Министерство образования и науки Российской Федерации
Федеральное агентство по образованию
Федеральное государственное бюджетное образовательное
учреждениевысшего образования
«Вятский государственный университет»

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Лабораторная работа №7
по курсу «Программирование»

Исследование фракталов

Выполнил студент группы ИВТ-11 _____/Рзаев А. Э./
Проверил преподаватель _____/Чистяков Г. А./

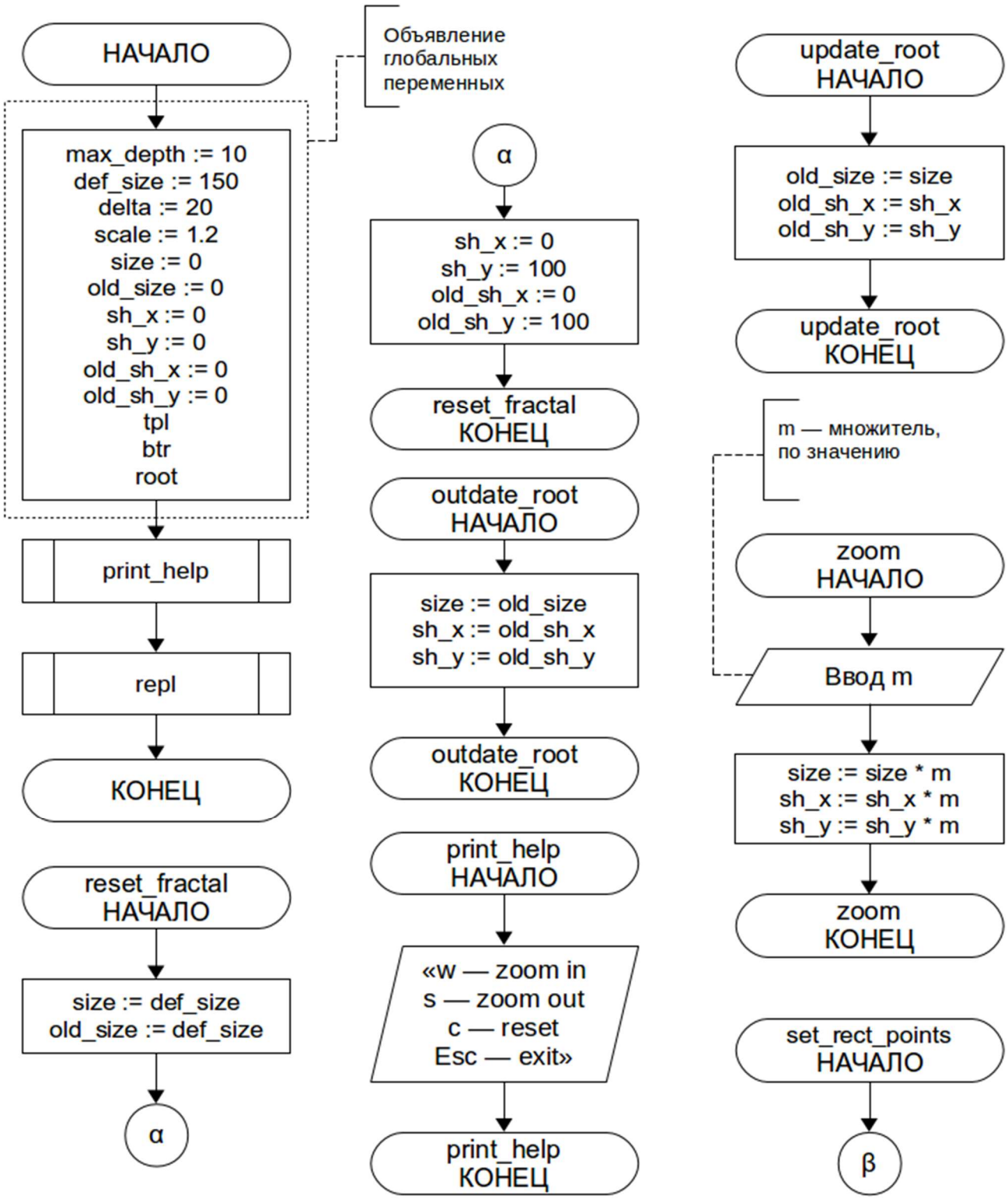
Киров 2016

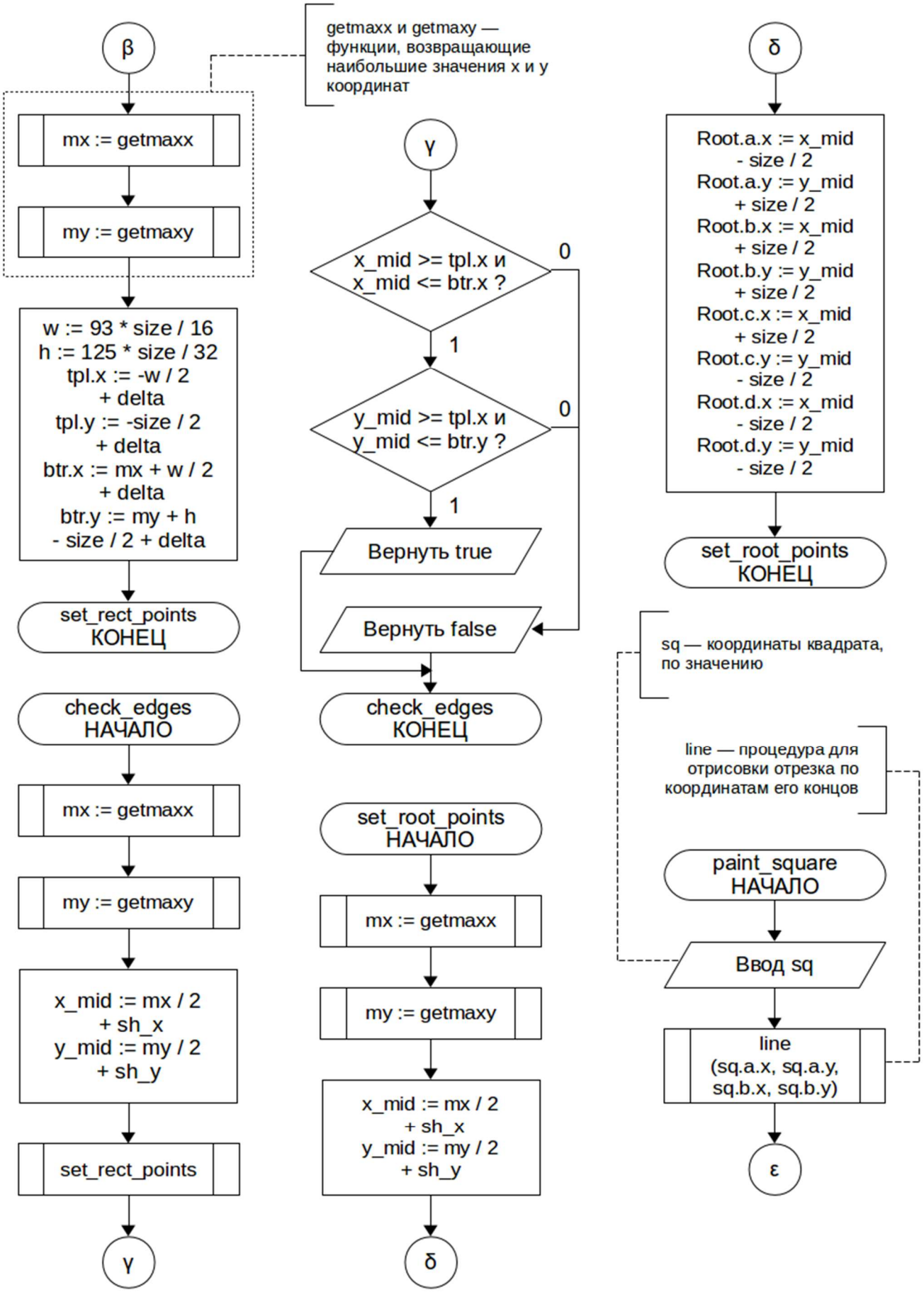
Цель работы: получить базовые навыки работы с модулем graph, изучить принципы построения рекурсивных подпрограмм.

