УДК 004:372.8

Рекурсивная графика в школьной информатике

Е. В. Хламов
Институт развития образования Иркутской области, г. Иркутск

Аннотация. *Рассматриваются понятие и примеры рекурсии в различных областях знаний. Приводится обоснование ее эффективности как средства развития алгоритмического мышления школьников. Описываются рекурсивные алгоритмы построения известных геометрических фракталов как дидактический элемент школьного курса информатики.*

Ключевые слова: *информатика, рекурсия, фрактал, алгоритм, программма, язык программирования Паскаль, графические команды.*

Рекурсия является одним из основополагающих понятий математики и служит эффективным инструментом моделирования объектов и явлений реального мира. В общем смысле рекурсия есть определение объекта через его часть, являющуюся объектом этого же типа, но на более низком уровне. Подобные объекты являются довольно распространёнными во многих областях знаний. Рассмотрим некоторые примеры использования рекурсии в разных предметных областях.

Лингвистика, наряду с математикой, оспаривает пальму первенства в применении рекурсии. Достаточно полное описание рекурсивных методов в лингвистике может быть найдено у Е. А. Лодатко [4]. Пожалуй, самым известным примером является классическое произведение «Дом, который построил Джек». Менее популярным, но более значимым являются формальные грамматики, использующиеся в математической лингвистике для описания синтаксиса языков.

Самая впечатляющая иллюстрация рекурсии в физике – бесконечное изображение предмета в паре поставленных друг против друга зеркал.

В математике рекурсия лежит в основе такого направления, как теория вычислимых функций. Одним из постулатов теории является гипотеза Черча-Клини о том, что класс вычислимых функций совпадает с классом рекурсивных функций. Существует множество примеров использования рекурсии в классической математике. Так, например, известны рекурентные формулы для вычисления определённых интегралов, рекурентные формулы для решения систем линейных уравнений. Широко известны рекурсивные геометриче-

ские фигуры – фракталы, описанию которых посвящена данная статья.

Рекурсия в информатике является одним из основных методов разработки алгоритмов. Рекурсивные программы позволяют решить задачи, которые невозможно решить иными средствами. Если даже нерекурсивное решение и существует, то рекурсивное является заведомо более компактным и понятным. Рекурсия инициировала развитие таких методологий алгоритмизации, как «разделяй и властвуй» и «ветвей и границ», она обусловила развитие целого направления информатики – функционального программирования, наиболее известным представителем которого является язык Лисп.

Трудно переоценить дидактическое значение рекурсии. Ее значимость для развития алгоритмического мышления исследовалась А. Р. Есаяном [2], С. А. Бешенковым [1] и др. Основные выводы этих исследований следующие:

- обучение алгоритмизации на основе рекурсии является эффективным;
- развитие навыков рекурсивного программирования способствует более интенсивному формированию алгоритмического стиля мышления;
- обнаружен высокий познавательный интерес обучаемых к рекурсии как методу решения практических задач.

Одной из проблем активного обучения медодам рекурсивного программирования в школьном курсе информатики является состав учебного материала. В настоящее время база школьных рекурсивных алгоритмов исчерпывается числами Фибоначчи и вычислением факториала. Такой состав задач не может мотивировать учащихся на изучение рекурсии как метода эффективного программирования. Рекурсия воспринимается как упражнение для ума, не имеющее сколь-нибудь значимого практического применения. Для преодоления этого стереотипа необходимо использовать задачи, позволяющие получать понятный и эффектный результат при компактном рекурсивном решении. Все другие способы решения должны быть при этом более трудоёмкими при конструировании. Этим требованиям как нельзя лучше удовлетворяют задачи построения рекурсивных геометрических фигур, называемых фракталами.

Фракталы известны уже почти век, хорошо изучены [3; 5; 6] и имеют многочисленные приложения в жизни. В основе этого явления лежит очень простая идея: множество фигур можно получить из относительно простых конструкций при помощи всего двух операций: копирования и масштабирования.

У данного понятия нет строгого определения. Поэтому слово «фрактал» не является математическим термином. Обычно так называют геометрическую фигуру, которая удовлетворяет одному или нескольким из следующих свойств:

- обладает сложной структурой при любом увеличении;
- является (приближенно) самоподобной;
- может быть построена рекурсивными процедурами.

На рубеже XIX и XX веков изучение фракталов носило, скорее, эпизодический, нежели систематический характер, потому что раньше математики в основном изучали «хорошие» объекты, которые поддавались исследованию при помощи общих методов и теорий. В 1872 году немецкий математик Карл Вейерштрасс построил пример непрерывной функции, которая нигде не дифференцируема. Однако его построение было целиком абстрактно и трудно для восприятия. Поэтому в 1904 году швед Хельге фон Кох придумал непрерывную кривую, которая нигде не имеет касательной, причём её довольно просто нарисовать. Оказалось, что она обладает свойствами фрактала. Один из вариантов этой кривой носит название «снежинка Коха».

Идеи самоподобия фигур подхватил француз Поль Пьер Леви, будущий наставник Бенуа Мандельброта – основателя нового направления математики «Теория фракталов». В 1938 году вышла его статья «Плоские и пространственные кривые и поверхности, состоящие из частей, подобных целому», в которой описан ещё один фрактал – С-кривая Леви. Все вышеперечисленные фракталы можно условно отнести к одному классу конструктивных (геометрических) фракталов. Их успех среди нематематиков во многом обусловлен тем, что с помощью весьма простых конструкций и формул, которые способен понять и старшеклассник, получаются удивительные по сложности и красоте изображения. Когда персональные компьютеры стали достаточно мощными, появилось даже целое направление в искусстве – фрактальная живопись, причём заниматься ею мог практически любой владелец компьютера.

