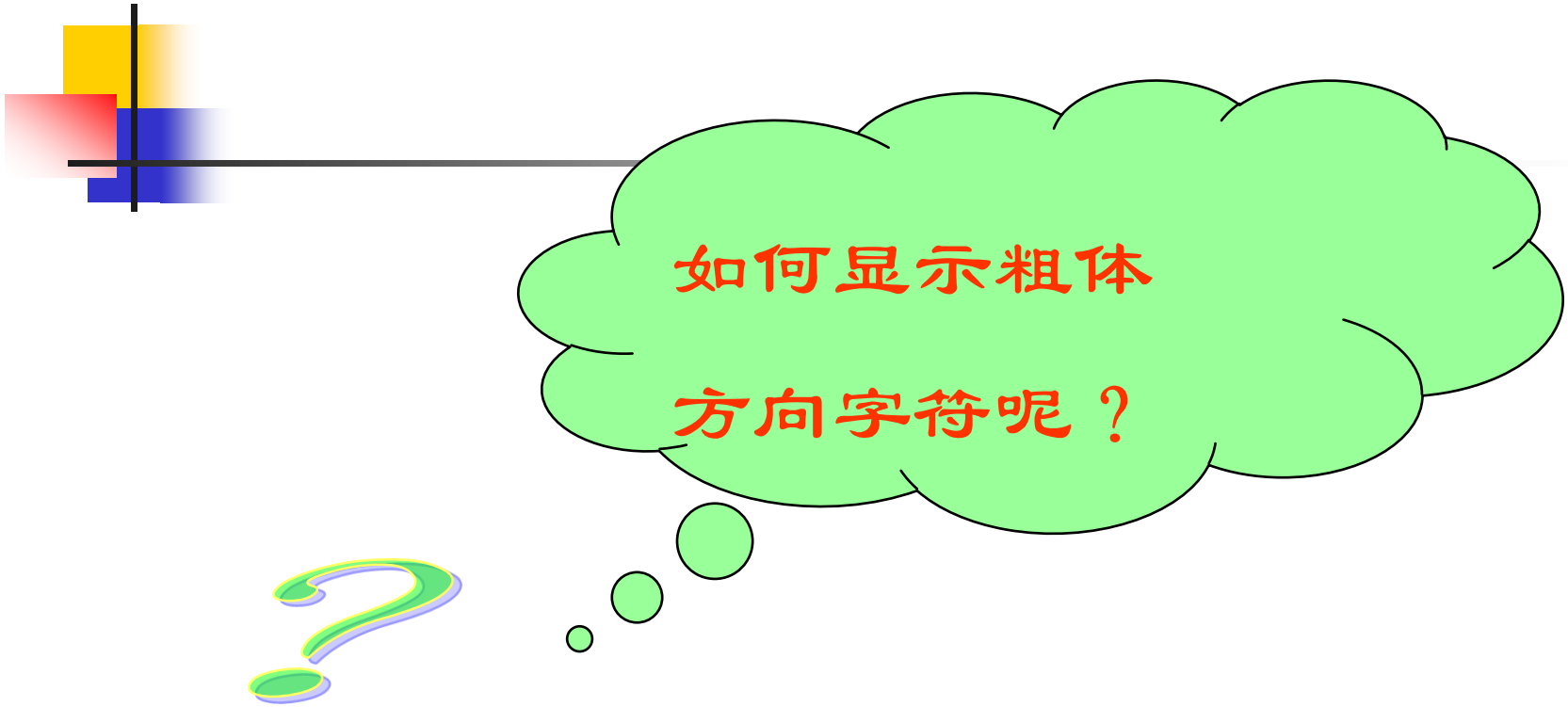


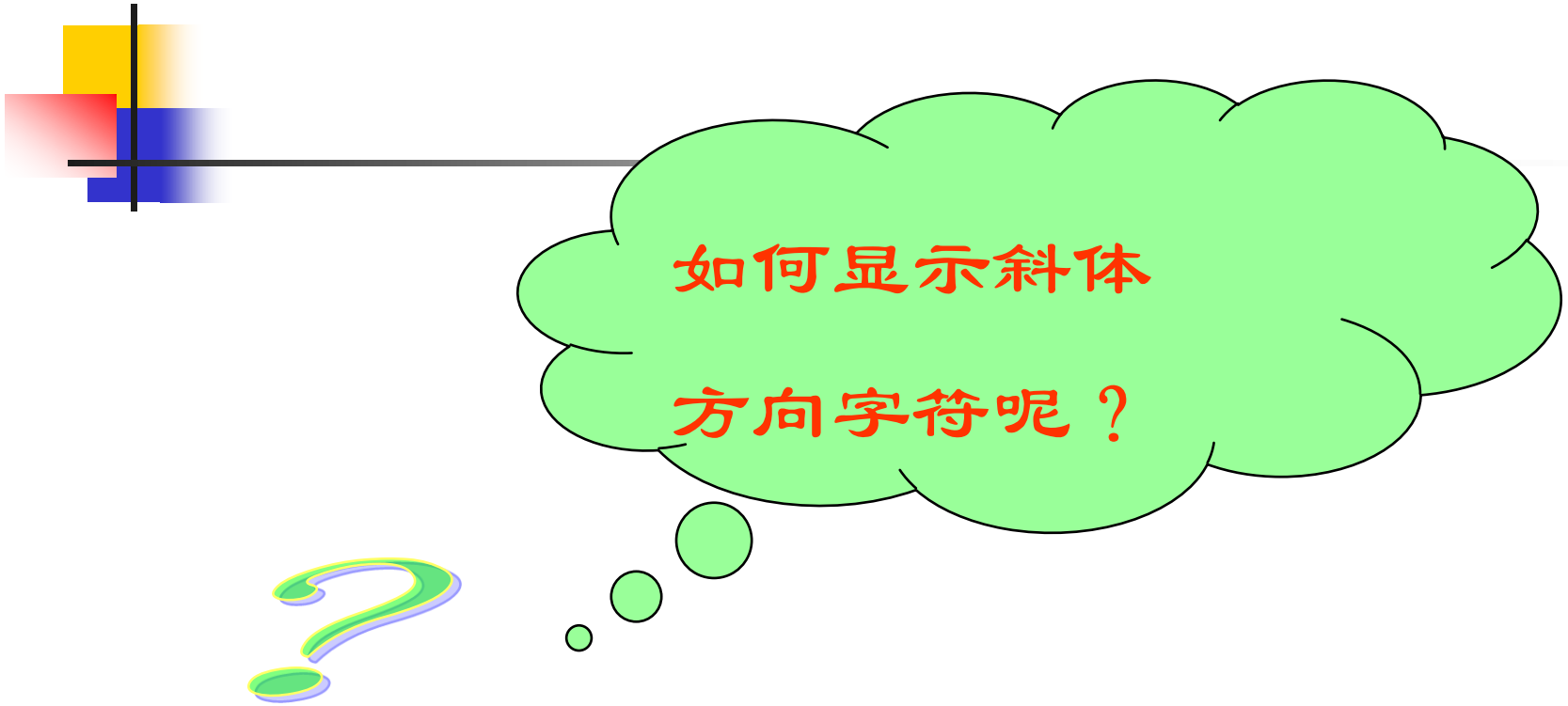
P-DirectText.c

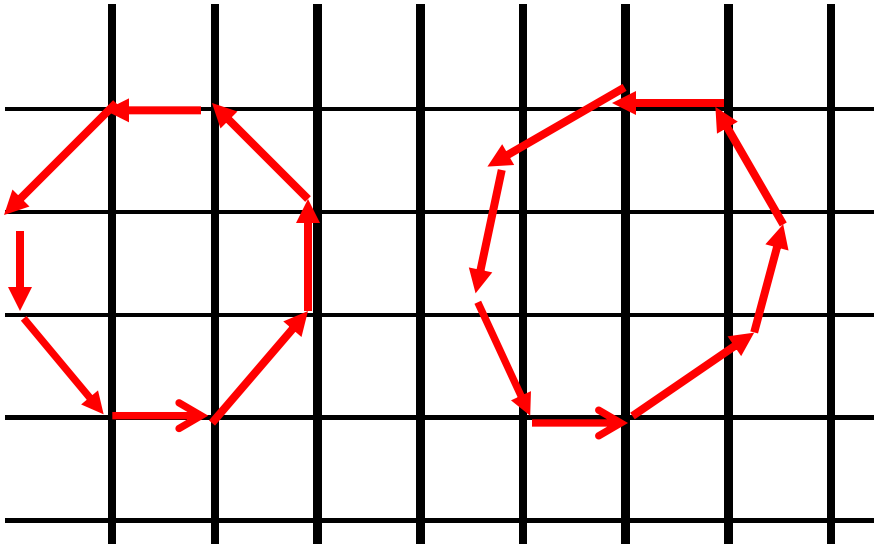


- **Model[8][2]= {{1,0},{1,-1}, {0,-1},{-1,-1},**
- **{-1,0}, {-1,1},{0,1},{1,1}};**

