Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Вятский государственный университет»

Колледж ВятГУ

**ОТЧЕТ**

**ПО ДОМАШНЕЙ КОНТРОЛЬНОЙ РАБОТЕ №1**

**«Исследование фракталов»**

**ПО ДИСЦИПЛИНЕ**

**«Разработка кода информационных систем»**

Выполнил: студент учебной группы

ИСПк-202-52-00

Гончаров Мирослав Даниилович

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2024

**Цель работы:** получение навыков реализации алгоритмов с рекурсивными вычислениями, знакомство с фракталами.

**Задание**

1. Написать программу для визуализации фрактала "Снежинка Коха".

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

3. Построение множества ломанных, образующих фрактал, должно осуществляться в отдельном модуле.

**Описание алгоритма.**

1. Задаются начальные координаты для создания равностороннего треугольника, который служит основой для снежинки Коха. Масштаб (длина стороны треугольника) определяет размер снежинки.
2. Вычисление координат трех точек, формирующих равносторонний треугольник. Используется геометрические соотношения для определения положения каждой вершины относительно начальной точки.
3. Для каждой стороны треугольника применяется процедура, которая делит линию на три равные части и заменяет среднюю часть двумя сторонами равностороннего треугольника, создавая уникальный узор снежинки Коха. Этот процесс повторяется рекурсивно для каждой новой линии до достижения заданной глубины рекурсии.
4. На каждом уровне рекурсии каждый отрезок линии заменяется на четыре новых отрезка, следуя правилам построения снежинки Коха: один отрезок сохраняется как есть, два формируют стороны нового треугольника, и четвертый отрезок замыкает фигуру, возвращаясь к исходной линии.
5. После достижения заданной глубины рекурсии процедура рисования завершается. Результатом является фрактальная фигура снежинки Коха, состоящая из множества повторяющихся элементов, каждый из которых уменьшается в размере с увеличением глубины рекурсии.
6. Предоставляется возможность динамически изменять положение снежинки и ее масштаб с помощью клавиш управления, что позволяет пользователю исследовать различные вариации фигуры.

