# Дополнительные главы компьютерной графики

Лекция 10


http://www.uni-vologda.ac.ru/~c3c

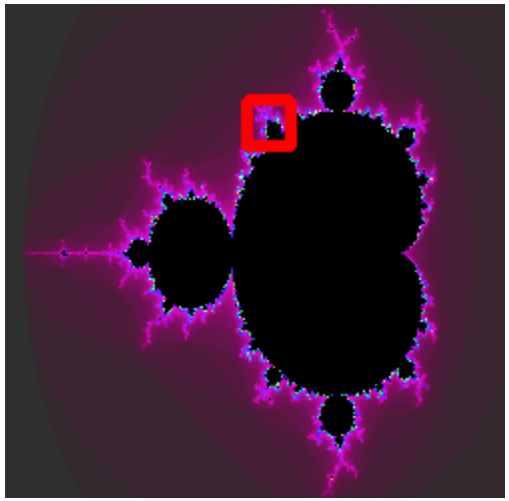
# Фракталы и *L*-системы

# Фракталы и *L*-системы

**Фракта́л** (лат. *fractus* — дроблёный, сломанный, разбитый) — математическое множество, обладающее свойством самоподобия (объект, в точности или приближённо совпадающий с частью себя самого, то есть целое имеет ту же форму, что и одна или более частей)

# Фракталы и *L*-системы

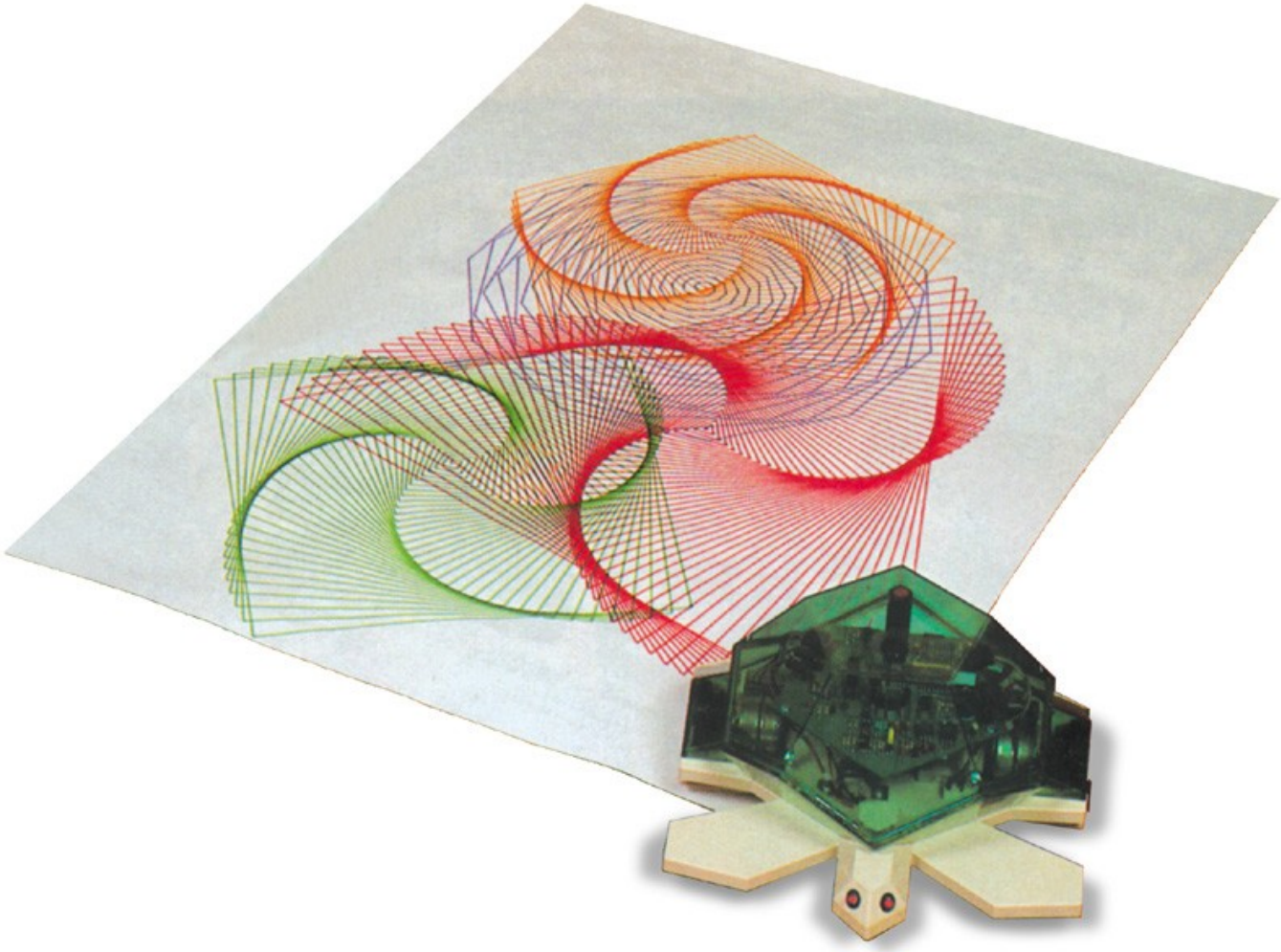
**Фракта́л** (лат. *fractus* — дроблёный, сломанный, разбитый) — математическое множество, обладающее свойством самоподобия (объект, в точности или приближённо совпадающий с частью себя самого, то есть целое имеет ту же форму, что и одна или более частей)

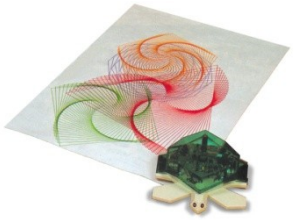


Множество Мандельброта

Фрактальное растение

# Черепашья графика





# Команды черепашьей графики

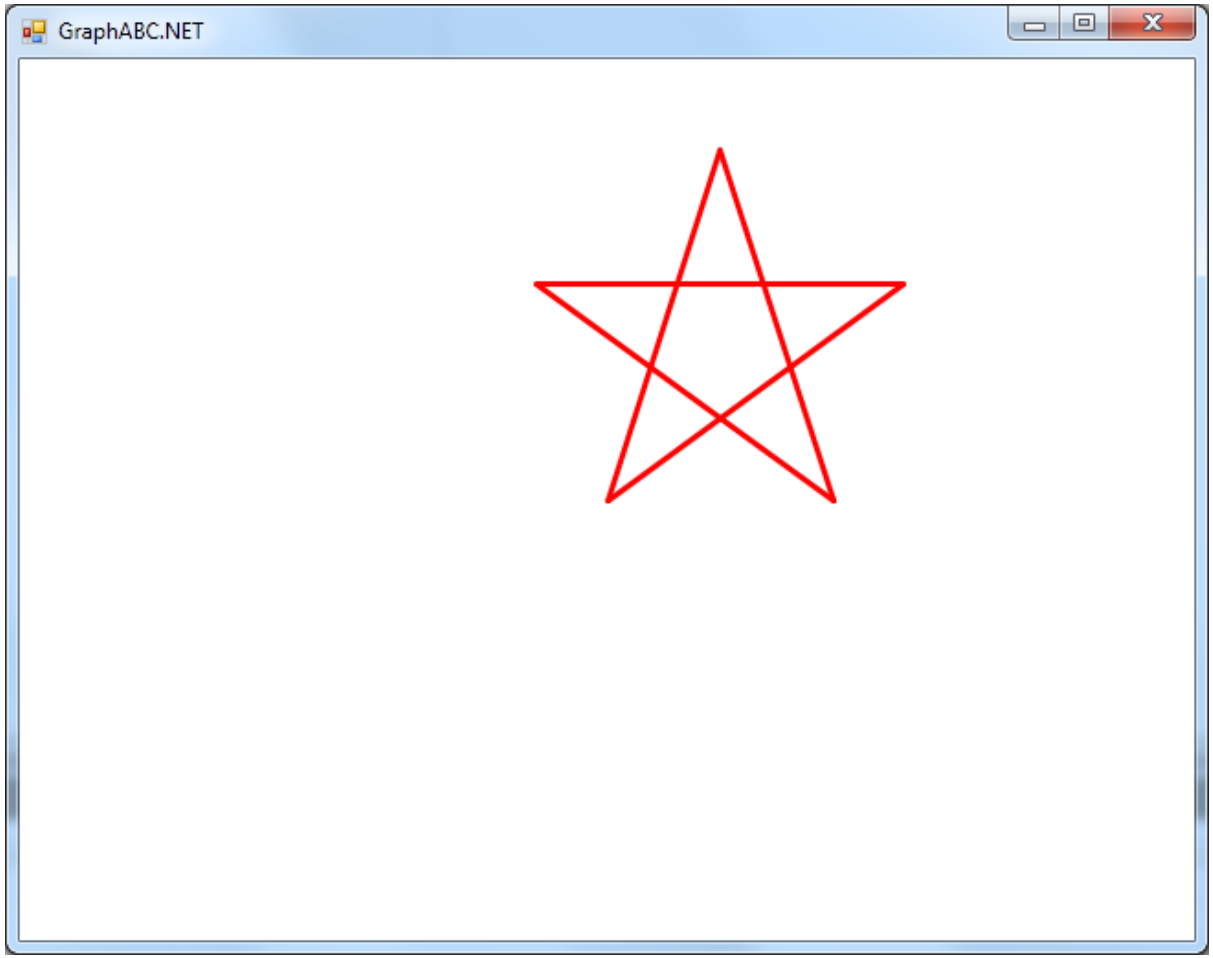
- Вперед на N шагов: Go(N);
- Поворот направо на M градусов: Right(M);
- Поворот налево на M градусов: Left(M);
- Опустить перо: Down;
- Поднять перо: Up;
- Исходное положение: Stand;

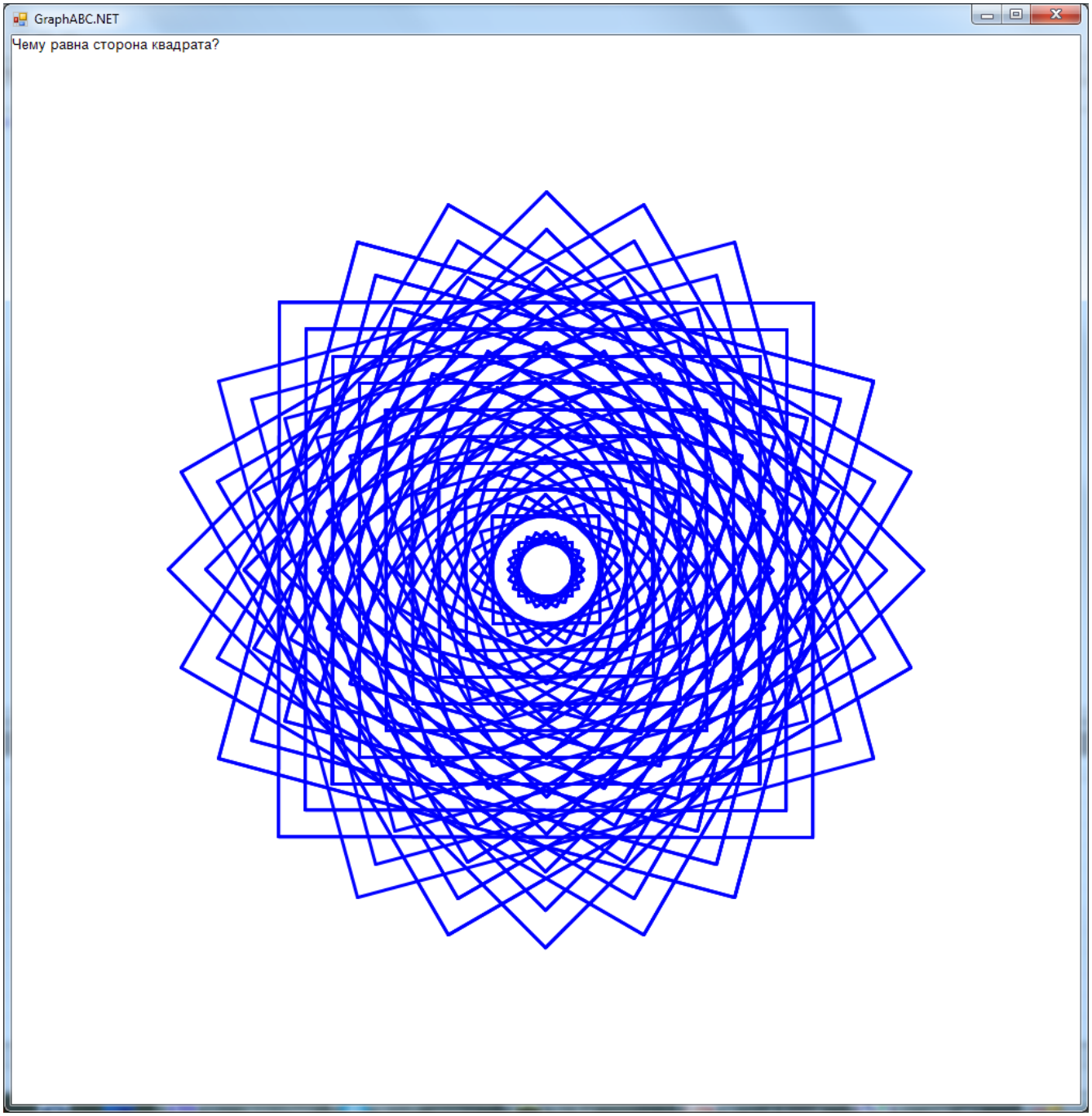
# Пятиконечная звезда (пентаграмма)

```
program Star;

uses
   Turtle;

var
   i: integer;

begin
   Down;
   Right(18);
   for i := 1 to 5 do begin
      Go(200);
      Right(144);
   end;
end.
```

# Пятиконечная звезда (пентаграмма)

```
program Star;

uses
    Turtle;

var
    i: integer;

begin
    Down;
    Right(18);
    for i := 1 to 5
        Go(200);
        Right(144);
    end;
end.
```

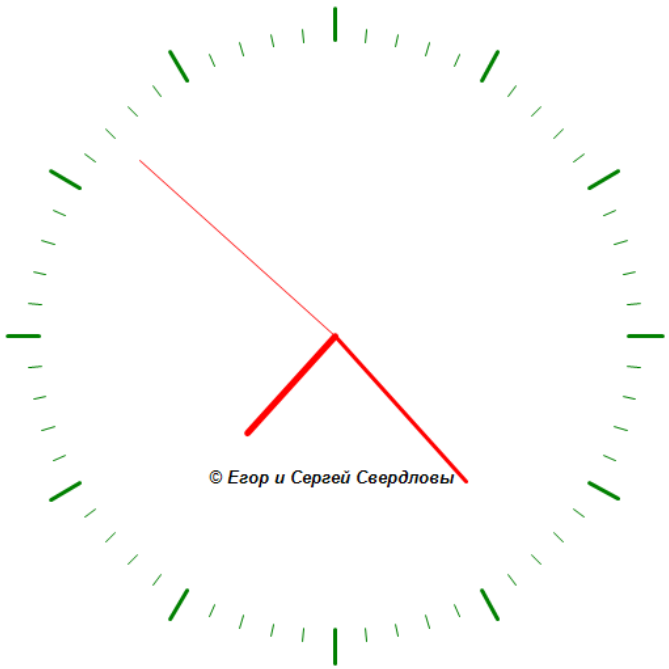



GraphABC.NET
Чему равна сторона квадрата?

