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 Разработка кода информационных систем»**

Выполнил: студент учебной группы ИСПк-202-52-00

Причинин Вадим Андреевич

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2024

Цель работы: получение навыков реализации алгоритмов с рекурсивными вычислениями, знакомство с фракталами.

Задание:

1. Написать программу для визуализации фрактала “Кривая Минковского”
2. Предусмотреть возможности масштабирования, изменения глубины прорисовки и перемещения полученной фигуры.
3. Построение множества ломанных, образующих фрактал, должно осуществляться в отдельном модуле.

Описание алгоритма:

**Кривая Минковского** — классический геометрический фрактал, предложенный Минковским. Инициатором является отрезок, а генератором является ломаная из восьми звеньев (два равных звена продолжают друг друга). Форма такого фрактала используется для антенн.

Рисуется по правилу:

**переменные**: F

**константы**: + −

**старт**: F

**правило**: (F → F−F+F+FF−F−F+F)

**угол**: 90°

Здесь F означают «рисуем отрезок», + означает «повернуть вправо на угол 90», а − означает «повернуть влево на угол 90».

Преувеличении числа F увеличивается глубина

**Схема алгоритма:**

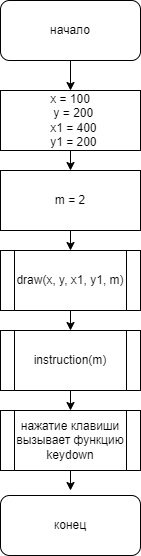


Рисунок 1 — Основное тело программы

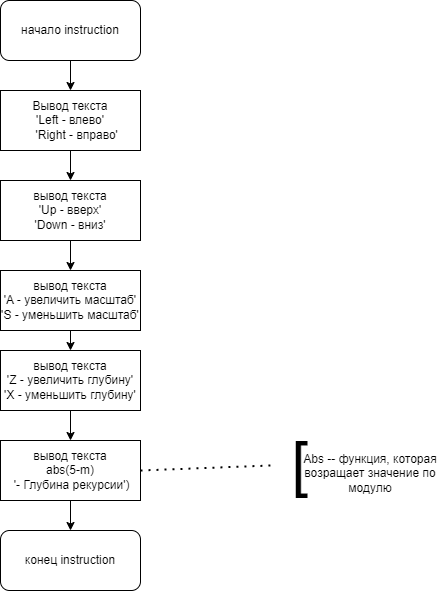


Рисунок 2 — подпрограмма instruction

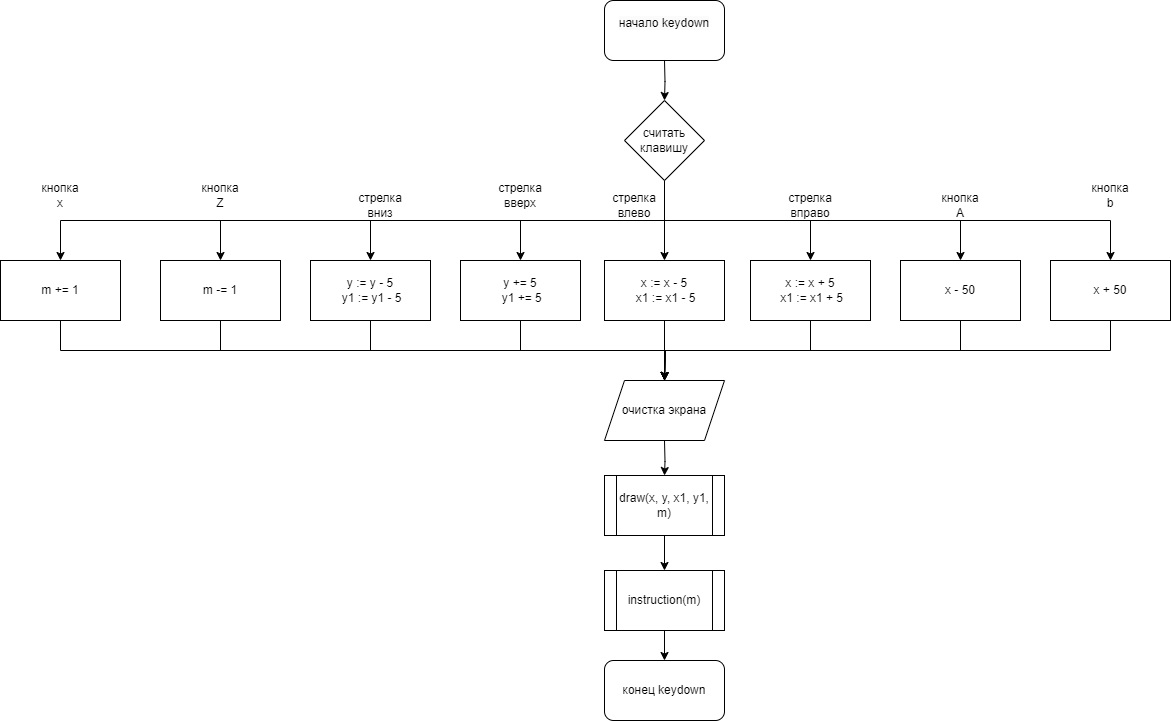


Рисунок 3 — считывание клавиш

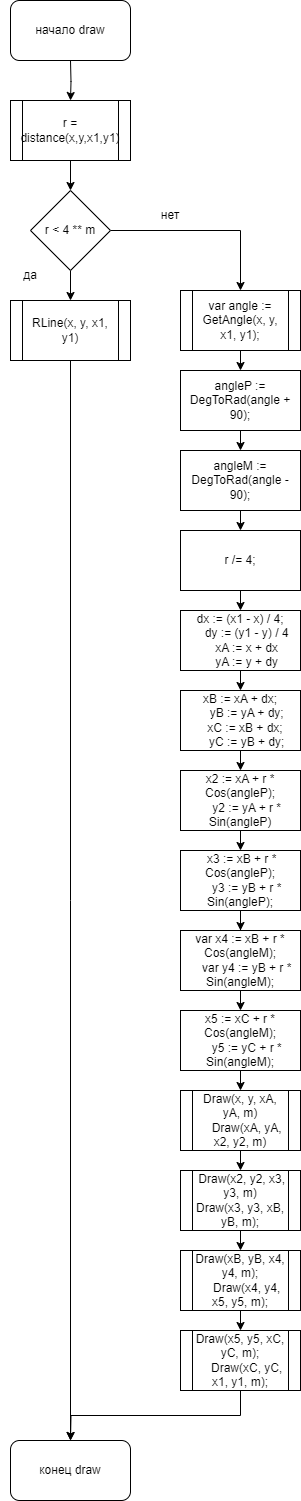


Рисунок 4 — подпрограмма draw

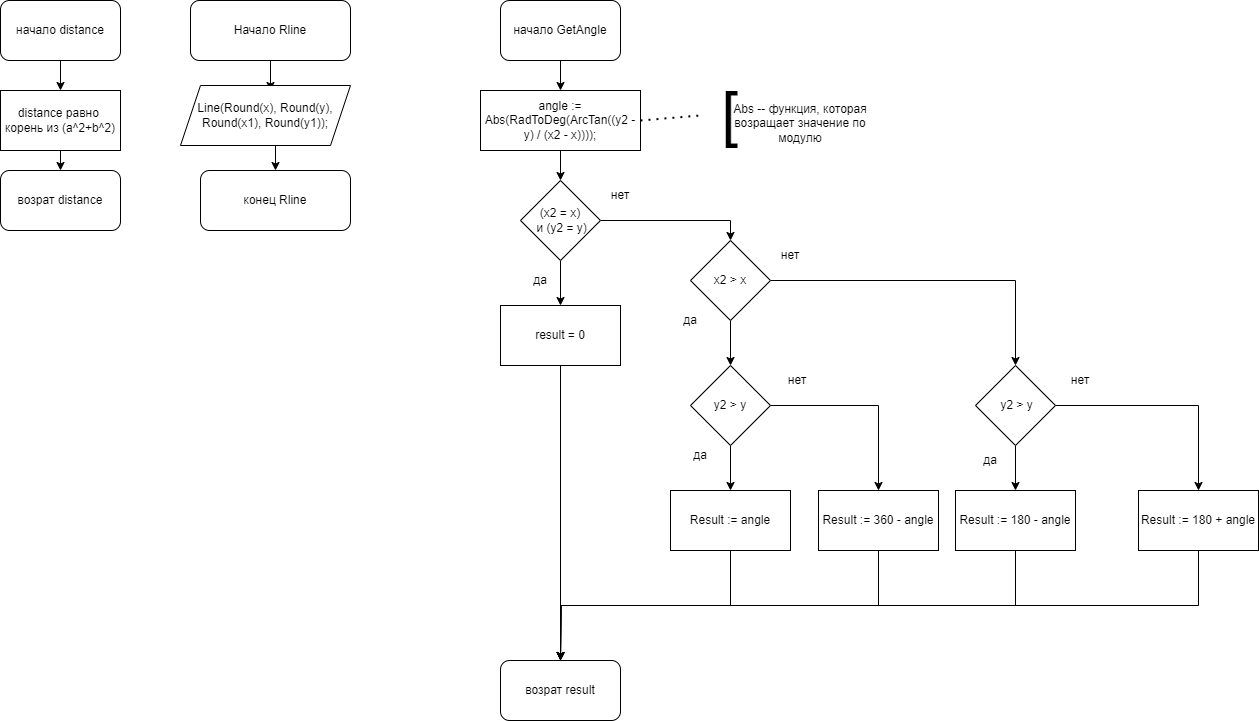


Рисунок 5 — подпрограммы distance, Rline, GetAngle

**Код программы**

**uses** GraphABC;

**uses** DrawFractal;

**var** m: integer;

**var** x, y, x1, y1: integer;

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

**begin**

**case** key **of**

VK\_Down: **begin** y := y - 5; y1 := y1 - 5 **end**;

VK\_Up: **begin** y += 5; y1 += 5 **end**;

VK\_Right: **begin** x := x - 5; x1 := x1 - 5 **end**;

VK\_Left: **begin** x := x + 5; x1 := x1 + 5 **end**;

VK\_A: x := x - 50;

Vk\_S: x := x + 50;

vk\_z: **if** m>0 **then** m -= 1;

vk\_x:**if** m<5 **then** m += 1;

vk\_F:

**end**;

Window.Clear;

draw(x, y, x1, y1, m);

