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-203-52-00

Широков Михаил Александрович

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2025

**1. Цель работы**

Получение навыков реализации алгоритмов с рекурсивными вычислениями, знакомство с фракталами

**2. Формулировка задания**

**Вариант 22:**

**Задание 1** – написать программу для визуализации фрактала «Кривая Пеано»

**Задание 2** – предусмотреть возможность масштабирования, изменения глубины прорисовки и перемещения полученной фигуры.

**Задание 3** – построение множества ломанных, образующих фрактал, должно осуществляться в отдельном модуле.

**3. Описание алгоритма**

**Фрактал** – это сложная, бесконечно самоподобная геометрическая фигура, каждый фрагмент которой повторяется при уменьшении масштаба.

**Кривая Пеано** — общее название для параметрических кривых, образ которых содержит квадрат (или, в более общем смысле, открытые области пространства). Другое название — заполняющая пространство кривая.

**Алгоритм построения кривой Пеано**

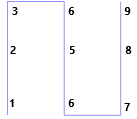


Рис. 1 – Первая итерация создания кривой Пеано

Рассмотрим первую итерацию создания кривой Пеано и отметим основные точки, которые она пересекает.

Следом, попробуем заменить эти точки элементами, подобными исходной кривой, причем заменять их необходимо в таком порядке, чтобы структура и направление исходной кривой сохранилась. Далее необходимо соединить концы этих элементов дополнительными линиями. В результате данных операций должна получиться вторая итерация кривой Пеано:

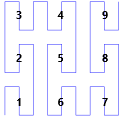


Рис. 2 –Вторая итерация создания кривой Пеано

Все последующие итерации аналогичны уже описанным. Подытожив можно выделить основной алгоритм построения кривой Пеано:

* Получить на вход какую-либо область.
* Осуществить проверку того, можно ли данную область поделить на 9 более мелких областей так, чтобы в них отчетливо отобразить основные элементы кривой.
* Если данную область можно поделить на 9, более мелких областей, то вначале необходимо отобразить все дополнительные линии, которые связывают это область между собой (рисунок 3), а затем рекурсивно применить данный алгоритм к каждой из 9 областей. Так же в рекурсивных вызовах необходимо учитывать направление тех элементов, которые будут отображены в будущем. Для этого необходимо пронумировать каждую из областей, и для каждой определить направление. Направление данных областей зависит от направления текущей области (рисунок 5). Всего есть 2 направления областей – «левый верхний угол-правый нижний угол»; «правый верхний угол-левый нижний угол» (рисунок 4). В зависимости от этих направлений будет отображен элемент кривой.
* Если данная область слишком мала, для последующего деления, значит необходимо отобразить на ней элемент с тем направлением, который был указан в рекурсивном вызове.

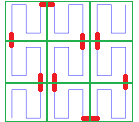


Рис. 3 – Схема построения кривой Пеано, где зеленым цветом выделены 9 областей, красным – связующие линии

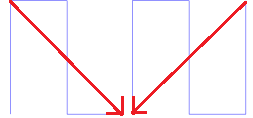


Рис. 4 – Элементы: «левый верхний угол-правый нижний угол»; «правый верхний угол-левый нижний угол»

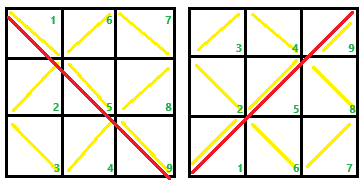


Рис 5. – Разложение областей с различными направлениями:

Красным отмечена основное направление области, зеленым – номера областей, на которые она рекурсивно разбивается, желтым – направление этих областей

Так же стоит отметить, что для более корректного построения кривой Пеано, так как области рекурсивно делятся на 9 частей, т.е. длинна и высота прямоугольника на 3, лучше использовать в качестве области квадрат, со стороной равной степени тройки.

