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 «МДК.05.02»**

Выполнил: студент учебной группы

ИСПк-203-52-00

Четвериков Денис Альбертович

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2025

**Цель работы:**

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

**Формулировка задания:**

* Написать программу для визуализации фрактала «Кривая Гильберта».
* Предусмотреть возможность масштабирования, изменения глубины прорисовки и перемещения полученной фигуры.
* Построение множества ломанных, образующих фрактал, должно осуществляться в отдельном модуле.

**Описание алгоритма и ответы на вопросы:**

Алгоритм программы

1. Инициализация окна:

- Устанавливается размер окна (`SetWindowSize(425, 500)`).

- Задается заголовок окна (`SetWindowCaption('Фракталы: Снежинка Коха')`).

2. Основная процедура `Draw`:

- Эта процедура рекурсивно рисует кривую Коха.

- Параметры:

- `x, y`: текущие координаты начала отрезка.

- `l`: длина текущего отрезка.

- `u`: угол наклона текущего отрезка.

- `t`: глубина рекурсии (количество итераций).

3. Рекурсивное построение кривой Коха:

- Если глубина рекурсии `t > 0`, то:

- Длина отрезка уменьшается в 3 раза (`l := l / 3`).

- Вызывается вспомогательная процедура `Draw2` четыре раза:

1. Прямой отрезок (`Draw2(x, y, l, u, t-1)`).

2. Отклонение на 60 градусов влево (`Draw2(x, y, l, u + pi/3, t-1)`).

3. Отклонение на 60 градусов вправо (`Draw2(x, y, l, u - pi/3, t-1)`).

4. Прямой отрезок (`Draw2(x, y, l, u, t-1)`).

- Если глубина рекурсии `t = 0`, то рисуется отрезок:

- Используется функция `Line`, которая рисует линию от текущей точки `(x, y)` до точки `(x + cos(u) \* l, y - sin(u) \* l)`.

4. Вспомогательная процедура `Draw2`:

- Эта процедура вызывает `Draw` для построения следующего отрезка.

- После вызова `Draw` обновляет координаты `x` и `y`:

- `x := x + l \* cos(u)`.

- `y := y - l \* sin(u)`.

- Это позволяет продолжить построение кривой из новой точки.

5. Построение снежинки Коха:

- Снежинка Коха состоит из трех кривых Коха, соединенных в форме треугольника.

- Программа вызывает процедуру `Draw` три раза с разными начальными углами:

- `Draw(10, 354, 400, pi/3, 4)`: первая сторона (угол 60 градусов).

- `Draw(410, 354, 400, pi, 4)`: вторая сторона (угол 180 градусов).

- `Draw(210, 8, 400, -pi/3, 4)`: третья сторона (угол -60 градусов).

6. Ожидание завершения программы:

- Программа ожидает нажатия любой клавиши (`Repeat Until KeyPressed`), чтобы окно не закрывалось сразу после отрисовки.