GraphABC.NET

03.04.2017 19:23:52

3 апреля Понедельник




23 минуты 8-го

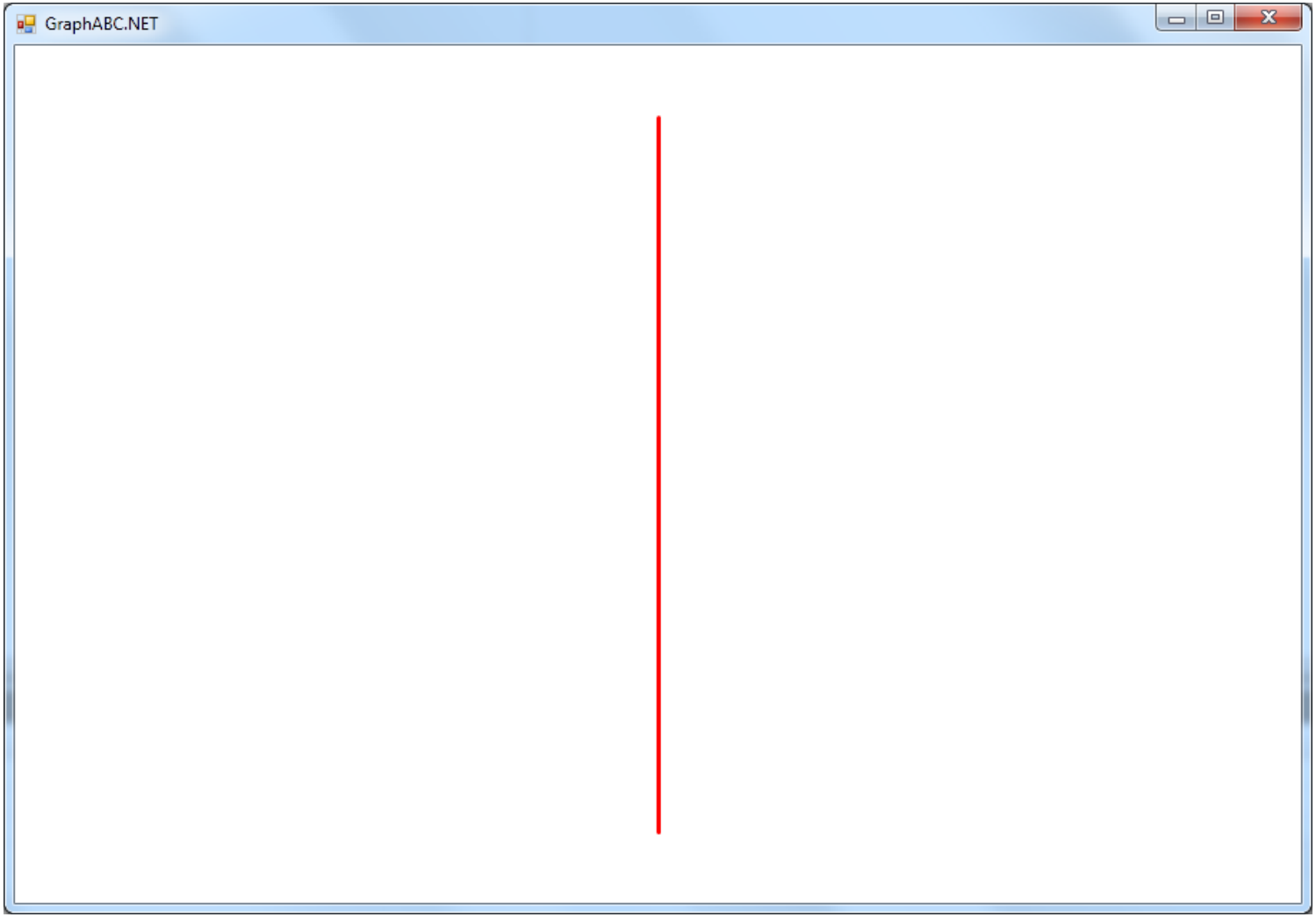
# Простой фрактал
## *Кривая Коха*

# Простой фрактал
# *Кривая Коха*

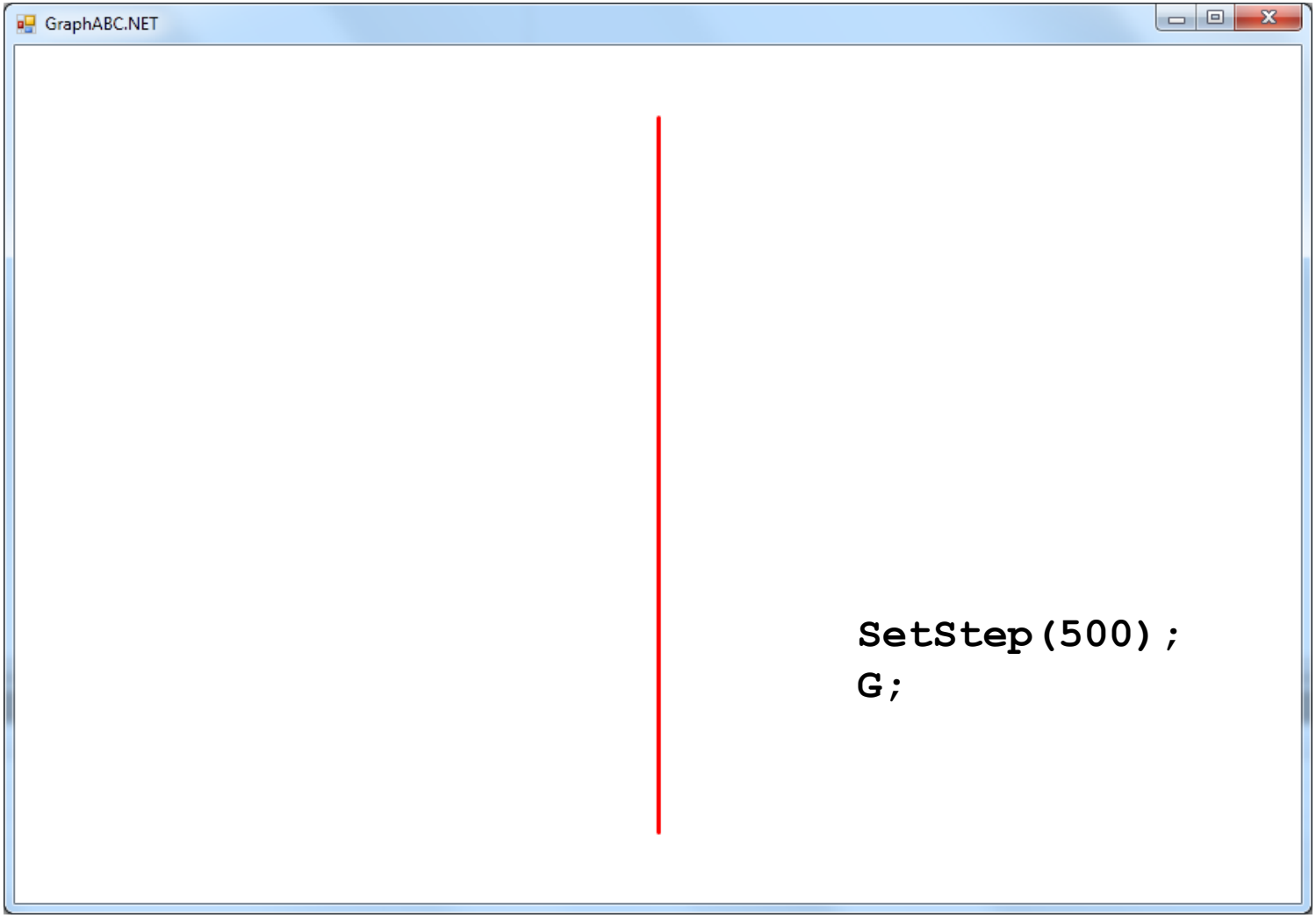
## Упрощенная черепашья графика (интерпретация *L*-системы)

- Фиксированный шаг: G;
- Повороты на фиксированный угол:
  - L – налево;
  - R – направо;