**Алгоритм программы**

**Дополнительные подпрограммы:**

**Процедура – key\_press** – считывает значение клавиш и меняет в соответствии с ними переменные начальной точки отображения фигуры и ее масштаба. Используется для обработчика событий нажатия клавиш.

**Она принимает параметры**:

**a)** По значению – key: integer – код нажатой клавиши.

**Алгоритм работы:**

a) Проверка значения переменной key:

a.1) Если равен клавише A – происходит уменьшение начальной точки отображения фигуры по x на 80 пикселей.

a.2) Если равен клавише D – происходит увеличение начальной точки отображения фигуры по x на 80 пикселей.

a.3) Если равен клавише W – происходит уменьшение начальной точки отображения фигуры по y на 80 пикселей.

a.4) Если равен клавише S – происходит увеличение начальной точки отображения фигуры по y на 80 пикселей.

a.5) Если равен клавише E – происходит увеличение переменной, отвечающей за масштаб фигуры – range\_key в 3 раза.

b) Дополнительная проверка значения переменной key. Если она равна коду клавиши Q и переменная масштаба не меньше 3, то происходит уменьшение переменной, отвечающей за масштаб фигуры – range\_key в 3 раза.

с) Происходит проверка того, входит ли переменная key во множество обрабатываемых клавиш. Если это так, то все окно заливается черным цветом и происходит перерисовка кривой Пеано в соответствии с измененными параметрами.

**Основной алгоритм:**

а) Присваиваем переменным координатам начала построения кривой - x0, y0 – начальные значения – 230, переменной масштаба – range\_key – значение 1.

b) Устанавливаем ширину и высоту экрана 800 на 800.

c) Заливаем весь экран черным и устанавливаем цвет кисти – белый.

d) Рисуем прямоугольник – рамку для кривой Пеано.

e) При помощи функции из модуля отображаем кривую Пеано.

f) Настраиваем обработчик событий нажатия клавиш, передавая ему функцию key\_press.

**4. Схема алгоритма с комментариями**

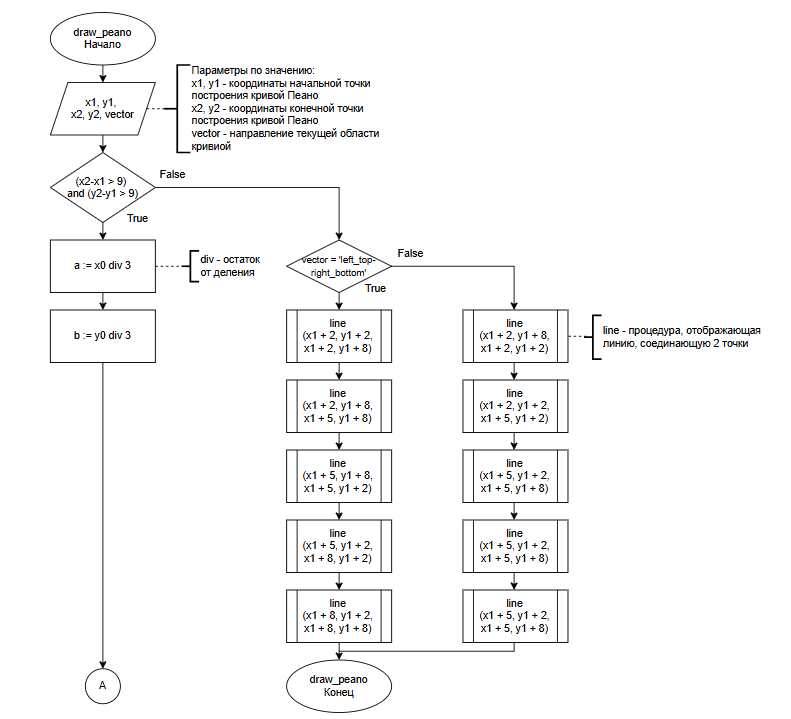
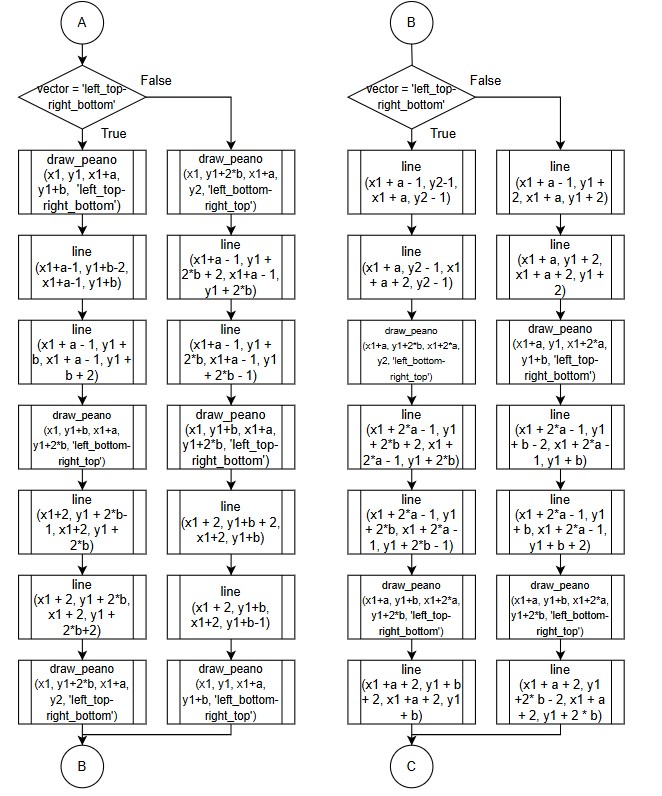