**Схемы алгоритмов:**

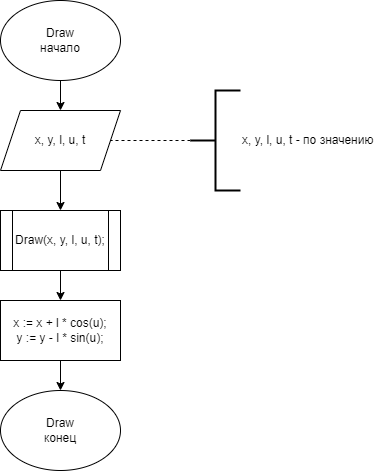


Рисунок 1 – схема алгоритма процедуры Draw в модуле KochModule

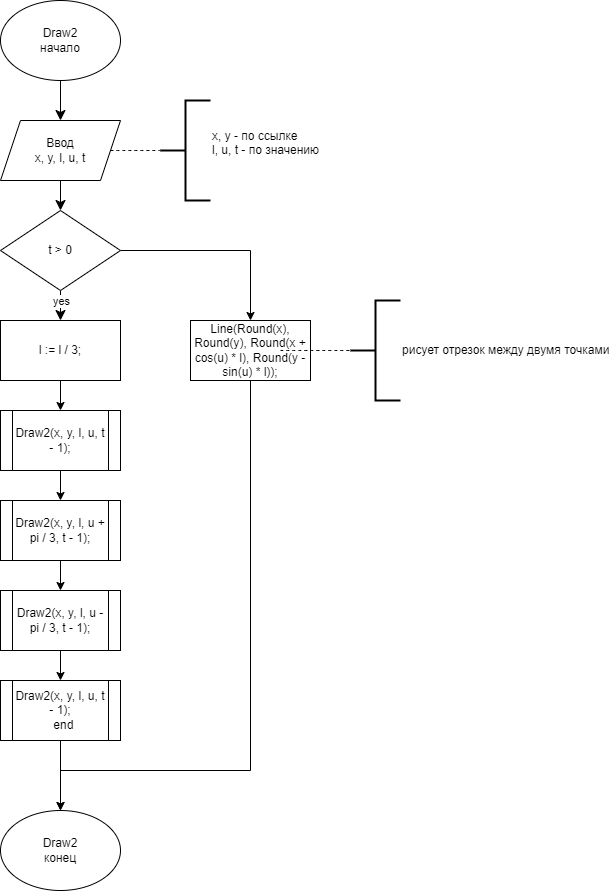


Рисунок 2 – схема алгоритма процедуры Draw2 в модуле KochModule

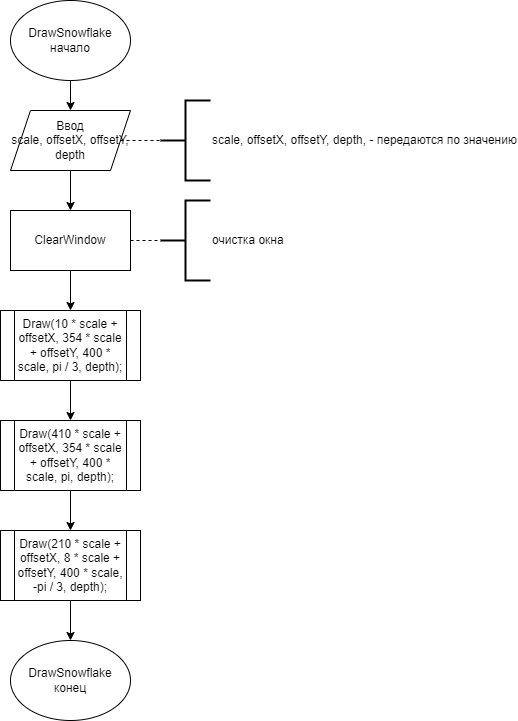


Рисунок 3 – схема алгоритма процедуры DrawSnowflake основной программы

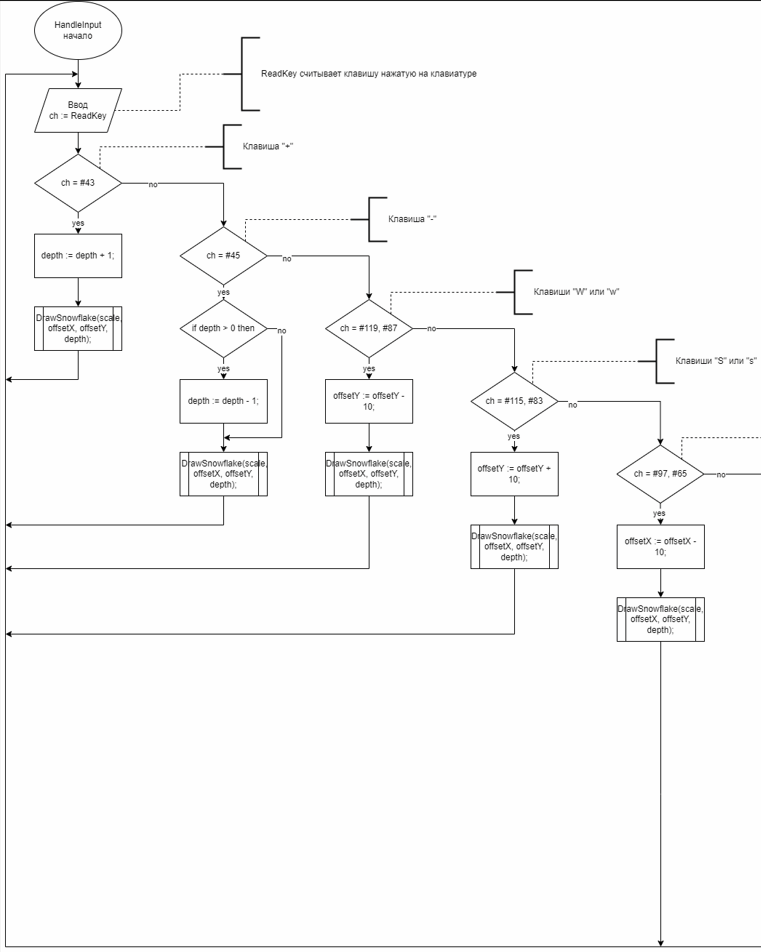


Рисунок 4.1 – схема алгоритма процедуры HandleInput

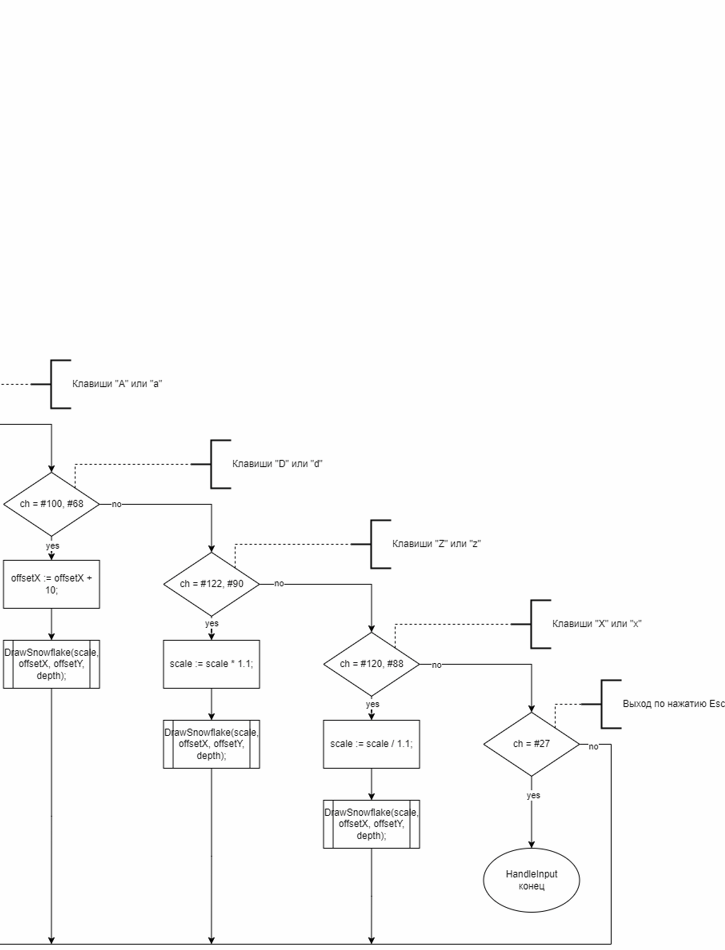


Рисунок 4.2 – схема алгоритма процедуры HandleInput

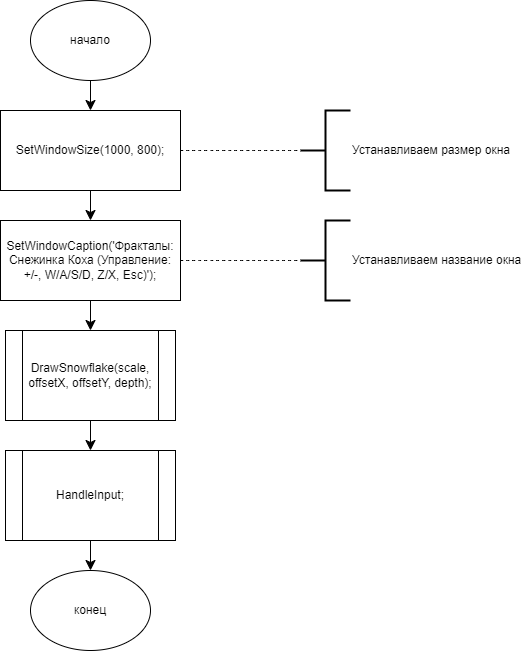


Рисунок 5 – схема алгоритма основной программы

**Код программы:**

**uses** CRT, GraphABC, KochModule;

**var**

scale: Real = 1.0; // Масштаб

depth: Integer = 6; // Глубина прорисовки

offsetX: Integer = 1;

offsetY: Integer = 1; // Смещение фигуры

**procedure** HandleInput;

**var**

ch: Char;

**begin**

**repeat**

ch := ReadKey;

**case** ch **of**

#43: // Клавиша "+"

**begin**

depth := depth + 1; // Увеличиваем глубину

DrawSnowflake(scale, offsetX, offsetY, depth);

**end**;

#45: // Клавиша "-"

**begin**

**if** depth > 0 **then**

depth := depth - 1; // Уменьшаем глубину

DrawSnowflake(scale, offsetX, offsetY, depth);

