**JAGAN INSTITUTE OF MANAGEMENT STUDIES**

**PRACTICAL**

**FILE**

**ON**

**COMPUTER GRAPHICS**

**Submitted To: Submitted My:**

**Mr. Jasmeet Singh Sunny Khurana MCA 3rd Sem**

**05450404414**

**INDEX**

|  |  |  |
| --- | --- | --- |
| S.No. | TOPIC | Sign |
| 1. | Smiley Animation Program . |  |
| 2. | DDA Line Drawing Program |  |
| 3. | Bresenham's Line Drawing Program |  |
| 4. | Bresenham's Circle Drawing Program |  |
| 5. | Midpoint Circle Drawing Program |  |
| 6. | Polygon creation and drawing program |  |
| 7. | Polygon Translation Program |  |
| 8. | Polygon Rotation Program (About an arbitrary point) |  |
| 9. | Polygon Scaling Program |  |
| 10. | Polygon Reflection Program (About an arbitrary line) |  |
| 11. | Polygon Shearing along x-axis Program |  |
| 12. | Polygon Shearing along y-axis Program |  |

**CERTIFICATE**

This is to certify that Sunny Khurana of MCA 3rdSemester has successfully completed the practical file on **Computer Graphics** for MCA III Practical examination of the GGSIPU in the year 2014. It is further certified that this project is the individual work of the Candidate.

**Mr. Jasmeet Singh**

Signature:

Date:

ACKNOWLEDGEMENT

I, Sunny Khurana**,** wish to express gratitude to all those who helped and co-operated me and enabled me to complete **Computer Graphics** practical file. I express sincere thanks to our teacher, **Mr. Jasmeet Singh Sir** for his valuable suggestion, informative and illuminative guidance.

Through this column, it would be my utmost pleasure to express warm thanks to his encouragement, co-operation and consent

Sunny Khurana

05450404414

Batch 2014 – 2017

**Q: WAP TO IMPLEMENT ANIMATED SMILEY.**

#include<iostream>

#include<graphics.h>

#include<stdio.h>

#include<conio.h>

#include<dos.h>

int main()

{

int gd=DETECT,gm;

initgraph(&gd, &gm, "..//bgi");

int i;

for(i=0;i<100;i++)

{

cleardevice();

line(150+i,150,100+i,100);

line(200+i,250,200+i,200);

line(250+i,150,300+i,100);

arc(300,250,200,350,50);

circle(200+i,200,150);

delay(10);

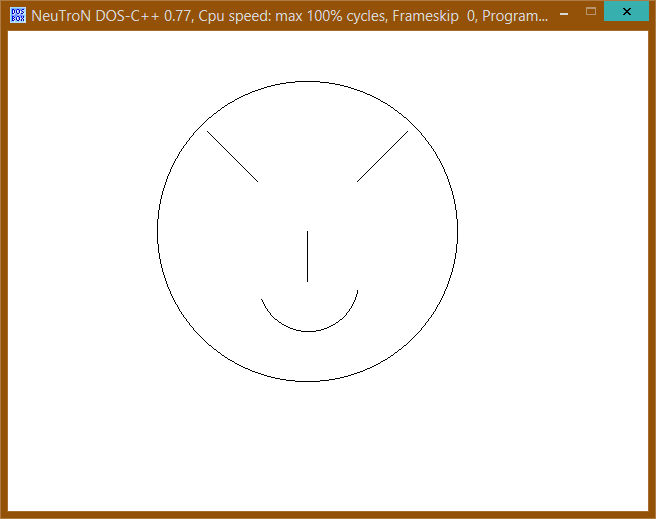
}

getch();

return(0);

}

**OUTPUT:**



**Q: WAP TO IMPLEMENT DDA LINE DRAWING ALGORITHM.**

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

#include<math.h>

void main()

{

int gdriver=DETECT,gmode;

int x0,y0,x1,y1,value,x;

double dx,dy,m,y;

initgraph(&gdriver,&gmode,”..//bgi”);

printf(“\n enter initial coordinate of line : ”);

scanf(“%d%d”,&x0,&y0);

printf(“you entered %d%d\n”,x0,y0);

printf(“\n enter end coordinate of line : ”);

scanf(“%d%d”,&x1,&y1);

printf(“you entered %d%d\n”,x1,y1);

printf(“enter the value of color in which you want to see output(1-14) : ”);

scanf(“%d”,value);

dy=y1-y0;

dx=x1-x0;

m=dy/dx;

y=y0;

for(x=x0;x<=x1;x++)

{

putpixel(x,floor(y),value);