**Схема алгоритма**

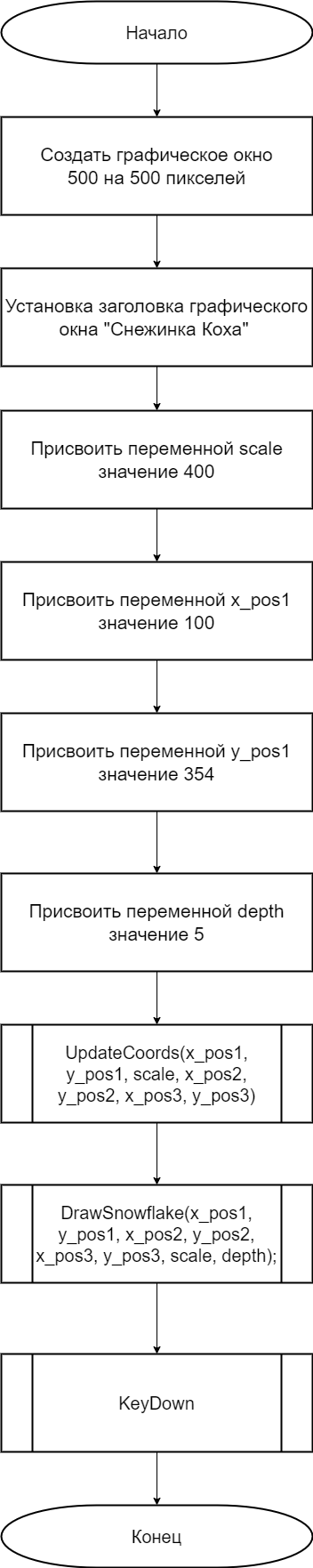


Рисунок 1 — Схема алгоритма программы



Рисунок 2 — Схема алгоритма программы

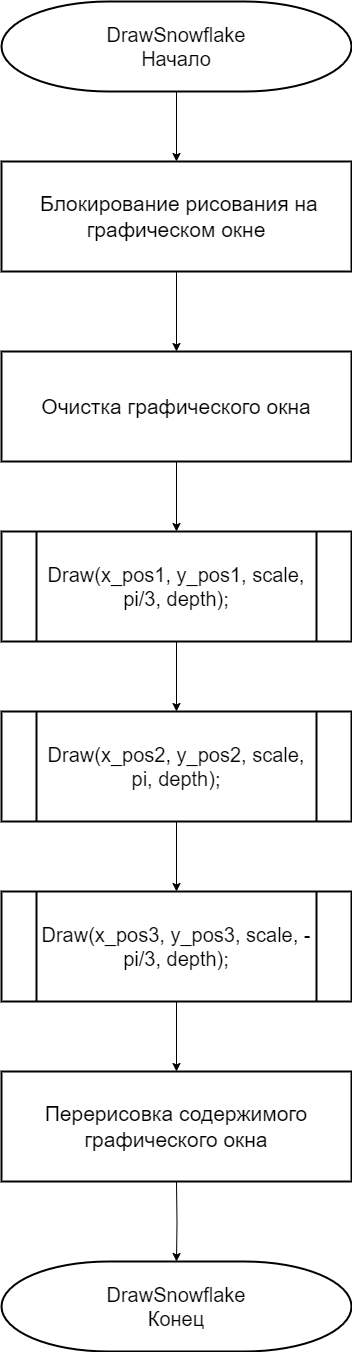


Рисунок 3 — Схема алгоритма программы

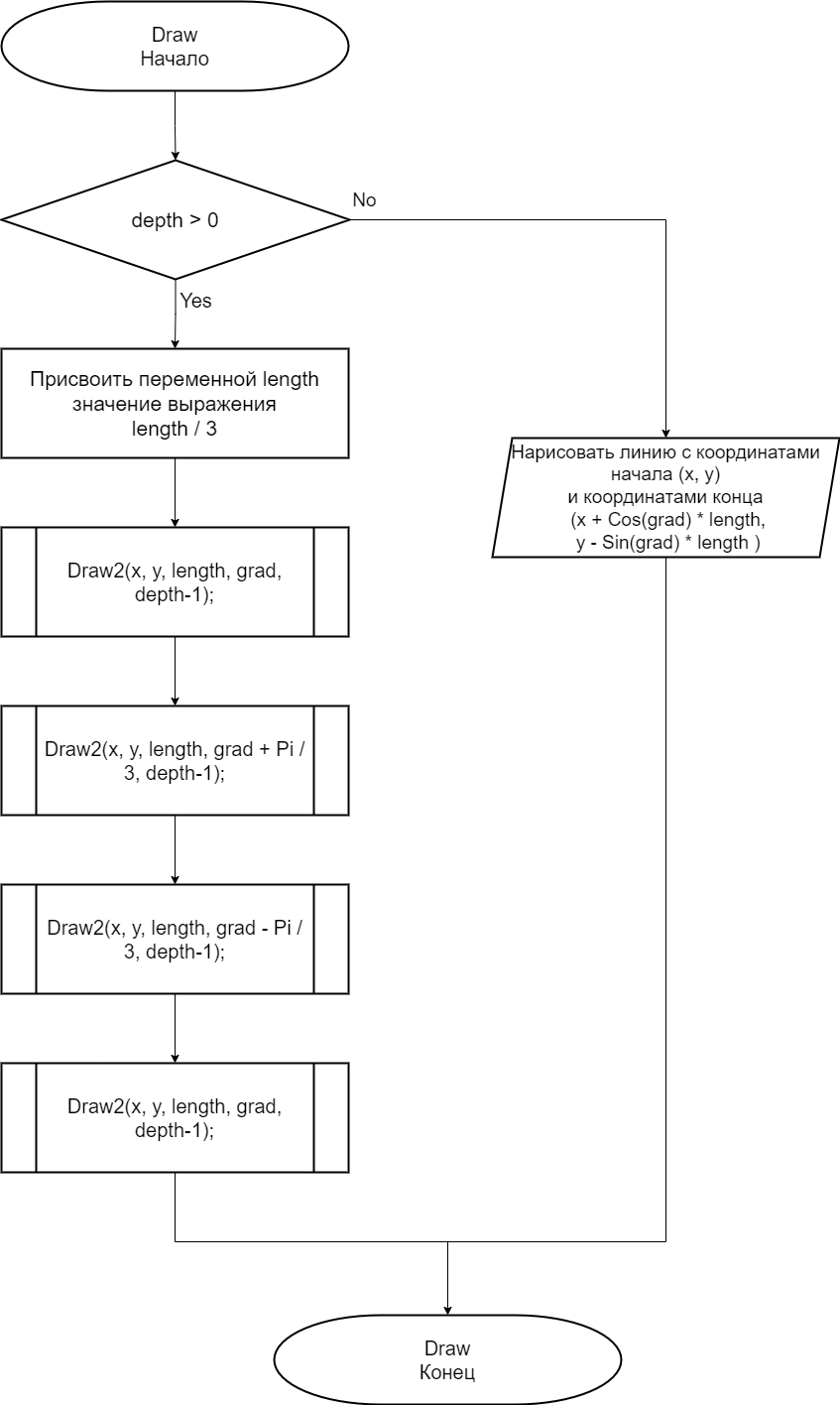


Рисунок 4 — Схема алгоритма программы



Рисунок 5 — Схема алгоритма программы

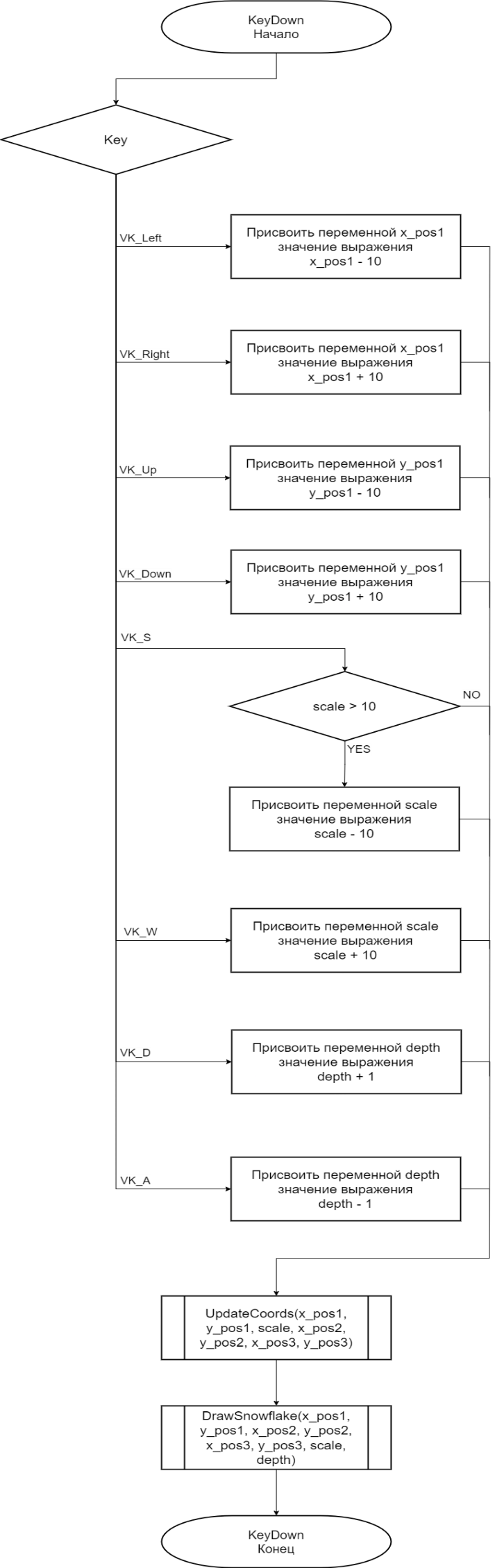


Рисунок 6 — Схема алгоритма программы

**Код программы**

**uses**

GraphABC, KochSnowflakeUnit;

**var**

x\_pos1, y\_pos1, x\_pos2, y\_pos2, x\_pos3, y\_pos3, scale: Real;

depth: Integer;

**procedure** KeyDown(Key: Integer);

**begin**

**case** Key **of**

VK\_Left: x\_pos1 := x\_pos1 - 10;

VK\_Right: x\_pos1 := x\_pos1 + 10;

VK\_Up: y\_pos1 := y\_pos1 - 10;

VK\_Down: y\_pos1 := y\_pos1 + 10;

VK\_S: **if** scale > 10 **then** scale := scale - 10;

VK\_W: scale := scale + 10;

VK\_D: depth := depth + 1;

VK\_A: depth := depth - 1;

**end**;

UpdateCoords(x\_pos1, y\_pos1, scale, x\_pos2, y\_pos2, x\_pos3, y\_pos3);

DrawSnowflake(x\_pos1, y\_pos1, x\_pos2, y\_pos2, x\_pos3, y\_pos3, scale, depth);

