**19066570003**

**Q1) Write a program to implement Mid-point circle drawing algorithm.**

#include<iostream.h>

#include<graphics.h>

#include<dos.h>

#include<conio.h>

void circlePoints(int x,int y,int x\_centre,int y\_centre)

{

putpixel(x+x\_centre,y+y\_centre,RED);

delay(100);

putpixel(-x+x\_centre,y+y\_centre,RED);

delay(100);

putpixel(x+x\_centre,-y+y\_centre,RED);

delay(100);

putpixel(-x+x\_centre,-y+y\_centre,RED);

delay(100);

putpixel(y+y\_centre,x+x\_centre,RED);

delay(100);

putpixel(-y+y\_centre,x+x\_centre,RED);

delay(100);

putpixel(y+y\_centre,-x+x\_centre,RED);

delay(100);

putpixel(-y+y\_centre,-x+x\_centre,RED);

delay(100);

}

void midPointCircle(int r,int x\_centre,int y\_centre)

{

int x=0;

int y=r;

double d=3-2\*r;

while(y>x)

{

if(d<0)

d=d+4\*x+6.0;

else

{

d=d+4\*(x-y)+10.0;

y=y-1;

}

x++;

circlePoints(x,y,x\_centre,y\_centre);

}

return;

}

void main()

{

int i,gd=DETECT,gm;

int x,y,r;

initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");

cout<<"Enter the centre coordinates:\n";

cout<<"x: ";

cin>>x;

cout<<"y: ";

cin>>y;

cout<<"Enter radius: ";

cin>>r;

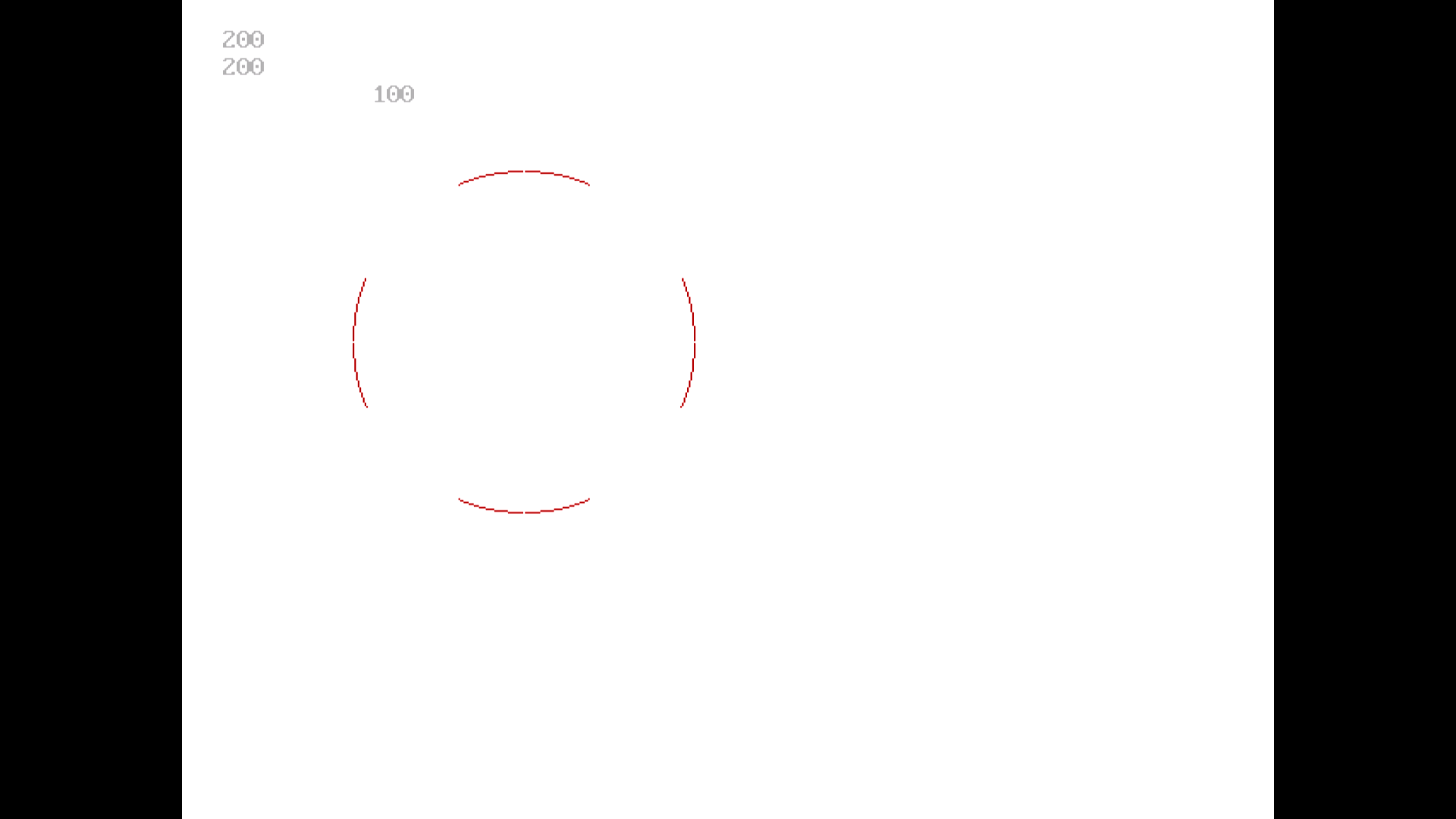
setbkcolor(WHITE);

