# 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 «МДК 05.02 РАЗРАБОТКА КОДА ИНФОРМАЦИОННЫХ СИСТЕМ»**

Выполнил: студент учебной группы

ИСПк-205-52-00

Первяков Алексей Дмитриевич

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2024

Содержание

1. Цель работы

2. Формулировка задания

3. Описание алгоритма

4. Схема алгоритма с комментариями

5. Код программы

6. Результат выполнения программы

7. Вывод

**1. Цель работы:** получение навыков организации алгоритмов с рекурсивными вычислениями. Знакомство с фракталами

**2. Формулировка задания**

1. Написать программу на паскале для визуализации фрактала "Ковер Серпинского". Предусмотреть возможность масштабирования, изменения глубины прорисовки и перемещение полученной фигуры. Построение множества ломанных должно осуществляться в отдельном модуле.

**3. Описание алгоритма**

Ковер Серпинского – это фрактальная структура, которая состоит из квадратов, которые повторяются и заполняют плоскость. Каждый квадрат делится на девять более маленьких квадратов и серединный квадрат вырезается, и этот процесс повторяется бесконечно. Ковер Серпинского обладает свойством самоподобия и может быть создан с помощью простых математических правил.

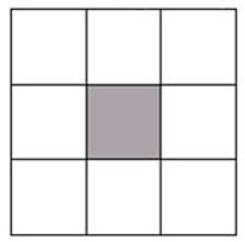


Рис.1. ломанная-генератор 1 глубина

Это ломанная-генератор, которая повторяется и уменьшается в масштабе каждый раз когда алгоритм запускается.

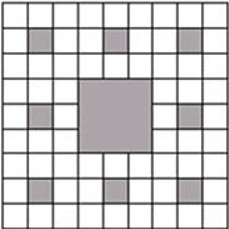


Рис.2. ломанная-генератор 2 глубина

Это второй запуск алгоритма. Здесь после 1 шага в каждом из девяти квадратов повторяется ломанная-генератор.

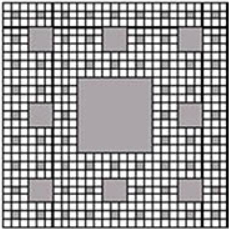


Рис.3. ломанная-генератор 3 глубина

Это третий запуск алгоритма. Здесь алгоритм повторяется циклично до бесконечности.

