**Лабораторная работа 1. Двумерные примитивы.**

**Студент:** Петров Артем Евгеньевич

**Группа:** НКНбд-01-21

**Студенческий билет:** 1032219251

**Цель:** написать компьютерную программу для построения двумерных примитивов «Линия» и «Окружность».

**Условие:** Программа должна соответствовать следующим требованиям:

- За построение примитива «Линия» должна отвечать функция

void myline(int x1,int y1, int x2,int y2, int c),

- За построение примитива «Окружность» должна отвечать функция

void mycirc (int x0,int y0, int r, int c)

- Для рисования примитивов «Линия» и «Окружность» должны использоваться целочисленные алгоритмы Бразенхайма;

- Функцию myline надо интегрировать в программу рисования дерева Пифагора.

**Дано:** x1, y1 – растровые координаты начала линии,

x2, y2 – растровые координаты конца линии,

c – цвет линии;

x0, y0 – растровые координаты центра окружности,

r – радиус окружности,

c – цвет окружности;

**Ход Работы:**

1. Подключим необходимые библиотеки(Рисунок 1):
2. #include <D:\documents\c++\Computer graphics\graphics.h>
3. #include <iostream>
4. #include "graphics.h"
5. #include <stdlib.h>
6. #include <time.h>
7. #include <cmath>
8. #include <cstdlib>
9. using namespace std;

Рисунок 1. Подключаемые библиотеки

**2.** Теперь реализуем функцию рисования линии следующим образом(Рисунок 2):

void myLine(int x0, int y0, int x1, int y1, int c){

    int dx = abs(x1-x0); int dy = abs(y1-y0);

    int error = 0;

    int deltaerr = (dy+1);

    int direction = y1-y0;

    if (direction > 0) direction = 1;

    else direction = -1;

    int y = y0;

    for(int x = x0; x <= x1; x++){

        putpixel(x, y, c);

        error += deltaerr;

        if(error >= dx + 1){

            y = y + direction;

            error -= (dx + 1);

        }

    }

}

Рисунок 2. Моя реализация алгоритма Бразенхейма.

Опишу свою реализацию: сначала мы определим коэффицент возрастания по x и у, ошибку и среднюю ошибку. Потом определим направление нашей функции – вверх или вниз. Создадим y, которая начинается с начальной координаты у и будет служить для хранения следующего значения у. Дальше запустим цикл, который и будет отрисовывать нашу прямую. Делать это он будет так: отмечает пиксель точки по координатам х и у. Делать он это будет как бы по горизонтали, но в случае, если накапливается ошибка больше коэффицентой, то она увеливает значение y на направление прямой(вверх или вниз на 1).

