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\_\_\_\_\_»**

Выполнил: студент учебной группы

ИСПк-202-52-00

Жидкова Светлана Дмитриевна

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2024

1. Цель работы: получение навыков реализации алгоритмов с рекурсивными вычислениями, знакомство с фракталами.
2. Задание: **6 вариант**
3. Написать программу для визуализации фрактала «Кривая Хартера-Хейтуэя»
4. Предусмотреть возможности масштабирования, изменения глубины прорисовки и перемещения полученной фигуры.
5. Построение множества ломанных, образующих фрактал, должно осуществляться в отдельном модуле.
6. Описание алгоритма

1. Инициализация начальных параметров и установка размера графического окна.

2. Создание массива для определения направления ломаных.

3. Настройка начальных значений переменных.

4. Начало цикла построения фрактала, который выполняется 128 раз.

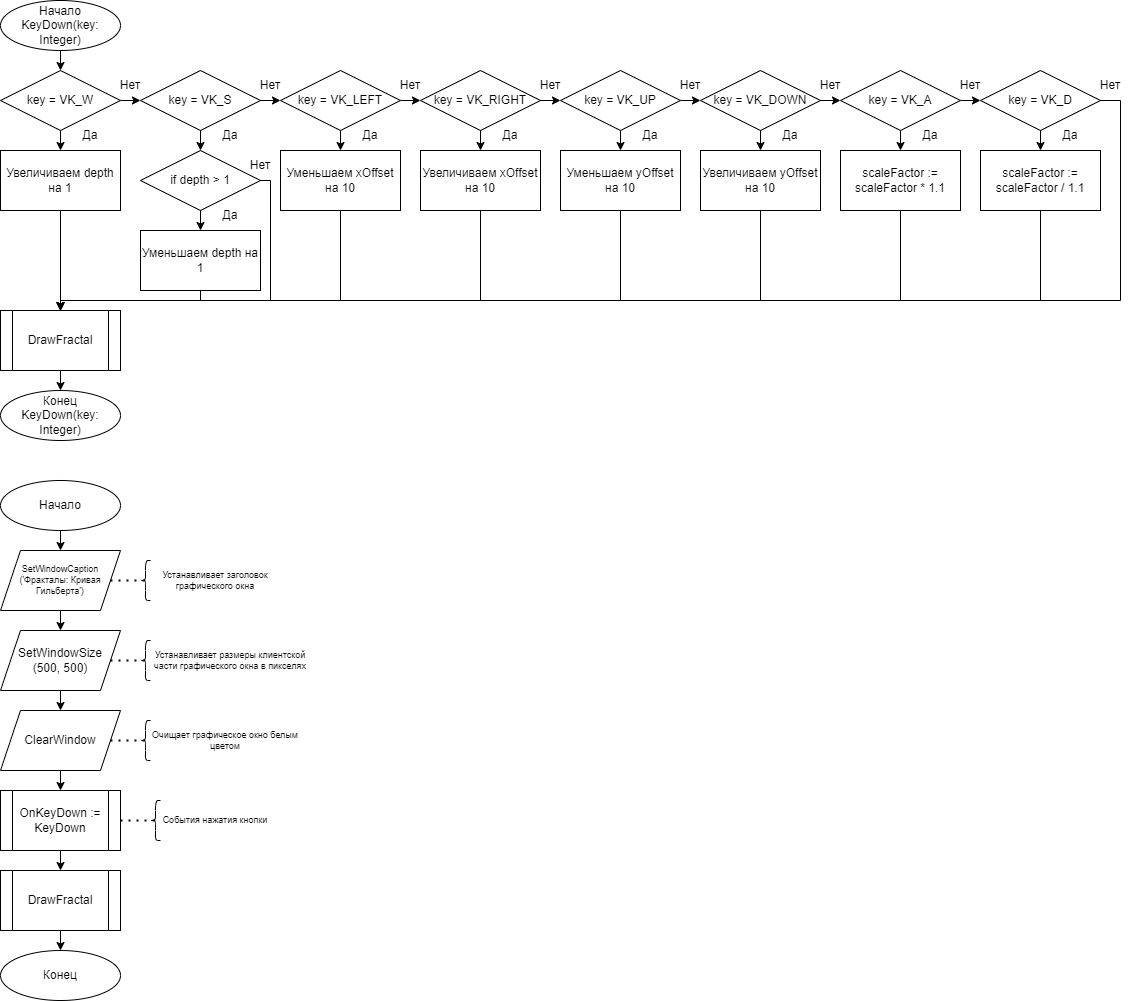
5. Вложенный цикл проводит отрисовку линий в заданном направлении.