**4. Схема алгоритма с комментариями**

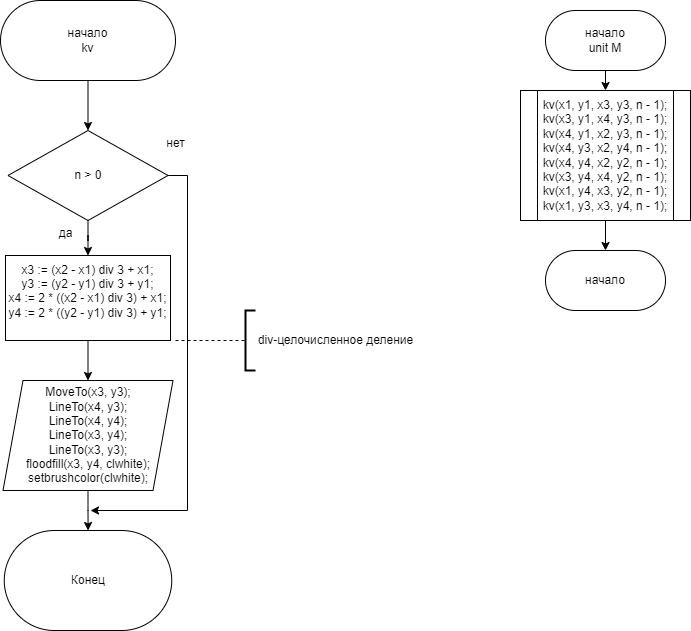


Рис 1. модуль M

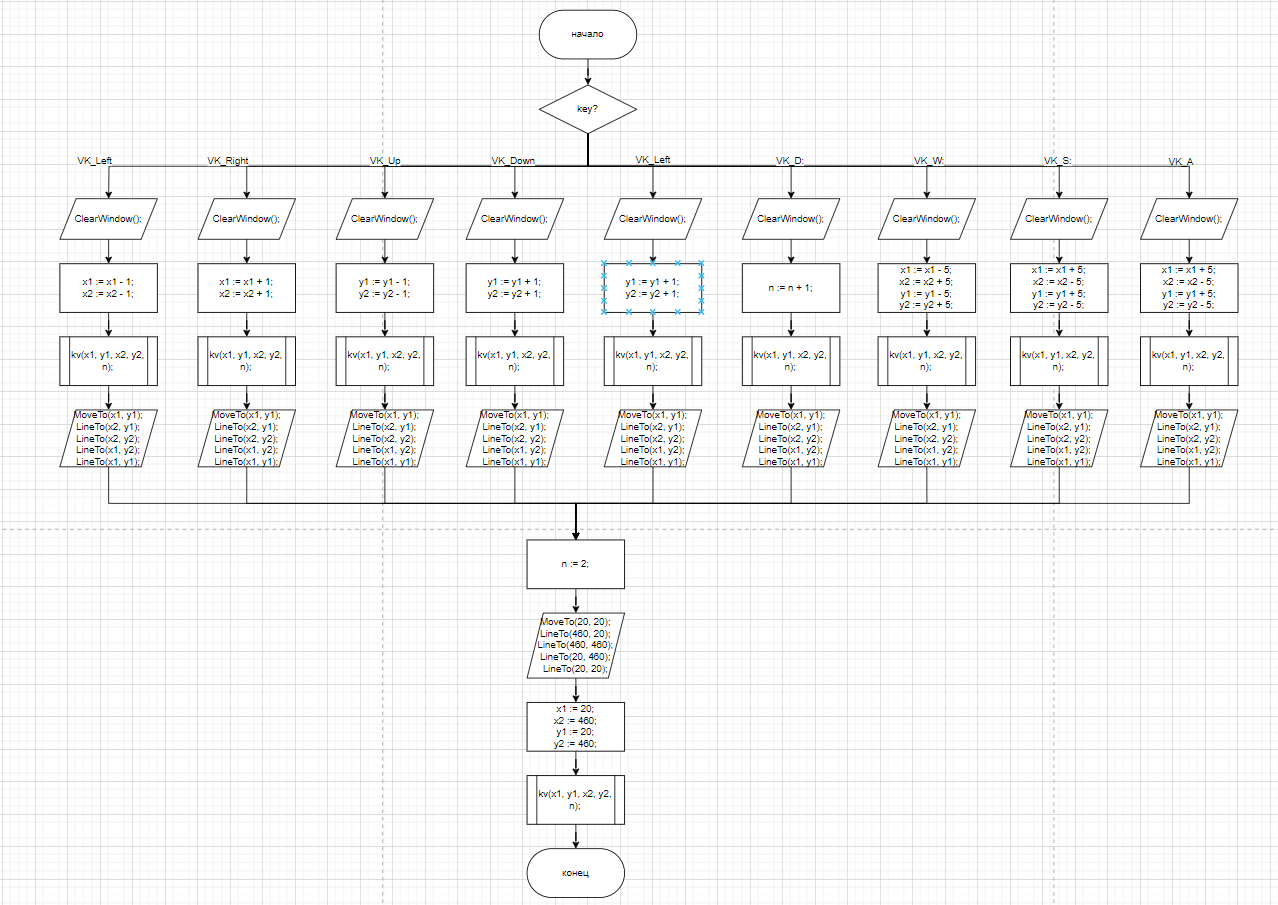


Рис 2. Код передвижения.

