

## Name : Anirvan Goswami

## University Roll no. :- 19020570004

## Course :- B. Sc. (H) Computer Science

## Subject :- Computer Graphics

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

**#include <cmath>**

**#include <cstdlib>**

**#include <graphics.h>**

**#include <iostream>**

**using namespace std;**

**void bresenhamLine(int x0, int y0, int x1, int y1, int val)**

**{**

**if (x0 == x1 && y0 == y1)**

**{**

**putpixel(x1, y1, val);**

**}**

**else**

**{**

**int dx = x1 - x0;**

**int dy = y1 - y0;**

**float m = float(dy) / (float)(dx);**

**if (m >= 1 || m <= 0)**

**{**

**cout << "ERROR: Slope must be between 0 and 1." << endl;**

**exit(1);**

**}**

**int d = 2 \* dy - dx;**

**int del\_E = 2 \* dy;**

**int del\_NE = 2 \* (dy - dx);**

**int x = x0;**

**int y = y0;**

**putpixel(x, y, val);**

**while (x < x1)**

**{**

**if (d <= 0)**

**{**

**d += del\_E;**

**x += 1;**

**}**

**else**

**{**

**d += del\_NE;**

**x += 1;**

**y += 1;**

**}**

**putpixel(x, y, val);**

**}**

**}**

**return;**

**}**

**int main(void)**

**{**

**int x0, y0, x1, y1;**

**cout << "Enter Left Endpoint (x0 y0): ";**

**cin >> x0 >> y0;**

**cout << "Enter Right Endpoint (x1 y1): ";**

**cin >> x1 >> y1;**

**cout << "Drawing Line..." << endl;**

**int gd = DETECT, gm;**

**initgraph(&gd, &gm, NULL);**

**bresenhamLine(x0, y0, x1, y1, WHITE);**

**delay(5e3);**

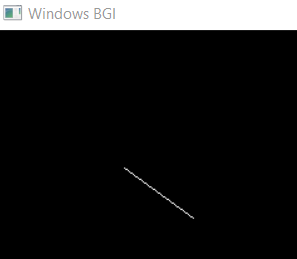
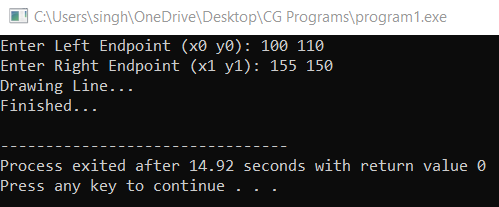
**closegraph();**