Рис. 6.1 – Алгоритм процедуры draw\_peano

 Рис. 6.2 – Алгоритм процедуры draw\_peano

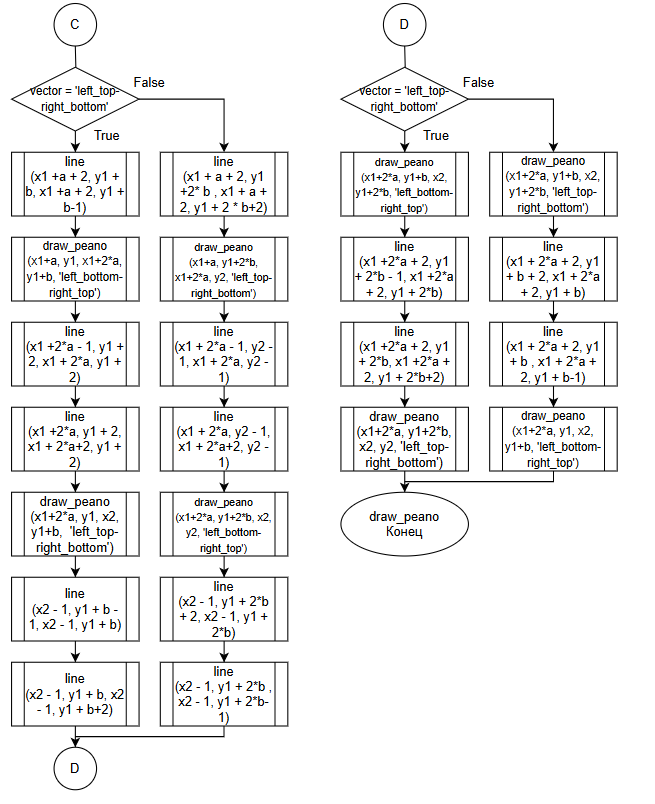


Рис. 6.3 – Алгоритм процедуры draw\_peano

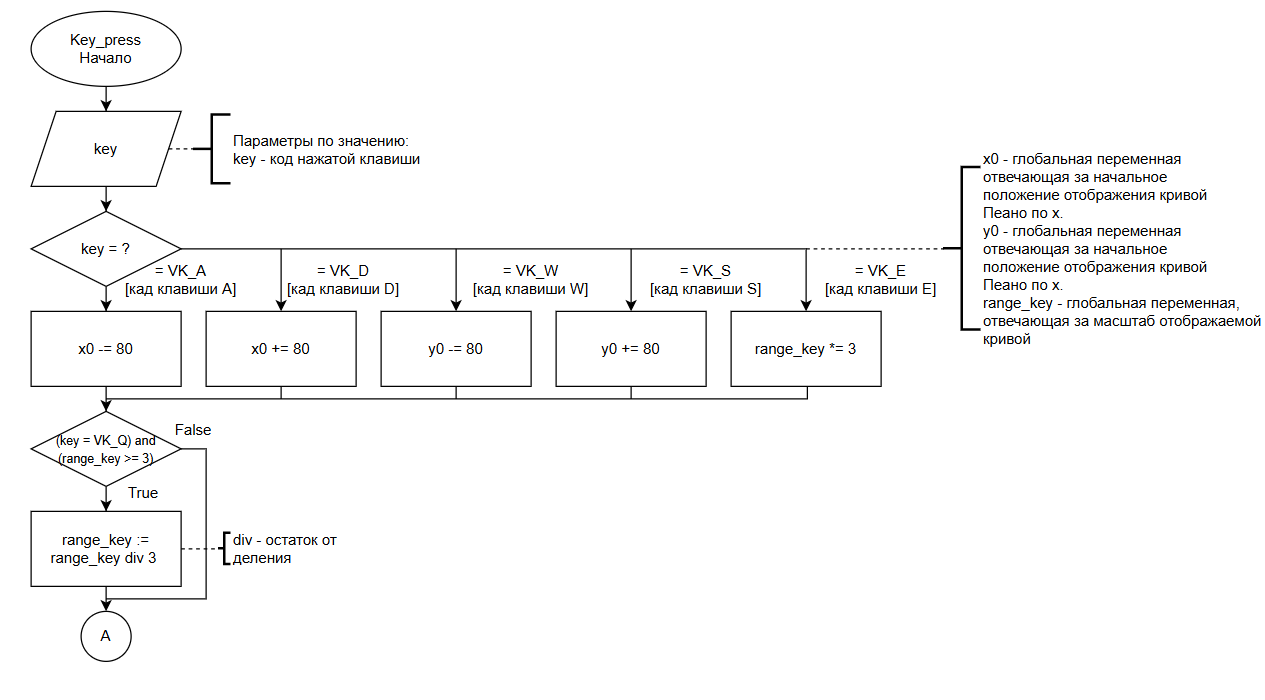


Рис. 7.1 – Алгоритм процедуры key\_press

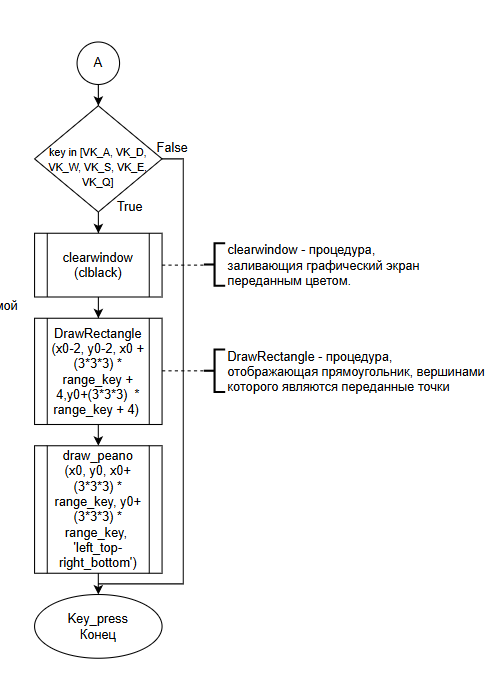


Рис. 7.2 – Алгоритм процедуры key\_press

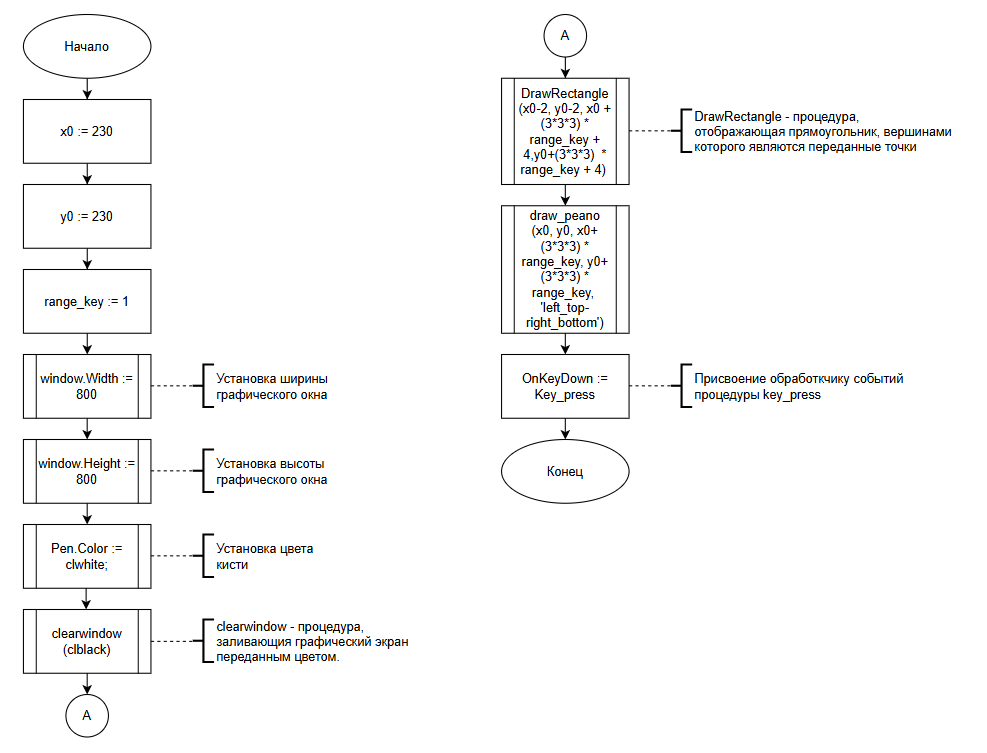


Рис. 8 – Алгоритм основной части программы

