

Trường Đại học Nha Trang
Bộ môn Kỹ thuật phần mềm



HƯỚNG DẪN GIẢI CHI TIẾT VÀ LẬP TRÌNH
MÔN HỌC: KỸ THUẬT ĐỒ HỌA

Biên soạn: ĐOÀN VŨ THỊNH

✉: thinhdv@ntu.edu.vn

☎: 0914949195

W: <https://github.com/thinhdoanvu>

Nha Trang, 2021

DDA Line Algorithm

Phương trình đường thẳng: $y=mx + b$

$$\frac{y_2 - y_1}{y - y_1} = \frac{x_2 - x_1}{x - x_1}$$

Đặt $dy = (y_2 - y_1)$ và $dx = (x_2 - x_1)$

$$\frac{dy}{y - y_1} = \frac{dx}{x - x_1}$$

$$(x - x_1)dy = (y - y_1)dx$$

Chia 2 vế cho dx và đặt $m = dy/dx$

$$y = mx - mx_1 + y_1 \text{ hay } y = mx + b, \text{ với } b = y_1 + mx_1$$

Phương trình đường thẳng: $Ax + By + C$

$$\text{Từ biểu thức: } \frac{dy}{y - y_1} = \frac{dx}{x - x_1}$$

$$\text{Ta có: } xd_y - yd_x + x_1d_y + y_1dx = 0$$

$$\text{Đặt } A = dy; B = -dx; \text{ và } C = x_1d_y + y_1dx$$

$$\text{Ta được: } Ax + By + C = 0$$

$$\text{Phương trình đường thẳng: } \begin{cases} x = tx_2 + (1-t)x_1 \\ y = ty_2 + (1-t)y_1 \end{cases}$$

$$\text{Ta có: } \frac{y_2 - y_1}{y - y_1} = \frac{x_2 - x_1}{x - x_1}$$

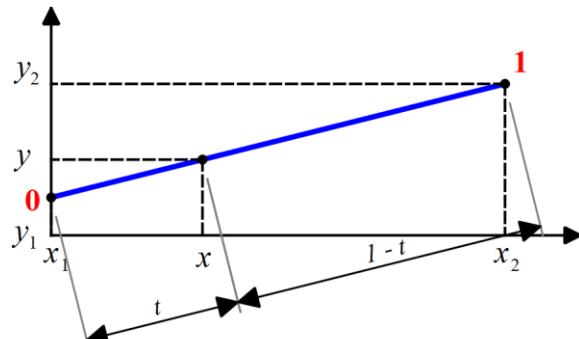
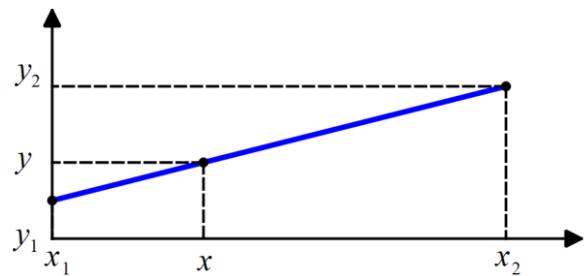
Hay

$$\frac{y_2 - y_1}{y - y_1} = \frac{1}{t}$$

$$t(y_2 - y_1) = y - y_1$$

$$y = ty_2 + (1-t)y_1$$

$$\text{Tương tự: } x = tx_2 + (1-t)x_1$$



DDA Line Algorithm

Câu hỏi 1. Sử dụng thuật toán DDA vẽ đoạn thẳng đi qua 2 điểm A(2;3) và B(12;8)

1.1. Trình bày các bước để thực hiện giải thuật trên

1.2. Lập trình mô phỏng các bước trên với x_A , y_A , x_B , y_B là các số nhập từ bàn phím

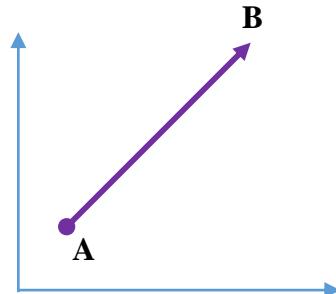
Bài làm:

1.1. Từ phương trình đường thẳng đi qua 2 điểm A và B

$$y = mx + b$$

Với $m = \frac{dy}{dx} = \frac{(y_B - y_A)}{(x_B - x_A)} = \frac{8 - 3}{12 - 2} = 0.5$

Và $x_A < x_B$, $y_A < y_B$



Nên: $x_{i+1} = x_i + 1$ và $y = round(f(x))$

Ta có:

$$y_0 = mx_0 + b$$

Suy ra:

$$y_i = mx_i + b$$

và

$$y_{i+1} = mx_{i+1} + b = m(x_i + 1) + b = mx_i + b + m$$

Vậy

$$y_{i+1} = round(y_i + m)$$

Bước thứ i	x_i	y_i	ROUND
0	2	3	3
1	3	$= 3 + 0.5 = 3.5$	4
2	4	$= 3.5 + 0.5 = 4$	4
3	5	$= 4 + 0.5 = 4.5$	5
4	6	$= 4.5 + 0.5 = 5$	5
5	7	$= 5 + 0.5 = 5.5$	6
6	8	$= 5.5 + 0.5 = 6$	6
7	9	$= 6 + 0.5 = 6.5$	7
8	10	$= 6.5 + 0.5 = 7$	7
9	11	$= 7 + 0.5 = 7.5$	8
10	12	$= 7.5 + 0.5 = 8$	8

DDA Line Algorithm

1.2. Lập trình mô phỏng

```
//khai bao thu vien
#include <graphics.h>
#include <math.h>
#define ROUND(a) ((int)(a+0.5))

//khai bao bien
int xa;
int xb;
int ya;
int yb;
float m;

//chuong trinh con
void nhapxy()
{
    printf("xA:= ");
    scanf("%d",&xa);
    printf("yA:= ");
    scanf("%d",&ya);
    printf("xB:= ");
    scanf("%d",&xb);
    printf("yB:= ");
    scanf("%d",&yb);
    m=(float) (yb-ya)/(xb-xa);
    printf("he so goc m = %f",m);
}

void DDA()
{
    int x;
    float y;
    initwindow(400,400);
    x=xa;
    y=ya;
    putpixel(x,ROUND(y),125);
    while(x<xb)
    {
        x=x+1;
        y=y+m;
        putpixel(x,ROUND(y),255);
        delay(100);
        printf("(%d;%d)\n",x,ROUND(y));
    }
}

//chuong trinh chinh
main()
{
    nhapxy();
    DDA();
    getch();
}
```

DDA Line Algorithm

Câu hỏi 2. Sử dụng thuật toán DDA vẽ đoạn thẳng đi qua 2 điểm A(12;8) và B(2;3)

1.1. Trình bày các bước để thực hiện giải thuật trên

1.2 Lập trình mô phỏng các bước trên với x_A , y_A , x_B , y_B là các số nhập từ bàn phím

Bài làm:

1.1. Từ phương trình đường thẳng đi qua 2 điểm A và B

$$y = mx + b$$

$$\text{Với } m = \frac{dy}{dx} = \frac{(y_B - y_A)}{(x_B - x_A)} = \frac{3 - 8}{2 - 12} = 0.5$$

Và $x_A > x_B$, $y_A > y_B$

Nên: $x_{i+1} = x_i - 1$ và $y = round(f(x))$

Ta có:

$$y_0 = mx_0 + b$$

Suy ra:

$$y_i = mx_i + b$$

và

$$y_{i+1} = mx_{i+1} + b = m(x_i - 1) + b = mx_i + b - m$$

Vậy

$$y_{i+1} = round(y_i - m)$$

Bước thứ i	x_i	y_i	ROUND
0	12	8	8
1	11	$= 8 - 0.5 = 7.5$	8
2	10	$= 7.5 - 0.5 = 7$	7
3	9	$= 7 - 0.5 = 6.5$	7
4	8	$= 6.5 - 0.5 = 6$	6
5	7	$= 6 - 0.5 = 5.5$	6
6	6	$= 5.5 - 0.5 = 5$	5
7	5	$= 5 - 0.5 = 4.5$	5
8	4	$= 4.5 - 0.5 = 4$	4
9	3	$= 4 - 0.5 = 3.5$	4
10	2	$= 3.5 - 0.5 = 3$	3

1.2. Lập trình mô phỏng

```
//khai bao thu vien
#include <graphics.h>
#include <math.h>
#define ROUND(a) ((int)(a+0.5))

//khai bao bien
int xa;
int xb;
int ya;
int yb;
float m;

//chuong trinh con
void nhapxy()
{
    printf("xA:= ");
    scanf("%d",&xa);
    printf("yA:= ");
    scanf("%d",&ya);
    printf("xB:= ");
    scanf("%d",&xb);
    printf("yB:= ");
    scanf("%d",&yb);
    m=(float) (yb-ya)/(xb-xa);
    printf("he so goc m = %f",m);
}

void DDA()
{
    int x;
    float y;
    initwindow(400,400);
    x=xa;
    y=ya;
    putpixel(x,ROUND(y),125);
    while(x>xb)
    {
        x=x-1;
        y=y-m;
        putpixel(x,ROUND(y),255);
        delay(100);
        printf("(%d;%d)\n",x,ROUND(y));
    }
}

//chuong trinh chinh
main()
{
    nhapxy();
    DDA();
    getch();
}
```

DDA Line Algorithm

Câu hỏi 3. Sử dụng thuật toán DDA vẽ đoạn thẳng đi qua 2 điểm A(12;3) và B(2;8)

1.1 Trình bày các bước để thực hiện giải thuật trên

1.2 Lập trình mô phỏng các bước trên với x_A, y_A, x_B, y_B là các số nhập từ bàn phím

Bài làm:

1.1. Từ phương trình đường thẳng đi qua 2 điểm A và B

$$y = mx + b$$

Với $m = \frac{dy}{dx} = \frac{(y_B - y_A)}{(x_B - x_A)} = \frac{8 - 3}{2 - 12} = -0.5$

Và $x_A > x_B, y_A < y_B$

Nên: $x_{i+1} = x_i - 1$ và $y = round(f(x))$

Ta có:

$$y_0 = mx_0 + b$$

Suy ra:

$$y_i = mx_i + b$$

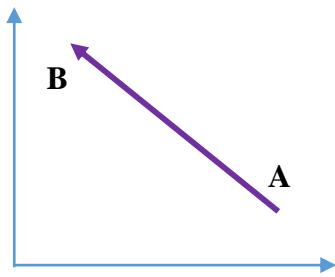
và

$$y_{i+1} = mx_{i+1} + b = m(x_i - 1) + b = mx_i + b - m$$

Vậy

$$y_{i+1} = round(y_i - m)$$

Bước thứ i	x_i	y_i	ROUND
0	12	3	3
1	11	$= 3 + 0.5 = 3.5$	4
2	10	$= 3.5 + 0.5 = 4$	4
3	9	$= 4 + 0.5 = 4.5$	5
4	8	$= 4.5 + 0.5 = 5$	5
5	7	$= 5 + 0.5 = 5.5$	6
6	6	$= 5.5 + 0.5 = 6$	6
7	5	$= 6 + 0.5 = 6.5$	7
8	4	$= 6.5 + 0.5 = 7$	7
9	3	$= 7 + 0.5 = 7.5$	8
10	2	$= 7.5 + 0.5 = 8$	8



1.2. Lập trình mô phỏng

```
//khai bao thu vien
#include <graphics.h>
#include <math.h>
#define ROUND(a) ((int)(a+0.5))

//khai bao bien
int xa;
int xb;
int ya;
int yb;
float m;

//chuong trinh con
void nhapxy()
{
    printf("xA:= ");
    scanf("%d",&xa);
    printf("yA:= ");
    scanf("%d",&ya);
    printf("xB:= ");
    scanf("%d",&xb);
    printf("yB:= ");
    scanf("%d",&yb);
    m=(float) (yb-ya)/(xb-xa);
    printf("he so goc m = %f",m);
}

void DDA()
{
    int x;
    float y;
    initwindow(400,400);
    x=xa;
    y=ya;
    putpixel(x,ROUND(y),125);
    while(x>xb)
    {
        x=x-1;
        y=y-m;
        putpixel(x,ROUND(y),255);
        delay(100);
        printf("(%d;%d)\n",x,ROUND(y));
    }
}

//chuong trinh chinh
main()
{
    nhapxy();
    DDA();
    getch();
}
```

DDA Line Algorithm

Câu hỏi 4. Sử dụng thuật toán DDA vẽ đoạn thẳng đi qua 2 điểm A(2;8) và B(12;3)

1.1 Trình bày các bước để thực hiện giải thuật trên

1.2 Lập trình mô phỏng các bước trên với x_A , y_A , x_B , y_B là các số nhập từ bàn phím

Bài làm:

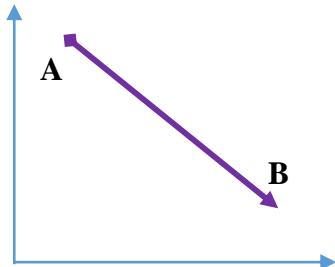
1.1. Từ phương trình đường thẳng đi qua 2 điểm A và B

$$y = mx + b$$

Với $m = \frac{dy}{dx} = \frac{(y_B - y_A)}{(x_B - x_A)} = \frac{3 - 8}{12 - 2} = -0.5$

Và $x_A < x_B$, $y_A < y_B$

Nên: $x_{i+1} = x_i + 1$ và $y = round(f(x))$



Ta có:

$$y_0 = mx_0 + b$$

Suy ra:

$$y_i = mx_i + b$$

và

$$y_{i+1} = mx_{i+1} + b = m(x_i + 1) + b = mx_i + b + m$$

Vậy

$$y_{i+1} = round(y_i + m)$$

Bước thứ i	x_i	y_i	ROUND
0	2	8	8
1	3	= 8 - 0.5 = 7.5	8
2	4	= 7.5 - 0.5 = 7	7
3	5	= 7 - 0.5 = 6.5	7
4	6	= 6.5 - 0.5 = 6	6
5	7	= 6 - 0.5 = 5.5	6
6	8	= 5.5 - 0.5 = 5	5
7	9	= 5 - 0.5 = 4.5	5
8	10	= 4.5 - 0.5 = 4	4
9	11	= 4 - 0.5 = 3.5	4
10	12	= 3.5 - 0.5 = 3	3

1.2. Lập trình mô phỏng

```
//khai bao thu vien
#include <graphics.h>
#include <math.h>
#define ROUND(a) ((int)(a+0.5))

//khai bao bien
int xa;
int xb;
int ya;
int yb;
float m;

//chuong trinh con
void nhapxy()
{
    printf("xA:= ");
    scanf("%d",&xa);
    printf("yA:= ");
    scanf("%d",&ya);
    printf("xB:= ");
    scanf("%d",&xb);
    printf("yB:= ");
    scanf("%d",&yb);
    m=(float) (yb-ya)/(xb-xa);
    printf("he so goc m = %f",m);
}

void DDA()
{
    int x;
    float y;
    initwindow(400,400);
    x=xa;
    y=ya;
    putpixel(x,ROUND(y),125);
    while(x<xb)
    {
        x=x+1;
        y=y+m;
        putpixel(x,ROUND(y),255);
        delay(100);
        printf("(%d;%d)\n",x,ROUND(y));
    }
}

//chuong trinh chinh
main()
{
    nhapxy();
    DDA();
    getch();
}
```

DDA Line Algorithm

Câu hỏi 5. Sử dụng thuật toán DDA vẽ đoạn thẳng đi qua 2 điểm A(3;2) và B(8;12)

1.1 Trình bày các bước để thực hiện giải thuật trên

1.2 Lập trình mô phỏng các bước trên với x_A , y_A , x_B , y_B là các số nhập từ bàn phím

Bài làm:

1.1. Từ phương trình đường thẳng đi qua 2 điểm A và B

$$y = mx + b$$

$$\text{Với } m = \frac{dy}{dx} = \frac{(y_B - y_A)}{(x_B - x_A)} = \frac{12 - 2}{8 - 3} = 2$$

Và $x_A < x_B$, $y_A < y_B$

Nên: $y_{i+1} = y_i + 1$ và $x = round(f(y))$

Ta có:

$$x_0 = \frac{y_0 - b}{m}$$

Suy ra:

$$x_i = \frac{y_i - b}{m}$$

và

$$x_{i+1} = \frac{y_{i+1} - b}{m} = \frac{y_i + 1 - b}{m} = \frac{y_i - b}{m} + \frac{1}{m}$$

Vậy

$$x_{i+1} = round(x_i + \frac{1}{m})$$

Bước thứ i	y_i	x_i	ROUND
0	2	3	3
1	3	= 3 + 0.5 = 3.5	4
2	4	= 3.5 + 0.5 = 4	4
3	5	= 4 + 0.5 = 4.5	5
4	6	= 4.5 + 0.5 = 5	5
5	7	= 5 + 0.5 = 5.5	6
6	8	= 5.5 + 0.5 = 6	6
7	9	= 6 + 0.5 = 6.5	7
8	10	= 6.5 + 0.5 = 7	7
9	11	= 7 + 0.5 = 7.5	8
10	12	= 7.5 + 0.5 = 8	8

1.2. Lập trình mô phỏng

```
//khai bao thu vien
#include <graphics.h>
#include <math.h>
#define ROUND(a) ((int)(a+0.5))

//khai bao bien
int xa;
int xb;
int ya;
int yb;
float m;

//chuong trinh con
void nhapxy()
{
    printf("xA:= ");
    scanf("%d",&xa);
    printf("yA:= ");
    scanf("%d",&ya);
    printf("xB:= ");
    scanf("%d",&xb);
    printf("yB:= ");
    scanf("%d",&yb);
    m=(float) (yb-ya)/(xb-xa);
    printf("he so goc m = %f",m);
}

void DDA()
{
    float x;
    int y;
    initwindow(400,400);
    x=xa;
    y=ya;
    putpixel(ROUND(x),y,125);
    while(y<yb)
    {
        y=y+1;
        x=x+1/m;
        putpixel(ROUND(x),y,255);
        delay(100);
        printf("(%d;%d)\n",ROUND(x),y);
    }
}

//chuong trinh chinh
main()
{
    nhapxy();
    DDA();
    getch();
}
```

DDA Line Algorithm

Câu hỏi 6. Sử dụng thuật toán DDA vẽ đoạn thẳng đi qua 2 điểm A(8;12) và B(3;2)

1.1 Trình bày các bước để thực hiện giải thuật trên

1.2 Lập trình mô phỏng các bước trên với x_A , y_A , x_B , y_B là các số nhập từ bàn phím

Bài làm:

1.1. Từ phương trình đường thẳng đi qua 2 điểm A và B

$$y = mx + b$$

$$\text{Với } m = \frac{dy}{dx} = \frac{(y_B - y_A)}{(x_B - x_A)} = \frac{2 - 12}{3 - 8} = 2$$

Và $x_A > x_B$, $y_A > y_B$

Nên: $y_{i+1} = y_i - 1$ và $x = round(f(y))$

Ta có:

$$x_0 = \frac{y_0 - b}{m}$$

Suy ra:

$$x_i = \frac{y_i - b}{m}$$

và

$$x_{i+1} = \frac{y_{i+1} - b}{m} = \frac{y_i - 1 - b}{m} = \frac{y_i - b}{m} - \frac{1}{m}$$

Vậy

$$x_{i+1} = round(x_i - \frac{1}{m})$$

Bước thứ i	y_i	x_i	ROUND
0	12	8	8
1	11	= 8 - 0.5 = 7.5	8
2	10	= 7.5 - 0.5 = 7	7
3	9	= 7 - 0.5 = 6.5	7
4	8	= 6.5 - 0.5 = 6	6
5	7	= 6 - 0.5 = 5.5	6
6	6	= 5.5 - 0.5 = 5	5
7	5	= 5 - 0.5 = 4.5	5
8	4	= 4.5 - 0.5 = 4	4
9	3	= 3 - 0.5 = 3.5	4
10	2	= 3.5 - 0.5 = 3	3

1.2. Lập trình mô phỏng

```
//khai bao thu vien
#include <graphics.h>
#include <math.h>
#define ROUND(a) ((int)(a+0.5))

//khai bao bien
int xa;
int xb;
int ya;
int yb;
float m;

//chuong trinh con
void nhapxy()
{
    printf("xA:= ");
    scanf("%d",&xa);
    printf("yA:= ");
    scanf("%d",&ya);
    printf("xB:= ");
    scanf("%d",&xb);
    printf("yB:= ");
    scanf("%d",&yb);
    m=(float) (yb-ya)/(xb-xa);
    printf("he so goc m = %f",m);
}

void DDA()
{
    float x;
    int y;
    initwindow(400,400);
    x=xa;
    y=ya;
    putpixel(ROUND(x),y,125);
    while(y>yb)
    {
        y=y-1;
        x=x-1/m;
        putpixel(ROUND(x),y,255);
        delay(100);
        printf("(%d;%d)\n",ROUND(x),y);
    }
}

//chuong trinh chinh
main()
{
    nhapxy();
    DDA();
    getch();
}
```

DDA Line Algorithm

Câu hỏi 7. Sử dụng thuật toán DDA vẽ đoạn thẳng đi qua 2 điểm A(8;2) và B(3;12)

1.1 Trình bày các bước để thực hiện giải thuật trên

1.2 Lập trình mô phỏng các bước trên với x_A , y_A , x_B , y_B là các số nhập từ bàn phím

Bài làm:

1.1. Từ phương trình đường thẳng đi qua 2 điểm A và B

$$y = mx + b$$

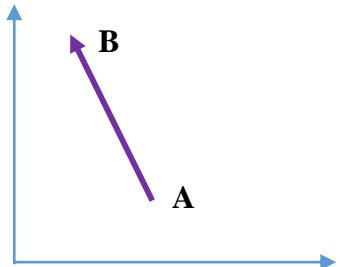
$$\text{Với } m = \frac{dy}{dx} = \frac{(y_B - y_A)}{(x_B - x_A)} = \frac{12 - 2}{3 - 8} = -2$$

Và $x_A > x_B$, $y_A < y_B$

Nên: $y_{i+1} = y_i + 1$ và $x = round(f(y))$

Ta có:

$$x_0 = \frac{y_0 - b}{m}$$



Suy ra:

$$x_i = \frac{y_i - b}{m}$$

và

$$x_{i+1} = \frac{y_{i+1} - b}{m} = \frac{y_i + 1 - b}{m} = \frac{y_i - b}{m} + \frac{1}{m}$$

Vậy

$$x_{i+1} = round\left(x_i + \frac{1}{m}\right)$$

Bước thứ i	y_i	x_i	ROUND
0	2	8	8
1	3	= 8 - 0.5 = 7.5	8
2	4	= 7.5 - 0.5 = 7	7
3	5	= 7 - 0.5 = 6.5	7
4	6	= 6.5 - 0.5 = 6	6
5	7	= 6 - 0.5 = 5.5	6
6	8	= 5.5 - 0.5 = 5	5
7	9	= 5 - 0.5 = 4.5	5
8	10	= 4.5 - 0.5 = 4	4
9	11	= 3 - 0.5 = 3.5	4
10	12	= 3.5 - 0.5 = 3	3

1.2. Lập trình mô phỏng

```
//khai bao thu vien
#include <graphics.h>
#include <math.h>
#define ROUND(a) ((int)(a+0.5))

//khai bao bien
int xa;
int xb;
int ya;
int yb;
float m;

//chuong trinh con
void nhapxy()
{
    printf("xA:= ");
    scanf("%d",&xa);
    printf("yA:= ");
    scanf("%d",&ya);
    printf("xB:= ");
    scanf("%d",&xb);
    printf("yB:= ");
    scanf("%d",&yb);
    m=(float) (yb-ya)/(xb-xa);
    printf("he so goc m = %f",m);
}

void DDA()
{
    float x;
    int y;
    initwindow(400,400);
    x=xa;
    y=ya;
    putpixel(ROUND(x),y,125);
    while(y>yb)
    {
        y=y-1;
        x=x+1/m;
        putpixel(ROUND(x),y,255);
        delay(100);
        printf("(%d;%d)\n",ROUND(x),y);
    }
}

//chuong trinh chinh
main()
{
    nhapxy();
    DDA();
    getch();
}
```

DDA Line Algorithm

Câu hỏi 8. Sử dụng thuật toán DDA vẽ đoạn thẳng đi qua 2 điểm A(3;12) và B(8;2)

1.1 Trình bày các bước để thực hiện giải thuật trên

1.2 Lập trình mô phỏng các bước trên với x_A , y_A , x_B , y_B là các số nhập từ bàn phím

Bài làm:

1.1. Từ phương trình đường thẳng đi qua 2 điểm A và B

$$y = mx + b$$

$$\text{Với } m = \frac{dy}{dx} = \frac{(y_B - y_A)}{(x_B - x_A)} = \frac{2 - 12}{8 - 3} = -2$$

Và $x_A < x_B$, $y_A > y_B$

Nên: $y_{i+1} = y_i - 1$ và $x = round(f(y))$

Ta có:

$$x_0 = \frac{y_0 - b}{m}$$

Suy ra:

$$x_i = \frac{y_i - b}{m}$$

và

$$x_{i+1} = \frac{y_{i+1} - b}{m} = \frac{y_i - 1 - b}{m} = \frac{y_i - b}{m} - \frac{1}{m}$$

Vậy

$$x_{i+1} = round(x_i - \frac{1}{m})$$

Bước thứ i	y_i	x_i	ROUND
0	12	3	3
1	11	= 3 + 0.5 = 3.5	4
2	10	= 3.5 + 0.5 = 4	4
3	9	= 4 + 0.5 = 4.5	5
4	8	= 4.5 + 0.5 = 5	5
5	7	= 5 + 0.5 = 5.5	6
6	6	= 5.5 + 0.5 = 6	6
7	5	= 6 + 0.5 = 6.5	7
8	4	= 6.5 + 0.5 = 7	7
9	3	= 7 + 0.5 = 7.5	8
10	2	= 7.5 + 0.5 = 8	8

1.2. Lập trình mô phỏng

```
//khai bao thu vien
#include <graphics.h>
#include <math.h>
#define ROUND(a) ((int)(a+0.5))

//khai bao bien
int xa;
int xb;
int ya;
int yb;
float m;

//chuong trinh con
void nhapxy()
{
    printf("xA:= ");
    scanf("%d",&xa);
    printf("yA:= ");
    scanf("%d",&ya);
    printf("xB:= ");
    scanf("%d",&xb);
    printf("yB:= ");
    scanf("%d",&yb);
    m=(float) (yb-ya)/(xb-xa);
    printf("he so goc m = %f",m);
}

void DDA()
{
    float x;
    int y;
    initwindow(400,400);
    x=xa;
    y=ya;
    putpixel(ROUND(x),y,125);
    while(y>yb)
    {
        y=y-1;
        x=x-1/m;
        putpixel(ROUND(x),y,255);
        delay(100);
        printf("(%d;%d)\n",ROUND(x),y);
    }
}

//chuong trinh chinh
main()
{
    nhapxy();
    DDA();
    getch();
}
```

Bresenham Line Algorithm

Câu hỏi 1. Sử dụng thuật toán Bresenham vẽ đoạn thẳng đi qua 2 điểm A(2;3) và B(12;8)

1.1. Trình bày các bước để thực hiện giải thuật trên

1.2. Lập trình mô phỏng các bước trên với x_A, y_A, x_B, y_B là các số nhập từ bàn phím

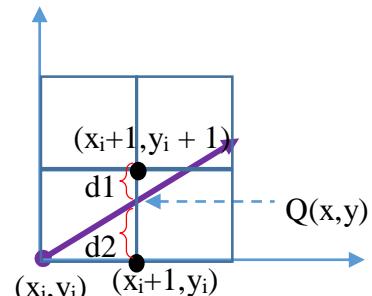
Bài làm:

1.1. Từ phương trình đường thẳng đi qua 2 điểm A và B

$$y = mx + b$$

$$\text{Với } m = \frac{dy}{dx} = \frac{(y_B - y_A)}{(x_B - x_A)} = \frac{8-3}{12-2} = 0.5$$

Đặt:



$$d1 = y_i + 1 - y$$

$$d2 = y - y_i$$

$$\text{pt đường thẳng đi qua } Q: dx(d1-d2) = dx(2y_i - 2y + 1)$$

Xét Q_i thuộc đường thẳng tại thời điểm i:

$$Q_i = dx[2y_i - 2[m(x_i + 1) + b] + 1]$$

$$\Leftrightarrow 2 \cdot dx \cdot y_i - 2 \cdot m \cdot x_i \cdot dx - 2 \cdot m \cdot dx - 2 \cdot b \cdot dx + dx$$

$$\Leftrightarrow 2 \cdot dx \cdot y_i - 2 \cdot \frac{dy}{dx} \cdot x_i \cdot dx - 2 \cdot \frac{dy}{dx} \cdot dx - 2 \cdot b \cdot dx + dx$$

$$\text{Suy ra: } Q_i = 2 \cdot dx \cdot y_i - 2 \cdot dy \cdot x_i - 2 \cdot dy - 2 \cdot b \cdot dx + dx$$

Xét Q_{i+1} là điểm kế tiếp sau Q_i

$$Q_{i+1} = 2 \cdot dx \cdot y_{i+1} - 2 \cdot dy \cdot x_{i+1} - 2 \cdot dy - 2 \cdot b \cdot dx + dx$$

$$\Leftrightarrow Q_{i+1} = 2 \cdot dx \cdot y_{i+1} - 2 \cdot dy \cdot (x_i + 1) - 2 \cdot dy - 2 \cdot b \cdot dx + dx$$

$$\Leftrightarrow Q_{i+1} = 2 \cdot dx \cdot y_{i+1} - 2 \cdot dy \cdot x_i - 2 \cdot dy - 2 \cdot dy - 2 \cdot b \cdot dx + dx$$

$$Q_{i+1} - Q_i = 2 \cdot dx(y_{i+1} - y_i) - 2 \cdot dy$$

Vì $dx = x_B - x_A > 0$

Nếu $d1 \leq d2$ hay $Q_i < 0$: $y_{i+1} = y_i + 1 \Rightarrow Q_{i+1} = Q_i + 2 \cdot dx - 2 \cdot dy$

Ngược lại $Q_i \geq 0$: $y_{i+1} = y_i \Rightarrow Q_{i+1} = Q_i - 2 \cdot dy$

Tại thời điểm ban đầu:

$$Q_0 = 2 \cdot dx \cdot y_0 - 2 \cdot dy \cdot x_0 - 2 \cdot dy - 2 \cdot b \cdot dx + dx$$

Bresenham Line Algorithm

$$\Leftrightarrow Q_0 = 2 \cdot dx \cdot (m \cdot x_0 + b) - 2 \cdot dy \cdot x_0 - 2 \cdot dy - 2 \cdot b \cdot dx + dx$$

$$\Leftrightarrow Q_0 = 2 \cdot dx \cdot m \cdot x_0 + 2 \cdot b \cdot dx - 2 \cdot dy \cdot x_0 - 2 \cdot dy - 2 \cdot b \cdot dx + dx$$

$$\Leftrightarrow Q_0 = 2 \cdot dx \cdot \frac{dy}{dx} \cdot x_0 + 2 \cdot b \cdot dx - 2 \cdot dy \cdot x_0 - 2 \cdot dy - 2 \cdot b \cdot dx + dx$$

Hay $Q_0 = dx - 2 \cdot dy$

- * $Q_0: dx - 2dy = (12-2) - 2(8-3) = 0$
- * $Qi < 0 (y_{i+1} = y_i + 1): Qi + 2dx - 2dy = Qi + 2(12-2) - 2(8-3) = 10$
- * $Qi >= 0 (y_{i+1} = y_i): Qi - 2dy = Qi - 2(8-3) = -10$

Bước thứ i	Công thức Qi	Qi	x_i	y_i
0	$dx - 2dy$	0	2	3
1	$Qi - 2dy$	-10	3	4
2	$Qi + 2dx - 2dy$	10	4	4
3	$Qi - 2dy$	-10	5	5
4	$Qi + 2dx - 2dy$	10	6	5
5	$Qi - 2dy$	-10	7	6
6	$Qi + 2dx - 2dy$	10	8	6
7	$Qi - 2dy$	-10	9	7
8	$Qi + 2dx - 2dy$	10	10	7
9	$Qi - 2dy$	-10	11	8
10	$Qi + 2dx - 2dy$	10	12	8

Bresenham Line Algorithm

1.2. Lập trình mô phỏng

```
//khai bao thu vien
#include <graphics.h>
#include <math.h>
#define ROUND(a) ((int)(a+0.5))

//khai bao bien
int xa;
int xb;
int ya;
int yb;
float m;

//chuong trinh con
void nhapxy()
{
    printf("xA:= ");
    scanf("%d",&xa);
    printf("yA:= ");
    scanf("%d",&ya);
    printf("xB:= ");
    scanf("%d",&xb);
    printf("yB:= ");
    scanf("%d",&yb);
    m=(float) (yb-ya)/(xb-xa);
    printf("he so goc m = %f",m);
}

void Bresenham()
{
    int x;
    int y;
    int q0;
    int q;
    int dy;
    int dx;
    initwindow(400,400);
    putpixel(xa,ya,255);
    dy=yb-ya;
    dx=xb-xa;
    q0=dx-2*dy;
    x=xa;
    y=ya;
    putpixel(x,y,125);
    q=q0;
    while(x<xb)
    {
        if(q<0)
        {
            q=q-2*dy+2*dx;
            y++;
        }
        else
        {
            q=q-2*dy;
        }
        x=x+1;
        putpixel(x,y,255);
        delay(100);
        printf("%d,%d\t",x,y);
    }
}

//chuong trinh chinh
main()
{
    nhapxy();
    Bresenham();
    getch();
}
```

Bresenham Line Algorithm

Câu hỏi 2. Sử dụng thuật toán Bresenham vẽ đoạn thẳng đi qua 2 điểm A(12;8) và B(2;3)

1.1. Trình bày các bước để thực hiện giải thuật trên

1.2 Lập trình mô phỏng các bước trên với x_A, y_A, x_B, y_B là các số nhập từ bàn phím

Bài làm:

1.1. Từ phương trình đường thẳng đi qua 2 điểm A và B

$$y = mx + b$$

$$\text{Với } m = \frac{dy}{dx} = \frac{(y_B - y_A)}{(x_B - x_A)} = \frac{3 - 8}{2 - 12} = 0.5$$

Đặt:

$$d1 = y_i - y$$

$$d2 = y - y_i - 1$$

$$\text{pt đường thẳng đi qua } Q: dx(d1 - d2) = dx(2y_i - 2y + 1)$$

Xét Q_i thuộc đường thẳng tại thời điểm i:

$$Q_i = dx[2y_i - 2[m(x_i - 1) + b] + 1]$$

$$\Leftrightarrow 2 \cdot dx \cdot y_i - 2 \cdot m \cdot x_i \cdot dx + 2 \cdot m \cdot dx - 2 \cdot b \cdot dx + dx$$

$$\Leftrightarrow 2 \cdot dx \cdot y_i - 2 \cdot \frac{dy}{dx} \cdot x_i \cdot dx + 2 \cdot \frac{dy}{dx} \cdot dx - 2 \cdot b \cdot dx + dx$$

$$\text{Suy ra: } Q_i = 2 \cdot dx \cdot y_i - 2 \cdot dy \cdot x_i + 2 \cdot dy - 2 \cdot b \cdot dx + dx$$

Xét Q_{i+1} là điểm kế tiếp sau Q_i

$$Q_{i+1} = 2 \cdot dx \cdot y_{i+1} - 2 \cdot dy \cdot x_{i+1} + 2 \cdot dy - 2 \cdot b \cdot dx + dx$$

$$\Leftrightarrow Q_{i+1} = 2 \cdot dx \cdot y_{i+1} - 2 \cdot dy \cdot (x_i - 1) + 2 \cdot dy - 2 \cdot b \cdot dx + dx$$

$$\Leftrightarrow Q_{i+1} = 2 \cdot dx \cdot y_{i+1} - 2 \cdot dy \cdot x_i + 2 \cdot dy + 2 \cdot dy - 2 \cdot b \cdot dx + dx$$

$$Q_{i+1} - Q_i = 2 \cdot dx(y_{i+1} - y_i) + 2 \cdot dy$$

Vì $dx = x_B - x_A < 0$

Nếu $d1 \leq d2$ hay $Q_i \geq 0$: $y_{i+1} = y_i \Rightarrow Q_{i+1} = Q_i + 2 \cdot dy$

Ngược lại $Q_i < 0$: $y_{i+1} = y_i - 1 \Rightarrow Q_{i+1} = Q_i - 2 \cdot dx + 2 \cdot dy$

Tại thời điểm ban đầu:

$$Q_0 = 2 \cdot dx \cdot y_0 - 2 \cdot dy \cdot x_0 + 2 \cdot dy - 2 \cdot b \cdot dx + dx$$

Bresenham Line Algorithm

$$\Leftrightarrow Q_0 = 2 \cdot dx \cdot (m \cdot x_0 + b) - 2 \cdot dy \cdot x_0 + 2 \cdot dy - 2 \cdot b \cdot dx + dx$$

$$\Leftrightarrow Q_0 = 2 \cdot dx \cdot m \cdot x_0 + 2 \cdot b \cdot dx - 2 \cdot dy \cdot x_0 + 2 \cdot dy - 2 \cdot b \cdot dx + dx$$

$$\Leftrightarrow Q_0 = 2 \cdot dx \cdot \frac{dy}{dx} \cdot x_0 + 2 \cdot b \cdot dx - 2 \cdot dy \cdot x_0 + 2 \cdot dy - 2 \cdot b \cdot dx + dx$$

Hay $Q_0 = dx + 2 \cdot dy$

- * $Q_0: dx + 2dy = (2-12) + 2(3-8) = 0$
- * $Qi >= 0 (y_{i+1} = y_i): 2dy = 2(3-8) = -10$
- * $Qi < 0 (y_{i+1} = y_i - 1): -2dx + 2dy = -2(2-12) + 2(3-8) = 10$

Bước thứ i	Công thức Q_i	Q_i	x_i	y_i
0	$dx + 2dy$	0	2	3
1	$Q_i + 2dy$	-10	3	4
2	$Q_i - 2dx + 2dy$	10	4	4
3	$Q_i + 2dy$	-10	5	5
4	$Q_i - 2dx + 2dy$	10	6	5
5	$Q_i + 2dy$	10	7	6
6	$Q_i - 2dx + 2dy$	-10	8	6
7	$Q_i + 2dy$	10	9	7
8	$Q_i - 2dx + 2dy$	-10	10	7
9	$Q_i + 2dy$	10	11	8
10	$Q_i - 2dx + 2dy$	-10	12	8

Bresenham Line Algorithm

1.2. Lập trình mô phỏng

```
//khai bao thu vien
#include <graphics.h>
#include <math.h>
#define ROUND(a) ((int)(a+0.5))

//khai bao bien
int xa;
int xb;
int ya;
int yb;
float m;

//chuong trinh con
void nhapxy()
{
    printf("xA:= ");
    scanf("%d",&xa);
    printf("yA:= ");
    scanf("%d",&ya);
    printf("xB:= ");
    scanf("%d",&xb);
    printf("yB:= ");
    scanf("%d",&yb);
    m=(float) (yb-ya)/(xb-xa);
    printf("he so goc m = %f",m);
}

void Bresenham()
{
    int x;
    int y;
    int q0;
    int q;
    int dy;
    int dx;
    initwindow(400,400);
    putpixel(xa,ya,255);
    dy=yb-ya;
    dx=xb-xa;
    q0=dx-2*dy;
    x=xa;
    y=ya;
    putpixel(x,y,125);
    q=q0;
    while(x>xb)
    {
        if(q<0)
        {
            q=q-2*dy+2*dx;
            y++;
        }
        else
        {
            q=q-2*dy;
            x=x-1;
            putpixel(x,y,255);
            delay(100);
            printf("%d,%d\t",x,y);
        }
    }
}

//chuong trinh chinh
main()
{
    nhapxy();
    Bresenham();
    getch();
}
```

Bresenham Line Algorithm

Câu hỏi 3. Sử dụng thuật toán Bresenham vẽ đoạn thẳng đi qua 2 điểm A(12;3) và B(2;8)

1.1 Trình bày các bước để thực hiện giải thuật trên

1.2 Lập trình mô phỏng các bước trên với x_A, y_A, x_B, y_B là các số nhập từ bàn phím

Bài làm:

1.1. Từ phương trình đường thẳng đi qua 2 điểm A và B

$$y = mx + b$$

$$\text{Với } m = \frac{dy}{dx} = \frac{(y_B - y_A)}{(x_B - x_A)} = \frac{8 - 3}{2 - 12} = -0.5$$

Đặt:

$$d_1 = y_i + 1 - y$$

$$d_2 = y - y_i$$

$$\text{pt đường thẳng đi qua } Q: dx(d_1 - d_2) = dx(2y_i - 2y + 1)$$

Xét Q_i thuộc đường thẳng tại thời điểm i:

$$Q_i = dx[2y_i - 2[m(x_i - 1) + b] + 1]$$

$$\Leftrightarrow 2 \cdot dx \cdot y_i - 2 \cdot m \cdot x_i \cdot dx + 2 \cdot m \cdot dx - 2 \cdot b \cdot dx + dx$$

$$\Leftrightarrow 2 \cdot dx \cdot y_i - 2 \cdot \frac{dy}{dx} \cdot x_i \cdot dx + 2 \cdot \frac{dy}{dx} \cdot dx - 2 \cdot b \cdot dx + dx$$

$$\text{Suy ra: } Q_i = 2 \cdot dx \cdot y_i - 2 \cdot dy \cdot x_i + 2 \cdot dy - 2 \cdot b \cdot dx + dx$$

Xét Q_{i+1} là điểm kế tiếp sau Q_i

$$Q_{i+1} = 2 \cdot dx \cdot y_{i+1} - 2 \cdot dy \cdot x_{i+1} + 2 \cdot dy - 2 \cdot b \cdot dx + dx$$

$$\Leftrightarrow Q_{i+1} = 2 \cdot dx \cdot y_{i+1} - 2 \cdot dy \cdot (x_i - 1) + 2 \cdot dy - 2 \cdot b \cdot dx + dx$$

$$\Leftrightarrow Q_{i+1} = 2 \cdot dx \cdot y_{i+1} - 2 \cdot dy \cdot x_i + 2 \cdot dy + 2 \cdot dy - 2 \cdot b \cdot dx + dx$$

$$Q_{i+1} - Q_i = 2 \cdot dx(y_{i+1} - y_i) + 2 \cdot dy$$

Vì $dx = x_B - x_A < 0$

Nếu $d_1 \leq d_2$ hay $Q_i \geq 0$: $y_{i+1} = y_i + 1 \Rightarrow Q_{i+1} = Q_i + 2 \cdot dx + 2 \cdot dy$