instruction(m);

redraw;

**end**;

**begin**

LockDrawing;

x := 100;

y := 200;

x1 := 400;

y1 := 200;

m:= 2;

draw(x, y, x1, y1, m);

instruction(m);

redraw;

OnkeyDown := keydown;

**end**.

**unit** DrawFractal;

**uses** GraphABC;

**procedure** RLine(x, y, x1, y1: real) := Line(Round(x), Round(y), Round(x1), Round(y1));

**function** GetAngle(x, y, x2, y2: real): real;

**begin**

**var** angle := Abs(RadToDeg(ArcTan((y2 - y) / (x2 - x))));

**if** (x2 = x) **and** (y2 = y) **then**

Result := 0

**else**

**if** x2 > x **then**

**if** y2 > y **then** Result := angle **else** Result := 360 - angle

**else if** y2 > y **then** Result := 180 - angle **else** Result := 180 + angle;

**end**;

**function** Distance(x, y, x1, y1: real) := Sqrt(Sqr(x1 - x) + Sqr(y1 - y));

**procedure** Draw(x, y, x1, y1: real; m:integer);

**begin**

**var** r := Distance(x, y, x1, y1);

**if** r < 4\*\*m **then**

RLine(x, y, x1, y1)

**else**

**begin**

**var** angle := GetAngle(x, y, x1, y1);

**var** angleP := DegToRad(angle + 90);

**var** angleM := DegToRad(angle - 90);

r /= 4;

