**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Вятский государственный университет»**

**Колледж ВятГУ**

**ОТЧЕТ**

**ПО ДОМАШНЕЙ КОНТРОЛЬНОЙ РАБОТЕ №7**

**«ИССЛЕДОВАНИЕ ФРАКТАЛОВ»**

**ПО ДИСЦИПЛИНЕ «МДК 05.02 Разработка кода информационных систем»**

Выполнила: студент учебной группы

ИСПк-204-52-00

Исупов Максим Михайлович

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2024

**Цель работы**

Получение навыков реализации алгоритмов с рекурсивными вычислениями, знакомство с фракталами.

**Формулировка задания**

1. Написать программу для визуализации фрактала “Кривая Минковского”

2. Предусмотреть возможности масштабирования, изменения глубины прорисовки и перемещения полученной фигуры.

3. Построение множества множества ломанных, образующих фрактал, изменения глубины прорисовки и перемещения полученной фигуры.

**Описание алгоритма**

Данный алгоритм рисует фрактал "Кривая дракона" с помощью графической библиотеки GraphABC и предоставлять возможность масштабировать фрактал.

Процесс построения фрактала Кривая Минковского можно описать следующим образом:

* Кривая Минковского нигде не дифференцируема и не спрямляема.
* Кривая Минковского не имеет самопересечений.
* Кривая Минковского имеет Хаусдорфову размерность ln8/ln4 = 3/2 (поскольку она состоит из восьми равных частей, каждая из которых подобна всей кривой с коэффициентом подобия 1/4).
* Кривая Минковского имеет нулевую меру Лебега.

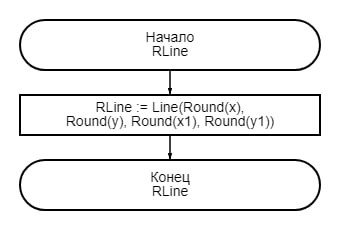
**Схема алгоритма с комментариями**

Рис. 1 «Схема алгоритма процедуры RLine»

