B. Star :

```
#include <windows.h>

#include <GL/glut.h>

#include <math.h>

void bresenhamsLine(int x1, int y1, int x2, int y2) {

    if (x1 > x2) {

        int tempX = x1;

        int tempY = y1;

        x1 = x2;

        y1 = y2;

        x2 = tempX;

        y2 = tempY;

    }

    float m = (x2 - x1) == 0 ? 999 : (float)(y2 - y1) / (float)(x2 - x1);

    int pk, x, y;

    if (m >= 0 && m <= 1) {

        int dx = x2 - x1;

        int dy = y2 - y1;

        pk = (2 * dy) - dx;

        x = x1;

        y = y1;

        glBegin(GL_POINTS);

        glColor3f(1.0f, 0.0f, 0.0f);

        for (int i = 0; i < dx; i++) {

            glVertex2f(x, y);

            if (pk < 0)

                pk = pk + 2 * dy;

            else {

                pk = pk + 2 * dy - 2 * dx;

                y = y + 1;

            }
        x = x + 1;
    }
    glEnd();
}
else if (m > 1) {
    int dx = x2 - x1;
    int dy = y2 - y1;
    pk = (2 * dx) - dy;
    x = x1;
    y = y1;
    glBegin(GL_POINTS);
    glColor3f(1.0f, 0.0f, 0.0f);
    for (int i = 0; i < dy; i++) {
        glVertex2f(x, y);
        if (pk < 0)
            pk = pk + 2 * dx;
        else {
            pk = pk + 2 * dx - 2 * dy;
            x = x + 1;
        }
        y = y + 1;
    }
    glEnd();
}
else if (m < 0 && m >= -1) {
    int dx = x2 - x1;
    int dy = y1 - y2;
    pk = (2 * dy) - dx;
    x = x1;
    y = y1;
    glBegin(GL_POINTS);
```

```
        glColor3f(1.0f, 0.0f, 0.0f);
        for (int i = 0; i < dx; i++) {
            glVertex2f(x, y);
            if (pk < 0)
                pk = pk + 2 * dy;
            else {
                pk = pk + 2 * dy - 2 * dx;
                y = y - 1;
            }
            x = x + 1;
        }
        glEnd();
    }
    else if (m < -1) {
        int dx = x2 - x1;
        int dy = y1 - y2;
        pk = (2 * dx) - dy;
        x = x1;
        y = y1;
        glBegin(GL_POINTS);
        glColor3f(1.0f, 0.0f, 0.0f);
        for (int i = 0; i < dy; i++) {
            glVertex2f(x, y);
            if (pk < 0)
                pk = pk + 2 * dx;
            else {
                pk = pk + 2 * dx - 2 * dy;
                x = x + 1;
            }
            y = y - 1;
        }
        glEnd();
    }
}
void display() {
    glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
    glClear(GL_COLOR_BUFFER_BIT);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-500.0, 500.0, -500.0, 500.0, -500.0, 500.0);
    glMatrixMode(GL_MODELVIEW);
    // --- Fill Star using 8 GL_POLYGON triangles ---
    glColor3f(0.0f, 1.0f, 0.0f); // Green color for fill
    glBegin(GL_POLYGON);
    glVertex2f(0, 0);
    glVertex2f(0, 300);
    glVertex2f(100, 100);
    glEnd();
    glBegin(GL_POLYGON);
    glVertex2f(0, 0);
    glVertex2f(100, 100);
    glVertex2f(300, 0);
    glEnd();
    glBegin(GL_POLYGON);
    glVertex2f(0, 0);
    glVertex2f(300, 0);
    glVertex2f(100, -100);
    glEnd();
    glBegin(GL_POLYGON);
    glVertex2f(0, 0);
    glVertex2f(100, -100);
    glVertex2f(0, -300);
```

```
glEnd();

glBegin(GL_POLYGON);