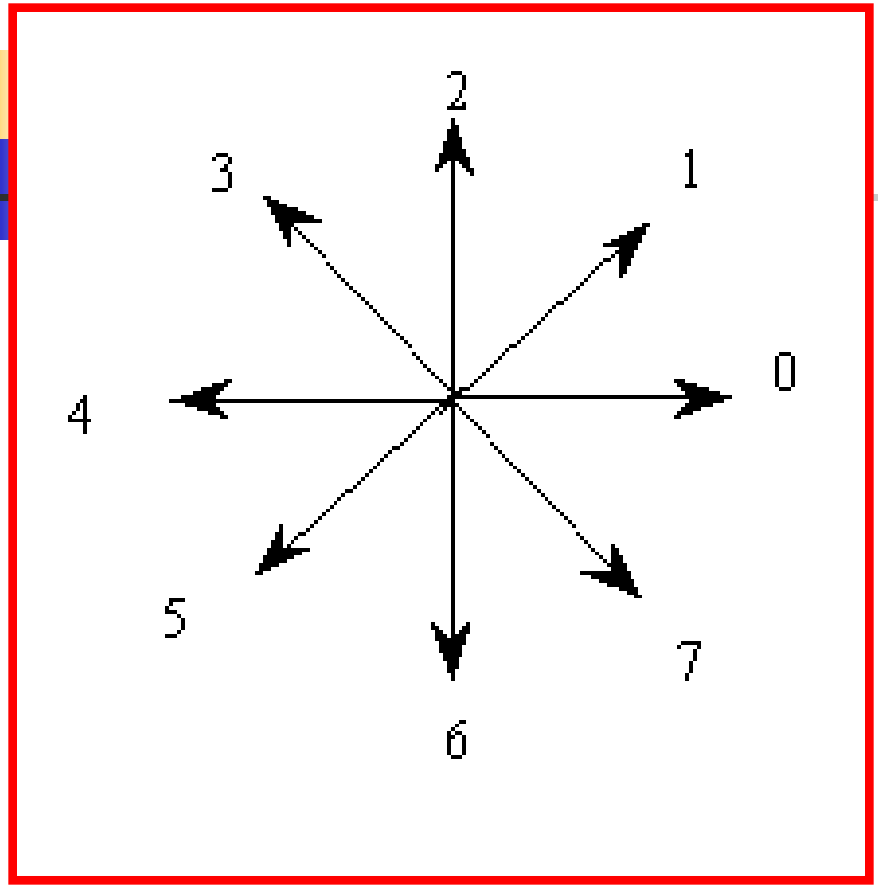
```
int direct[27]={0,0,0,0,1,  2,3,4,4,4,  0,0,0,1,2,
                3,4,4,4,4,    0,6,6,6,6,  6,4 };
void displayText(int x0,int y0,int direct[],
                                        int len,int size)
{  int i;
    float Model[8][2]=  {{1,0},{1,-1}, {0,-1},{-1,-1}, {-1,0},
                         {-1,1},{0,1},{1,1}};
   moveto(x0,y0);
   for (i=1;i<len;i++) {
       linerel(Model [direct[i]][0]*size,
                       Model[direct[i]][1]*size)  ;

   }
}
```