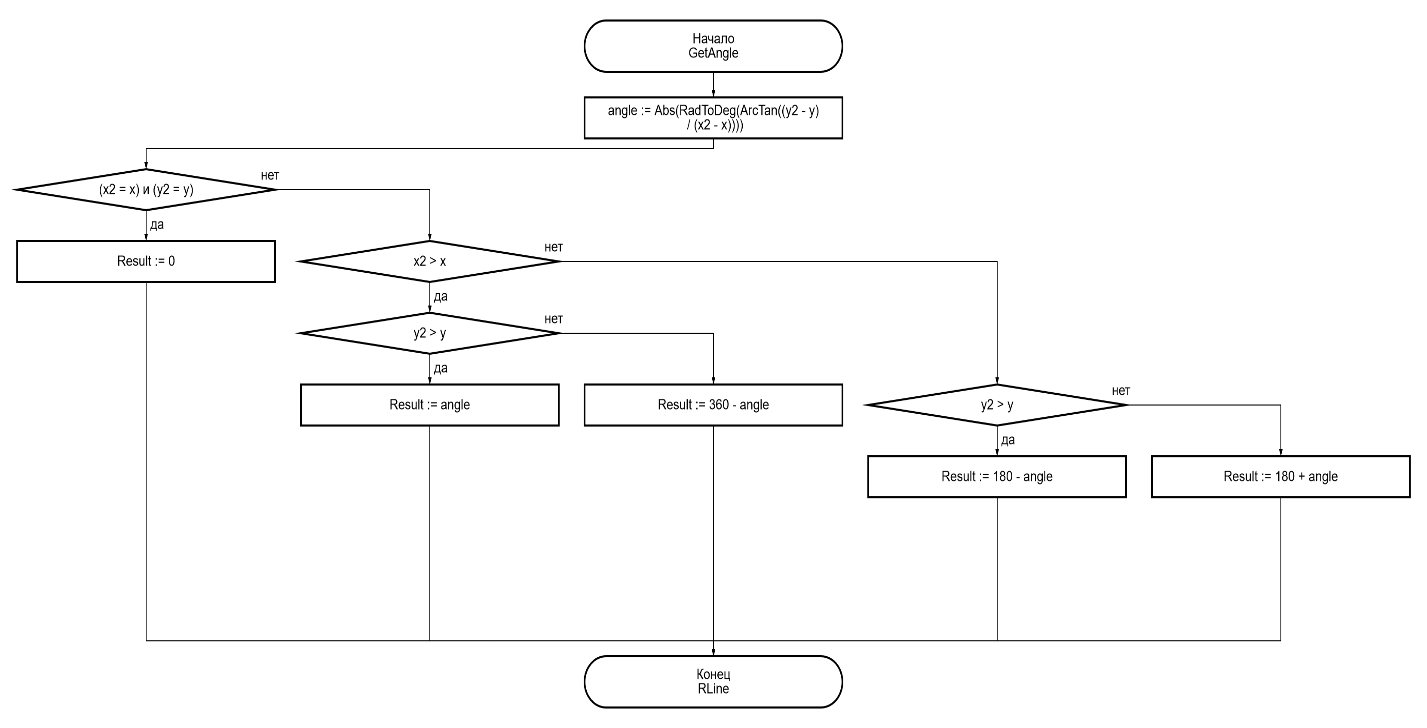


Рис. 2 «Схема алгоритма функции GetAngle»

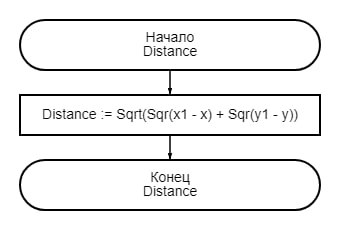
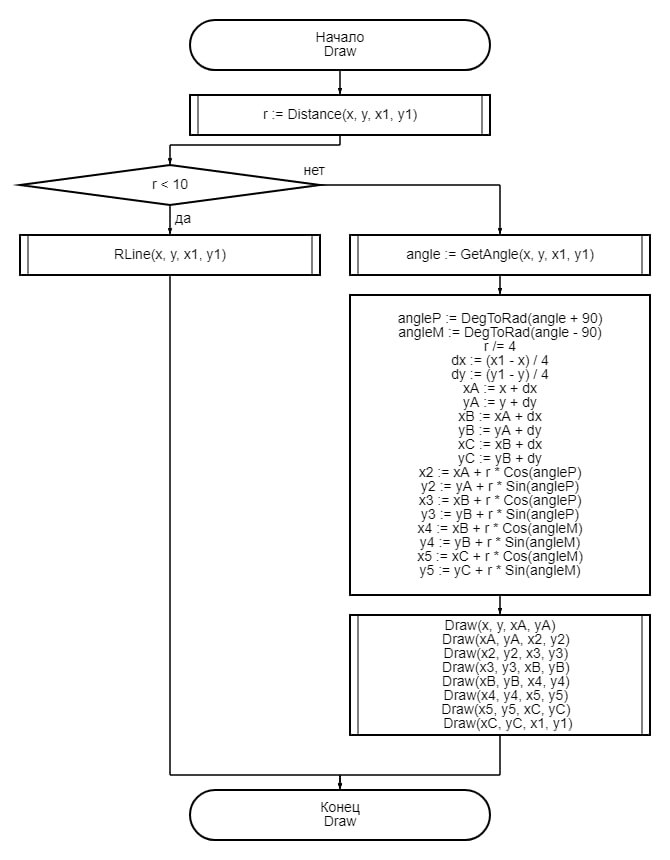


Рис. 3 «Схема алгоритма функции Distance»

  
Рис. 4 «Схема алгоритма процедуры Draw»