glVertex2f(0, 0);

glVertex2f(0, -300);

glVertex2f(-100, -100);

glEnd();

glBegin(GL_POLYGON);

glVertex2f(0, 0);

glVertex2f(-100, -100);

glVertex2f(-300, 0);

glEnd();

glBegin(GL_POLYGON);

glVertex2f(0, 0);

glVertex2f(-300, 0);

glVertex2f(-100, 100);

glEnd();

glBegin(GL_POLYGON);

glVertex2f(0, 0);

glVertex2f(-100, 100);

glVertex2f(0, 300);

glEnd();

// --- Draw Bresenham's Outline ---

bresenhamsLine(0, 300, 100, 100);

bresenhamsLine(100, 100, 300, 0);

bresenhamsLine(300, 0, 100, -100);

bresenhamsLine(100, -100, 0, -300);

bresenhamsLine(0, -300, -100, -100);

bresenhamsLine(-100, -100, -300, 0);

bresenhamsLine(-300, 0, -100, 100);

bresenhamsLine(-100, 100, 0, 300);
```

```
    glFlush();
}
int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(50, 50);
    glutCreateWindow("Star - Bresenham Outline");
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}
```

Squre :

```
#include <windows.h>
#include <GL/glut.h>
#include <math.h>
void bresenhamsLine(int x1, int y1, int x2, int y2) {
    if (x1 > x2) {
        int tempX = x1, tempY = y1;
        x1 = x2; y1 = y2;
        x2 = tempX; y2 = tempY;
    }
    float m = (x2 - x1) == 0 ? 999 : (float)(y2 - y1) / (float)(x2 - x1);
    int pk, x, y;
    if (m >= 0 && m <= 1) {
        int dx = x2 - x1, dy = y2 - y1;
        pk = 2 * dy - dx;
        x = x1; y = y1;
        glBegin(GL_POINTS);
        glColor3f(1, 0, 0);
        for (int i = 0; i < dx; i++) {
            glVertex2f(x, y);
            if (pk < 0) pk += 2 * dy;
            else { pk += 2 * dy - 2 * dx; y++; }
            x++;
        }
        glEnd();
    } else if (m > 1) {
        int dx = x2 - x1, dy = y2 - y1;
        pk = 2 * dx - dy;
        x = x1; y = y1;
        glBegin(GL_POINTS);
        glColor3f(1, 0, 0);
        for (int i = 0; i < dy; i++) {
            glVertex2f(x, y);
            if (pk < 0) pk += 2 * dx;
            else { pk += 2 * dx - 2 * dy; x++; }
            y++;
        }
        glEnd();
    } else if (m < 0 && m >= -1) {
        int dx = x2 - x1, dy = y1 - y2;
        pk = 2 * dy - dx;
        x = x1; y = y1;
        glBegin(GL_POINTS);
        glColor3f(1, 0, 0);
        for (int i = 0; i < dx; i++) {
            glVertex2f(x, y);
            if (pk < 0) pk += 2 * dy;
            else { pk += 2 * dy - 2 * dx; y--; }
            x++;
        }
        glEnd();
    } else if (m < -1) {
        int dx = x2 - x1, dy = y1 - y2;
        pk = 2 * dx - dy;
        x = x1; y = y1;
        glBegin(GL_POINTS);
        glColor3f(1, 0, 0);
        for (int i = 0; i < dy; i++) {
            glVertex2f(x, y);
            if (pk < 0) pk += 2 * dx;
            else { pk += 2 * dx - 2 * dy; x++; }
            y--;
        }
        glEnd();
    }
}
void display() {
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(0, 1, 0);
    glBegin(GL_POLYGON);
    glVertex2f(-100, -100);
    glVertex2f(-100, 100);
    glVertex2f(100, 100);
    glVertex2f(100, -100);
    glEnd();
    // Borders
    bresenhamsLine(-100, -100, -100, 100);
    bresenhamsLine(-100, 100, 100, 100);