**5. Код программы**

**1.1**

**uses**

GraphABC, Mart;

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

**begin**

**case** Key **of**

VK\_Left:

**begin**

ClearWindow();

x1 := x1 - 5;

x2 := x2 - 5;

kv(x1, y1, x2, y2, n);

MoveTo(x1, y1);

LineTo(x2, y1);

LineTo(x2, y2);

LineTo(x1, y2);

LineTo(x1, y1);

**end**;

VK\_Right:

**begin**

ClearWindow();

x1 := x1 + 5;

x2 := x2 + 5;

kv(x1, y1, x2, y2, n);

MoveTo(x1, y1);

LineTo(x2, y1);

LineTo(x2, y2);

LineTo(x1, y2);

LineTo(x1, y1);

**end**;

VK\_Up:

**begin**

ClearWindow();

y1 := y1 - 5;

y2 := y2 - 5;

kv(x1, y1, x2, y2, n);

MoveTo(x1, y1);

LineTo(x2, y1);

LineTo(x2, y2);

LineTo(x1, y2);

LineTo(x1, y1);

**end**;

VK\_Down:

**begin**

ClearWindow();

y1 := y1 + 5;

y2 := y2 + 5;

kv(x1, y1, x2, y2, n);

MoveTo(x1, y1);

LineTo(x2, y1);

LineTo(x2, y2);

LineTo(x1, y2);

LineTo(x1, y1);

**end**;

VK\_A:

**begin**

ClearWindow();

n := n - 5;

kv(x1, y1, x2, y2, n);

MoveTo(x1, y1);

LineTo(x2, y1);

LineTo(x2, y2);

LineTo(x1, y2);

LineTo(x1, y1);

**end**;

VK\_D:

**begin**

ClearWindow();

n := n + 5;

kv(x1, y1, x2, y2, n);

MoveTo(x1, y1);

LineTo(x2, y1);

LineTo(x2, y2);

LineTo(x1, y2);

LineTo(x1, y1);

**end**;

VK\_W:

**begin**

ClearWindow();

x1 := x1 - 5;

x2 := x2 + 5;

y1 := y1 - 5;

y2 := y2 + 5;

kv(x1, y1, x2, y2, n);

MoveTo(x1, y1);

LineTo(x2, y1);

LineTo(x2, y2);

LineTo(x1, y2);

LineTo(x1, y1);

**end**;

VK\_S:

**begin**

ClearWindow();

x1 := x1 + 5;

x2 := x2 - 5;

y1 := y1 + 5;

y2 := y2 - 5;

kv(x1, y1, x2, y2, n);

MoveTo(x1, y1);

LineTo(x2, y1);

LineTo(x2, y2);

LineTo(x1, y2);

LineTo(x1, y1);

**end**;

**end**;

**end**;

**begin**

n := 2;

MoveTo(20, 20);

LineTo(460, 20);

LineTo(460, 460);

LineTo(20, 460);

LineTo(20, 20);

x1 := 20;

x2 := 460;

y1 := 20;

y2 := 460;

kv(x1, y1, x2, y2, n);

**while** true **do**

**begin**

OnKeyDown := KeyDown;

**end**;

**end**.

**1.2**

**Unit** Mart;

**Uses** GraphABC;

**var**

x1, x2, y1, y2: integer;

n: integer;

**procedure** kv(x1, y1, x2, y2, n: integer);

**var**

x3, y3, x4, y4: integer;

**begin**

**if** n > 0 **then**

**begin**

x3 := (x2 - x1) **div** 3 + x1;

y3 := (y2 - y1) **div** 3 + y1;

x4 := 2 \* ((x2 - x1) **div** 3) + x1;

y4 := 2 \* ((y2 - y1) **div** 3) + y1;

MoveTo(x3, y3);

LineTo(x4, y3);

LineTo(x4, y4);

LineTo(x3, y4);

LineTo(x3, y3);

floodfill(x3, y4, clwhite);

setbrushcolor(clwhite);

kv(x1, y1, x3, y3, n - 1);

kv(x3, y1, x4, y3, n - 1);

kv(x4, y1, x2, y3, n - 1);

kv(x4, y3, x2, y4, n - 1);

kv(x4, y4, x2, y2, n - 1);

kv(x3, y4, x4, y2, n - 1);

kv(x1, y4, x3, y2, n - 1);

kv(x1, y3, x3, y4, n - 1);