Ngược lại $Q_i < 0$: $y_{i+1} = y_i \Rightarrow Q_{i+1} = Q_i + 2 \cdot dy$

Tại thời điểm ban đầu:

$$Q_0 = 2 \cdot dx \cdot y_0 - 2 \cdot dy \cdot x_0 + 2 \cdot dy - 2 \cdot b \cdot dx + dx$$

Bresenham Line Algorithm

$$\Leftrightarrow Q_0 = 2 \cdot dx \cdot (m \cdot x_0 + b) - 2 \cdot dy \cdot x_0 + 2 \cdot dy - 2 \cdot b \cdot dx + dx$$

$$\Leftrightarrow Q_0 = 2 \cdot dx \cdot m \cdot x_0 + 2 \cdot b \cdot dx - 2 \cdot dy \cdot x_0 + 2 \cdot dy - 2 \cdot b \cdot dx + dx$$

$$\Leftrightarrow Q_0 = 2 \cdot dx \cdot \frac{dy}{dx} \cdot x_0 + 2 \cdot b \cdot dx - 2 \cdot dy \cdot x_0 + 2 \cdot dy - 2 \cdot b \cdot dx + dx$$

Hay $Q_0 = dx + 2 \cdot dy$

- * $Q_0: dx + 2dy = (2-12) + 2(8-3) = 0$
- * $Qi >= 0 (y_{i+1} = y_i): 2dx + 2dy = 2(8-3) + 2(2-12) = -10$
- * $Qi < 0 (y_{i+1} = y_i + 1): 2dy = 2(8-3) = 10$

Bước thứ i	Công thức Q_i	Q_i	x_i	y_i
0	$dx + 2dy$	0	12	3
1	$Q_i + 2dx + 2dy$	-10	11	3
2	$Q_i + 2dy$	0	10	4
3	$Q_i + 2dx + 2dy$	-10	9	4
4	$Q_i + 2dy$	0	8	5
5	$Q_i + 2dx + 2dy$	-10	7	5
6	$Q_i + 2dy$	0	6	6
7	$Q_i + 2dx + 2dy$	-10	5	6
8	$Q_i + 2dy$	0	4	7
9	$Q_i + 2dx + 2dy$	-10	3	7
10	$Q_i + 2dy$	0	2	8

Bresenham Line Algorithm

1.2. Lập trình mô phỏng

```
//khai bao thu vien
#include <graphics.h>
#include <math.h>
#define ROUND(a) ((int)(a+0.5))

//khai bao bien
int xa;
int xb;
int ya;
int yb;
float m;

//chuong trinh con
void nhapxy()
{
    printf("xA:= ");
    scanf("%d",&xa);
    printf("yA:= ");
    scanf("%d",&ya);
    printf("xB:= ");
    scanf("%d",&xb);
    printf("yB:= ");
    scanf("%d",&yb);
    m=(float) (yb-ya)/(xb-xa);
    printf("he so goc m = %f",m);
}

void Bresenham()
{
    int x;
    int y;
    int q0;
    int q;
    int dy;
    int dx;
    initwindow(400,400);
    putpixel(xa,ya,255);
    dy=yb-ya;
    dx=xb-xa;
    q0=dx+2*dy;
    x=xa;
    y=ya;
    putpixel(x,y,125);
    q=q0;
    while(x>xb)
    {
        if(q<=0)
        {
            q=q+2*dy;
            y++;
        }
        else
        {
            q=q+2*dx+2*dy;
            x=x-1;
            putpixel(x,y,255);
            delay(100);
            printf("%d,%d\t",x,y);
        }
    }
}

//chuong trinh chinh
main()
{
    nhapxy();
    Bresenham();
    getch();
}
```

Bresenham Line Algorithm

Câu hỏi 4. Sử dụng thuật toán Bresenham vẽ đoạn thẳng đi qua 2 điểm A(2;8) và B(12;3)

1.1 Trình bày các bước để thực hiện giải thuật trên

1.2 Lập trình mô phỏng các bước trên với x_A, y_A, x_B, y_B là các số nhập từ bàn phím

Bài làm:

1.1. Từ phương trình đường thẳng đi qua 2 điểm A và B

$$y = mx + b$$

$$\text{Với } m = \frac{dy}{dx} = \frac{(y_B - y_A)}{(x_B - x_A)} = \frac{3 - 8}{12 - 2} = -0.5$$

Đặt:

$$d1 = y_i - y$$

$$d2 = y - y_i + 1$$

pt đường thẳng đi qua Q: $dx(d1 - d2) = dx(2y_i - 2y - 1)$

Xét Q_i thuộc đường thẳng tại thời điểm i:

$$Q_i = dx[2y_i - 2[m(x_i + 1) + b] - 1]$$

$$\Leftrightarrow 2 \cdot dx \cdot y_i - 2 \cdot m \cdot x_i \cdot dx - 2 \cdot m \cdot dx - 2 \cdot b \cdot dx - dx$$

$$\Leftrightarrow 2 \cdot dx \cdot y_i - 2 \cdot \frac{dy}{dx} \cdot x_i \cdot dx - 2 \cdot \frac{dy}{dx} \cdot dx - 2 \cdot b \cdot dx - dx$$

Suy ra: $Q_i = 2 \cdot dx \cdot y_i - 2 \cdot dy \cdot x_i - 2 \cdot dy - 2 \cdot b \cdot dx - dx$

Xét Q_{i+1} là điểm kế tiếp sau Q_i

$$Q_{i+1} = 2 \cdot dx \cdot y_{i+1} - 2 \cdot dy \cdot x_{i+1} - 2 \cdot dy - 2 \cdot b \cdot dx - dx$$

$$\Leftrightarrow Q_{i+1} = 2 \cdot dx \cdot y_{i+1} - 2 \cdot dy \cdot (x_i + 1) - 2 \cdot dy - 2 \cdot b \cdot dx - dx$$

$$\Leftrightarrow Q_{i+1} = 2 \cdot dx \cdot y_{i+1} - 2 \cdot dy \cdot x_i - 2 \cdot dy - 2 \cdot dy - 2 \cdot b \cdot dx - dx$$

$$Q_{i+1} - Q_i = 2 \cdot dx(y_{i+1} - y_i) - 2 \cdot dy$$

Vì $dx = x_B - x_A > 0$

Nếu $d1 \leq d2$ hay $Q_i \leq 0$: $y_{i+1} = y_i \Rightarrow Q_{i+1} = Q_i - 2 \cdot dy$

Ngược lại $Q_i > 0$: $y_{i+1} = y_i - 1 \Rightarrow Q_{i+1} = Q_i - 2 \cdot dx - 2 \cdot dy$

Tại thời điểm ban đầu:

$$Q_0 = 2 \cdot dx \cdot y_0 - 2 \cdot dy \cdot x_0 - 2 \cdot dy - 2 \cdot b \cdot dx - dx$$

Bresenham Line Algorithm

$$\Leftrightarrow Q_0 = 2 \cdot dx \cdot (m \cdot x_0 + b) - 2 \cdot dy \cdot x_0 - 2 \cdot dy - 2 \cdot b \cdot dx - dx$$

$$\Leftrightarrow Q_0 = 2 \cdot dx \cdot m \cdot x_0 + 2 \cdot b \cdot dx - 2 \cdot dy \cdot x_0 - 2 \cdot dy - 2 \cdot b \cdot dx - dx$$

$$\Leftrightarrow Q_0 = 2 \cdot dx \cdot \frac{dy}{dx} \cdot x_0 + 2 \cdot b \cdot dx - 2 \cdot dy \cdot x_0 - 2 \cdot dy - 2 \cdot b \cdot dx - dx$$

Hay $Q_0 = -dx - 2 \cdot dy$

- * $Q_0: -dx - 2dy = -(12-2) - 2(3-8) = 0$
- * $Qi <= 0 (y_{i+1} = y_i): -2dx = -2(3-8) = 10$
- * $Qi > 0 (y_{i+1} = y_i + 1): -2dx - 2dy = -2(3-8) - 2(12-2) = -10$

Bước thứ i	Công thức Qi	Qi	x_i	y_i
0	$dx + 2dy$	0	2	8
1	$Qi - 2dx$	10	3	8
2	$Qi - 2dx - 2dy$	0	4	7
3	$Qi - 2dx$	10	5	7
4	$Qi - 2dx - 2dy$	0	6	6
5	$Qi - 2dx$	10	7	6
6	$Qi - 2dx - 2dy$	0	8	5
7	$Qi - 2dx$	10	9	5
8	$Qi - 2dx - 2dy$	0	10	4
9	$Qi - 2dx$	10	11	4
10	$Qi - 2dx - 2dy$	0	12	3

Bresenham Line Algorithm

1.2. Lập trình mô phỏng

```
//khai bao thu vien
#include <graphics.h>
#include <math.h>
#define ROUND(a) ((int)(a+0.5))

//khai bao bien
int xa;
int xb;
int ya;
int yb;
float m;

//chuong trinh con
void nhapxy()
{
    printf("xA:= ");
    scanf("%d",&xa);
    printf("yA:= ");
    scanf("%d",&ya);
    printf("xB:= ");
    scanf("%d",&xb);
    printf("yB:= ");
    scanf("%d",&yb);
    m=(float) (yb-ya)/(xb-xa);
    printf("he so goc m = %f",m);
}

void Bresenham()
{
    int x;
    int y;
    int q0;
    int q;
    int dy;
    int dx;
    initwindow(400,400);
    putpixel(xa,ya,255);
    dy=yb-ya;
    dx=xb-xa;
    q0=-dx-2*dy;
    x=xa;
    y=ya;
    putpixel(x,y,125);
    q=q0;
    while(x<xb)
    {
        if(q<=0)
        {
            q=q-2*dx;
        }
        else
        {
            q=q-2*dx-2*dy;
            y--;
        }
        x=x+1;
        putpixel(x,y,255);
        delay(100);
        printf("%d,%d\t",x,y);
    }
}

//chuong trinh chinh
main()
{
    nhapxy();
    Bresenham();
    getch();
}
```

Bresenham Line Algorithm

Câu hỏi 5. Sử dụng thuật toán Bresenham vẽ đoạn thẳng đi qua 2 điểm A(3;2) và B(8;12)

1.1 Trình bày các bước để thực hiện giải thuật trên

1.2 Lập trình mô phỏng các bước trên với x_A, y_A, x_B, y_B là các số nhập từ bàn phím

Bài làm:

1.1. Từ phương trình đường thẳng đi qua 2 điểm A và B

$$y = mx + b$$

$$\text{Với } m = \frac{dy}{dx} = \frac{(y_B - y_A)}{(x_B - x_A)} = \frac{12 - 2}{8 - 3} = 2$$

Đặt:

$$d1 = x - x_i$$

$$d2 = x_i + 1 - x$$

pt đường thẳng đi qua Q: $dy(d1-d2) = dy(2x - 2x_i - 1)$

Xét Q_i thuộc đường thẳng tại thời điểm i:

$$\begin{aligned} Q_i &= dy[2(\frac{y_i + 1 - b}{m}) - 2x_i - 1] \\ &\Leftrightarrow \frac{2 \cdot dy \cdot y_i}{m} + \frac{2 \cdot dy}{m} - \frac{2 \cdot b \cdot dy}{m} - 2 \cdot x_i \cdot dy - dy \\ &\Leftrightarrow 2 \cdot dx \cdot y_i + 2 \cdot dx - 2 \cdot b \cdot dx - 2 \cdot x_i \cdot dy - dy \end{aligned}$$

Suy ra: $Q_i = 2 \cdot dx \cdot y_i - 2 \cdot dy \cdot x_i + 2 \cdot dx - 2 \cdot b \cdot dx - dy$

Xét Q_{i+1} là điểm kế tiếp sau Q_i

$$\begin{aligned} Q_{i+1} &= 2 \cdot dx \cdot y_{i+1} - 2 \cdot dy \cdot x_{i+1} - dy - 2 \cdot b \cdot dx + 2 \cdot dx \\ &\Leftrightarrow Q_{i+1} = 2 \cdot dx \cdot (y_i + 1) - 2 \cdot dy \cdot x_{i+1} - dy - 2 \cdot b \cdot dx + 2 \cdot dx \\ &\Leftrightarrow Q_{i+1} = 2 \cdot dx \cdot y_i + 2 \cdot dx - 2 \cdot dy \cdot x_{i+1} - dy - 2 \cdot b \cdot dx + 2 \cdot dx \end{aligned}$$

$$Q_{i+1} - Q_i = -2 \cdot dy(x_{i+1} - x_i) + 2 \cdot dx$$

Vì $dy = yB - yA > 0$

Nếu $d1 \leq d2$ hay $Q_i \leq 0$: $x_{i+1} = x_i \Rightarrow Q_{i+1} = Q_i + 2 \cdot dx$

Ngược lại $Q_i > 0$: $x_{i+1} = x_i + 1 \Rightarrow Q_{i+1} = Q_i - 2 \cdot dy + 2 \cdot dx$

Tại thời điểm ban đầu:

Bresenham Line Algorithm

$$Q_0 = 2 \cdot dx \cdot y_0 - 2 \cdot dy \cdot x_0 + 2 \cdot dx - 2 \cdot b \cdot dx - dy$$

$$\Leftrightarrow Q_0 = 2 \cdot dx \cdot (m \cdot x_0 + b) - 2 \cdot dy \cdot x_0 + 2 \cdot dx - 2 \cdot b \cdot dx - dy$$

$$\Leftrightarrow Q_0 = 2 \cdot dx \cdot m \cdot x_0 + 2 \cdot b \cdot dx - 2 \cdot dy \cdot x_0 + 2 \cdot dx - 2 \cdot b \cdot dx - dy$$

$$\Leftrightarrow Q_0 = 2 \cdot dx \cdot \frac{dy}{dx} \cdot x_0 + 2 \cdot b \cdot dx - 2 \cdot dy \cdot x_0 + 2 \cdot dx - 2 \cdot b \cdot dx - dy$$

Hay $Q_0 = 2 \cdot dx - dy$

- * $Q_0: 2dx - dy = 2(8-3) - (12-2) = 0$
- * $Qi \leq 0 (x_{i+1} = x_i): 2dx = 2(8-3) = 10$
- * $Qi > 0 (x_{i+1} = x_i + 1): -2dy + 2dx = -2(12-2) + 2(3-8) = -10$

Bước thứ i	Công thức Q_i	Q_i	y_i	x_i
0	$dx + 2dy$	0	2	3
1	$Q_i + 2dx$	10	3	4
2	$Q_i - 2dy + 2dx$	0	4	4
3	$Q_i + 2dx$	10	5	5
4	$Q_i - 2dy + 2dy$	0	6	5
5	$Q_i + 2dx$	10	7	6
6	$Q_i - 2dy + 2dy$	0	8	6
7	$Q_i + 2dx$	10	9	7
8	$Q_i - 2dy + 2dy$	0	10	7
9	$Q_i + 2dx$	10	11	8
10	$Q_i - 2dy - 2dx$	0	12	8

Bresenham Line Algorithm

1.2. Lập trình mô phỏng

```
//khai bao thu vien
#include <graphics.h>
#include <math.h>
#define ROUND(a) ((int)(a+0.5))

//khai bao bien
int xa;
int xb;
int ya;
int yb;
float m;

//chuong trinh con
void nhapxy()
{
    printf("xA:= ");
    scanf("%d",&xa);
    printf("yA:= ");
    scanf("%d",&ya);
    printf("xB:= ");
    scanf("%d",&xb);
    printf("yB:= ");
    scanf("%d",&yb);
    m=(float) (yb-ya)/(xb-xa);
    printf("he so goc m = %f",m);
}

void Bresenham()
{
    int x;
    int y;
    int q0;
    int q;
    int dy;
    int dx;
    initwindow(400,400);
    putpixel(xa,ya,255);
    dy=yb-ya;
    dx=xb-xa;
    q0=2*dx-dy;
    x=xa;
    y=ya;
    putpixel(x,y,125);
    q=q0;
    while(y<xb)
    {
        if(q<=0)
        {
            q=q+2*dx;
        }
        else
        {
            q=q-2*dx+2*dy;
            x++;
        }
        y=y+1;
        putpixel(x,y,255);
        delay(100);
        printf("%d,%d\t",x,y);
    }
}

//chuong trinh chinh
main()
{
    nhapxy();
    Bresenham();
    getch();
}
```

Bresenham Line Algorithm

Câu hỏi 6. Sử dụng thuật toán Bresenham vẽ đoạn thẳng đi qua 2 điểm A(8;12) và B(3;2)

1.1 Trình bày các bước để thực hiện giải thuật trên

1.2 Lập trình mô phỏng các bước trên với x_A, y_A, x_B, y_B là các số nhập từ bàn phím

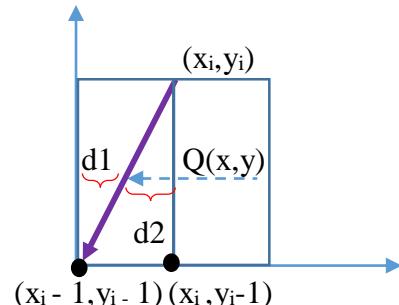
Bài làm:

1.1. Từ phương trình đường thẳng đi qua 2 điểm A và B

$$y = mx + b$$

$$\text{Với } m = \frac{dy}{dx} = \frac{(y_B - y_A)}{(x_B - x_A)} = \frac{2 - 12}{3 - 8} = 2$$

Đặt:



$$d1 = x - (x_i - 1)$$

$$d2 = x_i - x$$

$$\text{pt đường thẳng đi qua } Q: dy(d1-d2) = dy(2x - 2x_i + 1)$$

Xét Q_i thuộc đường thẳng tại thời điểm i:

$$\begin{aligned} Q_i &= dy[2(\frac{y_i - 1 - b}{m}) - 2x_i + 1] \\ &\Leftrightarrow \frac{2 \cdot dy \cdot y_i}{m} - \frac{2 \cdot dy}{m} - \frac{2 \cdot b \cdot dy}{m} - 2 \cdot x_i \cdot dy + dy \\ &\Leftrightarrow 2 \cdot dx \cdot y_i - 2 \cdot dx - 2 \cdot b \cdot dx - 2 \cdot x_i \cdot dy + dy \end{aligned}$$

$$\text{Suy ra: } Q_i = 2 \cdot dx \cdot y_i - 2 \cdot dx - 2 \cdot b \cdot dx - 2 \cdot dy \cdot x_i + dy$$

Xét Q_{i+1} là điểm kế tiếp sau Q_i

$$\begin{aligned} Q_{i+1} &= 2 \cdot dx \cdot y_{i+1} - 2 \cdot dx - 2 \cdot b \cdot dx - 2 \cdot dy \cdot x_{i+1} + dy \\ &\Leftrightarrow Q_{i+1} = 2 \cdot dx \cdot (y_i - 1) - 2 \cdot dx - 2 \cdot b \cdot dx - 2 \cdot dy \cdot x_{i+1} + dy \\ &\Leftrightarrow Q_{i+1} = 2 \cdot dx \cdot y_i - 2 \cdot dx - 2 \cdot dx - 2 \cdot b \cdot dx - 2 \cdot dy \cdot x_{i+1} + dy \end{aligned}$$

$$Q_{i+1} - Q_i = -2 \cdot dy(x_{i+1} - x_i) - 2 \cdot dx$$

$$\text{Vì } dy = y_B - y_A < 0$$

Nếu $d1 \leq d2$ hay $Q_i \geq 0$: $x_{i+1} = x_i - 1 \Rightarrow Q_{i+1} = Q_i + 2 \cdot dy - 2 \cdot dx$

Ngược lại $Q_i < 0$: $x_{i+1} = x_i \Rightarrow Q_{i+1} = Q_i - 2 \cdot dx$

Tại thời điểm ban đầu:

Bresenham Line Algorithm

$$Q_0 = 2 \cdot dx \cdot y_0 - 2 \cdot dx - 2 \cdot b \cdot dx - 2 \cdot dy \cdot x_0 + dy$$

$$\Leftrightarrow Q_0 = 2 \cdot dx \cdot (m \cdot x_0 + b) - 2 \cdot dx - 2 \cdot b \cdot dx - 2 \cdot dy \cdot x_0 + dy$$

$$\Leftrightarrow Q_0 = 2 \cdot dx \cdot m \cdot x_0 + 2 \cdot b \cdot dx - 2 \cdot dx - 2 \cdot b \cdot dx - 2 \cdot dy \cdot x_0 + dy$$

$$\Leftrightarrow Q_0 = 2 \cdot dx \cdot \frac{dy}{dx} \cdot x_0 + 2 \cdot b \cdot dx - 2 \cdot dx - 2 \cdot b \cdot dx - 2 \cdot dy \cdot x_0 + dy$$

Hay $Q_0 = -2 \cdot dx + dy$

- * $Q_0: -2dx + dy = -2(3-8) + (2-12) = 0$
- * $Qi >= 0 (x_{i+1} = x_i - 1): 2dy - 2dx = 2(2-12) - 2(3-8) = -10$
- * $Qi < 0 (x_{i+1} = x_i): -2dx = -2(3-8) = 10$

Bước thứ i	Công thức Q_i	Q_i	y_i	x_i
0	$dy - 2dx$	0	12	8
1	$Q_i + 2dy - 2dx$	-10	11	7
2	$Q_i - 2dx$	10	10	7
3	$Q_i + 2dy - 2dx$	-10	9	6
4	$Q_i - 2dx$	10	8	6
5	$Q_i + 2dy - 2dx$	-10	7	5
6	$Q_i - 2dx$	10	6	5
7	$Q_i + 2dy - 2dx$	-10	5	4
8	$Q_i - 2dx$	10	4	4
9	$Q_i + 2dy - 2dx$	-10	3	3
10	$Q_i - 2dx$	10	2	3

Bresenham Line Algorithm

1.2. Lập trình mô phỏng

```
//khai bao thu vien
#include <graphics.h>
#include <math.h>
#define ROUND(a) ((int)(a+0.5))

//khai bao bien
int xa;
int xb;
int ya;
int yb;
float m;

//chuong trinh con
void nhapxy()
{
    printf("xA:= ");
    scanf("%d",&xa);
    printf("yA:= ");
    scanf("%d",&ya);
    printf("xB:= ");
    scanf("%d",&xb);
    printf("yB:= ");
    scanf("%d",&yb);
    m=(float) (yb-ya)/(xb-xa);
    printf("he so goc m = %f",m);
}

void Bresenham()
{
    int x;
    int y;
    int q0;
    int q;
    int dy;
    int dx;
    initwindow(400,400);
    putpixel(xa,ya,255);
    dy=yb-ya;
    dx=xb-xa;
    q0=dy-2*dx;
    x=xa;
    y=ya;
    putpixel(x,y,125);
    q=q0;
    while(y>yb)
    {
        if(q>=0)
        {
            q=q+2*dy-2*dx;
            x--;
        }
        else
        {
            q=q-2*dx;
            y=y-1;
            putpixel(x,y,255);
            delay(100);
            printf("%d,%d\t",x,y);
        }
    }
}

//chuong trinh chinh
main()
{
    nhapxy();
    Bresenham();
    getch();
}
```

Bresenham Line Algorithm

Câu hỏi 7. Sử dụng thuật toán DDA vẽ đoạn thẳng đi qua 2 điểm A(8;2) và B(3;12)

1.1 Trình bày các bước để thực hiện giải thuật trên

1.2 Lập trình mô phỏng các bước trên với x_A, y_A, x_B, y_B là các số nhập từ bàn phím

Bài làm:

1.1. Từ phương trình đường thẳng đi qua 2 điểm A và B

$$y = mx + b$$

$$\text{Với } m = \frac{dy}{dx} = \frac{(y_B - y_A)}{(x_B - x_A)} = \frac{12 - 2}{3 - 8} = -2$$

Đặt:

$$d1 = x - (x_i - 1)$$

$$d2 = x_i - x$$

$$\text{pt đường thẳng đi qua } Q: dy(d1 - d2) = dy(2x - 2x_i + 1)$$

Xét Q_i thuộc đường thẳng tại thời điểm i:

$$\begin{aligned} Q_i &= dy[2(\frac{y_i + 1 - b}{m}) - 2x_i + 1] \\ &\Leftrightarrow \frac{2 \cdot dy \cdot y_i}{m} + \frac{2 \cdot dy}{m} - \frac{2 \cdot b \cdot dy}{m} - 2 \cdot x_i \cdot dy + dy \\ &\Leftrightarrow 2 \cdot dx \cdot y_i + 2 \cdot dx - 2 \cdot b \cdot dx - 2 \cdot x_i \cdot dy + dy \end{aligned}$$

$$\text{Suy ra: } Q_i = 2 \cdot dx \cdot y_i \mp 2 \cdot dx - 2 \cdot b \cdot dx - 2 \cdot dy \cdot x_i + dy$$

Xét Q_{i+1} là điểm kế tiếp sau Q_i

$$\begin{aligned} Q_{i+1} &= 2 \cdot dx \cdot y_{i+1} + 2 \cdot dx - 2 \cdot b \cdot dx - 2 \cdot dy \cdot x_{i+1} + dy \\ &\Leftrightarrow Q_{i+1} = 2 \cdot dx \cdot (y_i + 1) + 2 \cdot dx - 2 \cdot b \cdot dx - 2 \cdot dy \cdot x_{i+1} + dy \\ &\Leftrightarrow Q_{i+1} = 2 \cdot dx \cdot y_i + 2 \cdot dx + 2 \cdot dx - 2 \cdot b \cdot dx - 2 \cdot dy \cdot x_{i+1} + dy \end{aligned}$$

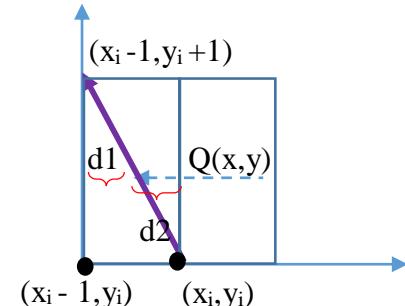
$$Q_{i+1} - Q_i = -2 \cdot dy(x_{i+1} - x_i) + 2 \cdot dx$$

Vì $dy = y_B - y_A > 0$

Nếu $d1 \leq d2$ hay $Q_i \leq 0$: $x_{i+1} = x_i - 1 \Rightarrow Q_{i+1} = Q_i + 2 \cdot dy + 2 \cdot dx$

Ngược lại $Q_i > 0$: $x_{i+1} = x_i \Rightarrow Q_{i+1} = Q_i + 2 \cdot dx$

Tại thời điểm ban đầu:



Bresenham Line Algorithm

$$Q_0 = 2 \cdot dx \cdot y_0 + 2 \cdot dx - 2 \cdot b \cdot dx - 2 \cdot dy \cdot x_0 + dy$$

$$\Leftrightarrow Q_0 = 2 \cdot dx \cdot (m \cdot x_0 + b) + 2 \cdot dx - 2 \cdot b \cdot dx - 2 \cdot dy \cdot x_0 + dy$$

$$\Leftrightarrow Q_0 = 2 \cdot dx \cdot m \cdot x_0 + 2 \cdot b \cdot dx + 2 \cdot dx - 2 \cdot b \cdot dx - 2 \cdot dy \cdot x_0 + dy$$

$$\Leftrightarrow Q_0 = 2 \cdot dx \cdot \frac{dy}{dx} \cdot x_0 + 2 \cdot b \cdot dx + 2 \cdot dx - 2 \cdot b \cdot dx - 2 \cdot dy \cdot x_0 + dy$$

Hay $Q_0 = 2 \cdot dx + dy$

- * $Q_0: 2dx + dy = 2(3-8) + (12-2) = 0$
- * $Qi <= 0 (x_{i+1} = x_i - 1): 2dy + 2dx = 2(12-2) + 2(3-8) = 10$
- * $Qi > 0 (x_{i+1} = x_i): 2dx = 2(3-8) = -10$

Bước thứ i	Công thức Q_i	Q_i	y_i	x_i
0	$2dx + dy$	0	2	8
1	$Q_i + 2dy + 2dx$	10	3	7
2	$Q_i + 2dx$	-10	4	7
3	$Q_i + 2dy + 2dx$	10	5	6
4	$Q_i + 2dx$	-10	6	6
5	$Q_i + 2dy + 2dx$	10	7	5
6	$Q_i + 2dx$	-10	8	5
7	$Q_i + 2dy + 2dx$	10	9	4
8	$Q_i + 2dx$	-10	10	4
9	$Q_i + 2dy + 2dx$	10	11	3
10	$Q_i + 2dx$	-10	12	3

Bresenham Line Algorithm

1.2. Lập trình mô phỏng

```
//khai bao thu vien
#include <graphics.h>
#include <math.h>
#define ROUND(a) ((int)(a+0.5))

//khai bao bien
int xa;
int xb;
int ya;
int yb;
float m;

//chuong trinh con
void nhapxy()
{
    printf("xA:= ");
    scanf("%d",&xa);
    printf("yA:= ");
    scanf("%d",&ya);
    printf("xB:= ");
    scanf("%d",&xb);
    printf("yB:= ");
    scanf("%d",&yb);
    m=(float) (yb-ya)/(xb-xa);
    printf("he so goc m = %f",m);
}

void Bresenham()
{
    int x;
    int y;
    int q0;
    int q;
    int dy;
    int dx;
    initwindow(400,400);
    putpixel(xa,ya,255);
    dy=yb-ya;
    dx=xb-xa;
    q0=dy-2*dx;
    x=xa;
    y=ya;
    putpixel(x,y,125);
    q=q0;
    while(y<yb)
    {
        if(q<=0)
        {
            q=q+2*dy+2*dx;
            x--;
        }
        else
        {
            q=q+2*dx;
            y=y+1;
            putpixel(x,y,255);
            delay(100);
            printf("%d,%d\t",x,y);
        }
    }
}

//chuong trinh chinh
main()
{
    nhapxy();
    Bresenham();
    getch();
}
```

Bresenham Line Algorithm

Câu hỏi 8. Sử dụng thuật toán Bresenham vẽ đoạn thẳng đi qua 2 điểm A(3;12) và B(8;2)

1.1 Trình bày các bước để thực hiện giải thuật trên

1.2 Lập trình mô phỏng các bước trên với x_A, y_A, x_B, y_B là các số nhập từ bàn phím

Bài làm:

1.1. Từ phương trình đường thẳng đi qua 2 điểm A và B

$$y = mx + b$$

$$\text{Với } m = \frac{dy}{dx} = \frac{(y_B - y_A)}{(x_B - x_A)} = \frac{2 - 12}{8 - 3} = -2$$

Đặt:

$$d1 = x - x_i$$

$$d2 = x_i + 1 - x$$

$$\text{pt đường thẳng đi qua Q: } dy(d1 - d2) = dy(2x - 2x_i - 1)$$

Xét Q_i thuộc đường thẳng tại thời điểm i:

$$\begin{aligned} Q_i &= dy[2(\frac{y_i - 1 - b}{m}) - 2x_i - 1] \\ &\Leftrightarrow \frac{2 \cdot dy \cdot y_i}{m} - \frac{2 \cdot dy}{m} - \frac{2 \cdot b \cdot dy}{m} - 2 \cdot x_i \cdot dy - dy \\ &\Leftrightarrow 2 \cdot dx \cdot y_i - 2 \cdot dx - 2 \cdot b \cdot dx - 2 \cdot x_i \cdot dy - dy \end{aligned}$$

$$\text{Suy ra: } Q_i = 2 \cdot dx \cdot y_i - 2 \cdot dx - 2 \cdot b \cdot dx - 2 \cdot dy \cdot x_i - dy$$

Xét Q_{i+1} là điểm kế tiếp sau Q_i

$$\begin{aligned} Q_{i+1} &= 2 \cdot dx \cdot y_{i+1} - 2 \cdot dx - 2 \cdot b \cdot dx - 2 \cdot dy \cdot x_{i+1} - dy \\ &\Leftrightarrow Q_{i+1} = 2 \cdot dx \cdot (y_i - 1) - 2 \cdot dx - 2 \cdot b \cdot dx - 2 \cdot dy \cdot x_{i+1} - dy \\ &\Leftrightarrow Q_{i+1} = 2 \cdot dx \cdot y_i - 2 \cdot dx - 2 \cdot b \cdot dx - 2 \cdot dy \cdot x_{i+1} - dy \end{aligned}$$

$$Q_{i+1} - Q_i = -2 \cdot dy(x_{i+1} - x_i) - 2 \cdot dx$$

Vì $dy = y_B - y_A < 0$

Nếu $d1 \leq d2$ hay $Q_i \geq 0: x_{i+1} = x_i \Rightarrow Q_{i+1} = Q_i - 2 \cdot dx$

Ngược lại $Q_i < 0: x_{i+1} = x_i + 1 \Rightarrow Q_{i+1} = Q_i - 2 \cdot dy - 2 \cdot dx$

Tại thời điểm ban đầu:

Bresenham Line Algorithm

$$Q_0 = 2 \cdot dx \cdot y_0 - 2 \cdot dx - 2 \cdot b \cdot dx - 2 \cdot dy \cdot x_0 - dy$$

$$\Leftrightarrow Q_0 = 2 \cdot dx \cdot (m \cdot x_0 + b) - 2 \cdot dx - 2 \cdot b \cdot dx - 2 \cdot dy \cdot x_0 - dy$$

$$\Leftrightarrow Q_0 = 2 \cdot dx \cdot m \cdot x_0 + 2 \cdot b \cdot dx - 2 \cdot dx - 2 \cdot b \cdot dx - 2 \cdot dy \cdot x_0 - dy$$

$$\Leftrightarrow Q_0 = 2 \cdot dx \cdot \frac{dy}{dx} \cdot x_0 + 2 \cdot b \cdot dx - 2 \cdot dx - 2 \cdot b \cdot dx - 2 \cdot dy \cdot x_0 - dy$$

Hay $Q_0 = -2 \cdot dx - dy$

- * $Q_0: -2dx - dy = -2(8-3) - (2-12) = 0$
- * $Qi >= 0 (x_{i+1} = x_i): -2dx = -2(8-3) = -10$
- * $Qi < 0 (x_{i+1} = x_i + 1): -2dy - 2dx = -2(2-12) - 2(8-3) = 10$

Bước thứ i	Công thức Q_i	Q_i	y_i	x_i
0	-2dx-dy	0	12	3
1	$Q_i - 2dx$	-10	11	3
2	$Q_i - 2dy - 2dx$	10	10	4
3	$Q_i - 2dx$	-10	9	4
4	$Q_i - 2dy - 2dx$	10	8	5
5	$Q_i - 2dx$	-10	7	5
6	$Q_i - 2dy - 2dx$	10	6	6
7	$Q_i - 2dx$	-10	5	6
8	$Q_i - 2dy - 2dx$	10	4	7
9	$Q_i - 2dx$	-10	3	7
10	$Q_i - 2dy - 2dx$	10	2	8

Bresenham Line Algorithm

1.2. Lập trình mô phỏng

```
//khai bao thu vien
#include <graphics.h>
#include <math.h>
#define ROUND(a) ((int)(a+0.5))

//khai bao bien
int xa;
int xb;
int ya;
int yb;
float m;

//chuong trinh con
void nhapxy()
{
    printf("xA:= ");
    scanf("%d",&xa);
    printf("yA:= ");
    scanf("%d",&ya);
    printf("xB:= ");
    scanf("%d",&xb);
    printf("yB:= ");
    scanf("%d",&yb);
    m=(float) (yb-ya)/(xb-xa);
    printf("he so goc m = %f",m);
}

void Bresenham()
{
    int x;
    int y;
    int q0;
    int q;
    int dy;
    int dx;
    initwindow(400,400);
    putpixel(xa,ya,255);
    dy=yb-ya;
    dx=xb-xa;
    q0=-2*dx-dy;
    x=xa;
    y=ya;
    putpixel(x,y,125);
    q=q0;
    while(y>yb)
    {
        if(q>=0)
        {
            q=q-2*dx;
        }
        else
        {
            q=q-2*dy-2*dx;
            x++;
        }
        y=y-1;
        putpixel(x,y,255);
        delay(100);
        printf("%d,%d\t",x,y);
    }
}

//chuong trinh chinh
main()
{
    nhapxy();
    Bresenham();
    getch();
}
```

Midpoint Line Algorithm

Câu hỏi 1. Sử dụng thuật toán Midpoint vẽ đoạn thẳng đi qua 2 điểm A(2;3) và B(12;8)

1.1. Trình bày các bước để thực hiện giải thuật trên

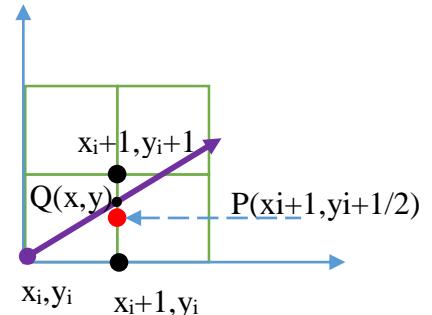
1.2. Lập trình mô phỏng các bước trên với x_A, y_A, x_B, y_B là các số nhập từ bàn phím

Bài làm:

1.3. Từ phương trình đường thẳng đi qua 2 điểm A và B

$$y = Ax + By + C$$

Với $m = \frac{dy}{dx} = \frac{(y_B - y_A)}{(x_B - x_A)} = \frac{8-3}{12-2} = 0.5$



Phương trình đường thẳng đi qua điểm Midpoint tại thời điểm i:

$$P_i = Ax_i + By_i + C$$

Với $A = dy = (y_B - y_A) = (8-3) = 5$

$B = -dx = -(x_B - x_A) = -(12-2) = -10$

Suy ra:

$$P_i = A(x_i + 1) + B(y_i + \frac{1}{2}) + C$$

$$P_i = Ax_i + A + By_i + \frac{B}{2} + C$$

Phương trình đường thẳng đi qua điểm Midpoint tại thời điểm $i+1$ là thời điểm liền kề i:

$$P_{i+1} = Ax_{i+1} + By_{i+1} + C$$

Suy ra:

$$P_{i+1} = A(x_i + 1) + A + B(y_{i+1} + \frac{1}{2}) + C$$

$$P_{i+1} = Ax_i + A + A + By_{i+1} + \frac{B}{2} + C$$

Đặt $P_{i+1} - P_i = A + B(y_{i+1} - y_i)$

Hay $P_{i+1} = P_i + A + B(y_{i+1} - y_i)$

Vậy:

$$P_{i+1} = P_i + A + B, \text{ nếu } P_i \geq 0 \text{ hay } y_{i+1} = y_i + 1$$

Midpoint Line Algorithm

$$P_{i+1} = P_i + A, \text{ nếu } P_i < 0 \text{ hay } y_{i+1} = y_i$$

Xác định P_0

$$P_i = Ax_0 + A + By_0 + \frac{B}{2} + C$$

Với

$$Ax_0 + By_0 + C = 0$$

Nên

$$P_0 = A + \frac{B}{2}$$

$A = 5$

$B = -10$

$P_0 = 0$

Bước thứ i	x_i	P_i	y_i
0	2	$P_0 = A+B/2 = 0$	$y_0 = 3$
1	3	$P_1 = P_0+A+B = -5$	$y_1 = y_0 + 1 = 4$
2	4	$P_2 = P_1+A = 0$	$y_2 = y_1 = 4$
3	5	$P_3 = P_2+A+B = -5$	$y_3 = y_2 + 1 = 5$
4	6	$P_4 = P_3 + A = 0$	$y_4 = y_3 = 5$
5	7	$P_5 = P_4 + A + B = -5$	$y_5 = y_4 + 1 = 6$
6	8	$P_6 = P_5 + A = 0$	$y_6 = y_5 = 6$
7	9	$P_7 = P_6 + A + B = -5$	$y_7 = y_6 + 1 = 7$
8	10	$P_8 = P_7 + A = 0$	$y_8 = y_7 = 7$
9	11	$P_9 = P_8 + A + B = -5$	$y_9 = y_8 + 1 = 8$
10	12	$P_{10} = P_9 + A = 0$	$y_9 = y_8 = 8$

1.2. Lập trình mô phỏng

```

//khai bao thu vien
#include <graphics.h>
#include <math.h>
#define ROUND(a) ((int)(a+0.5))

//khai bao bien
int xa;
int xb;
int ya;
int yb;
float m;

//chuong trinh con
void nhapxy()
{
    printf("xA:= ");
    scanf("%d",&xa);
    printf("yA:= ");
    scanf("%d",&ya);
    printf("xB:= ");
    scanf("%d",&xb);
    printf("yB:= ");
    scanf("%d",&yb);
    m=(float) (yb-ya)/(xb-xa);
    printf("he so goc m = %f",m);
}

void Midpoint()
{
    int x;
    int y;
    float p0;
    float p;
    int A;
    int B;
    initwindow(400,400);
    putpixel(xa,ya,255);
    x=xa;
    y=ya;
    putpixel(x,y,125);
    //xac dinh cac he so
    A=yb-ya;
    B=-(xb-xa);
    p0= (float) (A+B/2);
    p=p0;
    while(x<xb)
    {
        if(p<0)
        {
            p=(float)(p+A);
        }
        else
        {
            p=(float)(p+A+B);
            y++;
        }
        x=x+1;
        putpixel(x,y,255);
        delay(100);
        printf("%d,%d\t",x,y);
    }
    //closegraph();
}

```

Midpoint Line Algorithm

Câu hỏi 2. Sử dụng thuật toán Midpoint vẽ đoạn thẳng đi qua 2 điểm A(12;8) và B(2;3)

1.1. Trình bày các bước để thực hiện giải thuật trên

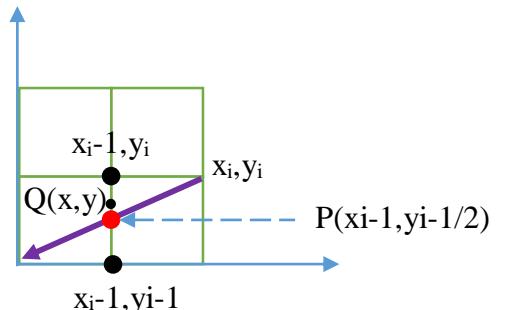
1.2 Lập trình mô phỏng các bước trên với x_A, y_A, x_B, y_B là các số nhập từ bàn phím

Bài làm:

1.4. Từ phương trình đường thẳng đi qua 2 điểm A và B

$$y = Ax + By + C$$

Với $m = \frac{dy}{dx} = \frac{(y_B - y_A)}{(x_B - x_A)} = \frac{3-8}{2-12} = 0.5$