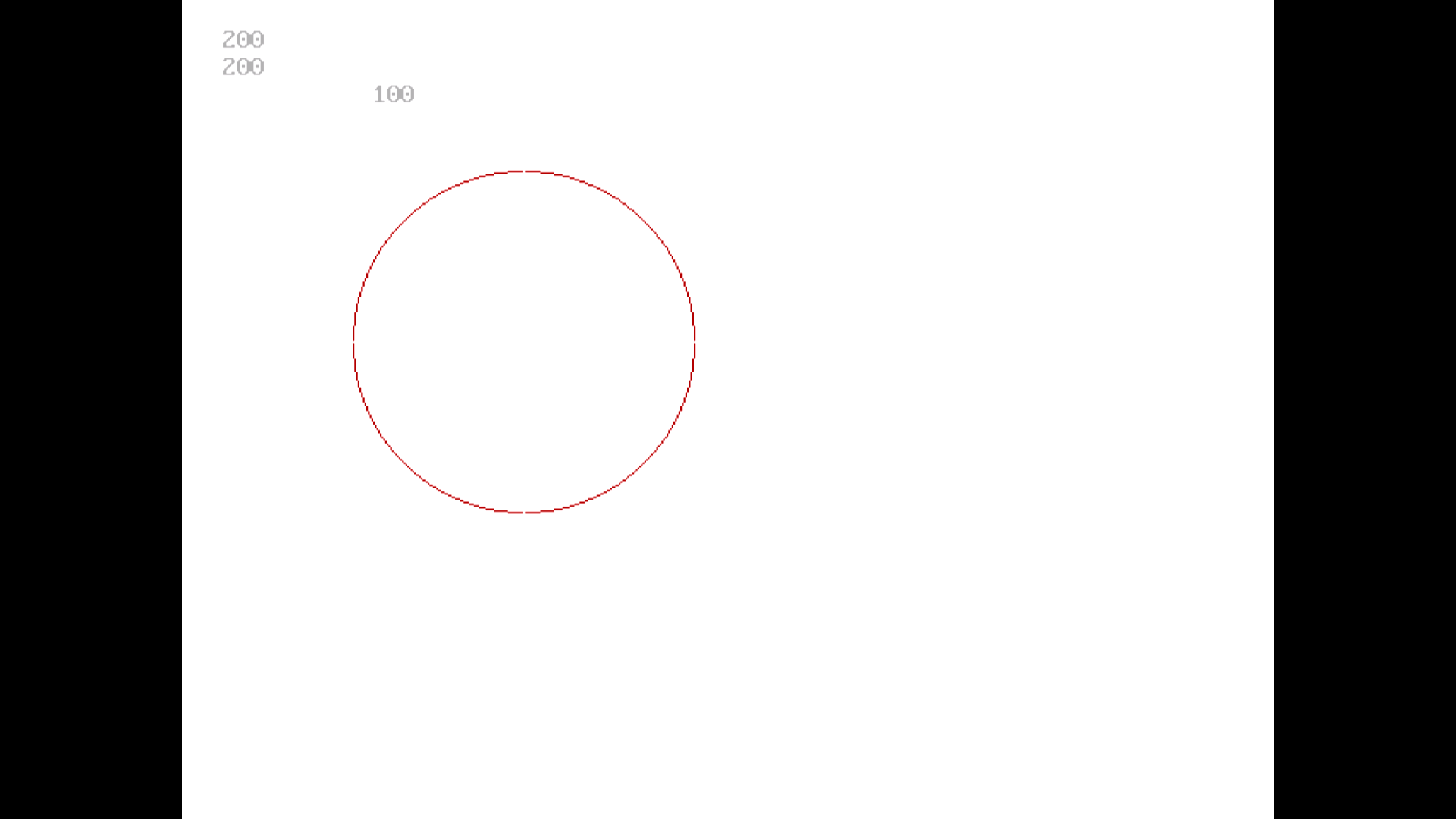
midPointCircle(r,x,y);