**cout << "Finished..." << endl;**

**return 0;**

**}**

**Output:**



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

**#include<graphics.h>**

**#include<stdlib.h>**

**#include<stdio.h>**

**#include<iostream>**

**#include<math.h>**

**#include<conio.h>**

**void plot8(int x, int y, int cx, int cy)**

**{**

**putpixel(x+cx,y+cy,GREEN);**

**putpixel(y+cx,x+cy,GREEN);**

**putpixel(y+cx,-x+cy,GREEN);**

**putpixel(x+cx,-y+cy,GREEN);**

**putpixel(-x+cx,-y+cy,GREEN);**

**putpixel(-y+cx,-x+cy,GREEN);**

**putpixel(-y+cx,x+cy,GREEN);**

**putpixel(-x+cx,y+cy,GREEN);**

**}**

**int main()**

**{**

**int gdriver = DETECT, gmode, errorcode;**

**int x,y,r,cx,cy,E,SE,D;**

**initgraph(&gdriver, &gmode, "..\\bgi");**

**errorcode = graphresult();**

**if (errorcode != grOk)**

**{**

**printf("Graphics error: %s\n", grapherrormsg(errorcode));**

**printf("Press any key to halt:");**

**getch();**

**exit(1);**

**}**

**std::cout<<"\nEnter centre(x): ";**

**std::cin>>cx;**

**std::cout<<"\nEnter centre(y): ";**

**std::cin>>cy;**

**std::cout<<"\nEnter radius: ";**

**std::cin>>r;**

**x=0;**

**y=r;**

**plot8(x,y,cx,cy);**

**D=1-r;**

**while(y>x)**

**{**

**if(D<=0)**

**{**

**E=(2\*x)+3;**

**D+=E;**

**x++;**

**}**

**else**

**{**

**SE=(2\*x)-(2\*y)+5;**

**D+=SE;**

**x++;**

**y--;**

**}**

**plot8(x,y,cx,cy);**

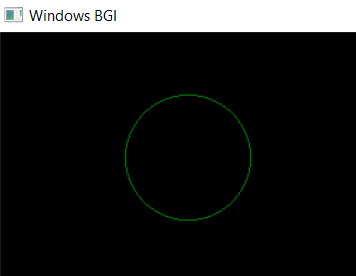
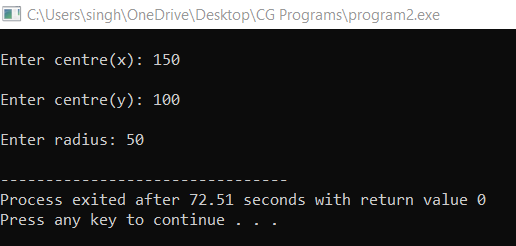
**}**

**getch();**

**closegraph();**

**}**

**Output:**



**Program 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], bite[4], i, var;**

**setcolor(YELLOW);**

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

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

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

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

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

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

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

**bite[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 (bite[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] & bite[i]);**

**if (val == 0)**

**temp += '0';**

**else**

**temp += '1';**

**}**

**if (temp != "0000")**

**return;**

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

**if (bits[i] == bite[i])**

**continue;**

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

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

**mylines.y1 = var;**

**mylines.x1 = xmin;**

**}**

**if (i == 0 && bite[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 && bite[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 && bite[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 && bite[i] == 1) {**

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

**mylines.y2 = ymax;**

**mylines.x2 = var;**

**}**

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

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

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

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

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

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

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

**bite[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 (bite[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);**