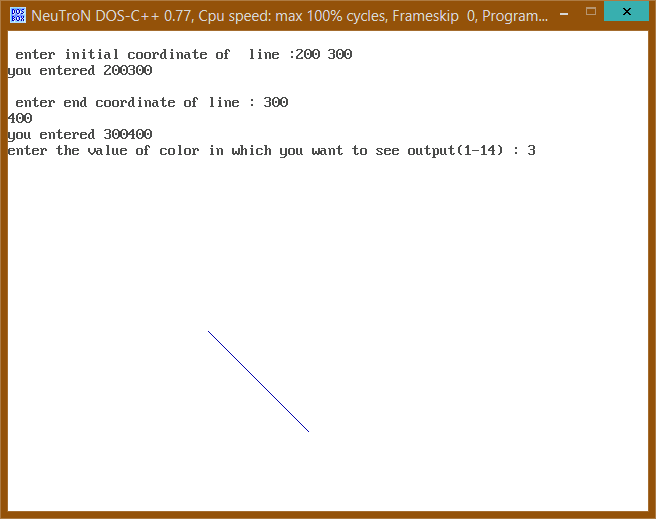
y+=m;

}

getch();

closegraph();

}



**Q: WAP TO IMPLEMENT BRESENHAM’S LINE DRAWING ALGORITHM.**

#include <stdio.h>

#include <dos.h>

#include <graphics.h>

#include <conio.h>

void lineBres(int, int, int, int);

void main()

{

int x1, y1, xn, yn;

int gd = DETECT, gm;

initgraph(&gd, &gm, "..//bgi");

printf("Enter starting coordinates of line: ");

scanf("%d %d", &x1, &y1);

printf("Enter ending coordinates of line: ");

scanf("%d %d", &xn, &yn);

lineBres(x1, y1, xn, yn);

getch();

}

void lineBres(int x1, int y1, int xn, int yn)

{

int dx = xn - x1, dy = yn - y1;

int di = 2 \* dy - dx;

int ds = 2 \* dy, dt = 2 \* (dy - dx);

putpixel(x1, y1, RED);

while (x1 < xn)

{

x1++;

if (di < 0)

di = di + ds;

else

{

y1++;

di = di + dt;

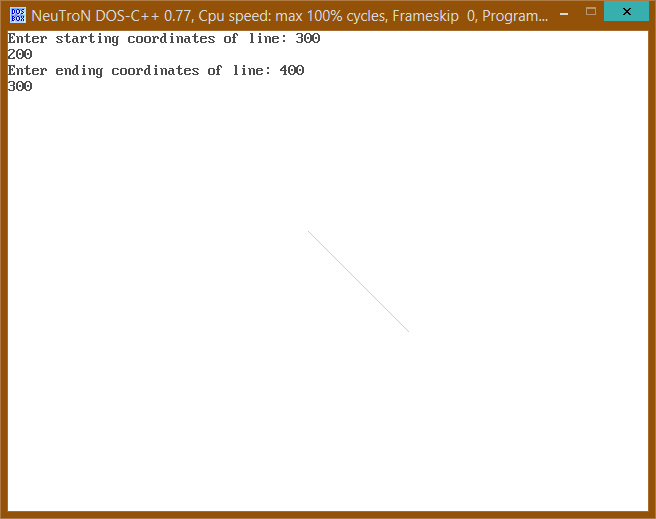
}

putpixel(x1, y1, RED);

delay(20);

}

}



**Q: WAP TO IMPLEMENT BRESENHAM’S CIRCLE DRAWING ALGORITHM.**

#include <stdio.h>

#include <dos.h>

#include <graphics.h>

#include<conio.h>

void circleBres(int, int, int);

void drawCircle(int, int, int, int);

void main()

{

int xc, yc, r;

int gd = DETECT, gm;

initgraph(&gd, &gm, "..//bgi");

printf("Enter center coordinates of circle: ");

scanf("%d %d", &xc, &yc);

printf("Enter radius of circle: ");

scanf("%d", &r);

circleBres(xc, yc, r);

getch();

}

void circleBres(int xc, int yc, int r)

{

int x = 0, y = r;

int d = 3 - 2 \* r;

while (x < y)

{

drawCircle(xc, yc, x, y);

x++;

if (d < 0)

d = d + 4 \* x + 6;

else

{

y--;

d = d + 4 \* (x - y) + 10;

}

drawCircle(xc, yc, x, y);

delay(50);

}

}

void drawCircle(int xc, int yc, int x, int y)

{

putpixel(xc+x, yc+y, RED);

putpixel(xc-x, yc+y, RED);

putpixel(xc+x, yc-y, RED);

putpixel(xc-x, yc-y, RED);

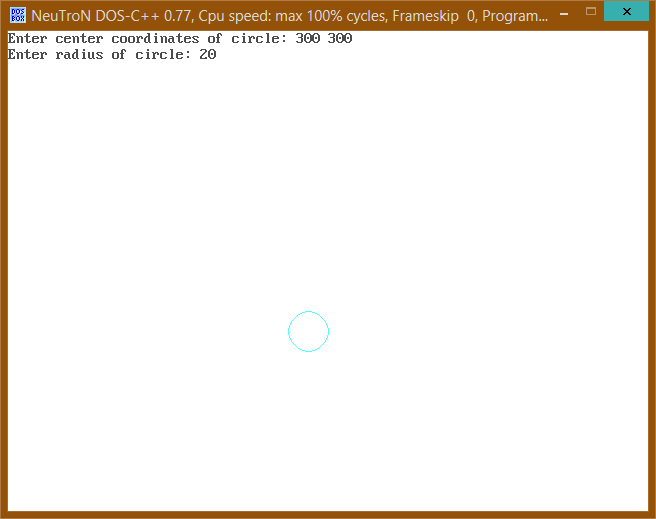
putpixel(xc+y, yc+x, RED);

putpixel(xc-y, yc+x, RED);

putpixel(xc+y, yc-x, RED);

putpixel(xc-y, yc-x, RED);