Phương trình đường thẳng đi qua điểm Midpoint tại thời điểm i:

$$P_i = Ax_i + By_i + C$$

Với $A = dy = (y_B - y_A) = (3-8) = -5$

$B = -dx = -(x_B - x_A) = -(2-12) = 10$

Suy ra:

$$P_i = A(x_i - 1) + B(y_i - \frac{1}{2}) + C$$

$$P_i = Ax_i - A + By_i - \frac{B}{2} + C$$

Phương trình đường thẳng đi qua điểm Midpoint tại thời điểm $i+1$ là thời điểm liền kề i:

$$P_{i+1} = Ax_{i+1} + By_{i+1} + C$$

Suy ra:

$$P_{i+1} = A(x_i - 1) - A + B(y_{i+1} - \frac{1}{2}) + C$$

$$P_{i+1} = Ax_i - A - A + By_{i+1} - \frac{B}{2} + C$$

Đặt $P_{i+1} - P_i = -A + B(y_{i+1} - y_i)$

Hay $P_{i+1} = P_i - A + B(y_{i+1} - y_i)$

Vậy:

$$P_{i+1} = P_i - A - B, \text{ nếu } P_i \geq 0 \text{ hay } y_{i+1} = y_i - 1$$

$$P_{i+1} = P_i - A, \text{ nếu } P_i < 0 \text{ hay } y_{i+1} = y_i$$

Midpoint Line Algorithm

Xác định P_0

$$P_i = Ax_0 - A + By_0 - \frac{B}{2} + C$$

Với

$$Ax_0 + By_0 + C = 0$$

Nên

$$P_0 = -A - \frac{B}{2}$$

$$A = -5$$

$$B = 10$$

$$P_0 = 0$$

Bước thứ i	xi	P_i	yi
0	12	0	8
1	11	-5	7
2	10	0	7
3	9	-5	6
4	8	0	6
5	7	-5	5
6	6	0	5
7	5	-5	4
8	4	0	4
9	3	-5	3
10	2	0	3

1.2. Lập trình mô phỏng

```

//khai bao thu vien
#include <graphics.h>
#include <math.h>
#define ROUND(a) ((int)(a+0.5))

//khai bao bien
int xa;
int xb;
int ya;
int yb;
float m;

//chuong trinh con
void nhapxy()
{
    printf("xA:= ");
    scanf("%d",&xa);
    printf("yA:= ");
    scanf("%d",&ya);
    printf("xB:= ");
    scanf("%d",&xb);
    printf("yB:= ");
    scanf("%d",&yb);
    m=(float) (yb-ya)/(xb-xa);
    printf("he so goc m = %f",m);
}

void Midpoint()
{
    int x;
    int y;
    float p0;
    float p;
    int A;
    int B;
    initwindow(400,400);
    putpixel(xa,ya,255);
    x=xa;
    y=ya;
    putpixel(x,y,125);
    //xac dinh cac he so
    A=yb-ya;
    B=-(xb-xa);
    p0= (float) (-A-B/2);
    p=p0;
    while(x>xb)
    {
        if(p<0)
        {
            p=(float)(p-A);
        }
        else
        {
            p=(float)(p-A-B);
            y--;
        }
        x=x-1;
        putpixel(x,y,255);
        delay(100);
        printf("%d,%d\t",x,y);
    }
    //closegraph();
}

```

Midpoint Line Algorithm

Câu hỏi 3. Sử dụng thuật toán Midpoint vẽ đoạn thẳng đi qua 2 điểm A(12;3) và B(2;8)

1.1 Trình bày các bước để thực hiện giải thuật trên

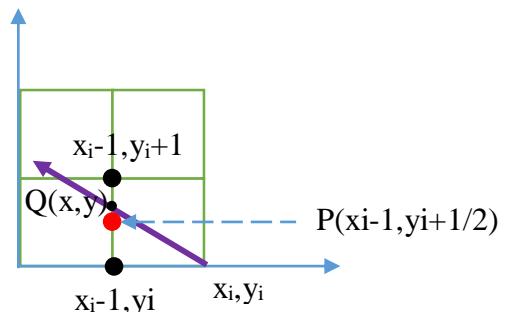
1.2 Lập trình mô phỏng các bước trên với x_A, y_A, x_B, y_B là các số nhập từ bàn phím

Bài làm:

1.5. Từ phương trình đường thẳng đi qua 2 điểm A và B

$$y = Ax + By + C$$

Với $m = \frac{dy}{dx} = \frac{(y_B - y_A)}{(x_B - x_A)} = \frac{8-3}{2-12} = -0.5$



Phương trình đường thẳng đi qua điểm Midpoint tại thời điểm i:

$$P_i = Ax_i + By_i + C$$

Với $A = dy = (y_B - y_A) = (8-3) = 5$

$B = -dx = -(x_B - x_A) = -(2-12) = 10$

Suy ra:

$$P_i = A(x_i - 1) + B(y_i + \frac{1}{2}) + C$$

$$P_i = Ax_i - A + By_i + \frac{B}{2} + C$$

Phương trình đường thẳng đi qua điểm Midpoint tại thời điểm $i+1$ là thời điểm liền kề i:

$$P_{i+1} = Ax_{i+1} + By_{i+1} + C$$

Suy ra:

$$P_{i+1} = A(x_i - 1) - A + B(y_{i+1} + \frac{1}{2}) + C$$

$$P_{i+1} = Ax_i - A - A + By_{i+1} + \frac{B}{2} + C$$

Đặt $P_{i+1} - P_i = -A + B(y_{i+1} - y_i)$

Hay $P_{i+1} = P_i - A + B(y_{i+1} - y_i)$

Vậy:

$$P_{i+1} = P_i - A + B, \text{ nếu } P_i < 0 \text{ hay } y_{i+1} = y_i + 1$$

$$P_{i+1} = P_i - A, \text{ nếu } P_i \geq 0 \text{ hay } y_{i+1} = y_i$$

Midpoint Line Algorithm

Xác định P_0

$$P_i = Ax_0 - A + By_0 + \frac{B}{2} + C$$

Với

$$Ax_0 + By_0 + C = 0$$

Nên

$$P_0 = -A + \frac{B}{2}$$

A = 5

B = 10

P₀ = 0

Bước thứ i	xi	Pi	yi
0	12	0	3
1	11	-5	3
2	10	0	4
3	9	-5	4
4	8	0	5
5	7	-5	5
6	6	0	6
7	5	-5	6
8	4	0	7
9	3	-5	7
10	2	0	8

Midpoint Line Algorithm

1.2. Lập trình mô phỏng

```
//khai bao thu vien
#include <graphics.h>
#include <math.h>
#define ROUND(a) ((int)(a+0.5))

//khai bao bien
int xa;
int xb;
int ya;
int yb;
float m;

//chuong trinh con
void nhapxy()
{
    printf("xA:= ");
    scanf("%d",&xa);
    printf("yA:= ");
    scanf("%d",&ya);
    printf("xB:= ");
    scanf("%d",&xb);
    printf("yB:= ");
    scanf("%d",&yb);
    m=(float) (yb-ya)/(xb-xa);
    printf("he so goc m = %f",m);
}

void Midpoint()
{
    int x;
    int y;
    float p0;
    float p;
    int A;
    int B;
    initwindow(400,400);
    putpixel(xa,ya,255);
    x=xa;
    y=ya;
    putpixel(x,y,125);
    //xac dinh cac he so
    A=yb-ya;
    B=-(xb-xa);
    p0= (float) (-A-B/2);
    p=p0;
    while(x<xb)
    {
        if(p>=0)
        {
            p=(float)(p-A);
        }
        else
        {
            p=(float)(p-A+B);
            y++;
        }
        x=x+1;
        putpixel(x,y,255);
        delay(100);
        printf("%d,%d\t",x,y);
    }
    //closegraph();
}
```

Midpoint Line Algorithm

Câu hỏi 4. Sử dụng thuật toán Midpoint vẽ đoạn thẳng đi qua 2 điểm A(2;8) và B(12;3)

1.1 Trình bày các bước để thực hiện giải thuật trên

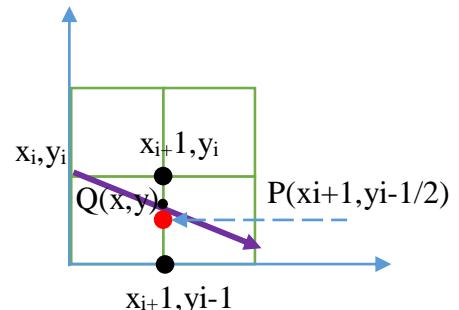
1.2 Lập trình mô phỏng các bước trên với x_A, y_A, x_B, y_B là các số nhập từ bàn phím

Bài làm:

1.1. Từ phương trình đường thẳng đi qua 2 điểm A và B

$$y = Ax + By + C$$

Với $m = \frac{dy}{dx} = \frac{(y_B - y_A)}{(x_B - x_A)} = \frac{3-8}{12-2} = -0.5$



Phương trình đường thẳng đi qua điểm Midpoint tại thời điểm i:

$$P_i = Ax_i + By_i + C$$

Với $A = dy = (y_B - y_A) = (3-8) = -5$

$B = -dx = -(x_B - x_A) = (2-12) = -10$

Suy ra:

$$P_i = A(x_i + 1) + B(y_i - \frac{1}{2}) + C$$

$$P_i = Ax_i + A + By_i - \frac{B}{2} + C$$

Phương trình đường thẳng đi qua điểm Midpoint tại thời điểm $i+1$ là thời điểm liền kề i:

$$P_{i+1} = Ax_{i+1} + By_{i+1} + C$$

Suy ra:

$$P_{i+1} = A(x_i + 1) + A + B(y_{i+1} - \frac{1}{2}) + C$$

$$P_{i+1} = Ax_i + A + A + By_{i+1} - \frac{B}{2} + C$$

Đặt $P_{i+1} - P_i = A + B(y_{i+1} - y_i)$

Hay $P_{i+1} = P_i + A + B(y_{i+1} - y_i)$

Vậy:

$$P_{i+1} = P_i + A - B, \text{ nếu } P_i < 0 \text{ hay } y_{i+1} = y_i - 1$$

$$P_{i+1} = P_i + A, \text{ nếu } P_i \geq 0 \text{ hay } y_{i+1} = y_i$$

Midpoint Line Algorithm

Xác định P_0

$$P_i = Ax_0 + A + By_0 - \frac{B}{2} + C$$

Với

$$Ax_0 + By_0 + C = 0$$

Nên

$$P_0 = A - \frac{B}{2}$$

$$A = -5$$

$$B = -10$$

$$P_0 = 0$$

Bước thứ i	xi	P_i	yi
0	2	0	8
1	3	-5	8
2	4	0	7
3	5	-5	7
4	6	0	6
5	7	-5	6
6	8	0	5
7	9	-5	5
8	10	0	4
9	11	-5	4
10	12	0	3

Midpoint Line Algorithm

1.2.Lập trình mô phỏng

```
//khai bao thu vien
#include <graphics.h>
#include <math.h>
#define ROUND(a) ((int)(a+0.5))

//khai bao bien
int xa;
int xb;
int ya;
int yb;
float m;

//chuong trinh con
void nhapxy()
{
    printf("xA:= ");
    scanf("%d",&xa);
    printf("yA:= ");
    scanf("%d",&ya);
    printf("xB:= ");
    scanf("%d",&xb);
    printf("yB:= ");
    scanf("%d",&yb);
    m=(float) (yb-ya)/(xb-xa);
    printf("he so goc m = %f",m);
}

void Midpoint()
{
    int x;
    int y;
    float p0;
    float p;
    int A;
    int B;
    initwindow(400,400);
    putpixel(xa,ya,255);
    x=xa;
    y=ya;
    putpixel(x,y,125);
    //xac dinh cac he so
    A=yb-ya;
    B=-(xb-xa);
    p0= (float) (-A-B/2);
    p=p0;
    while(x<xb)
    {
        if(p>0)
        {
            p=(float)(p+A);
        }
        else
        {
            p=(float)(p+A-B);
            y--;
        }
        x=x+1;
        putpixel(x,y,255);
        delay(100);
        printf("%d,%d\t",x,y);
    }
    //closegraph();
}
```

Midpoint Line Algorithm

Câu hỏi 5. Sử dụng thuật toán Midpoint vẽ đoạn thẳng đi qua 2 điểm A(3;2) và B(8;12)

1.1 Trình bày các bước để thực hiện giải thuật trên

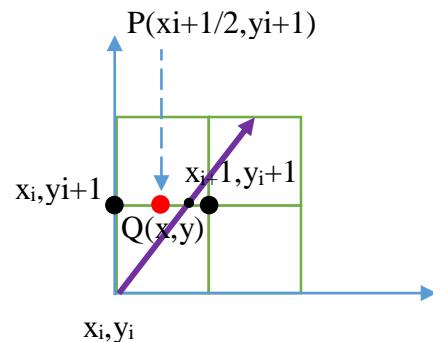
1.2 Lập trình mô phỏng các bước trên với x_A, y_A, x_B, y_B là các số nhập từ bàn phím

Bài làm:

1.1. Từ phương trình đường thẳng đi qua 2 điểm A và B

$$y = Ax + By + C$$

$$\text{Với } m = \frac{dy}{dx} = \frac{(y_B - y_A)}{(x_B - x_A)} = \frac{12 - 2}{8 - 3} = 2$$



Phương trình đường thẳng đi qua điểm Midpoint tại thời điểm i:

$$P_i = Ax_i + By_i + C$$

$$\text{Với } A = dy = (y_B - y_A) = (12 - 2) = 10$$

$$B = -dx = -(x_B - x_A) = -(8 - 3) = -5$$

Suy ra:

$$P_i = A\left(x_i + \frac{1}{2}\right) + B(y_i + 1) + C$$

$$P_i = Ax_i + \frac{A}{2} + By_i + B + C$$

Phương trình đường thẳng đi qua điểm Midpoint tại thời điểm $i+1$ là thời điểm liền kề i:

$$P_{i+1} = Ax_{i+1} + By_{i+1} + C$$

Suy ra:

$$P_{i+1} = A\left(x_{i+1} + \frac{1}{2}\right) + B(y_i + 1) + B + C$$

$$P_{i+1} = Ax_{i+1} + \frac{A}{2} + By_i + B + B + C$$

$$\text{Đặt } P_{i+1} - P_i = A(x_{i+1} - x_i) + B$$

$$\text{Hay } P_{i+1} = P_i + A(x_{i+1} - x_i) + B$$

Vậy:

$$P_{i+1} = P_i + A + B, \text{ nếu } P_i < 0 \text{ hay } x_{i+1} = x_i + 1$$

$$P_{i+1} = P_i + B, \text{ nếu } P_i \geq 0 \text{ hay } x_{i+1} = x_i$$

Midpoint Line Algorithm

Xác định P_0

$$P_i = Ax_0 + \frac{A}{2} + By_0 + B + C$$

Với

$$Ax_0 + By_0 + C = 0$$

Nên

$$P_0 = \frac{A}{2} + B$$

A = 10

B = -5

P₀ = 0

Bước thứ i	y _i	P _i	x _i
0	2	0	3
1	3	-5	3
2	4	0	4
3	5	-5	4
4	6	0	5
5	7	-5	5
6	8	0	6
7	9	-5	6
8	10	0	7
9	11	-5	7
10	12	0	8

Midpoint Line Algorithm

1.2. Lập trình mô phỏng

```
//khai bao thu vien
#include <graphics.h>
#include <math.h>
#define ROUND(a) ((int)(a+0.5))

//khai bao bien
int xa;
int xb;
int ya;
int yb;
float m;

//chuong trinh con
void nhapxy()
{
    printf("xA:= ");
    scanf("%d",&xa);
    printf("yA:= ");
    scanf("%d",&ya);
    printf("xB:= ");
    scanf("%d",&xb);
    printf("yB:= ");
    scanf("%d",&yb);
    m=(float) (yb-ya)/(xb-xa);
    printf("he so goc m = %f",m);
}

void Midpoint()
{
    int x;
    int y;
    float p0;
    float p;
    int A;
    int B;
    initwindow(400,400);
    putpixel(xa,ya,255);
    x=xa;
    y=ya;
    putpixel(x,y,125);
    //xac dinh cac he so
    A=yb-ya;
    B=-(xb-xa);
    p0= (float) (A/2+B);
    p=p0;
    while(y<yb)
    {
        if(p>0)
        {
            p=(float)(p+B);
        }
        else
        {
            p=(float)(p+A+B);
            x++;
        }
        y=y+1;
        putpixel(x,y,255);
        delay(100);
        printf("%d,%d\t",x,y);
    }
    //closegraph();
}
```

Midpoint Line Algorithm

Câu hỏi 6. Sử dụng thuật toán Midpoint vẽ đoạn thẳng đi qua 2 điểm A(8;12) và B(3;2)

1.1 Trình bày các bước để thực hiện giải thuật trên

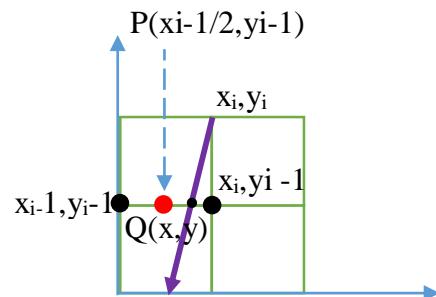
1.2 Lập trình mô phỏng các bước trên với x_A, y_A, x_B, y_B là các số nhập từ bàn phím

Bài làm:

1.1. Từ phương trình đường thẳng đi qua 2 điểm A và B

$$y = Ax + By + C$$

Với $m = \frac{dy}{dx} = \frac{(y_B - y_A)}{(x_B - x_A)} = \frac{2 - 12}{8 - 3} = -2$



Phương trình đường thẳng đi qua điểm Midpoint tại thời điểm i:

$$P_i = Ax_i + By_i + C$$

Với $A = dy = (y_B - y_A) = (2 - 12) = -10$

$B = -dx = -(x_B - x_A) = -(3 - 8) = 5$

Suy ra:

$$P_i = A(x_i - \frac{1}{2}) + B(y_i - 1) + C$$

$$P_i = Ax_i - \frac{A}{2} + By_i - B + C$$

Phương trình đường thẳng đi qua điểm Midpoint tại thời điểm $i+1$ là thời điểm liền kề i:

$$P_{i+1} = Ax_{i+1} + By_{i+1} + C$$

Suy ra:

$$P_{i+1} = A(x_{i+1} - \frac{1}{2}) + B(y_i - 1) - B + C$$

$$P_{i+1} = Ax_{i+1} - \frac{A}{2} + By_i - B - B + C$$

Đặt $P_{i+1} - P_i = A(x_{i+1} - x_i) - B$

Hay $P_{i+1} = P_i + A(x_{i+1} - x_i) - B$

Vậy:

$$P_{i+1} = P_i - A - B, \text{ nếu } P_i < 0 \text{ hay } x_{i+1} = x_i - 1$$

$$P_{i+1} = P_i - B, \text{ nếu } P_i \geq 0 \text{ hay } x_{i+1} = x_i$$

Midpoint Line Algorithm

Xác định P_0

$$P_i = Ax_0 - \frac{A}{2} + By_0 - B + C$$

Với

$$Ax_0 + By_0 + C = 0$$

Nên

$$P_0 = -\frac{A}{2} - B$$

A = -10

B = 5

P₀ = 0

Bước thứ i	y _i	P _i	x _i
0	12	0	8
1	11	-5	8
2	10	0	7
3	9	-5	7
4	8	0	6
5	7	-5	6
6	6	0	5
7	5	-5	5
8	4	0	4
9	3	-5	4
10	2	0	3

Midpoint Line Algorithm

1.2. Lập trình mô phỏng

```
//khai bao thu vien
#include <graphics.h>
#include <math.h>
#define ROUND(a) ((int)(a+0.5))

//khai bao bien
int xa;
int xb;
int ya;
int yb;
float m;

//chuong trinh con
void nhapxy()
{
    printf("xA:= ");
    scanf("%d",&xa);
    printf("yA:= ");
    scanf("%d",&ya);
    printf("xB:= ");
    scanf("%d",&xb);
    printf("yB:= ");
    scanf("%d",&yb);
    m=(float) (yb-ya)/(xb-xa);
    printf("he so goc m = %f",m);
}

void Midpoint()
{
    int x;
    int y;
    float p0;
    float p;
    int A;
    int B;
    initwindow(400,400);
    putpixel(xa,ya,255);
    x=xa;
    y=ya;
    putpixel(x,y,125);
    //xac dinh cac he so
    A=yb-ya;
    B=-(xb-xa);
    p0= (float) (-A/2-B);
    p=p0;
    while(y>yb)
    {
        if(p>0)
        {
            p=(float)(p-B);
        }
        else
        {
            p=(float)(p-A-B);
            x--;
        }
        y=y-1;
        putpixel(x,y,255);
        delay(100);
        printf("%d,%d\t",x,y);
    }
    //closegraph();
}
```

Midpoint Line Algorithm

Câu hỏi 7. Sử dụng thuật toán Midpoint vẽ đoạn thẳng đi qua 2 điểm A(8;2) và B(3;12)

1.1 Trình bày các bước để thực hiện giải thuật trên

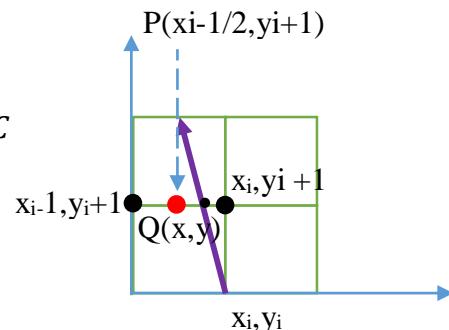
1.2 Lập trình mô phỏng các bước trên với x_A, y_A, x_B, y_B là các số nhập từ bàn phím

Bài làm:

1.1. Từ phương trình đường thẳng đi qua 2 điểm A và B

$$y = Ax + By + C$$

$$\text{Với } m = \frac{dy}{dx} = \frac{(y_B - y_A)}{(x_B - x_A)} = \frac{12 - 2}{3 - 8} = -2$$



Phương trình đường thẳng đi qua điểm Midpoint tại thời điểm i:

$$P_i = Ax_i + By_i + C$$

$$\text{Với } A = dy = (y_B - y_A) = (12 - 2) = 10$$

$$B = -dx = -(x_B - x_A) = -(3 - 8) = 5$$

Suy ra:

$$P_i = A\left(x_i - \frac{1}{2}\right) + B(y_i + 1) + C$$

$$P_i = Ax_i - \frac{A}{2} + By_i + B + C$$

Phương trình đường thẳng đi qua điểm Midpoint tại thời điểm $i+1$ là thời điểm liền kề i:

$$P_{i+1} = Ax_{i+1} + By_{i+1} + C$$

Suy ra:

$$P_{i+1} = A\left(x_{i+1} - \frac{1}{2}\right) + B(y_i + 1) + B + C$$

$$P_{i+1} = Ax_{i+1} - \frac{A}{2} + By_i + B + B + C$$

$$\text{Đặt } P_{i+1} - P_i = A(x_{i+1} - x_i) + B$$

$$\text{Hay } P_{i+1} = P_i + A(x_{i+1} - x_i) + B$$

Vậy:

$$P_{i+1} = P_i - A + B, \text{ nếu } P_i \geq 0 \text{ hay } x_{i+1} = x_i - 1$$

$$P_{i+1} = P_i + B, \text{ nếu } P_i < 0 \text{ hay } x_{i+1} = x_i$$

Midpoint Line Algorithm

Xác định P_0

$$P_i = Ax_0 - \frac{A}{2} + By_0 + B + C$$

Với

$$Ax_0 + By_0 + C = 0$$

Nên

$$P_0 = -\frac{A}{2} + B$$

A = 10

B = 5

P₀ = 0

Bước thứ i	y _i	P _i	x _i
0	2	0	8
1	3	-5	7
2	4	0	7
3	5	-5	6
4	6	0	6
5	7	-5	5
6	8	0	5
7	9	-5	4
8	10	0	4
9	11	-5	3
10	12	0	3

Midpoint Line Algorithm

1.2. Lập trình mô phỏng

```
//khai bao thu vien
#include <graphics.h>
#include <math.h>
#define ROUND(a) ((int)(a+0.5))

//khai bao bien
int xa;
int xb;
int ya;
int yb;
float m;

//chuong trinh con
void nhapxy()
{
    printf("xA:= ");
    scanf("%d",&xa);
    printf("yA:= ");
    scanf("%d",&ya);
    printf("xB:= ");
    scanf("%d",&xb);
    printf("yB:= ");
    scanf("%d",&yb);
    m=(float) (yb-ya)/(xb-xa);
    printf("he so goc m = %f",m);
}

void Midpoint()
{
    int x;
    int y;
    float p0;
    float p;
    int A;
    int B;
    initwindow(400,400);
    putpixel(xa,ya,255);
    x=xa;
    y=ya;
    putpixel(x,y,125);
    //xac dinh cac he so
    A=yb-ya;
    B=-(xb-xa);
    p0= (float) (-A/2+B);
    p=p0;
    while(y<yb)
    {
        if(p<0)
        {
            p=(float)(p+B);
        }
        else
        {
            p=(float)(p-A+B);
            x--;
        }
        y=y+1;
        putpixel(x,y,255);
        delay(100);
        printf("%d,%d\t",x,y);
    }
    //closegraph();
}
```

Midpoint Line Algorithm

Câu hỏi 8. Sử dụng thuật toán Midpoint vẽ đoạn thẳng đi qua 2 điểm A(3;12) và B(8;2)

1.1 Trình bày các bước để thực hiện giải thuật trên

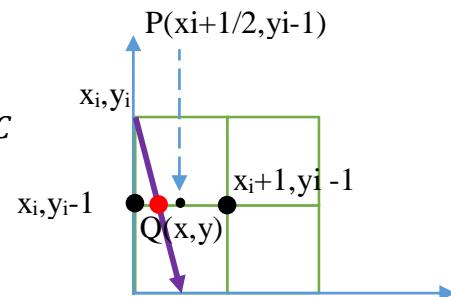
1.2 Lập trình mô phỏng các bước trên với x_A, y_A, x_B, y_B là các số nhập từ bàn phím

Bài làm:

1.1. Từ phương trình đường thẳng đi qua 2 điểm A và B

$$y = Ax + By + C$$

Với $m = \frac{dy}{dx} = \frac{(y_B - y_A)}{(x_B - x_A)} = \frac{2 - 12}{8 - 3} = -2$



Phương trình đường thẳng đi qua điểm Midpoint tại thời điểm i:

$$P_i = Ax_i + By_i + C$$

Với $A = dy = (y_B - y_A) = (2 - 12) = -10$

$B = -dx = -(x_B - x_A) = -(8 - 3) = -5$

Suy ra:

$$P_i = A(x_i + \frac{1}{2}) + B(y_i - 1) + C$$

$$P_i = Ax_i + \frac{A}{2} + By_i - B + C$$

Phương trình đường thẳng đi qua điểm Midpoint tại thời điểm $i+1$ là thời điểm liền kề i:

$$P_{i+1} = Ax_{i+1} + By_{i+1} + C$$

Suy ra:

$$P_{i+1} = A(x_{i+1} + \frac{1}{2}) + B(y_i - 1) - B + C$$

$$P_{i+1} = Ax_{i+1} + \frac{A}{2} + By_i - B - B + C$$

Đặt $P_{i+1} - P_i = A(x_{i+1} - x_i) - B$

Hay $P_{i+1} = P_i + A(x_{i+1} - x_i) - B$

Vậy:

$$P_{i+1} = P_i + A - B, \text{ nếu } P_i \geq 0 \text{ hay } x_{i+1} = x_i + 1$$

$$P_{i+1} = P_i - B, \text{ nếu } P_i < 0 \text{ hay } x_{i+1} = x_i$$

Midpoint Line Algorithm

Xác định P_0

$$P_i = Ax_0 + \frac{A}{2} + By_0 - B + C$$

Với

$$Ax_0 + By_0 + C = 0$$

Nên

$$P_0 = \frac{A}{2} - B$$

A = -10

B = -5

P₀ = 0

Bước thứ i	y _i	P _i	x _i
0	12	0	3
1	11	-5	4
2	10	0	4
3	9	-5	5
4	8	0	5
5	7	-5	6
6	6	0	6
7	5	-5	7
8	4	0	7
9	3	-5	8
10	2	0	8

Midpoint Line Algorithm

1.2. Lập trình mô phỏng

```
//khai bao thu vien
#include <graphics.h>
#include <math.h>
#define ROUND(a) ((int)(a+0.5))

//khai bao bien
int xa;
int xb;
int ya;
int yb;
float m;

//chuong trinh con
void nhapxy()
{
    printf("xA:= ");
    scanf("%d",&xa);
    printf("yA:= ");
    scanf("%d",&ya);
    printf("xB:= ");
    scanf("%d",&xb);
    printf("yB:= ");
    scanf("%d",&yb);
    m=(float) (yb-ya)/(xb-xa);
    printf("he so goc m = %f",m);
}

void Midpoint()
{
    int x;
    int y;
    float p0;
    float p;
    int A;
    int B;
    initwindow(400,400);
    putpixel(xa,ya,255);
    x=xa;
    y=ya;
    putpixel(x,y,125);
    //xac dinh cac he so
    A=yb-ya;
    B=-(xb-xa);
    p0= (float) (-A/2+B);
    p=p0;
    while(y>yb)
    {
        if(p<0)
        {
            p=(float)(p-B);
        }
        else
        {
            p=(float)(p+A-B);
            x++;
        }
        y=y-1;
        putpixel(x,y,255);
        delay(100);
        printf("%d,%d\t",x,y);
    }
    //closegraph();
}
```

Circle Bresenham

Câu hỏi. Sử dụng thuật toán Bresenham vẽ đường tròn tâm O(0;0) và R là 5

1.1 Trình bày các bước để thực hiện giải thuật trên

1.2 Lập trình mô phỏng các bước trên với xO, yO, và R là các số nhập từ bàn phím

Ta chia đường tròn là làm 8 phần, do tính chất đối xứng của nó.

Nên khi ta tính toán được 1 điểm có tọa độ x,y đồng nghĩa với việc ta xác định đầy đủ 8 điểm cho 8 phần như sau:

putpixel (x_c+x , y_c+y)

putpixel (x_c+x , y_c-y)

putpixel (x_c+y , y_c-x)

putpixel (x_c-y , y_c-x)

putpixel (x_c-x , y_c-y)

putpixel (x_c-x , y_c+y)

putpixel (x_c-y , y_c+x)

putpixel (x_c+y , y_c+x)

Trong đó: x_c , y_c là tâm của đường tròn

Ta xét các điểm tạo ra từ góc phần tư thứ 2: từ 45° đến 90° , thực hiện theo hướng x, y

Đặt $d_i = f(N) + f(S)$

Áp dụng định lý pithagoras: $x^2 + y^2 = r^2$ cho f(N) và f(S)

Cụ thể:

$$f(N) = (x+1)^2 + y^2 - r^2$$

$$f(S) = (x-1)^2 + (y-1)^2 - r^2$$

Hay $d_i = [(x_i + 1)^2 + y_i^2 - r^2] + [(x_i - 1)^2 + (y_i - 1)^2 - r^2]$

$$d_i = [2(x_i + 1)^2 + y_i^2 + (y_i - 1)^2 - 2r^2]$$

$$d_{i+1} = [2(x_{i+1} + 1)^2 + y_{i+1}^2 + (y_{i+1} - 1)^2 - 2r^2]$$

$$d_{i+1} - d_i = 2[(x_{i+1} + 1)^2 - (x_i + 1)^2] + y_{i+1}^2 - y_i^2 + (y_{i+1} - 1)^2 - (y_i - 1)^2$$

$$\text{Hay } d_{i+1} = d_i + 2[(x_i + 1 + 1)^2 - (x_i + 1)^2] + y_{i+1}^2 - y_i^2 + (y_{i+1} - 1)^2 - (y_i - 1)^2$$

$$d_{i+1} = d_i + 2(2x_i + 3) + y_{i+1}^2 - y_i^2 + (y_{i+1} - 1)^2 - (y_i - 1)^2$$

Nếu $d_i < 0$, $x_{i+1} = x_i + 1$ và $y_{i+1} = y_i$

$$d_{i+1} = d_i + 2(2x_i + 3) = d_i + 4x_i + 6$$

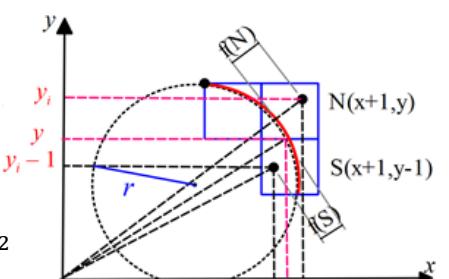
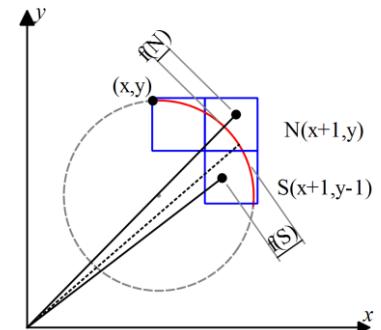
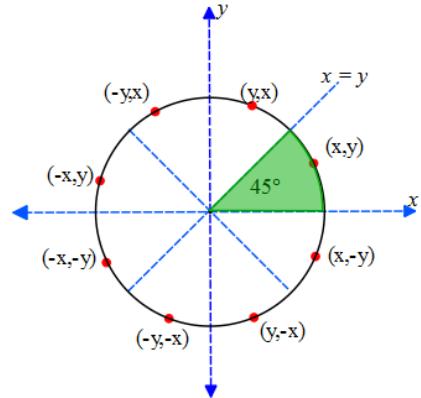
Ngược lại: $x_{i+1} = x_i + 1$ và $y_{i+1} = y_i - 1$

$$d_{i+1} = d_i + 2(2x_i + 3) + (y_i - 1)^2 - y_i^2 + (y_i - 2)^2 - (y_i - 1)^2$$

$$d_{i+1} = d_i + 4x_i - 4y_i + 10$$

Xác định d_0

$$\text{Từ } d_i = 2(x_i + 1)^2 + y_i^2 + (y_i - 1)^2 - 2r^2$$



Circle Bresenham

$$d_0 = 2(x_0 + 1)^2 + y_0^2 + (y_0 - 1)^2 - 2r^2$$

$$d_0 = 2(0 + 1)^2 + r^2 + (r - 1)^2 - 2r^2 = 3 - 2r$$

Lập bảng 1/8 đường tròn

$$d_0 = 3 - 2r = 3 - 2.5 = -7$$

Nếu $d_i < 0$, $x_{i+1} = x_i + 1$ và $y_{i+1} = y_i$

$$d_{i+1} = d_i + 4x_i + 6$$

Ngược lại: $x_{i+1} = x_i + 1$ và $y_{i+1} = y_i - 1$

$$d_{i+1} = d_i + 4x_i - 4y_i + 10$$

Bước i	x _i	y _i	d
0	0	5	-7
1	1	5	-7+4.0+6 = -1
2	2	5	-1+4.1+6=9
3	3	4	9+4.2-4.5+10=7
4	4	3	7+4.3-4.4+10=13
5	5	2	13+4.4-4.3+10=25

Circle Bresenham

1.2. Lập trình mô phỏng

```

//khai bao thu vien
#include<stdio.h>
#include <graphics.h>

//khai bao bien
int xc;
int yc;
int x;
int y;
int r;
int d;

//chuong trinh con

void nhapxy()
{
    printf("Nhập tam duong tron, xc= ");
    scanf("%d",&xc);
    printf("Nhập tam duong tron, yc= ");
    scanf("%d",&yc);
    printf("Nhập ban kinh duong tron, r= ");
    scanf("%d",&r);
}

void drawcircle()
{
    putpixel(xc+x,yc+y,255);
    putpixel(xc-x,yc+y,255);
    putpixel(xc+x,yc-y,255);
    putpixel(xc-x,yc-y,255);
    putpixel(xc+y,yc+x,255);
    putpixel(xc-y,yc+x,255);
    putpixel(xc+y,yc-x,255);
    putpixel(xc-y,yc-x,255);
}

void Bresenham()
{
    initwindow(400,400);
    x=0;
    y=r;
    d=3-2*r;
    while(x<y)
    {
        drawcircle();
        if(d<=0)
        {
            d=d+4*x+6;
        }
        else
        {
            y--;
            d=d+4*x-4*y+10;
        }
        drawcircle();
        x++;
        printf("%d,%d\n",x,y);
    }
}

```

Circle Midpoint

Câu hỏi. Sử dụng thuật toán Bresenham vẽ đường tròn tâm O(0;0) và R là 5

1.1 Trình bày các bước để thực hiện giải thuật trên

1.2 Lập trình mô phỏng các bước trên với xO, yO, và R là các số nhập từ bàn phím

Ta chia đường tròn là làm 8 phần, do tính chất đối xứng của nó.

Do tính đối xứng của đường tròn (C) nên ta chỉ cần vẽ cung ($C^{1/8}$) là cung 1/8 đường tròn, sau đó lấy đối xứng.

Cung ($C^{1/8}$) được mô tả như sau (cung của phần tô màu)

$$\text{Đặt } f(x, y) = x^2 + y^2 - r^2$$

$$f(x, y) \begin{cases} < 0, \text{ nếu } (x, y) \text{ nằm phía trong đường tròn} \\ = 0, \text{ nếu } (x, y) \text{ nằm trên đường tròn} \\ > 0, \text{ nếu } (x, y) \text{ nằm phía ngoài đường tròn} \end{cases}$$

$$\text{Xét } p_i = f(\text{Midpoint}) = f(x_i + 1, y_i - \frac{1}{2})$$

Nếu $p_i < 0$, điểm MidPoint nằm trong đường tròn.

Lúc này điểm thực Q gần S hơn nên ta chọn S, tức là $y_{i+1} = y_i$

Ngược lại, nếu $p_i \geq 0$, điểm MidPoint nằm ngoài đường tròn.

