```
// code for Bresenham circle

#include <GL/freeglut.h>
#include <GL/gl.h>
#include <iostream>

// Function to set a pixel using coordinates
void setPixel( int x, int y ) {
    glBegin( GL_POINTS );
        glVertex2f(x, y);
    glEnd();
    glFlush();
}

// Function to execute Bresenham's Circle Algo
void bresenhamCircle( int xc, int yc, int r ) {
    glutInitDisplayMode(GLUT_SINGLE);
    glutInitWindowSize(1000,1000);
    glutInitWindowPosition(100,100);
    glutCreateWindow("Bresengam Circle");
    glClearColor(0.0, 0.0, 0.0, 0.0);
            glMatrixMode(GL_PROJECTION);
            glLoadIdentity();
            gluOrtho2D(0.0, 1000.0, 0.0, 1000.0);

    // int xc = 500, yc = 500, r = 200;
    // Calculating initial decision parameter
    int d = 3 - 2 * r;

    int x = 0, y = r;

    while( x <= y ) {
        // Plotting points of circle in all 8 octants
        setPixel( xc + x, yc + y );
        setPixel( xc + y, yc + x );
        setPixel( xc + y, yc - x );
        setPixel( xc + x, yc - y );
        setPixel( xc - x, yc - y );
        setPixel( xc - y, yc - x );
        setPixel( xc - y, yc + x );
        setPixel( xc - x, yc + y );

        // Updating value of decision parameter
        if( d < 0 ) {
            d += 4 * x + 6;
        } else if( d > 0 ) {
            d += 4 * (x - y) + 10;
            y--;
        }
        x++;
    }
}
```

```
int main(int argc, char** argv) {
    glutInit( &argc, argv );
    bresenhamCircle( 500, 500, 100 );
    glutMainLoop();
    return 0;
}


// code for DDA and bresenham line


#include <iostream>
#include <GL/gl.h>
#include <GL/freeglut.h>

using namespace std;

#define sign(x) ((x > 0) ? 1 : ((x < 0) ? -1 : 0))

void setPixel(GLint x, GLint y)
{
  glBegin(GL_POINTS);
  glVertex2i(x, y);
  glEnd();
}

void dda(float x1, float y1, float x2, float y2, int line_type)
{
  float dx, dy, steps, x_in, y_in;

  dx = abs(x2 - x1);
  dy = abs(y2 - y1);

  if (dx >= dy)
    steps = abs(dx);
  else
    steps = abs(dy);

  x_in = abs(dx) / steps;
  y_in = abs(dy) / steps;

  float x_new = x1;
  float y_new = y1;

  setPixel(x_new, y_new);
  for (int i = 0; i < steps; i++)
  {
    x_new += x_in;
    y_new += y_in;
    // setPixel(x_new, y_new);
    switch (line_type)
    {
    case 1:
      glPointSize(1);
      setPixel(x_new, y_new);
      break;
    case 2:
      glPointSize(3);
      setPixel(x_new, y_new);
      break;
    case 3:
      if ((int)x_new % 4 == 0)
      {
        glPointSize(3);
        setPixel(x_new, y_new);
      }
    }
  }
  glFlush();
}

void bresenham(float x1, float y1, float x2, float y2, int line_type)
{
  float dx, dy, x, y, d, s1, s2, swap = 0, temp;
  dx = abs(x2 - x1);
  dy = abs(y2 - y1);
  s1 = sign(x2 - x1);
  s2 = sign(y2 - y1);
  if (dy > dx)
  {
    temp = dx;
    dx = dy;
    dy = temp;
    swap = 1;
  }

  d = 2 * dy - dx;
  x = x1;
  y = y1;
  setPixel(x, y);

  int i;
  for (i = 1; i <= dx; i++)
  {
    while (d >= 0)
    {
      if (swap)
        x = x + s1;
      else
      {
        y = y + s2;
        d = d - 2 * dx;
      }
    }
    if (swap)
      y = y + s2;
    else
      x = x + s1;
    d = d + 2 * dy;

    switch (line_type)
    {
    case 1:
      glPointSize(1);
      setPixel(x, y);
      break;
    case 2:
      glPointSize(3);
      setPixel(x, y);
      break;
    case 3:
      if ((int)x % 4 == 0)
      {
        glPointSize(3);
        setPixel(x, y);
      }
    }
  }
  glFlush();
}

void menu()
{
  bool loop = 1;
  while (loop)
  {
    cout << "\nDraw a line using :\n1)DDA\t\t2)Bresenham\t\t3)exit\n";
    int choice, line_type;
    cin >> choice;
    float x1 = 0, x2 = 0, y1 = 0, y2 = 0;
    if (choice != 3)
    {
      cout << "\nWhich line do you want to display:\n";
      cout << "1)Normal Line\t2)Bold line\t3)Dotted line\n";
      cin >> line_type;

      cout << "\nEnter Coordinates of lines..\n";
      cout << "x1 = ";
      cin >> x1;
      cout << "y1 = ";
      cin >> y1;
      cout << "x2 = ";
      cin >> x2;
      cout << "y2 = ";
      cin >> y2;
      cout << endl;

      x1 += 250;
      y1 += 250;
      x2 += 250;
      y2 += 250;
    }

    switch (choice)
    {
    case 1:
      dda(x1, y1, x2, y2, line_type);
      break;
    case 2:
      bresenham(x1, y1, x2, y2, line_type);
      break;
    case 3:
      loop = 0;
      break;
    default:
      cout << "----------Enter correct choice------\n";
    }
  }

  glFlush();
}

void draw(){
  glClear(GL_COLOR_BUFFER_BIT);

  bresenham(0, 250, 500, 250, 1);
  bresenham(250, 0, 250, 500, 1);

  menu();

  glFlush();
}

void init()
{
  glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
  glutInitWindowPosition(0, 0);
  glutInitWindowSize(500, 500);
  glutCreateWindow("Green Window");
  glClearColor(1.0, 1.0, 1.0, 1.0);
  glColor3f(0.0, 0.0, 0.0);
  gluOrtho2D(0, 500, 0, 500);
}

int main(int argc, char **argv)
{
  glutInit(&argc, argv);
  init();

  glutDisplayFunc(draw);
  glutMainLoop();
  return 0;
}
```