**Computer Graphics**

**Practical File**

*Name*  – Tairthik Roy

*College Roll number* - 2002061

*University Roll number* – 20009570052

*Year* – 3rd Year(6th Semester)

1. Write a program to implement Bresenham’s line drawing algorithm.

#include<stdio.h>

#include<graphics.h>

void drawline(int x0, int y0, int x1, int y1)

{

    int dx, dy, p, x, y;

    dx=x1-x0;

    dy=y1-y0;

    x=x0;

    y=y0;

    p=2\*dy-dx;

    while(x<x1)

    {

        if(p>=0)

        {

            putpixel(x,y,7);

            y=y+1;

            p=p+2\*dy-2\*dx;

        }

        else

        {

            putpixel(x,y,7);

            p=p+2\*dy;}

            x=x+1;

        }

}

int main()

{

    int gdriver=DETECT, gmode, error, x0, y0, x1, y1;

    initgraph(&gdriver, &gmode, NULL);

    printf("Enter co-ordinates of first point: ");

    scanf("%d%d", &x0, &y0);

    printf("Enter co-ordinates of second point: ");

    scanf("%d%d", &x1, &y1);

    drawline(x0, y0, x1, y1);

    return 0;

}



2. Write a program to implement mid-point circle drawing algorithm.

#include <iostream>

#include <graphics.h>

using namespace std;

int xCentre;

int yCentre;

void drawPoint(int x, int y)

{

    putpixel(x + xCentre, y + yCentre, WHITE);

    putpixel(y + yCentre, x + xCentre, WHITE);

    putpixel(-x + xCentre, y + yCentre, WHITE);

    putpixel(y + yCentre, -x + xCentre, WHITE);

    putpixel(x + xCentre, -y + yCentre, WHITE);

    putpixel(-y + yCentre, x + xCentre, WHITE);

    putpixel(-x + xCentre, -y + yCentre, WHITE);

    putpixel(-y + yCentre, -x + xCentre, WHITE);

}

int main()

{

    initwindow(960, 540);

    int radius;

    cout << "Enter Radius of the Circle : ";

    cin >> radius;

    cout << "Enter X Co-ordinate : ";

    cin >> xCentre;

    cout << "Enter Y Co-ordinate : ";

    cin >> yCentre;

    float d = float(5 / 4) - radius;

    int x = 0;

    int y = radius;

    drawPoint(x, y);

    while (y > x)

    {

        if (d < 0)

        {

            d = d + 2 \* x + 3;

            x++;

        }

        else

        {

            d = d + 2 \* x - 2 \* y + 5;

            x++;

            y--;

        }

        drawPoint(x, y);

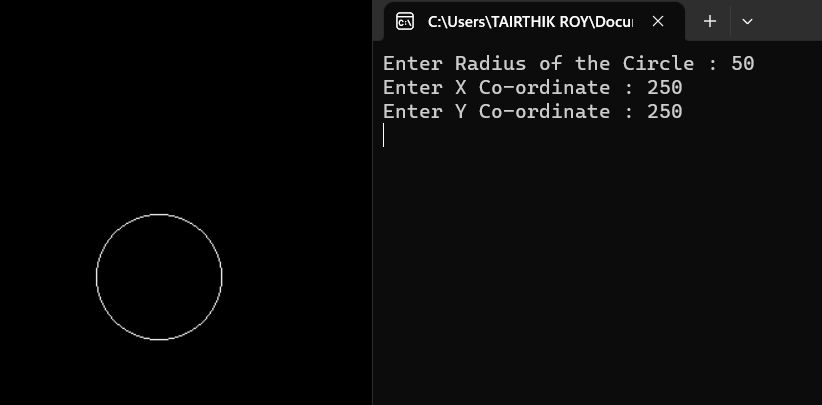
    }

    getch();

    closegraph();

    return 0;

}



3. Write a program to clip a line using Cohen and Sutherland line clipping algorithm.

#include <bits/stdc++.h>

#include <graphics.h>

using namespace std;

int xmin, xmax, ymin, ymax;

struct lines {

    int x1, y1, x2, y2;

};

int sign(int x)

{

    if (x > 0)

        return 1;

    else

        return 0;

}

void clip(struct lines mylines)