getch();

}







**Q2)Write a program to implement 3D axonometric projection**

|  |
| --- |
| #include<iostream.h> |
|  | #include<graphics.h> |
|  | #include<conio.h> |
|  | #include<dos.h> |
|  | #include<math.h> |
|  |  |
|  | #define pi 3.14285714 |
|  |  |
|  | class projections |
|  | { |
|  | double vertices[8][4]; //matrix contains vertices of the triangle |
|  | double t\_matrix[4][4]; //transformation matrix |
|  | double result[8][4]; |
|  |  |
|  | public: |
|  | projections(){}; |
|  |  |
|  | void get\_vertices(); |
|  | void display\_cube(); |
|  |  |
|  | void multiplication(); |
|  | void copyback(); |
|  |  |
|  |  |
|  | void axonometric(double angle\_x,double angle\_y); |
|  |  |
|  | }; |
|  |  |
|  |  |
|  | void projections::get\_vertices() |
|  | { |
|  | for(int i=0;i<8;i++) |
|  | { |
|  | cout<<"\nEnter vertex "<<i+1<<"..."; |
|  | cout<<"\nx : "; |
|  | cin>>vertices[i][0]; |
|  | result[i][0]=vertices[i][0]; |
|  |  |
|  | cout<<"y : "; |
|  | cin>>vertices[i][1]; |
|  | result[i][1]=vertices[i][1]; |
|  |  |
|  | cout<<"z : "; |
|  | cin>>vertices[i][2]; |
|  | result[i][2]=vertices[i][2]; |
|  |  |
|  | vertices[i][3]=result[i][3]=1; |
|  | } |
|  | } |
|  |  |
|  |  |
|  | void projections::display\_cube() |
|  | { |
|  | int i=0; |
|  |  |
|  | for(i=0;i<3;i++) |
|  | line(result[i][0],result[i][1],result[i+1][0],result[i+1][1]); |
|  | line(result[i][0],result[i][1],result[0][0],result[0][1]); |
|  |  |
|  | for(i=4;i<7;i++) |
|  | line(result[i][0],result[i][1],result[i+1][0],result[i+1][1]); |
|  | line(result[i][0],result[i][1],result[4][0],result[4][1]); |
|  |  |
|  | for(i=0;i<4;i++) |
|  | line(result[i][0],result[i][1],result[i+4][0],result[i+4][1]); |
|  | } |
|  |  |
|  |  |
|  | void projections::copyback() |
|  | { |
|  | int i=0,j=0; |
|  |  |
|  | for(i=0;i<8;i++) |
|  | for(j=0;j<4;j++) |
|  | result[i][j]=vertices[i][j]; |
|  | } |
|  |  |
|  |  |
|  | void projections::multiplication() |
|  | { |
|  | double r[8][4]; |
|  | int i=0,j=0,k=0; |
|  |  |
|  | for(i=0;i<8;i++) |
|  | { |
|  | for(j=0;j<4;j++) |
|  | { |
|  | r[i][j]=0; |
|  | for(k=0;k<4;k++) |
|  | r[i][j]+=result[i][k]\*t\_matrix[k][j]; |