**end**;

**begin**

SetWindowSize(500,500);

SetWindowCaption('Снежинка Коха');

scale := 400;

x\_pos1 := 100;

y\_pos1 := 354;

depth := 5;

UpdateCoords(x\_pos1, y\_pos1, scale, x\_pos2, y\_pos2, x\_pos3, y\_pos3);

DrawSnowflake(x\_pos1, y\_pos1, x\_pos2, y\_pos2, x\_pos3, y\_pos3, scale, depth);

OnKeyDown := KeyDown;

**end**.

**unit** KochSnowflakeUnit;

**interface**

**uses** GraphABC;

**procedure** DrawSnowflake(x\_pos1, y\_pos1, x\_pos2, y\_pos2, x\_pos3, y\_pos3, scale: Real; depth: Integer);

**procedure** UpdateCoords(**var** x\_pos1, y\_pos1: Real; scale: Real; **var** x\_pos2, y\_pos2, x\_pos3, y\_pos3: Real);

**implementation**

**procedure** Draw(x, y, length, grad: Real; t: Integer); **forward**;

**procedure** Draw2(**var** x, y: Real; l, grad: Real; t: Integer); **forward**;

**procedure** Draw(x, y, length, grad: Real; t: Integer);

**begin**

**if** t > 0 **then**

**begin**

length := length / 3;

Draw2(x, y, length, grad, t-1);

Draw2(x, y, length, grad + Pi / 3, t-1);

Draw2(x, y, length, grad - Pi / 3, t-1);

Draw2(x, y, length, grad, t-1);

**end**

**else**

Line(Round(x), Round(y), Round(x + Cos(grad) \* length), Round(y - Sin(grad) \* length));

**end**;

**procedure** Draw2(**var** x, y: Real; l, grad: Real; t: Integer);

**begin**

Draw(x, y, l, grad, t);

x := x + l \* Cos(grad);

y := y - l \* Sin(grad);

**end**;

**procedure** UpdateCoords(**var** x\_pos1, y\_pos1: Real; scale: Real; **var** x\_pos2, y\_pos2, x\_pos3, y\_pos3: Real);

**var**

katet: Real;

**begin**

x\_pos2 := x\_pos1 + scale;

katet := scale / 2;

x\_pos3 := x\_pos1 + katet;

y\_pos2 := y\_pos1;

y\_pos3 := y\_pos1 - katet \* Power(3, 1/2);

**end**;

**procedure** DrawSnowflake(x\_pos1, y\_pos1, x\_pos2, y\_pos2, x\_pos3, y\_pos3, scale: Real; depth: Integer);

**begin**

LockDrawing;

ClearWindow;

Draw(x\_pos1, y\_pos1, scale, pi/3, depth);

Draw(x\_pos2, y\_pos2, scale, pi, depth);

Draw(x\_pos3, y\_pos3, scale, -pi/3, depth);

Redraw;

**end**;

**end**.

**Результаты выполнения программыИзображение выглядит как снимок экрана, шаблон

Автоматически созданное описание**

Рисунок 7 — Результат выполнения программы

Изображение выглядит как текст, снимок экрана, зарисовка, диаграмма

Автоматически созданное описание

Рисунок 8 — Результат выполнения программы

Изображение выглядит как карта, текст, диаграмма

Автоматически созданное описание

Рисунок 9 — Результат выполнения программы

**Вывод**

В процессе выполнения данной лабораторной работы была достигнута основная цель: получение навыков реализации алгоритмов с рекурсивными вычислениями, а также знакомство с фракталами на примере визуализации фрактала "Снежинка Коха". Работа включала в себя разработку программы на языке Pascal, которая позволяла не только визуализировать упомянутый фрактал, но и предоставляла пользователю возможности для масштабирования, изменения глубины прорисовки и перемещения фигуры. Организация основных алгоритмов построения фрактала была реализована в отдельном модуле, что способствовало лучшей структуризации кода и его повторному использованию.

В процессе работы были преодолены различные технические сложности, связанные с реализацией рекурсивных алгоритмов и управлением графическим интерфейсом в среде Pascal. Особое внимание было уделено точности вычислений координат точек фрактала и оптимизации производительности программы для обеспечения плавного масштабирования и перемещения. Результаты выполнения программы демонстрируют успешную реализацию поставленных задач.