}



**Q: WAP TO IMPLEMENT MIDPOINT CIRCLE DRAWING ALGORITHM.**

#include <stdio.h>

#include <dos.h>

#include <graphics.h>

#include <conio.h>

void circleMidpoint(int, int, int);

void drawCircle(int, int, int, int);

void main()

{

int xc, yc, r;

int gd = DETECT, gm;

initgraph(&gd, &gm, "..//BGI");

printf("Enter center coordinates of circle: ");

scanf("%d %d", &xc, &yc);

printf("Enter radius of circle: ");

scanf("%d", &r);

circleMidpoint(xc, yc, r);

getch();

}

void circleMidpoint(int xc, int yc, int r)

{

int x = 0, y = r;

int p = 1 - r;

while (x < y)

{

drawCircle(xc, yc, x, y);

x++;

if (p < 0)

p = p + 2 \* x + 1;

else

{

y--;

p = p + 2 \* (x - y) + 1;

}

drawCircle(xc, yc, x, y);

delay(50);

}

}

void drawCircle(int xc, int yc, int x, int y)

{

putpixel(xc+x, yc+y, RED);

putpixel(xc-x, yc+y, RED);

putpixel(xc+x, yc-y, RED);

putpixel(xc-x, yc-y, RED);

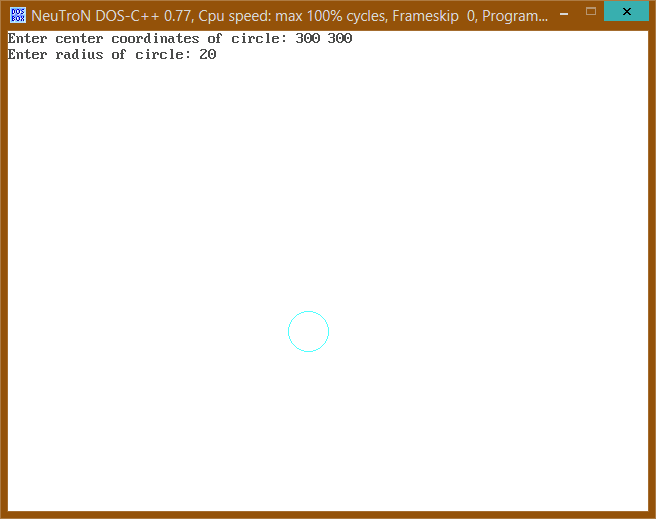
putpixel(xc+y, yc+x, RED);

putpixel(xc-y, yc+x, RED);

putpixel(xc+y, yc-x, RED);

putpixel(xc-y, yc-x, RED);

}



**Q: WAP TO IMPLEMENT MIDPOINT ELLIPSE DRAWING ALGORITHM.**

#include <stdio.h>

#include <dos.h>

#include <graphics.h>

#include<conio.h>

void ellipseMidpoint(float, float, float, float);

void drawEllipse(float, float, float, float);

void main()

{

float xc, yc, rx, ry;

int gd = DETECT, gm;

initgraph(&gd, &gm, "..//BGI");

printf("\nEnter the center coordinates of ellipse: ");

scanf("%f %f", &xc, &yc);

printf("\nEnter x-radius coordinate: ");

scanf("%f", &rx);

printf("\nEnter y-radius coordiante: ");

scanf("%f", &ry);

ellipseMidpoint(xc, yc, rx, ry);

getch();

}

void ellipseMidpoint(float xc, float yc, float rx, float ry)

{

float rxSq = rx \* rx;

float rySq = ry \* ry;

float x = 0, y = ry, p;

float px = 0, py = 2 \* rxSq \* y;

drawEllipse(xc, yc, x, y);

//Region 1

p = rySq - (rxSq \* ry) + (0.25 \* rxSq);

while (px < py)

{

x++;

px = px + 2 \* rySq;

if (p < 0)

p = p + rySq + px;

else

{

y--;

py = py - 2 \* rxSq;

p = p + rySq + px - py;

}

drawEllipse(xc, yc, x, y);

delay(30);

}

//Region 2

p = rySq\*(x+0.5)\*(x+0.5) + rxSq\*(y-1)\*(y-1) - rxSq\*rySq;

while (y > 0)

{

y--;

py = py - 2 \* rxSq;

if (p > 0)

p = p + rxSq - py;

else

{

x++;

px = px + 2 \* rySq;

p = p + rxSq - py + px;