|  | } |
|  | } |
|  |  |
|  | for(i=0;i<8;i++) //Copying back the result |
|  | for(j=0;j<4;j++) |
|  | result[i][j]=r[i][j]; |
|  | } |
|  |  |
|  |  |
|  | //AXONOMETRIC PROJECTION.................................................................. |
|  | void projections::axonometric(double angle\_x,double angle\_y) |
|  | { |
|  | angle\_x=((pi/180)\*angle\_x); //tan inverse(slope)=angle |
|  | angle\_y=((pi/180)\*angle\_y); |
|  |  |
|  | cleardevice(); |
|  | clearviewport(); |
|  | cout<<"\tAXONOMETRIC PROJECTION..........."; |
|  |  |
|  | copyback(); |
|  |  |
|  | t\_matrix[0][0]=cos(angle\_y); //[T] for parallel projection on plane z=0 |
|  | t\_matrix[0][1]=(sin(angle\_y))\*(sin(angle\_x)); |
|  | t\_matrix[0][2]=0; |
|  | t\_matrix[0][3]=0; |
|  |  |
|  | t\_matrix[1][0]=0; |
|  | t\_matrix[1][1]=cos(angle\_x); |
|  | t\_matrix[1][2]=0; |
|  | t\_matrix[1][3]=0; |
|  |  |
|  | t\_matrix[2][0]=sin(angle\_y); |
|  | t\_matrix[2][1]=(-1)\*(cos(angle\_y)\*sin(angle\_x)); |
|  | t\_matrix[2][2]=0; |
|  | t\_matrix[2][3]=0; |
|  |  |
|  | t\_matrix[3][0]=0; |
|  | t\_matrix[3][1]=0; |
|  | t\_matrix[3][2]=0; |
|  | t\_matrix[3][3]=1; |
|  |  |
|  | multiplication(); |
|  |  |
|  | setcolor(GREEN); |
|  | display\_cube(); |
|  | delay(20); |
|  | getch(); |
|  | } |
|  |  |
|  |  |
|  |  |
|  | void main() |
|  | { |
|  | clrscr(); |
|  | int gd=DETECT,gm; |
|  | projections t1; |
|  |  |
|  | double angle\_x,angle\_y,angle,ratio,ratio1,ratio2,ratio3; |
|  |  |
|  | cout<<"\n\n\t ........PROJECTIONS OF 3D OBJECTS........\n"; |
|  | cout<<"\nEnter the details of a cube(i.e. 3D object)....."; |
|  | t1.get\_vertices(); |
|  |  |
|  | initgraph(&gd,&gm,"C:\\Turboc3\\BGI"); |
|  |  |
|  |  |
|  | cout<<"\n\nFOR AXONOMETRIC PROJECTION.........."; |
|  | cout<<"\nEnter the angle of rotation about :-"; |
|  | cout<<"\nx-axis : "; |
|  | cin>>angle\_x; |
|  | cout<<"And, y-axis : "; |
|  | cin>>angle\_y; |
|  |  |
|  | t1.axonometric(angle\_x,angle\_y); |
|  |  |
|  |  |
|  |  |
|  | closegraph(); |
|  |  |
|  |  |
|  | } |

Coordinates to enter

100,200,0

200,200,0

200,100,0

100,100,0

100,200,100

200,200,100

200,100,100

100,100,100