В предлагаемой статье приводится описание алгоритмов и программ на языке Паскаль для построения самых известных геометрических фракталов.

Кривая Гильберта

Эта линия была впервые построена выдающимся немецким математиком Давидом Гильбертом в 1891 году. Кривая Гильберта имеет вид, приведённый на рис. 1.

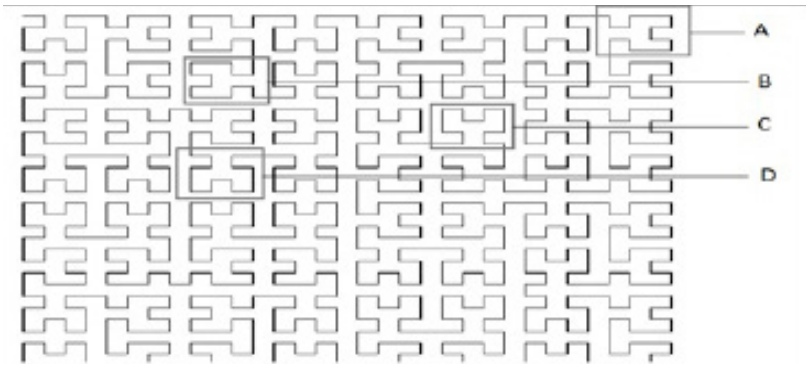

Рис. 1. Кривая Гильберта

Кривая полностью заполняет плоскость и не пересекается ни в одной точке. При ближайшем рассмотрении становится понятно, что кривая строится из четырёх различных элементов, соединённых разнонаправленными отрезками прямых. Эти элементы выделены на рис. 1 и обозначены как *A, B, C, D*. Заметим, что обозначение элементов носит сугубо произвольный характер. Заметим также, что можно выделить и более крупные фрагменты, которые будут повторяться при построении кривой, но это лишь усложнит разработку алгоритма.

Кривые *A, B, C, D* распадаются, в свою очередь, на более мелкие типовые фрагменты, похожие на разнонаправленную букву «П». Обозначим их *a, b, c, d*. Будем называть большие кривые кривыми второго порядка, а малые – кривыми первого порядка. Изображения кривых первого и второго порядка приведены на рис. 2.

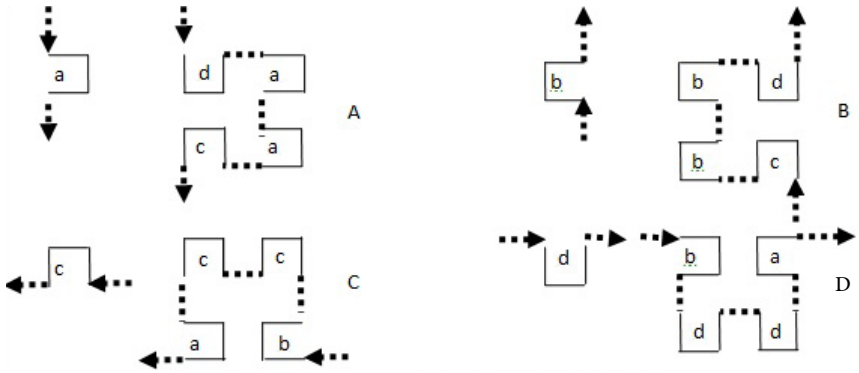

Рис. 2. Кривые Гильберта 1-го и 2-го порядка

Соответствие кривых первого порядка кривым второго порядка устанавливается по совпадению ориентации входа-выхода. Точка входа в каждую кривую второго порядка определяется из общего вида полной кривой Гильберта (рис.1) при условии, что её начало находится в левой верхней точке.

Разметим кривые второго порядка, то есть укажем, из каких кривых первого они состоят и в какой последовательности вычерчиваются. Разметка нанесена на рис.1. Теперь стало ясно, что кривые первых порядков получаются из соответствующих кривых вторых порядков, если в них оставить только соединительные отрезки (пунктиры).

Общий алгоритм построения кривых второго порядка с использованием кривых первого порядка с учётом разметки выглядит следующим образом:

Построение *A*:
Построить *d*
Построить соединительный отрезок
Построить *a*
Построить соединительный отрезок
Построить *a*
Построить соединительный отрезок
Построить *c*
Сокращённо это можно записать как *d-o-a-o-a-o-c*
Построение *B*: *c-o-b-o-b-o-d*
Построение *C*: *b-o-c-o-c-o-a*
Построение *D*: *d-o-b-o-b-o-c*

Очевидно, что этот алгоритм можно обобщить на случай кривой произвольного порядка К.

Пусть *A(k), B(k), C(k), D(k)* – кривые *k*-го порядка. Тогда алгоритмы построения кривых выглядят так:

Построение *A(K): D(k-1)–O–A(k-1)–O–A(k-1)–O–C(k-1)*
Построение *B(K): C(k-1)–O–B(k-1)–O–B(k-1)–O–D(k-1)*
Построение *C(K): B(k-1) –O–C(k-1) –O–C(k-1) – O–D(k-1)*
Построение *D(K): D(k-1) –O –B(k-1) –O–B(k-1) – O–C(k-1)*

Алгоритм является рекурсивным и использует как прямую, так и косвенную рекурсию.

