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-204-52-00

Максимов Антон Николаевич

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2023

**Цель работы**

Получение навыков реализации алгоритмов с рекурсивными вычислениями, знакомство с фракталами.

**Задание**

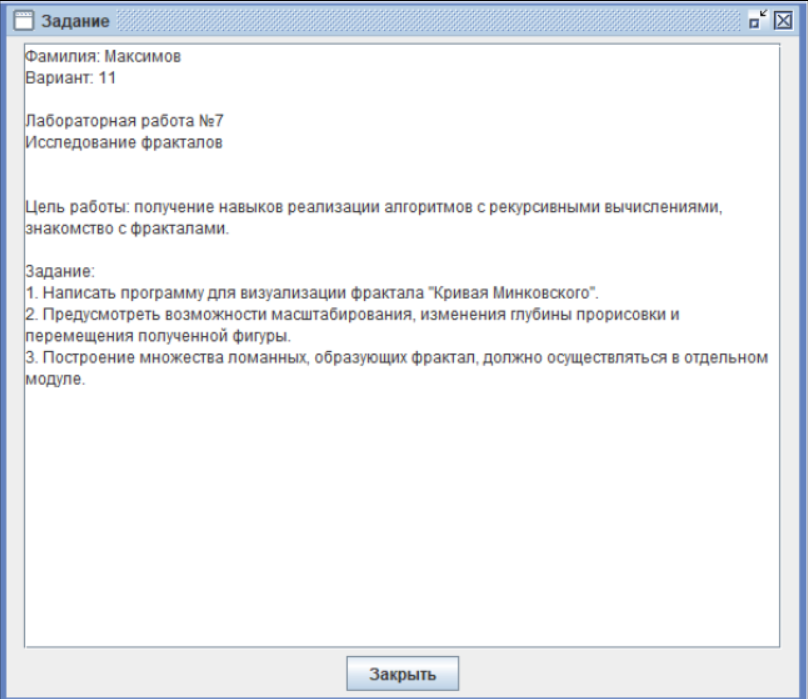


Рисунок 1 - Задание

**Описание алгоритма**

Программа вызывает процедуры для построения фрактала, которые находятся в отдельном модуле. Пользователю предоставляются следующие возможности: изменение масштаба, перемещение фрактала по координатной плоскости, изменение глубины прорисовки фрактала.

Изменение глубины осуществляется посредством рекурсивного вызова процедуры самой себя.

**Код программы**

**unit** Constr;

**interface**

**uses** GraphABC;

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

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

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

**function** Distance(x, y, x1, y1: real):real;

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

**implementation**

**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);

**begin**

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

**if** r < 10 **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);

Draw(xA, yA, x2, y2);

Draw(x2, y2, x3, y3);

Draw(x3, y3, xB, yB);

Draw(xB, yB, x4, y4);

Draw(x4, y4, x5, y5);

Draw(x5, y5, xC, yC);

Draw(xC, yC, x1, y1);

**end**;

**end**;

**end**.

**uses** GraphABC;

**uses** Constr;

**var** kof:real;

**var** b:byte;

**procedure** replace();

**begin**

Clearwindow;Draw(x,y,x1,y1);

textout(0,0,'Для перемещения по оси X введите 1');

textout(0,15,'Для перемещения по оси Y введите 2');

read(b);

**case** b **of**

1:**begin** Clearwindow;Draw(x,y,x1,y1);

textout(0,0,'Укажите расстояние для смещения');

read(kof);

x:=x+kof;

ClearWindow;

Draw(x,y,x1,y1);**end**;

2:**begin** Clearwindow;Draw(x,y,x1,y1);

textout(0,0,'Укажите расстояние для смещения');

read(kof);

ClearWindow;

y:=y+kof;

Draw(x,y,x1,y1);**end**;

**end**;

**end**;

**procedure** mastab();

**begin**

ClearWindow;

Draw(x,y,x1,y1);

textout(0,0,'Укажите коэффициент масштабирования');

read(kof);

x1:=x1\*kof;

ClearWindow;

Draw(x,y,x1,y1);

**end**;

**procedure** deep(n:integer);

**begin**

Draw(x,y,x1,y1);

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

deep(n-1);

**end**;

**end**;

**begin**

SetWindowCaption('Кривая Минковского');

CenterWindow;

SetWindowSize(800, 600);

x:=100;y:=250;

x1:=250;y1:=150;