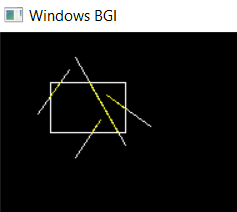
**getch();**

**closegraph();**

**return 0;**

**}**

**Output:**



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

**#include<graphics.h>**

**#include<iostream>**

**#include<stdlib.h>**

**#include<stdio.h>**

**#include<conio.h>**

**int poly[100],xmin,xmax,ymin,ymax,n;**

**void left\_clip(int xmin,int ymin,int xmax,int ymax)**

**{**

**int temp[100],i,j=0,count=0,x1,y1,x2,y2;**

**for(i=0;i<2\*n;i+=2)**

**{**

**x1 = poly[i];**

**y1 = poly[i+1];**

**x2 = poly[i+2];**

**y2 = poly[i+3];**

**if(x1<xmin && x2<xmin)**

**{**

**}**

**else if(x1>=xmin && x2>=xmin)**

**{**

**temp[j] = x2;**

**temp[j+1] = y2;**

**j+=2;**

**count++;**

**}**

**else if(x1=xmin)**

**{**

**int x=xmin;**

**int y= y1 + (xmin-x1)\*(float)(y2-y1)/(x2-x1);**

**temp[j]=x;**

**temp[j+1]=y;**

**temp[j+2]=x2;**

**temp[j+3]=y2;**

**j+=4;**

**count+=2;**

**}**

**else**

**{**

**int x=xmin;**

**int y= y1 + (xmin-x1)\*(float)(y2-y1)/(x2-x1);**

**temp[j] = x;**

**temp[j+1] = y;**

**j+=2;**

**count++;**

**}**

**}**

**n=count;**

**temp[j]=temp[0];**

**temp[j+1]=temp[1];**

**for(i=0;i<2\*(n+1);i++)**

**{**

**poly[i]=temp[i];**

**}**

**}**

**void right\_clip(int xmin,int ymin,int xmax,int ymax)**

**{**

**int temp[100],i,j=0,count=0,x1,y1,x2,y2;**

**for(i=0;i<2\*n;i+=2)**

**{**

**x1 = poly[i];**

**y1 = poly[i+1];**

**x2 = poly[i+2];**

**y2 = poly[i+3];**

**if(x1>xmax && x2>xmax)**

**{**

**}**

**if(x1<=xmax && x2<=xmax)**

**{**

**temp[j] = x2;**

**temp[j+1] = y2;**

**j+=2;**

**count+=1;**

**}**

**else if(x1>xmax && x2<=xmax)**

**{**

**int x=xmax;**

**int y= y1 + (xmax-x1)\*(float)(y2-y1)/(x2-x1);**

**temp[j]=x;**

**temp[j+1]=y;**

**temp[j+2]=x2;**

**temp[j+3]=y2;**

**j+=4;**

**count+=2;**

**}**

**else**

**{**

**int x=xmax;**

**int y= y1 + (xmax-x1)\*(float)(y2-y1)/(x2-x1);**

**temp[j] = x;**

**temp[j+1] = y;**

**j+=2;**

**count+=1;**

**}**

**}**

**n=count;**

**temp[j]=temp[0];**

**temp[j+1]=temp[1];**

**for(i=0;i<2\*(n+1);i++)**

**{**

**poly[i]=temp[i];**

**}**

**}**

**void bottom\_clip(int xmin,int ymin,int xmax,int ymax)**

**{**

**int temp[100],i,j=0,count=0,x1,y1,x2,y2;**

**for(i=0;i<2\*n;i+=2)**

**{**

**x1 = poly[i];**

**y1 = poly[i+1];**

**x2 = poly[i+2];**

**y2 = poly[i+3];**

**if(y1<ymin && y2<ymin)**

**{**

**}**

**else if(y1>=ymin && y2>=ymin)**

**{**

**temp[j] = x2;**

**temp[j+1] = y2;**

**j+=2;**

**count++;**

**}**

**else