Рассмотрим программу построения кривой Гильберта k-го порядка на Паскаль ABC.

```
PROGRAM GILBERT;
USES GraphABC;
VAR
X, Y, KU, R: INTEGER; // КООРДИНАТЫ ТЕКУЩЕЙ ТОЧКИ ДЛЯ ВЫЧЕРЧИВАНИЯ, ПОРЯДОК КРИВОЙ, ДЛИНА ОТРЕЗКА
PROCEDURE G_A(K: INTEGER); FORWARD; //ПОДПРОГРАММА ПОСТРОЕНИЯ КРИВОЙ A K-ГО ПОРЯДКА
PROCEDURE G_B(K: INTEGER); FORWARD; //ПОДПРОГРАММА ПОСТРОЕНИЯ КРИВОЙ B K-ГО ПОРЯДКА
PROCEDURE G_C(K: INTEGER); FORWARD; //ПОДПРОГРАММА ПОСТРОЕНИЯ КРИВОЙ C K-ГО ПОРЯДКА
PROCEDURE G_D(K: INTEGER); FORWARD; //ПОДПРОГРАММА ПОСТРОЕНИЯ КРИВОЙ D K-ГО ПОРЯДКА
PROCEDURE G_A(K: INTEGER);
BEGIN
    IF K> 0 THEN
    BEGIN
        //ПОСТРОЕНИЕ A(K): D(K-1) – O – B(K-1) – O – B(K-1) – O – C(K-1)
        G_D(K - 1); //ПОСТРОЕНИЕ КРИВОЙ D ПОРЯДКА K-1
        X := X + R;LINETO(X, Y); // ПОСТРОЕНИЕ СОЕДИНИТЕЛЬНОГО ОТРЕЗКА
        G_A(K - 1);Y := Y + R;LINETO(X, Y);
        G_A(K - 1);X := X - R;LINETO(X, Y);
        G_C(K - 1);
    END;
END;

PROCEDURE G_B(K: INTEGER);
BEGIN
    IF K> 0 THEN
    BEGIN
        //ПОСТРОЕНИЕ B(K): C(K-1) – O – B(K-1) – O– B(K-1) – O – D(K-1)
        G_C(K - 1);X := X - R;LINETO(X, Y);
        G_B(K - 1);Y := Y - R;LINETO(X, Y);
        G_B(K - 1);X := X + R;LINETO(X, Y);
        G_D(K - 1);
    END;
END;
PROCEDURE G_C(K: INTEGER);
BEGIN
    IF K> 0 THEN
    BEGIN
        //ПОСТРОЕНИЕ C(K): B(K-1) – O – C(K-1) – O – C(K-1) – O – D(K-1)
        G_B(K - 1);Y := Y - R;LINETO(X, Y );
        G_C(K - 1);X := X - R;LINETO(X, Y);
        G_C(K - 1);Y := Y + R;LINETO(X, Y);
        G_A(K - 1);
```

```
        END;
    END;

PROCEDURE G_D(K: INTEGER);
BEGIN
IF K> 0 THEN
BEGIN
    //ПОСТРОЕНИЕ D(K): D(K-1) – O – B(K-1) – O – B(K-1) – O – C(K-1)
    G_A(K - 1);Y := Y + R;LINETO(X, Y);
    G_D(K - 1);X := X + R;LINETO(X, Y);
    G_D(K - 1);Y := Y - R;LINETO(X, Y);
    G_B(K - 1);
  END;
END;
BEGIN
SetWindowSize(800, 600);
 SetWindowCaption('ФРАКТАЛ-КРИВАЯ ГИЛЬБЕРТА');
READLN(R); // ВВОД ДЛИНЫ ОТРЕЗКА
X := R; Y := R; MOVETO(X, Y); //УСТАНОВКА НАЧАЛЬНОЙ ТОЧКИ
                                ДЛЯ ПОЛНОЙ КРИВОЙ
READLN(KU); // ВВОД ПОРЯДКА КРИВОЙ
G_A(KU); // ПОСТРОЕНИЕ КРИВОЙ А ПОРЯДКА KU
END.
```

Кривая Дракона

Кривая дракона высоких порядков имеет вид, приведённый на рис 3.

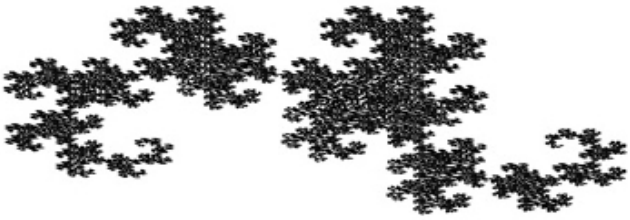

Рис. 3. Кривая дракона порядка 16

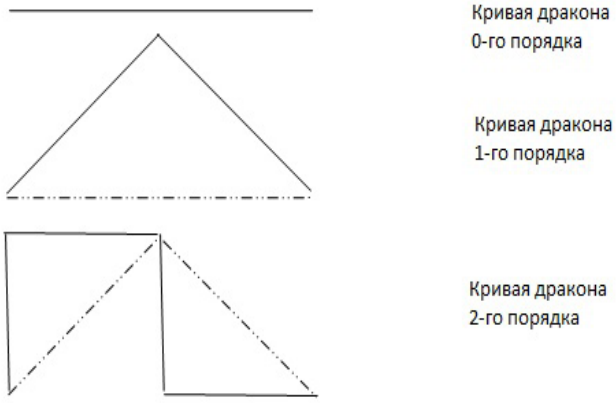

Рис. 4. Кривые дракона низких порядков

Для того чтобы понять, какие элементы используются при построении этой кривой, рассмотрим линии более низких порядков (рис. 4).