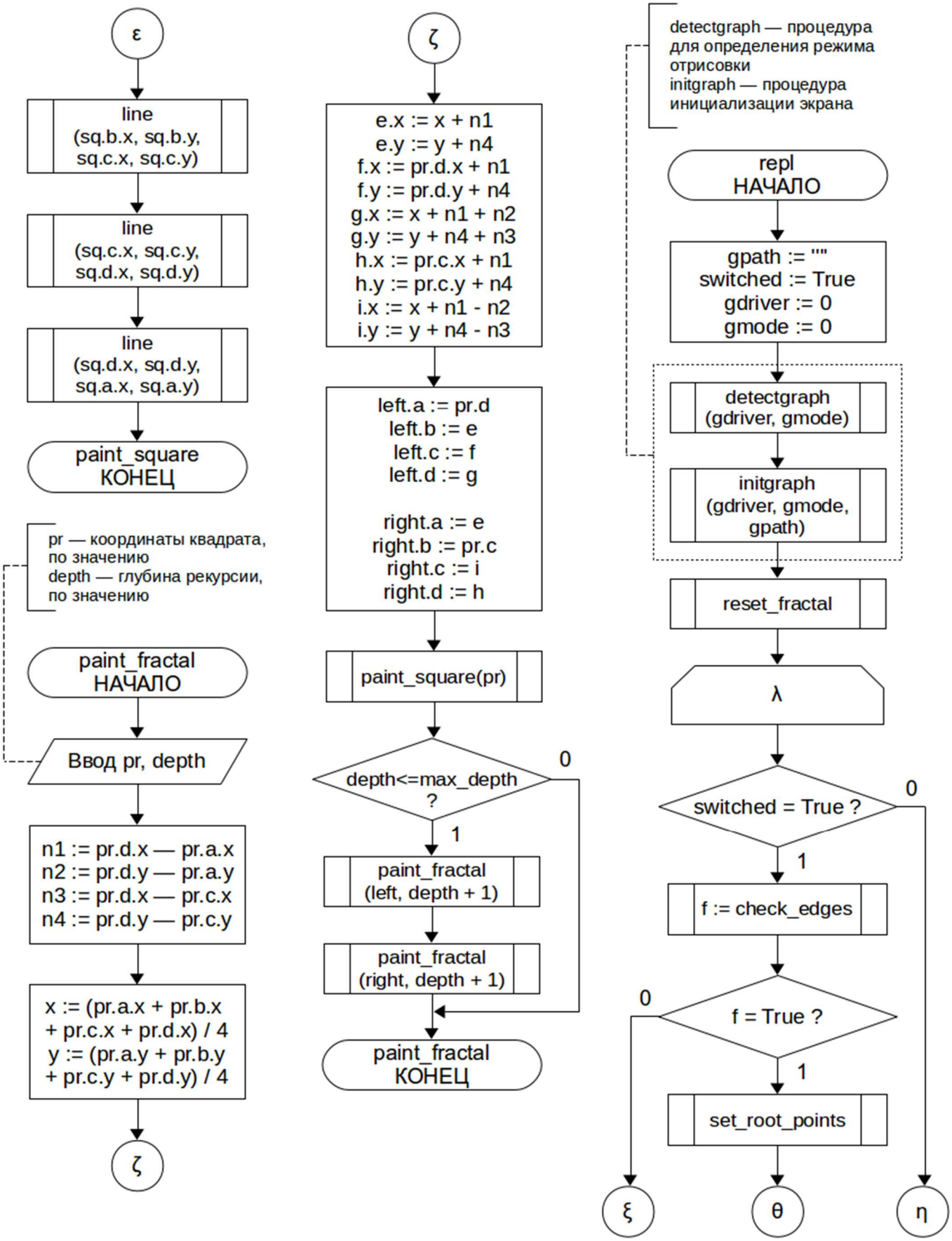
Задание:

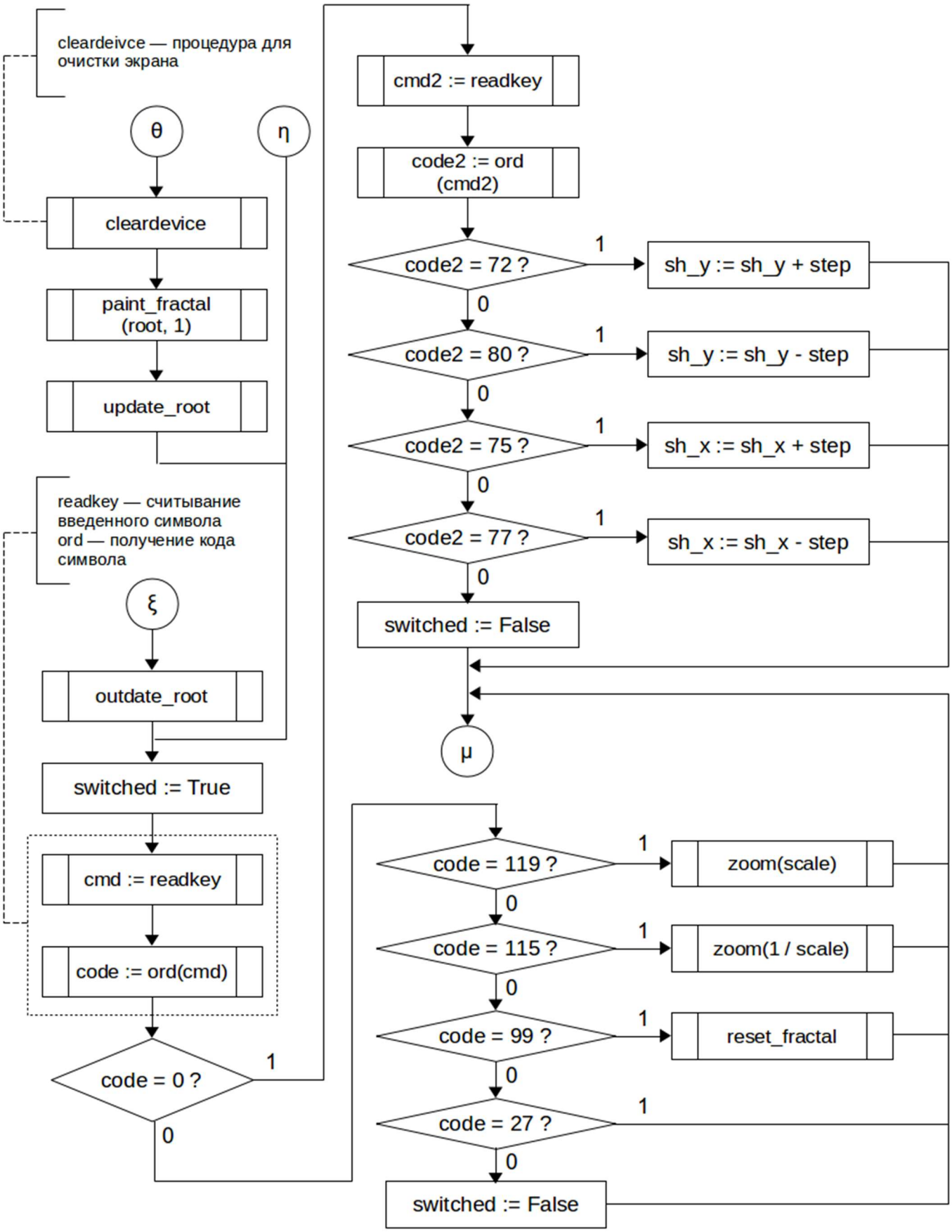
1. Используя модуль graph построить фрактал "Дерево Пифагора".
2. Реализовать возможность масштабирования изображения с изменением детальности прорисовки в зависимости от уровня приближения.

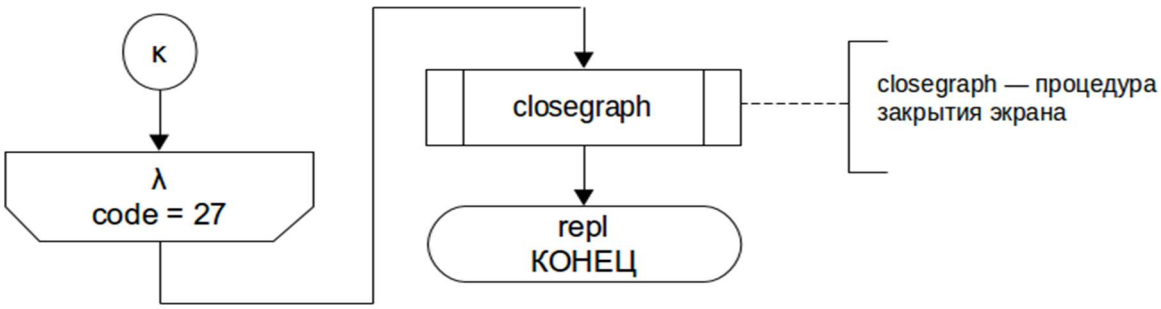
Схема алгоритма:




β
getmaxx и getmaxy — функции, возвращающие наибольшие значения x и y координат
mx := getmaxx
my := getmaxy
w := 93 * size / 16
h := 125 * size / 32
tpl.x := -w / 2 + delta
tpl.y := -size / 2 + delta
btr.x := mx + w / 2 + delta
btr.y := my + h - size / 2 + delta
set_rect_points КОНЕЦ
γ
x_mid >= tpl.x и x_mid <= btr.x ?
0
1
y_mid >= tpl.x и y_mid <= btr.y ?
0
1
Вернуть true
Вернуть false
check_edges КОНЕЦ
check_edges НАЧАЛО
mx := getmaxx
my := getmaxy
x_mid := mx / 2 + sh_x
y_mid := my / 2 + sh_y
set_rect_points
γ
set_root_points НАЧАЛО
mx := getmaxx
my := getmaxy
x_mid := mx / 2 + sh_x
y_mid := my / 2 + sh_y
δ
δ
Root.a.x := x_mid - size / 2
Root.a.y := y_mid + size / 2
Root.b.x := x_mid + size / 2
Root.b.y := y_mid + size / 2
Root.c.x := x_mid + size / 2
Root.c.y := y_mid - size / 2
Root.d.x := x_mid - size / 2
Root.d.y := y_mid - size / 2
set_root_points КОНЕЦ
sq — координаты квадрата, по значению
line — процедура для отрисовки отрезка по координатам его концов
paint_square НАЧАЛО
Ввод sq
line (sq.a.x, sq.a.y, sq.b.x, sq.b.y)
ε


ε
line
(sq.b.x, sq.b.y,
sq.c.x, sq.c.y)
line
(sq.c.x, sq.c.y,
sq.d.x, sq.d.y)
line
(sq.d.x, sq.d.y,
sq.a.x, sq.a.y)
paint_square
КОНЕЦ
pr — координаты квадрата,
по значению
depth — глубина рекурсии,
по значению
paint_fractal
НАЧАЛО
Ввод pr, depth
n1 := pr.d.x — pr.a.x
n2 := pr.d.y — pr.a.y
n3 := pr.d.x — pr.c.x
n4 := pr.d.y — pr.c.y
x := (pr.a.x + pr.b.x
+ pr.c.x + pr.d.x) / 4
y := (pr.a.y + pr.b.y
+ pr.c.y + pr.d.y) / 4
ζ
ζ
e.x := x + n1
e.y := y + n4
f.x := pr.d.x + n1
f.y := pr.d.y + n4
g.x := x + n1 + n2
g.y := y + n4 + n3
h.x := pr.c.x + n1
h.y := pr.c.y + n4
i.x := x + n1 - n2
i.y := y + n4 - n3
left.a := pr.d
left.b := e
left.c := f
left.d := g
right.a := e
right.b := pr.c
right.c := i
right.d := h
paint_square(pr)
depth<=max_depth
?
0
1
paint_fractal
(left, depth + 1)
paint_fractal
(right, depth + 1)
paint_fractal
КОНЕЦ
detectgraph — процедура
для определения режима
отрисовки
initgraph — процедура
инициализации экрана
repl
НАЧАЛО
gpath := ""
switched := True
gdriver := 0
gmode := 0
detectgraph
(gdriver, gmode)
initgraph
(gdriver, gmode,
gpath)
reset_fractal
λ
switched = True ?
0
1
f := check_edges
f = True ?
0
1
set_root_points
ξ
θ
η


cleardeivce — процедура для очистки экрана
θ
η
cleardevice
paint_fractal (root, 1)
update_root
readkey — считывание введенного символа
ord — получение кода символа
ξ
outdate_root
switched := True
cmd := readkey
code := ord(cmd)
code = 0 ?
1
0
cmd2 := readkey
code2 := ord (cmd2)
code2 = 72 ?
sh_y := sh_y + step
code2 = 80 ?
sh_y := sh_y - step
code2 = 75 ?
sh_x := sh_x + step
code2 = 77 ?
sh_x := sh_x - step
switched := False
μ
code = 119 ?
zoom(scale)
code = 115 ?
zoom(1 / scale)
code = 99 ?
reset_fractal
code = 27 ?
switched := False