**end**;

**end**;

**end**.

**6. Результат выполнения программы**

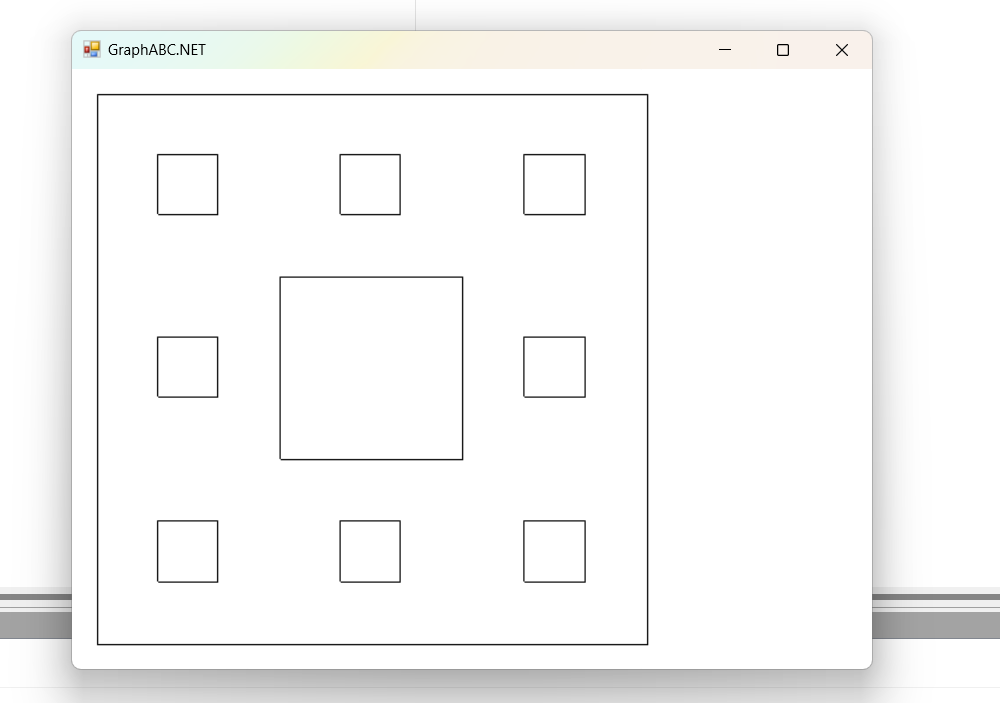
****

Рис 3. Ковер.

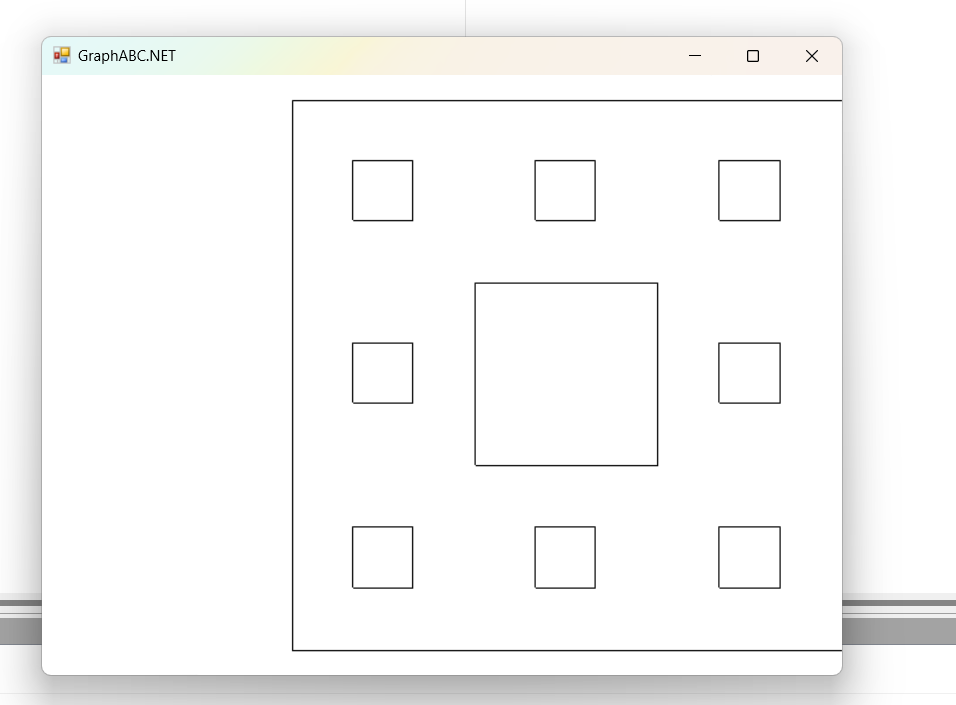


Рис 4. Ковер движение вправо.