    bresenhamsLine(100, 100, 100, -100);
    bresenhamsLine(100, -100, -100, -100);
    glFlush();
}
int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutCreateWindow("Square - Bresenham");
    glClearColor(0, 0, 0, 1);
    glOrtho(-300, 300, -300, 300, -1, 1);
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}
```

Diamon:

```
#include <windows.h>
#include <GL/glut.h>
#include <math.h>
void bresenhamsLine(int x1, int y1, int x2, int y2) {
    if (x1 > x2) { int tx=x1, ty=y1; x1=x2; y1=y2; x2=tx; y2=ty; }
    float m = (x2 - x1) == 0 ? 999 : (float)(y2 - y1) / (float)(x2 - x1);
    int pk, x, y;
    if (m >= 0 && m <= 1) { int dx=x2-x1, dy=y2-y1; pk=2*dy-dx; x=x1; y=y1;
        glBegin(GL_POINTS); glColor3f(1,0,0);
        for (int i=0;i<dx;i++){ glVertex2f(x,y); if(pk<0) pk+=2*dy; else{ pk+=2*dy-2*dx; y++; } x++; }
        glEnd();
    } else if (m > 1) { int dx=x2-x1, dy=y2-y1; pk=2*dx-dy; x=x1; y=y1;
        glBegin(GL_POINTS); glColor3f(1,0,0);
        for (int i=0;i<dy;i++){ glVertex2f(x,y); if(pk<0) pk+=2*dx; else{ pk+=2*dx-2*dy; x++; } y++; }
        glEnd();
    } else if (m < 0 && m >= -1) { int dx=x2-x1, dy=y1-y2; pk=2*dy-dx; x=x1; y=y1;
        glBegin(GL_POINTS); glColor3f(1,0,0);
        for (int i=0;i<dx;i++){ glVertex2f(x,y); if(pk<0) pk+=2*dy; else{ pk+=2*dy-2*dx; y--; } x++; }

        glEnd();

    } else if (m < -1) { int dx=x2-x1, dy=y1-y2; pk=2*dx-dy; x=x1; y=y1;

        glBegin(GL_POINTS); glColor3f(1,0,0);

        for (int i=0;i<dy;i++){ glVertex2f(x,y); if(pk<0) pk+=2*dx; else{ pk+=2*dx-2*dy; x++; } y--; }

        glEnd();

    }

}

void display() {

    glClear(GL_COLOR_BUFFER_BIT);

    glColor3f(0,1,0);

    glBegin(GL_POLYGON);

    glVertex2f(0,150);

    glVertex2f(150,0);

    glVertex2f(0,-150);

    glVertex2f(-150,0);

    glEnd();

    bresenhamsLine(0,150,150,0);

    bresenhamsLine(150,0,0,-150);

    bresenhamsLine(0,-150,-150,0);

    bresenhamsLine(-150,0,0,150);

    glFlush();

}

int main(int argc, char** argv){

    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);

    glutInitWindowSize(500,500);

    glutCreateWindow("Diamond - Bresenham");

    glClearColor(0,0,0,1);

    glOrtho(-300,300,-300,300,-1,1);

    glutDisplayFunc(display);
    glutMainLoop();

    return 0;

}
```

Rectangle :

```
#include <windows.h>

#include <GL/glut.h>

void bresenhamsLine(int x1, int y1, int x2, int y2) {

    if (x1 > x2) { int tx=x1, ty=y1; x1=x2; y1=y2; x2=tx; y2=ty; }

    float m = (x2-x1)==0?999:(float)(y2-y1)/(x2-x1);

    int pk,x,y;

    if (m>=0&&m<=1){int dx=x2-x1,dy=y2-y1;pk=2*dy-dx;x=x1;y=y1;

        glBegin(GL_POINTS);glColor3f(1,0,0);

        for(int i=0;i<dx;i++){glVertex2f(x,y);if(pk<0)pk+=2*dy;else{pk+=2*dy-2*dx;y++;}x++;}glEnd();}

    else if(m>1){int dx=x2-x1,dy=y2-y1;pk=2*dx-dy;x=x1;y=y1;

        glBegin(GL_POINTS);glColor3f(1,0,0);

        for(int i=0;i<dy;i++){glVertex2f(x,y);if(pk<0)pk+=2*dx;else{pk+=2*dx-2*dy;x++;}y++;}glEnd();}

    else if(m<0&&m>=-1){int dx=x2-x1,dy=y1-y2;pk=2*dy-dx;x=x1;y=y1;

        glBegin(GL_POINTS);glColor3f(1,0,0);