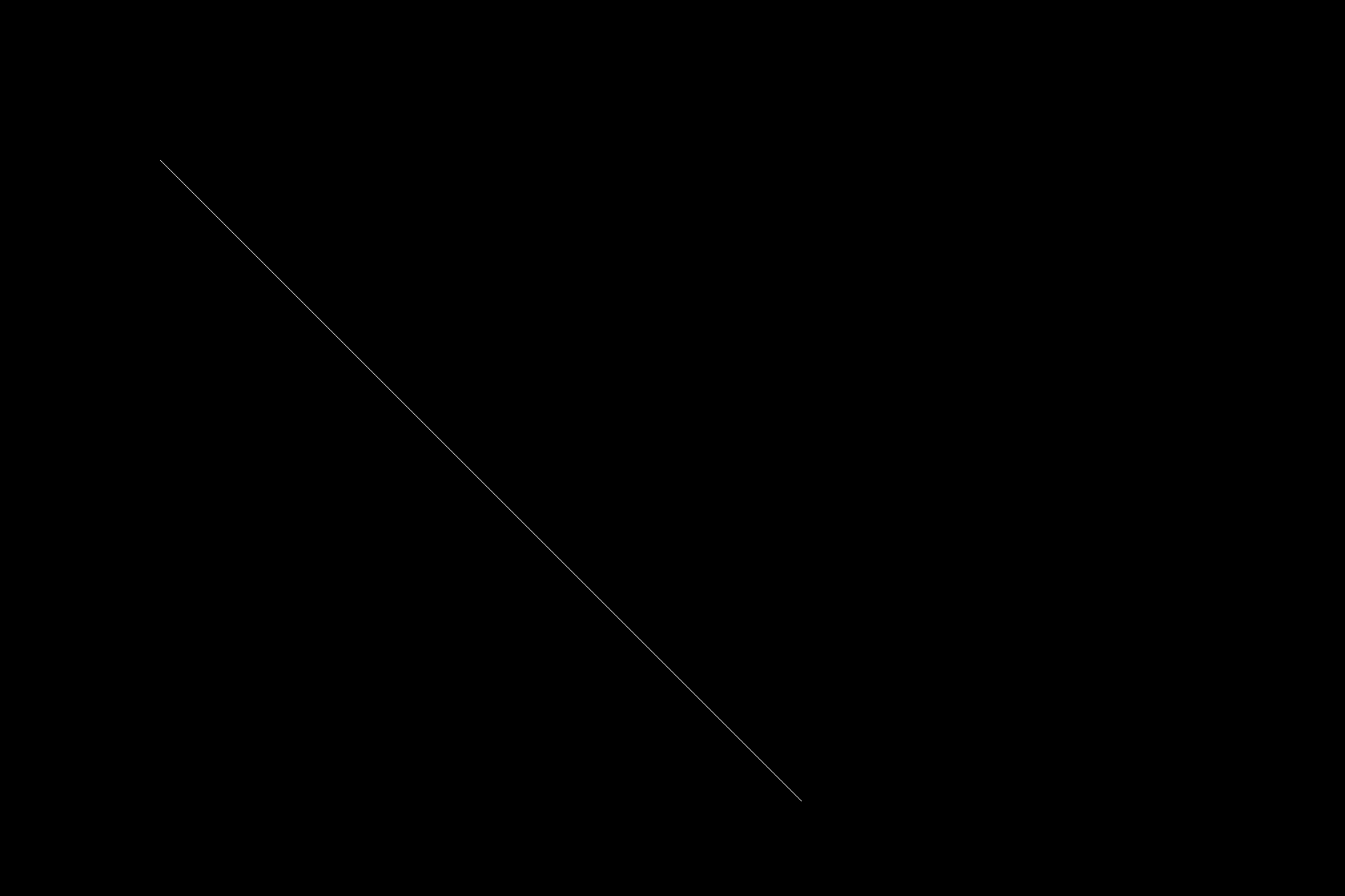
Вот какую прямую она нарисует по начальным координатам (200, 200, 1000, 1000)(нужно учитывать только то, что экран начинается с левого края)(Рисунок 3): 

Рисунок 3. Прямая (200, 200, 1000, 1000).

**3.** Функция main(Рисунок 4):

int main(){

    int gddriver = DETECT, gmode, errorcode;

    initgraph(&gddriver, &gmode, "");

    srand(time(NULL));

    myLine(200, 200, 1000, 1000, 15);

    getch();

    closegraph();

    return 0;

}

Рисунок 4. Тело main.

4. Алгоритм рисования окружности(Рисунок 5):

void myCirc(int x0, int y0, int r, int c){

    int x = 0;

    int y = r;

    int delta = 1 - 2\*r;

    int error = 0;

    while(y>=x){

        putpixel(x0 + x, y0 + y, c);

        putpixel(x0 + x, y0 + y, c);

        putpixel(x0 + x, y0 - y, c);

        putpixel(x0 - x, y0 + y, c);

        putpixel(x0 - x, y0 - y, c);

        putpixel(x0 + y, y0 + x, c);

        putpixel(x0 + y, y0 - x, c);

        putpixel(x0 - y, y0 + x, c);

        putpixel(x0 - y, y0 - x, c);

        error = 2 \* (delta+y) - 1;

        if((delta < 0) && (error <= 0)){

            x += 1;

            delta += 2 \* x + 1;

            continue;

        }

        if((delta > 0) && (error > 0)){

            y -= 1;

            delta -= 2 \* y + 1;

            continue;

        }

        x += 1;

        y -= 1;

        delta += 2 \* (x - y);

    }

}

Рисунок 5. Алгоритм Бразенхейма для рисования окружности.

Работает он следующим образом: В этом алгоритме строится дуга окружности для первого квадранта, а координаты точек окружности для остальных квадрантов получаются симметрично. На каждом шаге алгоритма рассматриваются три пикселя, и из них выбирается наиболее подходящий путём сравнения расстояний от центра до выбранного пикселя с радиусом окружности.