if(y1<ymin && y2>=ymin)**

**{**

**int x= x1 +(float)((ymin-y1)\*(x2-x1))/(y2-y1);**

**int y= ymin;**

**temp[j]=x;**

**temp[j+1]=y;**

**temp[j+2]=x2;**

**temp[j+3]=y2;**

**j+=4;**

**count+=2;**

**}**

**else**

**{**

**int x= x1 +(float)((ymin-y1)\*(x2-x1))/(y2-y1);**

**int y= ymin;**

**temp[j] = x;**

**temp[j+1] = y;**

**j+=2;**

**count++;**

**}**

**}**

**n=count;**

**temp[j]=temp[0];**

**temp[j+1]=temp[1];**

**for(i=0;i<2\*(n+1);i++)**

**{**

**poly[i]=temp[i];**

**}**

**}**

**void top\_clip(int xmin,int ymin,int xmax,int ymax)**

**{**

**int temp[100],i,j=0,count=0,x1,y1,x2,y2;**

**for(i=0;i<2\*n;i+=2)**

**{**

**x1 = poly[i];**

**y1 = poly[i+1];**

**x2 = poly[i+2];**

**y2 = poly[i+3];**

**if(y1>ymax && y2>ymax)**

**{**

**}**

**else if(y1<=ymax && y2<=ymax)**

**{**

**temp[j] = x2;**

**temp[j+1] = y2;**

**j+=2;**

**count++;**

**}**

**else if(y1>ymax && y2<=ymax)**

**{**

**int x= x1 + (ymax-y1)\*(float)(x2-x1)/(y2-y1);**

**int y= ymax;**

**temp[j]=x;**

**temp[j+1]=y;**

**temp[j+2]=x2;**

**temp[j+3]=y2;**

**j+=4;**

**count+=2;**

**}**

**else**

**{**

**int x= x1 + (ymax-y1)\*(float)(x2-x1)/(y2-y1);**

**int y= ymax;**

**temp[j] = x;**

**temp[j+1] = y;**

**j+=2;**

**count++;**

**}**

**}**

**n=count;**

**temp[j]=temp[0];**

**temp[j+1]=temp[1];**

**for(i=0;i<2\*(n+1);i++)**

**{**

**poly[i]=temp[i];**

**}**

**}**

**int main(void)**

**{**

**int gd = DETECT, gm, errorcode;**

**int xmax, ymax;**

**initgraph(&gd, &gm, "C:\\TC\\BGI");**

**errorcode = graphresult();**

**if (errorcode != grOk)**

**{**

**printf("Graphics error: %s\n", grapherrormsg(errorcode));**

**printf("Press any key to halt:");**

**getch();**

**exit(1);**

**}**

**int i;**

**setcolor(RED);**

**xmin=200;**

**ymin=200;**

**xmax=300;**

**ymax=300;**

**rectangle(100,100,300,300);**

**std::cout<<"Enter the no. of vertices: ";**

**std::cin>>n;**

**for(i=0;i<2\*n;i+=2)**

**{**

**std::cout<<"Enter co-ordinates of vertex "<<(i/2+1)<<": ";**

**std::cin>>poly[i]>>poly[i+1];**

**}**

**setcolor(YELLOW);**

**poly[2\*n]=poly[0];**

**poly[2\*n+1]=poly[1];**

**drawpoly(n+1,poly);**

**left\_clip(xmin,ymin,xmax,ymax);**

**right\_clip(xmin,ymin,xmax,ymax);**

**bottom\_clip(xmin,ymin,xmax,ymax);**

**top\_clip(xmin,ymin,xmax,ymax);**

**std::cout<<"after clipping"<<std::endl;**

**setcolor(BLUE);**

**drawpoly(n+1,poly);**

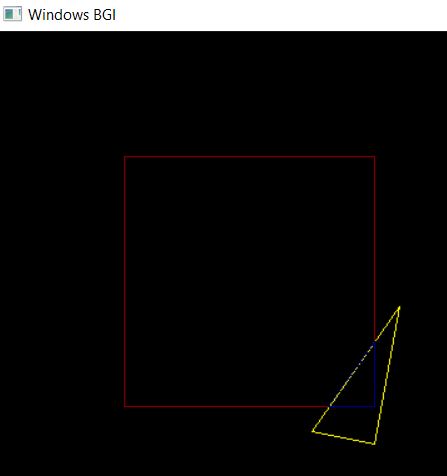
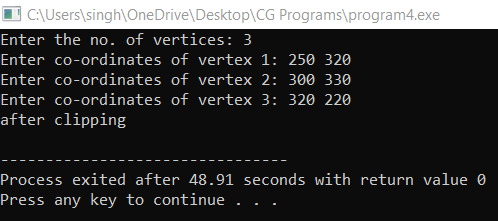
**getch();**