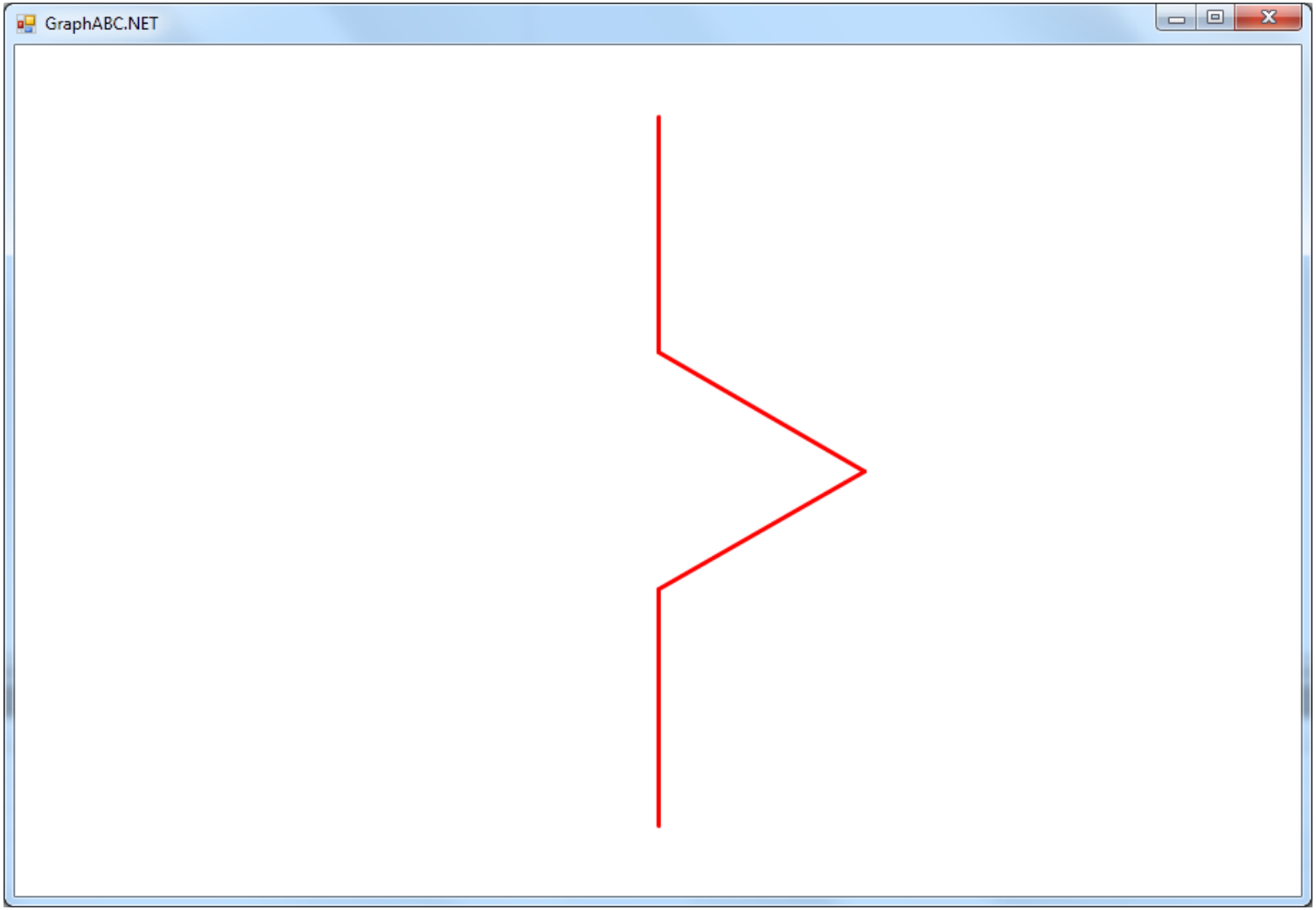
# Кривая Коха нулевого порядка



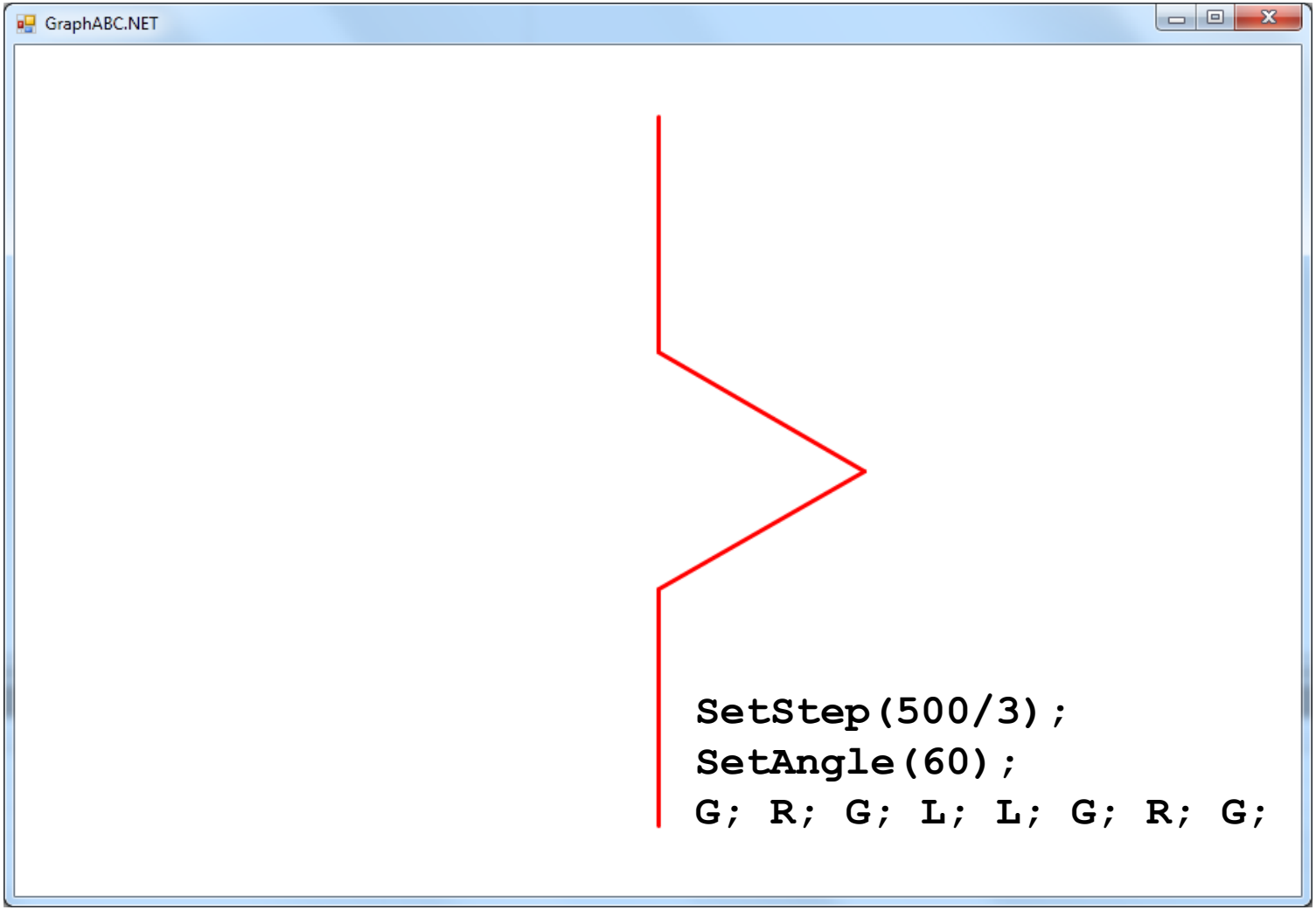
# *Кривая Коха нулевого порядка*



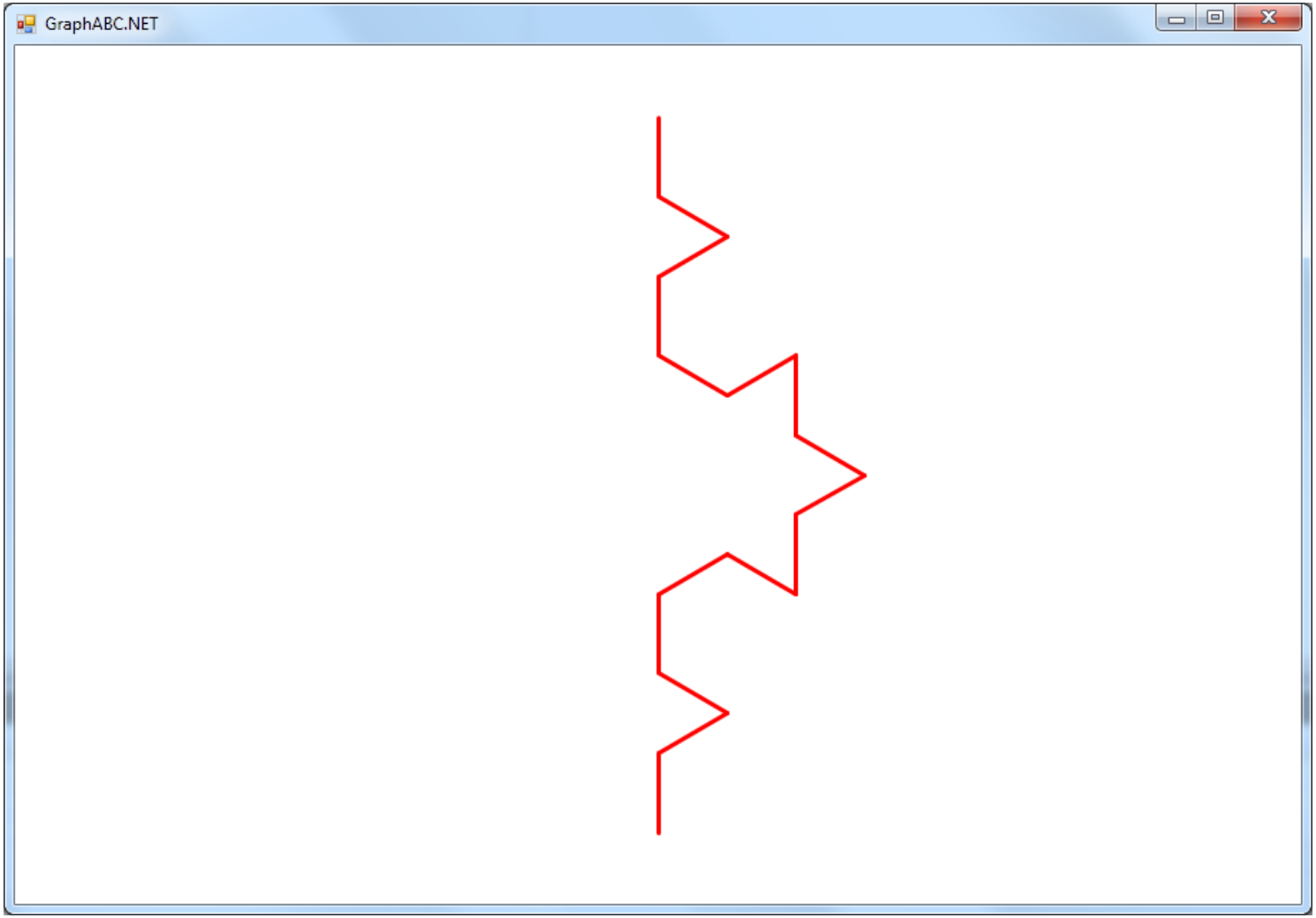
# Кривая Коха 1-го порядка



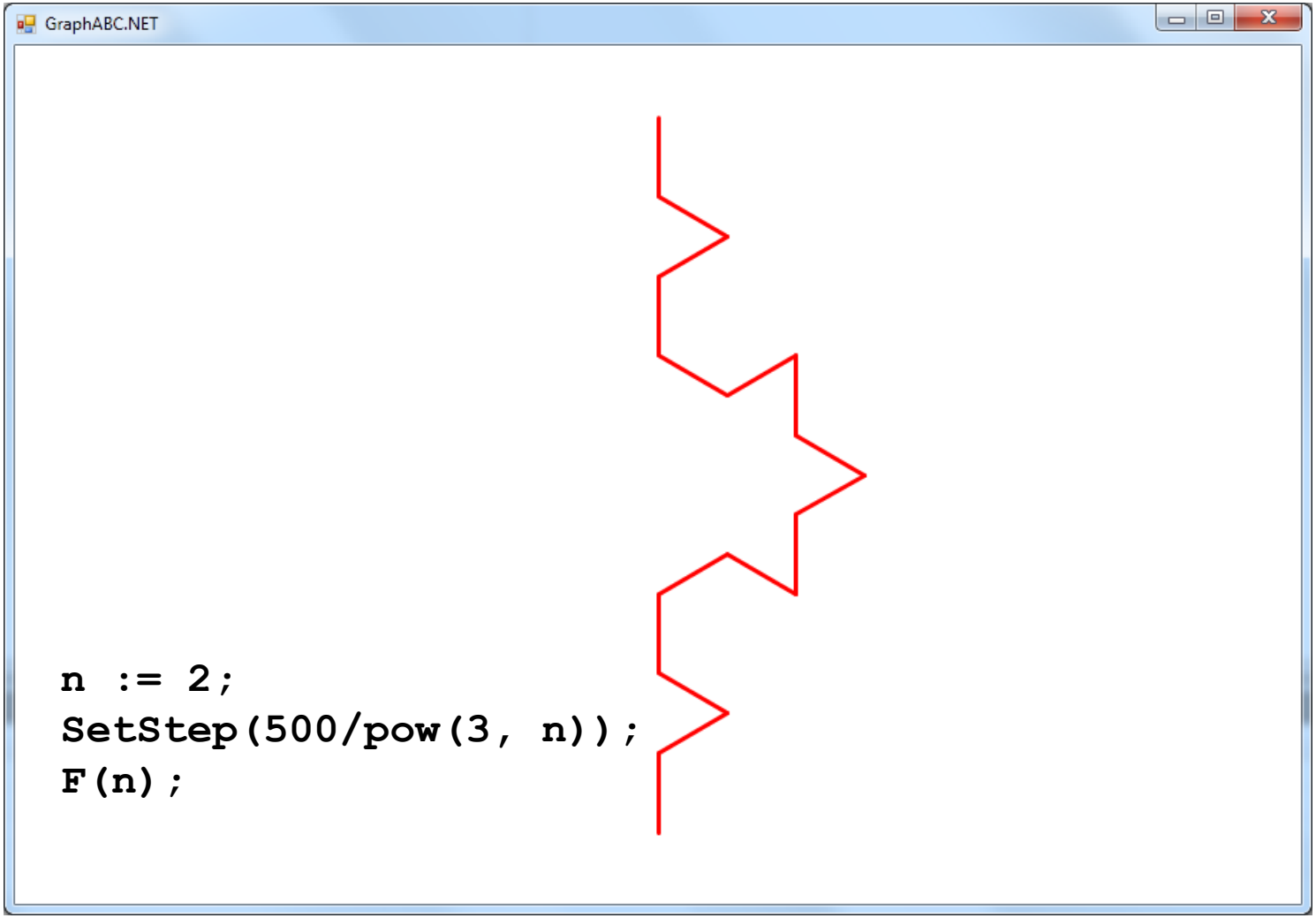
# Кривая Коха 1-го порядка



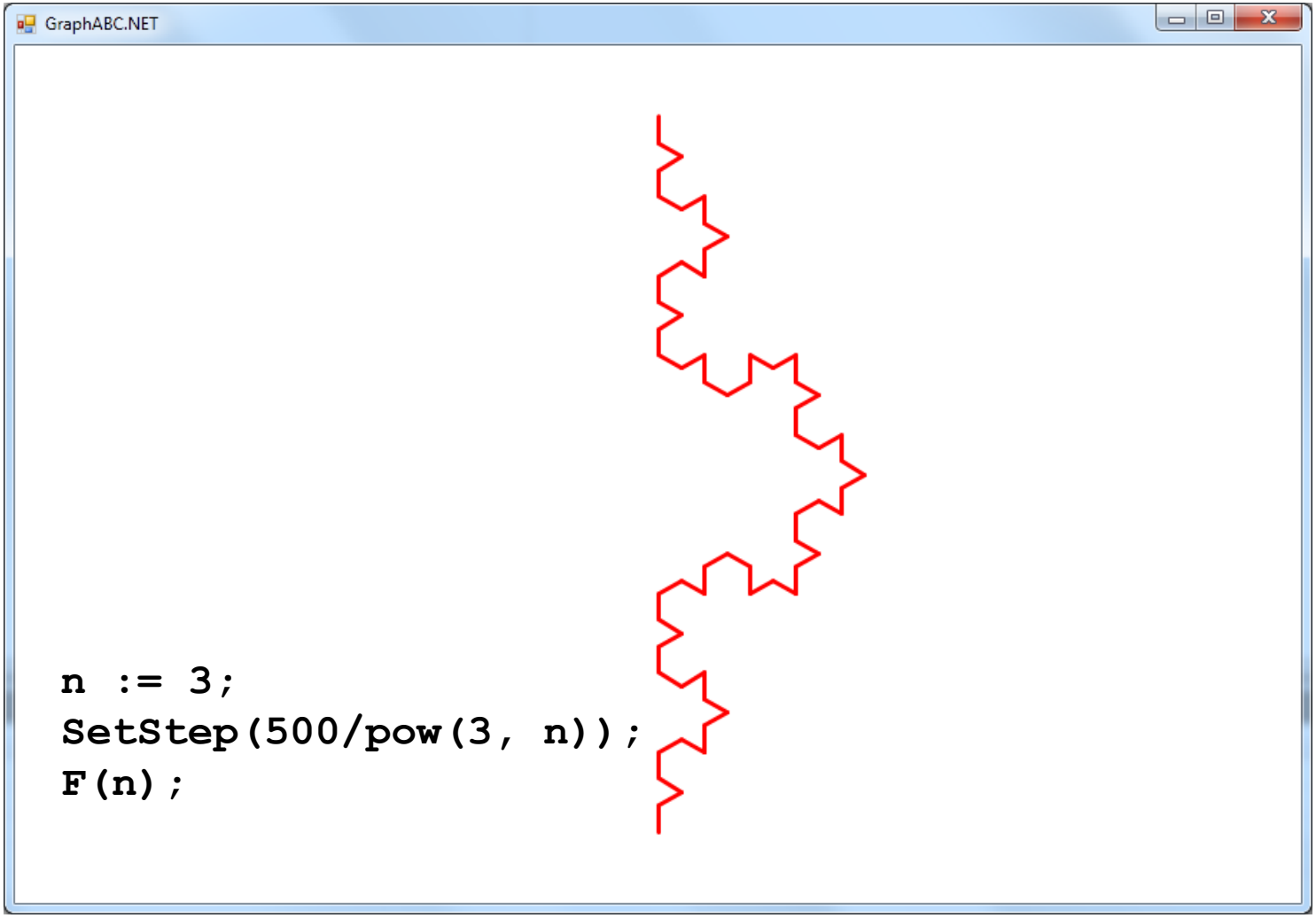
# *Кривая Коха 2-го порядка*



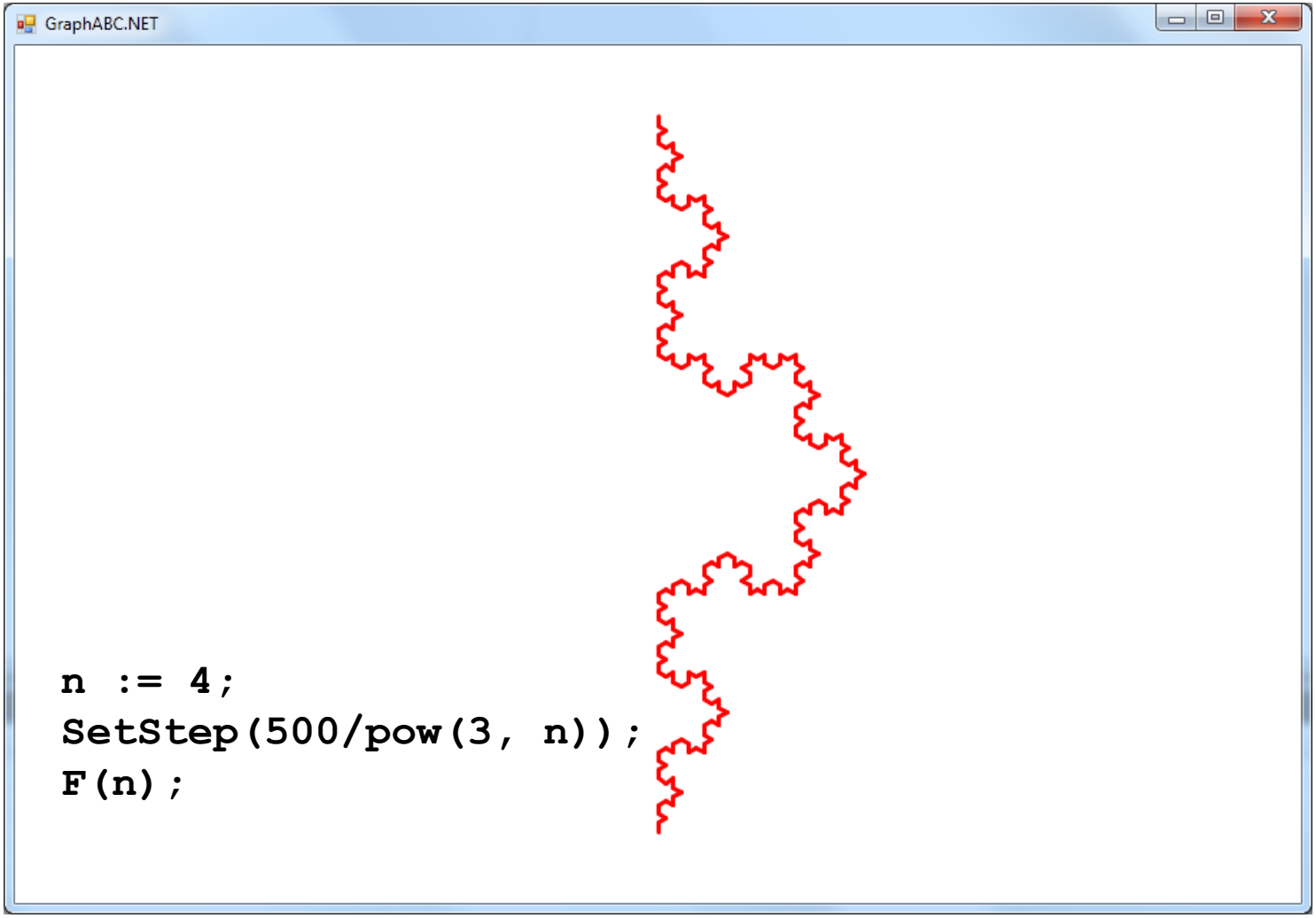