Алгоритм построения кривой сводится к построению равнобедренных треугольников на заданном отрезке прямой и затиранию основания, то есть нужно на основе кривой *0*-го порядка постро-

ить кривую *1*-го порядка, а на основе кривой *1*-го прядка нужно построить кривую *2*-го порядка. Если продолжить этот ряд, то любая кривая *k*-го порядка требует построения кривой (*k-1*)-го порядка.

Схематично последовательность рекурсивных вызовов выглядит так:

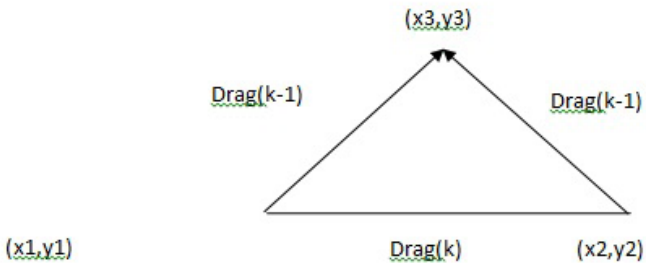

Рис. 5. Схема рекурсивных вызовов

Отметим различную ориентацию отрезков для двух кривых порядка *k-1*. Это объясняется асимметричностью кривой дракона, понятной из изображения линии *2*-го порядка (рис. 4).

Пусть *(x1,y1)* и *(x2,y2)* – начало и конец отрезка кривой *k*-го порядка. Тогда общий алгоритм её построения кривой выглядит следующим образом:

Drag(x1,y1, x2,y2,k):

Для k>0: Drag(x1,y1, x3,y3,k-1), Drag(x2,y2, x3,y3,k-1)

Для k=0: отрезок (x1,y1)-(x2,y2)

Теперь остаётся только определить значения (*x3, y3*). Для этого рассмотрим кривую *2*-го порядка на рис. 6, разбитую для удобства на два фрагмента: «верхний угол» и «нижний угол».

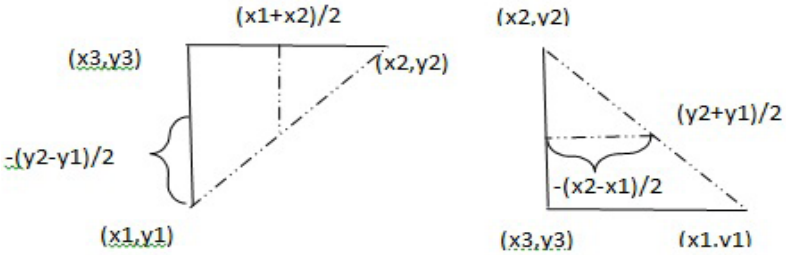

Рис. 6. Определение точки деления (x3, y3)

Сложность определения точки (*x3, y3*) состоит в том, что нужно построить общую формулу для координат и верхнего, и нижнего углов.

Это формулы вида:

x3=(x1+x2)/2+(y2-y1)/2

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

Программа построения кривой дракона на языке Паскаль ABC:

PROGRAM DRAGON;

USES GRAPHABC;

VAR K: INTEGER;

```
PROCEDURE DRAG(X1, Y1, X2, Y2, K:INTEGER);
VAR X3, Y3:INTEGER;

BEGIN
IF K>0 THEN
BEGIN
        XN:=(X1+X2) DIV 2 +(Y2-Y1) DIV 2;
        YN:=(Y1+Y2) DIV 2 -(X2-X1) DIV 2;
        DRAG (X1,Y1,X3,Y3,K-1);
        DRAG (X2,Y2,X3,Y3,K-1);
   END
ELSE LINE(X1,Y1,X2,Y2); // КРИВАЯ ПОРЯДКА 0
END;
BEGIN
      READLN ( K );
      DRAG(200,300,500,300,K);
END.
```

Кривая Леви

Эта кривая предложена французским математиком Пьером Леви и имеет общий вид, приведённый на рис.7.

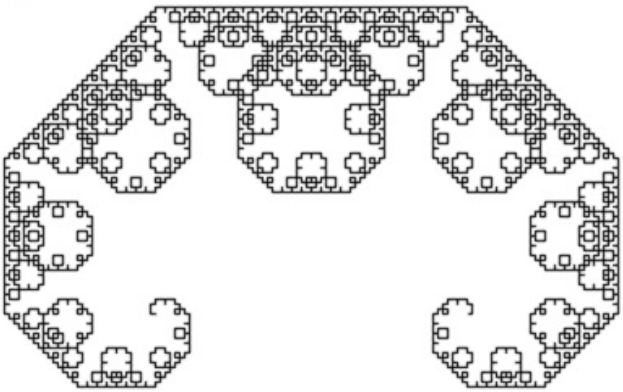

Рис. 7. Кривая Леви порядка 12

Кривая состоит из элементов, приведённых ниже на рис. 8.

Рис. 8. Кривые Леви низких порядков

Кривую можно построить по рекурсивному алгоритму, так как кривая *L(k)*

порядка *k* строится на основе кривой *L(k-1)* порядка *k-1*.

