```
/*WAP to implement Mid-Point Ellipse algorithm in Cpp.*/
#include <iostream>//ellipse(x, y, start_angle, end_angle, x_radius, y_radius)
#include <cmath>
#include <graphics.h>
using namespace std;
int x_c,y_c;
float x_n,y_n,p_1,p_2,r_x,r_y;
void draw_ellipse_region_1()
{
   if (p_1<0) //if p_1<0 , x_n = x_n + 1,  p_1=p_1+2*r_y^2*x_n+r_y^2;
   {
      x_n++; // change x_n only
      p_1=p_1+2*r_y*r_y*x_n+r_y*r_y;
   }
   else //if 0<=p_1 , x_n = x_n + 1,  y_n = y_n - 1,  p_1=p_1+2*r_y^2*x_n-
2*r_x^2*y_n+r_y^2
{
      x_n++;  // change both x_n and y_n
      y_n--;
      p_1=p_1+2*r_y*r_y*x_n-2*r_x*r_x*y_n+r_y*r_y;
   }
   putpixel(x_c+x_n,y_c+y_n,GREEN); //1st quadrant
   putpixel(x_c-x_n,y_c+y_n,GREEN); //2 nd quadrant
   putpixel(x_c-x_n,y_c-y_n,GREEN); //3 rd quadrant
   putpixel(x_c+x_n,y_c-y_n,GREEN); //4 th quadrant
}
void draw_ellipse_region_2()
{
   if (0<p_2) // if 0<p_2 , y_n=y_n-1, p_2=p_2-2*r_x^2*y_n+r_x^2
   {
      y_n--; // change y_n only
      p_2=p_2-2*r_x*r_x*y_n+r_x*r_x;
   }
   else // if p_2<=0 , y_n=y_n-1, x_n = x_n+1,  p_2=p_2+2*r_y^2*x_n-
2*r_x^2*y_n+r_x^2
   {
      x_n++; // change both x_n & y_n
```

```
      y_n--;
      p_2=p_2+2*r_y*r_y*x_n-2*r_x*r_x*y_n+r_x*r_x;
   }
   putpixel(x_c+x_n,y_c+y_n,GREEN); //1st quadrant
   putpixel(x_c-x_n,y_c+y_n,GREEN); //2 nd quadrant
   putpixel(x_c-x_n,y_c-y_n,GREEN); //3 rd quadrant
   putpixel(x_c+x_n,y_c-y_n,GREEN); //4 th quadrant
}
int main()
{
   int i;
   while(1)
   {
      cout<<"\n\n\n\t\t\t\t\t1366*768 ";
      cout<<"\n\n\n\t\t Enter ellipse coordinates (x,y,r_x,r_y) with in range (0,0)
to (1365,767)";
      cout<<"\n\n Enter (x_c,y_c)";
      cout<<"\n Enter x_c: ";
      cin>>x_c;
      cout<<" Enter y_c: ";
      cin>>y_c;
      cout<<"\n\n Enter r_x: ";
      cin>>r_x;
      cout<<" Enter r_y: ";
      cin>>r_y;
      x_n=0;
      y_n=r_y;
      p_1=r_y*r_y-r_x*r_x*r_y+r_x*r_x/4; //  p_1=r_y^2-r_x^2*r_y+r_x^2/4
      initwindow(1366,768);
      for(i=0; i<=1365; i++) // creates white background
      {
         line(0,i,1365,i);
      }
      //setcolor(GREEN);
      //ellipse(x_c+50,y_c+50,0,360,r_x,r_y);
      while ((r_y*r_y*x_n)<=(r_x*r_x*y_n)) //2*r_y^2*x_n<=2*r_x^2*y_n
      {
```

```
        draw_ellipse_region_1();
      }
      p_2=r_y*r_y*(x_n+0.5)*(x_n+0.5)+r_x*r_x*(y_n-1)*(y_n-1)-
r_x*r_x*r_y*r_y; //p_2=r_y^2*(x_n+0.5)^2+r_x^2*(y_n-1)^2-r_x^2*r_y^2;
      while (0<y_n)
      {
        draw_ellipse_region_2();
      }
      putpixel(x_c,y_c,GREEN); //At center of ellipse
      putpixel(x_c,y_c+r_y,GREEN); //At topmost point
      putpixel(x_c,y_c-r_y,GREEN); //At bottom point
      getch();
      closegraph();
   }
   return 0;
}
```

```
/*WAP to implement Mid-Point Ellipse algorithm in Cpp.*/
//MID_POINT_Ellipse
//Using GLUT
#include<GL/gl.h>
#include<GL/glu.h>
#include<GL/glut.h>

//#include <bits/stdc++.h>
#include<iostream>

//for animation purpose
#include<vector>


using namespace std;
```

```
void display();   //display function
void reshape(int,int);   //reshape the viewport
void timer(int);    //for displaying no of frames in a sec

void getinfo(); //info from user
void drawEllipse();  // drawing circle
float xc,yc,a,b,p;


void drawEllipseAnimation();  //animation
void keyboard(unsigned char,int,int); //for animation keyboard input
float ax,ay,aa,ab,ap; //for animation points
bool startAnimation=false;//for animation start
vector<float> point;//for animation
bool once=false;


void init(){

   glClearColor(0.1,0.1,0.1,1.0);  //background color
}


int main(int argc, char** argv){
   getinfo();