{

    int bits[4], byte[4], i, var;

    setcolor(RED);

    bits[0] = sign(xmin - mylines.x1);

    byte[0] = sign(xmin - mylines.x2);

    bits[1] = sign(mylines.x1 - xmax);

    byte[1] = sign(mylines.x2 - xmax);

    bits[2] = sign(ymin - mylines.y1);

    byte[2] = sign(ymin - mylines.y2);

    bits[3] = sign(mylines.y1 - ymax);

    byte[3] = sign(mylines.y2 - ymax);

    string initial = "", end = "", temp = "";

    for (i = 0; i < 4; i++) {

        if (bits[i] == 0)

            initial += '0';

        else

            initial += '1';

    }

    for (i = 0; i < 4; i++) {

        if (byte[i] == 0)

            end += '0';

        else

            end += '1';

    }

    float m = (mylines.y2 - mylines.y1) / (float)(mylines.x2 - mylines.x1);

    float c = mylines.y1 - m \* mylines.x1;

    if (initial == end && end == "0000") {

        line(mylines.x1, mylines.y1, mylines.x2, mylines.y2);

        return;

    }

    else {

        for (i = 0; i < 4; i++) {

            int val = (bits[i] & byte[i]);

            if (val == 0)

                temp += '0';

            else

                temp += '1';

        }

        if (temp != "0000")

            return;

        for (i = 0; i < 4; i++) {

            if (bits[i] == byte[i])

                continue;

            if (i == 0 && bits[i] == 1) {

                var = round(m \* xmin + c);

                mylines.y1 = var;

                mylines.x1 = xmin;

            }

            if (i == 0 && byte[i] == 1) {

                var = round(m \* xmin + c);

                mylines.y2 = var;

                mylines.x2 = xmin;

            }

            if (i == 1 && bits[i] == 1) {

                var = round(m \* xmax + c);

                mylines.y1 = var;

                mylines.x1 = xmax;

            }

            if (i == 1 && byte[i] == 1) {

                var = round(m \* xmax + c);

                mylines.y2 = var;

                mylines.x2 = xmax;

            }

            if (i == 2 && bits[i] == 1) {

                var = round((float)(ymin - c) / m);

                mylines.y1 = ymin;

                mylines.x1 = var;

            }

            if (i == 2 && byte[i] == 1) {

                var = round((float)(ymin - c) / m);

                mylines.y2 = ymin;

                mylines.x2 = var;

            }

            if (i == 3 && bits[i] == 1) {

                var = round((float)(ymax - c) / m);

                mylines.y1 = ymax;

                mylines.x1 = var;

            }

            if (i == 3 && byte[i] == 1) {

                var = round((float)(ymax - c) / m);

                mylines.y2 = ymax;

                mylines.x2 = var;

            }

            bits[0] = sign(xmin - mylines.x1);

            byte[0] = sign(xmin - mylines.x2);

            bits[1] = sign(mylines.x1 - xmax);

            byte[1] = sign(mylines.x2 - xmax);

            bits[2] = sign(ymin - mylines.y1);

            byte[2] = sign(ymin - mylines.y2);

            bits[3] = sign(mylines.y1 - ymax);

            byte[3] = sign(mylines.y2 - ymax);

        }

        initial = "", end = "";

        for (i = 0; i < 4; i++) {

            if (bits[i] == 0)

                initial += '0';

            else

                initial += '1';

        }

        for (i = 0; i < 4; i++) {

            if (byte[i] == 0)

                end += '0';

            else

                end += '1';

        }

        if (initial == end && end == "0000") {

            line(mylines.x1, mylines.y1, mylines.x2, mylines.y2);

            return;

        }

        else

            return;

    }

}

int main()

{

    int gd = DETECT, gm;

    xmin = 40;

    xmax = 100;

    ymin = 40;

    ymax = 80;

    initgraph(&gd, &gm, NULL);

    line(xmin, ymin, xmax, ymin);

    line(xmax, ymin, xmax, ymax);

    line(xmax, ymax, xmin, ymax);

    line(xmin, ymax, xmin, ymin);

    struct lines mylines[4];

    mylines[0].x1 = 30;

    mylines[0].y1 = 65;

    mylines[0].x2 = 55;

    mylines[0].y2 = 30;

    mylines[1].x1 = 60;

    mylines[1].y1 = 20;

    mylines[1].x2 = 100;

    mylines[1].y2 = 90;

    mylines[2].x1 = 60;

    mylines[2].y1 = 100;

    mylines[2].x2 = 80;

    mylines[2].y2 = 70;

    mylines[3].x1 = 85;

    mylines[3].y1 = 50;

    mylines[3].x2 = 120;

    mylines[3].y2 = 75;

    for (int i = 0; i < 4; i++) {

        line(mylines[i].x1, mylines[i].y1,

             mylines[i].x2, mylines[i].y2);

        delay(1000);