**closegraph();**

**return 0;**

**}**

**Output:**



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

**#include<iostream>**

**#include<conio.h>**

**#include<graphics.h>**

**using namespace std;**

**struct edge**

**{**

**int x1,y1,x2,y2,flag;**

**};**

**int main()**

**{**

**int gd=DETECT,gm,n,i,j,k;**

**edge ed[10], temped;**

**float dx,dy,m[10],x\_int[10],inter\_x[10];**

**int x[10],y[10],ymax=0,ymin=480,yy,temp;**

**initgraph(&gd,&gm,"");**

**cout<<"Enter the no. of vertices of the polygon : ";**

**cin>>n;**

**cout<<"\nEnter the vertices :- \n";**

**for(i=0;i<n;i++)**

**{**

**cout<<"P"<<i+1<<" : ";**

**cin>>x[i]>>y[i];**

**if(y[i]>ymax)**

**ymax=y[i];**

**if(y[i]<ymin)**

**ymin=y[i];**

**ed[i].x1=x[i];**

**ed[i].y1=y[i];**

**}**

**for(i=0;i<n-1;i++)**

**{**

**ed[i].x2=ed[i+1].x1;**

**ed[i].y2=ed[i+1].y1;**

**ed[i].flag=0;**

**}**

**ed[i].x2=ed[0].x1;**

**ed[i].y2=ed[0].y1;**

**ed[i].flag=0;**

**for(i=0;i<n;i++)**

**{**

**if(ed[i].y1 < ed[i].y2)**

**{**

**temp=ed[i].x1;**

**ed[i].x1=ed[i].x2;**

**ed[i].x2=temp;**

**temp=ed[i].y1;**

**ed[i].y1=ed[i].y2;**

**ed[i].y2=temp;**

**}**

**}**

**for(i=0;i<n;i++)**

**{**

**line(ed[i].x1, ed[i].y1,ed[i].x2,ed[i].y2);**

**}**

**for(i=0;i<n-1;i++)**

**{**

**for(j=0;j<n-1;j++)**

**{**

**if(ed[j].y1<ed[j+1].y1)**

**{**

**temped=ed[j];**

**ed[j]=ed[j+1];**

**ed[j+1]=temped;**

**}**

**if(ed[j].y1==ed[j+1].y1)**

**{**

**if(ed[j].y2<ed[j+1].y2)**

**{**

**temped=ed[j];**

**ed[j]=ed[j+1];**

**ed[j+1]=temped;**

**}**

**if (ed[j].y2==ed[j+1].y2)**

**{**

**if(ed[j].x1<ed[j+1].x1)**

**{**

**temped=ed[j];**

**ed[j]=ed[j+1];**

**ed[j+1]=temped;**

**}**

**}**

**}**

**}**

**}**

**for(i=0;i<n;i++)**

**{**

**dx=ed[i].x2-ed[i].x1;**

**dy=ed[i].y2-ed[i].y1;**

**if(dy==0)**

**{**

**m[i]=0;**

**}**

**else**

**{**

**m[i]=dx/dy;**

**}**

**inter\_x[i]=ed[i].x1;**

**}**

**yy=ymax;**

**while(yy>ymin)**

**{**

**for(i=0;i<n;i++)**

**{**

**if(yy>ed[i].y2 && yy<=ed[i].y1)**

**{**

**ed[i].flag=1;**

**}**

**else**

**{**

**ed[i].flag=0;**

**}**

**}**

**j=0;**

**for(i=0;i<n;i++)**

**{**

**if(ed[i].flag==1)**

**{**

**if(yy==ed[i].y1)**

**{**

**x\_int[j]==ed[i].x1;**

**j++;**

**if(ed[i-1].y1==yy && ed[i-1].y1<yy)**

**{**

**x\_int[j]=ed[i].x1;**

**j++;**

**}**

**if(ed[i+1].y1==yy && ed[i+1].y1<yy)**

**{**

**x\_int[j]=ed[i].x1;**

**j++;**

**}**

**}**

**else**

**{**

**x\_int[j]=inter\_x[i]+(-m[i]);**

**inter\_x[i]=x\_int[j];**

**j++;**

**}**

**}**

**}**

**for(i=0;i<j;i++)**

**{**

**for(k=0;k<j-1;k++)**

**{**

**if(x\_int[k]>x\_int[k+1])**

**{**

**temp=(int)x\_int[k];**

**x\_int[k]=x\_int[k+1];**

**x\_int[k+1]=temp;**

**}**

**}**

**}**

**for(i=0;i<j;i=i+2)**

**{**

**line((int)x\_int[i],yy,(int)x\_int[i+1],yy);**

**}**

**yy--;**

**delay(10);**

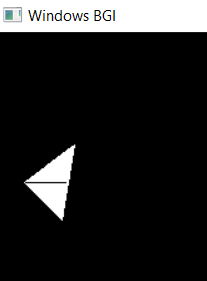
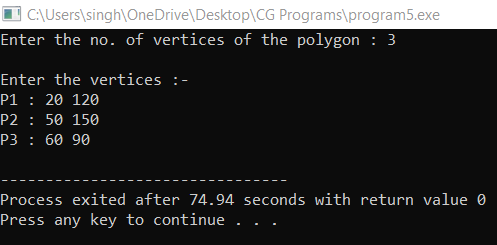
**}**