Lúc này điểm thực Q gần P hơn nên ta chọn P, tức là $y_{i+1} = y_i - 1$. Mặt khác:

$$\begin{aligned} p_{i+1} - p_i &= f\left(x_{i+1} + 1, y_{i+1} - \frac{1}{2}\right) - f(x_i + 1, y_i - \frac{1}{2}) \\ p_{i+1} - p_i &= \left[(x_{i+1} + 1)^2 + \left(y_{i+1} - \frac{1}{2}\right)^2 - r^2\right] - \left[(x_i + 1)^2 + \left(y_i - \frac{1}{2}\right)^2 - r^2\right] \\ p_{i+1} - p_i &= [(2x_i + 3) + y_{i+1}^2 - y_i^2 - y_{i+1} - y_i] \end{aligned}$$

Do $x_{i+1} = x_i + 1$

Vậy:

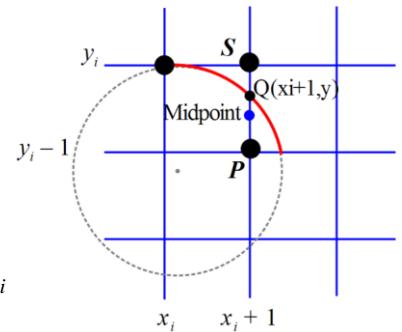
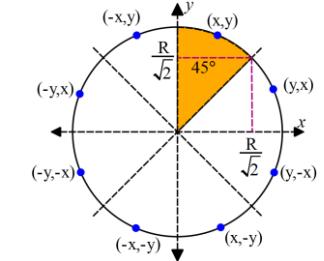
$$p_{i+1} = p_i + (2x_i + 3), \text{ nếu } p_i < 0 \text{ do ta chọn } y_{i+1} = y_i.$$

$$p_{i+1} = p_i + 2x_i - 2y_i + 5, \text{ nếu } p_i \geq 0 \text{ do ta chọn } y_{i+1} = y_i - 1$$

Ta tính giá trị p_0 ứng với điểm ban đầu $(x_0, y_0) = (0, r)$.

$$p_0 = f\left(x_0 + 1, y_0 - \frac{1}{2}\right) = f\left(1, r - \frac{1}{2}\right) = 1 - \left(r - \frac{1}{2}\right)^2 - r = \frac{5}{4} - r$$

Bước i	x_i	y_i	p
0	0	5	$5/4 - 5 = -15/4$
1	1	5	$-15/4 + (2.0+3) = -3/4$
2	2	5	$-3/4 + (2.1+3) = 17/4$
3	3	4	$17/4 + 2.2 - 2.5 + 5 = 3/4$
4	4	3	$3/4 + 2.3 - 2.4 + 5 = 15/4$
5	5	2	$15/4 + 2.4 - 2.3 + 5 = 21/2$



1.2. Lập trình mô phỏng

```

//khai bao thu vien
#include <graphics.h>
#include <stdio.h>
#define ROUND(a) ((int)(a+0.5))
//khai bao bien
int xc;
int yc;
int x;
int y;
int r;
int p;

//chuong trinh con

void nhapxy()
{
    printf("Nhập tam duong tron, xc= ");
    scanf("%d",&xc);
    printf("Nhập tam duong tron, yc= ");
    scanf("%d",&yc);
    printf("Nhập ban kinh duong tron, r= ");
    scanf("%d",&r);
}

void drawcircle()
{
    putpixel(xc+x,yc+y,255);
    putpixel(xc-x,yc+y,255);
    putpixel(xc+x,yc-y,255);
    putpixel(xc-x,yc-y,255);
    putpixel(xc+y,yc+x,255);
    putpixel(xc-y,yc+x,255);
    putpixel(xc+y,yc-x,255);
    putpixel(xc-y,yc-x,255);
}
void Midpoint()
{
    initwindow(400,400);
    x=0;
    y=r;
    p=ROUND(5/4-r);
    while(x<y)
    {
        drawcircle();
        if(p<=0)
        {
            p=p+2*x+3;
        }
        else
        {
            y--;
            p=p+2*x-2*y+5;
        }
        drawcircle();
        x++;
        printf("%d,%d\t",x,y);
    }
}

//chuong trinh chinh
main()
{
    //khai tao window
    nhapxy();
    Midpoint();
    //closegraph();
    getch();
}

```

Circle Midpoint

Tìm công thức của phương trình đường tròn:

Trong không gian 2D, khoảng cách d được quy định:

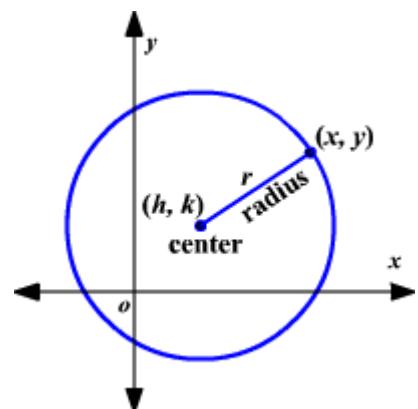
$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Ta có:

$$h = (x_h, y_h)$$

$$P = (x, y)$$

$$d = \sqrt{(x - h)^2 + (y - k)^2} = r$$



Bình phương 2 vế:

$$(x - h)^2 + (y - k)^2 = r^2$$

Nếu tâm đường tròn tại $0(0,0)$

Hay: $x^2 + y^2 = r^2$

Circle Midpoint

Chứng minh phương trình Ellipse:

Ta có: Khoảng cách $d = d_1 + d_2 = 2a$

Trong không gian 2D, khoảng cách d được tính:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

$$d_1 = \sqrt{(x + c)^2 + (y - 0)^2}$$

$$d_2 = \sqrt{(x - c)^2 + (y - 0)^2}$$

$$d_1 + d_2 = \sqrt{(x + c)^2 + (y - 0)^2} + \sqrt{(x - c)^2 + (y - 0)^2}$$

$$d_1 + d_2 = \sqrt{(x + c)^2 + y^2} + \sqrt{(x - c)^2 + y^2}$$

$$\sqrt{(x - c)^2 + y^2} = 2a - \sqrt{(x + c)^2 + y^2}$$

Bình phương 2 vế:

$$(x - c)^2 + y^2 = 4a^2 - 4a\sqrt{(x + c)^2 + y^2} + (x + c)^2 + y^2$$

$$x^2 - 2xc + c^2 + y^2 = 4a^2 - 4a\sqrt{(x + c)^2 + y^2} + x^2 + 2xc + c^2 + y^2$$

$$4a^2 + 4cx = 4a\sqrt{(x + c)^2 + y^2}$$

$$a^2 + cx = a\sqrt{(x + c)^2 + y^2}$$

Bình phương 2 vế:

$$a^4 + 2a^2cx + c^2x^2 = a^2[(x + c)^2 + y^2]$$

$$a^4 + 2acx + c^2x^2 = a^2[x^2 + 2cx + c^2 + y^2]$$

$$a^4 + 2a^2cx + c^2x^2 = a^2x^2 + 2a^2cx + a^2c^2 + a^2y^2$$

$$a^4 - a^2c^2 = a^2x^2 - c^2x^2 + a^2y^2$$

$$a^2(a^2 - c^2) = x^2(a^2 - c^2) + a^2y^2$$

Chia 2 vế cho $(a^2 - c^2)$

$$a^2 = x^2 + \frac{a^2y^2}{(a^2 - c^2)}$$

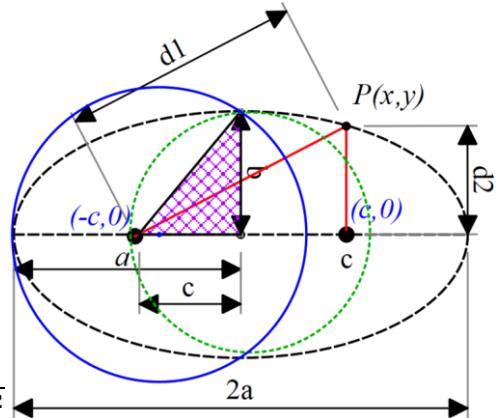
Chia 2 vế cho a^2 , ta có:

$$1 = \frac{x^2}{a^2} + \frac{a^2y^2}{a^2 - c^2}$$

Dựa vào hình tam giác (màu tím), áp dụng định lý pitago: $a^2 = b^2 + c^2$

Ta được:

$$1 = \frac{x^2}{a^2} + \frac{y^2}{b^2}$$



Circle Midpoint

Câu hỏi. Sử dụng thuật toán Midpoint vẽ Ellipse tâm O(0;0), $r_y=2$ và $r_x=4$

1.1 Trình bày các bước để thực hiện giải thuật trên

Các tính chất của Ellipse:

- 1) Đôi xứng qua các gốc phân tư
- 2) Không đôi xứng qua các quãng 8 của các gốc phân tư

Do đó, chúng ta sẽ tính toán các pixel của Ellipse thông qua 1 gốc phân tư thứ nhất. Sau đó, các điểm còn lại của các gốc phân tư khác sẽ thực hiện thông qua các phép đối xứng

Phương trình Ellipse: $f(x, y) = r_y^2x^2 + r_x^2y^2 = r_x^2r_y^2$

$$f(x, y) = \begin{cases} < 0, & \text{nếu } (x, y) \text{ nằm phía trong Ellipse} \\ = 0, & \text{nếu } (x, y) \text{ nằm trên Ellipse} \\ > 0, & \text{nếu } (x, y) \text{ nằm phía ngoài Ellipse} \end{cases}$$

Từ pt Ellipse ở trên, ta có hệ số góc: $m = \frac{dy}{dx} = \frac{2r_y^2x}{2r_x^2y}$

Tại vùng (1) ứng với hệ số góc $m < 1$, ta có: $x = x+1$ và $y = f(x)$

$$\begin{cases} x_{i+1} = x_i + 1, y_{i+1} = y_i - 1 \\ x_{i+1} = x_i + 1, y_{i+1} = y_i \end{cases}$$

Tại vùng (2) ứng với hệ số góc $m > 1$, ta có: $y = y-1$ và $x = f(y)$

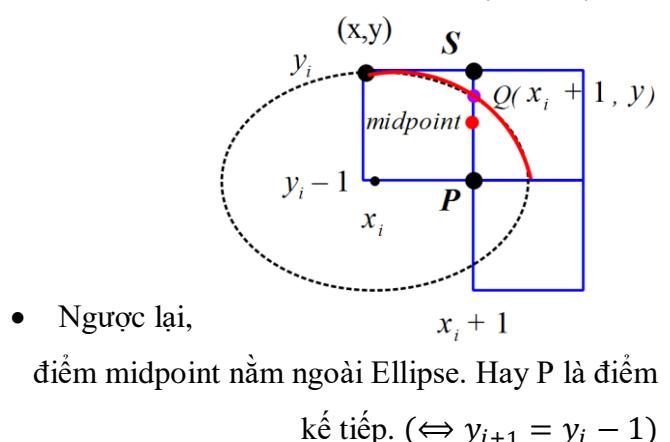
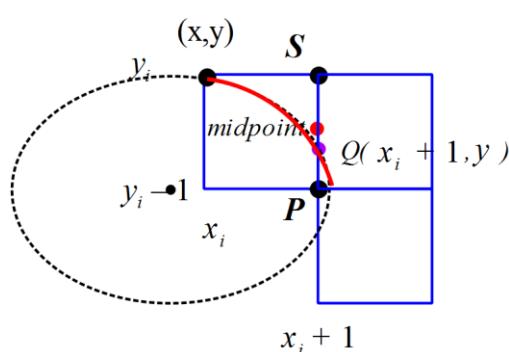
$$\begin{cases} y_{i+1} = y_i - 1, x_{i+1} = x_i + 1 \\ y_{i+1} = y_i - 1, x_{i+1} = x_i \end{cases}$$

Bắt đầu tại tâm (0,r), thực hiện dịch chuyển x tăng dần cho đến khi chạm biên của vùng 1 và vùng 2. Sau đó, dịch chuyển y tăng dần từ biên vùng 2 cho đến hết gốc phân tư thứ nhất.

Trường hợp 1 (vùng 1):

Ta có: $p_i = f_{mid}(x_i + 1, y_i - 1/2) = r_y^2(x_i + 1)^2 + r_x^2(y_i - 1/2)^2 - r_x^2r_y^2$

- Nếu $p_i < 0$, điểm midpoint nằm bên trong Ellipse. Hay S là điểm kế tiếp. ($\Leftrightarrow y_{i+1} = y_i$)



Ta có: $p_{1i} = r_y^2(x_i + 1)^2 + r_x^2(y_i - 1/2)^2 - r_x^2r_y^2$

Circle Midpoint

$$p1_{i+1} = r_y^2(x_{i+1} + 1)^2 + r_x^2(y_{i+1} - 1/2)^2 - r_x^2r_y^2$$

$$p1_{i+1} - p1_i = r_y^2[(x_i + 1) + 1]^2 - (x_i + 1)^2] + r_x^2[(y_{i+1} - 1/2)^2 - (y_i - 1/2)^2]$$

$$\begin{aligned} p1_{i+1} &= p1_i + r_y^2[(x_i + 1)^2 + 2(x_i + 1) + 1 - (x_i + 1)^2] + r_x^2[(y_{i+1} - 1/2)^2 \\ &\quad - (y_i - 1/2)^2] \end{aligned}$$

$$p1_{i+1} = p1_i + 2r_y^2(x_i + 1) + r_y^2 + r_x^2(y_{i+1}^2 - y_i^2) - r_x^2(y_{i+1} - y_i)$$

TH1. $p1_{i+1} = p1_i + 2r_y^2(x_i + 1) + r_y^2$, nếu $p1_i \leq 0$, hay $y_{i+1} = y_i$

TH2. $p1_{i+1} = p1_i + 2r_y^2(x_i + 1) + r_y^2 - 2r_x^2y_i$, nếu $p1_i > 0$, hay $y_{i+1} = y_i - 1$

Xác định $p1_0(0, r_y) = r_y^2(0 + 1)^2 + r_x^2(r_y - 1/2)^2 - r_x^2r_y^2$

$$p1_0 = r_y^2 - r_x^2r_y + \frac{r_x^2}{4}$$

Trường hợp 2 (vùng 2):

Từ pt: $p2_i = f\left(x + \frac{1}{2}, y - 1\right) = r_y^2(x_i + \frac{1}{2})^2 + r_x^2(y_i - 1)^2 - r_x^2r_y^2$

Nếu $p2_i > 0$, điểm midpoint nằm bên ngoài ellipse, x_i gần đường biên hơn

Ngược lại, $p2_i \leq 0$, điểm midpoint nằm bên trong ellipse, $x_i + 1$ gần đường biên hơn

$$\begin{aligned} p2_{i+1} &= r_y^2(x_{i+1} + \frac{1}{2})^2 + r_x^2(y_{i+1} - 1)^2 - r_x^2r_y^2 \\ &= r_y^2(x_{i+1} + \frac{1}{2})^2 + r_x^2((y_i - 1) - 1)^2 - r_x^2r_y^2 \end{aligned}$$

$$p2_{i+1} = r_y^2(x_{i+1} + \frac{1}{2})^2 + r_x^2((y_i - 1)^2 - 2(y_i - 1) + 1) - r_x^2r_y^2$$

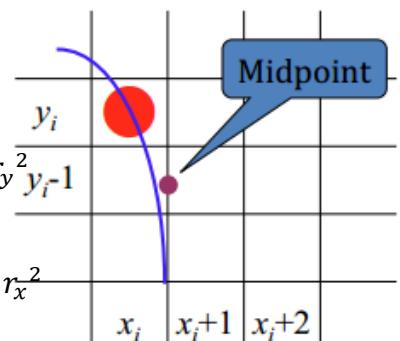
$$p2_{i+1} - p2_i = r_y^2 \left[\left(x_{i+1} + \frac{1}{2} \right)^2 - \left(x_i + \frac{1}{2} \right)^2 \right] + 2r_x^2(y_i - 1) + r_x^2$$

$$p2_{i+1} = p2_i + r_y^2 \left[\left(x_{i+1} + \frac{1}{2} \right)^2 - \left(x_i + \frac{1}{2} \right)^2 \right] - 2r_x^2(y_i - 1) + r_x^2$$

TH1. $p2_{i+1} = p2_i + 2r_x^2(y_i - 1) + r_x^2$, nếu $p2_i > 0$, hay $x_{i+1} = x_i$

TH2. $p2_{i+1} = p2_i + 2r_y^2(x_i + 1) - 2r_x^2(y_i - 1) + r_x^2$, nếu $p2_i \leq 0$, hay $x_{i+1} = x_i + 1$

Xác định $p2_0(r_x, 0) = r_y^2(r_x + \frac{1}{2})^2 + r_x^2(0 - 1)^2 - r_x^2r_y^2$



Circle Midpoint

$$p2_0 = r_y^2 + r_y^2 r_x + r_x^2 - \frac{r_y^2}{4}$$

1.2. Lập trình mô phỏng

```

//khai bao thu vien
#include <graphics.h>
#define ROUND(a) ((int)(a+0.5))
#include<stdio.h>
//khai bao bien
int xc;
int yc;
int rx;
int ry;
int x;
int y;

//chuong trinh con
void nhapxy()
{
    printf("Nhập tam ellipse, xc= ");
    scanf("%d",&xc);
    printf("Nhập tam ellipse, yc= ");
    scanf("%d",&yc);
    printf("Nhập bán kính trục ngang, rx= ");
    scanf("%d",&rx);
    printf("Nhập bán kính trục doc, ry= ");
    scanf("%d",&ry);
}
void drawellipse()
{
    putpixel(xc+x,yc+y,255);
    putpixel(xc-x,yc+y,255);
    putpixel(xc+x,yc-y,255);
    putpixel(xc-x,yc-y,255);
}
void Ellipse()
{
    int p1;
    int px;
    int py;
    int p2;
    initwindow(400,400);
    x=0;
    y=ry;
    p1=ROUND(ry*ry-rx*rx*ry+rx*rx/4);
    px=0;
    py=2*rx*rx*y;
    drawellipse();
}

while(px<py)
{
    x++;
    px=px+2*ry*ry;
    if(p1<=0)
    {
        p1=p1+ry*ry+px;
    }
    else
    {
        y--;
        py=py-2*rx*rx;
        p1=p1+ry*ry+px-py;
    }
    drawellipse();
}
//Region 2
p2=ROUND(ry*ry*(x+0.5)*(x+0.5)+rx*rx*(y-1)*(y-1)-rx*rx*ry*ry);
while (y>0)
{
    y--;
    py=py-2*rx*rx;
    if (p2>0)
    {
        p2=p2+rx*rx - py;
    }
    else
    {
        x++;
        px = px+2*ry*ry;
        p2= p2+rx*rx - py + px;
    }
    drawellipse();
}

//chuong trinh chinh
main()
{
    //khai tao window
    nhapxy();
    Ellipse();
    //closegraph();
    getch();
}

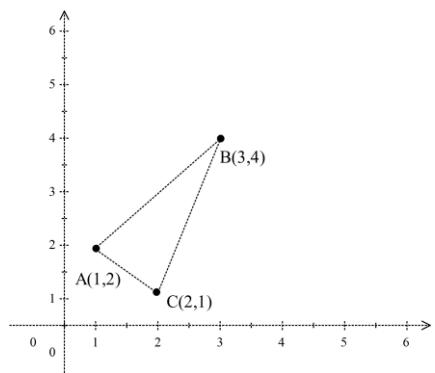
```

Scanline Algorithm

Tô màu đa giác có tọa độ như sau

Ma trận các đỉnh của tam giác

$$a = \begin{bmatrix} & X & Y \\ A & 1 & 2 \\ B & 3 & 4 \\ C & 2 & 1 \\ A & 1 & 2 \end{bmatrix}$$



Xác định giá trị dx và dy

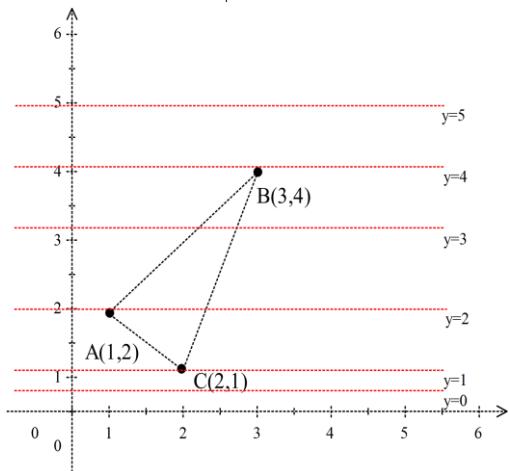
$$\begin{array}{ccc} AB & BC & CA \\ dx = [2 & -1 & -1] \\ dy = [2 & -3 & 1] \end{array}$$

Hệ số góc dy/dx

$$m = \begin{bmatrix} AB & BC & CA \\ 1 & 3 & -1 \end{bmatrix}$$

Hệ số góc dx/dy

$$m = \begin{bmatrix} AB & BC & CA \\ 1 & 1/3 & -1 \end{bmatrix}$$



Tìm các điểm giao cắt các cạnh của tam giác với các dòng quét $y=1$ (y_{min}) đến $y=4$ (y_{MAX}). Nếu tồn tại giao điểm, xác định tọa độ tại điểm cắt với giá trị ($x_{giao điểm}, y$)

- Nếu scanline y bất kỳ cắt qua cạnh XY nào đó khi và chỉ khi:

 □ $(y_X \leq y \text{ và } y_Y > y)$ hoặc $(y_X > y \text{ và } y_Y \leq y)$

- Tọa độ điểm cắt được xác định thông qua phương trình sau:

 □ $m = \frac{dy}{dx} = \frac{y_{sau} - y_{trước}}{x_{sau} - x_{trước}}$ hay $x_{sau} = x_{trước} + \frac{(y - y_{trước})}{m}$

 □ hay nếu đảo giá trị $m = \frac{dx}{dy}$ thì $x_{sau} = x_{trước} + m(y - y_{trước})$

Y=1

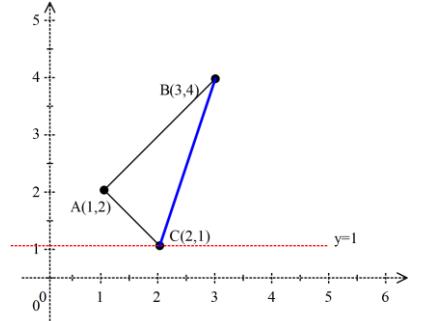
- Xét cạnh AB, rõ ràng cả 2 đỉnh A và B đều không thỏa mãn biểu thức để chứng tỏ đường thẳng $y=1$ cắt ngang cạnh AB ($y_A = 2 > 1$ và $y_B = 4 > 1$). Do đó không cần xác định tọa độ điểm giao nhau giữa $y=1$ và đoạn AB.
- Xét cạnh BC

$y_B = 4 > 1$ và $y_C = 1 \leq 1$. Nên hoành độ giao điểm của BC

$$x_{gd[0]} = x_B + m(y - y_B) \text{ hoặc } x_{gd[0]} = x_C + m(y - y_C)$$

$$x_{gd[0]} = 3 + 1/3(1-4) \text{ hoặc } x_{gd[0]} = 2 + 1/3(1-1)$$

$$x_{gd[0]} = 2 \text{ hoặc } x_{gd[0]} = 2$$



Scanline Algorithm

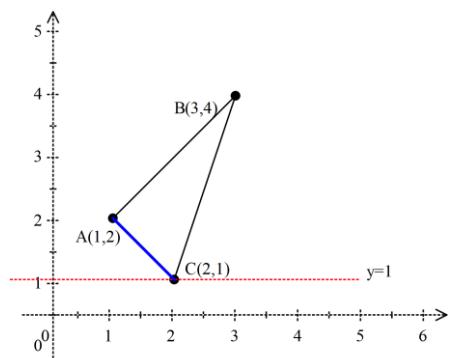
- Xét cạnh CA

$$y_C = 1 \leq 1 \text{ và } y_A = 2 > 1$$

$$x_{gd[1]} = x_A + m(y - y_A) \text{ hoặc } x_{gd[1]} = x_C + m(y - y_C)$$

$$x_{gd[1]} = 1 - 1(1-2) \text{ hoặc } x_{gd[1]} = 2 - 1(1-1)$$

$$x_{gd[1]} = 2 \text{ hoặc } x_{gd[1]} = 2$$



Y=2

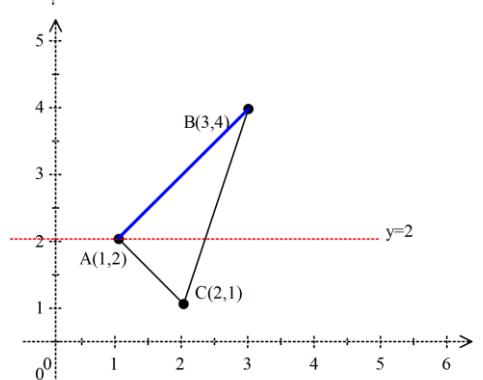
- Xét cạnh AB

$$y_A = 2 \leq 2 \text{ và } y_B = 4 > 2$$

$$x_{gd[2]} = x_A + m(y - y_A) \text{ hoặc } x_{gd[2]} = x_B + m(y - y_B)$$

$$x_{gd[2]} = 1 + 1(2-2) \text{ hoặc } x_{gd[2]} = 3 + 1(2-4)$$

$$x_{gd[2]} = 1 \text{ hoặc } x_{gd[2]} = 1$$



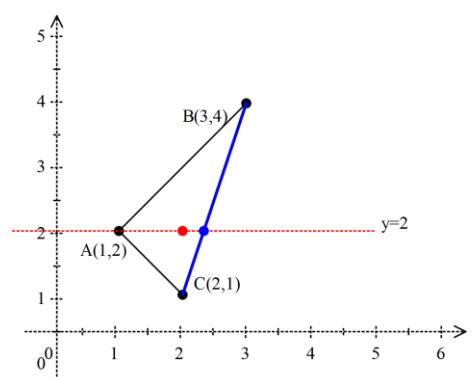
- Xét cạnh BC

$$y_C = 1 \leq 2 \text{ và } y_B = 4 > 2$$

$$x_{gd[3]} = x_B + m(y - y_B) \text{ hoặc } x_{gd[3]} = x_C + m(y - y_C)$$

$$x_{gd[3]} = 3 + 1/3(2-4) \text{ hoặc } x_{gd[3]} = 2 + 1/3(2-1)$$

$$x_{gd[3]} = 2 \text{ hoặc } x_{gd[3]} = 2$$



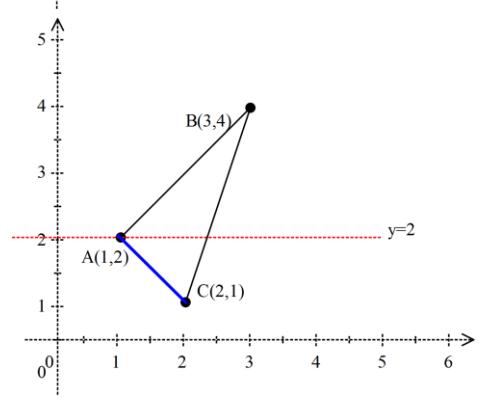
- Xét cạnh CA

$$y_C = 1 < 2 \text{ và } y_A = 2 \geq 2$$

$$x_{gd[4]} = x_A + m(y - y_A) \text{ hoặc } x_{gd[4]} = x_C + m(y - y_C)$$

$$x_{gd[4]} = 1 - 1(2-2) \text{ hoặc } x_{gd[4]} = 2 - 1(2-1)$$

$$x_{gd[4]} = 1 \text{ hoặc } x_{gd[4]} = 1$$



Y=3

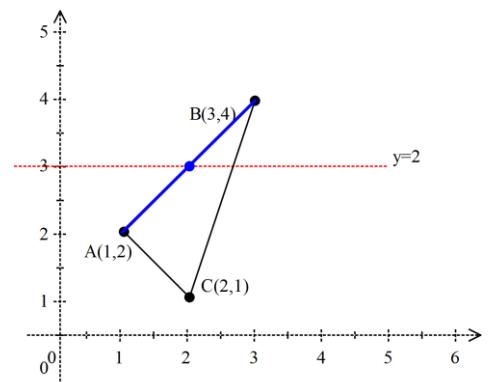
- Xét cạnh AB

$$y_A = 2 \leq 3 \text{ và } y_B = 4 > 3$$

$$x_{gd[5]} = x_A + m(y - y_A) \text{ hoặc } x_{gd[5]} = x_B + m(y - y_B)$$

$$x_{gd[5]} = 1 + 1(3-2) \text{ hoặc } x_{gd[5]} = 3 + 1(3-4)$$

$$x_{gd[5]} = 2 \text{ hoặc } x_{gd[5]} = 2$$



Scanline Algorithm

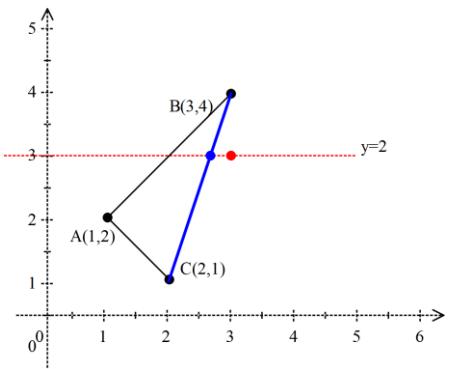
- Xét cạnh BC

$$y_C = 1 \leq 2 \text{ và } y_B = 4 > 2$$

$$x_{gd[3]} = x_B + m(y - y_B) \text{ hoặc } x_{gd[3]} = x_C + m(y - y_C)$$

$$x_{gd[3]} = 3 + 1/3(2-4) \text{ hoặc } x_{gd[3]} = 2 + 1/3(2-1)$$

$$x_{gd[3]} = 2 \text{ hoặc } x_{gd[3]} = 2$$



- Xét cạnh CA

Cả 2 đỉnh A và C đều không thỏa mãn biểu thức để chứng tỏ đường thẳng $y=3$ cắt ngang cạnh AC ($y_A = 2 < 3$ và $y_C = 1 < 3$).. Do đó không cần xác định tọa độ điểm giao nhau giữa $y=3$ và đoạn AC

Y=4

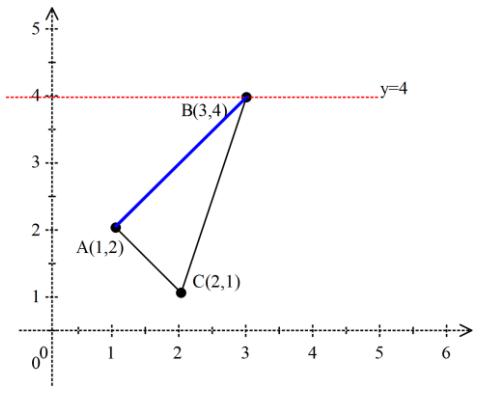
- Xét cạnh AB

$$y_A = 2 < 4 \text{ và } y_B = 4 \geq 4$$

$$x_{gd[7]} = x_A + m(y - y_A) \text{ hoặc } x_{gd[7]} = x_B + m(y - y_B)$$

$$x_{gd[7]} = 1 + 1(4-2) \text{ hoặc } x_{gd[7]} = 3 + 1(4-4)$$

$$x_{gd[7]} = 3 \text{ hoặc } x_{gd[7]} = 3$$



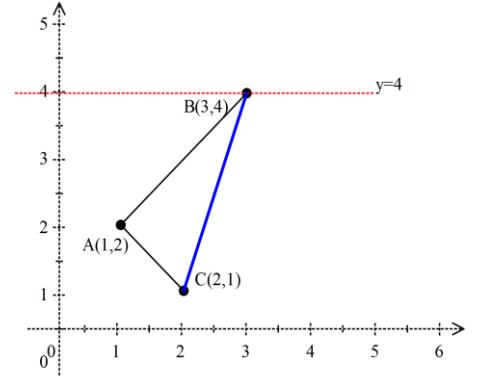
- Xét cạnh BC

$$y_C = 1 < 4 \text{ và } y_B = 4 \geq 4$$

$$x_{gd[8]} = x_B + m(y - y_B) \text{ hoặc } x_{gd[8]} = x_C + m(y - y_C)$$

$$x_{gd[8]} = 3 + 1/3(4-4) \text{ hoặc } x_{gd[8]} = 2 + 1/3(4-1)$$

$$x_{gd[8]} = 3 \text{ hoặc } x_{gd[8]} = 3$$



- Xét cạnh CA

Cả 2 đỉnh A và C đều không thỏa mãn biểu thức để chứng tỏ đường thẳng $y=4$ cắt ngang cạnh AC ($y_A = 2 < 4$ và $y_C = 1 < 4$).. Do đó không cần xác định tọa độ điểm giao nhau giữa $y=3$ và đoạn AC

Tập hợp các giao điểm: $x_{gd} = [2, 2, 1, 2, 1, 2, 2, 3, 3]$

Tọa độ các đoạn thẳng (x_{gd}, y) là:

$$\begin{pmatrix} y = 1 & y = 1 \\ x = 2 & x = 2 \end{pmatrix}, \begin{pmatrix} y = 2 & y = 2 \\ x = 1 & x = 2 \end{pmatrix}, \begin{pmatrix} y = 3 & y = 3 \\ x = 2 & x = 3 \end{pmatrix}, \begin{pmatrix} y = 4 & y = 4 \\ x = 3 & x = 3 \end{pmatrix}$$

Tiến hành tô màu: thực hiện vẽ các đường thẳng nối các tọa độ ở trên

Scanline Algorithm

Tìm công thức hoành độ giao điểm:

(1) Ta có tỷ lệ các đoạn thẳng theo công thức

$$\frac{(x_B - x_A)}{(x - x_A)} = \frac{(y_B - y_A)}{(y - y_A)}$$

Trong đó, y là giá trị của dòng quét (biết trước)

$$x = \frac{(x_B - x_A) * (y - y_A)}{(y_B - y_A)} + x_A$$

$$x = \frac{(y - y_A)}{1/m} + x_A$$

(2) Hay có thể được chứng minh qua công thức sau:

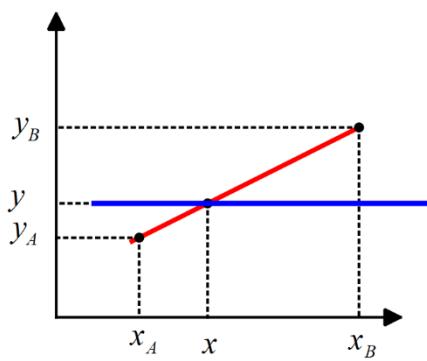
Từ hệ số góc $m = \frac{dy}{dx}$

$$m = \frac{y - y_A}{x - x_A}$$

hoặc

$$m = \frac{y - y_B}{x - x_B}$$

$$x = \frac{y - y_B}{m} + x_B$$



1.2. Lập trình mô phỏng

```
#include<stdio.h>
#include<graphics.h>
#include<math.h>
#define maxdinh 20
int toadodinh[maxdinh][2];
float hesogoc[maxdinh];
int xgd[maxdinh];
int dx;
int dy;
int sodinh;
int ymin;
int yMAX;

void nhaptoado()
{
    //nhap so dinh
    do
    {
        printf("Nhập số đỉnh của đa giác: ");
        scanf("%d",&sodinh);
    }
    while(sodinh<3 || sodinh >maxdinh);
    //nhap to do lan luot cho cac dinh
    int i;
    for(i=0;i<sodinh;i++)
    {
        printf("X[%d] = ",i);
        scanf("%d",&toadodinh[i][0]);
        printf("Y[%d] = ",i);
        scanf("%d",&toadodinh[i][1]);
    }
    //gan to do dinh cuoi ve dinh dau
    toadodinh[sodinh][0]=toadodinh[0][0];
    toadodinh[sodinh][1]=toadodinh[0][1];
    //xac dinh he so goc cac canh
    for(i=0;i<sodinh;i++)
    {
        dx=toadodinh[i+1][0] - toadodinh[i][0];
        dy=toadodinh[i+1][1] - toadodinh[i][1];
        if(dx==0)
        {
            hesogoc[i]=1.0;
        }
        else
        {
            if(dy==0)
            {
                hesogoc[i]=0.0;
            }
            else
            {
                hesogoc[i]=(float)dy/dx;
            }
        }
    }
}

//in ra danh sach cac thong so
printf("So dinh: %d\n",sodinh);
printf("Toa do cac dinh: \n");
for(i=0;i<sodinh;i++)
{
    printf("(%d,%d)\t",toadodinh[i][0],toadodinh[i][1]);
}
printf("\nHe so goc cua cac canh la:\n");
for(i=0;i<sodinh;i++)
{
    printf("%f\t",hesogoc[i]);
}
}

//Tim min max
void y_minmax()
{
    int i;
    //tim ymin
    ymin=toadodinh[0][1];
    for(i=0;i<sodinh;i++)
    {
        if(ymin>toadodinh[i][1])
        {
            ymin=toadodinh[i][1];
        }
    }
    //tim yMAX
    yMAX=toadodinh[0][1];
    for(i=0;i<sodinh;i++)
    {
        if(yMAX<toadodinh[i][1])
        {
            yMAX=toadodinh[i][1];
        }
    }
}

//in ra scanline tu ymin den yMAX
printf("\ny_min = %d \t y_MAX = %d",ymin,yMAX);

void vedagiac()
{
    int i;
    setcolor(BLUE);
    for(i=0;i<sodinh;i++)
    {
        line(toadodinh[i][0],toadodinh[i][1],toadodinh[i+1][0],toadodinh[i+1][1]);
    }
}
```

Scanline Algorithm

```

void scanline()
{
    int y;
    int i;
    int k;

    y=ymin;
    while(y<=yMAX)
    {
        //tim giao diem cac canh
        k=0;
        for(i=0;i<sodinh;i++)
        {
            //xet xem dong quet y co cat canh dang xet
            hay khong?
            if((toadodinh[i][1]<y &&
            toadodinh[i+1][1]<y)||((toadodinh[i][1]>y &&
            toadodinh[i+1][1]>y))
            {
                //khong co giao diem;
            }
            else
            {
                //Tim hoanh do giao diem
                xgd[k]=round(toadodinh[i][0]+(y-
                toadodinh[i][1])/hesogoc[i]);
                k++;
            }
        }

        //sap xep cac hoanh do giao diem va in ra
        int j;
        int t;
        int tam;

        for(j=0;j<k-1;j++)
        {
            for(t=j+1;t<k;t++)
            {
                if(xgd[j]>xgd[t])
                {
                    //doi cho
                    tam=xgd[j];
                    xgd[j]=xgd[t];
                    xgd[t]=tam;
                }
            }
        }

        //in ra cac giao diem sau khi da sap xep
        for(j=0;j<k;j++)
        {
            printf("\n(%d,%d)",xgd[j],y);
        }

        //Ve duong thang noi cac giao diem
        setcolor(RED);
        for(j=0;j<k-1;j=j+2)
        {
            line(xgd[j],y,xgd[j+1],y);
        }

        //Tang dong quet len 1
        y++;
    }

    int main()
    {
        nhaptoado();
        y_minmax();
        initwindow(400,400);
        vedagiac();
        scanline();
        getch();
    }
}

```

Transformation 2D

Bài 1. Thực hiện tịnh tiến đối tượng có tọa độ (0,0) (1,3) (-1,2) với $dx=2$ và $dy=1.5$

Ta có: $P(x,y)$ là điểm ban đầu, $P'(x',y')$ là điểm tại vị trí mới

$$x' = x + t_x \text{ và } y' = y + t_y$$

Cặp (t_x, t_y) gọi là vector dịch chuyển

$$\text{Ta có: } P = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$P' = \begin{bmatrix} x'_1 \\ x'_2 \end{bmatrix}$$

$$T = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

$$\text{Từ đó: } P' = P + T$$

Áp dụng:

Ta có phương trình biến đổi:

$$P' = P + T \text{ Với } T = \begin{bmatrix} 2 \\ 1.5 \end{bmatrix}$$

$$\text{Hay } \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 2 \\ 1.5 \end{bmatrix}$$

Ta có, tại tọa độ (0,0)

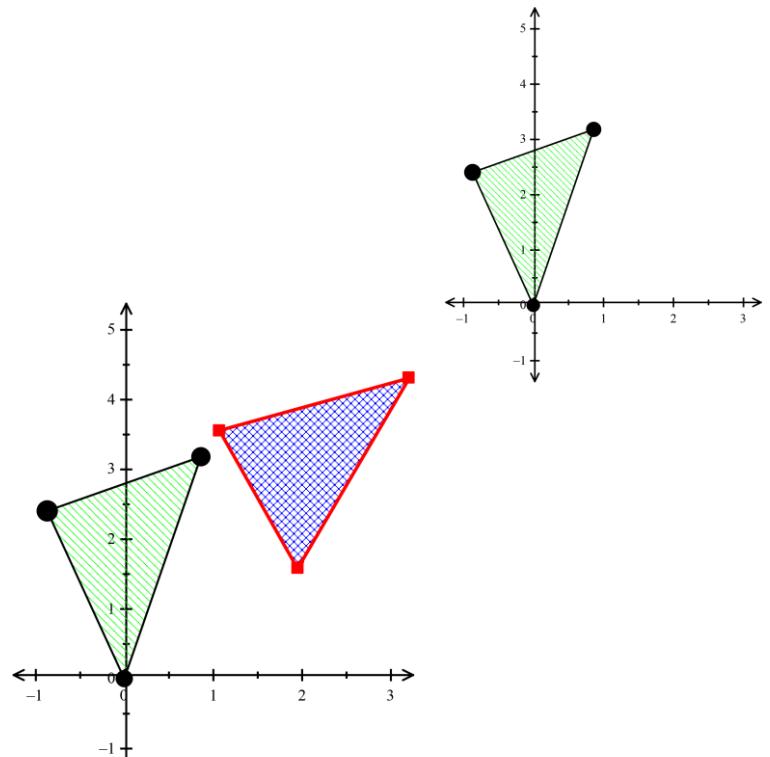
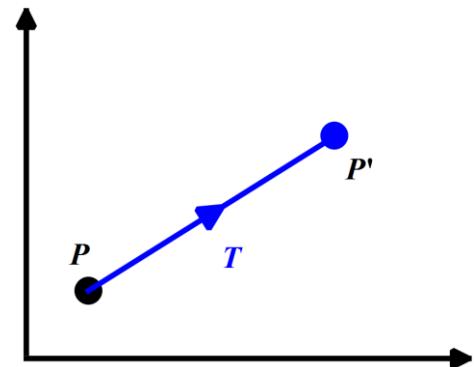
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 2 \\ 1.5 \end{bmatrix} = \begin{bmatrix} 2 \\ 1.5 \end{bmatrix}$$

Ta có, tại tọa độ (1,3)

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \end{bmatrix} + \begin{bmatrix} 2 \\ 1.5 \end{bmatrix} = \begin{bmatrix} 3 \\ 4.5 \end{bmatrix}$$

Ta có, tại tọa độ (-1,2)

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 \\ 2 \end{bmatrix} + \begin{bmatrix} 2 \\ 1.5 \end{bmatrix} = \begin{bmatrix} 1 \\ 3.5 \end{bmatrix}$$



1.2. Lập trình mô phỏng

```
#include <graphics.h>
//khai bao bien
int xshift;
int yshift;
int n;
int xc[100];
int yc[100];

//chuong trinh con
void draw_object()
{
    int i;

    for(i=0;i<n;i++)
    {
        line(xc[i],yc[i],xc[(i+1)%n],yc[(i+1)%n]);
        delay(1000);
    }
}

//nhap thong so
void nhapdulieu()
{
    int i;

    printf("Nhập số cạnh của đa giác = ");
    scanf("%d",&n);

    for (i=0;i<n;i++)
    {
        printf("Toa độ x cho cạnh %d = ",i);
        scanf("%d",&xc[i]);
        printf("Toa độ y cho cạnh %d = ",i);
        scanf("%d",&yc[i]);
    }
}

printf("Khoảng cách dịch chuyển theo trục
x = ");
scanf("%d",&xshift);
printf("Khoảng cách dịch chuyển theo trục
y = ");
scanf("%d",&yshift);
}

//dịch chuyển đối tượng
void translate()
{
    int i;

    for(i=0;i<n;i++)
    {
        xc[i]=xc[i]+xshift;
        yc[i]=yc[i]+yshift;
    }
}

//chương trình chính
int main()
{
    nhapdulieu();
    initwindow(800,800);

    draw_object();
    delay(1000);
    translate();
    draw_object();

    getch();
    return 0;
}
```

Transformation 2D

Bài 2. Thực hiện phép quay dương cho đối tượng có tọa độ (0,0) (2,4) (-3,4) với 1 góc $\varphi = 90^\circ$

Phương trình biến đổi để xoay điểm P khi tâm ở gốc tọa độ và mối quan với hệ góc quay

$$x' = r\cos(\varphi + \theta) = r\cos\varphi\cos\theta - r\sin\varphi\sin\theta$$

$$y' = r\sin(\varphi + \theta) = r\cos\varphi\sin\theta + r\sin\varphi\cos\theta$$

Trong đó: θ góc tại vị trí ban đầu

Các tọa độ ban đầu của điểm trong tọa độ cực là

$$x = r\cos\theta; y = r\sin\theta$$

Thay vào phương trình trước ta có:

$$x' = x\cos\varphi - y\sin\varphi$$

$$y' = x\sin\varphi + y\cos\varphi$$

Từ ma trận biến đổi P và P': $P = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ $P' = \begin{bmatrix} x'_1 \\ x'_2 \end{bmatrix}$

Ta có: $P' = P \cdot R$. VỚI $R = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix}$

Áp dụng:

Ta có phương trình biến đổi:

$$P' = P \cdot R. VỚI R = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix}$$

$$(\sin\varphi)=1; (\cos\varphi)=0$$

$$\text{Hay } \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} \cdot \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

Ta có, tại tọa độ (0,0)

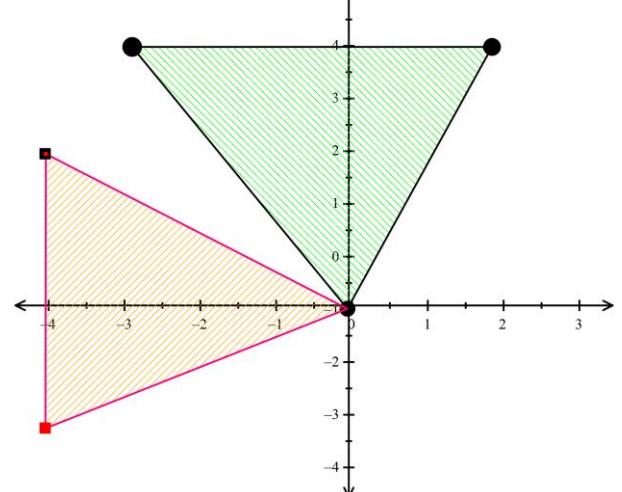
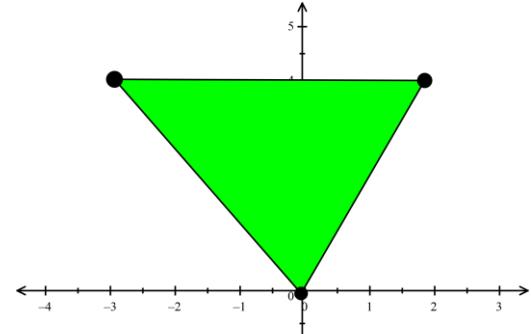
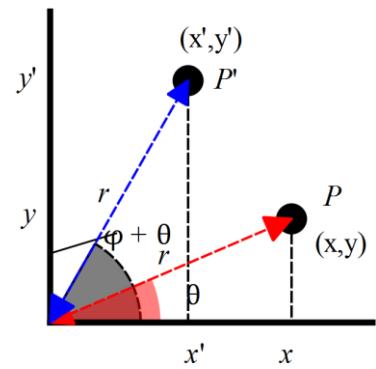
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Ta có, tại tọa độ (2,4)