    }

    for (int i = 0; i < 4; i++) {

        clip(mylines[i]);

        delay(1000);

    }

    delay(4000);

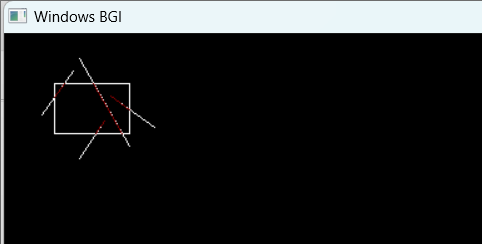
    getch();

    closegraph();

    return 0;

}





4. Write a program to clip a polygon using Sutherland Hodgeman algorithm.

#include<iostream>

using namespace std;

#include<conio.h>

#include<graphics.h>

#define round(a) ((int)(a+0.5))

int k;

float xmin,ymin,xmax,ymax,arr[20],m;

void clipl(float x1,float y1,float x2,float y2)

{

    if(x2-x1)

        m=(y2-y1)/(x2-x1);

    else

        m=100000;

    if(x1 >= xmin && x2 >= xmin)

    {

        arr[k]=x2;

        arr[k+1]=y2;

        k+=2;

    }

    if(x1 < xmin && x2 >= xmin)

    {

        arr[k]=xmin;

        arr[k+1]=y1+m\*(xmin-x1);

        arr[k+2]=x2;

        arr[k+3]=y2;

        k+=4;

    }

    if(x1 >= xmin  && x2 < xmin)

    {

        arr[k]=xmin;

        arr[k+1]=y1+m\*(xmin-x1);

        k+=2;

    }

}

void clipt(float x1,float y1,float x2,float y2)

{

    if(y2-y1)

        m=(x2-x1)/(y2-y1);

    else

        m=100000;

    if(y1 <= ymax && y2 <= ymax)

    {

        arr[k]=x2;

        arr[k+1]=y2;

        k+=2;

    }

    if(y1 > ymax && y2 <= ymax)

    {

        arr[k]=x1+m\*(ymax-y1);

        arr[k+1]=ymax;

        arr[k+2]=x2;

        arr[k+3]=y2;

        k+=4;

    }

    if(y1 <= ymax  && y2 > ymax)

    {

        arr[k]=x1+m\*(ymax-y1);

        arr[k+1]=ymax;

        k+=2;

    }

}

void clipr(float x1,float y1,float x2,float y2)

{

    if(x2-x1)

        m=(y2-y1)/(x2-x1);

    else

        m=100000;

    if(x1 <= xmax && x2 <= xmax)

    {

        arr[k]=x2;

        arr[k+1]=y2;

        k+=2;

    }

    if(x1 > xmax && x2 <= xmax)

    {

        arr[k]=xmax;

        arr[k+1]=y1+m\*(xmax-x1);

        arr[k+2]=x2;

        arr[k+3]=y2;

        k+=4;

    }

    if(x1 <= xmax  && x2 > xmax)

    {

        arr[k]=xmax;

        arr[k+1]=y1+m\*(xmax-x1);

        k+=2;

    }

}

void clipb(float x1,float y1,float x2,float y2)

{

    if(y2-y1)

        m=(x2-x1)/(y2-y1);

    else

        m=100000;

    if(y1 >= ymin && y2 >= ymin)

    {

        arr[k]=x2;

        arr[k+1]=y2;

        k+=2;

    }

    if(y1 < ymin && y2 >= ymin)

    {

        arr[k]=x1+m\*(ymin-y1);

        arr[k+1]=ymin;

        arr[k+2]=x2;

        arr[k+3]=y2;

        k+=4;

    }

    if(y1 >= ymin  && y2 < ymin)

    {

        arr[k]=x1+m\*(ymin-y1);

        arr[k+1]=ymin;

        k+=2;

    }

}

int main()