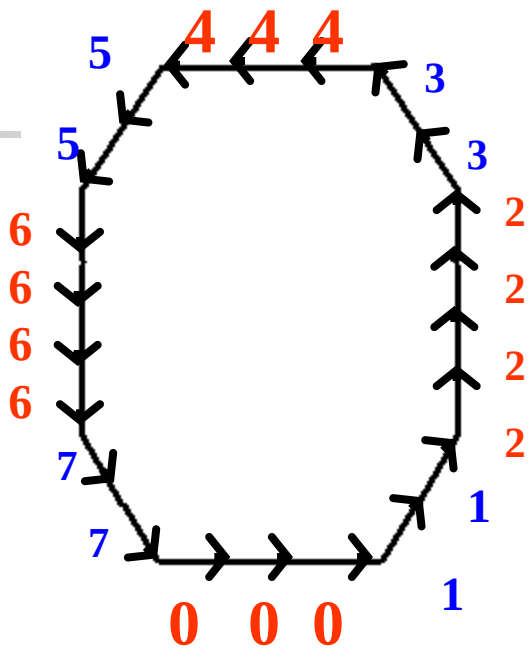

如何显示粗体
方向字符呢？


如何显示斜体
方向字符呢？



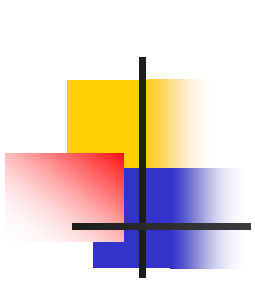
```
void displayText(int x0,int y0,int Text[],int len,int size)
 {  int i;
  float direct[8][2]=
  {  {1,0},
     {sqrt2 * cos30,-sqrt2*sin30},
     {+sin15,-cos15},
     {-sqrt2*sin30,-sqrt2*cos30},
     {-1,0},
     {-sqrt2*cos30,sqrt2*sin30},
     {-sin15,cos15},
     {-sqrt2*sin30,sqrt2*cos30}   };
  moveto(x0,y0);
  for (i=1;i<len;i++)
linerel(direct[Text[i]][0]*size,direct[Text[i]][1]*size);
 }
```