# Кривая Коха 2-го порядка



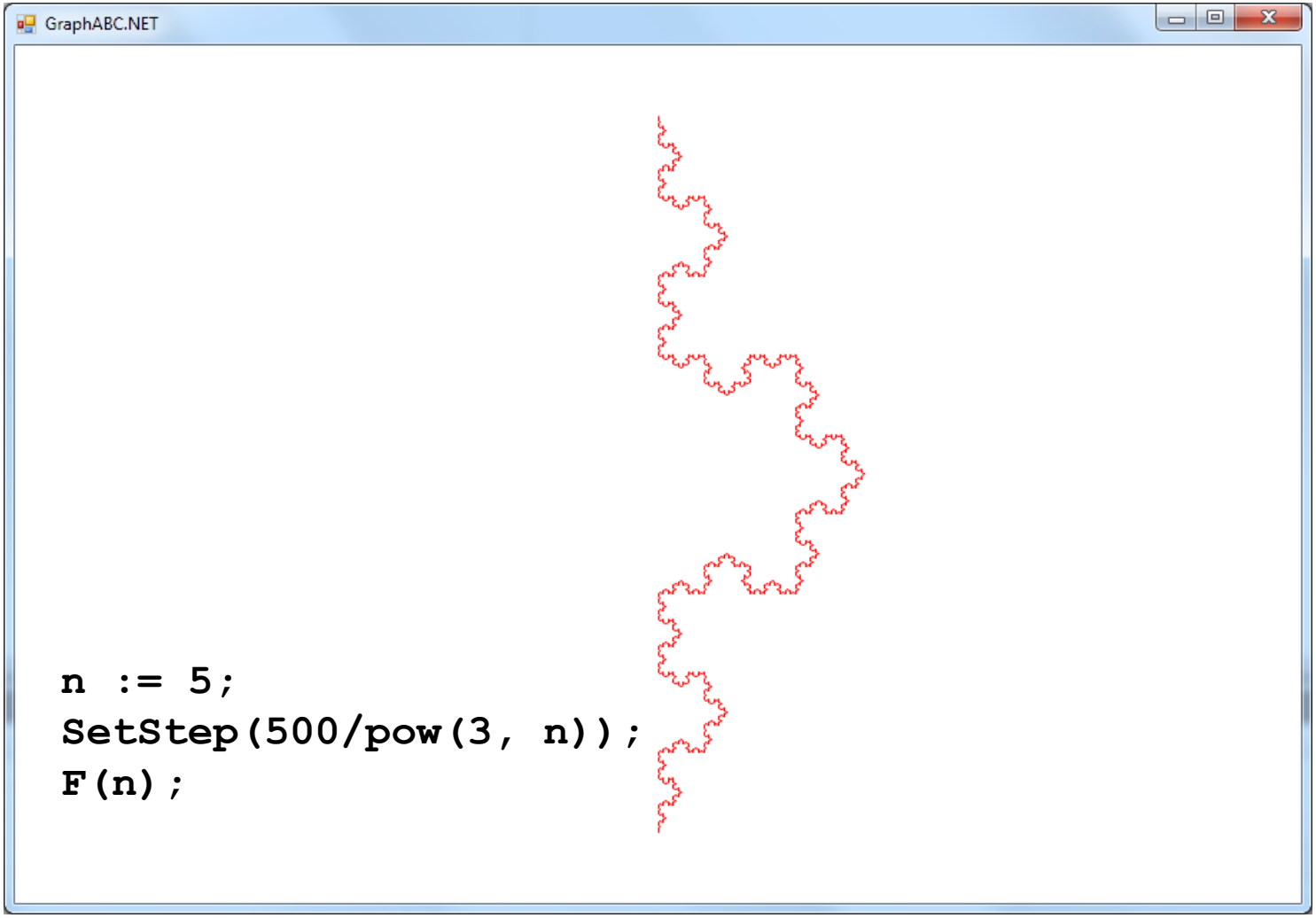
# Кривая Коха 3-го порядка



# Кривая Коха 4-го порядка



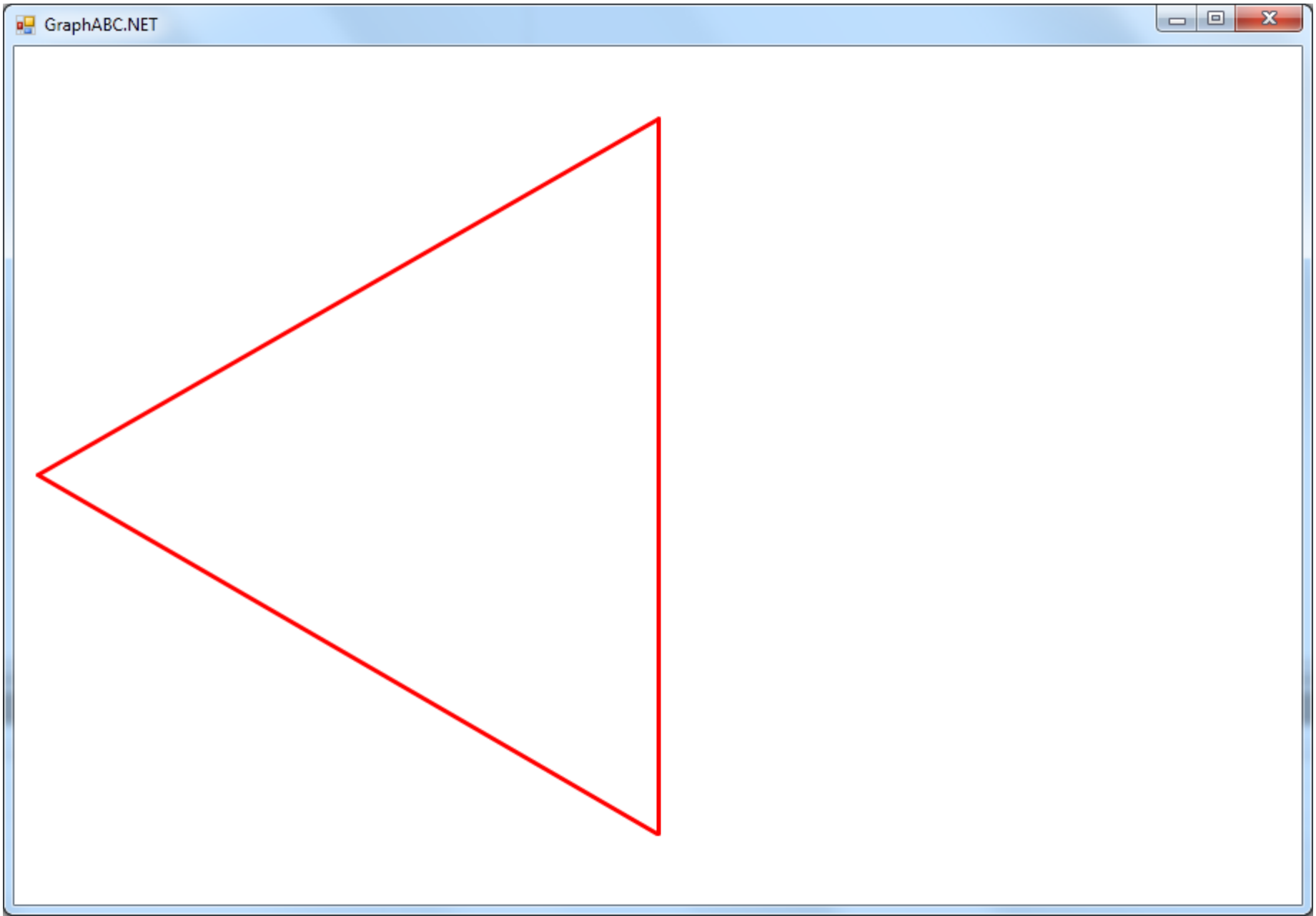
# *Кривая Коха 5-го порядка*



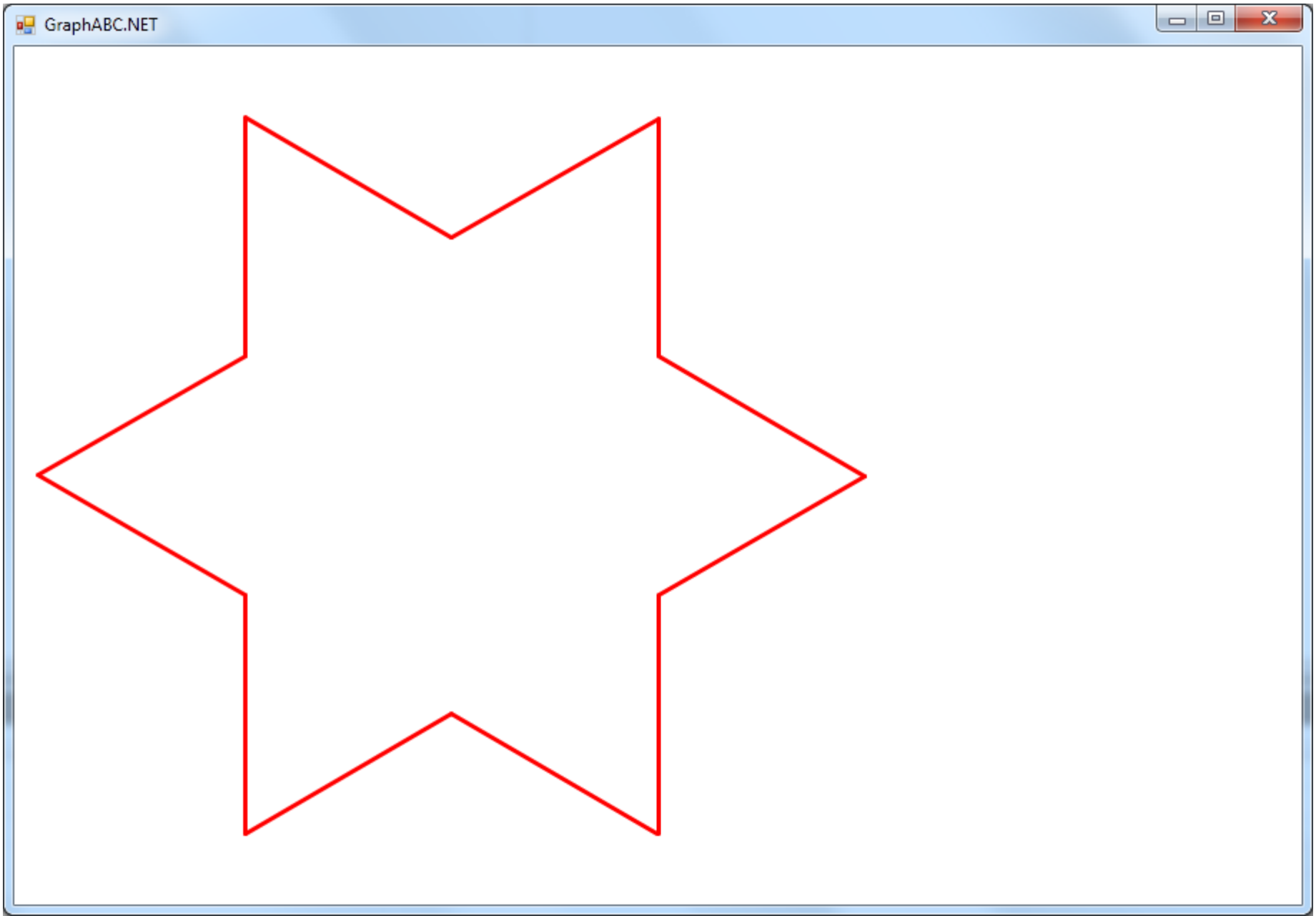
# *Кривая Коха k-го порядка*

```
procedure F(k: integer);
begin
   if k =0 then
      G
   else begin
      F(k-1); R; F(k-1); L; L; F(k-1); R; F(k-1);
   end;
end;
```