Алгоритм можно построить, если рассмотреть процесс построения кривой *2*-го порядка на основе кривой *1*-го порядка. Схема этого преобразования приведена на рис. 9

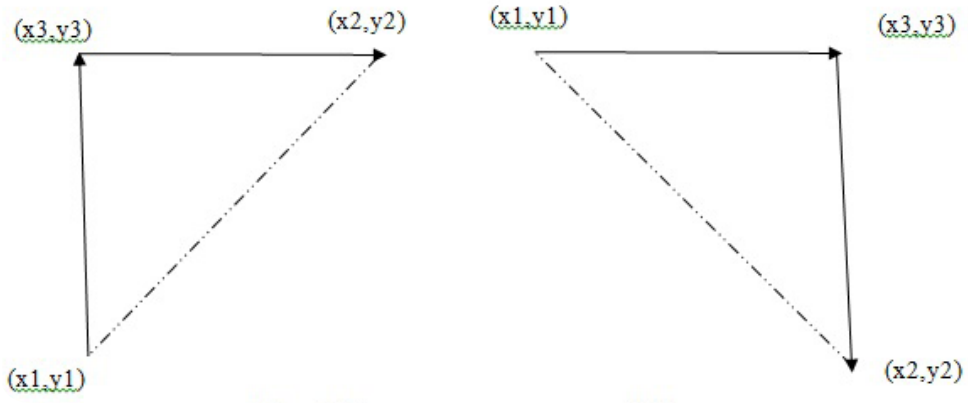

Рис. 9. Схема построения кривой Леви

Это напоминает построение кривой дракона, с той лишь разницей, что кривая Леви имеет вертикальную симметрию. Общий алгоритм построения кривой в этом случае имеет следующую структуру.

L(x1, y1, x2, y2, k):
Для k>0, L(x1, y1, x3, y3, k-1)
L(x3, y3, x2, y2, k-1)
Для k=0, отрезок(x1, y1, x2, y2)

Остаётся определить формулы для вычисления *x3, y3* таким образом, чтобы они вычисляли координаты и левого, и правого угла кривой.

x3:=(x1+x2) /2+(y2-y1) /2
y3:=(y1+y2) /2-(x2-x1) /2

Программа построения кривой Леви на языке Паскаль ABC выглядит следующим образом:

```
PROGRAM LEVY;
USES GraphABC;
VAR K:INTEGER;
PROCEDURE L(X1,Y1,X2,Y2,K:INTEGER);
VAR X3,Y3:INTEGER;
BEGIN
 IF K>0 THEN
 BEGIN
 X3:=(X1+X2) DIV 2 +(Y2-Y1) DIV 2;
 Y3:=(Y1+Y2) DIV 2 -(X2-X1) DIV 2;
 L(X1,Y1,X3,Y3,K-1);
 L(X3,Y3,X2,Y2,K-1);
END
 ELSE LINE(X1,Y1,X2,Y2);
END;
BEGIN
```

```
        READLN ( K );
        L(200,300,500,300,K);
READLN;
END
```

Рассмотрим ещё один алгоритм построения кривой Леви. В отличие от предыдущего он будет использовать полярные координаты. Схема построения кривой приведена на рис. 10.

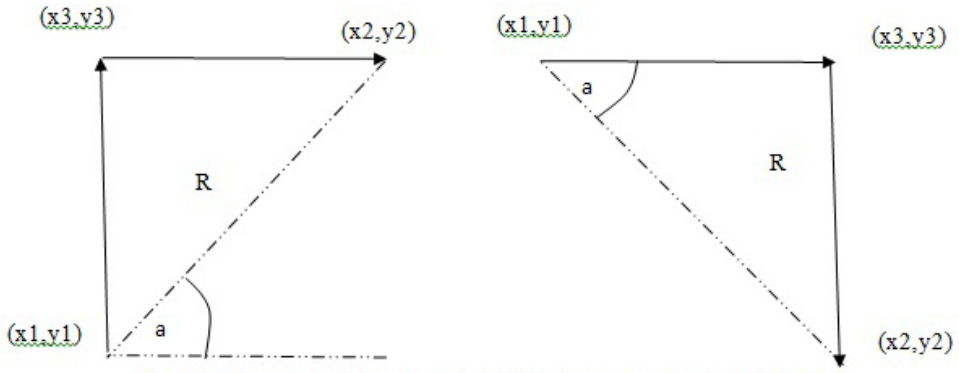

Рис. 10. Схема построения кривой Леви в полярных координатах

Обозначим через процедуру построения кривой Леви *k*-го порядка из точки (x,y), длины R. Очевидно, что для её построения нужно построить две кривые: *L(x1,y1,a+45°, Rcos(45°), k-1) и L(x3, y3, a - 45°, Rcos(45°), k-1)*. Для левого угла это будут отрезки, направленные под углами 90° и 0°, а для правого угла – отрезки, направленные под углами 0° и -90°. При этом координаты точки (*x3, y3*) вычисляются следующим образом:

x3=x1+Rcos(a + 45°)

y3=y1- Rsin(a + 45°)

Общий алгоритм построения кривой Леви k-го порядка имеет вид:

L(x1,y1,a,R,k):

Для k>0, L(x1, y1, a + 45°, Rcos(45°), k-1);

L(x1+Rcos(a + 45°), y1- Rsin(a + 45°), a - 45°,k-1)

Для k=0, отрезок (x1, y1, x2, y2)