6. Обработка клавиш клавиатуры для управления фракталом.

7. Очистка окна, вызов процедуры отрисовки фрактала и обновление изображения при изменениях.

8. Завершение программы с установкой обработчика клавиатуры и начальной отрисовкой фрактала.

1. Схема алгоритма с комментариями



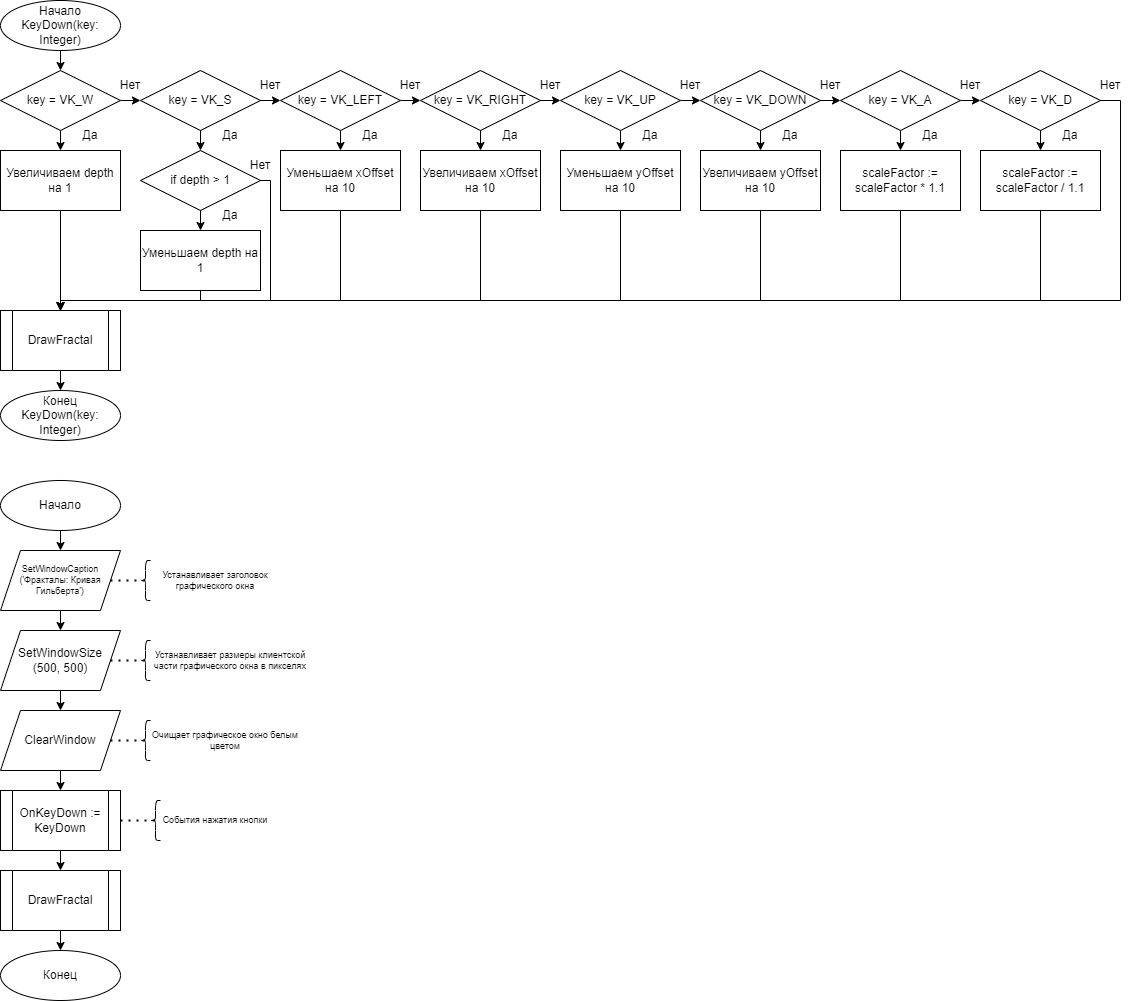


Рисунок 1 – алгоритм основной программы

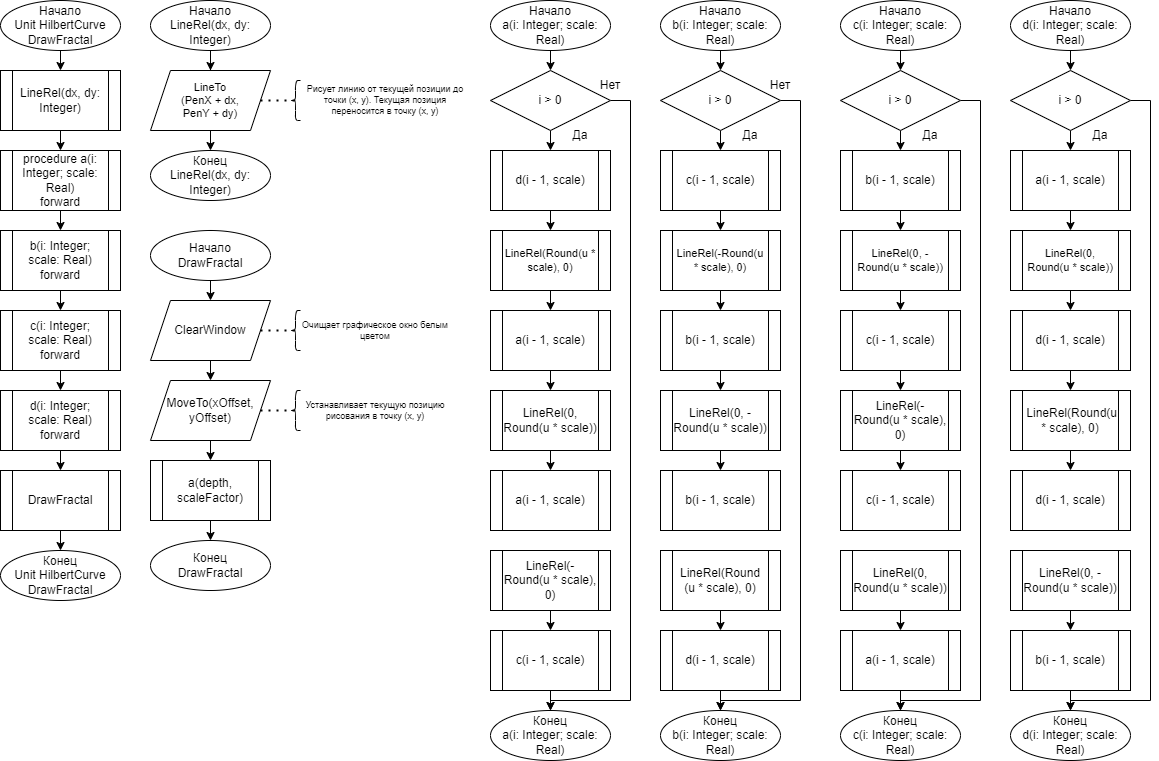


Рисунок 2 – алгоритм процедуры DrawDragonFractal

1. Код программы

**program** Dragon;

**uses**

GraphABC, DragonFractal;

**var**

x, y, dx, dy, i: integer;

scaleFactor: Double = 1.0; // начальный масштабный коэффициент

**procedure** KeyDown(key: integer);

**begin**

**case** key **of**

VK\_Up:

**begin**

y := y - 50;

**end**;

VK\_Down:

**begin**

y := y + 50;

**end**;

VK\_Left:

**begin**

x := x - 50;

**end**;

VK\_Right:

**begin**

x := x + 50;

**end**;

VK\_x:

**begin**

scaleFactor := scaleFactor \* 1.1; // увеличение масштаба на 10%

**end**;

VK\_z:

**begin**

scaleFactor := scaleFactor / 1.1; // уменьшение масштаба на 10%

**end**;

**end**;

Window.Clear;

DrawDragonFractal(x, y, dx, dy, scaleFactor);

redraw;

**end**;

**begin**

LockDrawing;

x := 200;

y := 140;

dx := 0;

dy := -4;