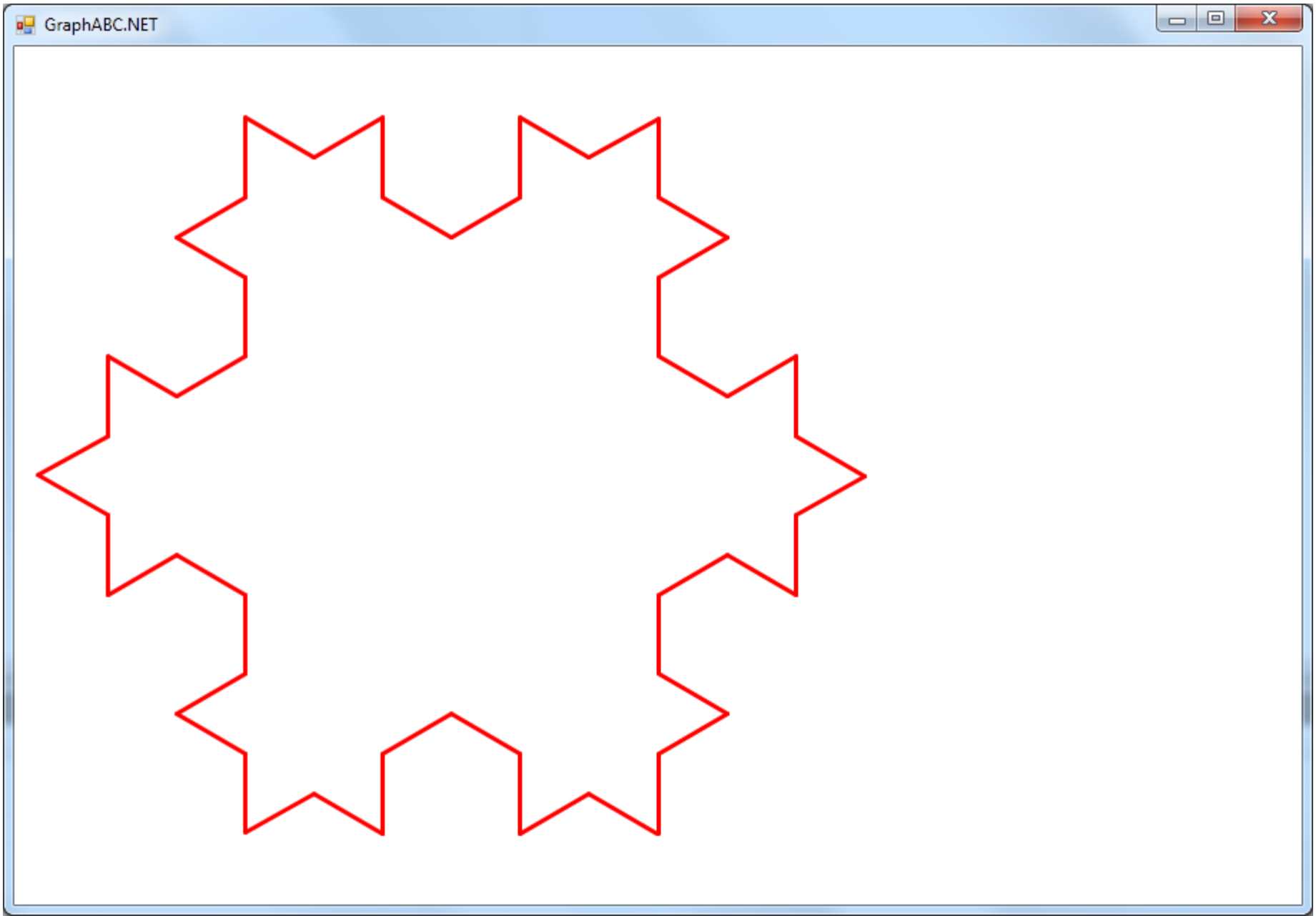
# Снежинка Коха нулевого порядка



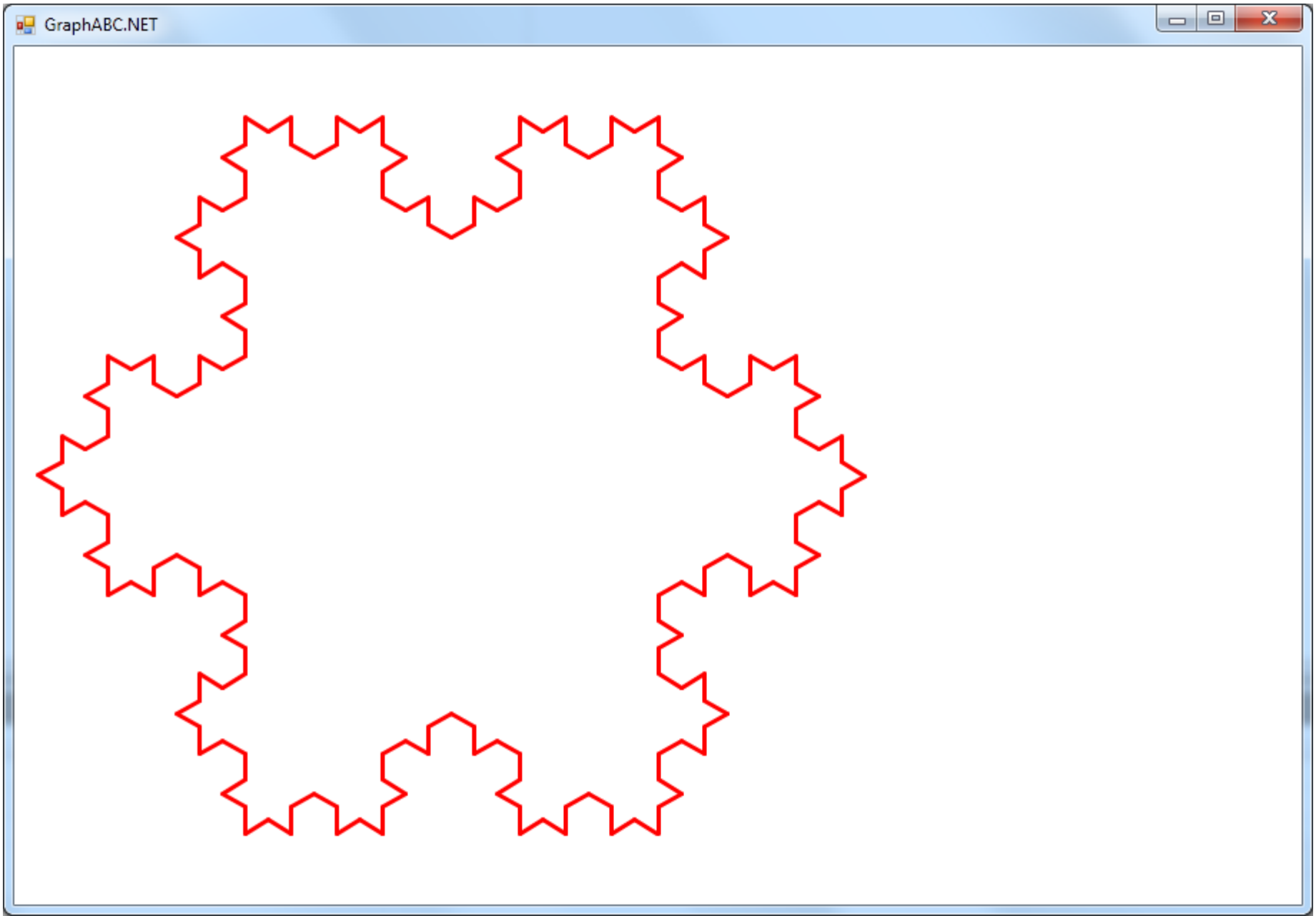
# Снежинка Коха 1-го порядка



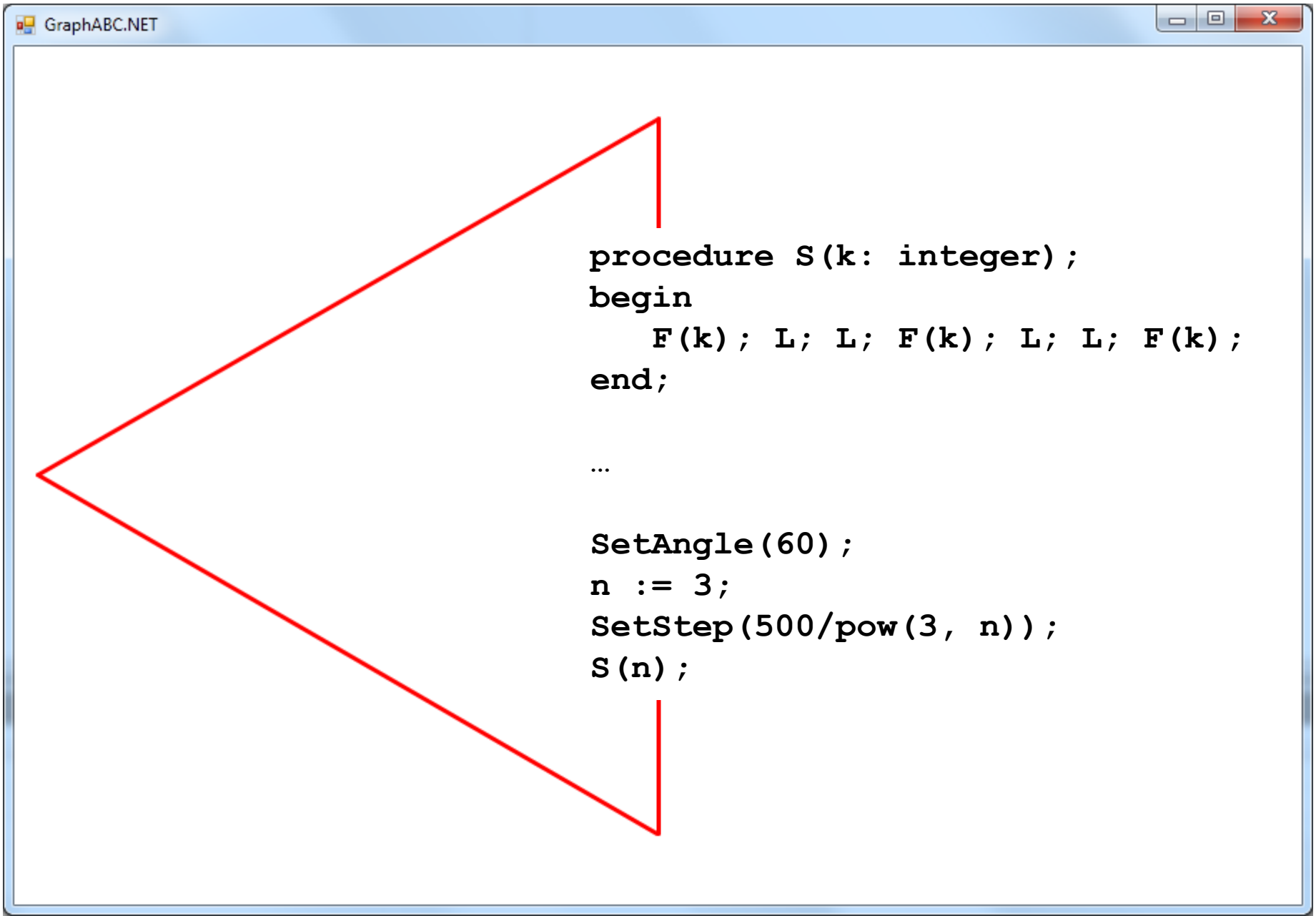
# Снежинка Коха 2-го порядка



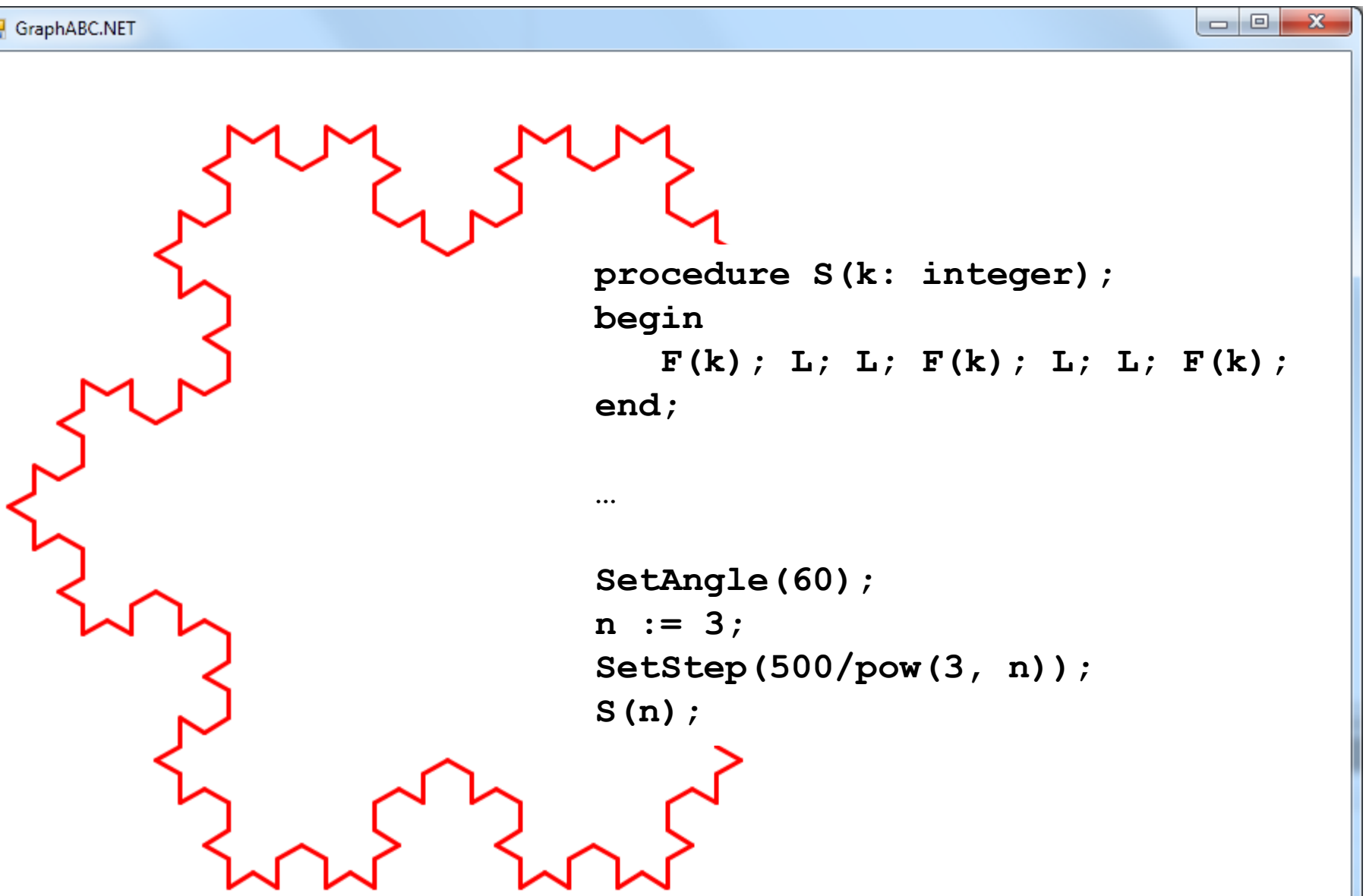
# Снежинка Коха 3-го порядка



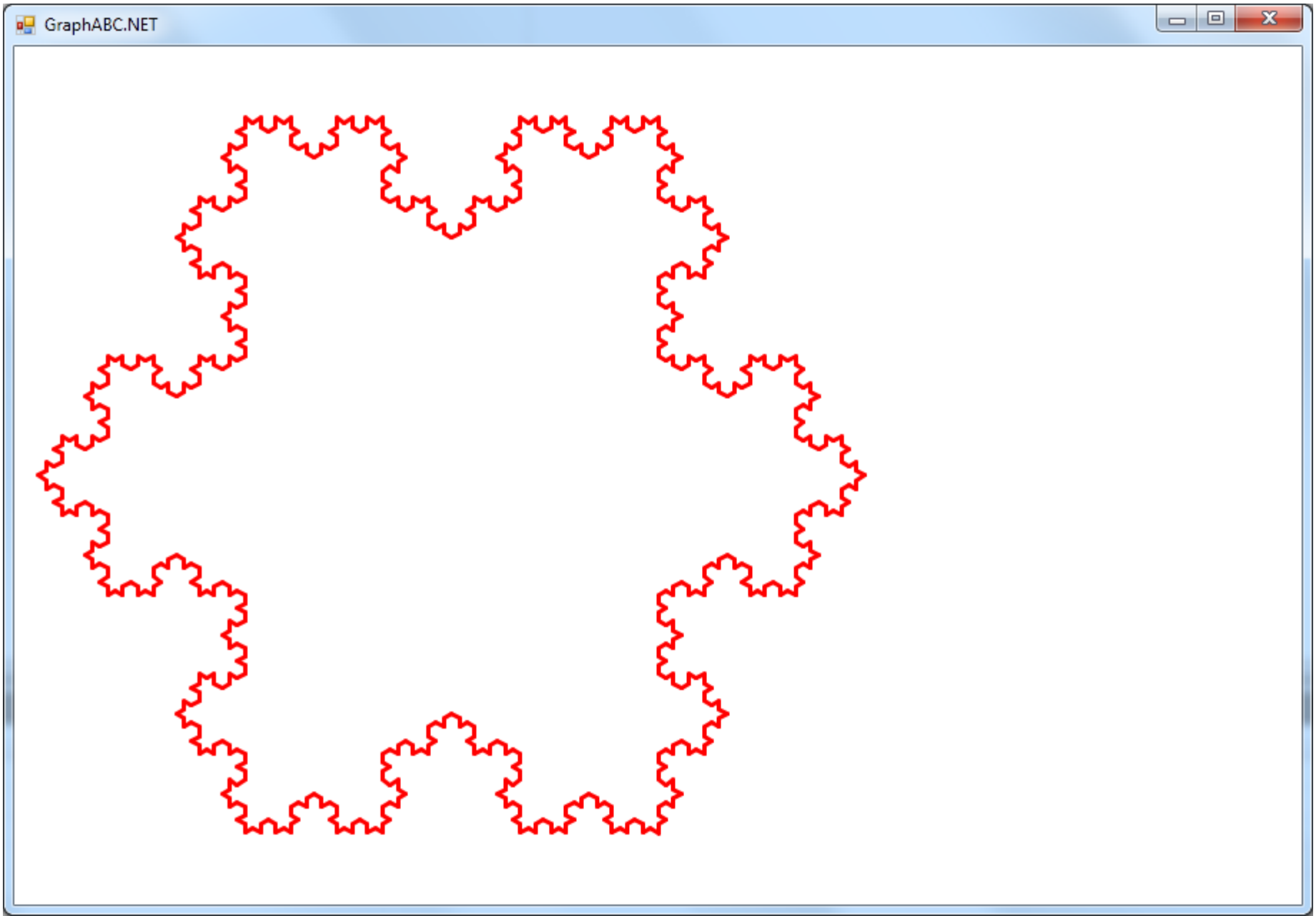
# Снежинка Коха k-го порядка



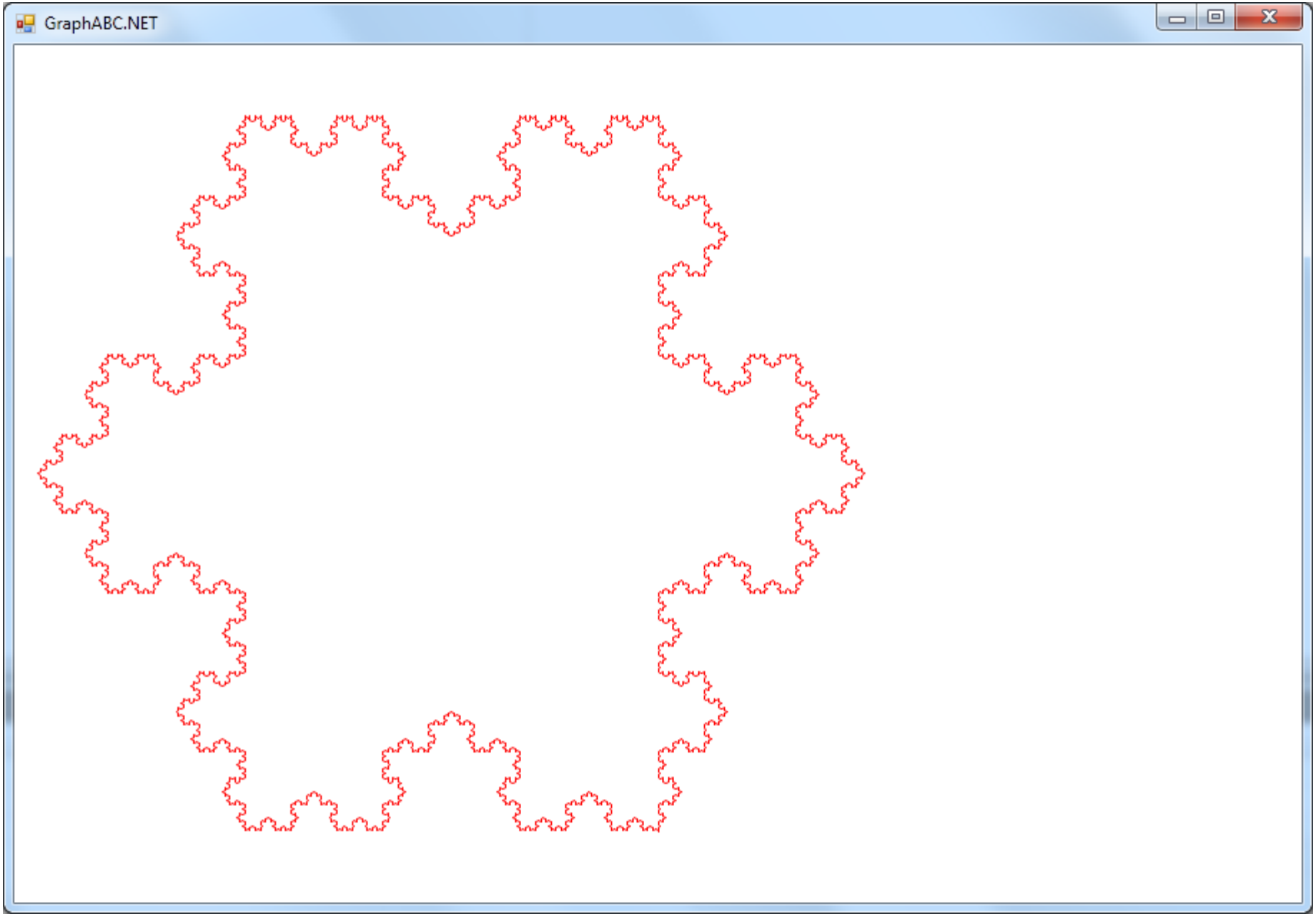
# Снежинка Коха 3-го порядка



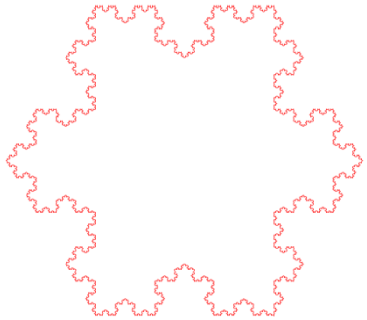
# Снежинка Коха 4-го порядка



# Снежинка Коха 5-го порядка

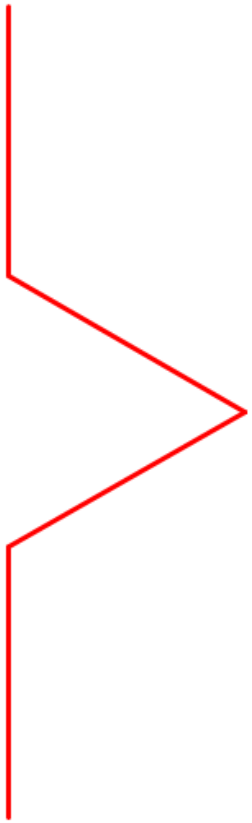


# Периметр и площадь снежинки Коха

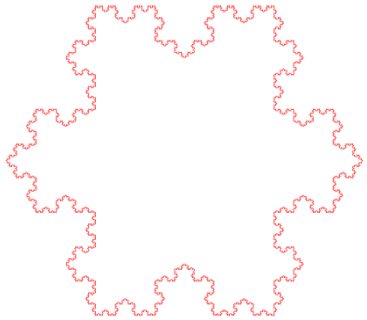


Количество звеньев: $N = 3 \cdot 4^n$

Длина звена: $L_n = \frac{L_0}{3^n}$



# Периметр и площадь снежинки Коха

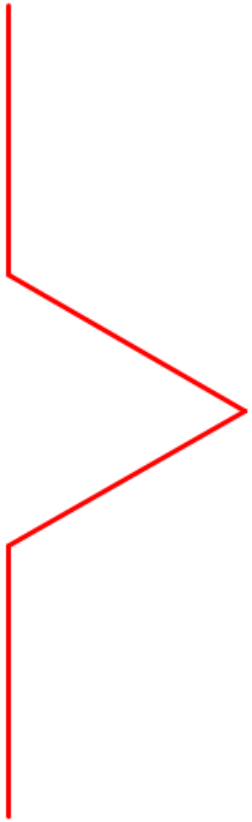


Количество звеньев: $N = 3 \cdot 4^n$

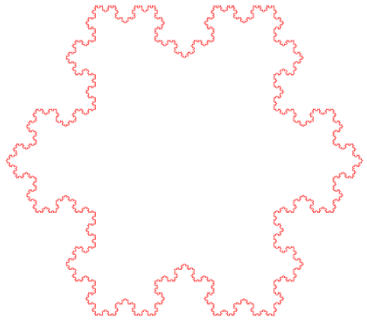
Длина звена: $L_n = \frac{L_0}{3^n}$

Периметр снежинки:

$$P_n = N \cdot L_n = 3 \cdot 4^n \cdot \frac{L_0}{3^n} = 3L_0 \left( \frac{4}{3} \right)^n$$



# Периметр и площадь снежинки Коха



Количество звеньев: $N = 3 \cdot 4^n$

Длина звена: $L_n = \frac{L_0}{3^n}$

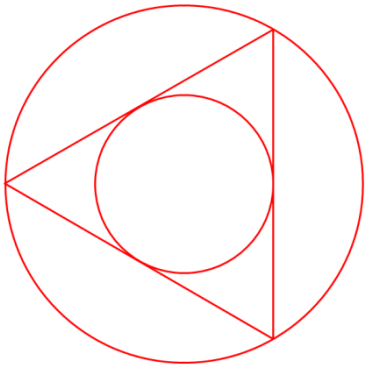
Периметр снежинки:

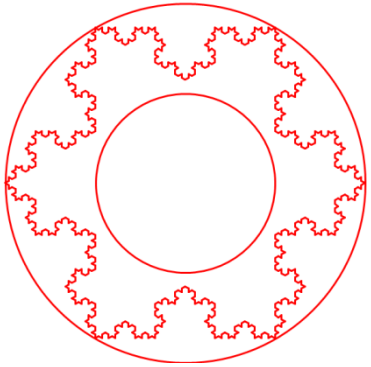
$$P_n = N \cdot L_n = 3 \cdot 4^n \cdot \frac{L_0}{3^n} = 3L_0 \left( \frac{4}{3} \right)^n$$

Площадь снежинки: $S_{вп} < S_n < S_{оп}$

$$r = \frac{\sqrt{3}}{6} L_0$$

$$R = 2r$$





# Размерность кривой Коха

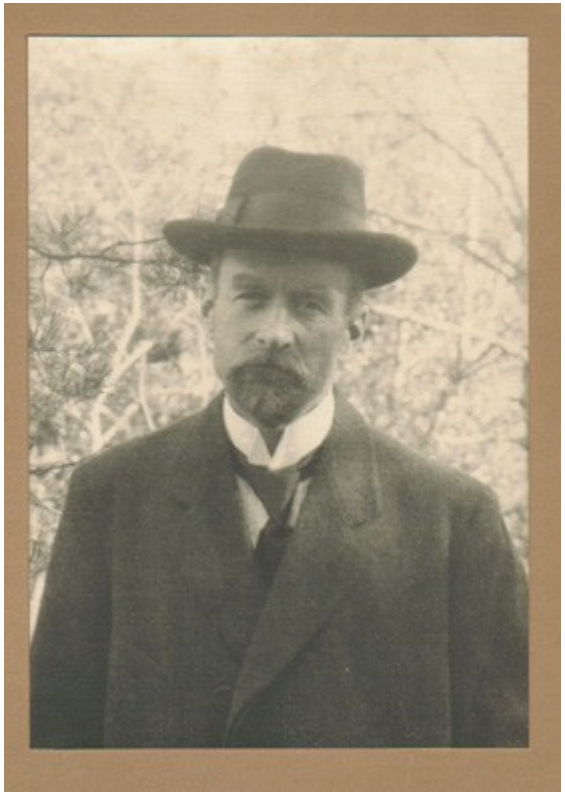
Кривая Коха имеет промежуточную (то есть не целую) хаусдорфову размерность

$$D_H = \lim_{L_n \to 0} \frac{\ln N}{\ln(1/L_n)} = \lim_{n \to \infty} \frac{\ln 4^n}{\ln(3^n / L_0)} = \lim_{n \to \infty} \frac{\ln 4^n}{\ln 3^n - \ln L_0} =$$

$$\lim_{n \to \infty} \frac{\ln 4^n}{\ln 3^n} = \lim_{n \to \infty} \frac{n \ln 4}{n \ln 3} = \frac{\ln 4}{\ln 3} = \log_3 4 \approx 1,26$$