        for(int i=0;i<dx;i++){glVertex2f(x,y);if(pk<0)pk+=2*dy;else{pk+=2*dy-2*dx;y--;}x++;}glEnd();}

    else if(m<-1){int dx=x2-x1,dy=y1-y2;pk=2*dx-dy;x=x1;y=y1;

        glBegin(GL_POINTS);glColor3f(1,0,0);

        for(int i=0;i<dy;i++){glVertex2f(x,y);if(pk<0)pk+=2*dx;else{pk+=2*dx-2*dy;x++;}y--;}glEnd();}

}

void display(){

    glClear(GL_COLOR_BUFFER_BIT);

    glColor3f(0,1,0);

    glBegin(GL_POLYGON);

    glVertex2f(-150,-100);

    glVertex2f(-150,100);

    glVertex2f(150,100);

    glVertex2f(150,-100);
    glEnd();

    bresenhamsLine(-150,-100,-150,100);

    bresenhamsLine(-150,100,150,100);

    bresenhamsLine(150,100,150,-100);

    bresenhamsLine(150,-100,-150,-100);

    glFlush();

}

int main(int argc,char**argv){

    glutInit(&argc,argv);

    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);

    glutInitWindowSize(500,500);

    glutCreateWindow("Rectangle - Bresenham");

    glClearColor(0,0,0,1);

    glOrtho(-300,300,-300,300,-1,1);

    glutDisplayFunc(display);

    glutMainLoop();

    return 0;

}
```

Polygon(pentagon):

```
#include <windows.h>

#include <GL/glut.h>

#include <math.h>

void bresenhamsLine(int x1,int y1,int x2,int y2){

    if(x1>x2){int tx=x1,ty=y1;x1=x2;y1=y2;x2=tx;y2=ty;}

    float m=(x2-x1)==0?999:(float)(y2-y1)/(x2-x1);

    int pk,x,y;

    if(m>=0&&m<=1){int dx=x2-x1,dy=y2-y1;pk=2*dy-dx;x=x1;y=y1;

        glBegin(GL_POINTS);glColor3f(1,0,0);

        for(int i=0;i<dx;i++){glVertex2f(x,y);if(pk<0)pk+=2*dy;else{pk+=2*dy-2*dx;y++;}x++;}glEnd();}

    else if(m>1){int dx=x2-x1,dy=y2-y1;pk=2*dx-dy;x=x1;y=y1;

        glBegin(GL_POINTS);glColor3f(1,0,0);

        for(int i=0;i<dy;i++){glVertex2f(x,y);if(pk<0)pk+=2*dx;else{pk+=2*dx-2*dy;x++;}y++;}glEnd();}
    else if(m<0&&m>=-1){int dx=x2-x1,dy=y1-y2;pk=2*dy-dx;x=x1;y=y1;
        glBegin(GL_POINTS);glColor3f(1,0,0);
        for(int i=0;i<dx;i++){glVertex2f(x,y);if(pk<0)pk+=2*dy;else{pk+=2*dy-2*dx;y--;}x++;}glEnd();}
    else if(m<-1){int dx=x2-x1,dy=y1-y2;pk=2*dx-dy;x=x1;y=y1;
        glBegin(GL_POINTS);glColor3f(1,0,0);
        for(int i=0;i<dy;i++){glVertex2f(x,y);if(pk<0)pk+=2*dx;else{pk+=2*dx-2*dy;x++;}y--;}glEnd();}
}
void display(){
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(0,1,0);
    glBegin(GL_POLYGON);
    glVertex2f(0,200);
    glVertex2f(190,60);
    glVertex2f(120,-160);
    glVertex2f(-120,-160);
    glVertex2f(-190,60);
    glEnd();
    // Border
    bresenhamsLine(0,200,190,60);
    bresenhamsLine(190,60,120,-160);
    bresenhamsLine(120,-160,-120,-160);
    bresenhamsLine(-120,-160,-190,60);
    bresenhamsLine(-190,60,0,200);
    glFlush();
}
int main(int argc,char**argv){
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
    glutInitWindowSize(600,600);
    glutCreateWindow("Polygon (Pentagon) - Bresenham");
    glClearColor(0,0,0,1);
    glOrtho(-300,300,-300,300,-1,1);
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}
```

Circle:

```
#include <windows.h>
#include <GL/glut.h>
#include <math.h>
// Function to draw symmetric points