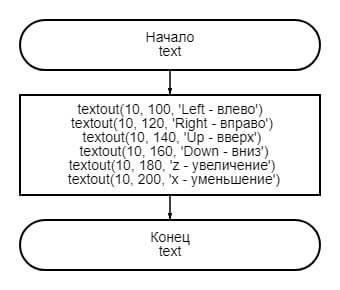
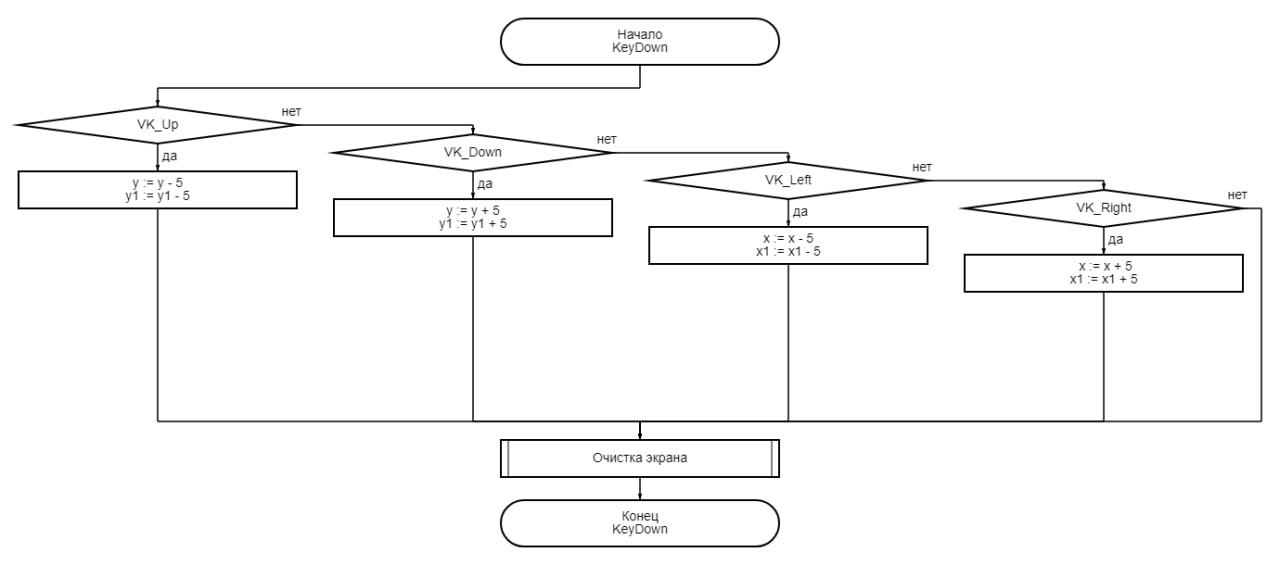
****

Рис. 5 «Схема алгоритма процедуры text» ****Рис. 6 «Схема алгоритма процедуры KeyDown»

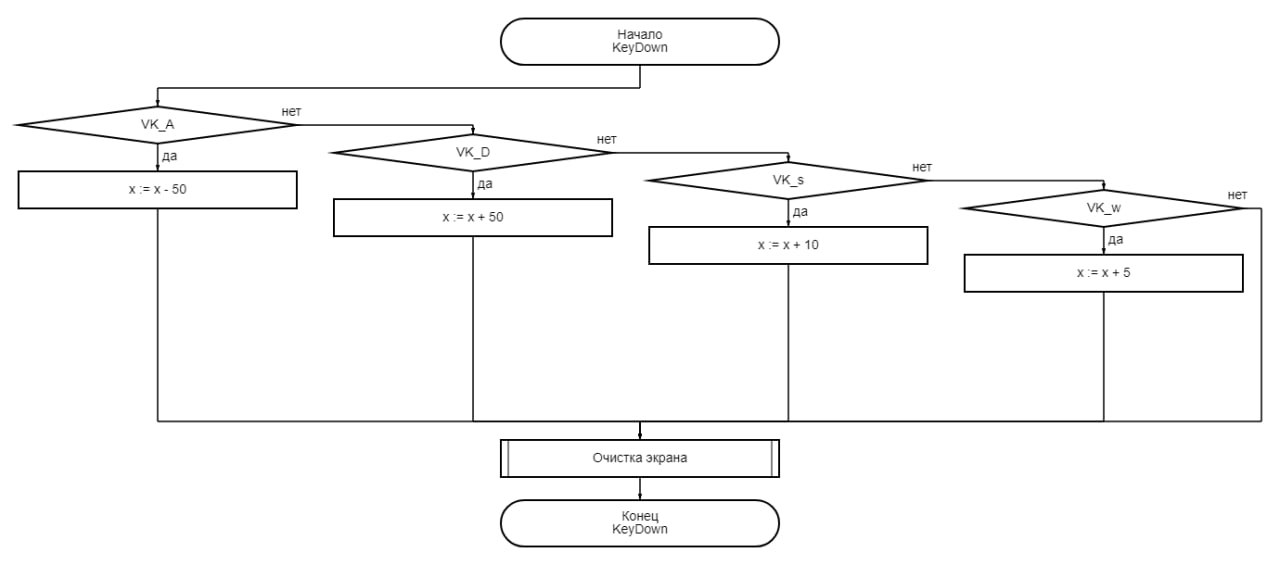


Рис. 7 «Схема алгоритма процедуры KeyUp»