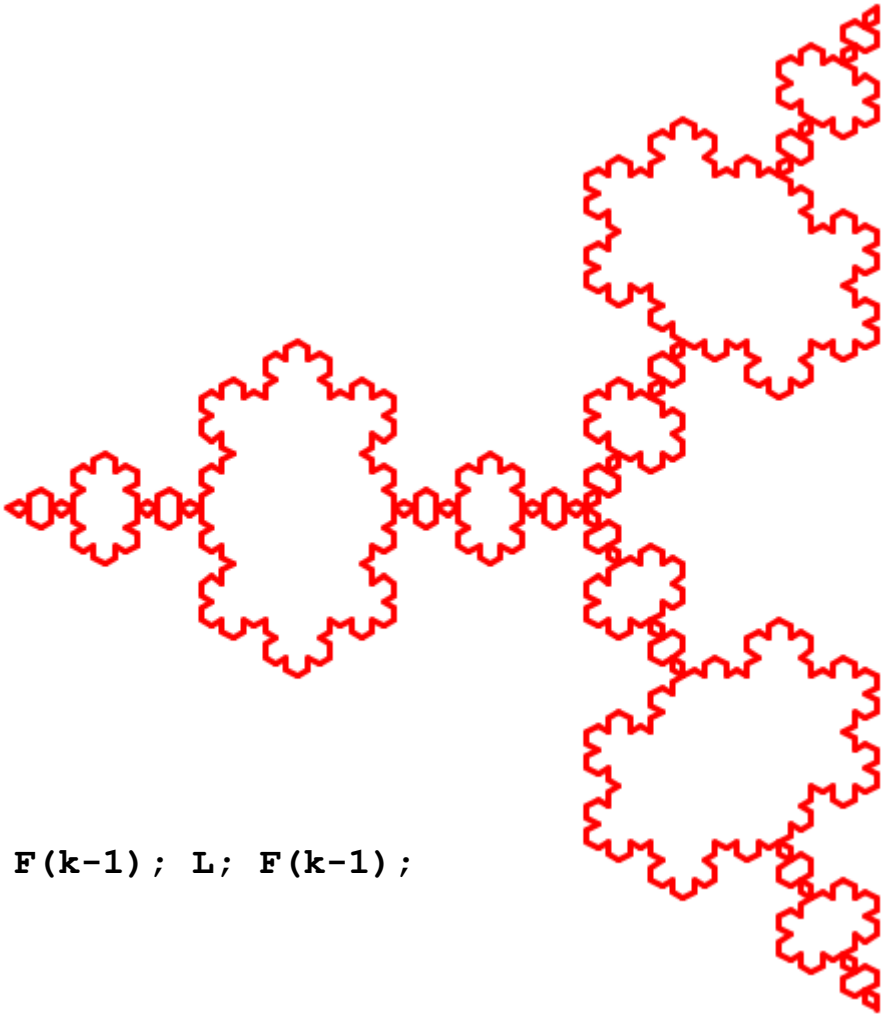
Для плавной кривой

$$D_H = \lim_{L \to 0} \frac{\ln N}{\ln(1/L)} = 1$$



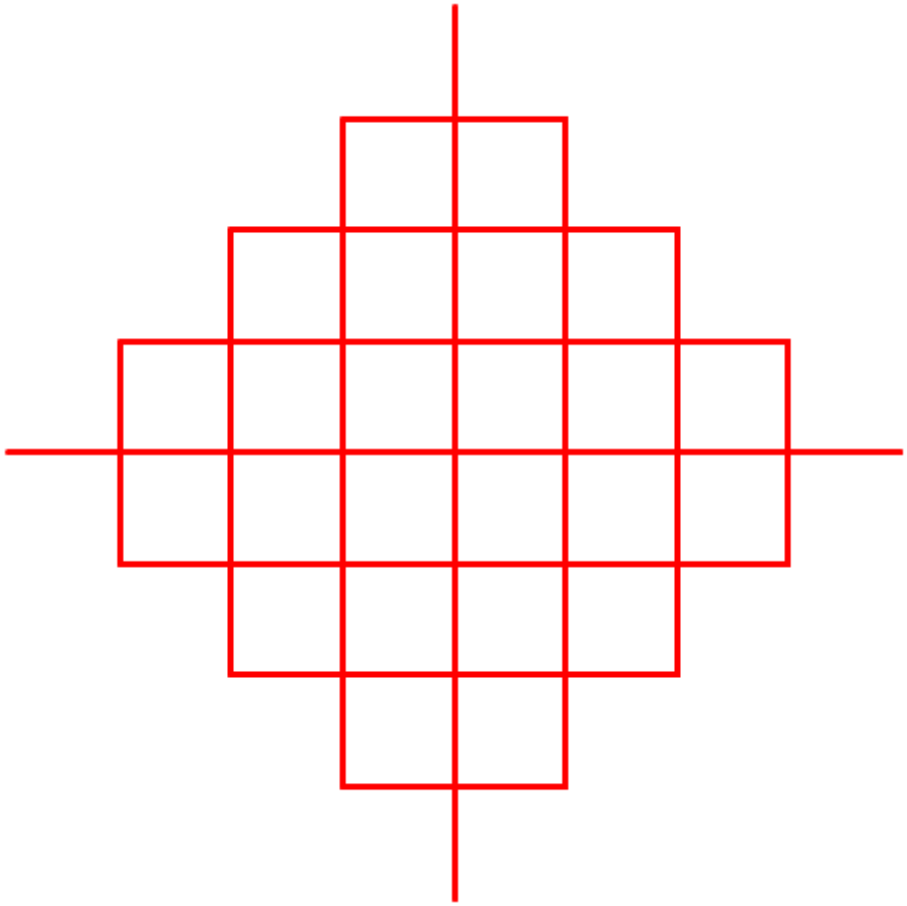
**Нильс Фабиан Хельге фон Кох** (швед. *Niels Fabian Helge von Koch*, 25 января 1870 — 11 марта 1924) — шведский математик, брат композитора Сигурда фон Коха.

# *Варианты снежинка Коха (инвертированы повороты)*



```
procedure F(k: integer);
begin
   if k =0 then
      G
   else begin
      F(k-1); L; F(k-1); R; R; F(k-1); L; F(k-1);
   end;
end;
```

# *Варианты снежинка Коха (угол 90º, n=3)*



# *L*-системы
# (Lindenmayer Systems)

# *L*-системы
# (Lindenmayer Systems)


**Аристид Линденмайер** (17 ноября 1925 — 30 октября 1989) — венгерский биолог. В 1968 году он создал формальный язык, который сейчас называют L-systems. Используя эти системы Линденмайер смоделировал поведение клеток растений.

# *L*-системы
# (Lindenmayer Systems)

## DOL-система
## (детерминированная контекстно-свободная *L*-система)

$$G = (V, \omega, P),$$

где

*V* (*алфавит*) — множество символов, содержащих как элементы, которые могут быть заменены (*переменные*), так и элементы, которые не могут быть заменены ("константы" или "терминальные символы"). $V = T \cup N$.

ω (*старт*, *аксиома* или *инициатор*) — строка (цепочка) символов из V, определяющая начальное состояние системы.

*P* — множество *порождающих правил*, определяющих, каким образом переменные могут быть заменены комбинациями констант и других переменных. Левые части правил уникальны.

$$N \rightarrow \alpha$$

Правила применяются не последовательно, а одновременно.

# *L*-системы

**Пример 1**

**Переменные**: A B
**Константы**: нет
**Аксиома**: A
**Правила**:

$A \rightarrow AB$
$B \rightarrow A$

Вывод (развитие):

$\Rightarrow$

A =>
AB =>
ABA =>
ABAAB =>
ABAABABA =>
ABAABABAABAAB =>
ABAABABAABAABABAABABAABAABABAABAAB

- Длины строк образуют последовательность чисел Фибоначчи.

- Последовательностями Фибоначчи будут также количества символов A и B в этих строках.

- Каждая строка является «суммой» (конкатенацией) двух предыдущих.

# *L*-системы

**Пример 2**

Аксиома: F
Правило: F → F-F++F-F

Вывод (развитие):

F =>
F-F++F-F =>
F-F++F-F - F-F++F-F ++ F-F++F-F - F-F++F-F =>
F-F++F-F - F-F++F-F ++ F-F++F-F - F-F++F-F - F-F++F-F - F-F++F-F ++ F-
F++F-F - F-F++F-F ++ F-F++F-F - F-F++F-F ++ F-F++F-F - F-
F++F-F - F-F++F-F - F-F++F-F ++ F-F++F-F - F-F++F-F

- Количество символов F учетверяется.
- Количество символов + равно количеству символов –
- Количество символов + и - увеличивается в 5 раз.

# *Интерпретация L*-системы

**Пример 2**

Аксиома: F
Правило: F → F-F++F-F

# *Интерпретация L*-системы

**Пример 2**

Аксиома: F
Правило: F → F-F++F-F

**Черепашья графика:**

F – вперед;
- – направо
+ – налево
Угол поворота 60º

# *Интерпретация L*-системы

**Пример 2**

Аксиома: F
Правило: F → F-F++F-F

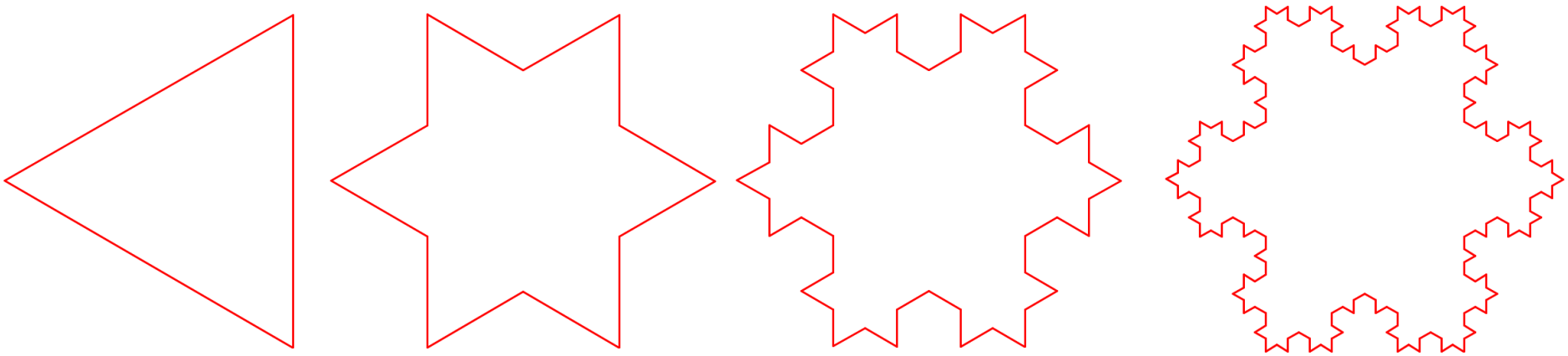
**Черепашья графика:**

F – вперед;
- – направо
+ – налево
Угол поворота 60º

**Кривая Коха**

# *Снежинка Коха*

Аксиома: F++F++F
Правило: F → F-F++F-F
Угол: 60º



# *Кривая дракона*

Аксиома: FX
Правила:
X → X+YF+
Y → -FX-Y
Угол: 90º

F – вперед;
\- – направо
\+ – налево
X, Y – ничего

# *Кривая дракона*

Аксиома: FX
Правила:
X → X+YF+
Y → -FX-Y
Угол: 90º

F – вперед;
\- – направо
\+ – налево
X, Y – ничего

# *Кривая дракона*

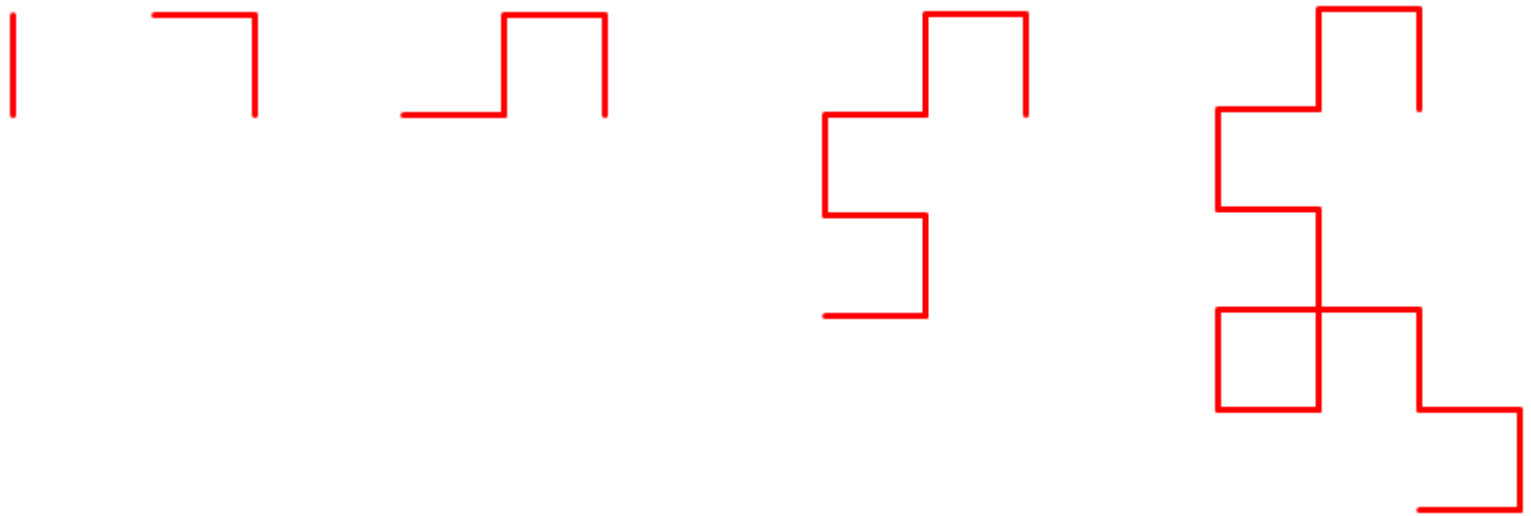
Аксиома: FX
Правила:
X → X+YF+
Y → -FX-Y
Угол: 90º

F – вперед;
\- – направо
\+ – налево
X, Y – ничего

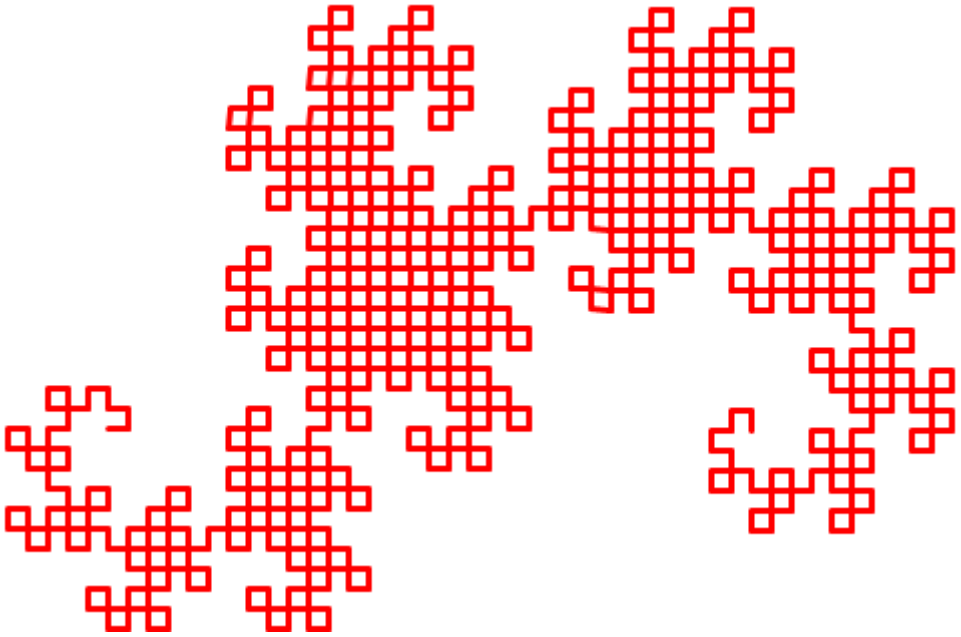
# *Кривая дракона*

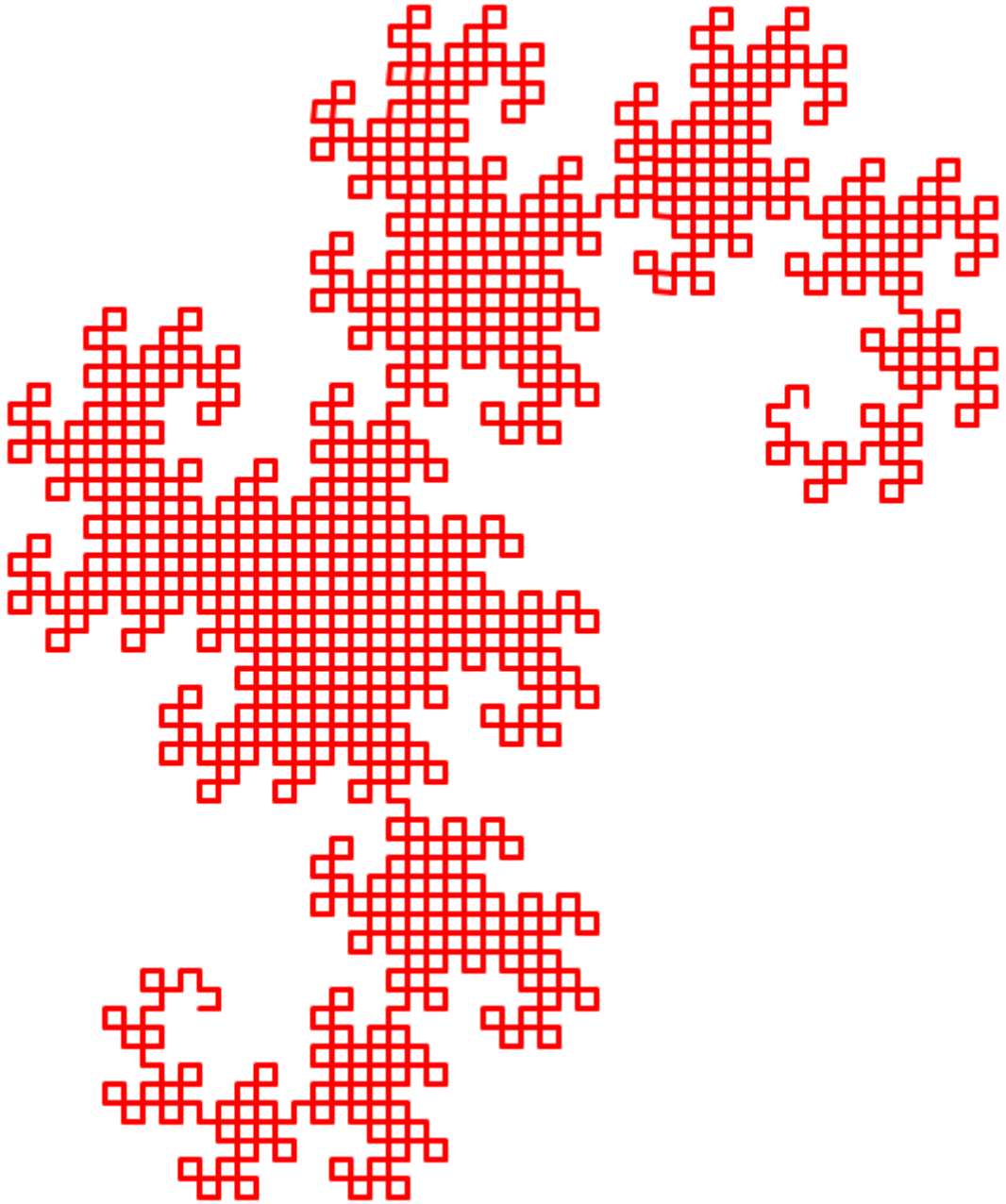
Аксиома: FX
Правила:
X → X+YF+
Y → -FX-Y
Угол: 90º