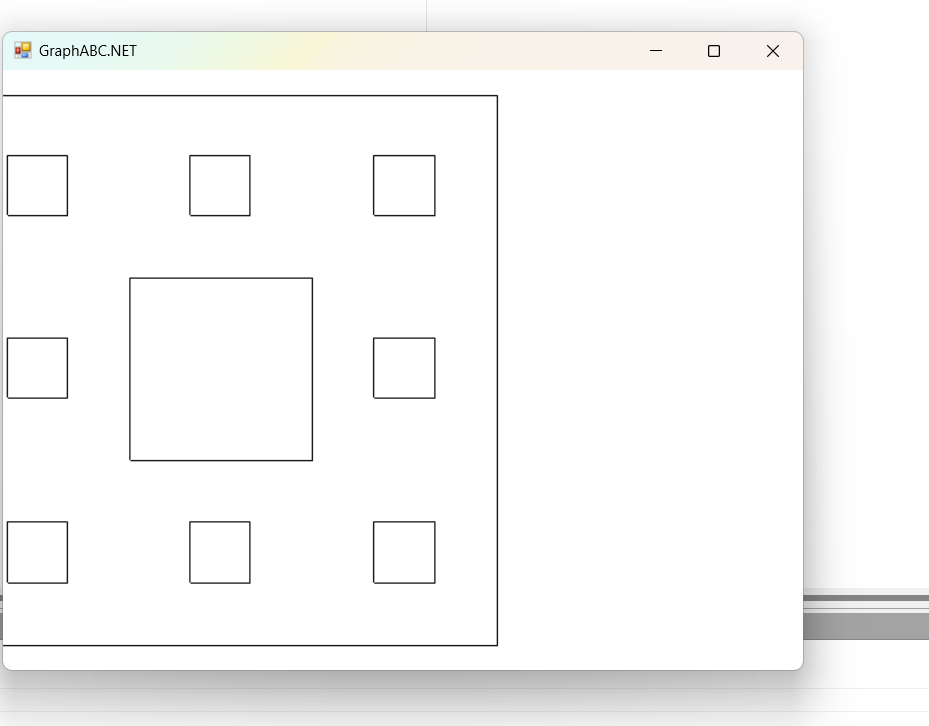
****

Рис 5. Ковер движение влево.