Листинг кода:

```
uses graph, wincrt;

const
    max_depth = 10;
    def_size = 150.0;
    step = 10.0;
    delta = 20.0;
    scale = 1.2;

type
    Point = record
        x, y : Real;
    end;

    Square = record
        a, b, c, d : Point;
    end;

var size, old_size : Real;
    sh_x, sh_y : Real;
    old_sh_x, old_sh_y : Real;
    tpl, btr : Point;
    root : Square;

procedure update_root;
begin
    old_size := size;
    old_sh_x := sh_x;
    old_sh_y := sh_y;
end;

procedure outdate_root;
begin
    size := old_size;
    sh_x := old_sh_x;
```

```
    sh_y := old_sh_y;
end;

procedure print_help;
begin
    writeln('w - zoom in');
    writeln('s - zoom out');
    writeln('c - reset');
    writeln('ESC - exit');
end;

procedure zoom(m : Real);
begin
    size := size * m;
    sh_x := sh_x * m;
    sh_y := sh_y * m;
end;

procedure set_rectangle_points;
var
    w, h : Real;
begin
    w := 93 * size / 16;
    h := 125 * size / 32;

    tpl.x := - w / 2 + delta; // левый верхний
    tpl.y := - size / 2 +delta;

    btr.x := getmaxx + w / 2 - delta; // правый нижний
    btr.y := getmaxy + h - size / 2 - delta;
end;

function check_edges : Boolean;
var
    x_mid, y_mid : Real;
begin
    x_mid := getmaxx / 2 + sh_x;
    y_mid := getmaxy / 2 + sh_y;

    set_rectangle_points;

    if ( (x_mid>= tpl.x) and (x_mid<= btr.x) and
         (y_mid>= tpl.y) and (y_mid<= btr.y) ) then
        check_edges := True
    else
        check_edges := False;
```