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \end{bmatrix} \cdot \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} -4 \\ 2 \end{bmatrix}$$

Ta có, tại tọa độ (-3,4)

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -3 \\ 4 \end{bmatrix} \cdot \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} -4 \\ -3 \end{bmatrix}$$



Transformation 2D

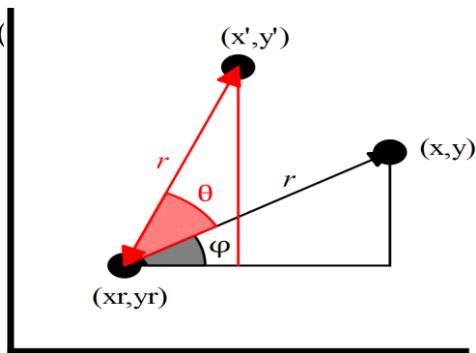
► Góc quay dương ngược chiều kim đồng hồ và ngược lại

Bài 3. Thực hiện phép quay dương cho đối tượng có tọa độ () với 1 góc $\varphi = 90^\circ$

Quay 1 điểm với tâm tùy ý (khác với gốc tọa độ)

$$x' = xr + (x - xr)\cos\theta - (y - yr)\sin\theta$$

$$y' = yr + (x - xr)\sin\theta + (y - yr)\cos\theta$$



Áp dụng công thức trên ta có:

Ta có, tại tọa độ (1,1)

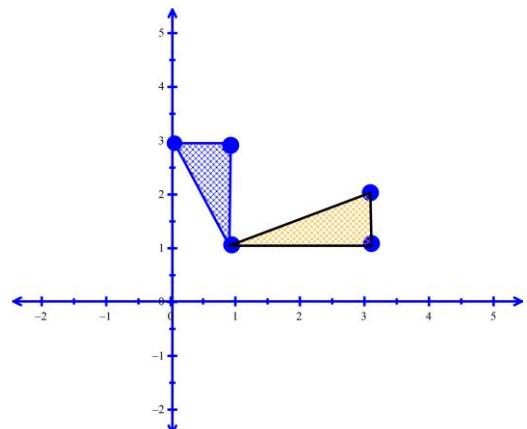
$$x' = 1 + (1 - 1)0 - (1 - 1)1 = 1$$

$$y' = 1 + (1 - 1)1 + (1 - 1)0 = 1$$

Ta có, tại tọa độ (3,1)

$$x' = 1 + (3 - 1)0 - (1 - 1)1 = 1$$

$$y' = 1 + (3 - 1)1 + (1 - 1)0 = 3$$



Ta có, tại tọa độ (3,2)

$$x' = 1 + (3 - 1)0 - (2 - 1)1 = 0$$

$$y' = 1 + (3 - 1)1 + (2 - 1)0 = 3$$

Chứng minh biểu thức:

$$x' = xr + (x - xr)\cos\alpha - (y - yr)\sin\alpha$$

$$y' = yr + (x - xr)\sin\alpha + (y - yr)\cos\alpha$$

Bước 1. Tịnh tiến đối tượng về gốc tọa độ từ vị trí (x_r, y_r)

$$x_A' = x_A - x_r \text{ và } y_A' = y_A - y_r$$

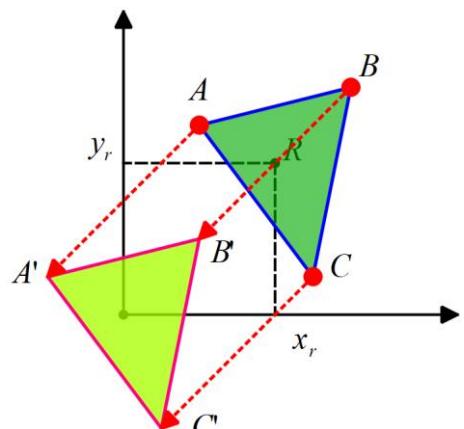
$$x_B' = x_B - x_r \text{ và } y_B' = y_B - y_r$$

$$x_C' = x_C - x_r \text{ và } y_C' = y_C - y_r$$

Hay

$$x = x - x_r \text{ và } y = y - y_r$$

Bước 2. Quay tại gốc tọa độ với góc quay α



Bước 1: Tịnh tiến về gốc tọa độ

Transformation 2D

Từ phương trình quay đối tượng: $x' = x\cos\alpha - y\sin\alpha$ và $y' = x\sin\alpha + y\cos\alpha$

Ta thế: $x = x - x_r$ và $y = y - y_r$ vào phương trình quay đối tượng, ta có:

$$x' = (x - x_r)\cos\alpha - (y - y_r)\sin\alpha$$

$$y' = (x - x_r)\sin\alpha + (y - y_r)\cos\alpha$$

Bước 3. Tịnh tiến đối tượng về vị trí tâm (x_r, y_r) ban đầu:

Từ pt: $x = x + x_r$ và $y = y + y_r$, ta lấy x' và y' cộng với giá trị lần lượt x_r và y_r

$$x' = x_r + (x - x_r)\cos\alpha - (y - y_r)\sin\alpha$$

$$y' = y_r + (x - x_r)\sin\alpha + (y - y_r)\cos\alpha$$

1.2. Lập trình mô phỏng

```

#include <stdio.h>
#include <graphics.h>
#include <math.h>
//khai bao bien
int xr;
int yr;
int n;
int xc[100];
int yc[100];
int degree;
float radian;
//chuong trinh con
void draw_object()
{
    int i;

    for(i=0;i<n;i++)
    {
        line(xc[i],yc[i],xc[(i+1)%n],yc[(i+1)%n]);
        delay(1000);
    }
}
//nhap thong so
void nhapdulieu()
{
    int i;
    printf("Nhập số cạnh của đa giác = ");
    scanf("%d",&n);

    for (i=0;i<n;i++)
    {
        printf("Toa độ x cho cạnh %d = ",i);
        scanf("%d",&xc[i]);
        printf("Toa độ y cho cạnh %d = ",i);
        scanf("%d",&yc[i]);
    }
    printf("Toa độ x của đối tượng tại vị trí mới
          = ");
    scanf("%d",&xr);
    printf("Toa độ y của đối tượng tại vị trí mới
          = ");
    scanf("%d",&yr);
    printf("Hệ số góc quay (degree) = ");
    scanf("%d",&degree);
}
void convert_degree_2_radian()
{
    radian=(float) degree/180*3.14;
}
//dịch chuyển đối tượng
void rotation()
{
    int i;
    float dx;
    float dy;
    for(i=0;i<n;i++)
    {
        dx=float(xc[i]-xr);
        dy=float(yc[i]-yr);
        xc[i]=floor(xr+dx*cos(radian)-
dy*sin(radian));
        yc[i]=floor(yr+dx*sin(radian)+dy*cos(radia
n));
    }
}
//vẽ hình vuông quay
void five_square()
{
    int i;
    float dx;
    float dy;
    int num_square;
    for(num_square=0; num_square<5;
num_square++)
    {
        for(i=0;i<n;i++)
        {
            dx=float(xc[i]-xr);
            dy=float(yc[i]-yr);
            xc[i]=floor(xr+dx*cos(radian)-
dy*sin(radian));

            yc[i]=floor(yr+dx*sin(radian)+dy*cos(radia
n));
        }
        draw_object();
    }
}
//chuong trinh chinh
int main()
{
    nhapdulieu();
    convert_degree_2_radian();
    initwindow(800,800);
    draw_object();
    delay(1000);
    rotation();
    draw_object();
    //five_square();
    getch();
}

```

Transformation 2D

}

Bài tập 4. Thực hiện phép biến đổi tỷ lệ cho đối tượng có tọa độ (-2,1) (-1,4) (1,2) với hệ số S(2,3)

Một phép biến đổi tỷ lệ làm thay đổi kích thước của một đối tượng. Thao tác này có thể được thực hiện cho đa giác bằng cách nhân các giá trị tọa độ (x, y) của từng đỉnh với các hệ số tỷ lệ s_x và s_y , để tạo ra tọa độ biến đổi (x' , y'): $x' = x \cdot s_x$ $y' = y \cdot s_y$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} \quad \text{hay } P' = S \cdot P$$

Trong đó s_x , s_y là các số nguyên dương. Nếu giá trị của s_x và s_y bé hơn 1 được phép thu nhỏ, ngược lại là phép phóng to.

Nếu cả s_x và s_y đều là 1, đối tượng không có sự biến đổi

Nếu s_x và s_y có cùng 1 giá trị, phép biến đổi được gọi là phép biến đổi đồng nhất

Áp dụng:

Ta có phương trình biến đổi:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

Với $s_x = 2$ và $s_y = 3$

$$\text{Hay } \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} \cdot \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}$$

Ta có, tại tọa độ (-2,1)

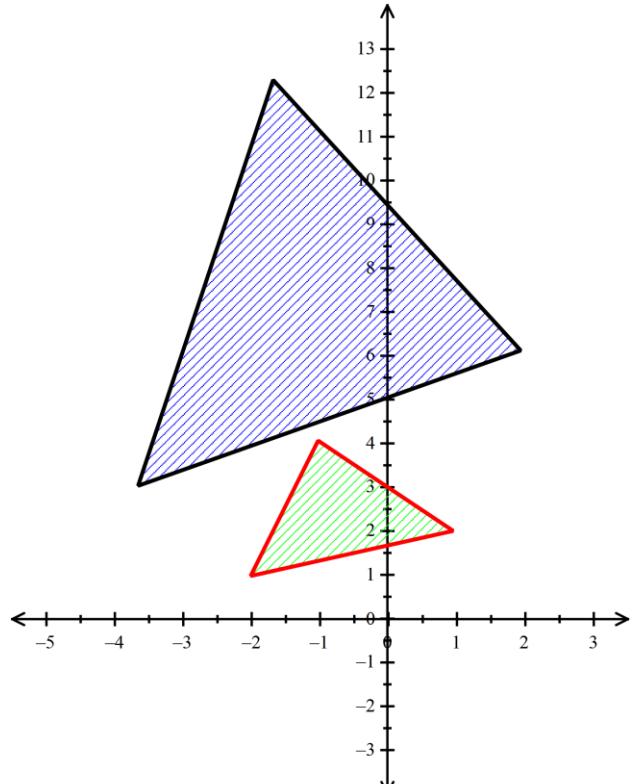
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -2 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix} = \begin{bmatrix} -4 \\ 3 \end{bmatrix}$$

Ta có, tại tọa độ (-1,4)

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 \\ 4 \end{bmatrix} \cdot \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix} = \begin{bmatrix} -2 \\ 12 \end{bmatrix}$$

Ta có, tại tọa độ (1,2)

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \cdot \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix} = \begin{bmatrix} 2 \\ 6 \end{bmatrix}$$



Transformation 2D

Bài tập 5. Thực hiện phép biến đổi tỷ lệ cho đối tượng có tọa độ (-2,1) (-1,4) (1,2) với hệ số S(2,3) tại tâm có tọa độ (1,3).

Ta có thể kiểm soát vị trí của một đối tượng được chia tỷ lệ bằng cách chọn một vị trí, được gọi là điểm cố định, nghĩa là không thay đổi sau khi chuyển đổi tỷ lệ. Điểm cố định (x_f, y_f) có thể được chọn làm một trong các đỉnh, tâm đối tượng hoặc bất kỳ vị trí nào khác. Một đa giác sau khi được chia tỷ lệ so với điểm cố định bằng cách chia tỷ lệ khoảng cách từ mỗi đỉnh đến điểm cố định.

$$x' = x_f + (x - x_f) \cdot s_x \quad y' = y_f + (y - y_f) \cdot s_y$$

$$\text{hay } x' = x \cdot s_x + (1 - s_x) \cdot x_f \quad y' = y \cdot s_y + (1 - s_y) \cdot y_f$$

Áp dụng:

$$S_x = 2; S_y = 3; x_f = -1 \text{ và } y_f = 3$$

Ta có, tại tọa độ (-2,1)

$$x' = (-2) \cdot 2 + (1 - 2) \cdot -1 = -2$$

$$y' = 1 \cdot 3 + (1 - 3) \cdot 3 = -3$$

Ta có, tại tọa độ (-1,4)

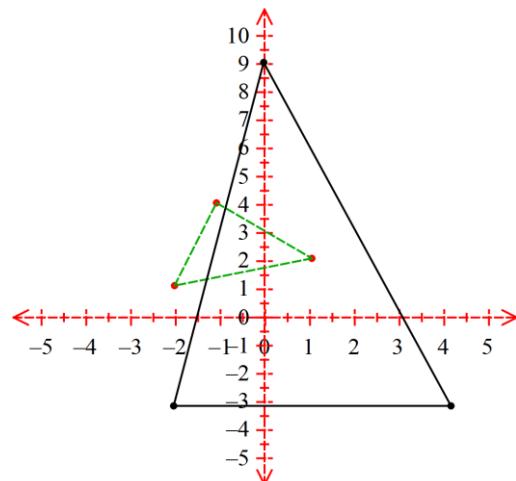
$$x' = (-1) \cdot 2 + (1 - 2) \cdot -1 = 0$$

$$y' = 4 \cdot 3 + (1 - 3) \cdot 3 = 9$$

Ta có, tại tọa độ (1,2)

$$x' = 1 \cdot 2 + (1 - 2) \cdot -1 = 4$$

$$y' = 2 \cdot 3 + (1 - 3) \cdot 3 = -3$$



1.2. Lập trình mô phỏng

```
#include <graphics.h>
#include <math.h>
//khai bao bien
int n;
int xc[100];
int yc[100];
int xs;
int ys;
int degree;
float scalex;
float scaley;
//chuong trinh con
void draw_object()
{
    int i;

    for(i=0;i<n;i++)
    {
        line(xc[i],yc[i],xc[(i+1)%n],yc[(i+1)%n]);
        delay(1000);
    }
}

//nhap thong so
void nhapdulieu()
{
    int i;
    printf("Nhập số cạnh của đa giác = ");
    scanf("%d",&n);

    for (i=0;i<n;i++)
    {
        printf("Toa độ x cho cạnh %d = ",i);
        scanf("%d",&xc[i]);
        printf("Toa độ y cho cạnh %d = ",i);
        scanf("%d",&yc[i]);
    }

    printf("Tỷ lệ biến đổi theo trục x = ");
    scanf("%f",&scalex);
    printf("Tỷ lệ biến đổi theo trục y = ");
    scanf("%f",&scaley);
    printf("Toa độ moi của doi tuong tren truc x = ");
    scanf("%d",&xs);
    printf("Toa độ moi của doi tuong tren truc y = ");
    scanf("%d",&ys);
}

//bien doi ty le
void scaling()
{
    int i;
    int dx;
    int dy;
    for(i=0;i<n;i++)
    {
        //chon tam cua doi tuong la diem x0,y0
        xc[i]=xs+(int)((float)(xc[i]-xs)*scalex);
        yc[i]=ys+(int)((float)(yc[i]-ys)*scaley);
        printf("%d,%d\n",xc[i],yc[i]);
    }
}

//bien doi chong cheo
void fivesquare()
{
    int i;
    int num_square;
    int dx;
    int dy;
    for(num_square=0; num_square<5;
num_square++)
    {
        for(i=0;i<n;i++)
        {
            //chon tam cua doi tuong la diem xs,ys
            xc[i]=xs+(int)((float)(xc[i]-xs)*scalex);
            yc[i]=ys+(int)((float)(yc[i]-ys)*scaley);
            printf("%d,%d\n",xc[i],yc[i]);
        }
        draw_object();
    }
}

//chuong trinh chinh
int main()
{
    nhapdulieu();
    initwindow(800,800);
    draw_object();
    delay(1000);

    //scaling();
    //draw_object();
    fivesquare();

    getch();
    return 0;
}
```

Transformation 2D

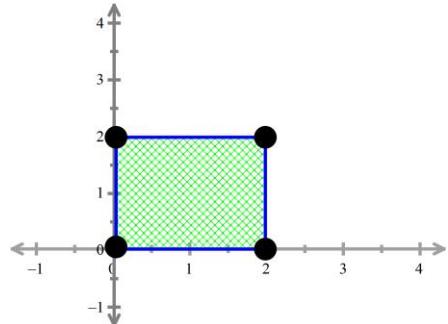
Bài tập 6. Thực hiện phép biến dạng cho hình sau (0,0) (2,0) (2,2) (0,2) theo trục x với $sh_x=2$, tương tự cho $sh_y=2$

Ta có phương trình biến đổi theo x:

$$\blacksquare \quad y' = y \quad x' = x + sh_x \cdot y$$

Ta có phương trình biến đổi theo y:

$$\blacksquare \quad x' = x \quad y' = y + sh_y \cdot x$$



Áp dụng:

Thực hiện phép biến dạng cho hình sau (0,0) (2,0) (2,2) (0,2) theo trục x với $sh_x=2$

Ta có phương trình biến đổi (0,0):

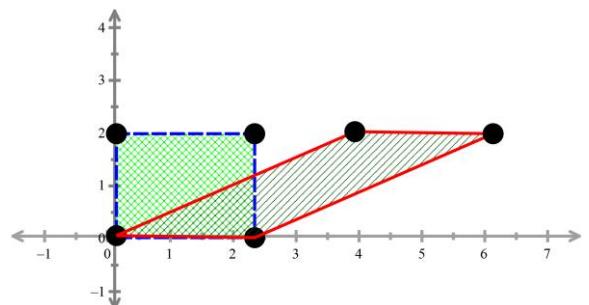
$$y' = 0 \quad x' = 0 + 2 \cdot 0 = 0$$

Ta có phương trình biến đổi (2,0):

$$y' = 0 \quad x' = 2 + 2 \cdot 0 = 2$$

Ta có phương trình biến đổi (2,2):

$$y' = 2 \quad x' = 2 + 2 \cdot 2 = 6$$



Ta có phương trình biến đổi (0,2):

$$y' = 2 \quad x' = 0 + 2 \cdot 2 = 4$$

Thực hiện phép biến dạng cho hình sau (0,0) (2,0) (2,2) (0,2) theo trục y với $sh_y=2$

Ta có phương trình biến đổi (0,0):

$$x' = 0 \quad y' = 0 + 2 \cdot 0 = 0$$

Ta có phương trình biến đổi (2,0):

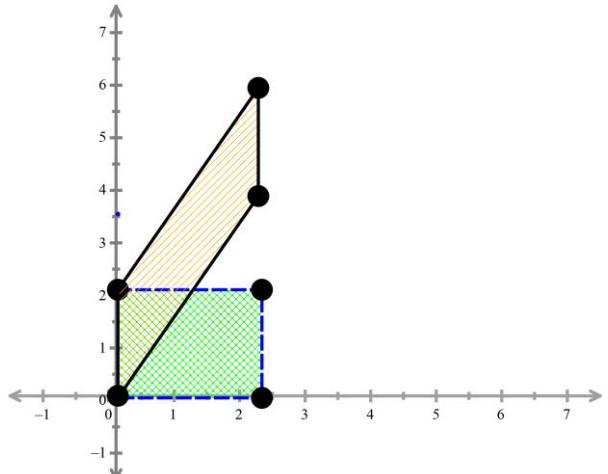
$$x' = 2 \quad y' = 2 + 2 \cdot 0 = 2$$

Ta có phương trình biến đổi (2,2):

$$x' = 2 \quad y' = 2 + 2 \cdot 2 = 6$$

Ta có phương trình biến đổi (0,2):

$$x' = 0 \quad y' = 2 + 2 \cdot 0 = 2$$



1.2. Lập trình mô phỏng

```
#include <graphics.h>
#include <math.h>
#define ROUND(a) ((int)(a+0.5))

//khai bao bien
int n;
int xc[100];
int yc[100];
int shx;
int shy;

//chuong trinh con
void draw_object()
{
    int i;

    for(i=0;i<n;i++)
    {
        line(xc[i],yc[i],xc[(i+1)%n],yc[(i+1)%n]);
        delay(1000);
    }
}

//nhap thong so
void nhapdulieu()
{
    int i;

    printf("Nhập số cạnh của đa giác = ");
    scanf("%d",&n);

    for (i=0;i<n;i++)
    {
        printf("Toa độ x cho cạnh %d = ",i);
        scanf("%d",&xc[i]);
        printf("Toa độ y cho cạnh %d = ",i);
        scanf("%d",&yc[i]);
    }

    printf("Hệ số biến đổi theo trục x = ");
    scanf("%d",&shx);
    printf("Hệ số biến đổi theo trục y = ");
    scanf("%d",&shy);
}
}

//Bien dang theo truc X
void shearingX()
{
    int i;

    for(i=0;i<n;i++)
    {
        xc[i]=xc[i]+shx*yc[i];
        printf("(%.2f,%.2f)\t",xc[i],yc[i]);
    }
}

//Bien dang theo truc Y
void shearingY()
{
    int i;

    for(i=0;i<n;i++)
    {
        yc[i]=yc[i]+shy*xc[i];
        printf("(%.2f,%.2f)\t",xc[i],yc[i]);
    }
}

//chuong trinh chinh
int main()
{
    nhapdulieu();
    initwindow(800,800);
    draw_object();
    delay(1000);

    shearingX();
    draw_object();

    shearingY();
    draw_object();

    getch();
    return 0;
}
```

Transformation 2D

Lập trình mô phỏng cho phép lật đối tượng (đối xứng qua các trục tọa độ)

1. Lật qua đường chéo chính

```

#include <graphics.h>
#include <math.h>

//khai bao bien
int n;
int xc[100];
int yc[100];
int xs;
int ys;
int degree;
float scalex;
float scaley;

//chuong trinh con
void draw_object()
{
    int i;

    for(i=0;i<n;i++)
    {
        line(xc[i],yc[i],xc[(i+1)%n],yc[(i+1)%n]);
        delay(1000);
    }
}

//nhap thong so
void nhapdulieu()
{
    int i;

    printf("Nhập số cạnh của đa giác = ");
    scanf("%d",&n);

    for (i=0;i<n;i++)
    {
        printf("Toa do x cho cạnh %d = ",i);
        scanf("%d",&xc[i]);
        printf("Toa do y cho cạnh %d = ",i);
        scanf("%d",&yc[i]);
    }
}

//Lat theo goc toa do la giao nhau y=x va
//x=y
void FlipDiagonal()
{
    int tempCord;
    int i;
    int tempX;
    int tempY;

    tempY=getmaxx()/2;
    tempX=getmaxy()/2;
    for(i=0;i<n;i++)
    {
        xc[i]=tempX+(tempY-xc[i]);
        yc[i]=tempY+(tempX-yc[i]);

        printf("(%d,%d,%d,%d)\t",xc[i],yc[i],getma
xx(),getmaxy());
    }

    //Ve duong phan cach theo duong cheo
    //chinh
    line(0,getmaxy(),getmaxx(),0);
}

//chuong trinh chinh
int main()
{
    nhapdulieu();
    initwindow(800,800);
    draw_object();
    delay(1000);

    FlipDiagonal();
    draw_object();

    getch();
    return 0;
}

```

2. Lật qua đường chéo phụ

```
#include <graphics.h>
#include <math.h>

//khai bao bien
int n;
int xc[100];
int yc[100];
int xs;
int ys;
int degree;
float scalex;
float scaley;

//chuong trinh con
void draw_object()
{
    int i;

    for(i=0;i<n;i++)
    {
        line(xc[i],yc[i],xc[(i+1)%n],yc[(i+1)%n]);
        delay(1000);
    }
}

//nhap thong so
void nhapdulieu()
{
    int i;

    printf("Nhập số cạnh của đa giác = ");
    scanf("%d",&n);

    for (i=0;i<n;i++)
    {
        printf("Toa độ x cho cạnh %d = ",i);
        scanf("%d",&xc[i]);
        printf("Toa độ y cho cạnh %d = ",i);
        scanf("%d",&yc[i]);
    }
}

//Lat theo goc toa do la giao nhau y=x va
//x=y
void FlipDiagonalSubmain()
{
    int tempCord;
    int i;
    int tempX;
    int tempY;

    tempY=getmaxx()/2;
    tempX=getmaxy()/2;
    for(i=0;i<n;i++)
    {
        xc[i]=tempY+(tempY-xc[i]);
        yc[i]=getmaxy()-yc[i];

        printf("(%.d,%.d,%.d,%.d)\t",xc[i],yc[i],getmaxx(),getmaxy());
    }

    //Ve duong phan cach theo duong cheo
    //chinh
    line(0,0,getmaxx(),getmaxy());
}

//chuong trinh chinh
int main()
{
    nhapdulieu();
    initwindow(800,800);
    draw_object();
    delay(1000);

    FlipDiagonalSubmain();
    draw_object();

    getch();
    return 0;
}
```

Transformation 2D

3. Lật qua gốc tọa độ

```

#include <graphics.h>
#include <math.h>

//khai bao bien
int n;
int xc[100];
int yc[100];
int xs;
int ys;
int degree;
float scalex;
float scaley;

//chuong trinh con
void draw_object()
{
    int i;

    for(i=0;i<n;i++)
    {
        line(xc[i],yc[i],xc[(i+1)%n],yc[(i+1)%n]);
        delay(1000);
    }
}

//nhap thong so
void nhapdulieu()
{
    int i;

    printf("Nhập số cạnh của đa giác = ");
    scanf("%d",&n);

    for (i=0;i<n;i++)
    {
        printf("Toa do x cho cạnh %d = ",i);
        scanf("%d",&xc[i]);
        printf("Toa do y cho cạnh %d = ",i);
        scanf("%d",&yc[i]);
    }
}

```

```

//Lat theo goc toa do la giao nhau y=x va
x=y
void FlipCordinate()
{
    int tempCord;
    int i;
    int tempX;
    int tempY;

    tempY=getmaxx()/2;
    tempX=getmaxy()/2;
    for(i=0;i<n;i++)
    {
        xc[i]=tempY+(tempY-xc[i]);
        yc[i]=tempX+(tempX-yc[i]);
    }

    //Ve duong phan cach theo chieu doc
    for(i=0;i<getmaxy();i++)
    {
        putpixel(tempY,i,GREEN);
        putpixel(i,tempX,RED);
    }
}

//chuong trinh chinh
int main()
{
    nhapdulieu();
    initwindow(800,800);
    draw_object();
    delay(1000);

    FlipCordinate();
    draw_object();

    getch();
    return 0;
}

```

Transformation 2D

4. Lật qua 2 trục x và y

```

#include <graphics.h>
#include <math.h>
#include <stdio.h>

//khai bao bien
int n;
int xc[100];
int yc[100];
int xs;
int ys;
int degree;
float scalex;
float scaley;

//chuong trinh con
void draw_object()
{
    int i;

    for(i=0;i<n;i++)
    {
        line(xc[i],yc[i],xc[(i+1)%n],yc[(i+1)%n]);
        delay(1000);
    }
}

//nhap thong so
void nhapdulieu()
{
    int i;

    printf("Nhập số cạnh của đa giác = ");
    scanf("%d",&n);

    for (i=0;i<n;i++)
    {
        printf("Toa độ x cho cạnh %d = ",i);
        scanf("%d",&xc[i]);
        printf("Toa độ y cho cạnh %d = ",i);
        scanf("%d",&yc[i]);
    }
}

//Lật theo trục y qua đường thẳng y=x
void FlipV()
{
    int tempY;
    int i;

    tempY=getmaxx()/2;
    for(i=0;i<n;i++)
    {
        xc[i]=tempY+(tempY-xc[i]);
    }

    //Ve duong phan cach theo chieu doc
    for(i=0;i<getmaxy();i++)
    {
        putpixel(tempY,i,RED);
    }
}

//Lật theo trục x=y
void FlipH()
{
    int tempX;
    int i;

    tempX=getmaxy()/2;
    for(i=0;i<n;i++)
    {
        yc[i]=tempX+(tempX-yc[i]);
    }

    //Ve duong phan cach theo chieu doc
    for(i=0;i<getmaxx();i++)
    {
        putpixel(i,tempX,RED);
    }
}

//chuong trinh chinh
int main()
{
    nhapdulieu();
    initwindow(800,800);
    draw_object();
    delay(1000);

    FlipV();
    draw_object();

    FlipH();
    draw_object();

    getch();
    return 0;
}

```

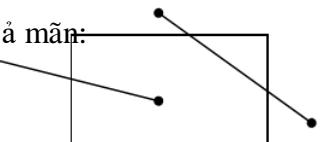
Clipping Point, Line, Polygon

Giải thuật Cohen-Sutherland

Giả sử (x, y) là tọa độ của một điểm, vậy điểm đó được hiển thị khi thoả mãn:

$$\triangleright \quad X_{W\min} \leq x \leq X_{W\max}$$

$$\triangleright \quad Y_{W\min} \leq y \leq Y_{W\max}$$

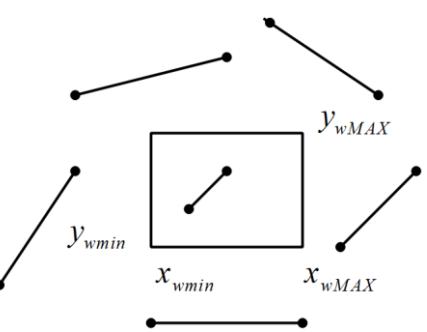


Ý tưởng: Các đoạn thẳng có thể rơi vào các trường hợp sau:

\triangleright Hiển thị (visible): cả hai đầu cuối của đoạn thẳng đều nằm bên trong cửa sổ

\triangleright Không hiển thị (invisible): đoạn thẳng xác định nằm ngang qua đoạn thẳng từ (x_1, y_1) đến (x_2, y_2) thoả mãn bất kỳ 1 troi

$$\begin{cases} x_1 < x_{w\min} \text{ và } x_2 < x_{w\min} \\ x_1 > x_{w\max} \text{ và } x_2 > x_{w\max} \\ y_1 < y_{w\min} \text{ và } y_2 < y_{w\min} \\ y_1 > y_{w\max} \text{ và } y_2 > y_{w\max} \end{cases}$$

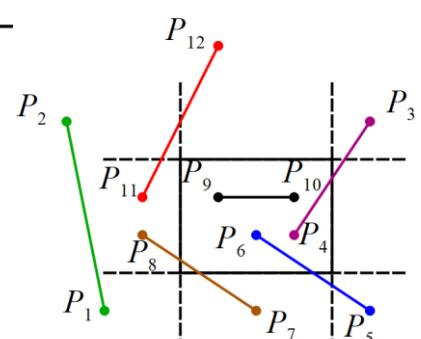
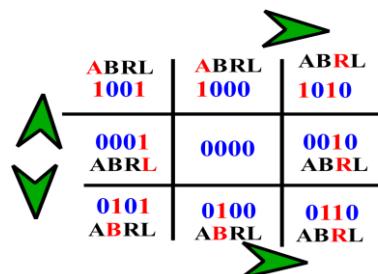
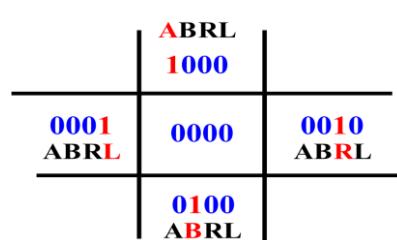


Giải thuật Cohen – Sutherland

Bước 1. Mã hóa các điểm

Gán mã vùng 4-bit cho mỗi điểm cuối của đoạn thẳng ABRL – Above Below – Right - Left

Mã vùng được xác định theo 9 vùng của mặt phẳng mà các điểm cuối nằm vào đó. Một bít được cài đặt true (1) hoặc false (0).



Ví dụ:

$$\text{Đoạn } P_1P_2 \begin{cases} P_1: 0101 \\ P_2: 1001 \end{cases}$$

$$\text{Đoạn } P_7P_8 \begin{cases} P_7: 0100 \\ P_8: 0001 \end{cases}$$

$$\text{Đoạn } P_3P_4 \begin{cases} P_3: 1010 \\ P_4: 0000 \end{cases}$$

$$\text{Đoạn } P_9P_{10} \begin{cases} P_9: 0000 \\ P_{10}: 0000 \end{cases}$$

$$\text{Đoạn } P_5P_6 \begin{cases} P_5: 0110 \\ P_6: 0000 \end{cases}$$

$$\text{Đoạn } P_{11}P_{12} \begin{cases} P_{11}: 0001 \\ P_{12}: 1000 \end{cases}$$

Bước 2. Tìm số điểm giới hạn bởi đường thẳng và cửa sổ cắt

- Nếu $P_i \text{ AND } P_j \neq 0$ chứng tỏ đoạn thẳng nằm ngoài cửa sổ cắt
- Nếu $P_i \text{ AND } P_j = 0$ chứng tỏ đoạn thẳng giao với x cạnh cửa sổ cắt. Cụ thể:

Clipping Point, Line, Polygon

- Nếu $P_i = P_j = 0$ thì cả 2 điểm (đoạn thẳng) đều thuộc cửa sổ cắt (không cần tìm tọa độ giao điểm)
- Nếu $P_i \neq 0$ hoặc $P_j \neq 0$ thì đoạn thẳng giao với 1 cạnh của cửa sổ cắt (có 1 giao điểm)
- Nếu $P_i \neq 0$ và $P_j \neq 0$ thì đoạn thẳng giao với 2 cạnh của cửa sổ cắt (có 2 giao điểm)

Chú ý: Trong trường hợp đoạn $P_{11}P_{12}$, mặc dù cả 2 mã hóa đều khác 0 và AND bit bằng 0. Nhưng thực tế đoạn thẳng này nằm bên ngoài cửa sổ cắt. Do đó, giải thuật được điều chỉnh: tìm tọa độ giao điểm – sau đó kiểm tra hoành độ và tung độ, nếu tọa độ giao điểm nằm ngoài cửa sổ cắt thì tiến hành loại bỏ các điểm này ra khỏi kết quả.

A AND B	Sơ đồ mạch điện:
0 AND 0: 0	
0 AND 1: 0	
1 AND 0: 0	
1 AND 1: 1	0: mạch hở / không có điện 1: ngắn mạch / dẫn điện

$P_1: 0101$ $P_2: 1001$ $P_1 \text{AND } P_2: \frac{P_1: 0101}{0001} \neq 0 \quad \rightarrow$ <p style="margin-left: 100px;">Không có giao điểm</p>	$P_7: 0100$ $P_8: 0001$ $P_7 \text{AND } P_8: \frac{P_7: 0100}{0000} = 0 \quad \rightarrow$ <p style="margin-left: 100px;">Có 2 giao điểm</p>
$P_3: 1010$ $P_4: 0000$ $P_3 \text{AND } P_4: \frac{P_3: 1010}{0000} = 0 \quad \rightarrow$ <p style="margin-left: 100px;">Có 1 giao điểm</p>	$P_9: 0000$ $P_{10}: 0000$ $P_9 \text{AND } P_{10}: \frac{P_9: 0000}{0000} = 0 \quad \rightarrow$ <p style="margin-left: 100px;">Thuộc cửa sổ cắt</p>
$P_5: 0110$ $P_6: 0000$ $P_5 \text{AND } P_6: \frac{P_5: 0110}{0000} = 0 \quad \rightarrow$ <p style="margin-left: 100px;">Có 1 giao điểm</p>	$P_{11}: 0001$ $P_{12}: 1000$ $P_{11} \text{AND } P_{12}: \frac{P_{11}: 0001}{0000} = 0 \quad \rightarrow$ <p style="margin-left: 100px;">Có 2 giao điểm</p>

Bước 3. Xác định hoành độ/Tung độ giao điểm

$$m = \frac{(y - y_1)}{(x - x_1)} \quad \text{hay } y - y_1 = m(x - x_1)$$

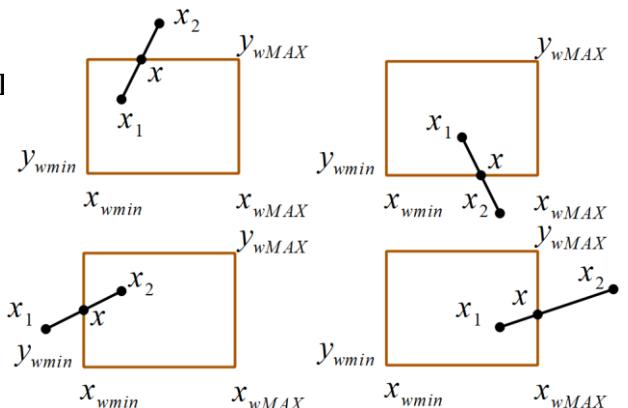
$$\text{Suy ra: } x = x_1 + \frac{(y - y_1)}{m} \quad \text{hay } y = y_1 + m(x - x_1)$$

Tù bộ 4 mã hóa: ABRL ta có (ưu tiên các bit từ A - 1

$$\text{Nếu bit } A = 1 \text{ thì } \begin{cases} x = x_1 + \frac{(y - y_1)}{m} \\ y = y_{wMAX} \end{cases}$$

$$\text{Nếu bit } B = 1 \text{ thì } \begin{cases} x = x_1 + \frac{(y - y_1)}{m} \\ y = y_{wmin} \end{cases}$$

$$\text{Nếu bit } R = 1 \text{ thì } \begin{cases} x = x_{wMAX} \\ y = y_1 + m(x - x_1) \end{cases}$$



Clipping Point, Line, Polygon

Nếu bit $L = 1$ thì $\begin{cases} x = x_{wmin} \\ y = y_1 + m(x - x_1) \end{cases}$

Hệ số góc (m) tại giao điểm: $m = \frac{(y-y_1)}{(x-x_1)} = \frac{(y_2-y_1)}{(x_2-x_1)}$

- Kiểm tra điều kiện thỏa mãn nếu rơi vào trường hợp đặc biệt như $P_{11} - P_{12}$

- Nếu $x < x_{wmin} \rightarrow$ loại
- Nếu $x > x_{wMAX} \rightarrow$ loại
- Nếu $y < y_{wmin} \rightarrow$ loại
- Nếu $y > y_{wMAX} \rightarrow$ loại

Bài tập áp dụng :

Cho cửa sổ cắt giới hạn bởi:

$$x_{wmin} = 50, x_{wMAX} = 100, y_{wmin} = 50, y_{wMAX} = 80$$

Cho đa giác giới hạn bởi các đỉnh:

$$A(40,90); B(60,100); C(70,70); D(110,70); E(40,40)$$

Xác định tọa độ giao điểm giới hạn bởi cửa sổ cắt và đa giác.

Bước 1. Xây dựng bộ 4 mã hóa và tìm số giao điểm

$A: 1001$	$C: 0000$
$B: 1000$	$D: 0010$
$\&: 1000$	$\&: 0000$
$B: 1000$	$D: 0010$
$C: 0000$	$E: 0101$
$\&: 0000$	$\&: 0000$
$E: 0101$	$D: 0010$
$A: 1001$	$E: 0101$
$\&: 0001$	$\&: 0000$

Không có giao điểm Có 1 giao điểm

Có 1 giao điểm Có 2 giao điểm

Không có giao điểm

Bước 2. Tìm tọa độ giao điểm theo thứ tự ưu tiên ABRL

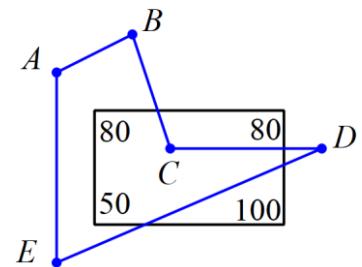
- Xét cạnh BC có $m_{BC} = \frac{(70 - 100)}{(70 - 60)} = -3$

o B (1000) nên: $\begin{cases} x = x_B + \frac{(y - y_B)}{m} \\ y = y_{wMAX} \end{cases}$

hay $\begin{cases} x = 60 + \frac{(80 - 100)}{-3} \sim 67 \\ y = 80 \end{cases}$

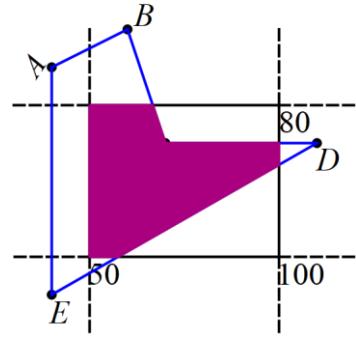
- Xét cạnh CD có $m_{CD} = \frac{(70 - 70)}{(110 - 70)} = 0$

o D (0001) nên: $\begin{cases} x = x_{wMAX} \\ y = y_D + m(x - x_D) \end{cases}$



Clipping Point, Line, Polygon

- hay $\begin{cases} x = 100 \\ y = 70 + 0(100 - 110) = 70 \end{cases}$
- Xét cạnh DE có $m_{DE} = (40 - 70)/(40 - 110) = 3/7$
 - D (0001) nên: $\begin{cases} x = x_{wMAX} \\ y = y_D + m(x - x_D) \end{cases}$
 - hay $\begin{cases} x = 100 \\ y = 70 + (110 - 100) * 3/7 \sim 66 \end{cases}$
 - E (0110) nên: $\begin{cases} x = x_1 + \frac{(y - y_1)}{m} \\ y = y_{wmin} \end{cases}$
 - hay $\begin{cases} x = 40 + (50 - 40)/3/7 \sim 63 \\ y = 50 \end{cases}$



Bài tập 2.

Cho cửa sổ cắt giới hạn bởi:

$$x_{wmin} = 50, x_{wMAX} = 80, y_{wmin} = 50, y_{wMAX} = 80$$

Cho đa giác giới hạn bởi các đỉnh:

$$A(30,70); B(60,100); C(60,40); D(30,20)$$

Xác định tọa độ giao điểm giới hạn bởi cửa sổ cắt và đa giác.