Программа построения кривой Леви с использованием полярных координат на языке Паскаль ABC выглядит следующим образом:

```
PROGRAM LEVY;
USES GRAPHABC;
VAR
K: INTEGER;
PROCEDURE L(X, Y, A, R: REAL; K: INTEGER);
BEGIN
 IF K > 0 THEN
BEGIN
    R:= R * SIN(PI / 4);
    L (X, Y, A + PI / 4, R, K - 1);
    L (X + R * COS(A + PI / 4), Y - R * SIN(A + PI / 4), A - PI / 4, R, K - 1);
```

```
END
ELSE
LINE(TRUNC(X), TRUNC(Y), TRUNC(X + R * COS(A)), TRUNC(Y - R * SIN(A)));
END;
BEGIN
 READLN(K);
 L (200, 300, 0, 250, K);
        END.
```

Снежинка Коха

Эта фрактальная кривая описана в 1904 году шведским математиком Хельге фон Кохом. Общий вид кривой Коха порядка 8 приведён ниже на рис. 11.

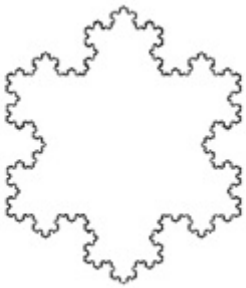

Рис. 11. Снежинка Коха 8-го порядка

Рассмотрим элементы кривой Коха. Это, как обычно, кривые порядка 0, 1, 2. Их изображение приведено на рис.12.

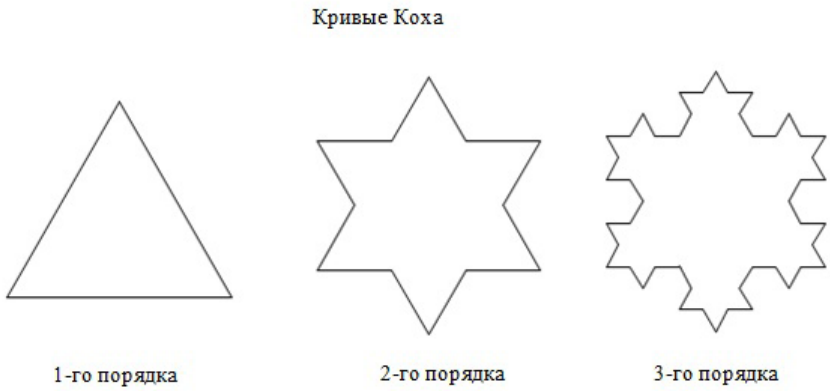

Рис. 12. Снежинки Коха низких порядков

Построение кривой сводится к делению каждого отрезка кривой на три части и построению на средней части равностороннего треугольника без основания. Алгоритм является рекурсивным, так как для построения кривой порядка *k* требуется построение кривой порядка *k*-1. Для разработки общего алгоритма рассмотрим кривую *2*-го порядка на основе кривой *1*-го порядка. Для постро-

ения снежинки будем использовать полярные координаты. Схема построения кривой второго порядка показана на рис.13. Для наглядности стороны треугольника на рисунке разнесены друг от друга.

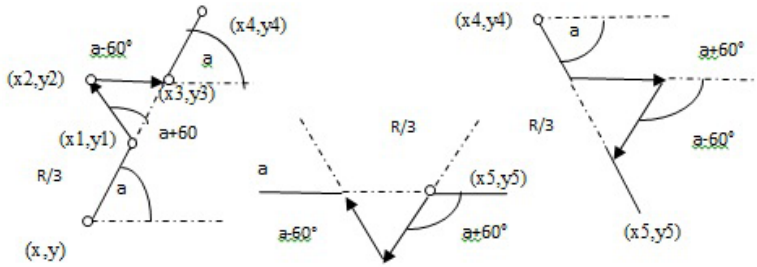

Рис. 13. Схема построения снежинки Коха

Обозначим за *KOX(x, y, a, R, k)* процедуру вычерчивания кривой Коха k-го порядка длины, начиная с точки *(x, y)*, под углом *a*.

Тогда общий алгоритм построения кривой Коха будет выглядеть следующим образом:

KOX(x, y, a, R, k):
Для k>0 *R=R/3;*
KOX(x, y, a,R,k-1);
KOX(x1, y1, a+60°, R, k-1);
KOX(x2, y2, a-60°, R, k-1);
KOX(x3, y3, a, R, k-1);
Для k=1 *Отрезок(x, y, x4, y4)*

Запуск процедуры производится отдельно для каждого из трёх фрагментов снежинки:

KOX(x, y, 60°, R, k-1);
KOX(x4, y4, -60°, R, k-1);
KOX(x5, y5, -180°, R, k-1);