}

drawEllipse(xc, yc, x, y);

delay(30);

}

}

void drawEllipse(float xc, float yc, float x, float y)

{

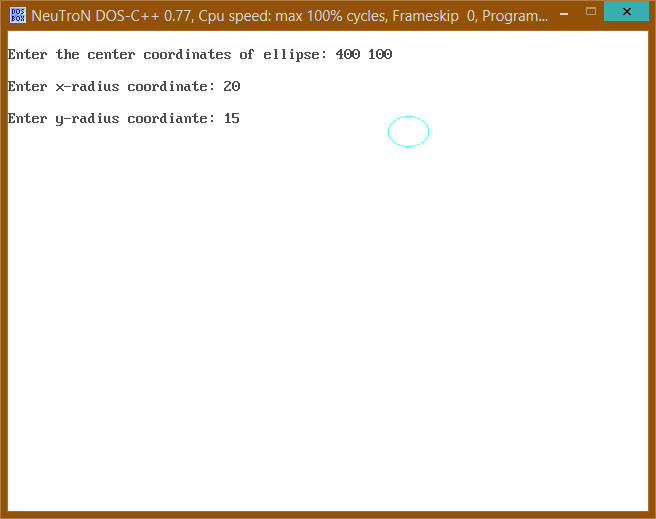
putpixel(xc+x, yc+y, RED);

putpixel(xc-x, yc+y, RED);

putpixel(xc+x, yc-y, RED);

putpixel(xc-x, yc-y, RED);

}



**Q: Polygon creation and drawing program.**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<math.h>

# define MAX 50

int n;

int pol[MAX][2];

void create\_poly();

void draw\_poly();

void main()

{

int gd=DETECT,gm;

initgraph(&gd,&gm,”..\\bgi");

create\_poly();

draw\_poly();

getch();

}

void create\_poly()

{

char ch;

n=0;

do

{

cout<<"\t Enter the "<<n+1<<"th vertex of the polygon:\n";

cin>>pol[n][0]>>pol[n][1];

n++;

cout<<"\t Do you want to enter another vertex: (Y/N)\n";

cin>>ch;

}

while(ch=='y' || ch=='Y');

}

void draw\_poly()

{

int i;

for(i=0;i<=n-2;i++)

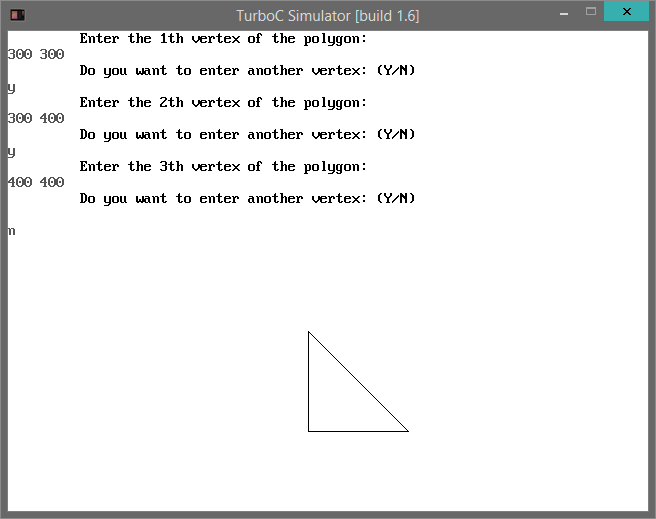
{

line(pol[i][0],pol[i][1], pol[i+1][0],pol[i+1][1]);

}

line(pol[i][0],pol[i][1], pol[0][0],pol[0][1]);

}



**Q: Polygon Translation Program.**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<math.h>

# define MAX 50

int n;

int pol[MAX][2];

voidcreate\_poly();

voiddraw\_poly();

void translate();

void main()

{

intgd=DETECT,gm;

initgraph(&gd,&gm,"..\\bgi");

create\_poly();

draw\_poly();

translate();

draw\_poly();

getch();

}

voidcreate\_poly()

{

charch;

n=0;

do

{

cout<<"\t Enter the "<<n+1<<"th vertex of the polygon:\n";

cin>>pol[n][0]>>pol[n][1];

n++;

cout<<"\t Do you want to enter another vertex: (Y/N)\n";

cin>>ch;

}

while(ch=='y' || ch=='Y');

}

voiddraw\_poly()

{

inti;

for(i=0;i<=n-2;i++)

{

line(pol[i][0],pol[i][1], pol[i+1][0],pol[i+1][1]);

}

line(pol[i][0],pol[i][1], pol[0][0],pol[0][1]);

}

void translate()

{

inti, x,y;

cout<<"\t Enter the cofficient of translation:";

cin>>x>>y;