**var** dx := (x1 - x) / 4;

**var** dy := (y1 - y) / 4;

**var** xA := x + dx;

**var** yA := y + dy;

**var** xB := xA + dx;

**var** yB := yA + dy;

**var** xC := xB + dx;

**var** yC := yB + dy;

**var** x2 := xA + r \* Cos(angleP);

**var** y2 := yA + r \* Sin(angleP);

**var** x3 := xB + r \* Cos(angleP);

**var** y3 := yB + r \* Sin(angleP);

**var** x4 := xB + r \* Cos(angleM);

**var** y4 := yB + r \* Sin(angleM);

**var** x5 := xC + r \* Cos(angleM);

**var** y5 := yC + r \* Sin(angleM);

Draw(x, y, xA, yA, m);

Draw(xA, yA, x2, y2, m);

Draw(x2, y2, x3, y3, m);

Draw(x3, y3, xB, yB, m);

Draw(xB, yB, x4, y4, m);

Draw(x4, y4, x5, y5, m);

Draw(x5, y5, xC, yC, m);

Draw(xC, yC, x1, y1, m);

**end**;

**end**;

**procedure** instruction(m:integer);

**begin**

textout(10,100,'Left - влево');

textout(10,120,'Right - вправо');

textout(10,140,'Up - вверх');

textout(10,160,'Down - вниз');

textout(10,180,'A - увеличить масштаб');

textout(10,200,'S - уменьшить масштаб');

textout(10,220,'Z - увеличить глубину');

textout(10,240,'X - уменьшить глубину');

textout(10,260, abs(5-m));

textout(20,260, '- Глубина рекурсии');