void drawCirclePoints(int xc, int yc, int x, int y) {
    glBegin(GL_POINTS);
    glVertex2f(xc + x, yc + y);
    glVertex2f(xc - x, yc + y);
    glVertex2f(xc + x, yc - y);
    glVertex2f(xc - x, yc - y);
    glVertex2f(xc + y, yc + x);
    glVertex2f(xc - y, yc + x);
    glVertex2f(xc + y, yc - x);
    glVertex2f(xc - y, yc - x);
    glEnd();
}
// Bresenham's (Midpoint) Circle Algorithm
void bresenhamCircle(int xc, int yc, int r) {
    int x = 0;
    int y = r;
    int d = 3 - (2 * r);
    glColor3f(1.0f, 0.0f, 0.0f); // Red color for border
    while (x <= y) {
        drawCirclePoints(xc, yc, x, y);
        if (d < 0)
```

```
            d = d + (4 * x) + 6;
        else {
            d = d + 4 * (x - y) + 10;
            y--;
        }
        x++;
    }
}
// Display function
void display() {
    glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
    glClear(GL_COLOR_BUFFER_BIT);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-500, 500, -500, 500, -1, 1);
    // Fill the circle using GL_POLYGON
    glColor3f(0.0f, 1.0f, 0.0f); // Green fill
    glBegin(GL_POLYGON);
    int xc = 0, yc = 0, r = 200;
    for (int angle = 0; angle <= 360; angle++) {
        float rad = angle * 3.1416 / 180;
        glVertex2f(xc + r * cos(rad), yc + r * sin(rad));
    }
    glEnd();
    // Circle border using Bresenham's algorithm
    bresenhamCircle(0, 0, 200);
    glFlush();
}
// Main
int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(100, 100);
    glutCreateWindow("Circle - Bresenham’s Algorithm");
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}
```

DDA:

SQR:

```
#include <windows.h>
#include <GL/glut.h>
#include <math.h>
void ddaLine(int x1, int y1, int x2, int y2) {
    int dx = x2 - x1;
    int dy = y2 - y1;
    int steps = abs(dx) > abs(dy) ? abs(dx) : abs(dy);
    float xInc = dx / (float)steps;
    float yInc = dy / (float)steps;
    float x = x1, y = y1;
    glBegin(GL_POINTS);
    glColor3f(1, 0, 0);
    for (int i = 0; i <= steps; i++) {
        glVertex2f(x, y);
        x += xInc;
        y += yInc;
    }
    glEnd();
}
void display() {
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(0, 1, 0);
    // Filled shape
    glBegin(GL_POLYGON);
    glVertex2f(-100, -100);
    glVertex2f(100, -100);
    glVertex2f(100, 100);
    glVertex2f(-100, 100);
    glEnd();
    // Border using DDA
    ddaLine(-100, -100, 100, -100);
    ddaLine(100, -100, 100, 100);
    ddaLine(100, 100, -100, 100);
    ddaLine(-100, 100, -100, -100);
    glFlush();
}
int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutCreateWindow("Square - DDA");
    glClearColor(0, 0, 0, 1);
    glOrtho(-300, 300, -300, 300, -1, 1);
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}
```

RCTNGL:

```
#include <windows.h>
#include <GL/glut.h>
#include <math.h>
```

```
void ddaLine(int x1, int y1, int x2, int y2) {
    int dx = x2 - x1;
    int dy = y2 - y1;
    int steps = abs(dx) > abs(dy) ? abs(dx) : abs(dy);
    float xInc = dx / (float)steps;
    float yInc = dy / (float)steps;
    float x = x1, y = y1;
    glBegin(GL_POINTS);