DrawDragonFractal(x, y, dx, dy, scaleFactor);

redraw;

onKeyDown += KeyDown;

**end**.

**unit** DragonFractal;

**interface**

**uses**

GraphABC;

**procedure** DrawDragonFractal(x, y, dx, dy: integer; scaleFactor: Double);

**implementation**

**procedure** DrawDragonFractal(x, y, dx, dy: integer; scaleFactor: Double);

**var**

turn: **array** [1..1000] **of** Boolean;

a, b, d, t: integer;

f: Boolean;

i: integer;

**begin**

f := true;

**for** a := 1 **to** 64 **do**

**begin**

turn[2 \* a - 1] := f;

f := **not** f;

turn[2 \* a] := turn[a];

**end**;

b := 0;

d := 1;

f := false;

MoveTo(Round(x \* scaleFactor), Round(y \* scaleFactor));

**for** a := 1 **to** 128 **do**

**begin**

**for** i := 1 **to** 127 \* 4 **do**

**begin**

b := b + d;

x := x + dx;

y := y + dy;

LineTo(Round(x \* scaleFactor), Round(y \* scaleFactor));

**if** f **and not** turn[b] **or not** f **and** turn[b] **then**

**begin**

t := dy;

dy := -dx;

**end**

**else**

**begin**

t := -dy;

dy := dx;

**end**;

dx := t;

**end**;

b := b + d;

d := -d;

f := **not** f;

x := x + dx;

y := y + dy;

LineTo(Round(x \* scaleFactor), Round(y \* scaleFactor));

**if** turn[a] **then**

**begin**

t := dy;

dy := -dx;

**end**

**else**

**begin**

t := -dy;

dy := dx;

**end**;

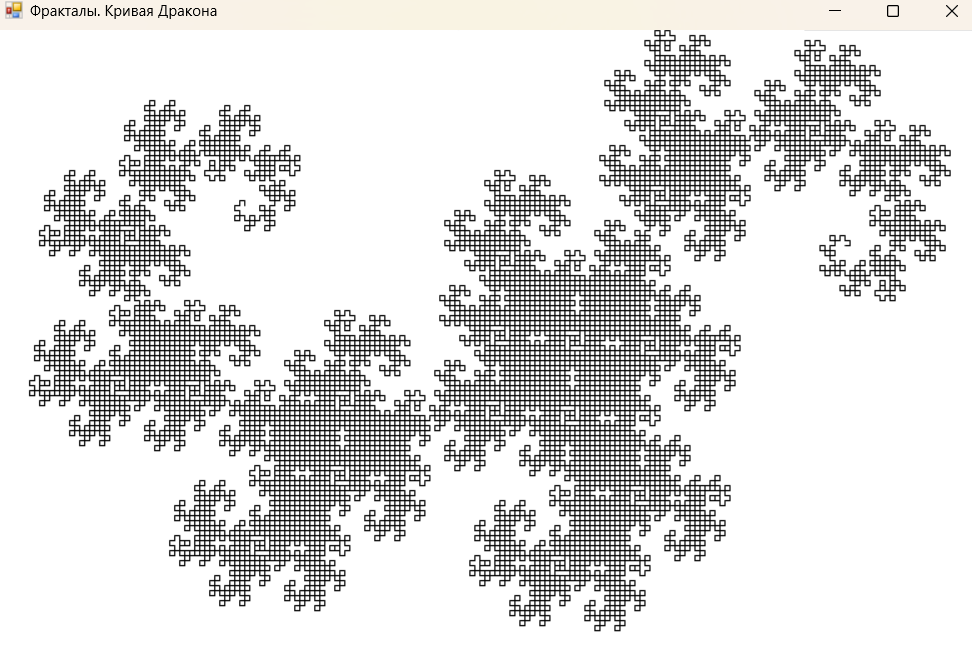
dx := t;