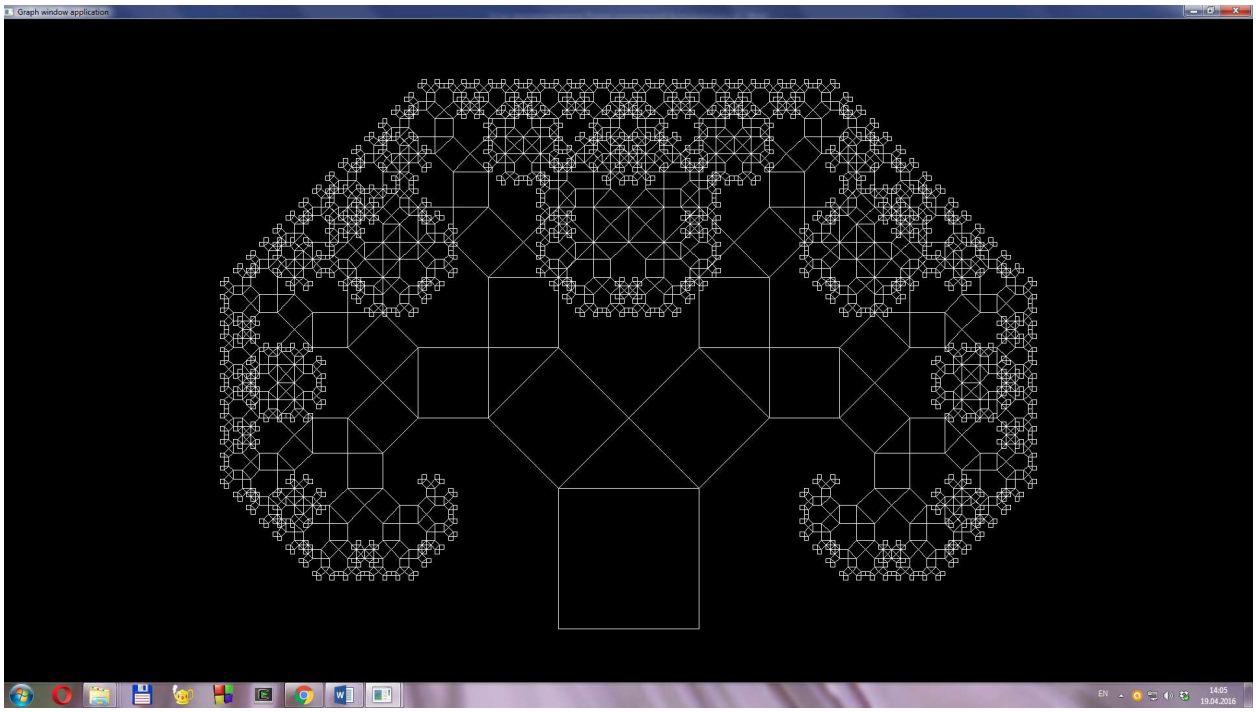
```
end;

procedure set_root_points;
var
    x_mid, y_mid : Real;
begin
    x_mid := getmaxx / 2 + sh_x;
    y_mid := getmaxy / 2 + sh_y;

    root.a.x := x_mid - size / 2;
    root.a.y := y_mid + size / 2;

    root.b.x := x_mid + size / 2;
    root.b.y := y_mid + size / 2;

    root.c.x := x_mid + size / 2;
    root.c.y := y_mid - size / 2;

    root.d.x := x_mid - size / 2;
    root.d.y := y_mid - size / 2;
end;

procedure paint_square(sq : Square);
begin
    line( round(sq.a.x), round(sq.a.y),
          round(sq.b.x), round(sq.b.y) );
    line( round(sq.b.x), round(sq.b.y),
          round(sq.c.x), round(sq.c.y) );
    line( round(sq.c.x), round(sq.c.y),
          round(sq.d.x), round(sq.d.y) );
    line( round(sq.d.x), round(sq.d.y),
          round(sq.a.x), round(sq.a.y) );
end;

procedure paint_fractal(pr : Square; depth : Integer);
var
    x, y : Real;
    n1, n2, n3, n4 : Real;
    left, right : Square;
    e, f, g, h, i : Point;
begin
    if keypressed then
        exit;

    x := (pr.a.x + pr.b.x + pr.c.x + pr.d.x) / 4;
    y := (pr.a.y + pr.b.y + pr.c.y + pr.d.y) / 4;
```

```
    n1 := pr.d.x - pr.a.x;
    n4 := pr.d.y - pr.a.y;
    n2 := pr.d.x - pr.c.x;
    n3 := pr.d.y - pr.c.y;

    e.x := x + n1;          e.y := y + n4;
    f.x := pr.d.x + n1;     f.y := pr.d.y + n4;
    g.x := x + n1 + n2;     g.y := y + n4 + n3;
    h.x := pr.c.x + n1;     h.y := pr.c.y + n4;
    i.x := x + n1 - n2;     i.y := y + n4 - n3;

    left.a := pr.d; left.b := e;
    left.c := f;        left.d := g;

    right.a := e;   right.b := pr.c;
    right.c := i;   right.d := h;

    paint_square(pr);

    if depth <= max_depth then
    begin
        paint_fractal(left, depth + 1);
        paint_fractal(right, depth + 1);
    end;
end;

procedure reset_fractal;
begin
    size := def_size;
    old_size := def_size;
    sh_x := 0;   old_sh_x := 0;
    sh_y := 100; old_sh_y := 100;
end;

procedure repl;
var cmd : Char;
    switched : Boolean;
    gpath : String;
    gdriver, gmode :SmallInt;
begin
    gpath := '';
    detectgraph(gdriver, gmode);
    initgraph(gdriver, gmode, gpath);
    reset_fractal;
    switched := True;
```

```
repeat
    if switched then
    begin
        if check_edges then
        begin
            set_root_points;
            cleardevice;
            paint_fractal(root, 1);
            update_root;
        end
        else
        begin
            outdate_root;
        end;
    end;
cmd := readkey;
switched := True;
    if ord(cmd) = 0 then
    begin
        case ord(readkey) of
        72 : // вверх
            sh_y := sh_y + step;
        80 : // вниз
            sh_y := sh_y - step;
        75 : // влево
            sh_x := sh_x + step;
        77 : // вправо
            sh_x := sh_x - step;
        else
            switched := False;
        end;
    end
    else
        case ord(cmd) of
        119 : // w - увеличить
            zoom(scale);
        115 : // s - уменьшить
            zoom(1 / scale);
        99 : // c - сброс
            reset_fractal;
        27 : // ESC - выйти
            ;
        else
            switched := False;
        end;
until ord(cmd) = 27;
```

```
    closegraph;
end;

begin
    print_help;
    repl;
end.
```

Экранная форма:



Вывод: в данной лабораторной работе были получены базовые навыки работы с модулем graph, изучены принципы построения рекурсивных подпрограмм.