Окружность с параметрами(500, 500, 400) будет выглядеть так(Рисунок 6):

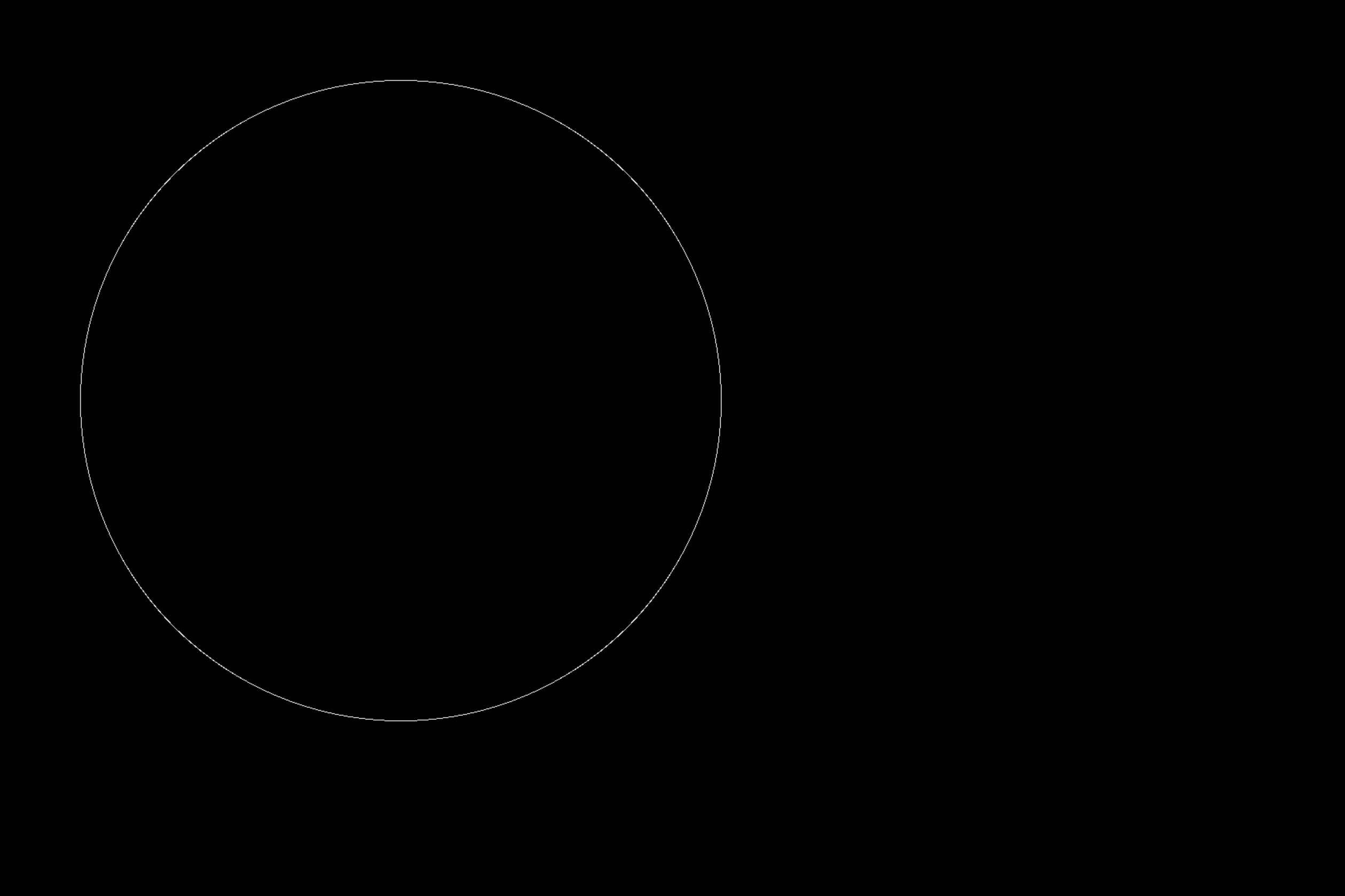


Рисунок 6. Окружность с параметрами(500, 500, 400).

**5.** Функция main(Рисунок 7).

int main(){

    int gddriver = DETECT, gmode, errorcode;

    initgraph(&gddriver, &gmode, "");

    srand(time(NULL));

    // myLine(200, 200, 1000, 1000, 15);

    myCirc(500, 500, 400, 15);

    getch();

    closegraph();

    return 0;

}

7. Тело main.

Вывод: с помощью данной лабораторной работы я научился использовать алгоритм Бразенхейма для рисования окружности и прямой.