**end**;

**begin**

**end**.

**Результаты выполнения программы**

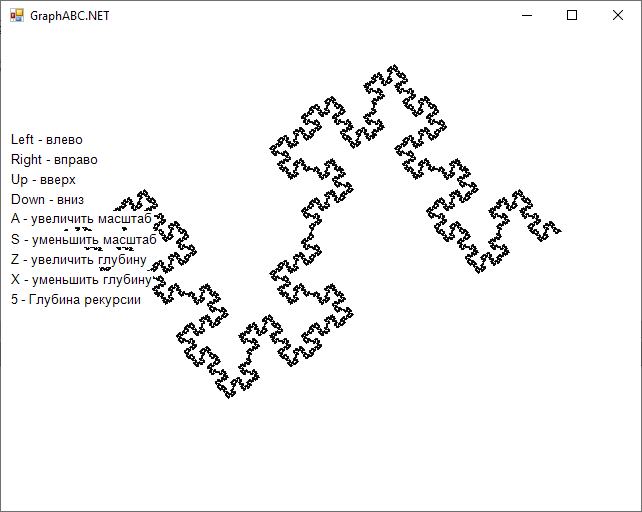


Рисунок 6 — Кривая Минковского

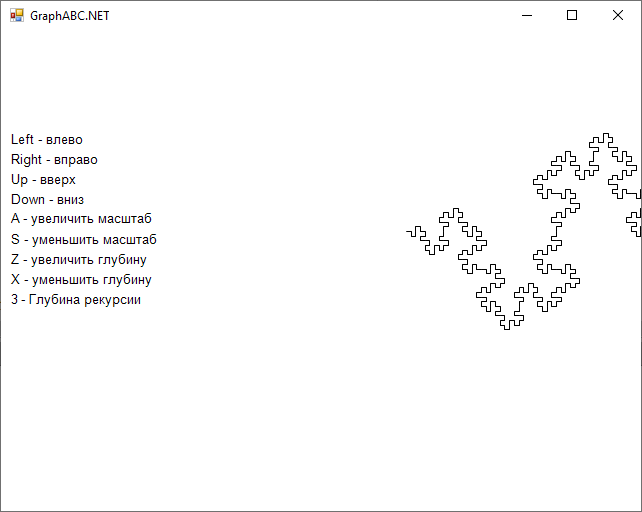


Рисунок 7 — Кнопка вправо

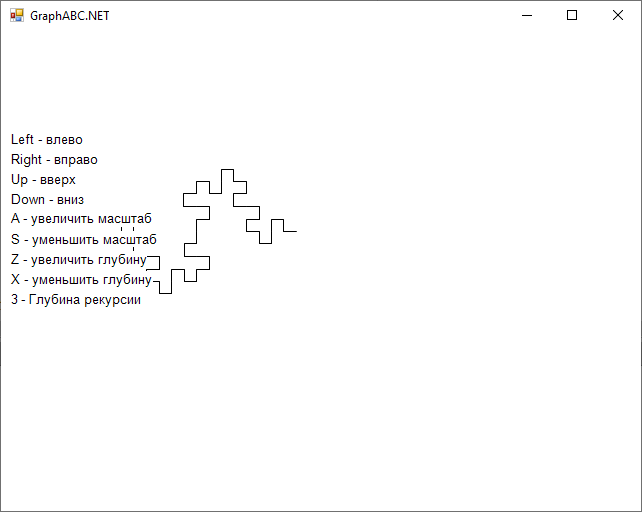


Рисунок 8— Кнопка влево

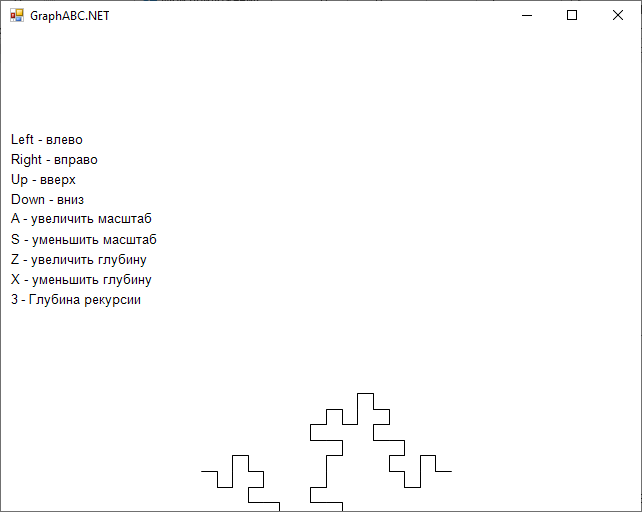


Рисунок 9 — Кнопка вниз

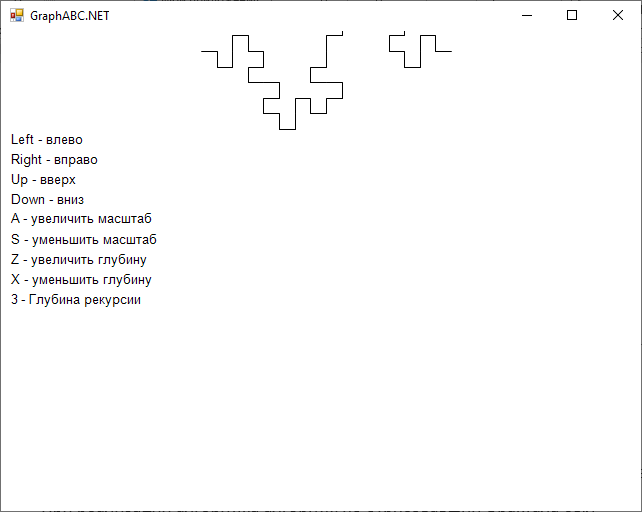


Рисунок 10 — Кнопка вниз

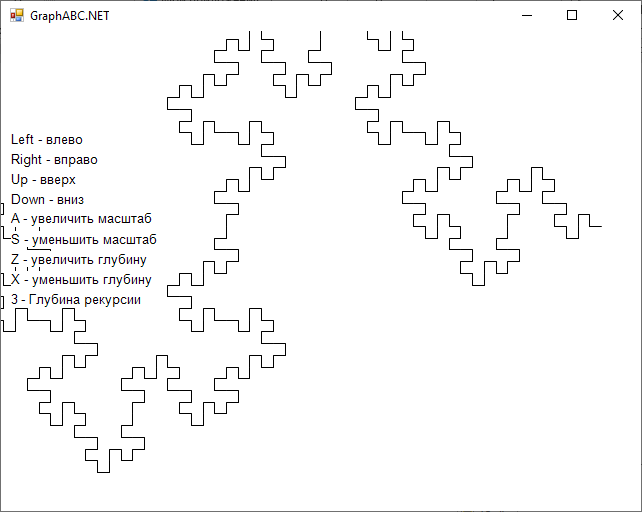


Рисунок 11 — Увеличение масштаба

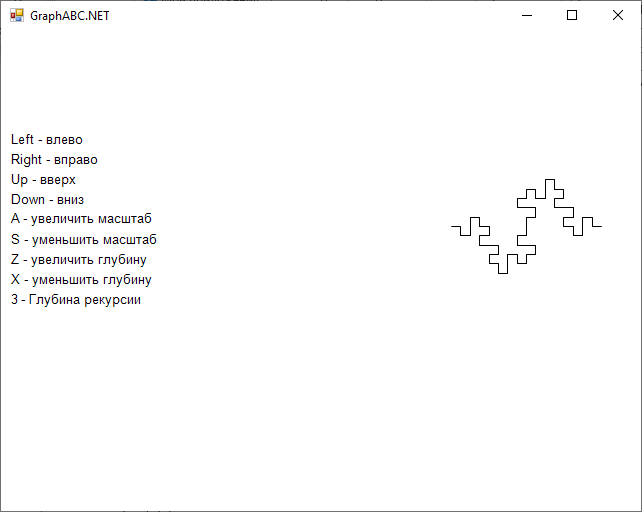


Рисунок 12 — Уменьшение масштаба