{

    int gdriver=DETECT,gmode,n,poly[20];

    float xi,yi,xf,yf,polyy[20];

    cout<<"Coordinates of rectangular clip window :\nxmin,ymin             :";

    cin>>xmin>>ymin;

    cout<<"xmax,ymax             :";

    cin>>xmax>>ymax;

    cout<<"\n\nPolygon to be clipped :\nNumber of sides       :";

    cin>>n;

    cout<<"Enter the coordinates :";

    int i = 0;

    for(i=0;i < 2\*n;i++)

        cin>>polyy[i];

    polyy[i]=polyy[0];

    polyy[i+1]=polyy[1];

    for(i=0;i < 2\*n+2;i++)

        poly[i]=round(polyy[i]);

    initgraph(&gdriver,&gmode,NULL);

    setcolor(RED);

    rectangle(xmin,ymax,xmax,ymin);

    cout<<"\t\tPolygon unclipped";

    setcolor(WHITE);

    fillpoly(n,poly);

    getch();

    cleardevice();

    k=0;

    for(i=0;i < 2\*n;i+=2)

        clipl(polyy[i],polyy[i+1],polyy[i+2],polyy[i+3]);

    n=k/2;

    for(i=0;i < k;i++)

        polyy[i]=arr[i];

    polyy[i]=polyy[0];

    polyy[i+1]=polyy[1];

    k=0;

    for(i=0;i < 2\*n;i+=2)

        clipt(polyy[i],polyy[i+1],polyy[i+2],polyy[i+3]);

    n=k/2;

    for(i=0;i < k;i++)

        polyy[i]=arr[i];

    polyy[i]=polyy[0];

    polyy[i+1]=polyy[1];

    k=0;

    for(i=0;i < 2\*n;i+=2)

        clipr(polyy[i],polyy[i+1],polyy[i+2],polyy[i+3]);

    n=k/2;

    for(i=0;i < k;i++)

        polyy[i]=arr[i];

    polyy[i]=polyy[0];

    polyy[i+1]=polyy[1];

    k=0;

    for(i=0;i < 2\*n;i+=2)

        clipb(polyy[i],polyy[i+1],polyy[i+2],polyy[i+3]);

    for(i=0;i < k;i++)

        poly[i]=round(arr[i]);

    if(k)

        fillpoly(k/2,poly);

    setcolor(RED);

    rectangle(xmin,ymax,xmax,ymin);

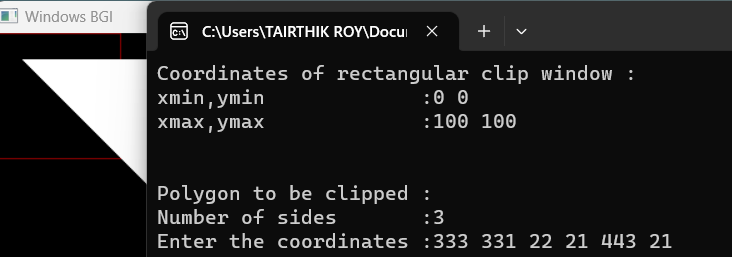
    cout<<"\tClipped polygon";

    getch();

    closegraph();

    return 0;

}



5. Write a program to fill a polygon using Scan line fill algorithm.

#include <conio.h>

#include <iostream>

#include <graphics.h>

#include <stdlib.h>

using namespace std;

class point

{

    public:

    int x,y;

};

class poly

{

        point p[20];

        int inter[20],x,y;

        int v,xmin,ymin,xmax,ymax;

    public:

        int c;

        void read();

        void calcs();

        void display();

        void ints(float);

        void sort(int);

};

void poly::read()

{

    int i;

    cout<<"\n Enter the no of vertices of polygon:";

    cin>>v;

    if(v>2)

    {

        for(i=0;i<v; i++)

        {

            cout<<"\nEnter the co-ordinate no.- "<<i+1<<" : ";

            cout<<"\n\tx"<<(i+1)<<"=";

            cin>>p[i].x;

            cout<<"\n\ty"<<(i+1)<<"=";

            cin>>p[i].y;

        }

        p[i].x=p[0].x;

        p[i].y=p[0].y;

        xmin=xmax=p[0].x;

        ymin=ymax=p[0].y;

    }

    else

        cout<<"\n Enter valid no. of vertices.";

}

void poly::calcs()

{

    for(int i=0;i<v;i++)

    {

        if(xmin>p[i].x)

        xmin=p[i].x;

        if(xmax<p[i].x)

        xmax=p[i].x;

        if(ymin>p[i].y)

        ymin=p[i].y;

        if(ymax<p[i].y)

        ymax=p[i].y;