**getch();**

**return 0;**

**}**

**Output:**



**Program 6: Write a program to apply various 2D transformations on a 2D object (use homogenous coordinates).**

**#include <conio.h>**

**#include <iostream>**

**#include <graphics.h>**

**using namespace std;**

**int c = 400;**

**class point**

**{**

**public:**

**float x,y;**

**};**

**class matrix**

**{**

**private:**

**point p[20];**

**point t[2];**

**point pp[1];**

**int points;**

**public:**

**void read()**

**{**

**setcolor(2);**

**cout<<"Enter no. of points : ";**

**cin>>points;**

**cout<<"\nCAUTION : Enter value between 0 and 400 for x and y values!!!\n";**

**cout<<"\nEnter coordinates :-\n";**

**for(int i=0;i<points;i++)**

**{**

**cout<<"Enter P"<<i+1<<" : ";**

**cin>>p[i].x>>p[i].y;**

**}**

**}**

**void readT()**

**{**

**cout<<"\nEnter Transformation matrix (2X2) :- \n";**

**for(int i=0;i<2;i++)**

**cin>>t[i].x>>t[i].y;**

**}**

**void transform()**

**{**

**setcolor(3);**

**for(int i=0;i<points;i++)**

**{**

**pp[0].x = p[i].x\*t[0].x + p[i].y\*t[1].x;**

**pp[0].y = p[i].x\*t[0].y + p[i].y\*t[1].y;**

**p[i].x = (int)(pp[0].x);**

**p[i].y = (int)(pp[0].y);**

**}**

**}**

**void draw()**

**{**

**for(int i=0;i<points;i++)**

**line(c+p[i].x, c-p[i].y, c+p[(i+1)%points].x, c-p[(i+1)%points].y);**

**}**

**void show()**

**{**

**cout<<endl;**

**for(int i=0;i<points;i++)**

**cout<<p[i].x<<" "<<p[i].y<<endl;**

**}**

**};**

**void printaxis()**

**{**

**setcolor(YELLOW);**

**line(c,0,c,c\*2);**

**line(0,c,c\*2,c);**

**for(int i=50;i<c\*2;i+=50) line(c-10,i,c+10,i);**

**for(int i=50;i<c\*2;i+=50) line(i,c-10,i,c+10);**

**setcolor(WHITE);**

**}**

**int main()**

**{**

**initwindow(c\*2,c\*2);**

**printaxis();**

**matrix m;**

**m.read();**

**m.show();**

**m.draw();**

**m.readT();**

**m.transform();**

**m.show();**

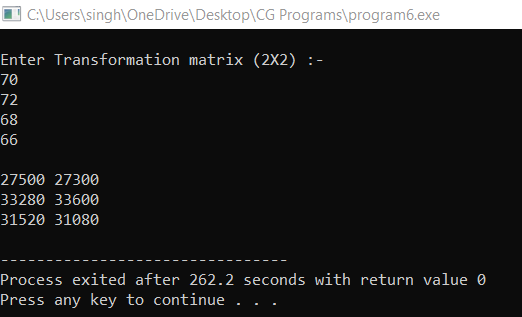