**5. Код программы**

**program** DKR7;

**uses** Peano\_curwe;

**uses** GraphABC;

**var**

x0, y0, range\_key:integer;

**procedure** key\_press(key:integer);

**begin**

**case** key **of**

VK\_A: x0 -= 80;

VK\_D: x0 += 80;

VK\_W: y0 -= 80;

VK\_S: y0 += 80;

VK\_E: range\_key \*= 3;

**end**;

**if** (key = VK\_Q) **and** (range\_key >= 3) **then**

range\_key := range\_key **div** 3;

**if** key **in** [VK\_A, VK\_D, VK\_W, VK\_S, VK\_E, VK\_Q] **then**

**begin**

clearwindow(clblack);

DrawRectangle(x0-2, y0-2, x0+(3\*3\*3) \* range\_key + 4, y0+(3\*3\*3) \* range\_key + 4);

Peano\_curwe.draw\_peano(x0, y0, x0+(3\*3\*3) \* range\_key, y0+(3\*3\*3) \* range\_key, 'left\_top-right\_bottom');

**end**;

**end**;

**begin**

x0 := 230;

y0:=230;

range\_key := 1;

window.Width := 800;

window.Height := 800;

Pen.Color := clwhite;

clearwindow(clblack);

DrawRectangle(x0-2, y0-2, x0+(3\*3\*3) \* range\_key + 4, y0+(3\*3\*3) \* range\_key + 4);

