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**

**«Лабораторная работа №7»**

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

Выполнил: студент учебной группы

ИСПк-101-51-00

Клыков Денис Борисович

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2025

В отчете должны отображаться:

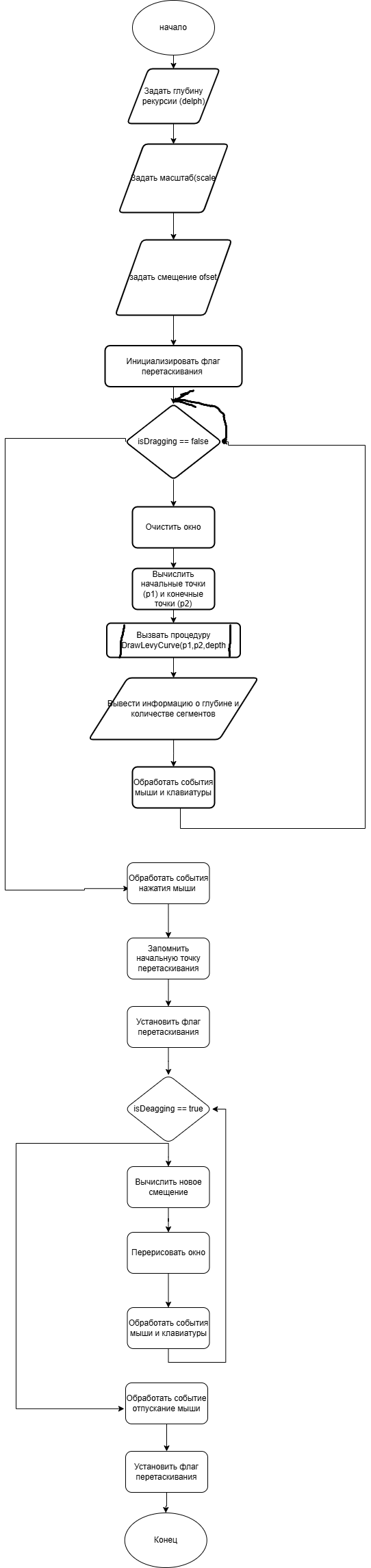
1. Цель работы
2. Формулировка задания (с вариантом)
3. Описание алгоритма и ответы на вопросы
4. Схема алгоритма с комментариями
5. Код программы
6. Результат выполнения программы
7. Вывод