**end**;

#119, #87: // Клавиши "W" или "w"

**begin**

offsetY := offsetY - 10; // Перемещение вверх

DrawSnowflake(scale, offsetX, offsetY, depth);

**end**;

#115, #83: // Клавиши "S" или "s"

**begin**

offsetY := offsetY + 10; // Перемещение вниз

DrawSnowflake(scale, offsetX, offsetY, depth);

**end**;

#97, #65: // Клавиши "A" или "a"

**begin**

offsetX := offsetX - 10; // Перемещение влево

DrawSnowflake(scale, offsetX, offsetY, depth);

**end**;

#100, #68: // Клавиши "D" или "d"

**begin**

offsetX := offsetX + 10; // Перемещение вправо

DrawSnowflake(scale, offsetX, offsetY, depth);

**end**;

#122, #90: // Клавиши "Z" или "z"

**begin**

scale := scale \* 1.1; // Увеличение масштаба

DrawSnowflake(scale, offsetX, offsetY, depth);

**end**;

#120, #88: // Клавиши "X" или "x"

**begin**

scale := scale / 1.1; // Уменьшение масштаба

DrawSnowflake(scale, offsetX, offsetY, depth);

**end**;

**end**;

**until** ch = #27; // Выход по нажатию Esc

**end**;

**begin**

SetWindowSize(1000, 800); // Устанавливаем размер окна

SetWindowCaption('Фракталы: Снежинка Коха (Управление: +/-, W/A/S/D, Z/X, Esc)');

DrawSnowflake(scale, offsetX, offsetY, depth); // Первоначальная отрисовка

HandleInput; // Обработка ввода

**end**.

**unit** KochModule;

**interface**

**uses** GraphABC;

**procedure** DrawSnowflake(scale: Real; offsetX, offsetY: Integer; depth: Integer);

**implementation**

**procedure** Draw(x, y, l, u: Real; t: Integer);

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

**begin**

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

x := x + l \* cos(u);

y := y - l \* sin(u);

**end**;

**begin**

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

**begin**

l := l / 3;

Draw2(x, y, l, u, t - 1);

Draw2(x, y, l, u + pi / 3, t - 1);

Draw2(x, y, l, u - pi / 3, t - 1);

Draw2(x, y, l, u, t - 1);

**end**

**else**

Line(Round(x), Round(y), Round(x + cos(u) \* l), Round(y - sin(u) \* l));

**end**;

**procedure** DrawSnowflake(scale: Real; offsetX, offsetY: Integer; depth: Integer);

**begin**

ClearWindow; // Очистка окна перед перерисовкой

// Рисуем три стороны снежинки Коха

Draw(10 \* scale + offsetX, 354 \* scale + offsetY, 400 \* scale, pi / 3, depth);

Draw(410 \* scale + offsetX, 354 \* scale + offsetY, 400 \* scale, pi, depth);

Draw(210 \* scale + offsetX, 8 \* scale + offsetY, 400 \* scale, -pi / 3, depth);