Draw(x,y,x1,y1);

**var** o:=1;

**var** n:integer;

**while** o<>0 **do begin**

textout(0,550,'Для изменения масштаба введите 1, для перемещения 2, для изменения глубины 3, для завершения программы 0');

readln(o);

**case** o **of**

1: mastab;

2: replace;

3: **begin**

writeln('Укажите глубину прорисовки');

readln(n);

ClearWindow;

deep(n);

**end**;

0:**exit**();

**end**;

**end**;

**end**.

**Результат выполнения программы**

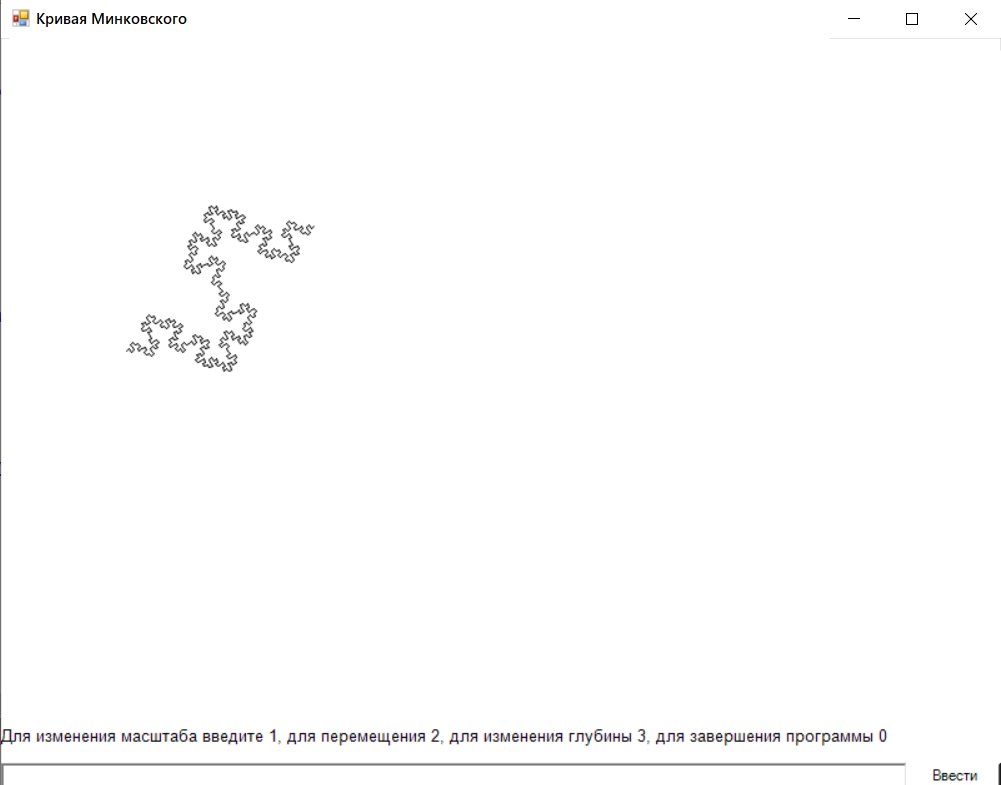


Рисунок 1 – Результат выполнения программы

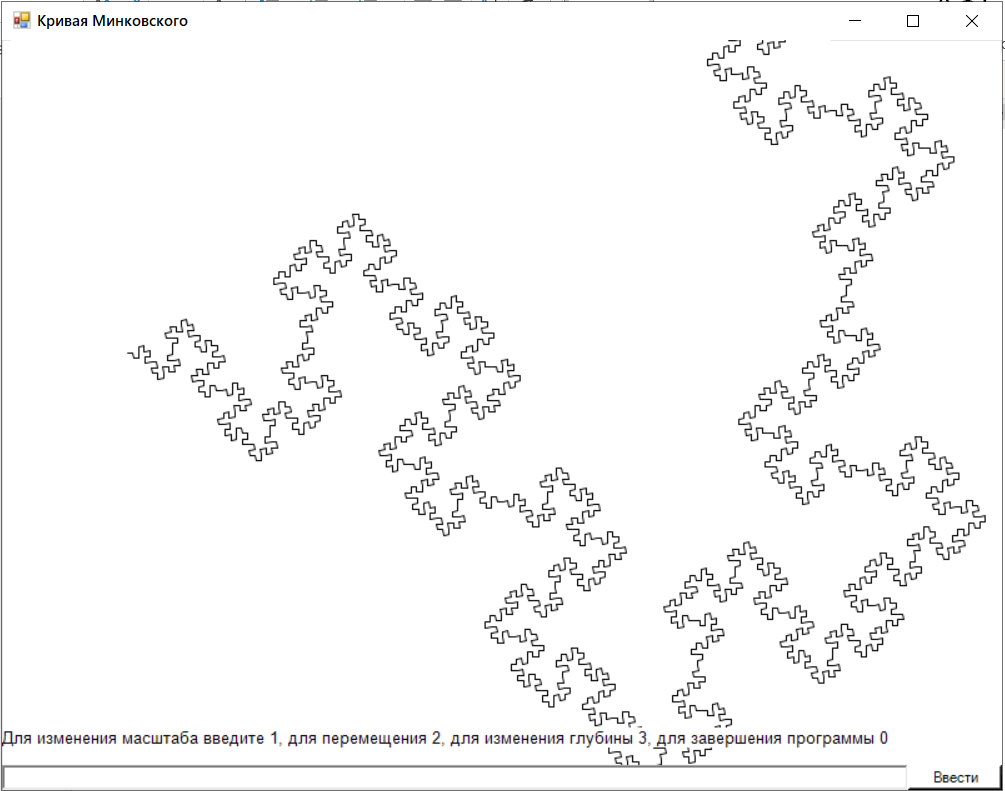


Рисунок 2 – Результат выполнения программы

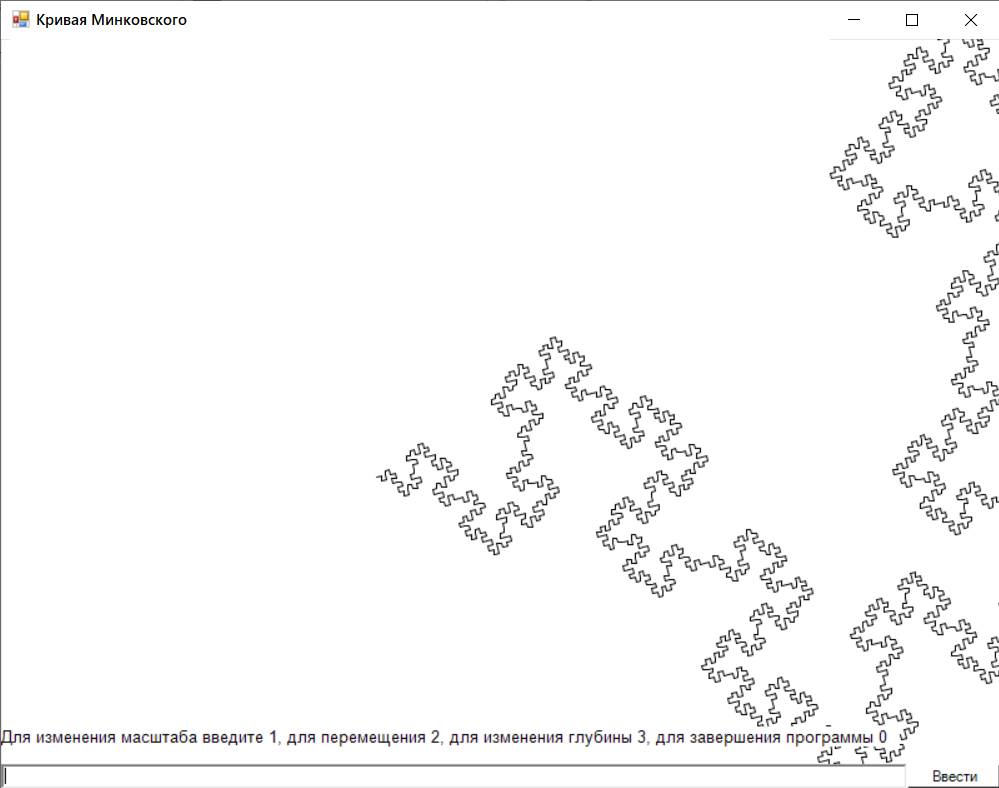


Рисунок 3 – Результат выполнения программы

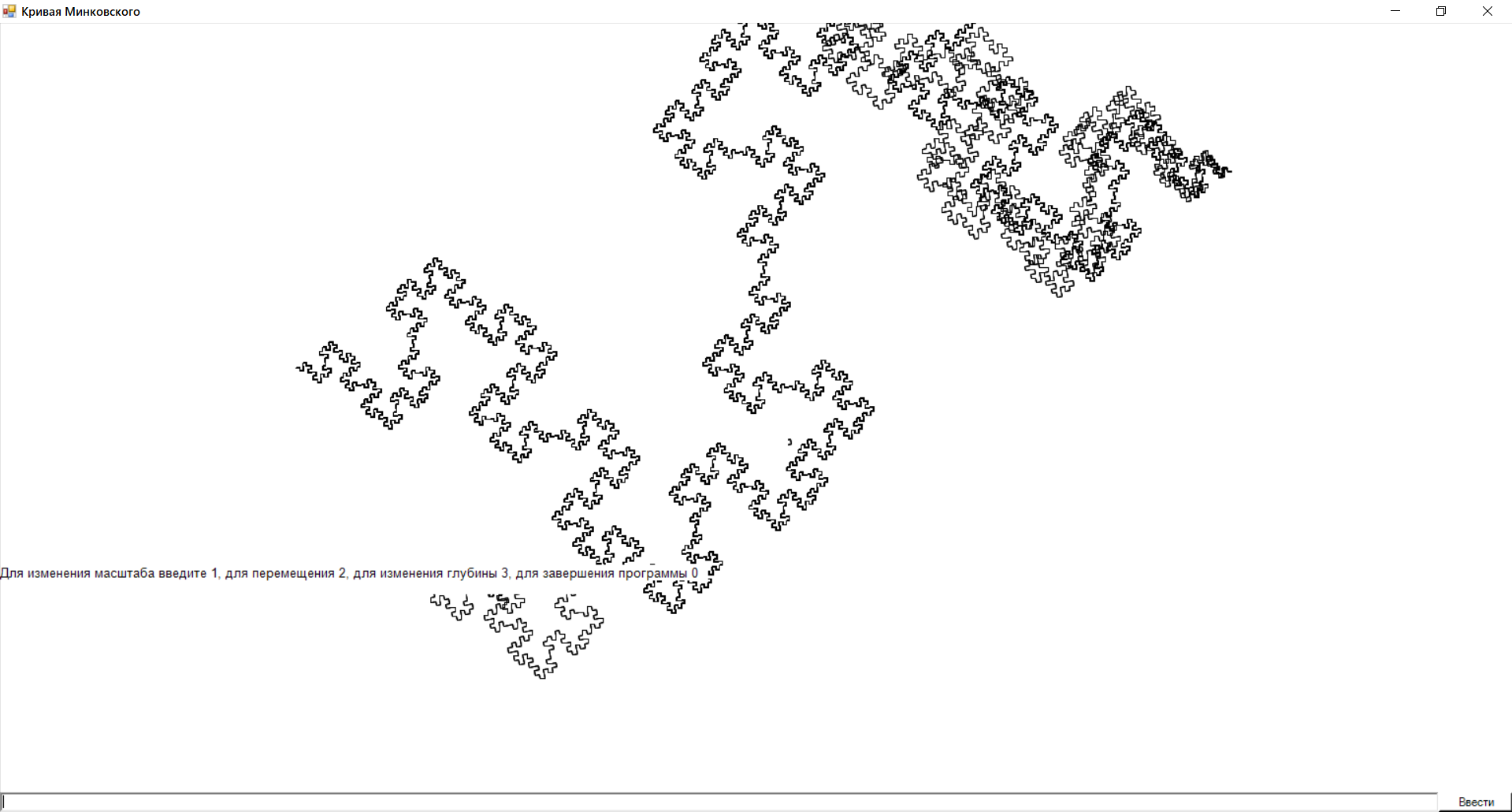


Рисунок 4 – Результат выполнения программы

**Вывод**

Для выполнения задания были использованы: библиотека GraphABC, процедуры и функции, был создан собственный модуль, который выполняет построение фрактала. Для изменения глубины прорисовки была использована рекурсия, находящаяся в процедуре и вызывающая сама себя. Рекурсия в процедуре вызывается с параметром процедуры n-1 пока n не равен 0, чтобы глубина прорисовки соответствовала значению, введённому пользователем.