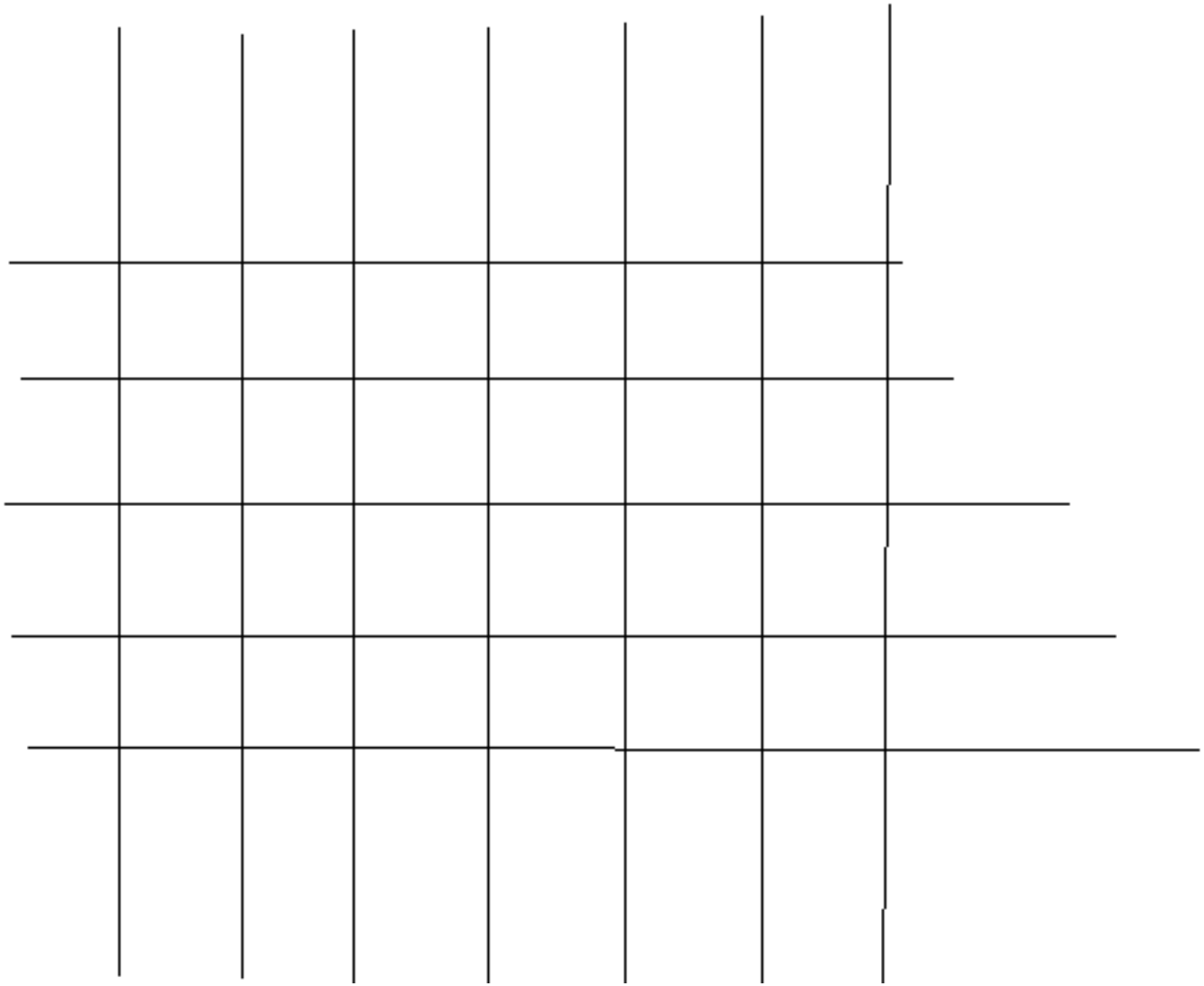

2
3
1
4
0
5
7
6


4 4 4
5
3
5
3
6
2
6
2
6
2
6
2
7
1
7
1
0 0 0

```
void displayText(int x0,int y0,int direct[],int len,int size，int alfa)
{  int i;
float Model[8][2]=  {{1,0},{sqrt2*cos(45-alfa),-sin(45-alfa)}, {+sin(alfa),-cos(alfa)},{-sin(45-alfa),-cos(45-alfa)},{-1,0}, {-cos(45-alfa),sin(45-lfa)},{sin(alfa),cos(alfa)},{sin(45-alfa),cos(45-alfa)}};
    moveto(x0,y0);
    for (i=1;i<len;i++) {
        linerel(Model [direct[i][0]*size,
                       Model[direct[i][1]*size);
    }
}
```

- 程序设计题（任选一）
- （1）建立一个十个字以上的字库，并能够显示该字库中的字符（选用任何字符输出方法）
- （2）实现DDA直线算法和DDA角度画圆算法
- (3)





程序：

```
#include  "stdio.h"
#include  "graphics.h"
#include  "math.h"
void draw_DDA_line(int x0,int y0,int x1,int y1)
 {float k;
  float x,y;
  int temp;
  float deltax,deltay ;
  k=abs(y1-y0)*1.0/abs(x1-x0) ;
  my_putpixel(x0,y0,1);
```

```
if (k<=1)
    {if (x0>x1)
       { temp=x0;
        x0=x1;
        x1=temp;
        temp=y0;
        y0=y1;
        y1=temp;
      }
    for(x=x0;x<x1;x++)
      { x=x+1;
      y=y+k;
      my_putpixel((int)x,(int)y,1) ;
      }
    }
```

```
if(k>1)
   {if(y0>y1)
     {temp=x0;
      x0=x1;
      x1=temp;
      temp=y0;
      y0=y1;
      y1=temp;
     }
   for(y=y0;y<y1;y++)
     { y=y+1;
       x=x+1/k;
       my_putpixel((int)x,(int)y,1);
       }
   }
}
```