Программа построения снежинки Коха на языке Паскаль ABC выглядит следующим образом:

```
PROGRAM KOKH_SNOW;
USES
GraphABC;
VAR
X0, Y0, R0: REAL;
KT: INTEGER; // ПОРЯДОК КРИВОЙ
PROCEDURE KOX(X, Y, A, R: REAL; N: INTEGER);
BEGIN
 IF N > 0
THEN
    BEGIN
      R:= R / 3;
      KOX(X, Y, A, R, N - 1);
      X:= X + R * COS(A);Y := Y - R * SIN(A);
```

```
        KOX(X, Y, A + PI / 3, R, N - 1);
        X:= X + R * COS(A + PI / 3);Y := Y - R * SIN(A + PI / 3);
        KOX(X, Y, A - PI / 3, R, N - 1);
        X:= X + R * COS(A - PI / 3);Y := Y - R * SIN(A - PI / 3);
        KOX(X, Y, A, R, N - 1);
      END
    ELSE
     LINE(TRUNC(X), TRUNC(Y), TRUNC(X + R * COS(A)), TRUNC(Y - R *
SIN(A)));
    END;
    BEGIN
        SetWindowSize(800, 600);
        SetWindowCaption('ФРАКТАЛЫ: СНЕЖИНКА КОХА');
        R0 := 300;X0 := 400 - R0 / 2;Y0 := 300 + R0 / 2;
        READLN(KT);
        KOX(X0, Y0, PI / 3, R0, KT - 1);
        KOX(X0 + R0 * COS(PI / 3), Y0 - R0 * SIN(PI / 3), -PI / 3, R0, KT - 1);
        KOX(X0 + R0, Y0, -PI, R0, KT - 1);
    END.
```

Дерево Пифагора

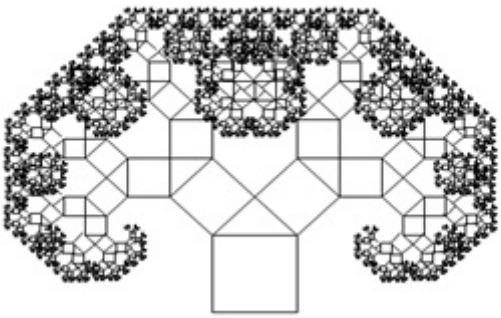

Рис. 14. Дерево Пифагора 12-го порядка

Кривая названа в честь древнегреческого математика Пифагора, так как строится из элементов, известных как «пифагоровы штаны». Вид кривой порядка 12 приведён на рис.14.

Для выделения элементов рассмотрим деревья *1*-го, *2*-го порядков. Их изображения приведены на рис.15.

Для построения алгоритма рассмотрим дерево Пифагора *2*-го порядка. Очевидно, что оно использует в своём построении дерево *1*-го порядка, которое выглядит как квадрат. Будем строить дерево с использованием полярных координат. Детальная схема построения кривой порядка *2* на основе кривой порядка *1* приведена на рис.16.

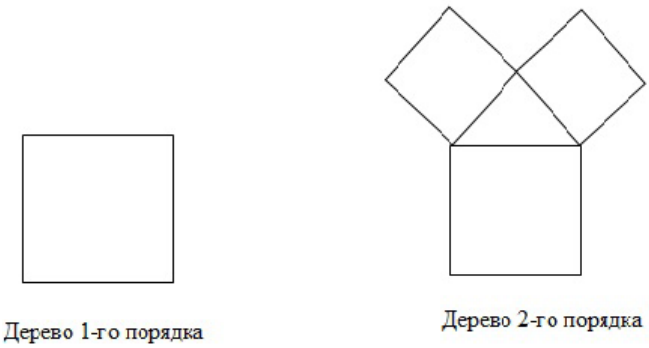

Рис. 15. Деревья Пифагора низких порядков

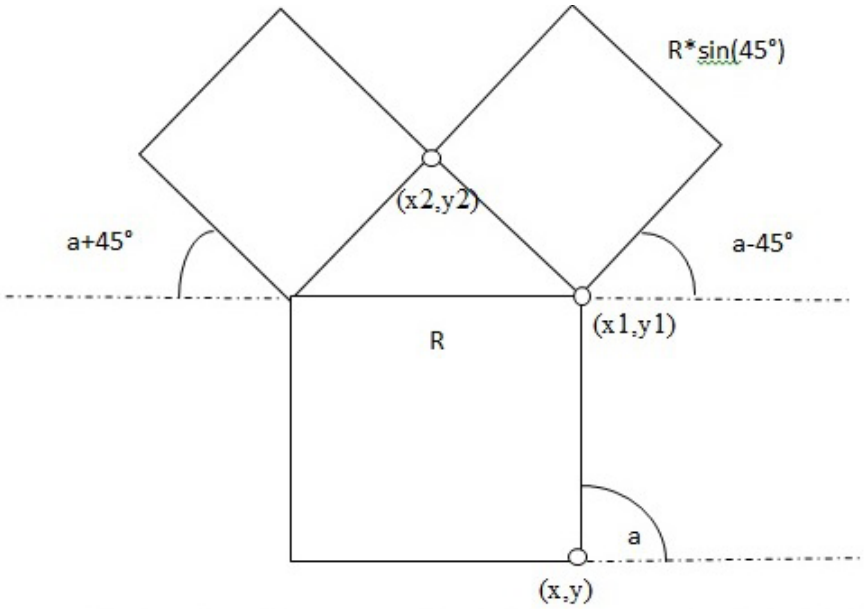

Рис. 16. Схема построения дерева Пифагора второго порядка

Для того чтобы построить дерево *2*-го порядка из точки *(x,y)*, надо построить два дерева *1*-го порядка из точек *(x1,y1)* и *(x2,y2)* и одно дерево *1*-го порядка из точки *(x,y)*. Причём эти деревья отличаются друг от друга как длиной стороны квадрата, так и его направлением.