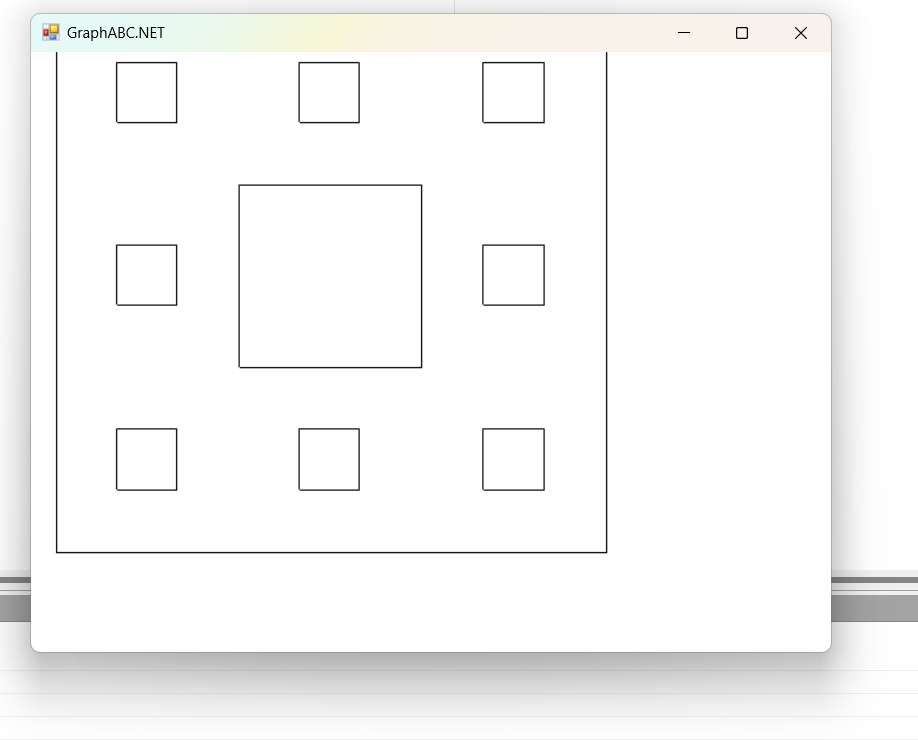


Рис 6. Ковер движение вверх.

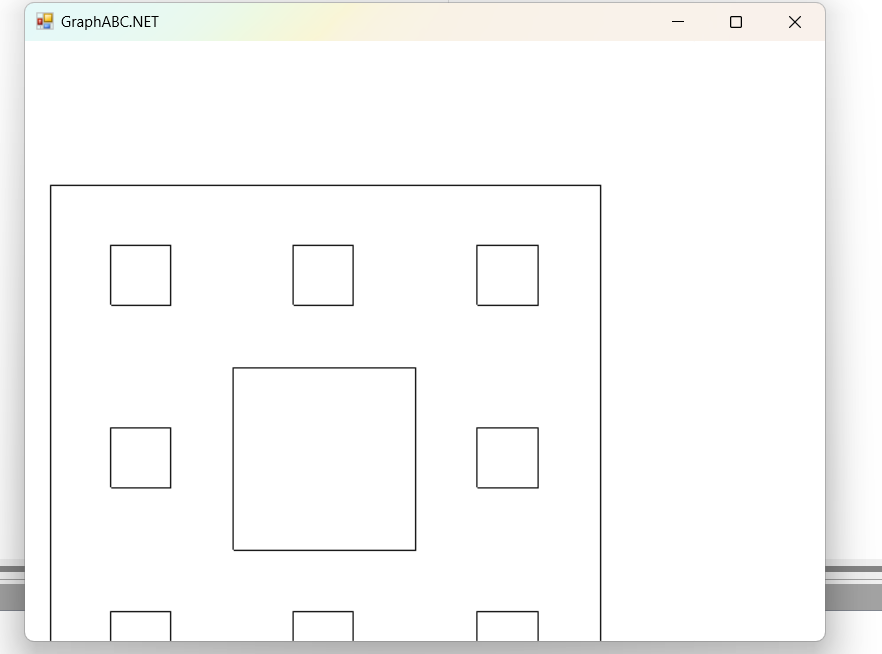


Рис 7. Ковер движение вниз.