**m.draw();**

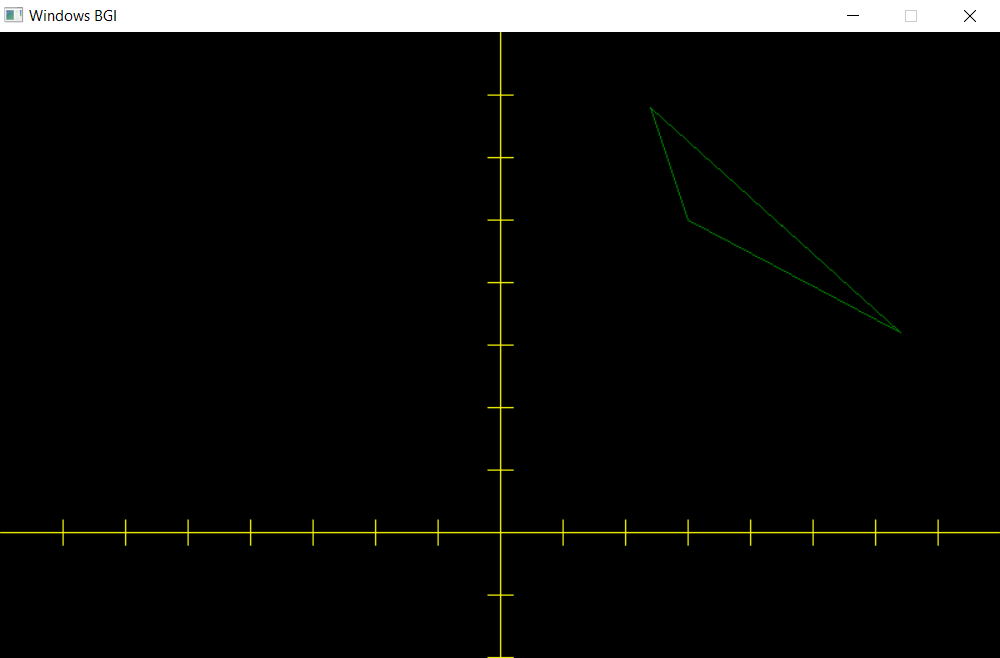
**getch();**

**return 0;**

**}**

**Output:**





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

**#include<iostream>**

**#include<math.h>**

**#include<conio.h>**

**#include<graphics.h>**

**using namespace std;**

**class matrix**

**{**

**int nodes[10][4];**

**float T[4][4];**

**int size;**

**public:**

**matrix(int s)**

**{**

**size = s;**

**cout << "\nThe number of nodes are : \n" << size;**

**for(int i =0;i<size;++i)**

**nodes[i][3] = 1;**

**}**

**void input()**

**{**

**cout << endl;**

**for(int i=0;i<size;++i)**

**{**

**cout << "Enter P" << i<<" : ";**

**for(int j=0;j<3;++j)**

**cin >> nodes[i][j];**

**}**

**cout << "\nBefore\n";**

**drawMy(nodes);**

**cout << endl;**

**}**

**void drawMy(int x[][4])**

**{**

**cout << "\nTransformation"<<endl;**

**for(int i=0;i<size;++i)**

**{**

**line(200 + x[i%size][0],200 + x[i%size][1], 200 + x[(i+1)%size][0],200 + x[(i+1)%size][1]);**

**}**

**cout << endl;**

**}**

**void rotation()**

**{**

**setcolor(RED);**

**for(int i=0;i<size;++i)**

**{**

**for(int j=0;j<4;++j)**

**{**

**cout << nodes[i][j] << " ";**

**}**

**cout << endl;**

**}**

**int angle;**

**cout << "Enter angle along x axis : ";**

**cin >> angle;**

**float r = angle\*(3.14/180);**

**float T[4][4];**

**T[0][0] = 1;**

**for (int i=1;i<=3;++i)**

**T[0][i] = 0;**

**for(int i=1;i<=3;++i)**

**T[i][0] = 0;**

**T[3][1] = 0;**

**T[3][2] = 0;**

**T[3][3] = 1;**

**T[0][3] = 0;**

**T[1][3] = 0;**

**T[2][3] = 0;**

**T[1][1] = cos(r);**

**T[1][2] = sin(r);**

**T[2][1] = -1\*sin(r);**

**T[2][2] = cos(r);**

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