**end**;

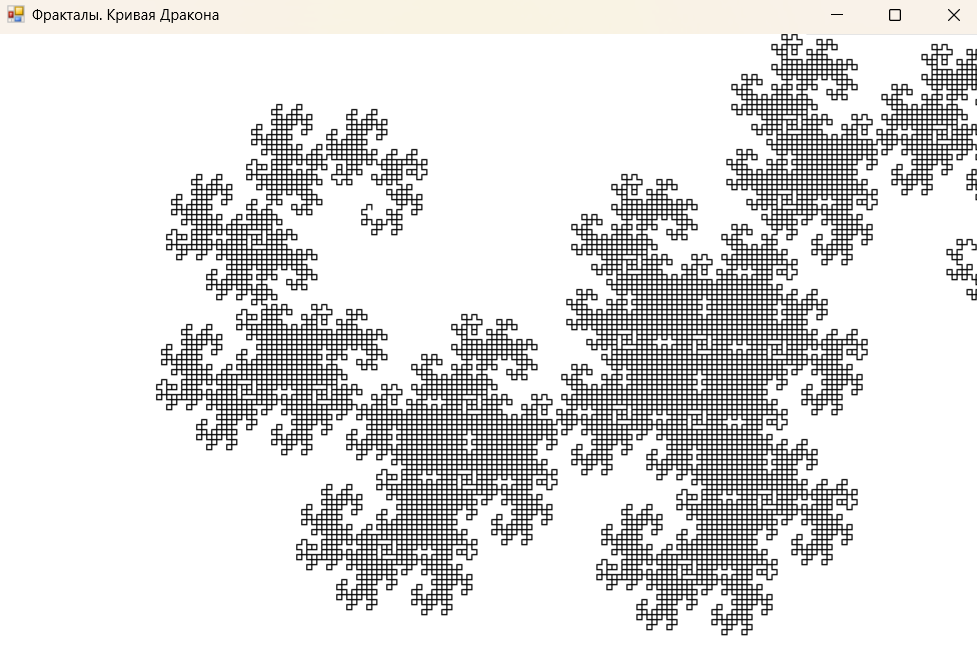
**end**;

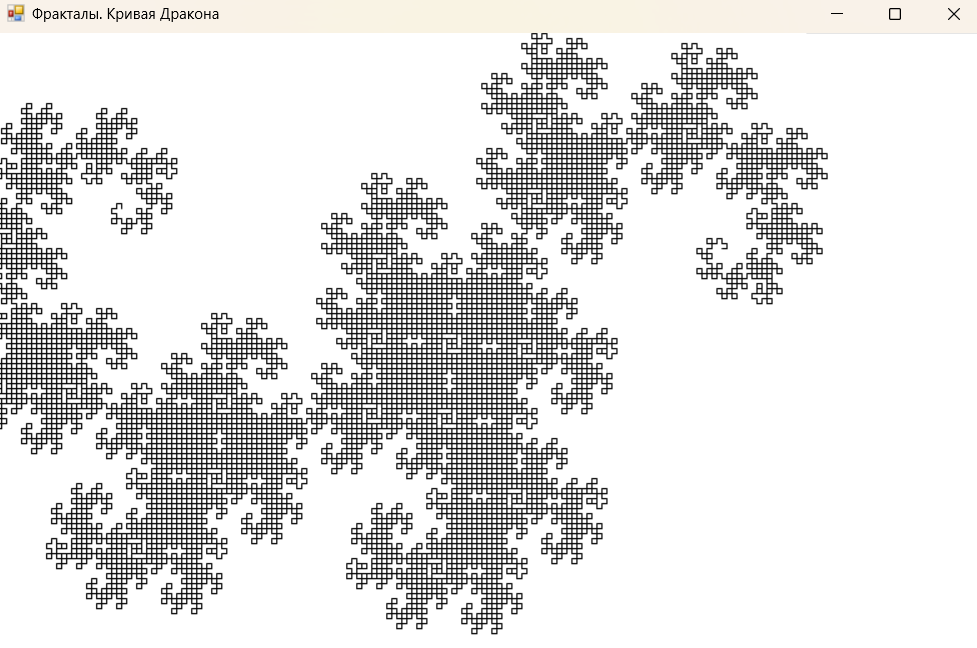
**end**.

1. Результат выполнения программы









1. Вывод

В ходе работы были получены навыки реализации алгоритмов с рекурсивными вычислениями. Была проведена работа с фракталом «Кривая Хартера-Хейтуэя». Предусмотрены возможности масштабирования, изменения глубины прорисовки и перемещения полученной фигуры.

Осуществлён отдельный модуль для построения ломанных.

В результате программа была успешно произведена.