Обозначим процедуру построения дерева Пифагора *k*-го порядка *PIF (x,y,R,a,k)*. Здесь *x, y* – координаты начала дерева, *R* – длина стороны квадрата, a – угол направления квадрата, *k* – порядок кривой. Тогда общий алгоритм построения дерева Пифагора выглядит следующим образом:

PIF (x,y,R,a,k):
Для k>0

*R1=R*sin(45°);x1:=x + R*cos(a); y1:=y - R*sin(a); a1:=a - 45°;*
PIF(x1, y1, R1, a1 , k - 1);
*x2:=x + R*cos(a)+R1*cos(a +45°);*
*y2:=y - R*sin(a)-R1*sin(a +45°), a1:=a +45°;*
PIF(x2, y2, R1, a1 , k - 1);

Для k>=0 Построить квадрат с началом в точке (x,y), стороной R и направлением *a*.

Программа построения дерева Пифагора на языке Паскаль ABC выглядит следующим образом:

```
PROGRAM PIFAGOR;
USES GraphABC;
VAR
R, X, Y: REAL; // КООРДИНАТЫ И РАДИУС
K: INTEGER //ПОРЯДОК КРИВОЙ

PROCEDURE PIF(XX, YY, RR, A: REAL; K: INTEGER);
VAR X1,Y1,R1,A1:REAL;
```

```
BEGIN
IF (K > 1) AND (RR > 2) THEN
BEGIN
R1:=RR * SQRT(2) / 2;
X1:=XX + RR * COS(A); Y1:=YY - RR * SIN(A); A1:=A - PI / 4;
PIF(TRUNC(X1), TRUNC(Y1), R1, A1 , K - 1);
X1:= XX + RR * COS(A) + RR * SQRT(2) / 2 * COS(A + PI / 4);
Y1:=YY - RR * SIN(A) - RR * SQRT(2) / 2 * SIN(A + PI / 4); A1:=A + PI / 4;
PIF(TRUNC(X1), TRUNC(Y1), R1, A1, K - 1);
END;

MOVETO(TRUNC(XX), TRUNC(YY));
LINETO(TRUNC(XX - RR * SIN(A)), TRUNC(YY - RR * COS(A)));
XX := TRUNC(XX - RR * SIN(A));YY := TRUNC(YY - RR * COS(A));
LINETO(TRUNC(XX + RR * COS(A)), TRUNC(YY - RR * SIN(A)));
XX := TRUNC(XX + RR * COS(A));YY := TRUNC(YY - RR * SIN(A));
LINETO(TRUNC(XX + RR * SIN(A)), TRUNC(YY + RR * COS(A)));
XX := TRUNC(XX + RR * SIN(A));YY := TRUNC(YY + RR * COS(A));
LINETO(TRUNC(XX - RR * COS(A)), TRUNC(YY + RR * SIN(A)));
END;
BEGIN
    SETWINDOWSIZE(800, 600); // РАЗМЕР ОКНА
    SETWINDOWCAPTION('ФРАКТАЛ- ДЕРЕВО ПИФАГОРА. ВВЕДИТЕ КОЛИЧЕСТВО УРОВНЕЙ');
    X := WINDOWWIDTH / 2;
    R := 100;
    X := 400 + R / 2;Y := 500;
    READLN(K);
    PIF(X, Y, R, PI / 2, K);
    READLN(K);
    END.
```

Список литературы

1. Бешенков, С. А. Развитие содержания обучения информатике в школе на основе понятий и методов формализации : дис. … д-ра пед. наук : 13.00.02 / С. А. Бешенков. – М., 1994. – 224 с.
2. Есаян, А. Р. Обучение алгоритмизации на основе рекурсии : учеб. пособ. для студ. пед. вузов / А. Р. Есаян. – Тула : Тул. гос. пед. ун-т им. Л. Н. Толстого, 2001. – 216 с.
3. Кроновер, Ричард М. Фракталы и хаос в динамических системах / Ричард М. Кроновер. – М. : Техносфера, 2006. – 352 с.
4. Лодатко, Е. А. Рекурсивные лингвистические структуры / Е. А. Лодатко // Теоретические и прикладные проблемы русской филологии – Славянск: СГПУ, 2004. – С. 86–95.
5. Мандельброт, Б. Фракталы, случай и финансы / Б. Мандельброт. – М. : Регулярная и хаотическая динамика, 2004. – 341 с.
6. Мандельброт, Б. Фрактальная геометрия природы / Б. Мандельброт. – М. : Институт компьютерных исследований», 2004. – 425 с.

Recursive Graphics in School Informatics

E. V. Khlamov
Institute of Education Development of Irkutsk Region, Irkutsk

Abstract. *In the article the notion of recursion and its examples in different spheres of knowledge are concerned. The author proves the effectiveness of recursion as a means of schoolchildren algorithmic way of thinking. The recursive algorithms of making well-known geometric fractals as didactic elements of School course of Informatics are described.*

Keywords: *Informatics, recursion, fractal, algorithm, programme, Pascal language, graphic commands.*

Хламов Евгений Валерьевич
кандидат физико-математических наук, доцент кафедры предметной области

Институт развития образования Иркутской области
664023, г. Иркутск, ул. Красноказачья, 10-А
тел.: 8 (3952)500904
e-mail: e.khlamov@iro38.ru

Khlamov Evgeniy Vladimirovich
Candidate of Sciences (Physics and Mathematics), Associate Professor, Associate Professor of the School Subjects Department

Institute of Educational Development of Irkutsk Region
10-A Krasnokazachiya St, Irkutsk, 664007
tel.: 8(3952)207602
e-mail: e.khlamov@iro38.ru