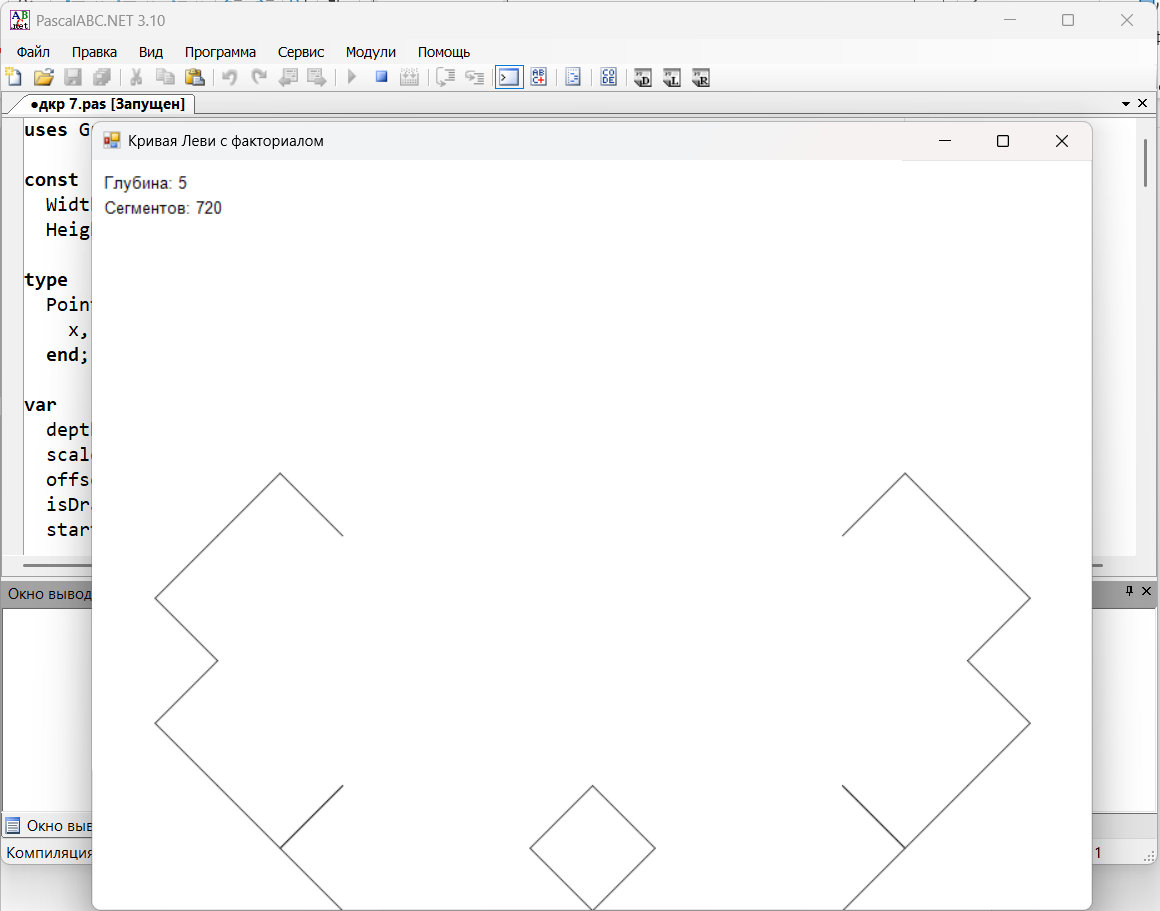
Вариант 6

Цель работы: Получение навыков реализации алгоритма с рекурсивными вычислением. Знакомство с фракталами.

Задание:

1. Написать программу для визуалиции фрактала «Кривая Леви»
2. Предусмотреть возможности масштабирования, изменения глубины прорисовки перемещения полученной фигуры.
3. Построение множества ломанных, образующих фрактал, должно осуществляться в отдельном модуле.





Код программы:

﻿uses GraphABC;

const

Width = 800;

Height = 600;

type

Point = record

x, y: real;

end;

var

depth: integer = 5; // Глубина рекурсии

scale: real = 1.0; // Масштаб

offset: Point = (x: 0; y: 0); // Смещение

isDragging: boolean = false; // Флаг для перемещения

startDrag: Point; // Начальная точка перемещения

// Функция для вычисления факториала

function Factorial(n: integer): integer;

begin

if n <= 1 then

Result := 1

else

Result := n \* Factorial(n - 1);

end;

// Рекурсивная процедура для рисования кривой Леви

procedure DrawLevyCurve(p1, p2: Point; depth: integer);

var

p3: Point;

begin

if depth = 0 then

begin

Line(Round(p1.x), Round(p1.y), Round(p2.x), Round(p2.y));

end

else

begin

// Вычисляем промежуточную точку

p3.x := (p1.x + p2.x) / 2 - (p2.y - p1.y) / 2;

p3.y := (p1.y + p2.y) / 2 + (p2.x - p1.x) / 2;

// Рекурсивно рисуем две части кривой

DrawLevyCurve(p1, p3, depth - 1);

DrawLevyCurve(p3, p2, depth - 1);

end;

end;

// Перерисовка окна

procedure Redraw;

begin

ClearWindow;

var p1: Point := (x: Width / 4 + offset.x; y: Height / 2 + offset.y);

var p2: Point := (x: 3 \* Width / 4 + offset.x; y: Height / 2 + offset.y);