Peano\_curwe.draw\_peano(x0, y0, x0+(3\*3\*3) , y0+(3\*3\*3), 'left\_top-right\_bottom');

OnKeyDown := key\_press;

**end**.

**Unit** Peano\_curwe;

**uses** GraphABC;

**procedure** draw\_peano(x1, y1, x2, y2: integer; vector: string);

**var**

a, b:integer;

**begin**

**if** (x2-x1 > 9) **and** (y2-y1 > 9) **then** // Проверка длинны и ширины области

**begin**

a := (x2-x1) **div** 3; // Выделение 1/3 от длинны и ширины (для удобства использования в формулах)

b := (y2-y1) **div** 3;

**if** vector = 'left\_top-right\_bottom' **then** // Разделение области Левый верхний угол-Правый нижний угол

**begin**

draw\_peano(x1, y1, x1+a, y1+b, 'left\_top-right\_bottom'); //1

line(x1+a-1, y1+b-2, x1+a-1, y1+b);

line(x1 + a - 1, y1 + b, x1 + a - 1, y1 + b + 2);

draw\_peano(x1, y1+b, x1+a, y1+2\*b, 'left\_bottom-right\_top'); //2

line(x1+2, y1 + 2\*b-1, x1+2, y1 + 2\*b);

line(x1 + 2, y1 + 2\*b, x1 + 2, y1 + 2\*b+2);

draw\_peano(x1, y1+2\*b, x1+a, y2, 'left\_top-right\_bottom'); //3

line(x1 + a - 1, y2-1, x1 + a, y2 - 1);

line(x1 + a, y2 - 1, x1 + a + 2, y2 - 1);

draw\_peano(x1+a, y1+2\*b, x1+2\*a, y2, 'left\_bottom-right\_top'); //4

line(x1 + 2\*a - 1, y1 + 2\*b + 2, x1 + 2\*a - 1, y1 + 2\*b);

line(x1 + 2\*a - 1, y1 + 2\*b, x1 + 2\*a - 1, y1 + 2\*b - 1);

draw\_peano(x1+a, y1+b, x1+2\*a, y1+2\*b, 'left\_top-right\_bottom'); //5

line(x1 +a + 2, y1 + b + 2, x1 +a + 2, y1 + b);

line(x1 +a + 2, y1 + b, x1 +a + 2, y1 + b-1);

draw\_peano(x1+a, y1, x1+2\*a, y1+b, 'left\_bottom-right\_top'); //6

line(x1 +2\*a - 1, y1 + 2, x1 + 2\*a, y1 + 2);

line(x1 +2\*a, y1 + 2, x1 + 2\*a+2, y1 + 2);

draw\_peano(x1+2\*a, y1, x2, y1+b, 'left\_top-right\_bottom'); //7

line(x2 - 1, y1 + b - 1, x2 - 1, y1 + b);

line(x2 - 1, y1 + b, x2 - 1, y1 + b+2);

draw\_peano(x1+2\*a, y1+b, x2, y1+2\*b, 'left\_bottom-right\_top'); //8

line(x1 +2\*a + 2, y1 + 2\*b - 1, x1 +2\*a + 2, y1 + 2\*b);

line(x1 +2\*a + 2, y1 + 2\*b, x1 +2\*a + 2, y1 + 2\*b+2);

draw\_peano(x1+2\*a, y1+2\*b, x2, y2, 'left\_top-right\_bottom'); //9

**end**

**else** // Разделение области Правый верхний угол-Левый нижний угол

**begin**

draw\_peano(x1, y1+2\*b, x1+a, y2, 'left\_bottom-right\_top'); //1

line(x1+a - 1, y1 + 2\*b + 2, x1+a - 1, y1 + 2\*b);

line(x1+a - 1, y1 + 2\*b, x1+a - 1, y1 + 2\*b - 1);

draw\_peano(x1, y1+b, x1+a, y1+2\*b, 'left\_top-right\_bottom'); //2

line(x1 + 2, y1+b + 2, x1+2, y1+b);