    }

}

void poly::display()

{

    int ch1;

    char ch='y';

    float s,s2;

    do

    {

        cout<<"\n\nMENU:";

        cout<<"\n\n\t1 . Scan line Fill ";

        cout<<"\n\n\t2 . Exit ";

        cout<<"\n\nEnter your choice:";

        cin>>ch1;

        switch(ch1)

        {

            case 1:

                s=ymin+0.01;

                delay(100);

                cleardevice();

                while(s<=ymax)

                {

                    ints(s);

                    sort(s);

                    s++;

                }

                break;

            case 2:

                exit(0);

        }

        cout<<"Do you want to continue?: ";

        cin>>ch;

    }while(ch=='y' || ch=='Y');

}

void poly::ints(float z)

{

    int x1,x2,y1,y2,temp;

    c=0;

    for(int i=0;i<v;i++)

    {

        x1=p[i].x;

        y1=p[i].y;

        x2=p[i+1].x;

        y2=p[i+1].y;

        if(y2<y1)

        {

            temp=x1;

            x1=x2;

            x2=temp;

            temp=y1;

            y1=y2;

            y2=temp;

        }

        if(z<=y2&&z>=y1)

        {

            if((y1-y2)==0)

            x=x1;

            else

            {

                x=((x2-x1)\*(z-y1))/(y2-y1);

                x=x+x1;

            }

            if(x<=xmax && x>=xmin)

            inter[c++]=x;

        }

    }

}

void poly::sort(int z)

{

    int temp,j,i;

        for(i=0;i<v;i++)

        {

            line(p[i].x,p[i].y,p[i+1].x,p[i+1].y);

        }

        delay(100);

        for(i=0; i<c;i+=2)

        {

            delay(100);

            line(inter[i],z,inter[i+1],z);

        }

}

int main()

    {

    int cl;

    initwindow(500,600);

    cleardevice();

    poly x;

    x.read();

    x.calcs();

    cleardevice();

    cout<<"\n\tEnter the colour u want:(0-15)->";

    cin>>cl;

    setcolor(cl);

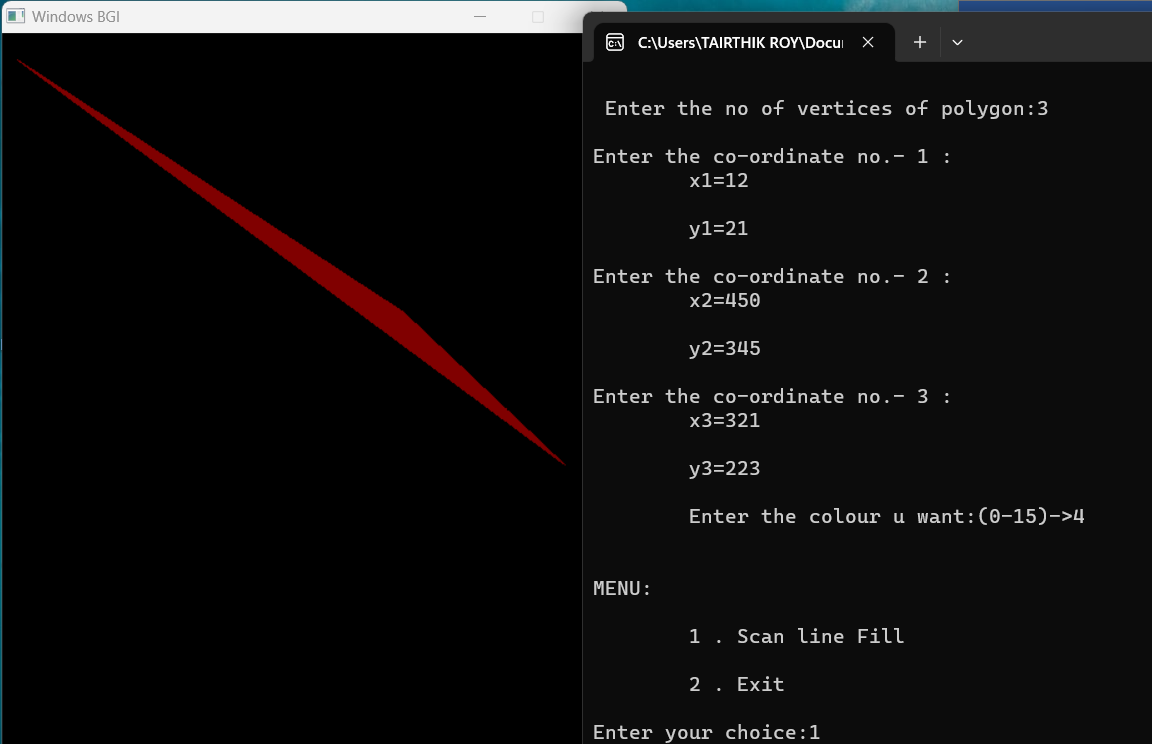
    x.display();