DrawLevyCurve(p1, p2, depth);

// Выводим информацию о количестве сегментов (используем факториал)

TextOut(10, 10, 'Глубина: ' + IntToStr(depth));

TextOut(10, 30, 'Сегментов: ' + IntToStr(Factorial(depth + 1)));

end;

// Обработка событий мыши

procedure MouseDown(x, y, mb: integer);

begin

if mb = 1 then

begin

isDragging := true;

startDrag.x := x - offset.x;

startDrag.y := y - offset.y;

end;

end;

procedure MouseMove(x, y, mb: integer);

begin

if isDragging then

begin

offset.x := x - startDrag.x;

offset.y := y - startDrag.y;

Redraw;

end;

end;

procedure MouseUp(x, y, mb: integer);

begin

if mb = 1 then

isDragging := false;

end;

// Обработка событий клавиатуры

procedure KeyDown(key: integer);

begin

case key of

VK\_W: // Увеличение глубины рекурсии

begin

depth := depth + 1;

Redraw;

end;

VK\_S: // Уменьшение глубины рекурсии

begin

if depth > 1 then

depth := depth - 1;

Redraw;

end;

VK\_Up: // Перемещение вверх

begin

offset.y := offset.y - 10;

Redraw;

end;

VK\_Down: // Перемещение вниз

begin

offset.y := offset.y + 10;

Redraw;

end;

VK\_Left: // Перемещение влево

begin

offset.x := offset.x - 10;

Redraw;

end;

VK\_Right: // Перемещение вправо

begin

offset.x := offset.x + 10;

Redraw;

end;

end;

end;

begin

SetWindowSize(Width, Height);

SetWindowTitle('Кривая Леви с факториалом');

Redraw;

OnMouseDown := MouseDown;

OnMouseMove := MouseMove;

OnMouseUp := MouseUp;

OnKeyDown := KeyDown;

end.

Код модуля LevyCurve

unit LevyCurve;

interface

uses

GraphABC;

procedure DrawLevyCurve(x1, y1, x2, y2: real; depth: integer);

implementation

procedure DrawLevyCurve(x1, y1, x2, y2: real; depth: integer);// процедура

var

x3, y3, x4, y4: real;

begin

if depth = 0 then // база

begin

Line(Round(x1), Round(y1), Round(x2), Round(y2));

end

else

begin

x3 := (x1 + x2) / 2 + (y1 - y2) / Sqrt(2);// декомпозиция

y3 := (y1 + y2) / 2 + (x2 - x1) / Sqrt(2);

x4 := (x1 + x2) / 2 - (y1 - y2) / Sqrt(2);

y4 := (y1 + y2) / 2 - (x2 - x1) / Sqrt(2);

DrawLevyCurve(x1, y1, x3, y3, depth - 1);

DrawLevyCurve(x3, y3, x4, y4, depth - 1);

DrawLevyCurve(x4, y4, x2, y2, depth - 1);

end;

end;

end.

Вывод

Фракталы представляют собой примеры рекурсивных структур, которые можно визуализировать. Они демонстрируют, как простые правила могут приводить к сложным и красивым формам.

Изучение фракталов, таких как треугольник Серпинского или множество Мандельброта, позволяет увидеть, как рекурсивные алгоритмы могут быть использованы для генерации сложных графических изображений.

Рекурсивные алгоритмы и фракталы находят применение в различных областях, включая компьютерную графику, моделирование природных форм, обработку изображений и даже в алгоритмах поиска и сортировки.

Понимание этих концепций может быть полезным в разработке игр, симуляций и других приложений, где требуется работа с иерархическими структурами или сложными визуальными эффектами.

Реализация рекурсивных алгоритмов требует от программиста внимательности к деталям и способности предвидеть возможные проблемы, такие как переполнение стека или бесконечная рекурсия.

Практика в написании рекурсивных функций способствует улучшению навыков программирования, включая умение оптимизировать код и выбирать подходящие структуры данных.