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

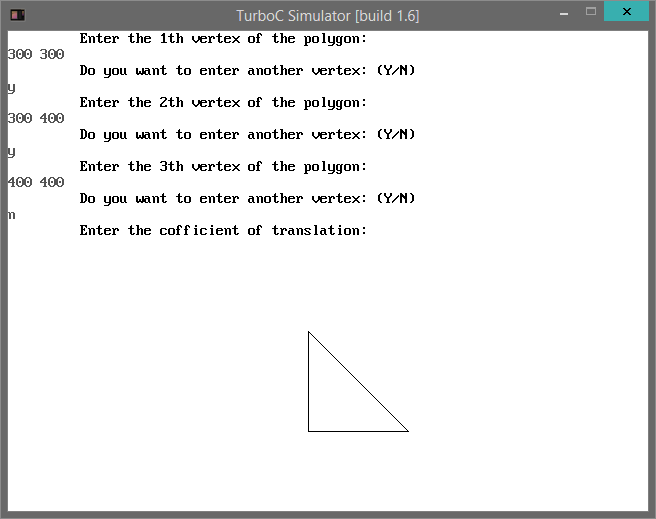
{

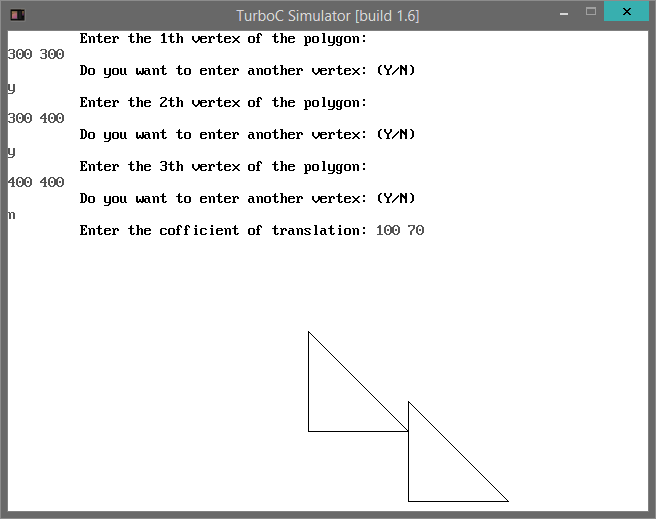
pol[i][0] += x;

pol[i][1] += y;

}

}





**Q: Polygon Rotation Program (About an arbitrary point).**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<math.h>

# define MAX 50

int n;

int pol[MAX][2];

voidcreate\_poly();

voiddraw\_poly();

void rotate();

void main()

{

intgd=DETECT,gm;

initgraph(&gd,&gm,"..\\bgi");

create\_poly();

draw\_poly();

rotate();

draw\_poly();

getch();

}

voidcreate\_poly()

{

charch;

n=0;

do

{

cout<<"\t Enter the "<<n+1<<"th vertex of the polygon:\n";

cin>>pol[n][0]>>pol[n][1];

n++;

cout<<"\t Do you want to enter another vertex: (Y/N)\n";

cin>>ch;

}

while(ch=='y' || ch=='Y');

}

voiddraw\_poly()

{

inti;

for(i=0;i<=n-2;i++)

{

line(pol[i][0],pol[i][1], pol[i+1][0],pol[i+1][1]);

}

line(pol[i][0],pol[i][1], pol[0][0],pol[0][1]);

}

void rotate()

{

inti, x,y, h,k;

float theta;

cout<<"\t Enter the angle of rotation:";

cin>>theta;

cout<<"\t Enter the center of rotation:";

cin>>h>>k;

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

{

x = pol[i][0];

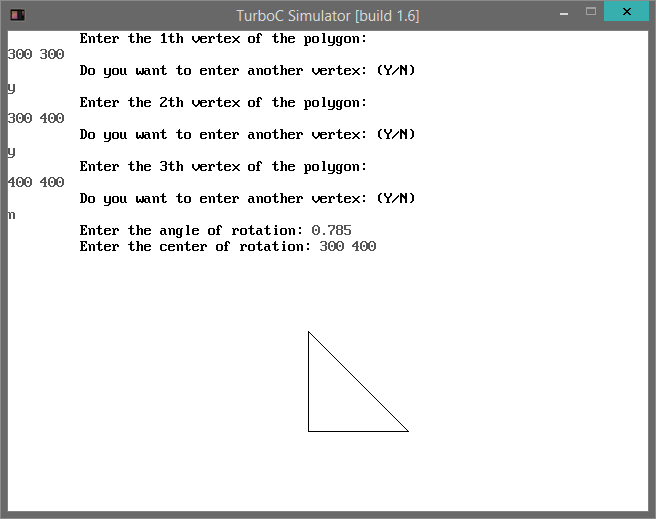
y = pol[i][1];