    closegraph();

    getch();

    return 0;

}



6. Write a program to apply various 2D transformations on a 2D object (use homogenous 64 Coordinates).

#include <graphics.h>

#include <stdlib.h>

#include <stdio.h>

#include <conio.h>

#include<math.h>

void main()

{

    int gm;

    int gd=DETECT;

    int x1,x2,x3,y1,y2,y3,nx1,nx2,nx3,ny1,ny2,ny3,c;

    int sx,sy,xt,yt,r;

    float t;

    initgraph(&gd,&gm,"c:\tc\bg:");

    printf("\t Program for basic transactions");

    printf("\n\t Enter the points of triangle");

    setcolor(1);

    scanf("%d%d%d%d%d%d",&x1,&y1,&x2,&y2,&x3,&y3);

    line(x1,y1,x2,y2)

    line(x2,y2,x3,y3);

    line(x3,y3,x1,y1);

    getch();

    printf("\n 1.Transaction\n 2.Rotation\n 3.Scalling\n 4.exit");

    printf("Enter your choice:");

    scanf("%d",&c);

        switch(c)

        {

            case 1:

                printf("\n Enter the translation factor");

                scanf("%d%d",&xt,&yt);

                nx1=x1+xt;

                ny1=y1+yt;

                nx2=x2+xt;

                ny2=y2+yt;

                nx3=x3+

                ny3=y3+yt;

                line(nx1,ny1,nx2,ny2);

                line(nx2,ny2,nx3,ny3);

                line(nx3,ny3,nx1,ny1);

                getch();

            case 2:

                printf("\n Enter the angle of rotation");

                scanf("%d",&r);

                t=3.14\*r/180;

                nx1=abs(x1\*cos(t)-y1\*sin(t));

                ny1=abs(x1\*sin(t)+y1\*cos(t));

                nx2=abs(x2\*cos(t)-y2\*sin(t));

                ny2=abs(x2\*sin(t)+y2\*cos(t));

                nx3=abs(x3\*cos(t)-y3\*sin(t));

                ny3=abs(x3\*sin(t)+y3\*cos(t));

                line(nx1,ny1,nx2,ny2);

                line(nx2,ny2,nx3,ny3);

                line(nx3,ny3,nx1,ny1);

                getch();

            case 3:

                printf("\n Enter the scalling factor");

                scanf("%d%d",&sx,&sy);

                nx1=x1\*sx;

                ny1=y2\*sy;

                nx2=x2\*sx;

                ny2=y2\*sy;

                nx3=x3\*sx;

                ny3=y3\*sy;

                line(nx1,ny1,nx2,ny2);

                line(nx2,ny2,nx3,ny3);

                line(nx3,ny3,nx1,ny1);

                getch();

            case 4:

                break;

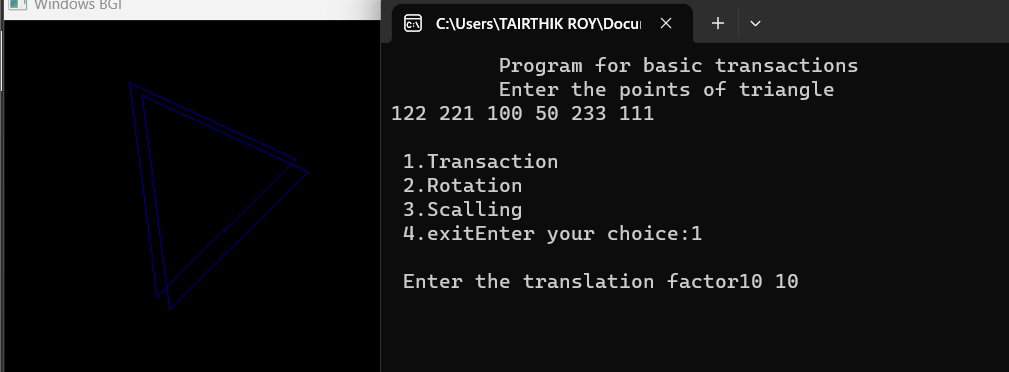
            default:

                printf("Enter the correct choice");

        }

    closegraph();

}



7. Write a program to apply various 3D transformations on a 3D object and then apply parallel and perspective projection on it.