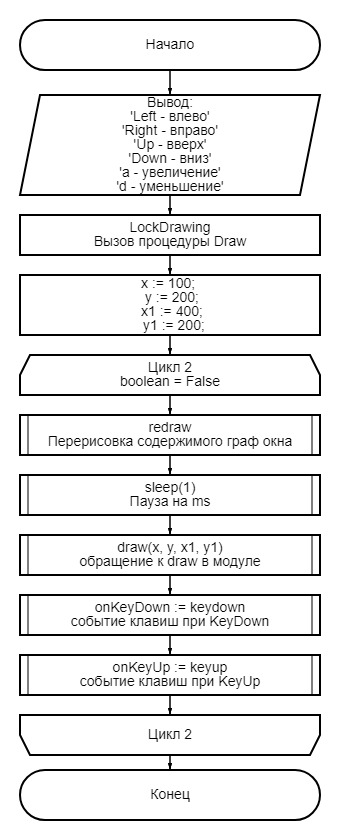
****

Рис. 8 «Схема алгоритма тела»

**Код программы**

**USES** Modulio;

**uses** GraphABC;

**var**

x, y, x1, y1, k: integer;

**procedure** text; //отвечает за вывод подсказок на экран

**begin**

// Вывод подсказок на экран

textout(10, 100, 'Left - влево');

textout(10, 120, 'Right - вправо');

textout(10, 140, 'Up - вверх');

textout(10, 160, 'Down - вниз');

textout(10, 180, 'a - увеличение');

textout(10, 200, 'd - уменьшение');

**end**;

**procedure** KeyDown(key: integer);//обрабатывает нажатия клавиш на клавиатуре для перемещения рисунка в соответствующем направлении

**begin**

**case** key **of**

VK\_Up:

**begin**

y := y - 5;

y1 := y1 - 5;

**end**;

VK\_Down:

**begin**

y := y + 5;

y1 := y1 + 5;

**end**;

VK\_Left:

**begin**

x := x - 5;

x1 := x1 - 5;

**end**;

VK\_Right:

**begin**

x := x + 5;

x1 := x1 + 5;

**end**;

**end**;

Window.Clear;

**end**;

**procedure** KeyUp(a: integer);//станавливает глубину отрисовки фрактала в зависимости от нажатой клавиши

**begin**

**case** a **of**

VK\_A:

**begin**

x := x - 50;

**end**;

Vk\_D:

**begin**

x := x + 50;

**end**;

vk\_s:

**begin**

x := x + 10;

**end**;

vk\_w:

**begin**

x := x + 5;

**end**;

**end**;

Window.Clear;

**end**;

**begin**

// Вывод подсказок на экран

writeln('Left - влево');

writeln('Right - вправо');

writeln('Up - вверх');

writeln('Down - вниз');

writeln('a - увеличение');

writeln('d - уменьшение');

LockDrawing; //вызывается процедура Draw для его отрисовки, а также устанавливаются обработчики событий клавиш для управления фракталом

x := 100;

y := 200;

x1 := 400;

y1 := 200;

**repeat**

redraw;

sleep(1);

draw(x, y, x1, y1);

onKeyDown := keydown;

onKeyUp := keyup;

**until**(False);

**end**.

**Модуль**

**UNIT** Modulio;

**Uses** GraphABC;

**procedure** RLine(x, y, x1, y1: real) := Line(Round(x), Round(y), Round(x1), Round(y1)); //нарисовать линию между округленными координатами точек // параметризация

**function** GetAngle(x, y, x2, y2: real): real; //вычисляет угол наклона отрезка, соединяющего точки (x, y) и (x2, y2), относительно оси X // параметризация

**begin**

**var** angle := Abs(RadToDeg(ArcTan((y2 - y) / (x2 - x)))); //декомпозиция

**if** (x2 = x) **and** (y2 = y) **then**

Result := 0

**else**

**if** x2 > x **then**

**if** y2 > y **then** Result := angle **else** Result := 360 - angle

**else**

**if** y2 > y **then** Result := 180 - angle **else** Result := 180 + angle;

**end**;

**function** Distance(x, y, x1, y1: real) := Sqrt(Sqr(x1 - x) + Sqr(y1 - y)); //вычисляет расстояние между двумя точками (x, y) и (x1, y1) в декартовой системе координат