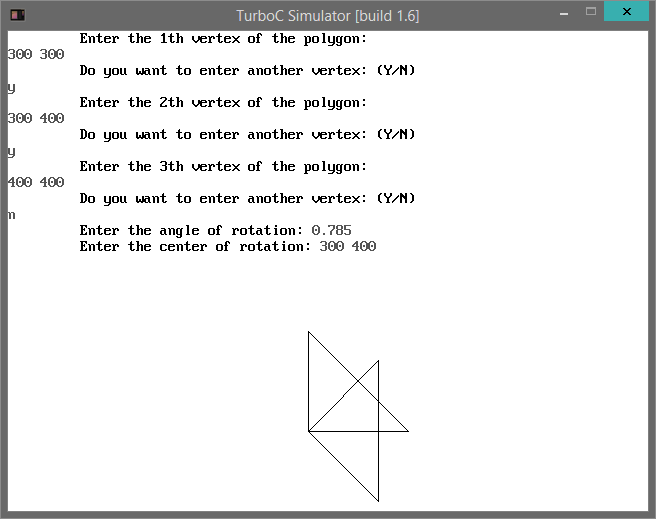
pol[i][0] = (x-h) \* cos(theta) - (y-k) \* sin(theta) + h;

pol[i][1] = (x-h) \* sin(theta) + (y-k) \* cos(theta) + k;

}

}





**Q: Polygon Scaling Program.**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<math.h>

# define MAX 50

int n;

int pol[MAX][2];

voidcreate\_poly();

voiddraw\_poly();

void scale();

void main()

{

intgd=DETECT,gm;

initgraph(&gd,&gm,"..\\bgi");

create\_poly();

draw\_poly();

scale();

draw\_poly();

getch();

}

voidcreate\_poly()

{

charch;

n=0;

do

{

cout<<"\t Enter the "<<n+1<<"th vertex of the polygon:\n";

cin>>pol[n][0]>>pol[n][1];

n++;

cout<<"\t Do you want to enter another vertex: (Y/N)\n";

cin>>ch;

}

while(ch=='y' || ch=='Y');

}

voiddraw\_poly()

{

inti;

for(i=0;i<=n-2;i++)

{

line(pol[i][0],pol[i][1], pol[i+1][0],pol[i+1][1]);

}

line(pol[i][0],pol[i][1], pol[0][0],pol[0][1]);

}

void scale()

{

inti, h,k;

floatsx,sy;

cout<<"\t Enter the scaling factors:\n";

cin>>sx>>sy;

cout<<"\t Enter center of scaling:\n";

cin>>h>>k;

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

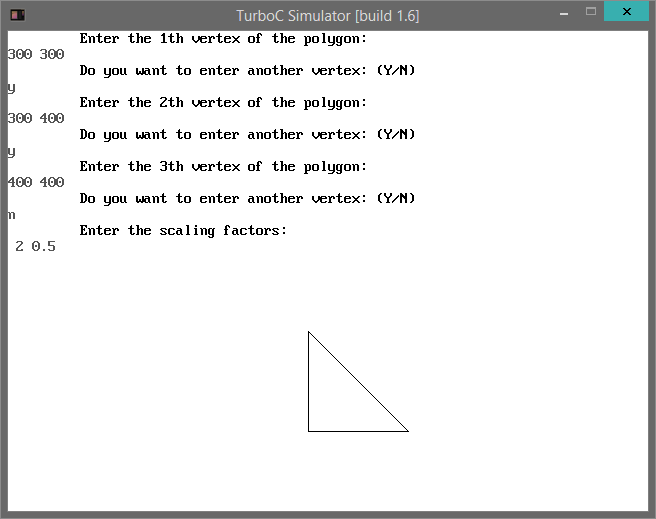
{

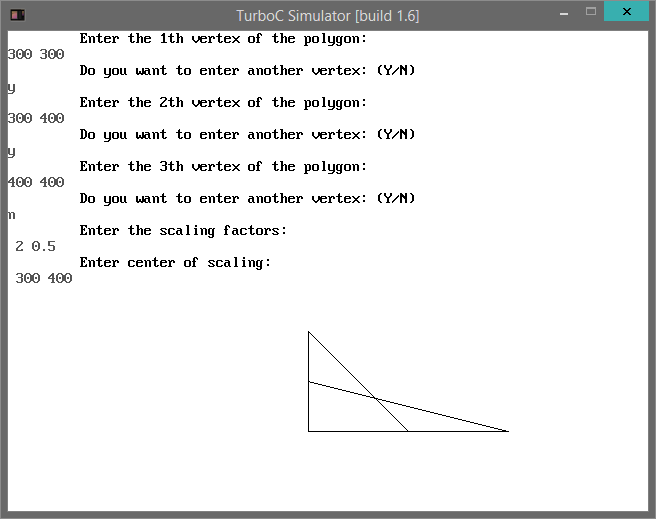
pol[i][0] = (pol[i][0]-h) \* sx + h;

pol[i][1] = (pol[i][1]-k) \* sy + k;

}

}





**Q: Polygon Reflection Program (About an arbitrary line).**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<math.h>

# define MAX 50

int n;

int pol[MAX][2];

voidcreate\_poly();

voiddraw\_poly();

void reflect();

void main()

{

intgd=DETECT,gm;

initgraph(&gd,&gm,"..\\bgi");

create\_poly();

draw\_poly();

reflect();

draw\_poly();

getch();