F – вперед;
\- – направо
\+ – налево
X, Y – ничего



# *Кривая дракона 10 порядка*





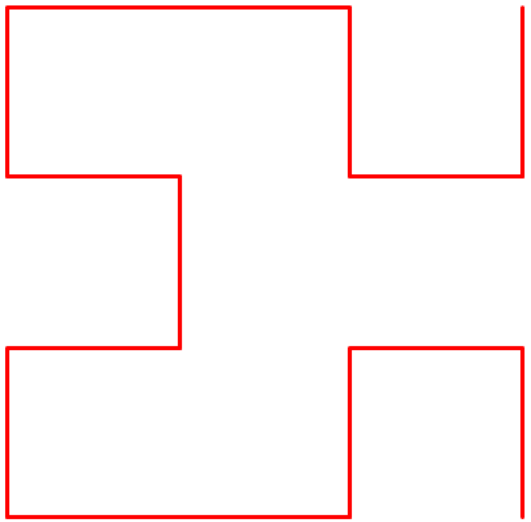
*Кривая дракона*
*11 порядка*

# *Кривая Гильберта*

**Кривая Гильберта**

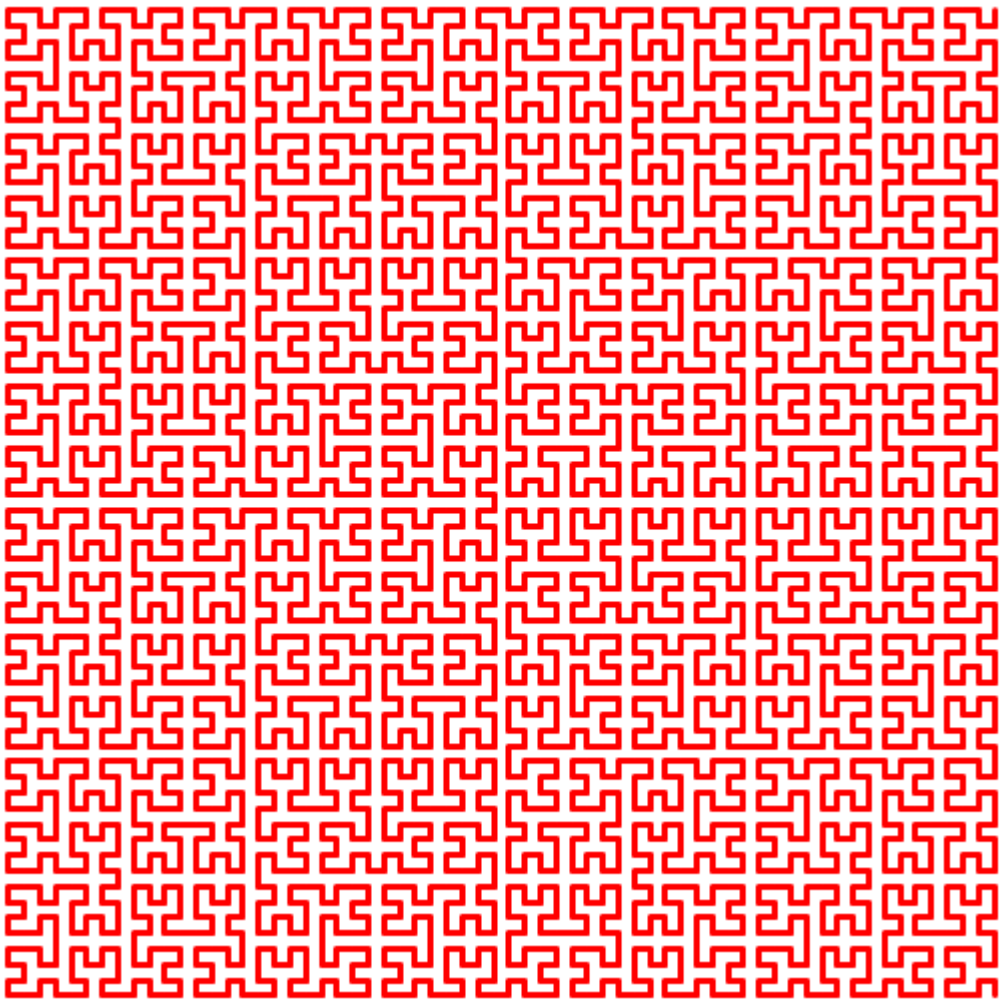
Аксиома: X
Правила:
X → -YF+XFX+FY-
Y → +XF-YFY-FX+
Угол: 90º

1-го порядка



2-го порядка

# *Кривая Гильберта 6-го порядка*



# Рекурсивная реализация

**Кривая Гильберта**

Аксиома: X
Правила:
X → -YF+XFX+FY-
Y → +XF-YFY-FX+
Угол: 90°

```
procedure Y(k: integer);
{ Y → +XF−YFY−FX+ }
begin
   if k>0 then begin
      R; X(k-1); G; L; Y(k-1); G; Y(k-1); L; G; X(k-1); R;
   end;
end;

procedure X(k: integer);
{ X → −YF+XFX+FY− }
begin
   if k>0 then begin
      L; Y(k-1); G; R; X(k-1); G; X(k-1); R; G; Y(k-1); L;
   end;
end;

procedure S(k: integer);
begin
   X(k);
end;
```

# Прямая реализация. Типы

```
program Fractals;

uses
   LSystem;

const
   nmax = 10;

type

   tRule = record
      left: char;
      right: string;
   end;

   tLSystem = record
      axiom : string;
      n     : integer;
      rules : array [1..nmax] of tRule;
      angle : real;
   end;
```

# Прямая реализация. Порождение

```
function Generate(LS: tLSystem; generation: integer): string;
var
   s1, s2: string;
   i, j, k: integer;
   replace: Boolean;
begin
   s1 := LS.axiom;
   for i := 1 to generation do begin
      s2 := '';
      for j := 1 to length(s1) do begin
         replace := false;
         for k:= 1 to LS.n do begin
            if s1[j] = LS.rules[k].left then begin
               s2 := s2 + LS.rules[k].right;
               replace := true;
            end;
         end;
         if not replace then
            s2 := s2 + s1[j]
      end;
      s1 := s2;
   end;
   Generate := s1;
end;
```

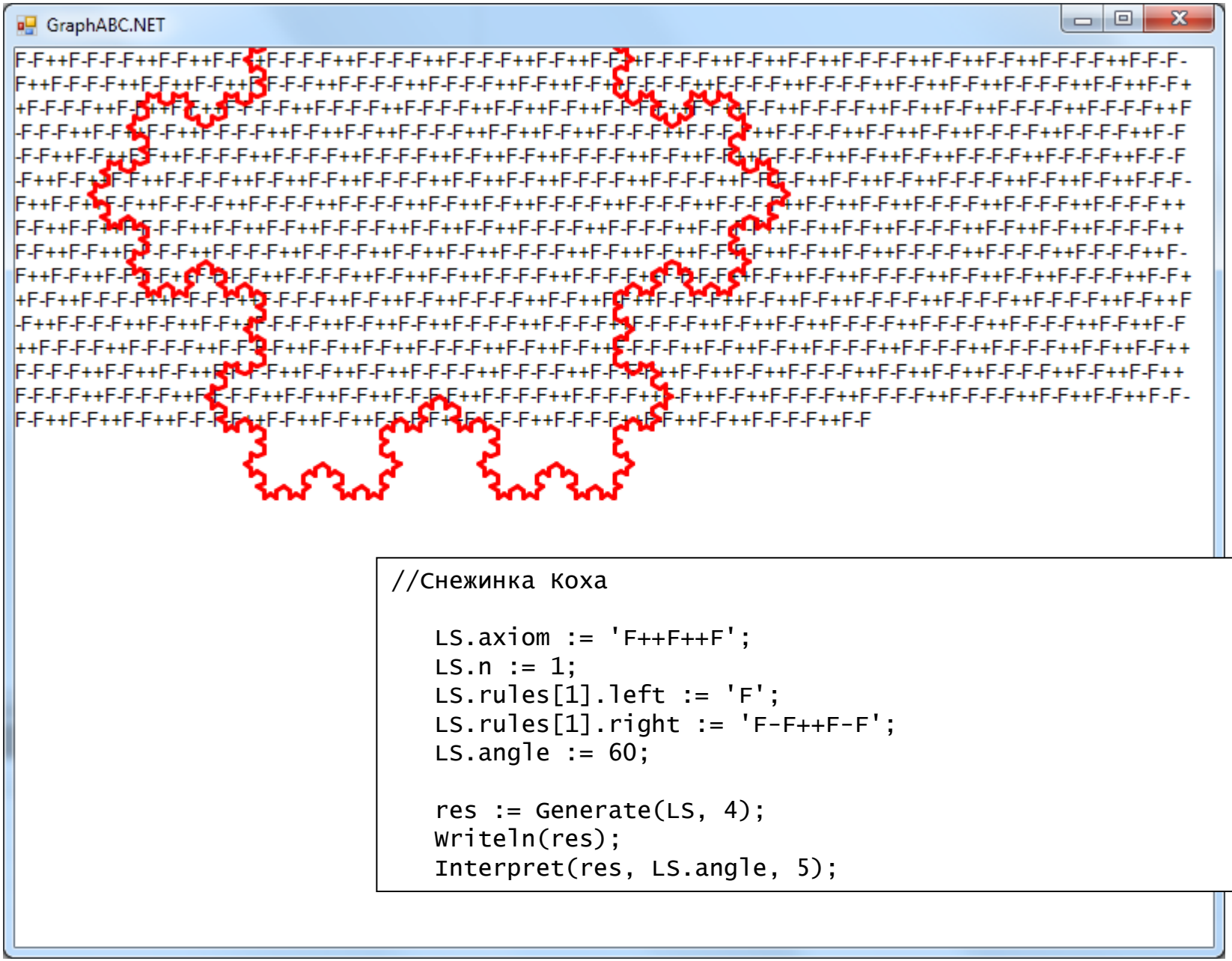
# Прямая реализация. Интерпретация

```
procedure Interpret(s: string; angle, step: real);
var
   i: integer;
begin
   SetAngle(angle);
   SetStep(step);
   for i := 1 to length(s) do
      if s[i] = 'F' then
         G
      else if s[i] = '+' then
         L
      else if s[i] = '-' then
         R;
end;
```

# Прямая реализация. Снежинка Коха

```
//Снежинка Коха

   LS.axiom := 'F++F++F';
   LS.n := 1;
   LS.rules[1].left := 'F';
   LS.rules[1].right := 'F-F++F-F';
   LS.angle := 60;

   res := Generate(LS, 4);
   Writeln(res);
   Interpret(res, LS.angle, 5);
```

# Прямая реализация. Снежинка Коха




```
//Снежинка Коха

   LS.axiom := 'F++F++F';
   LS.n := 1;
   LS.rules[1].left := 'F';
   LS.rules[1].right := 'F-F++F-F';
   LS.angle := 60;

   res := Generate(LS, 4);
   Writeln(res);
   Interpret(res, LS.angle, 5);
```

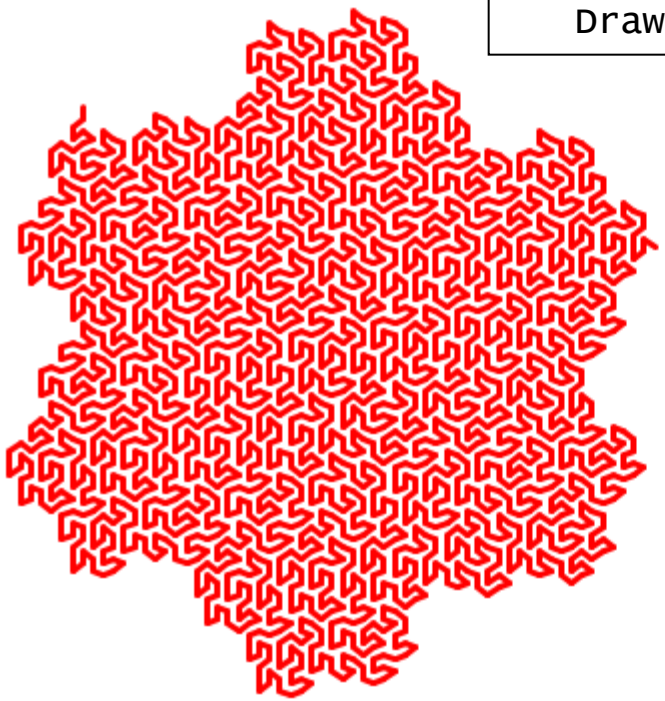
# Прямая реализация. Усовершенствование

```
procedure Axiom(var LS: tLSystem; ax: string);
begin
   LS.axiom := ax;
   LS.n := 0;
end;

procedure AddRule(var LS: tLSystem; left: char; right: string);
begin
   LS.n := LS.n + 1;
   LS.rules[LS.n].left := left;
   LS.rules[LS.n].right := right;
end;

procedure Angle(var LS: tLSystem; angle: real);
begin
   LS.angle := angle;
end;

// Рисование
procedure Draw(LS: tLSystem; generation: integer; step: real);
begin
   Interpret(Generate(LS, generation), LS.angle, step);
end;
```

# Прямая реализация. Кривая Госпера

GraphABC.NET

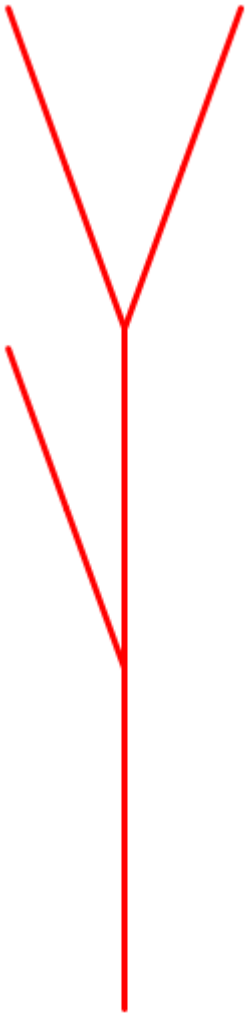
```
Axiom(LS, 'XF');
AddRule(LS, 'X', 'X+YF++YF-FX--FXFX-YF+');
AddRule(LS, 'Y', '-FX+YFYF++YF+FX--FX-Y');
Angle(LS, 60);

Draw(LS, 4, 6);
```



# Моделирование растений. Скобочные *L*-системы

**Куст**

Аксиома: X
Правила:
F → FF
X → F[+X]F[-X]+X
Угол: 20º



[ – запомнить состояние черепахи (Store)
] – вспомнить состояние черепахи (Restore)

# Моделирование растений. Скобочные *L*-системы

```
procedure Interpret(s: string; angle, step: real);
var
   i: integer;
begin
   SetAngle(angle);
   SetStep(step);
   for i := 1 to length(s) do
      if s[i] = 'F' then
         G
      else if s[i] = '+' then
         L
      else if s[i] = '-' then
         R
      else if s[i] = '[' then
         Store
      else if s[i] = ']' then
         Restore;
end;
```

# Моделирование растений. Скобочные *L*-системы

**Куст**

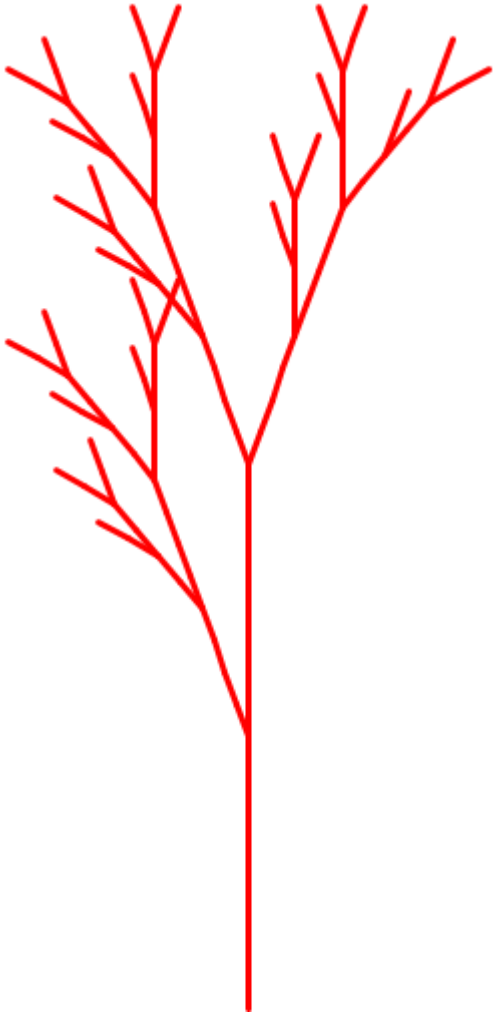
Аксиома: X
Правила:
F → FF
X → F[+X]F[-X]+X
Угол: 20º

```
Axiom(LS, 'X');
AddRule(LS, 'F', 'FF');
AddRule(LS, 'X', 'F[+X]F[-X]+X');
Angle(LS, 20);

ScaleDraw(LS, 3);
```

# Моделирование растений. Скобочные *L*-системы

**Куст**

Аксиома: X
Правила:
F → FF
X → F[+X]F[-X]+X
Угол: 20º

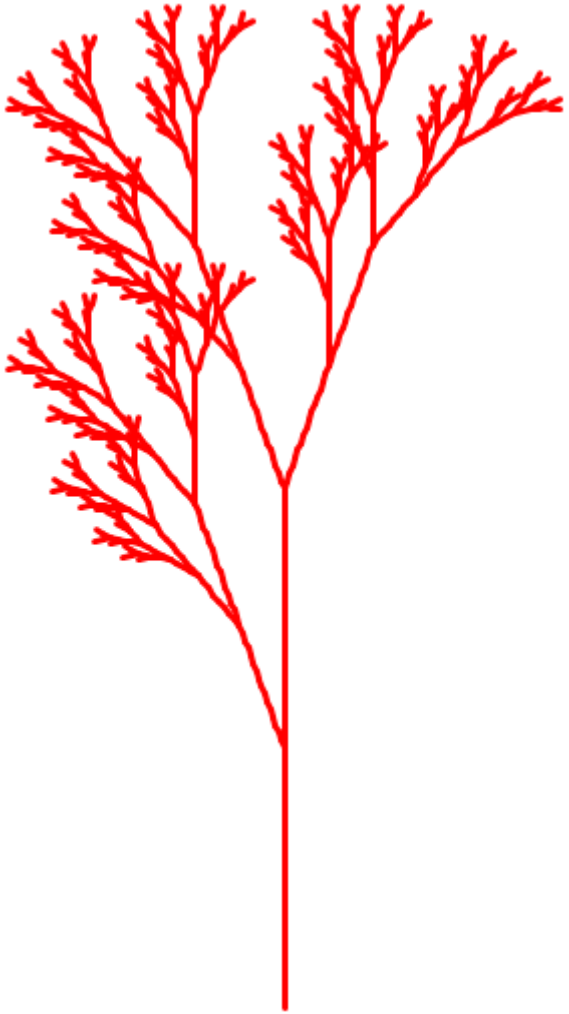


```
Axiom(LS, 'X');
AddRule(LS, 'F', 'FF');
AddRule(LS, 'X', 'F[+X]F[-X]+X');
Angle(LS, 20);

ScaleDraw(LS, 4);
```