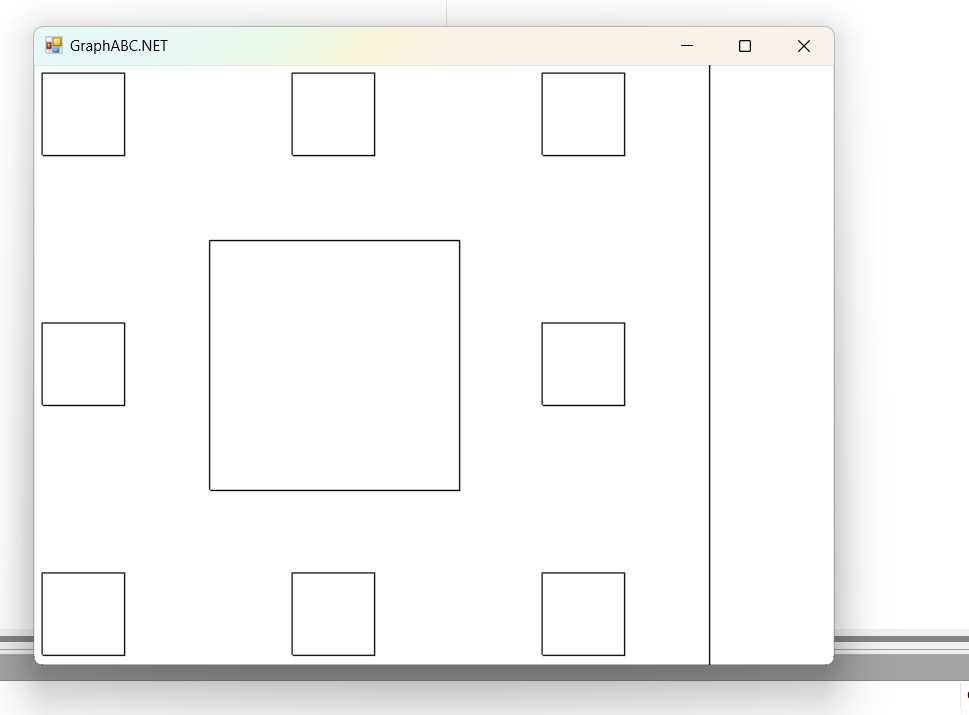


Рис 8. Ковер увеличение масштаба.

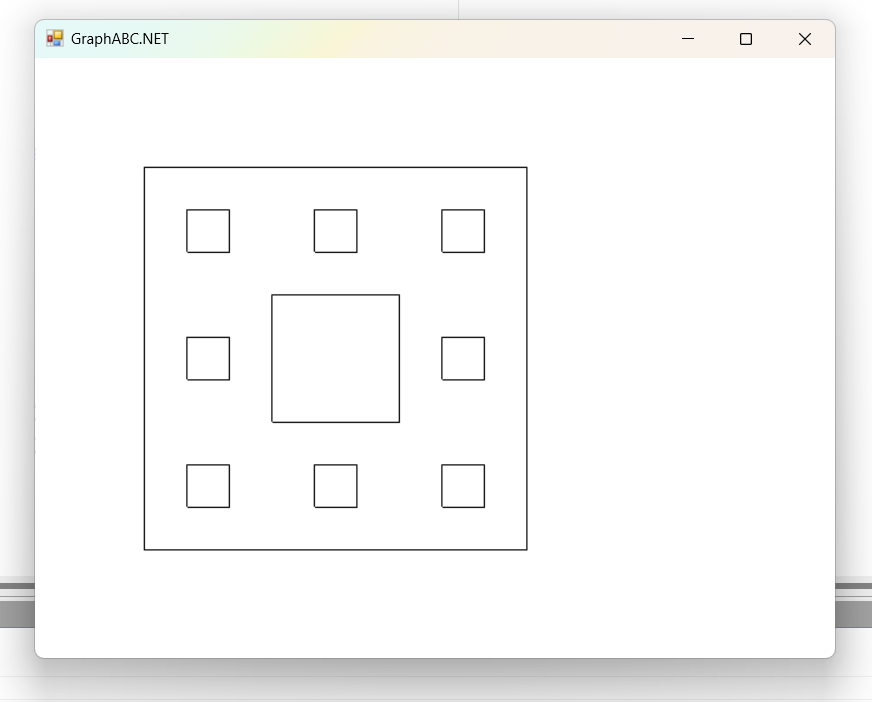


Рис 9. Ковер уменьшение масштаба.