**procedure** Draw(x, y, x1, y1: real); //реализует построение фрактала Кривой Минковского //параметризация

**begin**

**var** r := Distance(x, y, x1, y1); //ВЫЧИСЛЕНИЕ ДЛИНЫ ОТРЕЗКА МЕЖДУ ТОЧКАМИ

**if** r < 10 **then** //база рекурсии

RLine(x, y, x1, y1) //ЕСЛИ R<10 ТО ВЫЗЫВАЕМ RLINE И РИСУЕМ ПРЯМУЮ

**else**

**begin**

**var** angle := GetAngle(x, y, x1, y1); // Вычисляем угол наклона отрезка между точками (x, y) и (x1, y1)

**var** angleP := DegToRad(angle + 90); //декомпозиция

**var** angleM := DegToRad(angle - 90); //декомпозиция

r /= 4;//Разбиваем текущий отрезок на четыре части

**var** dx := (x1 - x) / 4; //декомпозиция

**var** dy := (y1 - y) / 4; //декомпозиция

**var** xA := x + dx; //декомпозиция //вычисляем координаты промежуточных точек A

**var** yA := y + dy; //декомпозиция //вычисляем координаты промежуточных точек A

**var** xB := xA + dx; //декомпозиция //вычисляем координаты промежуточных точек B

**var** yB := yA + dy; //декомпозиция //вычисляем координаты промежуточных точек B

**var** xC := xB + dx; //декомпозиция //вычисляем координаты промежуточных точек C

**var** yC := yB + dy; //декомпозиция //вычисляем координаты промежуточных точек C

//Вычисляем координаты точек x2, y2, x3, y3, x4, y4, x5, y5, используя углы angleP и angleM.

**var** x2 := xA + r \* Cos(angleP); //декомпозиция

**var** y2 := yA + r \* Sin(angleP); //декомпозиция

**var** x3 := xB + r \* Cos(angleP); //декомпозиция

**var** y3 := yB + r \* Sin(angleP); //декомпозиция

**var** x4 := xB + r \* Cos(angleM); //декомпозиция

**var** y4 := yB + r \* Sin(angleM); //декомпозиция

**var** x5 := xC + r \* Cos(angleM); //декомпозиция

**var** y5 := yC + r \* Sin(angleM); //декомпозиция

//Рекурсивный вызов процедуры Draw для каждого отрезка

Draw(x, y, xA, yA);

Draw(xA, yA, x2, y2);

Draw(x2, y2, x3, y3);

Draw(x3, y3, xB, yB);

Draw(xB, yB, x4, y4);

Draw(x4, y4, x5, y5);

Draw(x5, y5, xC, yC);

Draw(xC, yC, x1, y1);

**end**;

**end**;

**begin**

**end**.

**Результат выполнения программы**

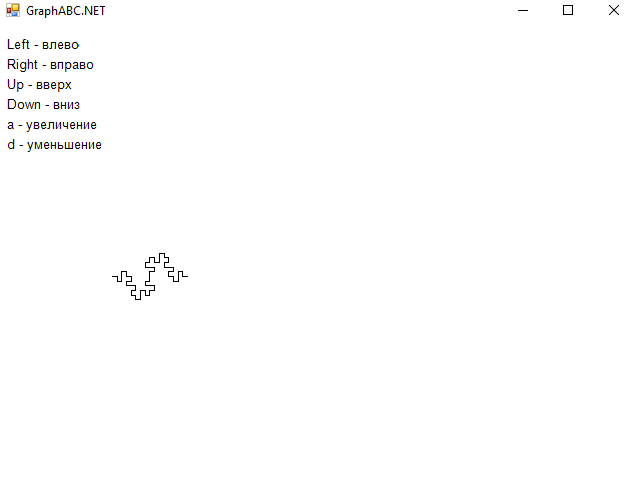


Рис. 9 «Результат выполнения программы»

**Вывод**

В процессе выполнения данной работы мы успешно применили изученный материал лабораторных занятий на практике, включая процедуры, условные операторы, циклы создание переменных и операторов ввода-вывода.

Благодаря разработанным программам мы усовершенствовали навыки реализации алгоритмов с применением рекурсивных вычислений. Мы изучили различные виды фракталов и научились создавать их с помощью рекурсии. Также мы освежили знания по работе с GraphABC и научились разделять блоки программы на отдельные модули для упрощения чтения и оптимизации кода. Мы научились использовать виртуальные клавиши для управления фигурами и добавили функционал изменения масштаба и глубины прорисовки изображения. Эти навыки будут полезны нам при работе с GraphABC и управлении клавишами в будущем.