РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ
Факультет физико-математических и естественных наук

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

дисциплина: Компьютерная графика

Студент: Мухамедияр Адиль

Группа: НКНбд-01-20

МОСКВА 2022г.

Задание

Задание: написать компьютерную программу для построения двумерных примитивов «Линия» и «Окружность». Программа должна соответствовать следующим требованиям:

- За построение примитива «Линия» должна отвечать функция
  void myline(int x1,int y1, int x2,int y2, int c), в которой
  x1, y1 – растровые координаты начала линии,
  x2, y2 – растровые координаты конца линии,
  c – цвет линии;

- За построение примитива «Окружность» должна отвечать функция
  void mycirc (int x0,int y0, int r, int c), в которой
  x0, y0 – растровые координаты центра окружности,
  r – радиус окружности,
  c – цвет окружности;

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

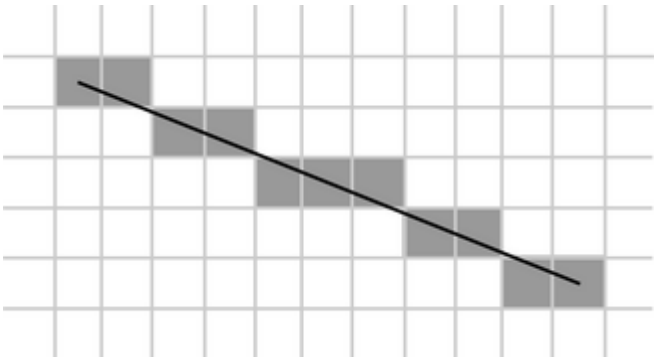
- Функцию myline надо интегрировать в программу рисования дерева Пифагора.

Результаты выполнения работы должны содержать:
- Отчет о выполнении лабораторной работы;
- Текст компьютерной программы;
- Изображения – результат работы компьютерной программы.

Теоретическая справка

В данной лабораторной работе мы рассматриваем Алгоритм Брезенхема

Это алгоритм, определяющий, какие точки n-мерного растра нужно закрасить, чтобы получить близкое приближение прямой линии между двумя заданными точками.



Хотя алгоритм Брезенхема был первоначально разработан для цифровых графопостроителей, однако он в равной степени подходит для использования растровыми устройствами с ЭЛТ. Алгоритм выбирает оптимальные растровые координаты для представления отрезка. Берётся отрезок и его начальная координата x. К иксу в цикле прибавляем по единичке в сторону конца отрезка. На каждом шаге вычисляется ошибка — расстояние между реальной координатой y в этом месте и ближайшей ячейкой сетки. Если ошибка не превышает половину высоты ячейки, то она заполняется. Вот и весь алгоритм.

Также существует алгоритм Брезенхема для рисования окружностей. По методу построения он похож на рисование линии. 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.

Ход решения

```
#include <stdlib.h>
#include <math.h>
#include <conio.h>
#include <graphics.h>

void myline(double x1,double y1,double x2,double y2,int c) {
    int dx,dy,sx,sy;
    putpixel(x2,y2,c);
    if(x2 - x1 >= 0){dx = (int)(x2 - x1);}
    else dx = (int)(x1 - x2);
    if(y2 - y1 >= 0){dy = (int)(y2 - y1);}
    else dy = (int)(y1 - y2);
    if(x1 < x2){sx = 1;}
    else sx = -1;
    if(y1 < y2){sy = 1;}
    else sy = -1;
    int error = dx - dy;
    while(x1 != x2 || y1 != y2)
   {
        putpixel(x1, y1, c);
        int error2 = error * 2;
        if(error2 > -dy){
            error -= dy;
            x1 += sx;}
        if(error2 < dx){
            error += dx;
            y1 += sy;}
    }
}
```

```
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 >= 0) {
        error = 2*(delta + y) - 1;
        putpixel(x0 - x, y0 - y, c);
        putpixel(x0 + x, y0 - y, c);
        putpixel(x0 + x, y0 + y, c);
        putpixel(x0 - x, y0 + y, c);
        if(delta < 0 && error <= 0) {
            x++;
            delta += 2 * x + 1;
            continue;
        }
        error = 2 * (delta - x) - 1;
        if(delta > 0 && error > 0) {
            y--;
            delta += 1-2*y;
            continue;
        }
        x++;
        delta += 2*(x-y);
        y--;
    }
}
```

```
void pifagor(int n, double x0, double y0, double a, double fi, double alpha, int c) {
    double x1, y1, x2, y2, x3, y3, x4, y4;
    double grad = M_PI/180;
    /* Вычисляем координаты точек */
    x1 = x0 - a * cos(fi * grad);
    y1 = y0 - a * sin(fi * grad);
    x2 = x1 + a * sin(fi * grad);
    y2 = y1 - a * cos(fi * grad);
    x3 = x0 + a * sin(fi * grad);
    y3 = y0 - a * cos(fi * grad);
    x4 = x3 - a * cos(alpha * grad) * cos((fi + alpha) * grad);
    y4 = y3 - a * cos(alpha * grad) * sin((fi + alpha) * grad);
    /* Соединяем линиями вершины использую собвственную функцию myline */
    myline(x0, y0, x1, y1, c);
    myline(x1, y1, x2, y2, c);
    myline(x2, y2, x3, y3, c);
    myline(x0, y0, x3, y3, c);
    if (n > 0) {
        myline(x3, y3, x4, y4, c);
        myline(x2, y2, x4, y4, c);
        /* Проверяем глубину и вызываем для катетов, если не дошли */
        pifagor(n - 1, x3, y3, a * cos(alpha * grad), fi + alpha, alpha, c);
        pifagor(n - 1, x4, y4, a * sin(alpha * grad), fi + alpha - 90, alpha, c);
    }
}
```

```
int main()
{
    /* Запрос автоопределения */
    int gddriver = DETECT, gmode, errorcode;
    /* Инициализация графики и локальных переменных */
    initgraph(&gddriver, &gmode, "");
    pifagor(6, 200, 200, 40, 0, 45, 2);

    myline(25, 325, 600, 325, 4);

    mycirc(200, 400, 50, 8);
    mycirc(275, 400, 50, 9);
    mycirc(350, 400, 50, 10);
    mycirc(425, 400, 50, 11);
    /* Очистка */
    getch();
    closegraph;
    return 0;
}
```

Исполнение программы