Bước 1. Xây dựng bộ 4 mã hóa và tìm số giao điểm

$$\begin{array}{l} A: 0001 \\ B: 1000 \\ \&: 0000 \end{array} \rightarrow \text{Có 2 giao điểm}$$

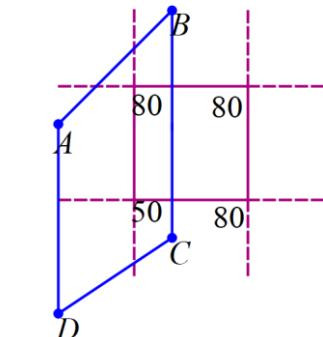
$$\begin{array}{l} C: 0100 \\ D: 0101 \\ \&: 0100 \end{array} \rightarrow \text{Không giao điểm}$$

$$\begin{array}{l} B: 1000 \\ C: 0100 \\ \&: 0000 \end{array} \rightarrow \text{Có 2 giao điểm}$$

$$\begin{array}{l} D: 0101 \\ A: 0001 \\ \&: 0001 \end{array} \rightarrow \text{Không có giao điểm}$$

Bước 2. Tìm tọa độ giao điểm theo thứ tự ưu tiên ABRL

- Xét cạnh AB có $m_{AB} = (100 - 70)/(60 - 30) = 1$
 - B (1000) nên: $\begin{cases} x = x_B + \frac{(y - y_B)}{m} \\ y = y_{wMAX} \end{cases}$
 - hay $\begin{cases} x = 60 + (80 - 100)/1 = 40 < x_{wmin}: \text{loại} \\ y = 80 \end{cases}$
 - A(0001) nên: $\begin{cases} x = x_{wmin} \\ y = y_A + m(x - x_A) \end{cases}$
 - hay $\begin{cases} x = 50 \\ y = 70 + 1(50 - 30) = 90 > y_{wMAX}: \text{loại} \end{cases}$



Clipping Point, Line, Polygon

- Xét cạnh BC có m_{BC} có $dx = 0$

- B (1000) nên: $\begin{cases} x = x_B \\ y = y_{wMAX} \end{cases}$

Hay $\begin{cases} x = 60 \\ y = 80 \end{cases}$

- C (0100) nên: $\begin{cases} x = x_C \\ y = y_{wmin} \end{cases}$

Hay $\begin{cases} x = 60 \\ y = 50 \end{cases}$

Bài tập 3 .

Cho cửa sổ cắt giới hạn bởi:

$$x_{wmin} = 50, x_{wMAX} = 100, y_{wmin} = 50, y_{wMAX} = 90$$

Cho đa giác giới hạn bởi các đỉnh:

$$A(40,80); B(110,80); C(80,40)$$

Xác định tọa độ giao điểm giới hạn bởi cửa sổ cắt và đa giác.

Bước 1. Xây dựng bộ 4 mã hóa và tìm số giao điểm

$$\begin{array}{l} A: 0001 \\ B: 0010 \\ \&: 0000 \end{array} \quad \rightarrow \text{Có 2 giao điểm}$$

$$\begin{array}{l} C: 0100 \\ A: 0001 \\ \&: 0000 \end{array} \quad \rightarrow \text{Có 2 giao điểm}$$

$$\begin{array}{l} B: 0010 \\ C: 0100 \\ \&: 0000 \end{array} \quad \rightarrow \text{Có 2 giao điểm}$$

Bước 2. Tìm tọa độ giao điểm theo thứ tự ưu tiên ABRL

- Xét cạnh AB có $m_{AB} = (80 - 80) / (100 - 40) = 0$

- B (0010) nên: $\begin{cases} y = y_B + m(x - x_B) \\ x = x_{wMAX} \end{cases}$

hay $\begin{cases} y = 80 + (80 - 110)0 = 80 \\ x = 100 \end{cases}$

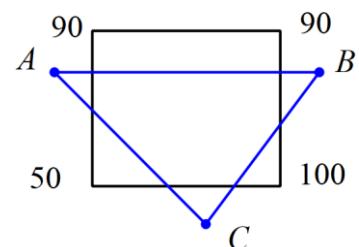
- A(0001) nên: $\begin{cases} x = x_{wmin} \\ y = y_A + m(x - x_A) \end{cases}$

hay $\begin{cases} x = 50 \\ y = 80 + 0(50 - 40) = 90 \end{cases}$

- Xét cạnh BC có $m_{BC} = (40 - 80) / (80 - 110) = 4/3$

- B (0010) nên: $\begin{cases} y = y_B + m(x - x_B) \\ x = x_{wMAX} \end{cases}$

Hay $\begin{cases} x = 100 \\ y = 80 + \frac{4}{3} * (100 - 110) \sim 67 \end{cases}$



Clipping Point, Line, Polygon

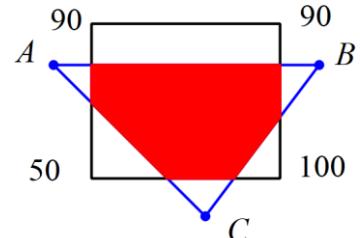
- C (0100) nêu: $\begin{cases} x = xC + \frac{(y - yC)}{m} \\ y = y_{wmin} \end{cases}$
 Hay $\begin{cases} x = 80 + (50 - 40):(4/3) \sim 88 \\ y = 50 \end{cases}$

- Xét cạnh AC có $m_{AC} = \frac{(40 - 80)}{(80 - 40)} = -1$
 - C (0100) nêu: $\begin{cases} x = xC + \frac{(y - yC)}{m} \\ y = y_{wmin} \end{cases}$
 Hay $\begin{cases} x = 80 + \frac{(50 - 40)}{-1} = 70 \\ y = 50 \end{cases}$
 - A(0001) nêu: $\begin{cases} x = x_{wmin} \\ y = yA + m(x - xA) \end{cases}$
 hay $\begin{cases} x = 50 \\ y = 80 + (50 - 40) * -1 = 70 \end{cases}$

Có thể rút ra nhận xét:

Trong trường hợp ABRL của ($P_1 \neq 0$) và ($P_2 \neq 0$) và ($P_1 \text{ AND } P_2 = 0$) và ($dy = 0$) thì:

$$\begin{cases} x = x_{wmin} \\ y = yP_1 \end{cases} \text{ và } \begin{cases} x = x_{wmax} \\ y = yP_2 \end{cases}$$



Lập trình mô phỏng:

```
#include<stdio.h>
#include<graphics.h>
#define maxgd 100
#include<math.h>

int xwmin = 50;
int xwmax = 100;
int ywmin = 50;
int ywmax = 80;

int xa;
int ya;
int xb;
int yb;
float m;

int X;
int Y;
int X1;
int X2;
int Y1;
int Y2;

int mahoadinh[2];

void nhaptoado()
{
    printf("Nhập tọa độ 2 điểm của đường thang: ");
    printf("xa = "); scanf("%d", &xa);
    printf("ya = "); scanf("%d", &ya);
    printf("xb = "); scanf("%d", &xb);
    printf("yb = "); scanf("%d", &yb);

    // Vẽ cua so cat va duong thang AB
    initwindow(400, 400);
    setcolor(RED);
    line(xwmin, ywmin, xwmax, ywmin);
    line(xwmax, ywmin, xwmax, ywmax);
    line(xwmax, ywmax, xwmin, ywmax);
    line(xwmin, ywmax, xwmin, ywmin);

    setcolor(BLUE);
    line(xa, ya, xb, yb);
}

int ABRL(int tmpx, int tmpy)
{
    int code;
    // Ben trong
    if(xwmin<=tmpx && tmpx<=xwmax && ywmin<=tmpy && tmpy<=ywmax)
    {
        code=0;
    }
    // Ben trai
    if(tmpx<xwmin && ywmin<tmpy && tmpy<ywmax)
    {
        code=1;
    }
    // Ben phai
    if(tmpx>xwmax && ywmin<=tmpy && tmpy<=ywmax)
    {
```

Clipping Point, Line, Polygon

```
        code=2;
    }
//ben tren
if(tmpy>ywmax && tmpx>xwmin && tmpx<xwmax)
{
    code=8;
}
//ben duoi
if(tmpy<ywmin && tmpx>xwmin && tmpx<xwmax)
{
    code=4;
}
//goc duoi ben trai
if(tmpy<ywmin && tmpx<xwmin)
{
    code=5;
}
//goc duoi ben phai
if(tmpy<ywmin && tmpx>xwmax)
{
    code=6;
}
//goc tren ben trai
if(tmpy>ywmax && tmpx<xwmin)
{
    code=9;
}
//goc tren ben phai
if(tmpy>ywmax && tmpx>xwmax)
{
    code=10;
}
return code;
}

void mahoavasogiaodiem()
{
    mahoadinh[0]=ABRL(xa,ya);
    mahoadinh[1]=ABRL(xb,yb);
    printf("\nMa hoa diem A: %d",ABRL(xa,ya));
    printf("\nMa hoa diem B: %d",ABRL(xb,yb));

    if((mahoadinh[0]& mahoadinh[1])!=0)
    {
        printf("\nDoan thang nam ben ngoai cua so");
    }
    else
    {
        if(mahoadinh[0]==0 && mahoadinh[1]==0)
        {
            printf("\nDoan thang nam ben trong cua so");
        }
        else
        {
            if((mahoadinh[0]!=0 && mahoadinh[1])!=0)
            {
                printf("\nCo 2 giao diem");
            }
            else
            {
                printf("\nCo 1 giao diem");
            }
        }
    }
}
```

Clipping Point, Line, Polygon

```
        }
    }

void cohen_sutherland(int CODE)
{
    X=0;
    Y=0;
    if(CODE==1) //diem nam ben trai L=1
    {
        X=xwmin;
        Y=round(ya+m*(X-xa));
    }
    if(CODE==2) //diem nam ben phai R=1
    {
        X=xwmax;
        Y=round(ya+m*(X-xa));
    }
    if(CODE==4) //diem nam ben duoi B=1
    {
        Y=ywmin;
        X=round(xa+(Y-ya)/m);
    }
    if(CODE==8) //diem nam ben tren A=1
    {
        Y=ywmax;
        X=round(xa+(Y-ya)/m);
    }

    if(CODE==9) //diem nam tren ben trai A=1 va L=1 (CODE 1)
    {
        X=xwmin;
        Y=round(ya+m*(X-xa));
    }
    if(CODE==10) //diem nam tren ben phai A=1 va R=1 (CODE 8)
    {
        Y=ywmax;
        X=round(xa+(Y-ya)/m);
    }
    if(CODE==5) //diem nam duoi ben trai B=1 va L=1 (CODE 1)
    {
        X=xwmin;
        Y=round(ya+m*(X-xa));
    }
    if(CODE==6) //diem nam duoi ben phai B=1 va R=1 (CODE 4)
    {
        Y=ywmin;
        X=round(xa+(Y-ya)/m);
    }

    //kiem tra dieu kien 2 dinh deu nam ngoai cua so cat
    if(X<xwmin|| X>xwmax)
    {
        X=0;
    }
    if(Y<ywmin|| Y>ywmax)
    {
        Y=0;
    }
}

void timgiaodiem()
{
    //ca 2 diem deu nam ben trong cua so cat
```

Clipping Point, Line, Polygon

```
if (mahoadinh[0]==0 && mahoadinh[1]==0)
{
    X1=xa;Y1=ya;X2=xb;Y2=yb;
}

//Khi dy=0, duong thang song song voi truc x
if(ya==yb)
{
    //diem a nam ben trong va diem b nam ben ngoai
    if(mahoadinh[0]==0 && mahoadinh[1]!=0)
    {
        //diem b nam ben trai cua so cat
        if(xa>xb)
        {
            X1=xa;Y1=ya;X2=xwmin;Y2=ya;//Y2=yb
        }
        //diem b nam ben phai cua so cat
        if(xa<xb)
        {
            X1=xa;Y1=ya;X2=xwmax;Y2=ya;//Y2=yb
        }
    }
    //diem b nam ben trong va diem a nam ben ngoai
    if(mahoadinh[0]!=0 && mahoadinh[1]==0)
    {
        //diem a nam ben trai cua so cat
        if(xa<xb)
        {
            X1=xb;Y1=yb;X2=xwmin;Y2=ya;//Y2=yb
        }
        //diem a nam ben phai cua so cat
        else
        {
            X1=xb;Y1=yb;X2=xwmax;Y2=ya;//Y2=yb
        }
    }
    //ca 2 diem deu ben ngoai
    else
    {
        X1=xwmin;Y1=ya;X2=xwmax;Y2=ya;//Y2=yb
    }
}

//Khi dx==0, doan thang song song voi truc y
if(xa==xb)
{
    //diem a nam ben trong va diem b nam ben ngoai
    if(mahoadinh[0]==0 && mahoadinh[1]!=0)
    {
        //diem b nam ben tren cua so cat
        if(ya>yb)
        {
            X1=xa;Y1=ya;X2=xa;Y2=ywmin;//X2=xb
        }
        //diem b nam ben duoi cua so cat
        else
        {
            X1=xa;Y1=ya;X2=xa;Y2=ywmax;//X2=yb
        }
    }
    //diem b nam ben trong va diem a nam ben ngoai
    if(mahoadinh[0]!=0 && mahoadinh[1]==0)
    {
```

Clipping Point, Line, Polygon

```
//diem a nam ben tren cua so cat
if(ya<yb)
{
    X1=xb;Y1=yb;Y2=ywmin;X2=xa;//X2=xb
}
//diem a nam ben duoi cua so cat
else
{
    X1=xb;Y1=yb;Y2=ywmax;X2=xa;//X2=xb
}
}
//ca 2 diem deu ben ngoai
else
{
    X1=xa;Y1=ywmin;X2=xb;Y2=ywmax;//X2=xa
}

//khi ca dx va dy deu <> 0, co 2 diem cat
if(mahoadinh[0]!=0 && mahoadinh[1]!=0 && (mahoadinh[0] &
mahoadinh[1])==0)
{
    m=(float) (yb-ya) / (xb-xa);
    cohen_sutherland(mahoadinh[0]);
    X1=X;
    Y1=Y;
    cohen_sutherland(mahoadinh[1]);
    X2=X;
    Y2=Y;
    printf("\nX1= %d, Y1= %d, X2= %d, Y2= %d ",X1,Y1,X2,Y2);
}

//ve duong thang noi 2 diem
setcolor(GREEN);
setlinestyle(1,0,1);
line(X1,Y1,X2,Y2);
}

int main()
{
    nhaptoado();
    mahoavasogiaodiem();
    timgiaodiem();
    getch();
}
```

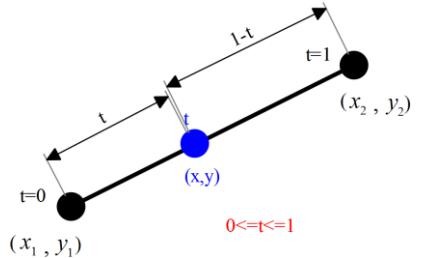
Clipping Point, Line, Polygon

Thuật toán Liang – Barsky

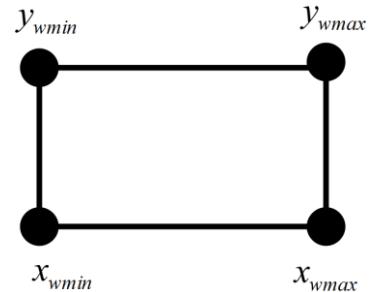
Xét đường thẳng đi qua 2 điểm (x_1, y_1) và (x_2, y_2) như sau:

Gọi $t=0$ tại điểm (x_1, y_1) và $t=1$ tại điểm (x_2, y_2)

Xét 1 điểm thuộc đường thẳng có tọa độ (x, y) . Khi đó:



- $x = t \cdot x_2 + (1 - t)x_1 = t \cdot x_2 + x_1 - t \cdot x_1 = x_1 + t(x_2 - x_1) = x_1 + t \cdot \Delta x$
- $y = y_1 + t \cdot \Delta y$; với $0 < t < 1$
- Xét cửa sổ clip được giới hạn bởi (x_{wmin}, y_{wmin}) , (x_{wmax}, y_{wmax})
- Do vậy:
 - $x_{wmin} \leq x \leq x_{wmax}$
 - $y_{wmin} \leq y \leq y_{wmax}$
- Hay:
 - $x_{wmin} \leq x_1 + t \cdot \Delta x \leq x_{wmax}$
 - $y_{wmin} \leq y_1 + t \cdot \Delta y \leq y_{wmax}$
- Ta có 4 bất đẳng thức sau:
 - $x_1 + t \cdot \Delta x \geq x_{wmin}$
 - $x_1 + t \cdot \Delta x \leq x_{wmax}$
 - $y_1 + t \cdot \Delta y \geq y_{wmin}$
 - $y_1 + t \cdot \Delta y \leq y_{wmax}$



Hay

- $t \cdot \Delta x \geq x_{wmin} - x_1$
- $t \cdot \Delta x \leq x_{wmax} - x_1$
- $t \cdot \Delta y \geq y_{wmin} - y_1$
- $t \cdot \Delta y \leq y_{wmax} - y_1$

Nhân 2 vế với giá trị -1:

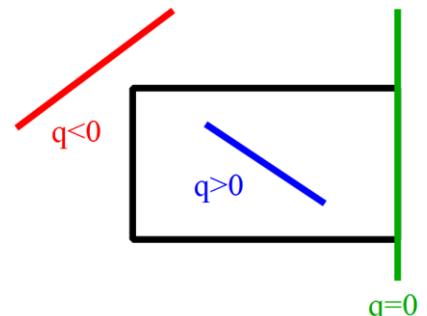
- $-t \cdot \Delta x \leq x_1 - x_{wmin}$
- $t \cdot \Delta x \leq x_{wmax} - x_1$
- $-t \cdot \Delta y \leq y_1 - y_{wmin}$
- $t \cdot \Delta y \leq y_{wmax} - y_1$
- Tổng quát ta có: $t \cdot p_k \leq q_k$ với $k = 1, 2, 3, 4$
- Từ đó, ta có:

- $p_1 = -\Delta x$
- $p_2 = \Delta x$
- $p_3 = -\Delta y$
- $p_4 = \Delta y$

- $q_1 = x_1 - x_{wmin}$
- $q_2 = x_{wmax} - x_1$
- $q_3 = y_1 - y_{wmin}$
- $q_4 = y_{wmax} - y_1$

Clipping Point, Line, Polygon

- Nếu $p_k = 0$: điều đó tương đương với việc đoạn thẳng đang xét song song với cạnh thứ k của hình chữ nhật clipping (đường thẳng có $dy = 0$ hoặc $dx = 0$)



- ❶ Nếu $q_k < 0$ đoạn thẳng nằm ngoài cửa sổ (hệ bất phương trình trên vô nghiệm)
- ❷ Nếu $q_k > 0$ thì đoạn thẳng nằm trong 1 phần của cửa sổ clipping
- ❸ Nếu $q_k = 0$ thì đoạn thẳng nằm trên đường biên của cạnh

- Nếu $p_k < 0$, từ bất đẳng thức $t.p_k \leq qx \rightarrow$ xác định t_1 .

❹ Ta có: $t_1 = \max(0, \frac{q_k}{p_k})$

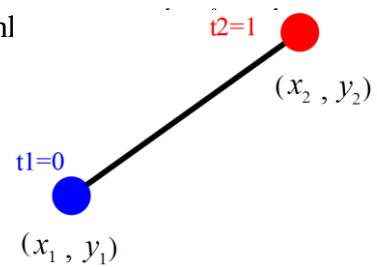
$$x = x_1 + t_1 \cdot \Delta x; y = y_1 + t_1 \cdot \Delta y$$

- Nếu $p_k > 0$, từ bất đẳng thức $t.p_k \leq qx \rightarrow$ xác định t_2 .

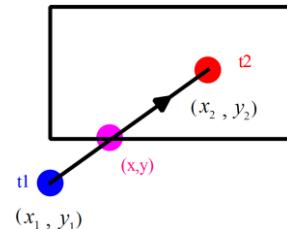
❺ Ta có: $t_2 = \min(1, \frac{q_k}{p_k})$

$$x = x_1 + t_2 \cdot \Delta x; y = y_1 + t_2 \cdot \Delta y$$

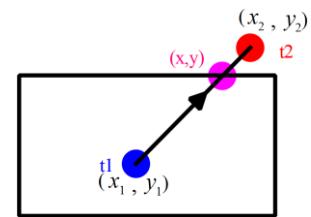
- Nếu $t_1 < 0$ và $t_2 > 1$ thì cả 2 điểm đều nằm bên ngoài cửa sổ clipping
- Nếu $t_1 = 0$ và $t_2 = 1$ thì cả 2 điểm đều nằm bên trong cửa sổ clipping
- Nếu $0 < t_1$ và $t_2 < 1$ thì cả 2 điểm đều cần được xem xét
- Nhận xét: $t_1 = \max(0, \frac{q_k}{p_k}); t_2 = \min(1, \frac{q_k}{p_k})$



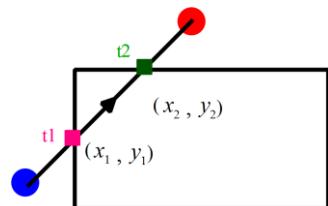
- Nếu t_1 thay đổi giá trị tăng dần từ 0 đến $\max(\frac{q_k}{p_k})$, có nghĩa là đường thẳng có chiều từ ngoài vào trong cửa sổ clipping.



- Nếu t_2 thay đổi giá trị giảm dần từ 1 đến $\min(\frac{q_k}{p_k})$, có nghĩa là đường thẳng có chiều trong cửa sổ clipping ra ngoài.



- Nếu t_1 thay đổi giá trị tăng dần từ 0 đến $\max(\frac{q_k}{p_k})$ còn t_2 thay đổi giá trị giảm dần từ 1 đến $\min(\frac{q_k}{p_k})$, có nghĩa là đường thẳng có chiều từ ngoài cửa sổ clipping ra bên ngoài cửa sổ clipping.



Bài tập áp dụng :

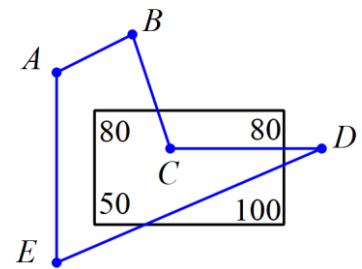
Cho cửa sổ cắt giới hạn bởi:

$$x_{wmin} = 50, x_{wMAX} = 100, y_{wmin} = 50, y_{wMAX} = 80$$

Cho đa giác giới hạn bởi các đỉnh:

$$A(40,90); B(60,100); C(70,70); D(110,70); E(40,40)$$

Xác định tọa độ giao điểm giới hạn bởi cửa sổ cắt và đa giác.



Xét cạnh BC:

- $p1 = -\Delta x = -10$
- $q1 = x1 - xwmin = 10$
- $p2 = \Delta x = 10$
- $q2 = xwmax - x1 = 40$
- $p3 = -\Delta y = 30$
- $q3 = y1 - ywmin = 50$
- $p4 = \Delta y = -30$
- $q4 = ywmax - y1 = -20$

➤ Xét trường hợp $Pk < 0$ (ứng với P1 và P4)

$$t1 = \max\left(0, \frac{q1}{p1}, \frac{q4}{p4}\right) = \max\left(0, -1, \frac{2}{3}\right) = 2/3$$

$$x = x1 + t1 \cdot \Delta x = 60 + \frac{2}{3} * 10 = 67 \quad ; y = y1 + t1 \cdot \Delta y = 100 + \frac{2}{3} * -30 = 80$$

➤ Xét trường hợp $Pk > 0$ (ứng với P2 và P3)

$$t2 = \min\left(1, \frac{q2}{p2}, \frac{q3}{p3}\right) = \min\left(1, \frac{5}{3}, \frac{2}{3}\right) = 1 \text{ (Không xét)}$$

Xét cạnh CD:

$$\text{Do } dy = 0, \begin{cases} x = x_{wmax} = 100 \\ y = y_C = 70 \end{cases}$$

Xét cạnh DE:

- $p1 = -\Delta x = 70$
- $q1 = x1 - xwmin = 60$
- $p2 = \Delta x = -70$
- $q2 = xwmax - x1 = -10$
- $p3 = -\Delta y = 30$
- $q3 = y1 - ywmin = 20$
- $p4 = \Delta y = -30$
- $q4 = ywmax - y1 = 10$

➤ Xét trường hợp $Pk < 0$ (ứng với P2, P4)

$$t1 = \max\left(0, \frac{q2}{p2}, \frac{q4}{p4}\right) = \max\left(0, \frac{1}{7}, \frac{-1}{3}\right) = 1/7$$

$$x = x1 + t1 \cdot \Delta x = 110 + \frac{1}{7} * -70 = 100 \quad ; y = y1 + t1 \cdot \Delta y = 70 + \frac{1}{7} * -30 = \sim 66$$

➤ Xét trường hợp $Pk > 0$ (ứng với P1, P3)

$$t2 = \min\left(1, \frac{q1}{p1}, \frac{q3}{p3}\right) = \min\left(1, \frac{6}{7}, \frac{2}{3}\right) = 2/3$$

$$x = x1 + t2 \cdot \Delta x = 110 + \frac{2}{3} * -70 \sim 63; y = y1 + t2 \cdot \Delta y = 70 + \frac{2}{3} * -30 = 50$$

Lập trình mô phỏng

```
#include<stdio.h>
#include<graphics.h>
#include<math.h>

int xmin;
int xmax;
int ymin;
int ymax;

int xa;
int ya;
int xb;
int yb;

int dx;
int dy;

int p[4];
int q[4];

int X1;
int Y1;
int X2;
int Y2;

void nhaphuongtrinh()
{
    xmin=100;
    xmax=400;
    ymin=100;
    ymax=400;

    printf("xA = "); scanf("%d", &xa);
    printf("yA = "); scanf("%d", &ya);
    printf("xB = "); scanf("%d", &xb);
    printf("yB = "); scanf("%d", &yb);

    //đường cheo cắt 2 điểm

    //xa=60;
    //ya=40;
    //xb=110;
    //yb=60;

    //xa=70;
    //ya=90;
    //xb=110;
    //yb=60;

    //xa=70;
    //ya=90;
    //xb=30;
    //yb=60;

    //xa=60;
    //ya=40;
    //xb=30;
    //yb=60;

    //song song trục hoành
```

Clipping Point, Line, Polygon

```
//xa=30;
//ya=60;
//xb=70;
//yb=60;

//xa=70;
//ya=120;
//xb=70;
//yb=60;

//tinh gia tri pk va qk
dx=xb-xa;
dy=yb-ya;

p[0]=-dx;
p[1]=dx;
p[2]=-dy;
p[3]=dy;

q[0]=xa-xmin;
q[1]=xmax-xa;
q[2]=ya-ymin;
q[3]=ymax-ya;
/*
printf("P[i] va Q[i] la:");
int i;
for(i=0;i<4;i++)
{
    printf("\n%d \t",p[i]);
    printf("\n%d \t",q[i]);
}
*/
initwindow(500,500);

line(xmin,ymin,xmax,ymin);
line(xmax,ymin,xmax,ymax);
line(xmax,ymax,xmin,ymax);
line(xmin,ymax,xmin,ymin);

setcolor(BLUE);
line(xa,ya,xb,yb);
}

int check()
{
    if(xa<xmin && xb<xmin)
    {
        return 0;
    }
    else
    {
        if(xa>xmax && ya>ymax)
        {
            return 0;
        }
        else
        {
            if(ya<ymin && yb<ymin)
            {
                return 0;
            }
            else
            {

```

Clipping Point, Line, Polygon

```
        if(ya>ymax && yb>ymax)
        {
            return 0;
        }
        else
        {
            return 1;
        }
    }
}

float MIN(float a, float b)
{
    float min;
    float max;

    if(a<b)
    {
        min=a;
    }
    else
    {
        min=b;
    }
    return min;
}

float MAX(float a, float b)
{
    float max;

    if(a<b)
    {
        max=b;
    }
    else
    {
        max=a;
    }
    return max;
}

void Liang_Barsky()
{
    int i;
    float t1;
    float t2;

    t1=0.0;
    t2=1.0;

    //printf("\nCheck = %d", check());

    if(check() !=0)
    {
        for(i=0;i<4;i++)
        {
            if(p[i]<0)//doi voi pk<0
            {
```

Clipping Point, Line, Polygon

```
t1=MAX(t1, (float)q[i]/p[i]);
//printf("\nnp[%d]=%d q[%d]= %d
%.2f", i, p[i], i, q[i], (float)q[i]/p[i]);
}
else //doi voi pk>0
{
    t2=MIN(t2, (float)q[i]/p[i]);
    //printf("\nnp[%d]=%d q[%d]= %d
%.2f", i, p[i], i, q[i], (float)q[i]/p[i]);
}
}

printf("\nt1 = %f, t2 = %f", t1, t2);
//tim X va Y
if(dx!=0 && dy!=0) //co 2 giao diem
{
    X1=round(xa+t1*dx);
    Y1=round(ya+t1*dy);
    X2=round(xa+t2*dx);
    Y2=round(ya+t2*dy);

    printf("\n(X1 = %d, Y1 = %d, X2 = %d, Y2 =
%d ", X1, Y1, X2, Y2);

    //kiem tra giao diem co thoat man nam trong cua so?
    if(X1<xmin || X1 >xmax || Y1 < ymin || Y1 > ymax ||
X2<xmin || X2 >xmax || Y2 < ymin || Y2 > ymax)
    {
        X1=0;
        Y1=0;
        X2=0;
        Y2=0;
    }
}
//Co 1 giao diem
else
{
    //duong thang song song truc x
    if(dy==0)
    {
        //diem a nam trong cua so
        if(xa>=xmin && xa<=xmax)
        {
            //diem b nam trong
            if(xb>=xmin && xb<=xmax)
            {
                X1=xa;Y1=ya;X2=xb;Y2=yb;
            }
            //diem b nam ben trai diem a
            if(xb<xmin)
            {
                X1=xa;Y1=ya;X2=xmin;Y2=ya;//Y2=yb
            }
            //diem b nam ben phai
            if(xb>xmax)
            {
                X1=xa;Y1=ya;X2=xmax;Y2=ya;//Y2=yb
            }
            //diem b nam ben trong cua so
            if(xb>=xmin && xb<=xmax)
```

Clipping Point, Line, Polygon

```
{  
    //diem a nam trong  
    if(xa>=xmin && xa<=xmax)  
    {  
        X1=xb;Y1=yb;X2=xa;Y2=ya;  
    }  
    //diem a nam ben trai diem a  
    if(xa<xmin)  
    {  
  
        X1=xb;Y1=yb;X2=xmin;Y2=yb;//Y2=ya  
    }  
    //diem a nam ben phai  
    if(xa>xmax)  
    {  
  
        X1=xb;Y1=yb;X2=xmax;Y2=yb;//Y2=ya  
    }  
    //ca 2 diem deu nam ben ngoai  
    if((xa<xmin && xb>xmax) || (xa>xmax &&  
xb<xmin))  
    {  
        X1=xmin;Y1=ya;X2=xmax;Y2=ya;  
    }  
    printf("\n(X1 = %d, Y1 = %d, X2 = %d, Y2 =  
%d)",X1,Y1,X2,Y2);  
}  
else  
//dx==0 //nam song song voi truc tung  
{  
    //diem a nam trong cua so  
    if(ya>=ymin && ya<=ymax)  
    {  
        //diem b nam trong  
        if(yb>=ymin && yb<=ymax)  
        {  
            X1=xa;Y1=ya;X2=xb;Y2=yb;  
        }  
        //diem b nam ben duoi diem a  
        if(yb<ymin)  
        {  
  
            X1=xa;Y1=ya;X2=xa;Y2=xb;//X2=xb  
        }  
        //diem b nam ben tren diem a  
        if(yb>ymax)  
        {  
  
            X1=xa;Y1=ya;X2=xb;//X2=xb  
        }  
    }  
    //diem b nam ben trong cua so  
    if(yb>=ymin && yb<=ymax)  
    {  
        //diem a nam trong  
        if(ya>=ymin && ya<=ymax)  
        {  
            X1=xa;Y1=ya;X2=xb;Y2=yb;  
        }  
        //diem a nam ben duoi diem b  
        if(ya<ymin)  
        {  
    }
```

Clipping Point, Line, Polygon

```
X1=xb;Y1=yb;Y2=ymin;X2=xa;//X2=xb
    }
    //diem a nam ben tren diem b
    if(ya>ymax)
    {

X1=xb;Y1=yb;Y2=ymax;X2=xa;//X2=xb
    }
    //ca 2 diem deu nam ben ngoai
    if((ya<ymin && yb>ymax) || (ya>ymax &&
yb<ymin))
    {
        X1=xa;Y1=ymin;X2=xb;Y2=ymax;
    }
}

//ve 2 giao diem
setcolor(GREEN);
setlinestyle(1,0,1);
line(X1,Y1,X2,Y2);

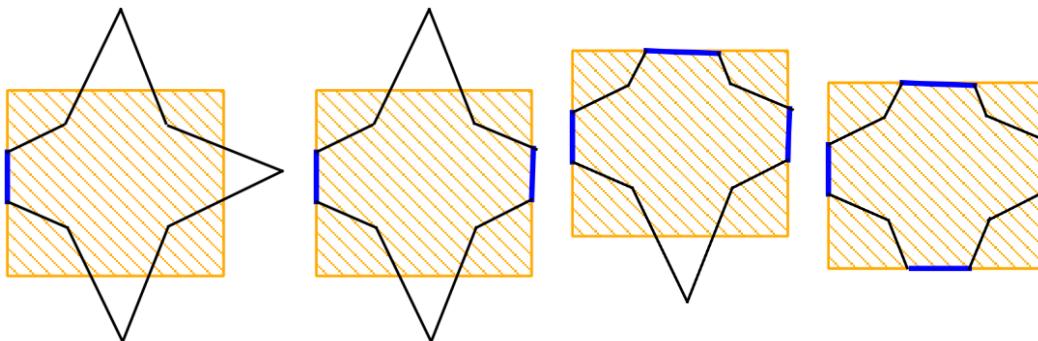
}
else
{
    printf("\nKhong co giao diem");
}
}

int main()
{
    nhapthongso();
    Liang_Barsky();
    getch();
}
```

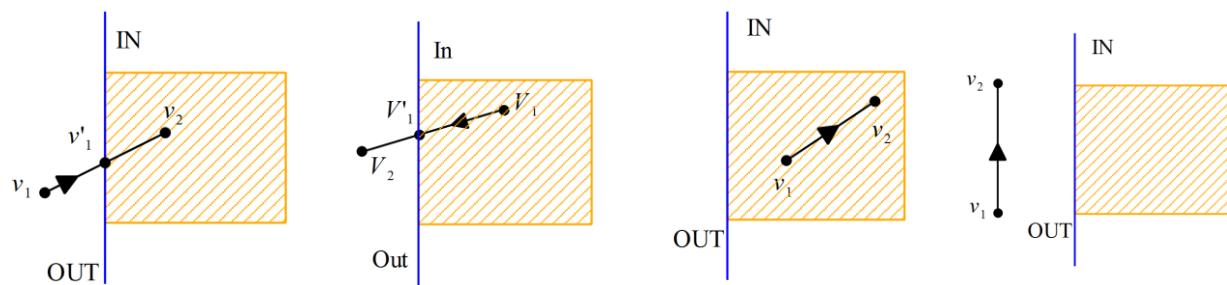
Clipping Point, Line, Polygon

Giải thuật Hodgman

Cho P1, ..., PN là danh sách các đỉnh của đa giác. Cho cửa sổ cắt tia ABCD. Thứ tự các trường hợp sẽ được xén như sau:



Xét 4 trường hợp:



Output (Out-In): V'1V2

Output (In-Out): V'1

Output (In-In): V2

Output: NULL

Bài tập:

Sử dụng thuật toán Hodgman để cắt xén đoạn thẳng nối P1(-1,2) đến P2(6,4) trên cửa sổ A(1,1), B(1,3), D(3,5) và E(5,3)

Theo thuật toán Hodgman ta xén P1P2 dựa trên từng cạnh.

❶ AB:

Điểm P1 nằm bên trái AB

Điểm P2 nằm bên phải AB

Chiều Out-IN

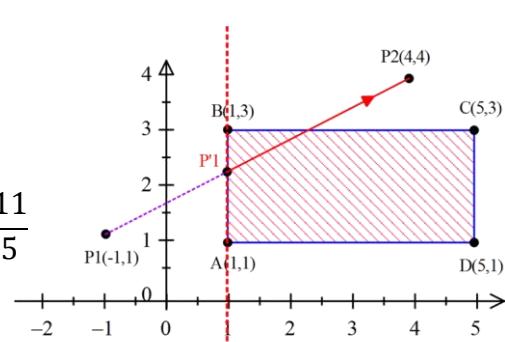
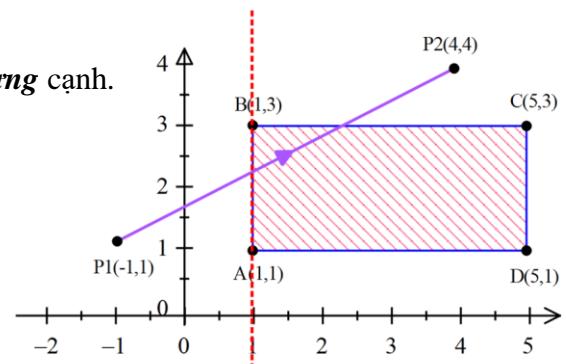
Vậy P'1 P2 được lưu

Tìm giao điểm P'1 thuộc đường thẳng P1P2

$$m = \frac{4-1}{4+1} = \frac{3}{5}$$

$$\begin{cases} x_{P'} = 1 \\ y_{P'} = y_{P1} + (x_{P'} - x_{P1})m = 1 + (1 - (-1))\frac{3}{5} = \frac{11}{5} \end{cases}$$

P'1(1;11/5)



Clipping Point, Line, Polygon

Ta xén P'1P2 dựa trên cạnh CD

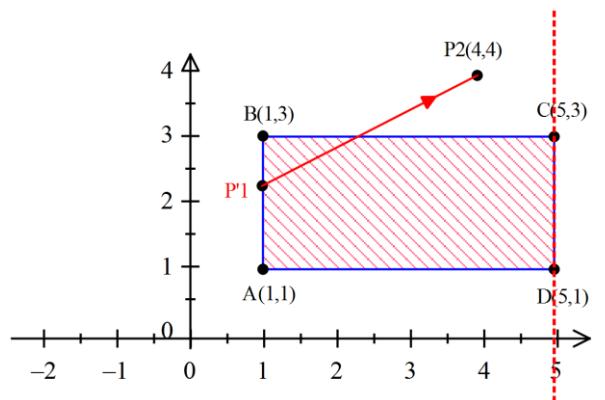
② CD:

Điểm P'1: nằm bên trái CD

Điểm P2: nằm bên trái CD

Chiều In-In

Vậy P2 được lưu, với P2(4,4)



Ta xén P'1P2 dựa trên cạnh

③ BC:

Điểm P'1: nằm bên trong BC

Điểm P2: nằm bên ngoài BC

Chiều In - Out

Vậy P''1 được lưu

Tìm giao điểm P''1 thuộc đường thẳng P'1P2

$$m = \frac{4 - 11/5}{4 - 1} = \frac{3}{5}$$

$$\begin{cases} y_{P''1} = 3 \\ x_{P''1} = xP'1 + (y_{P''1} - yP'1)/m = 1 + (3 - 11/5)\frac{5}{3} = \frac{7}{3} \end{cases}$$

P''1(7/3,3)

Ta xén P'1P''1 dựa trên từng cạnh.