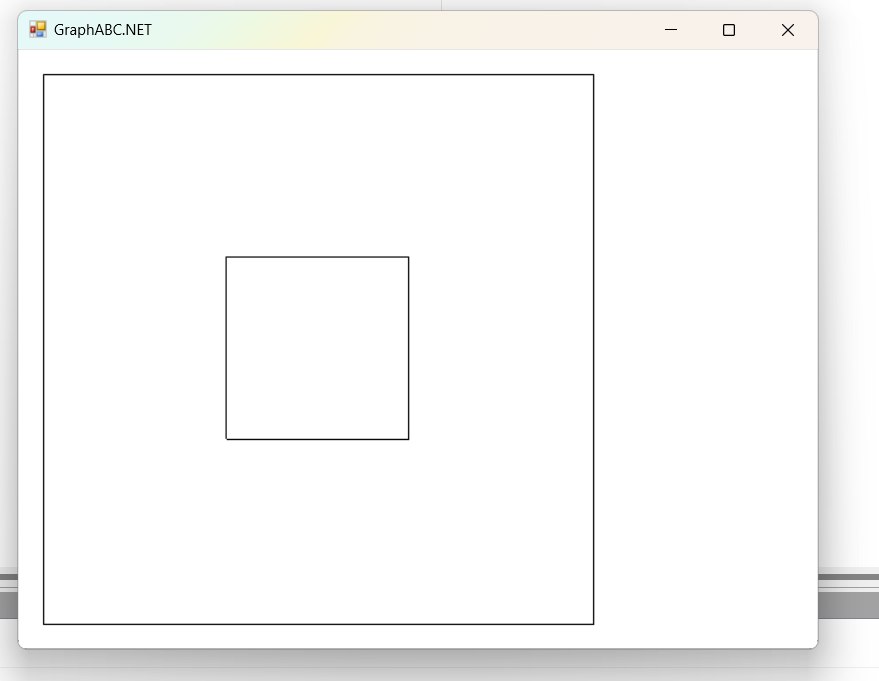


Рис 10. Ковер уменьшение глубины.

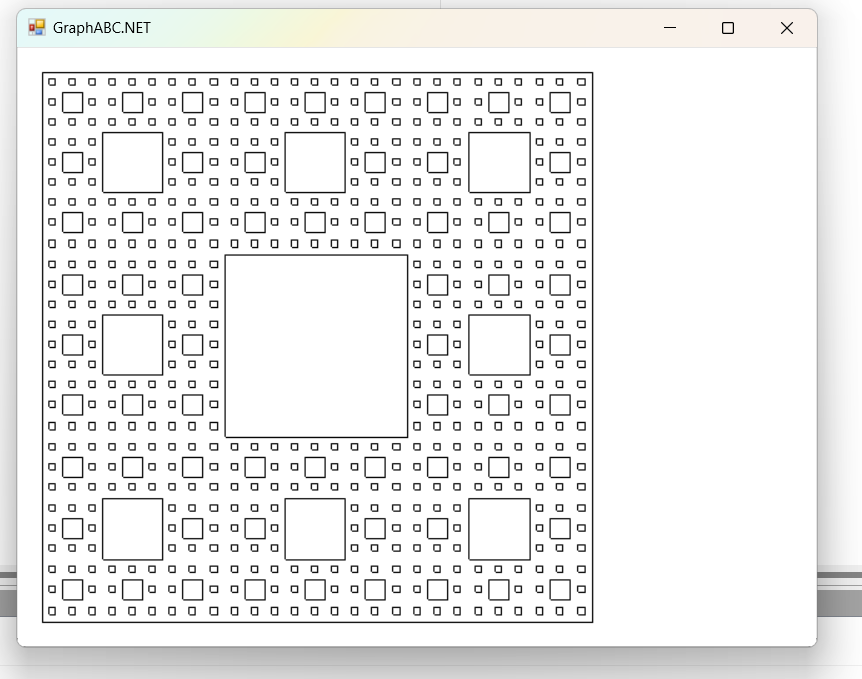


Рис 11. Ковер увеличение глубины.

**7. Вывод**

В ходе данной работы были написаны две программы. Одна создает новый модуль **«**M**»**, вторая определяет какое из слов встречается в заданной строке чаще всего.

Также мы столкнулись с некоторой трудностью, а именно выводом ломанных на отдельном модуле, однако воспользовавшись интернетом решили данную проблему.

В процессе выполнения работы мы использовали программу [draw.io](https://vk.com/away.php?to=http%3A%2F%2Fdraw.io&cc_key=) – это бесплатный онлайн-сервис, помогающий создавать блок-схемы, прототипы, инфографику и диаграммы любого вида. Нам он помог лучше представить алгоритм решения задачи.