    glColor3f(1, 0, 0);
    for (int i = 0; i <= steps; i++) {
        glVertex2f(x, y);
        x += xInc;
        y += yInc;
    }
    glEnd();
}
void display() {
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(0, 1, 0);
    // Filled rectangle
    glBegin(GL_POLYGON);
    glVertex2f(-150, -100);
    glVertex2f(150, -100);
    glVertex2f(150, 100);
    glVertex2f(-150, 100);
    glEnd();
    // Border using DDA
    ddaLine(-150, -100, 150, -100);
    ddaLine(150, -100, 150, 100);
    ddaLine(150, 100, -150, 100);
    ddaLine(-150, 100, -150, -100);
```

```
 glFlush();
}
int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutCreateWindow("Rectangle - DDA");
    glClearColor(0, 0, 0, 1);
    glOrtho(-300, 300, -300, 300, -1, 1);
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}
```

Diamond:

```
#include <windows.h>
#include <GL/glut.h>
#include <math.h>
void ddaLine(int x1, int y1, int x2, int y2) {
    int dx = x2 - x1;
    int dy = y2 - y1;
    int steps = abs(dx) > abs(dy) ? abs(dx) : abs(dy);
    float xInc = dx / (float)steps;
    float yInc = dy / (float)steps;
    float x = x1, y = y1;
    glBegin(GL_POINTS);
    glColor3f(1, 0, 0);
    for (int i = 0; i <= steps; i++) {
        glVertex2f(x, y);
        x += xInc;
        y += yInc;
    }
```

```
 glEnd();
}
void display() {
   glClear(GL_COLOR_BUFFER_BIT);
   glColor3f(0, 1, 0);
   // Diamond shape fill
   glBegin(GL_POLYGON);
   glVertex2f(0, 150);
   glVertex2f(150, 0);
   glVertex2f(0, -150);
   glVertex2f(-150, 0);
   glEnd();
   // Border using DDA
   ddaLine(0,150,150,0);
   ddaLine(150,0,0,-150);
   ddaLine(0,-150,-150,0);
   ddaLine(-150,0,0,150);
   glFlush();
}
int main(int argc, char** argv) {
   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
   glutInitWindowSize(500, 500);
   glutCreateWindow("Polygon (Diamond) - DDA");
   glClearColor(0, 0, 0, 1);
   glOrtho(-300, 300, -300, 300, -1, 1);
   glutDisplayFunc(display);
   glutMainLoop();
   return 0;
}
```

CIRCLE:

```
#include <windows.h>

#include <GL/glut.h>

#include <math.h>

void ddaCircle(int xc, int yc, int r) {

    glBegin(GL_POINTS);

    glColor3f(1, 0, 0);

    for (float theta = 0; theta <= 360; theta += 0.1) {

        float x = xc + r * cos(theta * 3.1416 / 180);

        float y = yc + r * sin(theta * 3.1416 / 180);

        glVertex2f(x, y);

    }

    glEnd();

}

void display() {

    glClear(GL_COLOR_BUFFER_BIT);

    glColor3f(0, 1, 0);

    // Filled circle

    glBegin(GL_POLYGON);

    for (float theta = 0; theta <= 360; theta += 1) {

        float x = 0 + 150 * cos(theta * 3.1416 / 180);

        float y = 0 + 150 * sin(theta * 3.1416 / 180);

        glVertex2f(x, y);

    }

    glEnd();

    // Border using DDA circle

    ddaCircle(0, 0, 150);

    glFlush();

}

int main(int argc, char** argv) {

    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500, 500);

    glutCreateWindow("Circle - DDA");

    glClearColor(0, 0, 0, 1);

    glOrtho(-300, 300, -300, 300, -1, 1);

    glutDisplayFunc(display);

    glutMainLoop();

    return 0;

}
```