}

voidcreate\_poly()

{

charch;

n=0;

do

{

cout<<"\t Enter the "<<n+1<<"th vertex of the polygon:\n";

cin>>pol[n][0]>>pol[n][1];

n++;

cout<<"\t Do you want to enter another vertex: (Y/N)\n";

cin>>ch;

}

while(ch=='y' || ch=='Y');

}

voiddraw\_poly()

{

inti;

for(i=0;i<=n-2;i++)

{

line(pol[i][0],pol[i][1], pol[i+1][0],pol[i+1][1]);

}

line(pol[i][0],pol[i][1], pol[0][0],pol[0][1]);

}

void reflect()

{

inti, x,y;

float theta, c;

cout<<"\t Enter the angle line of reflection makes with horizontal:";

cin>>theta;

cout<<"\t Enter the y intercept of line of reflection:";

cin>>c;

line( 0,c, 600, tan(theta)\*600+c );

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

{

x = pol[i][0];

y = pol[i][1]-c;

pol[i][0] = x \* cos(theta) + y \* sin(theta);

pol[i][1] = (-x) \* sin(theta) + y \* cos(theta);

pol[i][1] \*= -1;

x = pol[i][0];

y = pol[i][1];

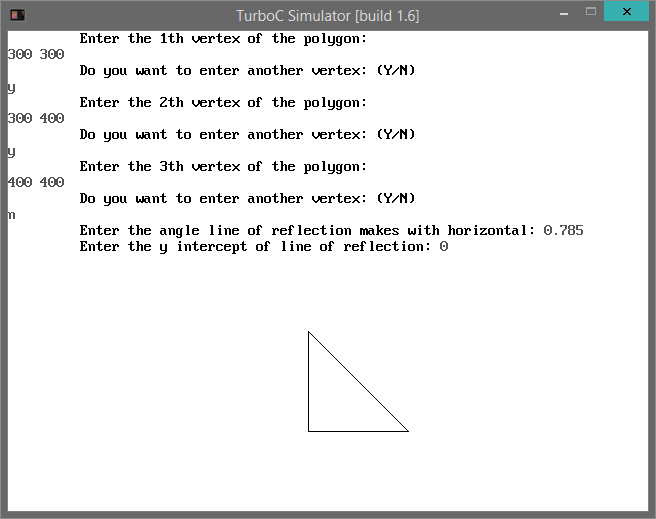
pol[i][0] = x \* cos(theta) - y \* sin(theta);

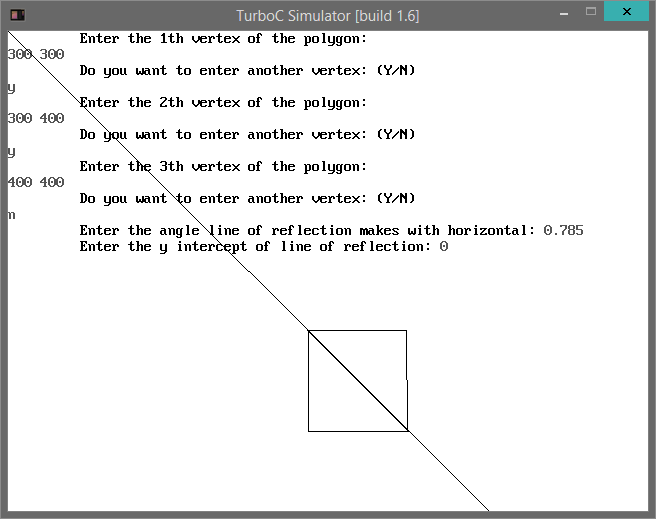
pol[i][1] = x \* sin(theta) + y \* cos(theta);

pol[i][1]+=c;

}

}





**Q: Polygon Shearing along x-axis Program.**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<math.h>

# define MAX 50

int n;

int pol[MAX][2];

voidcreate\_poly();

voiddraw\_poly();

voidsheary();

void main()

{

intgd=DETECT,gm;

initgraph(&gd,&gm,"..\\bgi");

create\_poly();

draw\_poly();

sheary();

draw\_poly();

getch();

}

voidcreate\_poly()

{

charch;

n=0;

do

{

cout<<"\t Enter the "<<n+1<<"th vertex of the polygon:\n";

cin>>pol[n][0]>>pol[n][1];

n++;

cout<<"\t Do you want to enter another vertex: (Y/N)\n\t ";

cin>>ch;

}

while(ch=='y' || ch=='Y');

}

voiddraw\_poly()

{

inti;

for(i=0;i<=n-2;i++)

{

line(pol[i][0],pol[i][1], pol[i+1][0],pol[i+1][1]);

}

line(pol[i][0],pol[i][1], pol[0][0],pol[0][1]);

}

voidsheary()

{

inti;

floatshx, xref;

cout<<"\t Enter the shearing factor along x-axis:";

cin>>shx;

cout<<"\t Enter the y-intercept of the reference line:";

cin>>xref;