**{**

**for(int j=0;j<4;++j)**

**{**

**cout << T[i][j] << " ";**

**}**

**cout << endl;**

**}**

**float rr[3][4];**

**int n = size;**

**cout << "\nSize is : " << n << endl;**

**for(int i=0;i<n;i++)**

**for(int j=0;j<4;j++)**

**rr[i][j]=0;**

**for(int i=0;i<size;i++)**

**{**

**for(int j=0;j<4;j++)**

**{**

**rr[i][j] = 0;**

**for(int k=0;k<4;k++)**

**rr[i][j] += nodes[i][k]\*T[k][j];**

**}**

**}**

**cout << endl;**

**for(int i=0;i<size;++i)**

**{**

**for(int j=0;j<4;++j)**

**cout << rr[i][j] << " ";**

**cout << endl;**

**}**

**for(int i=0;i<size;++i)**

**{**

**line(round(200 + rr[i%size][0]),round(200 + rr[i%size][1]), round(200 + rr[(i+1)%size][0]),round(200 + rr[(i+1)%size][1]));**

**}**

**}**

**};**

**int main()**

**{**

**int gdriver = DETECT, gmode;**

**initgraph(&gdriver, &gmode, "");**

**matrix m(3);**

**m.input();**

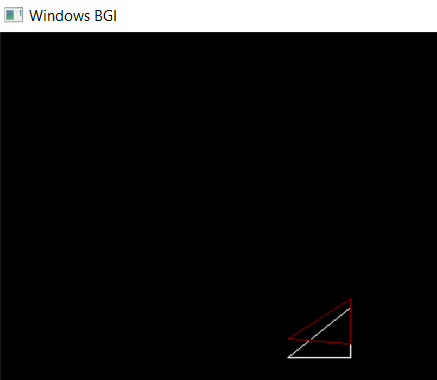
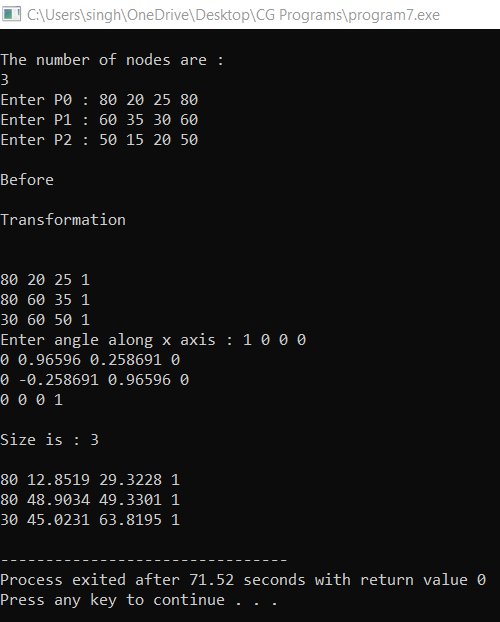
**m.rotation();**

**getch();**

**return 0;**

**}**

**Output:**



**Program 8: Write a program to draw Bezier/Hermite Curve.**

**#include<stdio.h>**

**#include<graphics.h>**

**#include<iostream>**

**#include<conio.h>**

**#include<stdlib.h>**

**#include<math.h>**

**void bezier(int x[4], int y[4])**

**{**

**double t;**

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

**{**

**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,BLUE);**

**}**

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

**putpixel(x[i],y[i],YELLOW);**

**getch();**

**closegraph();**

**return;**

**}**

**int main()**

**{**

**int gdriver = DETECT, gmode, errorcode;**

**initgraph(&gdriver, &gmode, "..\\bgi");**

**errorcode = graphresult();**

**if (errorcode != grOk)**

**{**

**printf("Graphics error: %s\n", grapherrormsg(errorcode));**

**printf("Press any key to halt:");**

**getch();**

**exit(1);**

**}**

**int x[4],y[4];**

**int i;**

**std::cout<<"Enter x and y coordinates"<<std::endl;**

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

**{**

**std::cin>>x[i];**

**std::cout<<std::endl;**

**std::cin>>y[i];**

**}**

**bezier(x,y);**

**}**

**Output:**