#include<graphics.h>

#include<iostream>

#include<conio.h>

using namespace std;

int gd,gm,x1,y1,x2,y2,dep,ch;

int main()

{

cout<<"\n Enter top-left and bottom-right corner:";

cin>>x1>>y1>>x2>>y2;

cout<<"\n Enter the depth along z axis:";

cin>>dep;

do

{

cout<<"Choose any one projection:\n\t1.Parallel Projection\n\t2.Perspective Projection\nEnter your choice:";

cin>>ch;

initgraph(&gd,&gm,NULL);

switch(ch)

{

case 1:

rectangle(x2+100,y1,x2+100+dep,y2);

outtextxy(x2+100,y1-10,NULL);

rectangle(x1,y1,x2,y2);

outtextxy(x1,y1-10,NULL);

rectangle(x1,y1-(y2-y1),x2,x1+dep-(y2-y1));

outtextxy(x1,y1-(y2-y1)-10,NULL);

getch();

closegraph();

break;

case 2:

bar3d(x1,y1,x2,y2,dep,1);

getch();

closegraph();

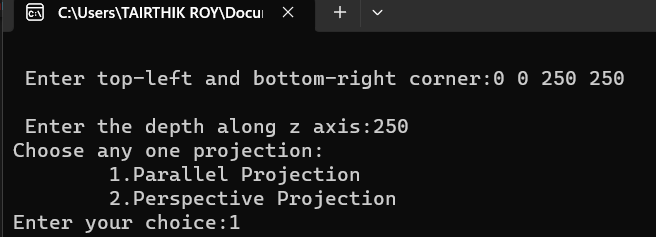
break;

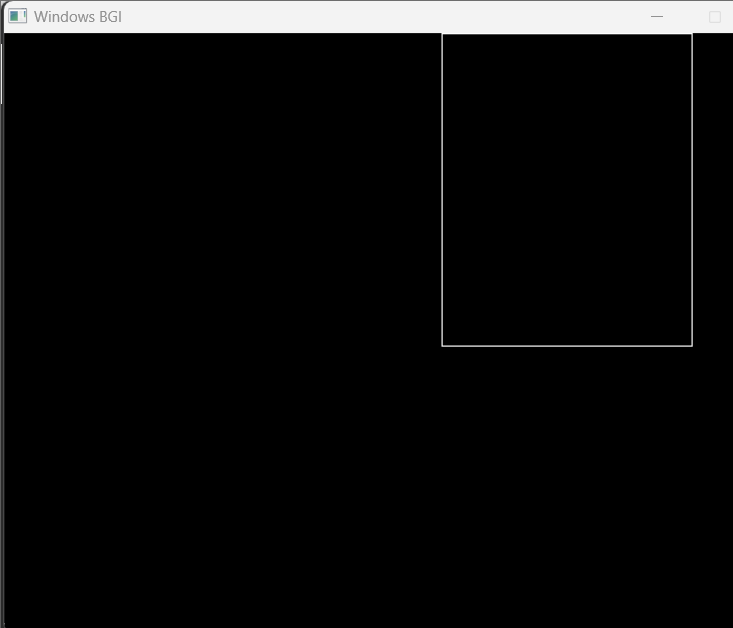
}

}while(ch<3);

return 0;

}





8. Write a program to draw Hermite /Bezier curve.

#include <stdio.h>

#include <graphics.h>

#include <math.h>

int x[4]={200,100,200,250};

int y[4]={200,150,75,100};

void bezier ()

{

int i;

double t,xt,yt;

for (t = 0.0; t < 1.0; t += 0.0005)

{

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];

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);

}

for (i=0; i<4; i++)

 putpixel (x[i], y[i], YELLOW);

getch();

closegraph();

}

int main()

{

int gd = DETECT, gm;

initgraph (&gd, &gm, NULL);

bezier ();

return 0;

}