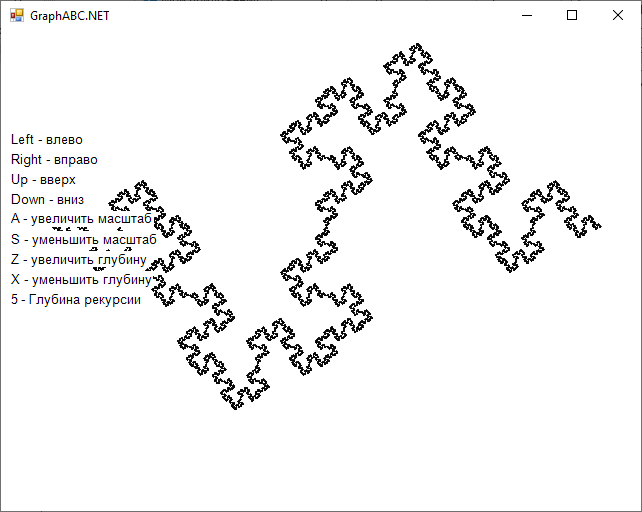


Рисунок 13 — Увеличение глубины

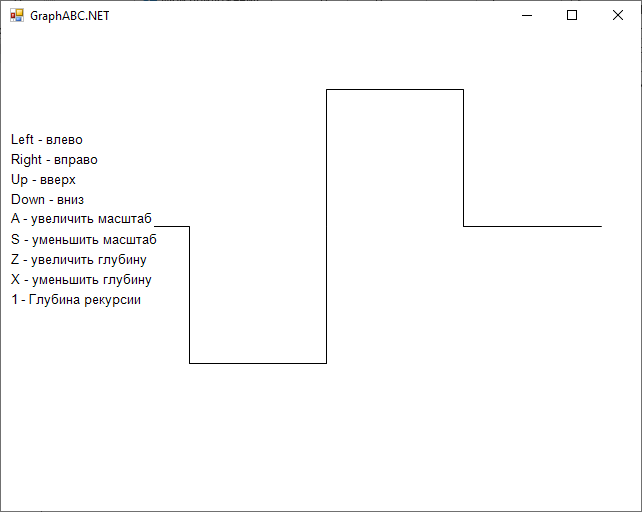


Рисунок 14 — Уменьшение глубины

**Вывод:**

В ходе работы был изучен рекурсивный алгоритм для построения фрактала. При реализации алгоритма алгоритм по отрисовавший фрактала был вынесен в отдельный модуль, что позволит использовать данный алгоритм в других программах. Также в программе реализовано управление: положением, масштабом и глубиной, с помощью кнопок. В ходе работы возникли проблемы с задачей масштабирования, а также с выделением базового случая.