line(x1 + 2, y1+b, x1+2, y1+b-1);

draw\_peano(x1, y1, x1+a, y1+b, 'left\_bottom-right\_top'); //3

line(x1 + a - 1, y1 + 2, x1 + a, y1 + 2);

line(x1 + a, y1 + 2, x1 + a + 2, y1 + 2);

draw\_peano(x1+a, y1, x1+2\*a, y1+b, 'left\_top-right\_bottom'); //4

line(x1 + 2\*a - 1, y1 + b - 2, x1 + 2\*a - 1, y1 + b);

line(x1 + 2\*a - 1, y1 + b, x1 + 2\*a - 1, y1 + b + 2);

draw\_peano(x1+a, y1+b, x1+2\*a, y1+2\*b, 'left\_bottom-right\_top'); //5

line(x1 + a + 2, y1 +2\* b - 2, x1 + a + 2, y1 + 2 \* b);

line(x1 + a + 2, y1 +2\* b , x1 + a + 2, y1 + 2 \* b+2);

draw\_peano(x1+a, y1+2\*b, x1+2\*a, y2, 'left\_top-right\_bottom'); //6

line(x1 + 2\*a - 1, y2 - 1, x1 + 2\*a, y2 - 1);

line(x1 + 2\*a, y2 - 1, x1 + 2\*a+2, y2 - 1);

draw\_peano(x1+2\*a, y1+2\*b, x2, y2, 'left\_bottom-right\_top'); //7

line(x2 - 1, y1 + 2\*b + 2, x2 - 1, y1 + 2\*b);

line(x2 - 1, y1 + 2\*b , x2 - 1, y1 + 2\*b-1);

draw\_peano(x1+2\*a, y1+b, x2, y1+2\*b, 'left\_top-right\_bottom'); //8

line(x1 + 2\*a + 2, y1 + b + 2, x1 + 2\*a + 2, y1 + b);

line(x1 + 2\*a + 2, y1 + b , x1 + 2\*a + 2, y1 + b-1);

draw\_peano(x1+2\*a, y1, x2, y1+b, 'left\_bottom-right\_top'); //9

**end**;

**end**

**else**

**begin**

**if** vector = 'left\_top-right\_bottom' **then** // Отрисовка элемента Левый верхний угол-Правый нижний угол

**begin**

line(x1 + 2, y1 + 2, x1 + 2, y1 + 8);

line(x1 + 2, y1 + 8, x1 + 5, y1 + 8);

line(x1 + 5, y1 + 8, x1 + 5, y1 + 2);

line(x1 + 5, y1 + 2, x1 + 8, y1 + 2);

line(x1 + 8, y1 + 2, x1 + 8, y1 + 8);

**end**

**else** // Отрисовка элемента Правый верхний угол-Левый нижний угол

**begin**

line(x1 + 2, y1 + 8, x1 + 2, y1 + 2);

line(x1 + 2, y1 + 2, x1 + 5, y1 + 2);

line(x1 + 5, y1 + 2, x1 + 5, y1 + 8);

line(x1 + 5, y1 + 8, x1 + 8, y1 + 8);

line(x1 + 8, y1 + 8, x1 + 8, y1 + 2);