**end**;

**end**.

**Результат выполнения программы:**

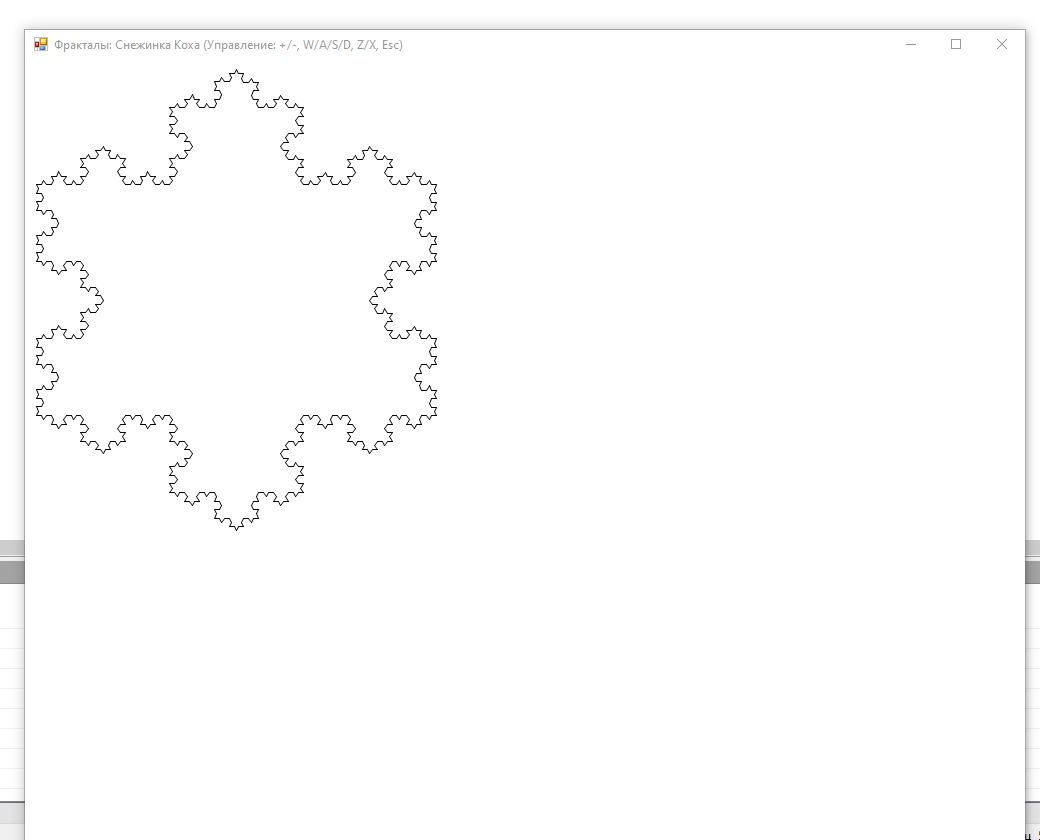


Рисунок 6 – результат выполнения работы одной из кнопок

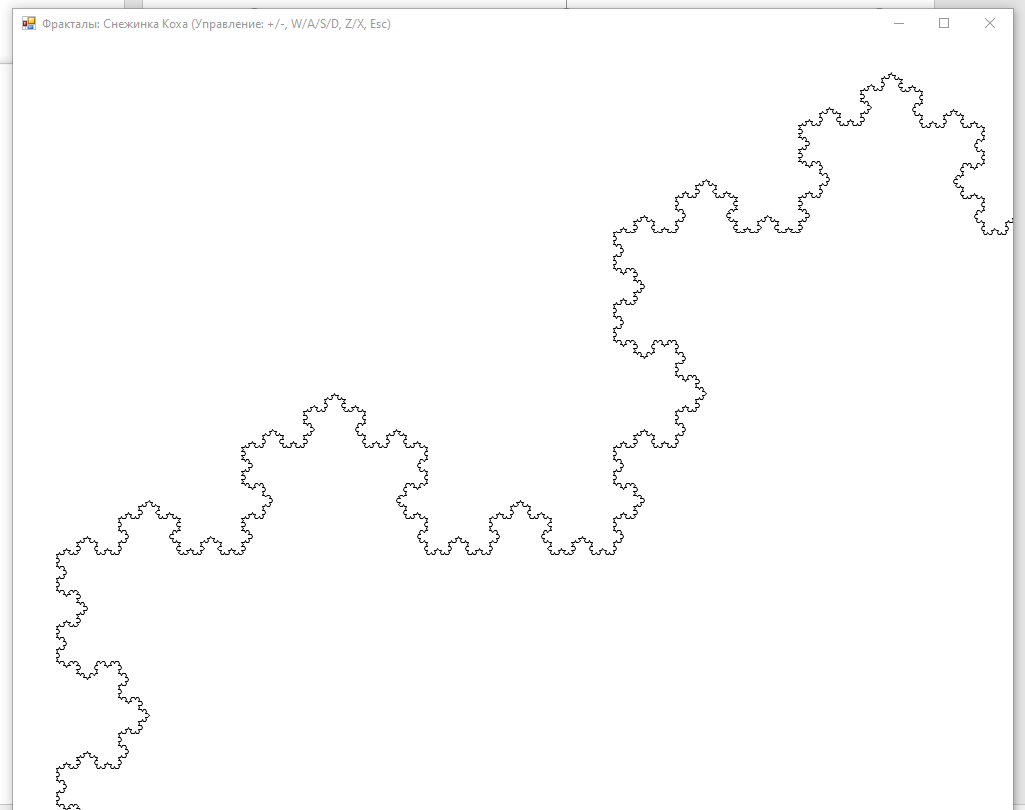


Рисунок 7 – результат выполнения одной из кнопок

**Вывод:**

В ходе выполнения данной работы был изучен алгоритм построения фрактала снежинка Коха, способы реализации масштабирования, а также создание отдельных модулей. Работа показалась интересной, были приобретены более глубокое понимание графического модуля в паскале, а также его использование при построении фракталов.