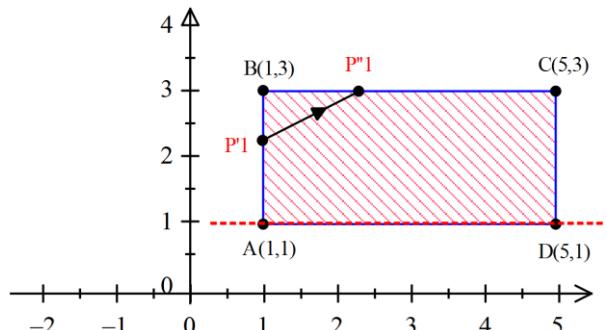
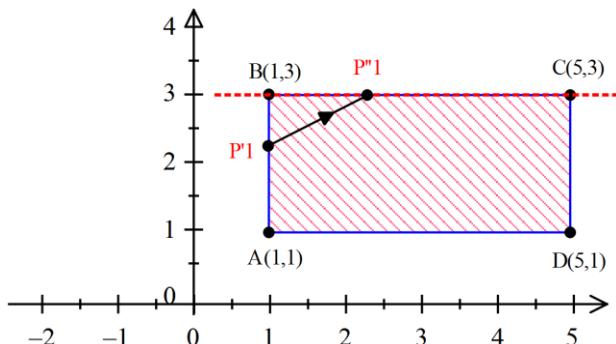
④ AD:

Điểm P'1: nằm bên trong AD

Điểm P''1: nằm bên trong AD

Chiều In-In

Vậy P'1P''1 được lưu



Clipping Point, Line, Polygon

Lập trình mô phỏng

```
#include <graphics.h>
#include <math.h>
#define ROUND(a) ((int)(a+0.5))

#define OUT2IN 1
#define INSIDE 2
#define IN2OUT 3
#define OUTSIDE 4
#define INVALID 5

//khai bao bien
int xc[100];
int yc[100];
int wx[100];
int wy[100];
int pointsize;
int windowsize;
int xc_tam[100];
int yc_tam[100];
int pointsize_tam;

//chuong trinh con
void draw_object()
{
    int i;
    //ve cac canh cua da giac
    setcolor(RED);
    for(i=0;i<pointsize;i++)
    {
        line(xc[i],yc[i],xc[(i+1)%pointsize],yc[(i+1)%pointsize]);
        delay(1000);
    }
    //ve cac canh cua so clipping
    setcolor(BLUE);
    for(i=0;i<windowsize;i++)
    {
        line(wx[i],wy[i],wx[(i+1)%windowsize],wy[(i+1)%windowsize]);
        delay(1000);
    }
}

//nhap thong so
void nhapdulieu()
{
    int i;

    printf("Nhập số cạnh của đa giác = ");
    scanf("%d",&pointsize);

    for (i=0;i<pointsize;i++)
    {
        printf("Toa độ x cho cạnh %d = ",i);
        scanf("%d",&xc[i]);
        printf("Toa độ y cho cạnh %d = ",i);
        scanf("%d",&yc[i]);
    }

    printf("Nhập số cạnh của đa giác = ");
    scanf("%d",&windowsize);
}
```

Clipping Point, Line, Polygon

```
for (i=0;i<windowsize;i++)
{
    printf("Toa do x cho canh window %d = ",i);
    scanf("%d",&wx[i]);
    printf("Toa do y cho canh window %d = ",i);
    scanf("%d",&wy[i]);
}
}

int clipLeft(int x,int x1,int x2)
{
    if(x1<x && x2>=x)
    {
        return OUT2IN;
    }
    else
    {
        if(x1>=x && x2>=x)
        {
            return INSIDE;
        }
        else
        {
            if(x1>=x && x2<x)
            {
                return IN2OUT;
            }
            else
            {
                if(x1<x && x2<x)
                {
                    return OUTSIDE;
                }
                else
                {
                    return INVALID;
                }
            }
        }
    }
}

int clipBottom(int y,int y1,int y2)
{
    if(y1<y && y2>=y)
    {
        return OUT2IN;
    }
    else
    {
        if(y1>=y && y2>=y)
        {
            return INSIDE;
        }
        else
        {
            if(y1>=y && y2<y)
            {
                return IN2OUT;
            }
            else
            {
                if(y1<y && y2<y)
```

Clipping Point, Line, Polygon

```
        {
            return OUTSIDE;
        }
    else
    {
        return INVALID;
    }
}
}

int clipRight(int x,int x1,int x2)
{
    if(x1>x && x2<=x)
    {
        return OUT2IN;
    }
    else
    {
        if(x1<=x && x2<=x)
        {
            return INSIDE;
        }
        else
        {
            if(x1<=x && x2>x)
            {
                return IN2OUT;
            }
            else
            {
                if(x1>x && x2>x)
                {
                    return OUTSIDE;
                }
                else
                {
                    return INVALID;
                }
            }
        }
    }
}

int clipTop(int y,int y1,int y2)
{
    if(y1>y && y2<=y)
    {
        return OUT2IN;
    }
    else
    {
        if(y1<=y && y2<=y)
        {
            return INSIDE;
        }
        else
        {
            if(y1<=y && y2>y)
            {
                return IN2OUT;
            }
        }
    }
}
```

Clipping Point, Line, Polygon

```
else
{
    if(y1>y && y2>y)
    {
        return OUTSIDE;
    }
    else
    {
        return INVALID;
    }
}
}

void leftclipper()
{
    int chieu;
    int i;
    int xwmin;
    int xa;
    int ya;
    int xb;
    int yb;
    int j;
    int xnew;
    int ynew;

    xwmin=wx[0];//dinh 1
    j=0;

    for(i=0;i<pointsize;i++)
    {
        xa=xc[i];
        ya=yc[i];
        xb=xc[i+1];
        yb=yc[i+1];

        chieu=clipLeft(xwmin,xa,xb);
        switch (chieu)
        {
            case OUT2IN:
            {
                xnew = xwmin;
                ynew = ya + (float) (yb-ya) * (float) (xwmin-xa) / (float) (xb-xa);
                xc_tam[j]=xnew;
                yc_tam[j]=ynew;
                j++;
                xc_tam[j]=xb;
                yc_tam[j]=yb;
                j++;
                break;
            }
            case INSIDE:
            {
                xnew = xb;
                ynew = yb;
                xc_tam[j]=xnew;
                yc_tam[j]=ynew;
                j++;
                break;
            }
            case IN2OUT:
        }
    }
}
```

Clipping Point, Line, Polygon

```
{  
    xnew = xwmin;  
    ynew = ya + (float) (yb-ya) * (float) (xwmin-xa) / (float) (xb-xa);  
    xc_tam[j]=xnew;  
    yc_tam[j]=ynew;  
    j++;  
    break;  
}  
case OUTSIDE:  
{  
    break;  
}  
}  
printf("\n chieu = %d\n", chieu);  
}  
  
//gan lai danh sach canh moi  
for(i=0;i<j;i++)  
{  
    xc[i]=xc_tam[i];  
    yc[i]=yc_tam[i];  
}  
//gan so dinh moi sau khi leftclipper  
pointsize=j;  
  
for(i=0;i<j;i++)  
{  
    printf("(%d,%d) ", xc[i], yc[i]);  
}  
printf("\n");  
}  
  
void rightclipper()  
{  
    int chieu;  
    int i;  
    int xwmax;  
    int xa;  
    int ya;  
    int xb;  
    int yb;  
    int j;  
    int xnew;  
    int ynew;  
  
    xwmax=wx[2];;/dinh 3  
    j=0;  
  
    for(i=0;i<pointsize;i++)  
    {  
        xa=xc[i];  
        ya=yc[i];  
        xb=xc[i+1];  
        yb=yc[i+1];  
  
        chieu=clipRight(xwmax,xa,xb);  
        switch (chieu)  
        {  
            case OUT2IN:  
            {  
                xnew = xwmax;  
                ynew = ya + (float) (yb-ya) * (float) (xwmax-xa) / (float) (xb-xa);  
            }  
        }  
    }  
}
```

Clipping Point, Line, Polygon

```
xc_tam[j]=xnew;
yc_tam[j]=ynew;
j++;
xc_tam[j]=xb;
yc_tam[j]=yb;
j++;
break;
}
case INSIDE:
{
    xnew = xb;
    ynew = yb;
    xc_tam[j]=xnew;
    yc_tam[j]=ynew;
    j++;
    break;
}
case IN2OUT:
{
    xnew = xwmax;
    ynew = ya + (float) (yb-ya) * (float) (xwmax-xa) / (float) (xb-xa);
    xc_tam[j]=xnew;
    yc_tam[j]=ynew;
    j++;
    break;
}
case OUTSIDE:
{
    break;
}
}
printf("\n chieu = %d\n",chieu);
}

//gan lai danh sach canh moi
for(i=0;i<j;i++)
{
    xc[i]=xc_tam[i];
    yc[i]=yc_tam[i];
}
//gan so dinh moi sau khi rightclipper
pointsize=j;

for(i=0;i<j;i++)
{
    printf("(%d,%d); ",xc[i],yc[i]);
}
printf("\n");
}

void bottomclipper()
{
    int chieu;
    int i;
    int ywmax;
    int xa;
    int ya;
    int xb;
    int yb;
    int j;
    int xnew;
    int ynew;
```

Clipping Point, Line, Polygon

```
ywmax=wy[2];//dinh 3
j=0;

for(i=0;i<pointszie;i++)
{
    xa=xc[i];
    ya=yc[i];
    xb=xc[i+1];
    yb=yc[i+1];

    chieu=clipTop(ywmax,ya,yb);
    switch (chieu)
    {
        case OUT2IN:
        {
            xnew = xa + (float) (ywmax-ya) * (xb-xa) / (yb-ya);
            ynew = ywmax;
            xc_tam[j]=xnew;
            yc_tam[j]=ynew;
            j++;
            xc_tam[j]=xb;
            yc_tam[j]=yb;
            j++;
            break;
        }
        case INSIDE:
        {
            xnew = xb;
            ynew = yb;
            xc_tam[j]=xnew;
            yc_tam[j]=ynew;
            j++;
            break;
        }
        case IN2OUT:
        {
            xnew = xa + (float) (ywmax-ya) * (xb-xa) / (yb-ya);
            ynew = ywmax;
            xc_tam[j]=xnew;
            yc_tam[j]=ynew;
            j++;
            break;
        }
        case OUTSIDE:
        {
            break;
        }
    }
    printf("\n chieu = %d\n",chieu);
}

//gan lai danh sach canh moi
for(i=0;i<j;i++)
{
    xc[i]=xc_tam[i];
    yc[i]=yc_tam[i];
}
//gan so dinh moi sau khi rightclipper
pointszie=j;

for(i=0;i<j;i++)
{
    printf("(%d,%d); ",xc[i],yc[i]);
```

Clipping Point, Line, Polygon

```
    }
    printf("\n");
}

void topclipper()
{
    int chieu;
    int i;
    int ywmin;
    int xa;
    int ya;
    int xb;
    int yb;
    int j;
    int xnew;
    int ynew;

    ywmin=wy[0];//dinh 1
    j=0;

    for(i=0;i<pointszie;i++)
    {
        xa=xc[i];
        ya=yc[i];
        xb=xc[i+1];
        yb=yc[i+1];

        chieu=clipBottom(ywmin, ya, yb);

        switch (chieu)
        {
            case OUT2IN:
            {
                xnew = xa + (float) (ywmin-ya) * (xb-xa) / (yb-ya);
                ynew = ywmin;
                xc_tam[j]=xnew;
                yc_tam[j]=ynew;
                j++;
                xc_tam[j]=xb;
                yc_tam[j]=yb;
                j++;
                break;
            }
            case INSIDE:
            {
                xnew = xb;
                ynew = yb;
                xc_tam[j]=xnew;
                yc_tam[j]=ynew;
                j++;
                break;
            }
            case IN2OUT:
            {
                xnew = xa + (float) (ywmin-ya) * (xb-xa) / (yb-ya);
                ynew = ywmin;
                xc_tam[j]=xnew;
                yc_tam[j]=ynew;
                j++;
                break;
            }
            case OUTSIDE:
            {

```

Clipping Point, Line, Polygon

```
        break;
    }
}
printf("\n chieu = %d\n",chieu);
}

//gan lai danh sach canh moi
for(i=0;i<j;i++)
{
    xc[i]=xc_tam[i];
    yc[i]=yc_tam[i];
}
//gan so dinh moi sau khi rightclipper
pointsize=j;

for(i=0;i<j;i++)
{
    printf("(%d,%d) ", xc[i],yc[i]);
}
printf("\n");

void draw_object_afterclip()
{
    int i;
    //ve cac canh cua da giac
    setcolor(YELLOW);
    setlinestyle(1,0,1);
    for(i=0;i<pointsize;i++)
    {
        line(xc[i],yc[i],xc[(i+1)%pointsize],yc[(i+1)%pointsize]);
        delay(1000);
    }
}

//chuong trinh chinh
int main()
{
    nhapdulieu();
    initwindow(800,800);
    draw_object();
    //gan them 1 dinh o cuoi cung chinh la dinh xuat phat
    xc[pointsize]=xc[0];
    yc[pointsize]=yc[0];

    leftclipper();
    //gan them 1 dinh o cuoi cung chinh la dinh xuat phat
    xc[pointsize]=xc[0];
    yc[pointsize]=yc[0];
    rightclipper();
    //gan them 1 dinh o cuoi cung chinh la dinh xuat phat
    xc[pointsize]=xc[0];
    yc[pointsize]=yc[0];
    bottomclipper();
    //gan them 1 dinh o cuoi cung chinh la dinh xuat phat
    xc[pointsize]=xc[0];
    yc[pointsize]=yc[0];
    topclipper();
    draw_object_afterclip();
    getch();
    return 0;
}
```

Phép chiếu trực giao (Orthographic projection)

Phép chiếu trực giao (Orthographic projection) là phép chiếu song song và tia chiếu vuông góc với mặt phẳng chiếu thường dùng mặt phẳng $z=0$

Ứng với mỗi mặt phẳng chiếu ta có 1 ma trận chiếu tương ứng

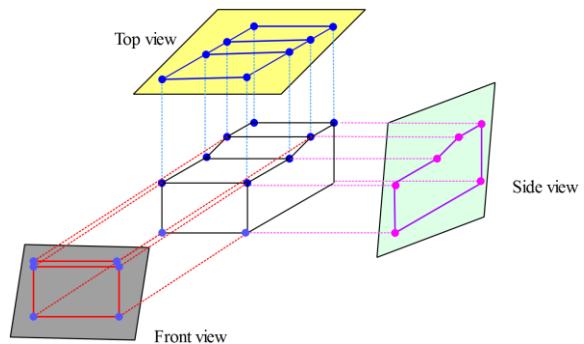
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$[T_x] = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad [T_y] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$[T_z] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Các phép chiếu trực giao hầu như được dùng để tạo ra quang cảnh nhìn từ phía trước, bên sườn, và trên đỉnh của đối tượng.

- Quang cảnh phía trước, bên sườn, và phía sau của đối tượng được gọi là “mặt chiếu” (elevation).
- Quang cảnh phía trên được gọi là “mặt phẳng” (plane).



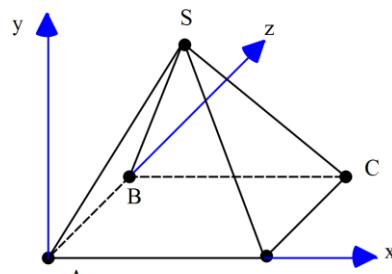
Các bản vẽ trong kỹ thuật thường dùng các phép chiếu trực giao này, vì các chiều dài và góc miêu tả chính xác và có thể đo được từ bản vẽ.

Bài tập 1.

Xác định các hình chiếu của đối tượng 3D lên các mặt phẳng chiếu (top, left, front)

Ta có:

- Front= $P_{chop} * T_z$
- Left= $P_{chop} * T_x$
- Top= $P_{chop} * T_y$

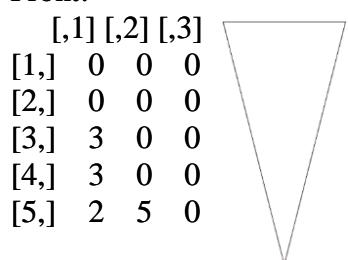


$$\begin{bmatrix} A & x & y & z \\ B & 0 & 0 & 0 \\ C & 3 & 0 & 3 \\ D & 3 & 0 & 0 \\ S & 2 & 5 & 2 \end{bmatrix}$$

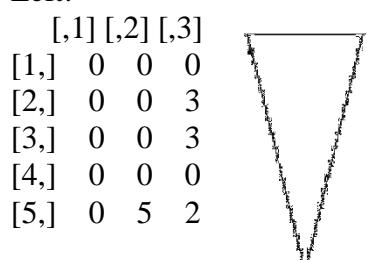
Ta được:

$$T_x = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad T_y = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad T_z = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

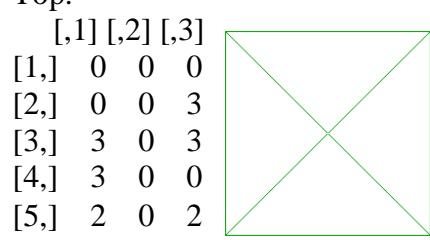
Front:



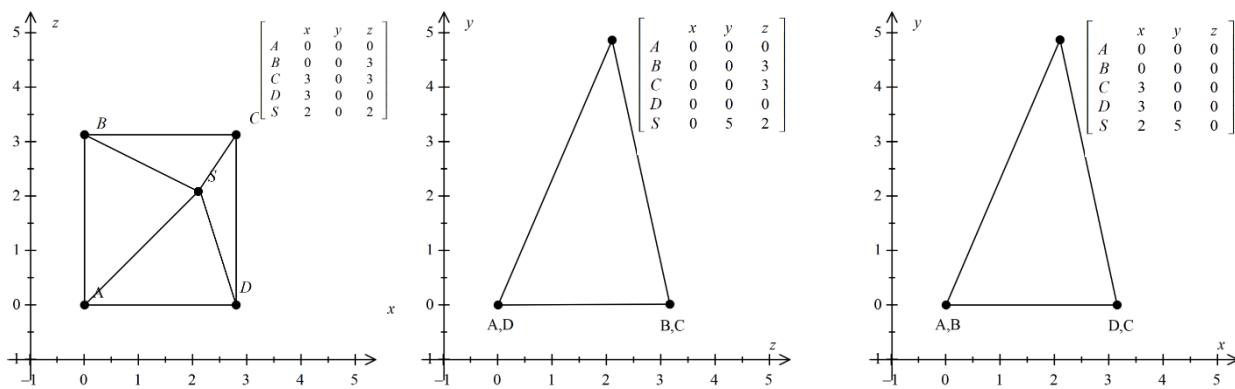
Left:



Top:

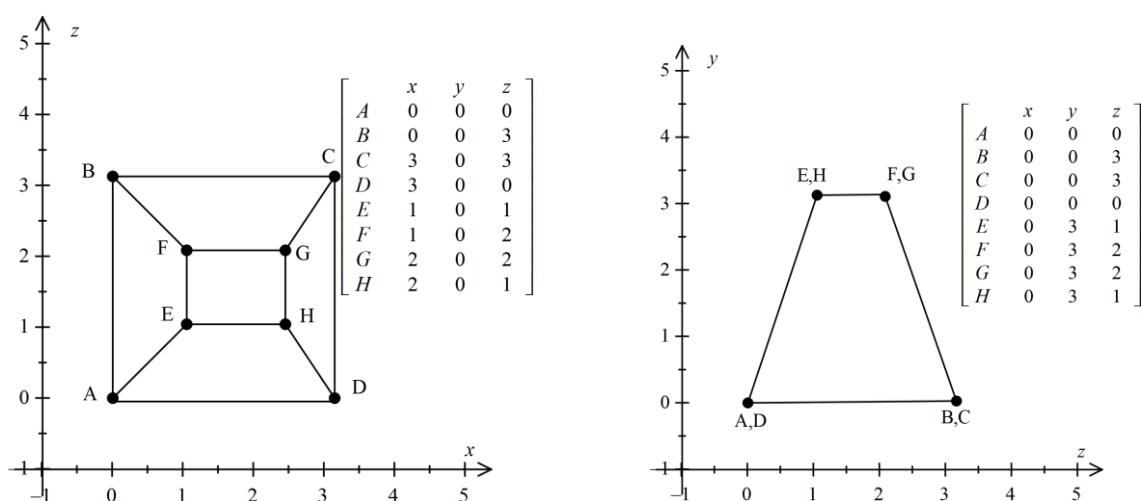
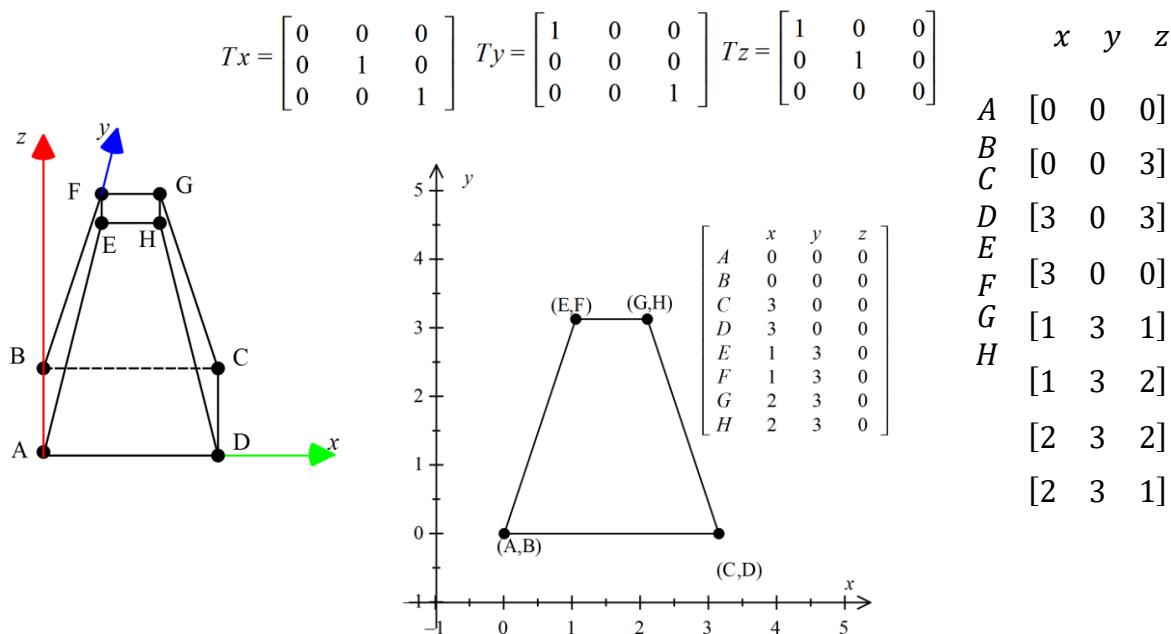


Phép chiếu trực giao (Orthographic projection)



Bài tập 2.

Xác định các hình chiếu của đối tượng 3D lên các mặt phẳng chiếu (top, left, front)



Phép chiếu trực giao (Orthographic projection)

Lập trình mô phỏng (chóp nhọn)

```
//khai bao thu vien
#include<graphics.h>
//khai bao bien
//int X[10],Y[10],Z[10]; //toi da 10 dinh cua doi tuong 3D
int X[5]={100,100,300,300,200};
int Y[5]={100,100,100,100,500};
int Z[5]={100,300,300,100,200};

//int D[10][10]; //ma tran ke noi cac dinh
int D[5][5]=
{
    {1,1,0,1,1},
    {1,1,1,0,1},
    {0,1,1,1,1},
    {1,0,1,1,1},
    {1,1,1,1,1},
};

int sodinh =5;

//chuong trinh con
void nhapdiem()
{
    int i,j;
    printf("Nhập số lượng đỉnh: ");
    scanf("%d",&sodinh);

    for(i=0;i<sodinh;i++)
    {
        printf("X[%d]= ",i);
        scanf("%d",&X[i]);
        printf("Y[%d]= ",i);
        scanf("%d",&Y[i]);
        printf("Z[%d]= ",i);
        scanf("%d",&Z[i]);
    }
}

void intoadoadiem()
{
    int i,j;
    for(i=0;i<sodinh;i++)
    {
        printf("%d\t",X[i]);
    }

    printf("\n");
    for(i=0;i<sodinh;i++)
    {
        printf("%d\t",Y[i]);
    }

    printf("\n");
    for(i=0;i<sodinh;i++)
    {
        printf("%d\t",Z[i]);
    }
}
```

Phép chiếu trực giao (Orthographic projection)

```
void matranke()
{
    int i,j;

    for(i=0;i<sodinh;i++)
    {
        D[i][i]=1;
    }

    for(i=0;i<sodinh-1;i++)
    {
        for(j=i+1;j<sodinh;j++)
        {
            printf("D[%d,%d]= ",i,j);
            scanf("%d",&D[i][j]);
        }
    }

    for(i=0;i<sodinh;i++)
    {
        for(j=0;j<sodinh;j++)
        {
            if(i>j)
            {
                D[i][j]=D[j][i];
            }
        }
    }
}

void inmatranke()
{
    int i,j;

    //in ra ma tran ke
    printf("\nDanh sach ma tran ke:\n");
    for(i=0;i<sodinh;i++)
    {
        for(j=0;j<sodinh;j++)
        {
            printf("%d\t",D[i][j]);
        }
        printf("\n");
    }
}

void oxy()
{
    int i,j;

    //cai nay do minh dat ban dau, neu sau nay tu nhap tu ban phim thi
    thay bang so cu the

    for(i=0;i<sodinh-1;i++)
    {
        for(j=i+1;j<sodinh;j++)
        {
            if(D[i][j]==1)
            {
                line(X[i],Y[i],X[j],Y[j]);
            }
        }
    }
}
```

Phép chiếu trực giao (Orthographic projection)

```
    }  
}  
  
void oxz()  
{  
    int i,j;  
  
    //cai nay do minh dat ban dau, neu sau nay tu nhap tu ban phim thi  
thay bang so cu the  
  
    for(i=0;i<sodinh-1;i++)  
    {  
        for(j=i+1;j<sodinh;j++)  
        {  
            if(D[i][j]==1)  
            {  
                line(X[i],Z[i],X[j],Z[j]);  
            }  
        }  
    }  
  
void oyz()  
{  
    int i,j;  
  
    //cai nay do minh dat ban dau, neu sau nay tu nhap tu ban phim thi  
thay bang so cu the  
  
    for(i=0;i<sodinh-1;i++)  
    {  
        for(j=i+1;j<sodinh;j++)  
        {  
            if(D[i][j]==1)  
            {  
                line(Z[i],Y[i],Z[j],Y[j]);  
            }  
        }  
    }  
  
void ve3D()  
{  
    outtextxy(0,0,"Hinh chieu tren mat phang XY");  
    setcolor(RED);  
    oxy();  
    getch();  
  
    cleardevice();  
    outtextxy(0,0,"Hinh chieu tren mat phang XZ");  
    setcolor(GREEN);  
    oxz();  
    getch();  
  
    cleardevice();  
    outtextxy(0,0,"Hinh chieu tren mat phang YZ");  
    setcolor(YELLOW);  
    oyz();  
}
```

Phép chiếu trực giao (Orthographic projection)

```
//chuong trinh chinh
int main()
{
    //nhap toa do cac diem
    //nhapdiem();

    intoadoadiem();

    //nhap ma tran ke
    //matranke();

    //in ra ma tran ke
    inmatranke();

    //khai tao man hinh do hoa
    initwindow(600,600);

    ve3D();

    getch();
}
```

Lập trình mô phỏng (chóp cùt)

```
//khai bao thu vien
#include<graphics.h>

//khai bao bien
//int X[10],Y[10],Z[10]; //toi da 10 dinh cua doi tuong 3D
int X[8]={100,100,400,400,200,200,300,300};
int Y[8]={100,100,100,100,400,400,400,400};
int Z[8]={100,400,400,100,200,300,300,200};

//int D[10][10]; //ma tran ke noi cac dinh
int D[8][8]=
{
    //A,B,C,D,E,F,G,H
    {1,1,0,1,1,0,0,0},//A
    {1,1,1,0,0,1,0,0},//B
    {0,1,1,1,0,0,1,0},//C
    {1,0,1,1,0,0,0,1},//D
    {1,0,0,0,1,1,0,1},//E
    {0,1,0,0,1,1,1,0},//F
    {0,0,1,0,0,1,1,1},//G
    {0,0,0,1,1,0,1,1},//H
};

int sodinh =8;
//chuong trinh con
```

Phép chiếu trực giao (Orthographic projection)

```
void nhapdiem()
{
    int i,j;
    printf("Nhập số lượng đỉnh: ");
    scanf("%d",&sodinh);

    for(i=0;i<sodinh;i++)
    {
        printf("X[%d]= ",i);
        scanf("%d",&X[i]);
        printf("Y[%d]= ",i);
        scanf("%d",&Y[i]);
        printf("Z[%d]= ",i);
        scanf("%d",&Z[i]);
    }
}

void intoadoadiem()
{
    int i,j;
    for(i=0;i<sodinh;i++)
    {
        printf("%d\t",X[i]);
    }

    printf("\n");
    for(i=0;i<sodinh;i++)
    {
        printf("%d\t",Y[i]);
    }

    printf("\n");
    for(i=0;i<sodinh;i++)
    {
        printf("%d\t",Z[i]);
    }
}

void matranke()
{
    int i,j;

    for(i=0;i<sodinh;i++)
    {
        D[i][i]=1;
    }

    for(i=0;i<sodinh-1;i++)
    {
        for(j=i+1;j<sodinh;j++)
        {
            printf("D[%d,%d]= ",i,j);
            scanf("%d",&D[i][j]);
        }
    }

    for(i=0;i<sodinh;i++)
    {
        for(j=0;j<sodinh;j++)
        {
            if(i>j)
            {
                D[i][j]=D[j][i];
            }
        }
    }
}
```

Phép chiếu trực giao (Orthographic projection)

```
        }
    }

void inmatranke()
{
    int i,j;

    //in ra ma tran ke
    printf("\nDanh sach ma tran ke:\n");
    for(i=0;i<sodinh;i++)
    {
        for(j=0;j<sodinh;j++)
        {
            printf("%d\t",D[i][j]);
        }
        printf("\n");
    }
}

void oxy()
{
    int i,j;

    //cai nay do minh dat ban dau, neu sau nay tu nhap tu ban phim thi
    thay bang so cu the

    for(i=0;i<sodinh-1;i++)
    {
        for(j=i+1;j<sodinh;j++)
        {
            if(D[i][j]==1)
            {
                line(X[i],Y[i],X[j],Y[j]);
            }
        }
    }
}

void oxz()
{
    int i,j;

    //cai nay do minh dat ban dau, neu sau nay tu nhap tu ban phim thi
    thay bang so cu the

    for(i=0;i<sodinh-1;i++)
    {
        for(j=i+1;j<sodinh;j++)
        {
            if(D[i][j]==1)
            {
                line(X[i],Z[i],X[j],Z[j]);
            }
        }
    }
}

void oyza()
{
    int i,j;
```

Phép chiếu trực giao (Orthographic projection)

```
//cai nay do minh dat ban dau, neu sau nay tu nhap tu ban phim thi  
thay bang so cu the  
  
for(i=0;i<sodinh-1;i++)  
{  
    for(j=i+1;j<sodinh;j++)  
    {  
        if(D[i][j]==1)  
        {  
            line(Z[i],Y[i],Z[j],Y[j]);  
        }  
    }  
}  
  
void ve3D()  
{  
    outtextxy(0,0,"Hinh chieu tren mat phang XY");  
    setcolor(RED);  
    oxy();  
    getch();  
  
    cleardevice();  
    outtextxy(0,0,"Hinh chieu tren mat phang XZ");  
    setcolor(GREEN);  
    oxz();  
    getch();  
  
    cleardevice();  
    outtextxy(0,0,"Hinh chieu tren mat phang YZ");  
    setcolor(YELLOW);  
    oyz();  
}  
//chuong trinh chinh  
int main()  
{  
  
    //nhap toa do cac diem  
    //nhapdiem();  
  
    intoadoadiem();  
  
    //nhap ma tran ke  
    //matranke();  
  
    //in ra ma tran ke  
    inmatranke();  
  
    //khoi tao man hinh do hoa  
    initwindow(600,600);  
  
    ve3D();  
  
    getch();  
}
```

Phép chiếu xiên (Oblique projection)

Phép chiếu xiên (Oblique Projection)

Một phép chiếu xiên đạt được bằng việc chiếu các điểm theo các đường thẳng song song, các đường thẳng này không vuông góc với mặt phẳng chiếu.

Hình chiếu xiên của điểm (x, y, z) theo một đường thẳng chiếu đến vị trí $(x', y', 0)$.

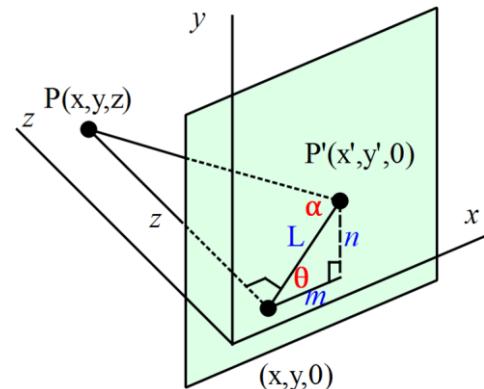
☐ Ta có: $x' = x + m$ và $\cos \theta = \frac{m}{L}$

☐ $y' = y + n$ và $\sin \theta = \frac{n}{L}$

☐ Suy ra:

$$\begin{cases} x' = x + L \cos \theta \\ y' = y + L \sin \theta \end{cases}$$

$$\tan \alpha = \frac{z}{L} = \frac{1}{L_1}$$



L_1 là chiều dài của các đường chiếu từ (x, y) đến (x', y') khi $z = 1$

Suy ra: $L = z \cdot L_1$

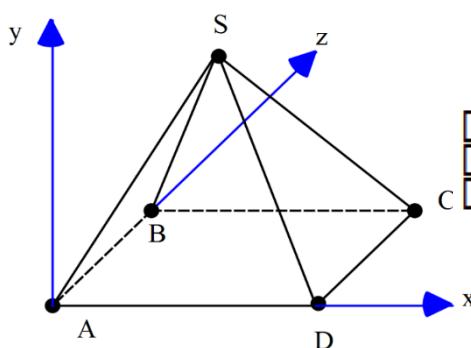
$$\begin{cases} x' = x + z L \cos \theta \\ y' = y + z L \sin \theta \end{cases}$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & L_1 \cdot \cos \theta & 0 \\ 0 & 1 & L_1 \cdot \sin \theta & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Bài tập 1.

Xác định hình chiếu của đối tượng 3D lên các mặt phẳng chiếu như sau:

- ☐ $\theta = 60^\circ$
- ☐ $L_1 = 0.5$



Using R studio
R command

phi=30
L1=0.5
phi_rad = 3.14*phi/180;

Pchop=matrix(c(100,100,100,100,100,300,300,100,300,300,100,100,200,500,200),nrow=3,byrow=TRUE)

T = matrix(c(1,0, L1*cos(phi_rad),0,1, L1*sin(phi_rad),0,0,0),ncol=3,byrow=TRUE)

Pnew=round(T %*% Pchop,0)

Phép chiếu xiên (Oblique projection)

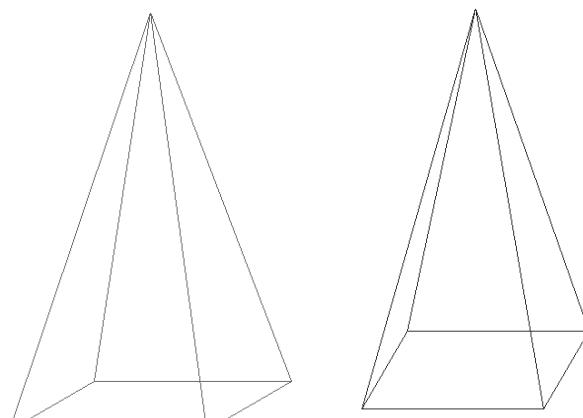
Kết quả ta có (Pnew):

[,1] [,2] [,3] [,4] [,5]

[1,] 143 143 187 317 187

[2,] 325 325 150 425 350

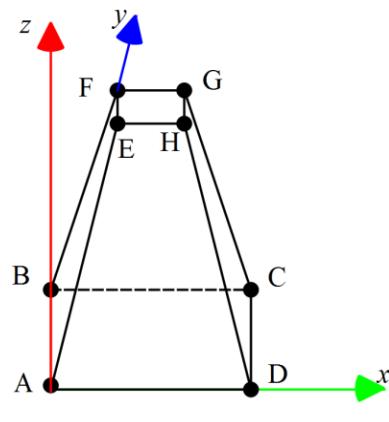
[3,] 0 0 0 0 0



$$\theta = 30^\circ \quad \theta = 60^\circ$$

Bài tập 2:

Xác định các hình chiếu của đối tượng 3D lên các mặt phẳng chiếu (top, left, front)



R command

phi=30

L1=0.5

phi_rad = 3.14*phi/180;

Pchop=matrix(c(0,0,0,0,0,3,3,0,3,3,0,0,2,5,2),nrow=5,byrow=TRUE)

T = matrix(c(1,0, L1*cos(phi_rad),0,1, L1*sin(phi_rad),0,0,0),ncol=3,byrow=TRUE)

Pnew=round(Pchop%*%T%)

[,1] [,2] [,3]

[1,] 0 0 0

[2,] 0 0 0

[3,] 3 0 1

[4,] 3 0 1

[5,] 2 5 2

x y z

A [0 0 0]

B [0 0 3]

C [3 0 3]

D [3 0 0]

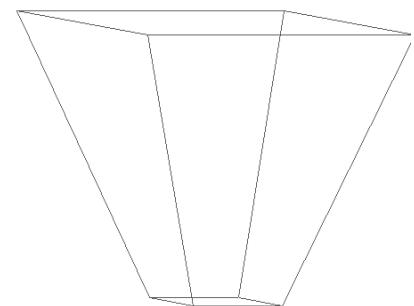
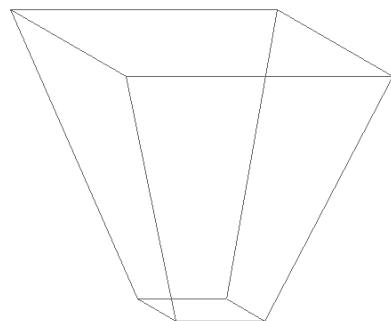
E [1 3 1]

F [1 3 2]

G [2 3 2]

H [2 3 1]

$\theta = 30^\circ$



$$\theta = 10^\circ$$

Phép chiếu xiên (Oblique projection)

Lập trình mô phỏng (chóp nhọn)

```
#include<stdio.h>
#include<math.h>
#include<graphics.h>
#define dinh 5

//Phep chieu cabinet. cho hinh hop chu nhat co dinh = 8

//khai bao bien
int phi;
double L1 = 0.5;
double phi_rad;

int a[3][dinh]; //ma tran hinh hop chu nhat
int P[3][dinh]; //ma tran ket qua
float T[3][3]; //ma tran bien doi

int X[dinh];
int Y[dinh];

int m=3; //so hang cua ma tran toa do dinh
int p=5; //so cot cua ma tran toa do dinh
int n=3; //so hang = so cot cua ma tran bien doi T

//int D[10][10]; //ma tran ke noi cac dinh
int D[5][5]=
{
    {1,1,0,1,1},
    {1,1,1,0,1},
    {0,1,1,1,1},
    {1,0,1,1,1},
    {1,1,1,1,1},
};

//chuong trinh con
void matranke()
{
    int i,j;

    for(i=0;i<dinh;i++)
    {
        D[i][i]=1;
    }

    for(i=0;i<dinh-1;i++)
    {
        for(j=i+1;j<dinh;j++)
        {
            printf("D[%d,%d]= ",i,j);
            scanf("%d",&D[i][j]);
        }
    }

    for(i=0;i<dinh;i++)
    {
        for(j=0;j<dinh;j++)
        {
            if(i>j)
            {
                D[i][j]=D[j][i];
            }
        }
    }
}
```

Phép chiếu xiên (Oblique projection)

```
    }
}

void inmatranke()
{
    int i,j;

    //in ra ma tran ke
    printf("\nDanh sach ma tran ke:\n");
    for(i=0;i<dinh;i++)
    {
        for(j=0;j<dinh;j++)
        {
            printf("%d\t",D[i][j]);
        }
        printf("\n");
    }
}

void nhaptoado()
{
    //X
    Y
    Z
    a[0][0] = 100; a[1][0] = 100; a[2][0] = 100;//A
    a[0][1] = 100; a[1][1] = 100; a[2][1] = 300;//B
    a[0][2] = 300; a[1][2] = 100; a[2][2] = 300;//C
    a[0][3] = 300; a[1][3] = 100; a[2][3] = 100;//D
    a[0][4] = 200; a[1][4] = 500; a[2][4] = 200;//S

    printf("\nNhập hệ số góc phi = \n");
    scanf("%d",&phi);
    while(phi<0 || phi >180)
    {
        printf("\nNhập lại hệ số góc phi từ 0 - 180 do\n");
        scanf("%d",&phi);
    }
    phi_rad = (double) (3.14*phi)/180;
}

void matranbiendoi()
{
    T[0][0] = 1; T[0][1] = 0; T[0][2] = L1*cos(phi_rad); T[0][3] = 0;
    T[1][0] = 0; T[1][1] = 1; T[1][2] = L1*sin(phi_rad);
    T[1][3] = 0;
    T[2][0] = 0; T[2][1] = 0; T[2][2] = 0; T[2][3] =
0;
}

void cabinet()
{
    int i,j,k;

    //khởi tạo ma trận kết quả
    for(i=0; i<m; i++) //so hàng
    {
        for(k=0; k<p; k++) //so cột
        {
            P[i][k] = 0;
        }
    }
}
```

Phép chiếu xiên (Oblique projection)

```
//nhận 2 ma tran
    for(i=0; i<m; i++) //so hang a
    {
        for(j=0; j<p; j++)//so cot a
        {
            for(k=0; k<n; k++)//so cot T
            {
                P[i][j] += (float)T[i][k]*a[k][j];
            }
        }
    }

void inketqua()
{
    int i,j;

    //in toa do cac dinh
    printf("Toa do cac dinh cua hinh hop:\n");
    for(i=0;i<m;i++)
    {
        for(j=0;j<p;j++)
        {
            printf("%d\t",a[i][j]);
        }
        printf("\n");
    }

    printf("\nGia tri cua ma tran cabinet\n");
    for(i=0;i<m;i++)
    {
        for(j=0;j<n;j++)
        {
            printf("%.2f\t",T[i][j]);
        }
        printf("\n");
    }

    printf("\nToa do cac dinh cua hinh hop sau khi
chieu:\n");
    for(i=0;i<m;i++)
    {
        for(j=0;j<p;j++)
        {
            printf("%d\t",P[i][j]);
        }
        printf("\n");
    }
}

//in hinh chieu ra man hinh
void inhinhchieu()
{
    int i,j;

    for(i=0; i<dinh;i++)
    {
        X[i]=P[0][i];
        Y[i]=P[1][i];
    }

    //in ra mang X va Y
```

Phép chiếu xiên (Oblique projection)

```
for(i=0;i<dinh;i++)
{
    printf(" (%d,%d) \t",X[i],Y[i]);
}

void oxy()
{
    int i,j;

    //cai nay do minh dat ban dau, neu sau nay tu nhap tu ban phim thi
    thay bang so cu the
    setcolor(RED);
    for(i=0;i<dinh-1;i++)
    {
        for(j=i+1;j<dinh;j++)
        {
            if(D[i][j]==1)
            {
                line(X[i],Y[i],X[j],Y[j]);
                delay(100);
            }
        }
    }
}

//chuong trinh chinh
int main()
{
    //nhap toa do cac dinh cua hinh hop
    nhaptoado();

    //nhap ma tran bien doi
    matranbiendoi();

    //nhap ma tran ke
    //nhapmatranke()

    //in ra ma tran ke
    inmatranke();

    //tinh toan ma tran ket qua
    cabinet();

    //in ket qua cac phep toan
    inketqua();

    //in ket qua cua phep chieu
    inhhinhchieu();

    //khoi tao man hinh do hoa
    initwindow(600,600);
    oxy();

    getch();
}
```

Phép chiếu xiên (Oblique projection)

Lập trình mô phỏng (chóp cüt)

```
#include<stdio.h>
#include<math.h>
#include<graphics.h>
#define dinh 8

//Phep chieu cabinet. cho hinh hop chu nhat co dinh = 8
//khai bao bien
int phi;
double L1 = 0.5;
double phi_rad;

int a[4][dinh]; //ma tran hinh hop chu nhat
int P[4][dinh]; //ma tran ket qua
float T[4][4]; //ma tran bien doi

int X[dinh];
int Y[dinh];

int m=3; //so hang cua ma tran toa do dinh
int p=8; //so cot cua ma tran toa do dinh
int n=3; //so hang = so cot cua ma tran bien doi T

//int D[10][10]; //ma tran ke noi cac dinh
int D[8][8]=
{
    //A,B,C,D,E,F,G,H

    {1,1,0,1,1,0,0,0},//A
    {1,1,1,0,0,1,0,0},//B
    {0,1,1,1,0,0,1,0},//C
    {1,0,1,1,0,0,0,1},//D
    {1,0,0,0,1,1,0,1},//E
    {0,1,0,0,1,1,1,0},//F
    {0,0,1,0,0,1,1,1},//G
    {0,0,0,1,1,0,1,1},//H
};

//chuong trinh con
void matranke()
{
    int i,j;

    for(i=0;i<dinh;i++)
    {
        D[i][i]=1;
    }

    for(i=0;i<dinh-1;i++)
    {
        for(j=i+1;j<dinh;j++)
        {
            printf("D[%d,%d]= ",i,j);
            scanf("%d",&D[i][j]);
        }
    }
}
```