**end**;

**end**;

**end**;

**end**.

**6. Результаты выполнения программы**

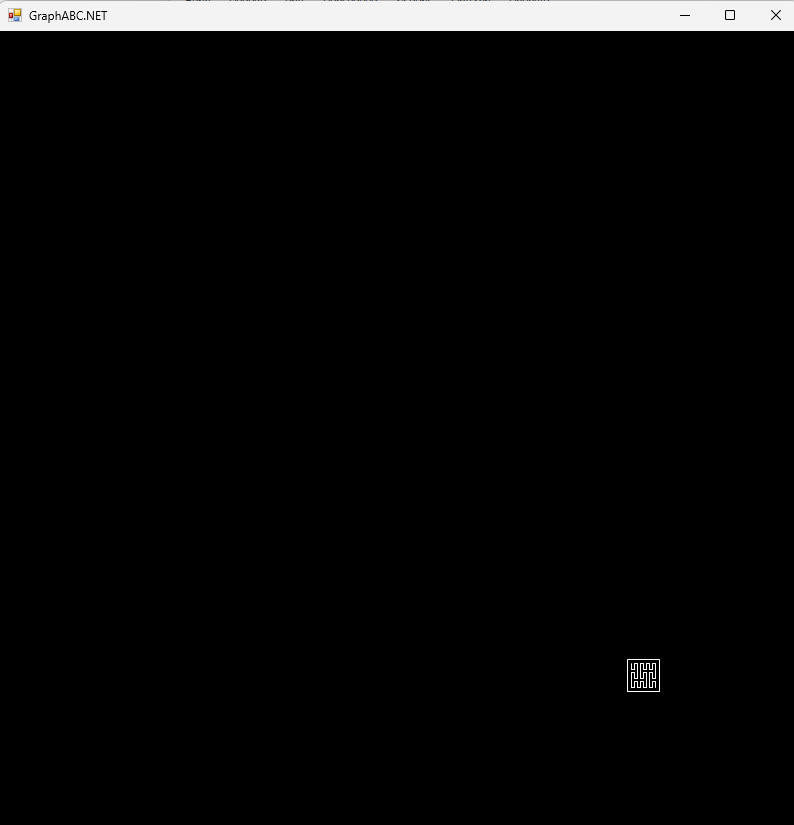
****

Рис 9.1 – Результат выполнения программы

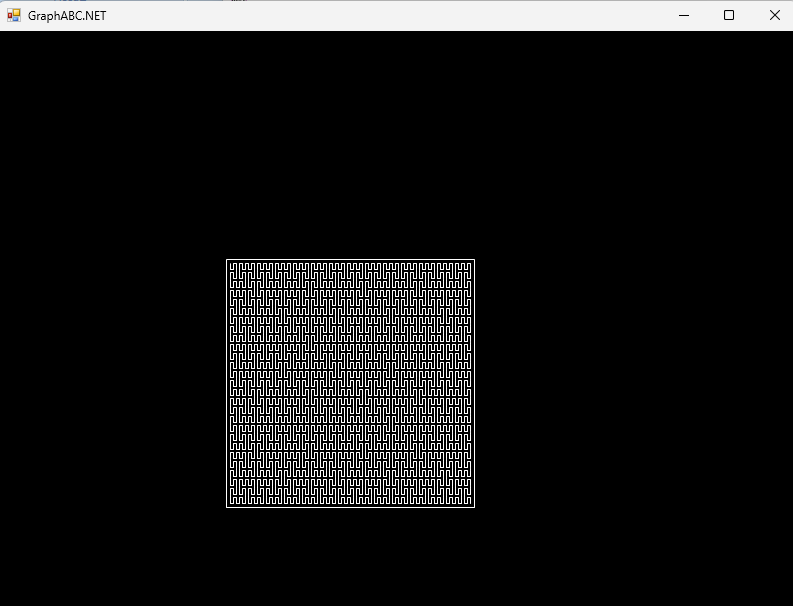


Рис 9.2 – Результат выполнения программы

**7. Вывод**

Работа завершена. В ходе работы были комплексно изучены основные принципы построения и работы рекурсивных алгоритмов. Были изучены и применены на практике в построении алгоритма такие понятия как параметризация, декомпозиция и база рекурсии, рекурсивный подъем и рекурсивный спуск. Помимо этого в процессе работы были исследованы такие геометрические фигуры, как фракталы, в частности кривая Пеано, а также получены навыки по созданию и использованию собственных модулей в языке программирования Pascal. В результате работы была разработана рекурсивная процедура по построению кривой Пеано, а так же дополнительный алгоритм, позволяющий перемещать ее по графическому экрану и менять размер.