// line\_styles(1);

line (0,xref, 600,xref);

// line\_styles(0);

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

{

pol[i][1] -= xref;

pol[i][0] += pol[i][1] \* shx;

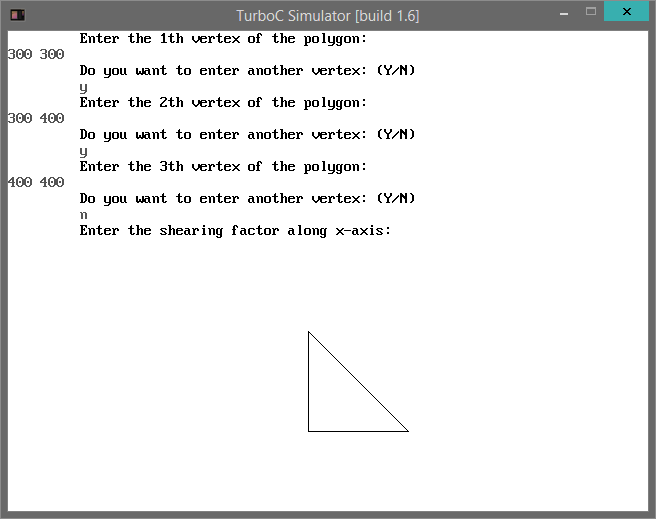
pol[i][1] += xref;

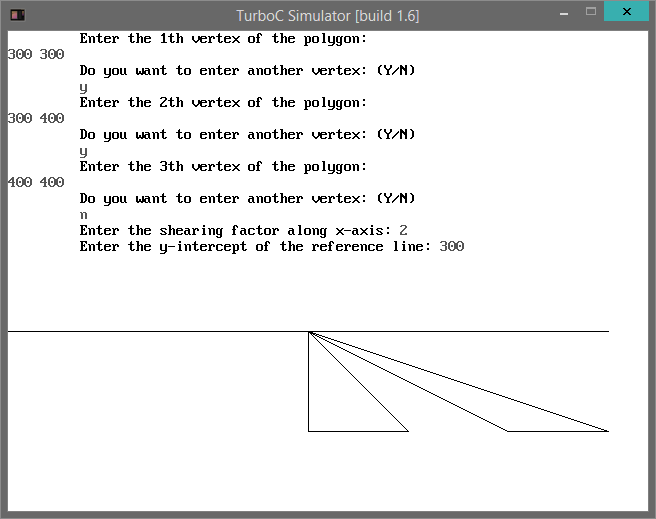
}

}

switch(ch)  
{  
case 1:  
{  
printf(“\nEnter the shearing value\t”);  
scanf(“%d”,&shx);  
for(i=0;i  
{  
ax[i]=ax[i]+(ay[i]\*shx);  
}  
break;  
}  
case 2:  
{  
printf(“\nEnter the shearing value\t”);  
scanf(“%d”,&shx);  
for(i=0;i  
{  
ay[i]=ax[i]\*shx+ay[i];  
}  
break;  
}

}  
display();  
}





**Q: Polygon Shearing along y-axis Program.**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<math.h>

# define MAX 50

int n;

int pol[MAX][2];

voidcreate\_poly();

voiddraw\_poly();

voidsheary();

void main()

{

intgd=DETECT,gm;

initgraph(&gd,&gm,"..\\bgi");

create\_poly();

draw\_poly();

sheary();

draw\_poly();

getch();

}

voidcreate\_poly()

{

charch;

n=0;

do

{

cout<<"\t Enter the "<<n+1<<"th vertex of the polygon:\n";

cin>>pol[n][0]>>pol[n][1];

n++;

cout<<"\t Do you want to enter another vertex: (Y/N)\n\t ";

cin>>ch;

}

while(ch=='y' || ch=='Y');

}

voiddraw\_poly()

{

inti;

for(i=0;i<=n-2;i++)

{

line(pol[i][0],pol[i][1], pol[i+1][0],pol[i+1][1]);

}

line(pol[i][0],pol[i][1], pol[0][0],pol[0][1]);

}

voidsheary()

{

inti;

floatshx, xref;

cout<<"\t Enter the shearing factor along x-axis:";

cin>>shx;

cout<<"\t Enter the y-intercept of the reference line:";

cin>>xref;

// line\_styles(1);

line (0,xref, 600,xref);

// line\_styles(0);

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

{

pol[i][1] -= xref;

pol[i][0] += pol[i][1] \* shx;

pol[i][1] += xref;

}

}

switch(ch)  
{  
case 1:  
{  
printf(“\nEnter the shearing value\t”);  
scanf(“%d”,&shx);  
for(i=0;i  
{  
ax[i]=ax[i]+(ay[i]\*shx);  
}  
break;  
}  
case 2:  
{  
printf(“\nEnter the shearing value\t”);  
scanf(“%d”,&shx);  
for(i=0;i  
{  
ay[i]=ax[i]\*shx+ay[i];  
}  
break;  
}

}  
display();  
}