   glutInit(&argc,argv);
   glutInitDisplayMode(GLUT_RGB|GLUT_DOUBLE);

   glutInitWindowSize(500,500);
   glutInitWindowPosition(200,200);

   glutCreateWindow("Mid-Point-Ellipse");
   glutDisplayFunc(display);
   glutReshapeFunc(reshape);
```

```
   glutSetKeyRepeat(GLUT_KEY_REPEAT_OFF);
   glutKeyboardFunc(keyboard);
   glutTimerFunc(0,timer,0);
   init();
   glutMainLoop();
   return 0;


}



void display(){


   glClear(GL_COLOR_BUFFER_BIT);
   glLoadIdentity();


   glColor3f(.7,.7,.7);//axis line color
   glBegin(GL_LINES);
   glVertex2f(250,0);
   glVertex2f(-250,0);
   glVertex2f(0,250);
   glVertex2f(0,-250);
   glEnd();
   glPointSize(3);
   glBegin(GL_POINTS);
   glVertex2f(xc,yc);
   glEnd();
   glPointSize(1);
   drawEllipse();
   drawEllipseAnimation();



   glutSwapBuffers();
```

```
}


void reshape(int w,int h){
   glViewport(0,0,w,h);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity;
   gluOrtho2D(-250,250,-250,250);
   glMatrixMode(GL_MODELVIEW);
}

void timer(int){
   glutPostRedisplay();
   glutTimerFunc(1000/30,timer,0);

}



void getinfo(){
   cout<<endl<<endl<<"\t Enter the following:"<<endl;
   cout<<"\t Center x: ";
   cin>>xc;
   cout<<"\t Center y: ";
   cin>>yc;
   cout<<"\t a: ";
   cin>>a;
   cout<<"\t b: ";
   cin>>b;


   //for animation

   ax=0;
   ay=b;
```

```
   aa=a;
   ab=b;
   ap=b*b-a*a*b+(a*a)/4;
}


void drawEllipse(){
   float x,y;
   p=b*b-a*a*b+(a*a)/4;
   x=0;
   y=b;
   glColor3f(1,1,1);//ellipse color
   glBegin(GL_POINTS);
   while(2*b*b*x < 2*a*a*y){
      glVertex2f(xc+x,yc+y);
      glVertex2f(xc+x,yc-y);
      glVertex2f(xc-x,yc+y);
      glVertex2f(xc-x,yc-y);
      x=x+1;
      if(p<0){
         p=p+2*b*b*x+b*b;
      }
      else{
         y=y-1;
         p=p+2*b*b*x+b*b - 2*a*a*y;
      }
   }
   p=b*b*(x+.5)*(x+.5) + a*a*(y-1)*(y-1)-a*a*b*b;
   while(y>=0){
      glVertex2f(xc+x,yc+y);
      glVertex2f(xc+x,yc-y);
      glVertex2f(xc-x,yc+y);
      glVertex2f(xc-x,yc-y);
      y=y-1;
      if(p>0){
         p=p-2*a*a*y+a*a;
      }
```

```
      else{
         x=x+1;
         p=p+2*b*b*x+a*a - 2*a*a*y;
      }
   }
   glEnd();

}



//For animation below here

void drawEllipseAnimation(){

   if( 2*ab*ab*ax < 2*aa*aa*ay && startAnimation==true){

      point.push_back(xc+ax);
      point.push_back(yc+ay);
      point.push_back(xc+ax);
      point.push_back(yc-ay);
      point.push_back(xc-ax);
      point.push_back(yc+ay);
      point.push_back(xc-ax);
      point.push_back(yc-ay);

      ax=ax+1;
      if(ap<0){
         ap=ap+2*ab*ab*ax+ab*ab;
      }
      else{
         ay=ay-1;
         ap=ap+2*ab*ab*ax+ab*ab - 2*aa*aa*ay;
      }
   }
```

```
if(ay>=0 && 2*ab*ab*ax > 2*aa*aa*ay  && startAnimation==true ){
   if(!once){
      ap=ab*ab*(ax+.5)*(ax+.5) + aa*aa*(ay-1)*(ay-1)-aa*aa*ab*ab;
      once=true;
   }

   point.push_back(xc+ax);
   point.push_back(yc+ay);
   point.push_back(xc+ax);
   point.push_back(yc-ay);
   point.push_back(xc-ax);
   point.push_back(yc+ay);
   point.push_back(xc-ax);
   point.push_back(yc-ay);

   ay=ay-1;
   if(ap>0){
      ap=ap-2*aa*aa*ay+aa*aa;
   }
   else{
      ax=ax+1;
      ap=ap+2*ab*ab*ax+aa*aa - 2*aa*aa*ay;
   }

}


if(ay>=0 || 2*ab*ab*ax < 2*aa*aa*ay)
   glColor3f(1,0,0);
else
   glColor3f(1,1,1);
glPointSize(1);
glBegin(GL_POINTS);
for(int i=0;i<point.size();i+=2){
   glVertex2f(point.at(i),point.at(i+1));
}
glEnd();
```

```
   glPointSize(1);
}


void keyboard(unsigned char key,int x,int y){

   if(key=='p')
      startAnimation=true;

   if(key=='o')
      startAnimation=false;

}
```