Phép chiếu xiên (Oblique projection)

```
}

for(i=0;i<dinh;i++)
{
    for(j=0;j<dinh;j++)
    {
        if(i>j)
        {
            D[i][j]=D[j][i];
        }
    }
}

void inmatranke()
{
    int i,j;

    //in ra ma tran ke
    printf("\nDanh sach ma tran ke:\n");
    for(i=0;i<dinh;i++)
    {
        for(j=0;j<dinh;j++)
        {
            printf("%d\t",D[i][j]);
        }
        printf("\n");
    }
}

void nhaptoado()
{
    //X
    Y
    Z

    a[0][0] = 100; a[1][0] = 100; a[2][0] = 100;//A
    a[0][1] = 100; a[1][1] = 100; a[2][1] = 400;//B
    a[0][2] = 400; a[1][2] = 100; a[2][2] = 400;//C
    a[0][3] = 400; a[1][3] = 100; a[2][3] = 100;//D
    a[0][4] = 200; a[1][4] = 400; a[2][4] = 200;//E
    a[0][5] = 200; a[1][5] = 400; a[2][5] = 300;//F
    a[0][6] = 300; a[1][6] = 400; a[2][6] = 300;//G
    a[0][7] = 300; a[1][7] = 400; a[2][7] = 200;//H

    printf("\nNhập hệ số góc phi = \n");
    scanf("%d",&phi);
    while(phi<0 || phi >180)
    {
        printf("\nNhập lại hệ số góc phi từ 0 - 180 do
\n");
        scanf("%d",&phi);
    }
    phi_rad = (double) (3.14*phi)/180;
}

void matranbiendoi()
{
    T[0][0] = 1; T[0][1] = 0; T[0][2] = L1*cos(phi_rad); T[0][3] = 0;
    T[1][0] = 0; T[1][1] = 1; T[1][2] = L1*sin(phi_rad);
    T[1][3] = 0;
    T[2][0] = 0; T[2][1] = 0; T[2][2] = 0; T[2][3] =
0;
```

Phép chiếu xiên (Oblique projection)

```
//T[3][0] = 0; T[3][1] = 0; T[3][2] = 0; T[3][3] =
1;
}

void cabinet()
{
    int i,j,k;

    //khoi tao ma tran ket qua
    for(i=0; i<m; i++) //so hang
    {
        for(k=0; k<p; k++) //so cot
        {
            P[i][k] = 0;
        }
    }

    //nhan 2 ma tran
    for(i=0; i<m; i++) //so hang
    {
        for(k=0; k<p; k++)//so cot
        {
            for(j=0; j<n; j++)
            {
                P[i][k] += (float)T[i][j] *
a[j][k];
            }
        }
    }
}

void inketqua()
{
    int i,j;

    //in toa do cac dinh
    printf("Toa do cac dinh cua hinh hop:\n");
    for(i=0;i<m;i++)
    {
        for(j=0;j<p;j++)
        {
            printf("%d\t",a[i][j]);
        }
        printf("\n");
    }

    printf("\nGia tri cua ma tran cabinet\n");
    for(i=0;i<m;i++)
    {
        for(j=0;j<n;j++)
        {
            printf("%.2f\t",T[i][j]);
        }
        printf("\n");
    }

    printf("\nToa do cac dinh cua hinh hop sau khi
chieu:\n");
    for(i=0;i<m;i++)
    {
        for(j=0;j<p;j++)
        {
            printf("%d\t",P[i][j]);
        }
    }
}
```

Phép chiếu xiên (Oblique projection)

```
        }
        printf("\n");
    }

//in hình chiếu ra màn hình
void inhinhchieu()
{
    int i,j;
    for(i=0; i<dinh;i++)
    {
        X[i]=P[0][i];
        Y[i]=P[1][i];
    }

    //in ra mảng X và Y
    for(i=0;i<dinh;i++)
    {
        printf("(%d,%d)\t",X[i],Y[i]);
    }
}

void oxy()
{
    int i,j;
    //cai nay do minh dat ban dau, neu sau nay tu nhap tu ban phim thi
    thay bang so cu the
    setcolor(RED);
    for(i=0;i<dinh-1;i++)
    {
        for(j=i+1;j<dinh;j++)
        {
            if(D[i][j]==1)
            {
                line(X[i],Y[i],X[j],Y[j]);
                delay(500);
            }
        }
    }
}

//chuong trinh chinh
int main()
{
    //nhap toa do cac dinh cua hinh hop
    nhaptado();

    //nhap ma tran bien doi
    matranbiendoi();

    //nhap ma tran ke
    //nhapmatranke()
    //in ra ma tran ke
    inmatranke();

    //tinh toan ma tran ket qua
    cabinet();

    //in ket qua cac phep toan
    inketqua();
    //in ket qua cua phep chieu
    inhinhchieu();
    //khoi tao man hinh do hoa
```

Phép chiếu xiên (Oblique projection)

```
initwindow(600,600);
oxy();

getch();
}

Lập trình mô phỏng (Hình hộp)

#include<stdio.h>
#include<math.h>
#include<graphics.h>
#define dinh 8

//Phep chieu cabinet. cho hinh hop chu nhat co dinh = 8

//khai bao bien
int phi;
double L1 = 0.5;
double phi_rad;

int a[4][dinh]; //ma tran hinh hop chu nhat
int P[4][dinh]; //ma tran ket qua
float T[4][4]; //ma tran bien doi

int X[dinh];
int Y[dinh];

int m=3; //so hang cua ma tran toa do dinh
int p=8; //so cot cua ma tran toa do dinh
int n=3; //so hang = so cot cua ma tran bien doi T

//int D[10][10]; //ma tran ke noi cac dinh
int D[8][8]=
{
    {1,1,0,1,1,0,0,0},//A
    {1,1,1,0,0,1,0,0},//B
    {0,1,1,1,0,0,1,0},//C
    {1,0,1,1,0,0,0,1},//D
    {1,0,0,0,1,1,0,1},//E
    {0,1,0,0,1,1,1,0},//F
    {0,0,1,0,0,1,1,1},//G
    {0,0,0,1,1,0,1,1}//H
};

//chuong trinh con
void matranke()
{
    int i,j;

    for(i=0;i<dinh;i++)
    {
        D[i][i]=1;
    }

    for(i=0;i<dinh-1;i++)
    {
```

Phép chiếu xiên (Oblique projection)

```

for(j=i+1;j<dinh;j++)
{
    printf("D[%d,%d]= ",i,j);
    scanf("%d",&D[i][j]);
}
}

for(i=0;i<dinh;i++)
{
    for(j=0;j<dinh;j++)
    {
        if(i>j)
        {
            D[i][j]=D[j][i];
        }
    }
}
}

void inmatranke()
{
    int i,j;

    //in ra ma tran ke
    printf("\nDanh sach ma tran ke:\n");
    for(i=0;i<dinh;i++)
    {
        for(j=0;j<dinh;j++)
        {
            printf("%d\t",D[i][j]);
        }
        printf("\n");
    }
}

void nhaptoado()
{
    //X
    Y
    Z
    a[0][0] = 100; a[1][0] = 100; a[2][0] = 100;//A
    a[0][1] = 100; a[1][1] = 100; a[2][1] = 200;//B
    a[0][2] = 200; a[1][2] = 100; a[2][2] = 200;//C
    a[0][3] = 200; a[1][3] = 100; a[2][3] = 100;//D
    a[0][4] = 100; a[1][4] = 200; a[2][4] = 100;//E
    a[0][5] = 100; a[1][5] = 200; a[2][5] = 200;//F
    a[0][6] = 200; a[1][6] = 200; a[2][6] = 200;//G
    a[0][7] = 200; a[1][7] = 200; a[2][7] = 100;//H

    printf("\nNhap he so goc phi = \n");
    scanf("%d",&phi);
    while(phi<0 || phi >180)
    {
        printf("\nNhap lai he so goc phi tu 0 - 180 do
\n");
        scanf("%d",&phi);
    }
    phi_rad = (double) (3.14*phi)/180;
}

void matranbiendoi()
{
    T[0][0] = 1; T[0][1] = 0; T[0][2] = L1*cos(phi_rad); T[0][3] = 0;
    T[1][0] = 0; T[1][1] = 1; T[1][2] = L1*sin(phi_rad);
    T[1][3] = 0;
}

```

Phép chiếu xiên (Oblique projection)

```
T[2][0] = 0; T[2][1] = 0;      T[2][2] = 0; T[2][3] =
0;
//T[3][0] = 0; T[3][1] = 0;      T[3][2] = 0; T[3][3] =
1;
}
void cabinet()
{
    int i,j,k;

    //khoi tao ma tran ket qua
    for(i=0; i<m; i++) //so hang
    {
        for(k=0; k<p; k++) //so cot
        {
            P[i][k] = 0;
        }
    }

    //nhan 2 ma tran
    for(i=0; i<m; i++) //so hang
    {
        for(k=0; k<p; k++) //so cot
        {
            for(j=0; j<n; j++)
            {
                P[i][k] += (float)T[i][j] *
a[j][k];
            }
        }
    }
}

void inketqua()
{
    int i,j;

    //in toa do cac dinh
    printf("Toa do cac dinh cua hinh hop:\n");
    for(i=0;i<m;i++)
    {
        for(j=0;j<p;j++)
        {
            printf("%d\t",a[i][j]);
        }
        printf("\n");
    }

    printf("\nGia tri cua ma tran cabinet\n");
    for(i=0;i<m;i++)
    {
        for(j=0;j<n;j++)
        {
            printf("%.2f\t",T[i][j]);
        }
        printf("\n");
    }

    printf("\nToa do cac dinh cua hinh hop sau khi
chieu:\n");
    for(i=0;i<m;i++)
    {
        for(j=0;j<p;j++)
        {
            printf("%d\t",P[i][j]);
        }
    }
}
```

Phép chiếu xiên (Oblique projection)

```
        }
        printf("\n");
    }
}

//in hinh chieu ra man hinh
void inhhchieu()
{
    int i,j;

    for(i=0; i<dinh;i++)
    {
        X[i]=P[0][i];
        Y[i]=P[1][i];
    }

    //in ra mang X va Y
    for(i=0;i<dinh;i++)
    {
        printf("(%d,%d)\t",X[i],Y[i]);
    }
}

void oxy()
{
    int i,j;

    //cai nay do minh dat ban dau, neu sau nay tu nhap tu ban phim thi
    thay bang so cu the
    setcolor(RED);
    for(i=0;i<dinh-1;i++)
    {
        for(j=i+1;j<dinh;j++)
        {
            if(D[i][j]==1)
            {
                line(X[i],Y[i],X[j],Y[j]);
                delay(100);
            }
        }
    }
}

//chuong trinh chinh
int main()
{
    //nhap toa do cac dinh cua hinh hop
    nhaptodo();
    //nhap ma tran bien doi
    matranbiendoi();

    //nhap ma tran ke
    //nhapmatranke()
    //in ra ma tran ke
    inmatranke();

    //tinh toan ma tran ket qua
    cabinet();
    //in ket qua cac phep toan
    inketqua();
    //in ket qua cua phep chieu
    inhhchieu();
    //khai tao man hinh do hoa
    initwindow(600,600);
    oxy();

getch();
}
```

Phép chiếu phối cảnh (Perspective projection)

Phép chiếu phối cảnh (Perspective Projection)

Phép chiếu phối cảnh là phép chiếu mà các tia chiếu không song song với nhau mà xuất phát từ một điểm gọi là tâm chiếu.

Phép chiếu phối cảnh tạo ra hiệu ứng về luật xa gần tạo cảm giác về độ sâu của đối tượng trong thế giới thật mà phép chiếu song song không lột tả được.

Các đoạn thẳng song song của mô hình 3D sau phép chiếu hội tụ tại một điểm gọi là điểm triệt tiêu (vanishing point).

Phân loại phép chiếu phối cảnh dựa vào tâm chiếu (Centre Of Projection - COP) và mặt phẳng chiếu (projection plane)

Để đạt được phép chiếu phối cảnh của đối tượng ba chiều, chúng ta chiếu các điểm theo đường thẳng chiếu để các đường này gặp nhau ở tâm chiếu.

Ma trận biến đổi một điểm phối cảnh [Tr] có dạng:

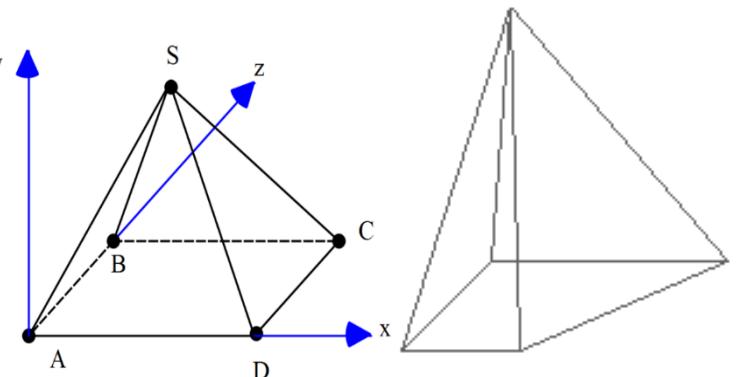
$$[x' \quad y' \quad z' \quad 1] = \left[\frac{x}{rz+1} \quad \frac{y}{rz+1} \quad 0 \quad 1 \right]$$

Ví dụ: Cho đối tượng 3D là hình chóp nhọn có tọa độ như hình.

Tâm chiếu có tọa độ (xp, yp, zp) là $(50, 50, 100)$

Hãy xác định các điểm trên mặt phẳng chiếu.

```
//X           Y           Z
a[0][0] = 100; a[1][0] = 100; a[2][0] = 100;//A
a[0][1] = 100; a[1][1] = 100; a[2][1] = 300;//B
a[0][2] = 300; a[1][2] = 100; a[2][2] = 300;//C
a[0][3] = 300; a[1][3] = 100; a[2][3] = 100;//D
a[0][4] = 200; a[1][4] = 500; a[2][4] = 200;//S
```



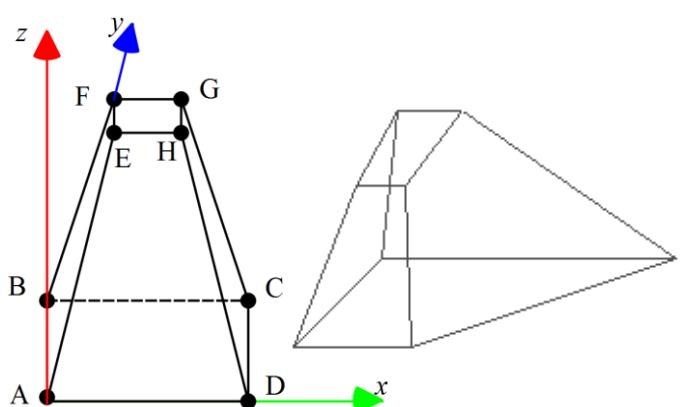
Ví dụ 2:

Cho đối tượng 3D là hình chóp cụt có tọa độ như hình.

Tâm chiếu có tọa độ (xp, yp, zp) là $(50, 50, 100)$

Hãy xác định các điểm trên mặt phẳng chiếu.

```
//X           Y           Z
a[0][0] = 100; a[1][0] = 100; a[2][0] = 100;//A
a[0][1] = 100; a[1][1] = 100; a[2][1] = 400;//B
a[0][2] = 400; a[1][2] = 100; a[2][2] = 400;//C
a[0][3] = 400; a[1][3] = 100; a[2][3] = 100;//D
a[0][4] = 200; a[1][4] = 400; a[2][4] = 200;//E
a[0][5] = 200; a[1][5] = 400; a[2][5] = 300;//F
a[0][6] = 300; a[1][6] = 400; a[2][6] = 300;//G
a[0][7] = 300; a[1][7] = 400; a[2][7] = 200;//H
```



Phép chiếu phối cảnh (Perspective projection)

Lập trình mô phỏng (Hình chót nhọn)

```
#include<stdio.h>
#include<math.h>
#include<graphics.h>
#define dinh 5

//Phep chieu cabinet. cho hinh hop chu nhat co dinh = 8
//khai bao bien
int xp, yp, zp; //tam chieu
int a[4][dinh]; //ma tran hinh hop chu nhat
int P[4][dinh]; //ma tran ket qua
int X[dinh];
int Y[dinh];
int m=5;//so dinh
int p=5;

//int D[10][10]; //ma tran ke noi cac dinh
int D[5][5]=
{
    {1,1,0,1,1},
    {1,1,1,0,1},
    {0,1,1,1,1},
    {1,0,1,1,1},
    {1,1,1,1,1},
};

//chuong trinh con
void matranke()
{
    int i,j;

    for(i=0;i<dinh;i++)
    {
        D[i][i]=1;
    }

    for(i=0;i<dinh-1;i++)
    {
        for(j=i+1;j<dinh;j++)
        {
            printf("D[%d,%d]= ",i,j);
            scanf("%d",&D[i][j]);
        }
    }

    for(i=0;i<dinh;i++)
    {
        for(j=0;j<dinh;j++)
        {
            if(i>j)
            {
                D[i][j]=D[j][i];
            }
        }
    }
}

void inmatranke()
{
    int i,j;
    //in ra ma tran ke
```

Phép chiếu phối cảnh (Perspective projection)

```
printf("\nDanh sach ma tran ke:\n");
for(i=0;i<dinh;i++)
{
    for(j=0;j<dinh;j++)
    {
        printf("%d\t",D[i][j]);
    }
    printf("\n");
}
}

void nhaptoado()
{
    //X
    Y
    Z
    a[0][0] = 100; a[1][0] = 100; a[2][0] = 100;//A
    a[0][1] = 100; a[1][1] = 100; a[2][1] = 300;//B
    a[0][2] = 300; a[1][2] = 100; a[2][2] = 300;//C
    a[0][3] = 300; a[1][3] = 100; a[2][3] = 100;//D
    a[0][4] = 200; a[1][4] = 500; a[2][4] = 200;//S

    printf("\nNhập tọa độ tam chiếu (xp, yp, zp)=\n");
    printf("xp = ");
    scanf("%d",&xp);
    printf("yp = ");
    scanf("%d",&yp);
    printf("zp = ");
    scanf("%d",&zp);

    //xp = 300; yp = 320; zp = 100;
}

void perspective_1tam()
{
    int i,j;
    int x;
    int y;
    int z;

    for(i=0;i<dinh;i++)
    {
        for(j=0;j<dinh;j++)
        {
            P[i][j]=0;
        }
    }

    //tính toán giá trị các đỉnh trên man chiếu
    for(i=0;i<dinh;i++)
    {
        x=a[0][i];
        y=a[1][i];
        z=a[2][i];

        P[0][i]= xp+(float)(x+xp)/(z+zp)*zp; //Tọa độ x'
        P[1][i]= yp+(float)(y+yp)/(z+zp)*zp;//Tọa độ y'

        //P[0][i]= (float)(x*zp)/(z+zp); //Tọa độ x'
        //P[1][i]= (float)(y*zp)/(z+zp); //Tọa độ y'
    }
}
```

Phép chiếu phối cảnh (Perspective projection)

```
    }
}

void inketqua()
{
    int i,j;

    //in toa do cac dinh
    printf("Toa do cac dinh cua hinh hop:\n");
    for(i=0;i<m;i++)
    {
        for(j=0;j<p;j++)
        {
            printf("%d\t",a[i][j]);
        }
        printf("\n");
    }

    printf("\nToa do cac dinh cua hinh hop sau khi
chieu:\n");
    for(i=0;i<m;i++)
    {
        for(j=0;j<p;j++)
        {
            printf("%d\t",P[i][j]);
        }
        printf("\n");
    }
}

//in hinh chieu ra man hinh
void inhinhchieu()
{
    int i,j;

    for(i=0; i<dinh;i++)
    {
        X[i]=P[0][i];
        Y[i]=P[1][i];
    }

    //in ra mang X va Y
    for(i=0;i<dinh;i++)
    {
        printf("(%d,%d)\t",X[i],Y[i]);
    }
}

void oxy()
{
    int i,j;

    //cai nay do minh dat ban dau, neu sau nay tu nhap tu ban phim thi
    thay bang so cu the
    setcolor(RED);
    for(i=0;i<dinh-1;i++)
    {
        for(j=i+1;j<dinh;j++)
        {
            if(D[i][j]==1)
            {
                line(X[i],Y[i],X[j],Y[j]);
                delay(100);
            }
        }
    }
}
```

Phép chiếu phối cảnh (Perspective projection)

```
    }  
}  
  
//chuong trinh chinh  
int main()  
{  
    //nhap toa do cac dinh cua hinh hop  
    nhaptodo();  
  
    //nhap ma tran ke  
    //nhapmatranke()  
  
    //in ra ma tran ke  
    inmatranke();  
  
    //tinh toan ma tran ket qua  
    perspective_1tam();  
  
    //in ket qua cac phep toan  
    inketqua();  
  
    //in ket qua cua phep chieu  
    inhinhchieu();  
  
    //khoi tao man hinh do hoa  
    initwindow(600,600);  
    oxy();  
  
    getch();  
}
```

Phép chiếu phối cảnh (Perspective projection)

Lập trình mô phỏng (Chóp cùt)

```
#include<stdio.h>
#include<math.h>
#include<graphics.h>
#define dinh 8

//Phep chieu cabinet. cho hinh hop chu nhat co dinh = 8

//khai bao bien
int xp, yp, zp; //tam chieu

int a[4][dinh]; //ma tran hinh hop chu nhat
int P[4][dinh]; //ma tran ket qua

int X[dinh];
int Y[dinh];

int m=8;//so dinh
int p=8;

//int D[10][10]; //ma tran ke noi noi cac dinh
int D[8][8]=
{
    //A,B,C,D,E,F,G,H

    {1,1,0,1,1,0,0,0},//A
    {1,1,1,0,0,1,0,0},//B
    {0,1,1,1,0,0,1,0},//C
    {1,0,1,1,0,0,0,1},//D
    {1,0,0,0,1,1,0,1},//E
    {0,1,0,0,1,1,1,0},//F
    {0,0,1,0,0,1,1,1},//G
    {0,0,0,1,1,0,1,1},//H
};

//chuong trinh con
void matranke()
{
    int i,j;

    for(i=0;i<dinh;i++)
    {
        D[i][i]=1;
    }

    for(i=0;i<dinh-1;i++)
    {
        for(j=i+1;j<dinh;j++)
        {
            printf("D[%d,%d]= ",i,j);
            scanf("%d",&D[i][j]);
        }
    }
}
```

Phép chiếu phối cảnh (Perspective projection)

```
for(i=0;i<dinh;i++)
{
    for(j=0;j<dinh;j++)
    {
        if(i>j)
        {
            D[i][j]=D[j][i];
        }
    }
}

void inmatranke()
{
    int i,j;

    //in ra ma tran ke
    printf("\nDanh sach ma tran ke:\n");
    for(i=0;i<dinh;i++)
    {
        for(j=0;j<dinh;j++)
        {
            printf("%d\t",D[i][j]);
        }
        printf("\n");
    }
}

void nhaptoado()
{
    //X
    Y
    Z
    a[0][0] = 100; a[1][0] = 100; a[2][0] = 100;//A
    a[0][1] = 100; a[1][1] = 100; a[2][1] = 400;//B
    a[0][2] = 400; a[1][2] = 100; a[2][2] = 400;//C
    a[0][3] = 400; a[1][3] = 100; a[2][3] = 100;//D
    a[0][4] = 200; a[1][4] = 400; a[2][4] = 200;//E
    a[0][5] = 200; a[1][5] = 400; a[2][5] = 300;//F
    a[0][6] = 300; a[1][6] = 400; a[2][6] = 300;//G
    a[0][7] = 300; a[1][7] = 400; a[2][7] = 200;//H

    printf("\nNhập tọa độ tam chiếu (xp,yp,zp)=\n");
    printf("xp = ");
    scanf("%d",&xp);
    printf("yp = ");
    scanf("%d",&yp);
    printf("zp = ");
    scanf("%d",&zp);

    //xp = 300; yp = 320; zp = 100;
}

void perspective_1tam()
{
    int i,j;
    int x;
    int y;
    int z;
```

Phép chiếu phối cảnh (Perspective projection)

```
for(i=0;i<dinh;i++)
{
    for(j=0;j<dinh;j++)
    {
        P[i][j]=0;
    }
}

//tinh toan gia tri cac dinh tren man chieu
for(i=0;i<dinh;i++)
{
    x=a[0][i];
    y=a[1][i];
    z=a[2][i];

    P[0][i]= xp+(float)(x+xp)/(z+zp)*zp; //Toa do x'
    P[1][i]= yp+(float)(y+yp)/(z+zp)*zp;//Toa do y'

    //P[0][i]= (float)(x*zp)/(z+zp); //Toa do x'
    //P[1][i]= (float)(y*zp)/(z+zp); //Toa do y'

}
}

void inketqua()
{
    int i,j;

    //in toa do cac dinh
    printf("Toa do cac dinh cua hinh hop:\n");
    for(i=0;i<m;i++)
    {
        for(j=0;j<p;j++)
        {
            printf("%d\t",a[i][j]);
        }
        printf("\n");
    }

    printf("\nToa do cac dinh cua hinh hop sau khi
chieu:\n");
    for(i=0;i<m;i++)
    {
        for(j=0;j<p;j++)
        {
            printf("%d\t",P[i][j]);
        }
        printf("\n");
    }
}

//in hinh chieu ra man hinh
void inhinhchieu()
{
    int i,j;

    for(i=0; i<dinh;i++)
    {
        X[i]=P[0][i];
        Y[i]=P[1][i];
    }

    //in ra mang X va Y
```

Phép chiếu phối cảnh (Perspective projection)

```
for(i=0;i<dinh;i++)
{
    printf(" (%d,%d) \t",X[i],Y[i]);
}
}

void oxy()
{
    int i,j;

    //cai nay do minh dat ban dau, neu sau nay tu nhap tu ban phim thi
    thay bang so cu the
    setcolor(GREEN);
    for(i=0;i<dinh-1;i++)
    {
        for(j=i+1;j<dinh;j++)
        {
            if(D[i][j]==1)
            {
                line(X[i],Y[i],X[j],Y[j]);
                delay(100);
            }
        }
    }
}

//chuong trinh chinh
int main()
{
    //nhap toa do cac dinh cua hinh hop
    nhaptoado();

    //nhap ma tran ke
    //nhapmatranke()

    //in ra ma tran ke
    inmatranke();

    //tinh toan ma tran ket qua
    perspective_ltam();

    //in ket qua cac phep toan
    inketqua();

    //in ket qua cua phep chieu
    inhhinhchieu();

    //khoi tao man hinh do hoa
    initwindow(600,600);
    oxy();

    getch();
}
```

Phép chiếu phối cảnh (Perspective projection)

Lập trình mô phỏng (Hình hộp)

```
#include<stdio.h>
#include<math.h>
#include<graphics.h>
#define dinh 8

//Phep chieu cabinet. cho hinh hop chu nhat co dinh = 8

//khai bao bien
int xp, yp, zp; //tam chieu

int a[4][dinh]; //ma tran hinh hop chu nhat
int P[4][dinh]; //ma tran ket qua

int X[dinh];
int Y[dinh];

int m=8;//so dinh
int p=8;

//int D[10][10]; //ma tran ke noi noi cac dinh
int D[8][8]=
{
    {1,1,0,1,1,0,0,0},//A
    {1,1,1,0,0,1,0,0},//B
    {0,1,1,1,0,0,1,0},//C
    {1,0,1,1,0,0,0,1},//D
    {1,0,0,0,1,1,0,1},//E
    {0,1,0,0,1,1,1,0},//F
    {0,0,1,0,0,1,1,1},//G
    {0,0,0,1,1,0,1,1}//H
};

//chuong trinh con
void matranke()
{
    int i,j;

    for(i=0;i<dinh;i++)
    {
        D[i][i]=1;
    }

    for(i=0;i<dinh-1;i++)
    {
        for(j=i+1;j<dinh;j++)
        {
            printf("D[%d,%d]= ",i,j);
            scanf("%d",&D[i][j]);
        }
    }
}

for(i=0;i<dinh;i++)
```

Phép chiếu phối cảnh (Perspective projection)

```
{  
    for(j=0;j<dinh;j++)  
    {  
        if(i>j)  
        {  
            D[i][j]=D[j][i];  
        }  
    }  
}  
  
void inmatranke()  
{  
    int i,j;  
  
    //in ra ma tran ke  
    printf("\nDanh sach ma tran ke:\n");  
    for(i=0;i<dinh;i++)  
    {  
        for(j=0;j<dinh;j++)  
        {  
            printf("%d\t",D[i][j]);  
        }  
        printf("\n");  
    }  
}  
  
void nhaptoado()  
{  
    //X  
    Y  
    Z  
    a[0][0] = 100; a[1][0] = 100; a[2][0] = 100;//A  
    a[0][1] = 100; a[1][1] = 100; a[2][1] = 200;//B  
    a[0][2] = 200; a[1][2] = 100; a[2][2] = 200;//C  
    a[0][3] = 200; a[1][3] = 100; a[2][3] = 100;//D  
    a[0][4] = 100; a[1][4] = 200; a[2][4] = 100;//E  
    a[0][5] = 100; a[1][5] = 200; a[2][5] = 200;//F  
    a[0][6] = 200; a[1][6] = 200; a[2][6] = 200;//G  
    a[0][7] = 200; a[1][7] = 200; a[2][7] = 100;//H  
  
    printf("\nNhập tọa độ tam chiếu (xp, yp, zp)=\n");  
    printf("xp = ");  
    scanf("%d", &xp);  
    printf("yp = ");  
    scanf("%d", &yp);  
    printf("zp = ");  
    scanf("%d", &zp);  
  
    //xp = 300; yp = 320; zp = 100;  
}  
  
void perspective_1tam()  
{  
    int i,j;  
    int x;  
    int y;  
    int z;  
  
    for(i=0;i<dinh;i++)  
    {
```

Phép chiếu phối cảnh (Perspective projection)

```
for(j=0;j<dinh;j++)
{
    P[i][j]=0;
}

//tinh toan gia tri cac dinh tren man chieu
for(i=0;i<dinh;i++)
{
    x=a[0][i];
    y=a[1][i];
    z=a[2][i];

    P[0][i]= xp+(float)(x+xp)/(z+zp)*zp; //Toa do x'
    P[1][i]= yp+(float)(y+yp)/(z+zp)*zp;//Toa do y'

    //P[0][i]= (float)(x*zp)/(z+zp); //Toa do x'
    //P[1][i]= (float)(y*zp)/(z+zp); //Toa do y'

}
}

void inketqua()
{
    int i,j;

    //in toa do cac dinh
    printf("Toa do cac dinh cua hinh hop:\n");
    for(i=0;i<m;i++)
    {
        for(j=0;j<p;j++)
        {
            printf("%d\t",a[i][j]);
        }
        printf("\n");
    }

    printf("\nToa do cac dinh cua hinh hop sau khi
chieu:\n");
    for(i=0;i<m;i++)
    {
        for(j=0;j<p;j++)
        {
            printf("%d\t",P[i][j]);
        }
        printf("\n");
    }
}

//in hinh chieu ra man hinh
void inhinhchieu()
{
    int i,j;

    for(i=0; i<dinh;i++)
    {
        X[i]=P[0][i];
        Y[i]=P[1][i];
    }

    //in ra mang X va Y
    for(i=0;i<dinh;i++)
    {
```

Phép chiếu phối cảnh (Perspective projection)

```
        printf("(%d,%d)\t",X[i],Y[i]);
    }
}

void oxy()
{
    int i,j;

    //cai nay do minh dat ban dau, neu sau nay tu nhap tu ban phim thi
thay bang so cu the
    setcolor(RED);
    for(i=0;i<dinh-1;i++)
    {
        for(j=i+1;j<dinh;j++)
        {
            if(D[i][j]==1)
            {
                line(X[i],Y[i],X[j],Y[j]);
                delay(100);
            }
        }
    }
}

//chuong trinh chinh
int main()
{
    //nhap toa do cac dinh cua hinh hop
    nhaptodo();

    //nhap ma tran ke
    //nhapmatranke()

    //in ra ma tran ke
    inmatranke();

    //tinh toan ma tran ket qua
    perspective_1tam();

    //in ket qua cac phep toan
    inketqua();

    //in ket qua cua phep chieu
    inhinhchieu();

    //khoi tao man hinh do hoa
    initwindow(600,600);
    oxy();

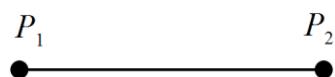
    getch();
}
```

Đường cong Bezier

Bezier đã sử dụng các điểm kiểm soát cho đường cong tại những đỉnh của đa giác và tiếp tuyến tại đó ($p_0, p_1, p_2, p_3, \dots$).

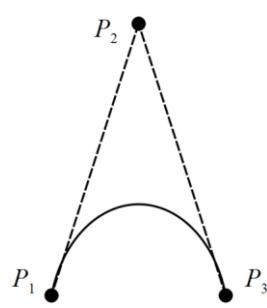
Với 2 điểm kiểm soát

(p_1, p_2) :



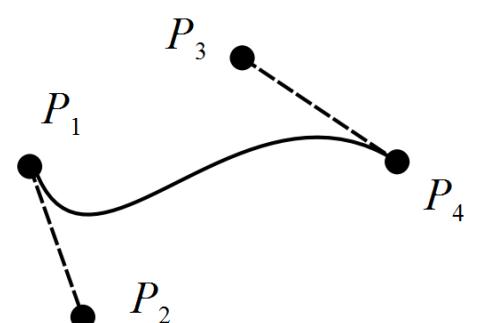
Với 3 điểm kiểm soát

(p_1, p_2, p_3) :



Với 4 điểm kiểm soát

(p_1, p_2, p_3, p_4)



Biểu thức biểu diễn đường cong Bezier:

$$p(u) = \sum_{i=0}^n p_i * B_{i,n}(u)$$

Trong đó, $B_{i,n}(u) = C_n^i * u^i * (1-u)^{n-i}$

Với $u \in [0,1]$

i, các điểm kiểm soát

Cụ thể:

n = 2

$$B_{0,2} = \frac{2!}{0!.2!} \cdot u^0 \cdot (1-u)^{2-0} = (1-u)^2$$

$$B_{1,2} = \frac{2!}{1!.1!} \cdot u^1 \cdot (1-u)^{2-1} = 2u \cdot (1-u)$$

$$B_{2,2} = \frac{2!}{2!.0!} \cdot u^2 \cdot (1-u)^{2-2} = u^2$$

$$B_{0,2} = (1-u)^2$$

$$B_{1,2} = 2u \cdot (1-u)$$

$$B_{2,2} = u^2$$

n = 3

$$B_{0,3} = \frac{3!}{0!.3!} \cdot u^0 \cdot (1-u)^{3-0} = (1-u)^3$$

$$B_{1,3} = \frac{3!}{1!.2!} \cdot u^1 \cdot (1-u)^{3-1} = 3u \cdot (1-u)^2$$

$$B_{2,3} = \frac{3!}{2!.1!} \cdot u^2 \cdot (1-u)^{3-2} = 3u^2 \cdot (1-u)$$

$$B_{3,3} = \frac{3!}{3!.0!} \cdot u^3 \cdot (1-u)^{3-3} = u^3$$

$$B_{0,3} = (1-u)^3$$

Đường cong Bezier

$$B_{1,3} = 3u \cdot (1-u)^2$$

$$B_{2,3} = 3u^2 \cdot (1-u)$$

$$B_{3,3} = u^3$$

n = 4, 5, 6 ...

Sử dụng tam giác Pascal để đổi chiều

[C-programming/Chuong1_Bai19_TamgiacPascal.c at master · thinhdoanvu/C-programming \(github.com\)](https://github.com/thinhdoanvu/C-programming/tree/master/TamgiacPascal.c)

Trong tam giác số này, bắt đầu từ hàng thứ hai, mỗi số ở hàng thứ n từ cột thứ hai đến cột n-1 bằng tổng hai số đứng ở hàng trên cùng cột và cột trước nó. Số đĩ có quan hệ này là do có công thức truy hồi $C_n^k = C_{n-1}^{k-1} + C_{n-1}^k$. (Với $1 < k < n$)

$$(a+b)^3 = a^3 + 3a^2b + 3ab^2 + b^3$$

Tam giác này có thể ứng dụng cho việc khai triển hệ số của các luỹ thừa bậc cao của các nhị thức, ví dụ:

$$(a+b)^0 = 1$$

$$(a+b)^1 = a + b$$

$$(a+b)^2 = a^2 + 2ab + b^2$$

$$(a+b)^3 = a^3 + 3a^2b + 3ab^2 + b^3$$

$$(a+b)^4 = a^4 + 4a^3b + 6a^2b^2 + 4ab^3 + b^4$$

$$(a+b)^5 = a^5 + 5a^4b + 10a^3b^2 + 10a^2b^3 + 5ab^4 + b^5$$

$$(a+b)^n = a^n + C_n^0 a^{n-1}b + \dots + C_n^1 ab^{n-1} + b^n$$

Trong đó các công thức hoán vị thay bằng các số tương ứng của tam giác Pascal theo quy tắc: luỹ thừa bậc n của nhị thức là hàng thứ n của tam giác.

Bài tập:

Xác định đường cong Bezier qua 4 điểm kiểm soát ($p_0..3$) có giá trị như sau ($n = 4-1 = 3$).

Pi cho trước.

$$Ta có: p(u) = \sum_{i=0}^n p_i * B_{i,n}(u)$$

$$p(u) = p_0 * B_{0,3}(u) + p_1 * B_{1,3}(u) + p_2 * B_{2,3}(u) + p_3 * B_{3,3}(u)$$

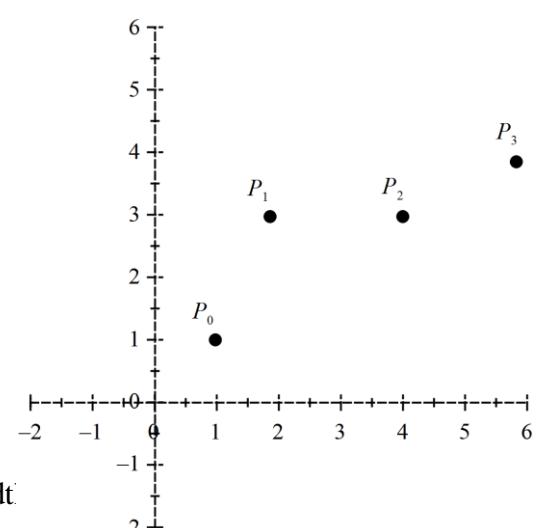
$$p(u) = p_0 * (1-u)^3 + p_1 * 3 * (1-u)^2 + p_2 * 3 * (1-u) + p_3 * u^3$$

$$p(u) = \begin{cases} p_{0,x} * (1-u)^3 + p_{1,x} * 3 * u * (1-u)^2 + p_{2,x} * 3 * (1-u) * u^2 + p_{3,x} * u^3 \\ p_{0,y} * (1-u)^3 + p_{1,y} * 3 * u * (1-u)^2 + p_{2,y} * 3 * (1-u) * u^2 + p_{3,y} * u^3 \end{cases}$$

$$p(u) = \begin{cases} 1 * (1-u)^3 + 2 * 3 * u * (1-u)^2 + 4 * 3 * (1-u) * u^2 + 6 * u^3 \\ 1 * (1-u)^3 + 3 * 3 * u * (1-u)^2 + 3 * 3 * (1-u) * u^2 + 4 * u^3 \end{cases}$$

Lập bảng tính 10 điểm

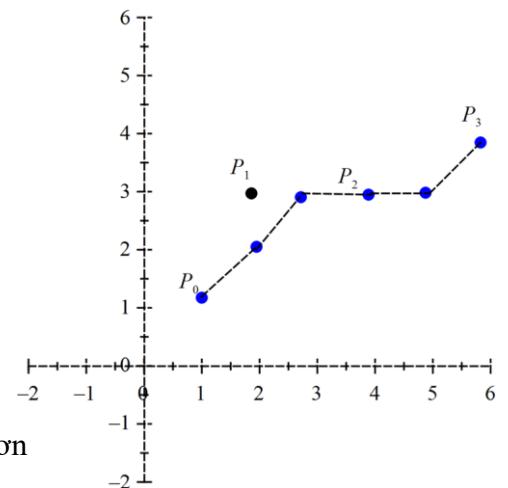
u	x(u)	y(u)	~p(u)
0	1	1	(1,1)
0.1	1.329	1.543	(1,2)
0.2	1.712	1.984	(2,2)
0.3	2.143	2.341	(2,2)
0.4	2.616	2.632	(3,3)



Hướng dẫn giải chi tiết và lập trình Kỹ thuật đồ họa © dt

Đường cong Bezier

0.5	3.125	2.875	(3,3)
0.6	3.664	3.088	(4,3)
0.7	4.227	3.289	(4,3)
0.8	4.808	3.496	(5,3)
0.9	5.401	3.727	(5,4)
1	6	4	(6,4)

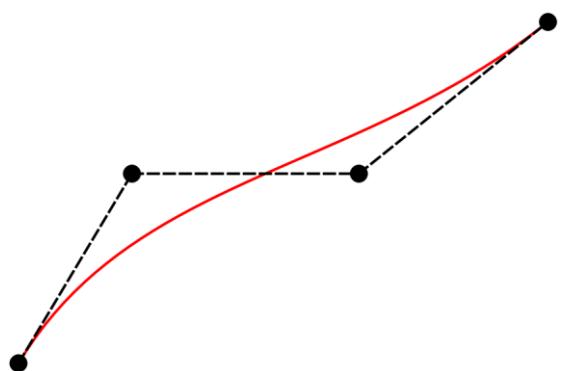


Khi tăng số lượng điểm ta sẽ có kết quả là đường sẽ mịn hơn

$n = 3$

$u \in [0,1]; u = u + 0.0005$

~ 10000 points (p0 – p9999)



Lập trình mô phỏng

```
#include<stdio.h>
#include<graphics.h>
#include<math.h>
#define max_dinh 4

int x[4], y[4];
int i;

//nhap cac dinh
void nhapdinh()
{
    for(i=0;i<4;i++)
    {
        printf("x[%d]= ",i);
        scanf("%d",&x[i]);
        printf("y[%d]= ",i);
        scanf("%d",&y[i]);
    }
}

//ve cac dinh
void vedinh()
{
    for(i=0;i<3;i++)
    {
        line(x[i],y[i],x[i+1],y[i+1]);
    }
}

void bezier()
{
```

Đường cong Bezier

```
double t;

for(t=0.0; t<1.0; t=t+0.00005)
{
    double xt=pow(1-t,3)*x[0] + 3*t*pow(1-t,2)*x[1] +
3*pow(t,2)*(1-t)*x[2] + pow(t,3)*x[3];
    double yt=pow(1-t,3)*y[0] + 3*t*pow(1-t,2)*y[1] +
3*pow(t,2)*(1-t)*y[2] + pow(t,3)*y[3];
    putpixel(xt,yt,WHITE);
}

//ve 4 dinh da set
for(i=0;i<4;i++)
{
    putpixel(x[i],y[i],RED);
}

getch();
}

int main()
{
    //nhap toa do
    nhapdinh();

    initwindow(600,600);

    //ve cac dinh
    setcolor(GREEN);
    vedinh();

    //ve duong cong
    bezier();

    getch();
}
```