# Моделирование растений. Скобочные *L*-системы

**Куст**

Аксиома: X
Правила:
F → FF
X → F[+X]F[-X]+X
Угол: 20º

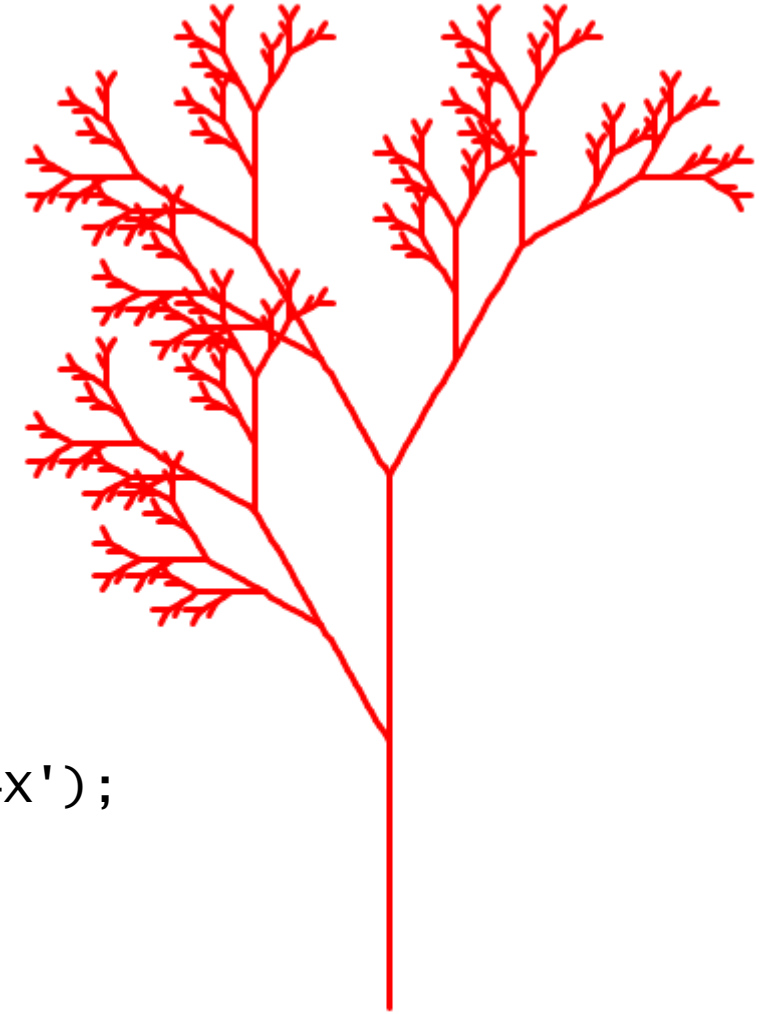


```
Axiom(LS, 'X');
AddRule(LS, 'F', 'FF');
AddRule(LS, 'X', 'F[+X]F[-X]+X');
Angle(LS, 20);

ScaleDraw(LS, 6);
```

# Моделирование растений. Скобочные *L*-системы

**Куст**

Аксиома: X
Правила:
F → FF
X → F[+X]F[-X]+X
Угол: 20º

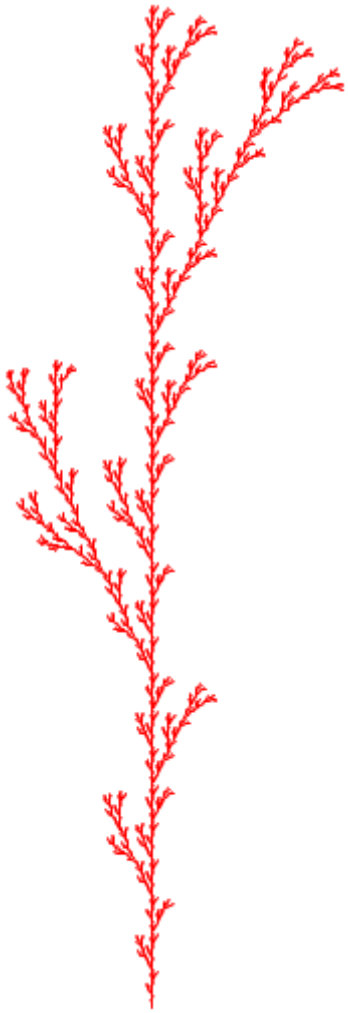


```
Axiom(LS, 'X');
AddRule(LS, 'F', 'FF');
AddRule(LS, 'X', 'F[+X]F[-X]+X');
Angle(LS, 30);

ScaleDraw(LS, 6);
```

# Моделирование растений. Скобочные *L*-системы

**Сорняк**

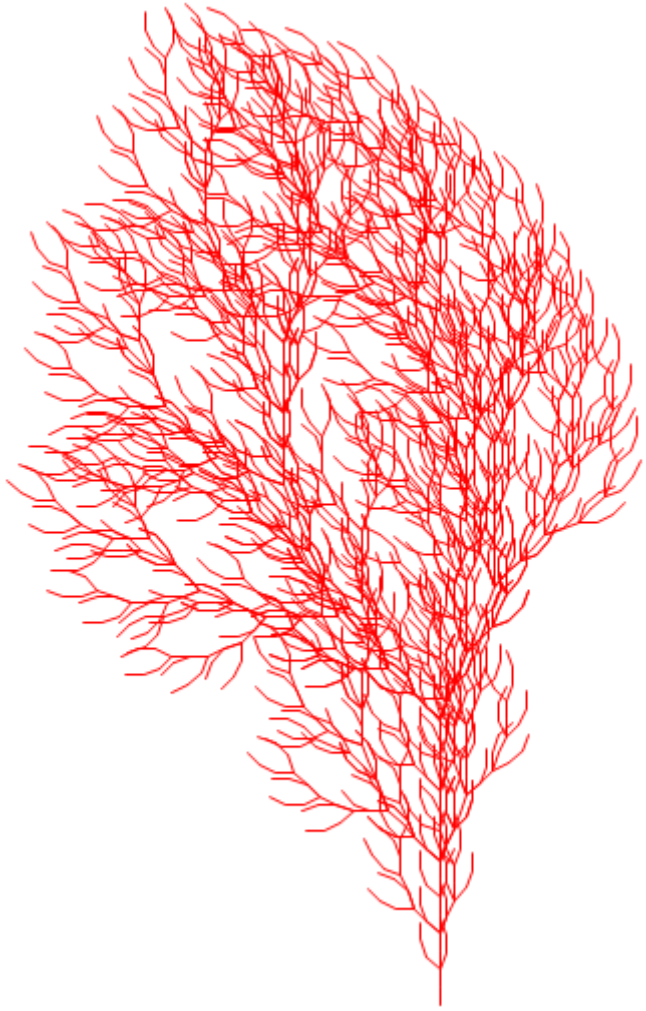
Аксиома: F
Правило: F → F[+F]F[-F]F
Угол: 180/7º



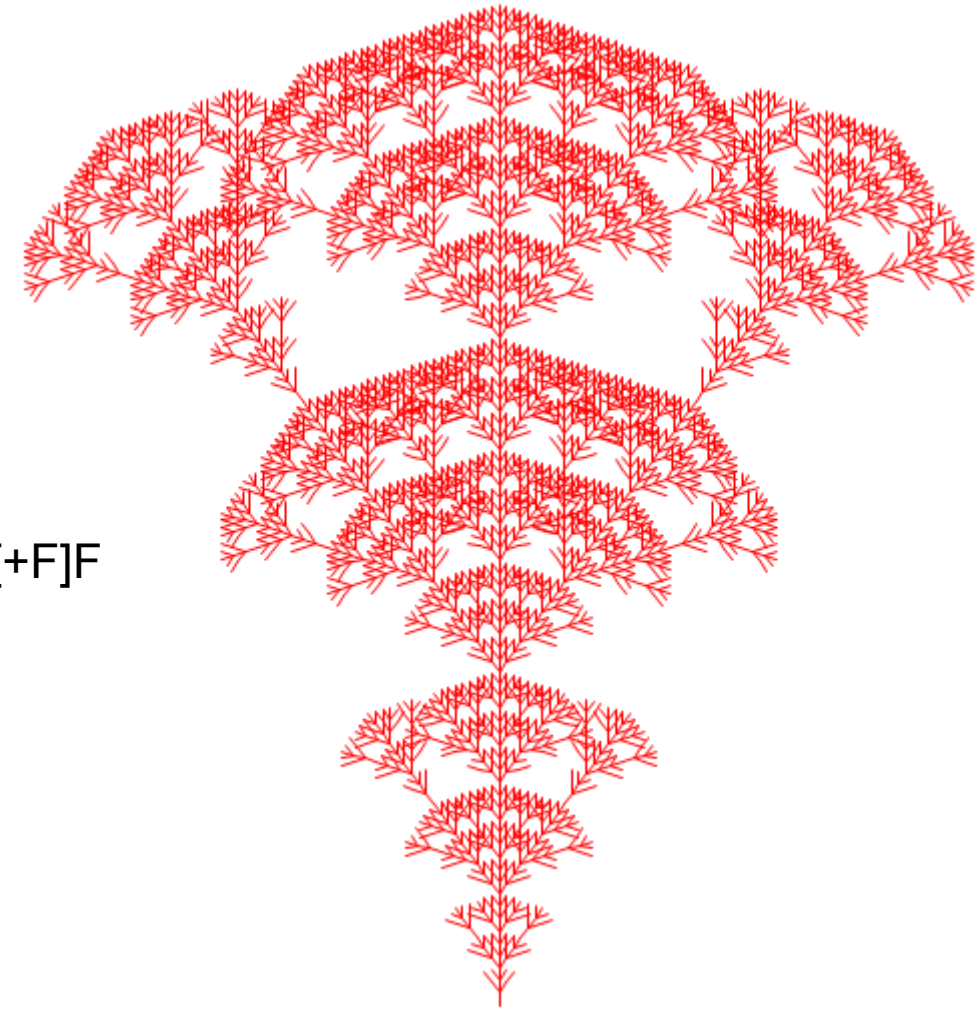
# Моделирование растений. Скобочные *L*-системы

**Куст**

Аксиома: F
Правило: F → FF+[+F-F-F]-[-F+F+F]
Угол: 180/8º



# Моделирование растений. Скобочные *L*-системы



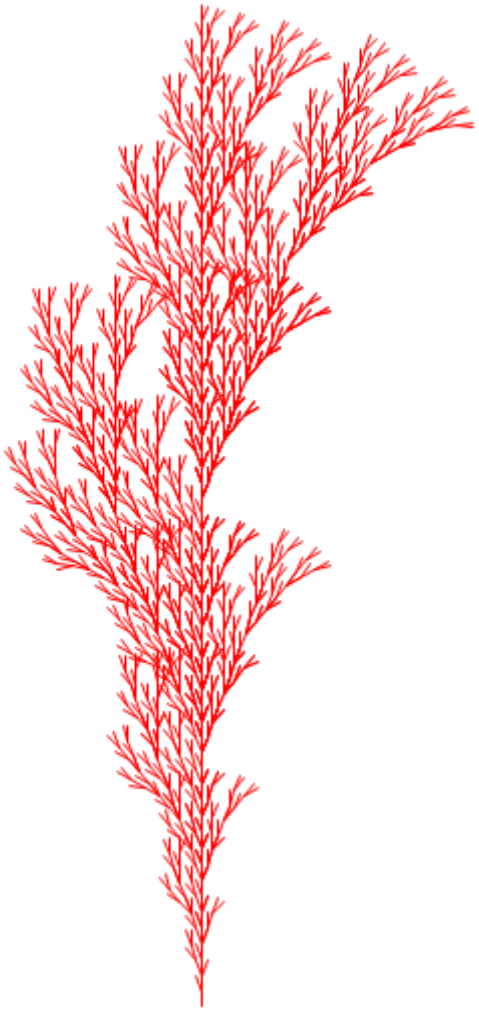
**Сорняк**

Аксиома: F
Правило: F → F[+FF][-FF]F[-F][+F]F
Угол: 180/5º

# Моделирование растений. Скобочные *L*-системы

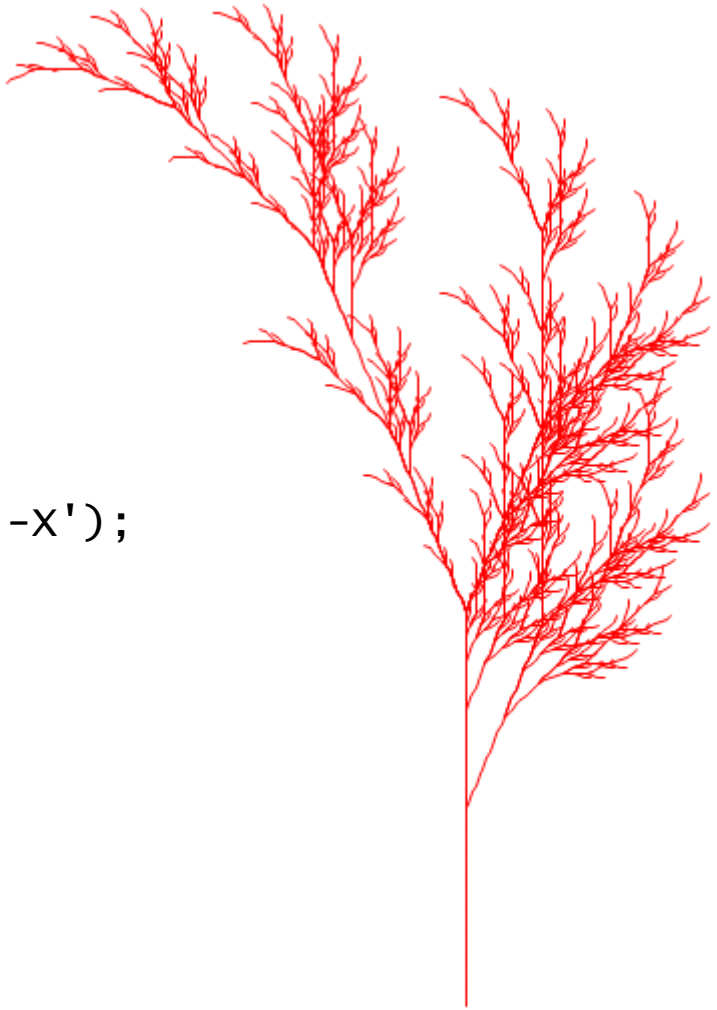
**Растение**

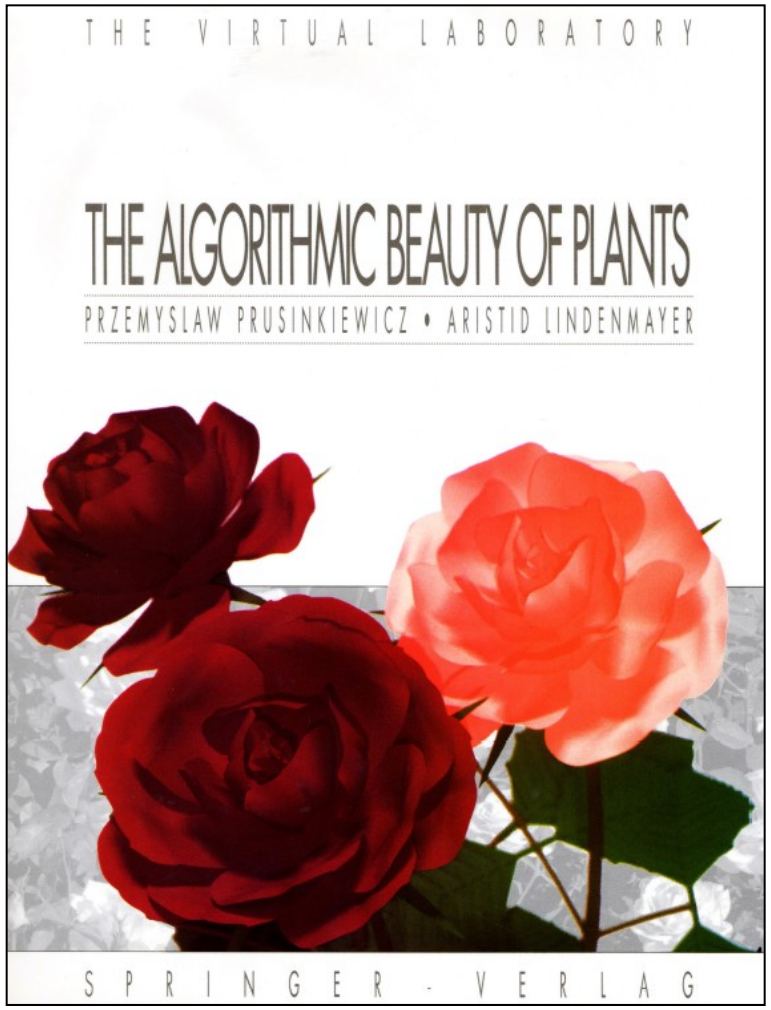
```
Axiom(LS, 'F');
AddRule(LS, 'F', 'F[+F]F[-F][F]');
Angle(LS, 20);
```

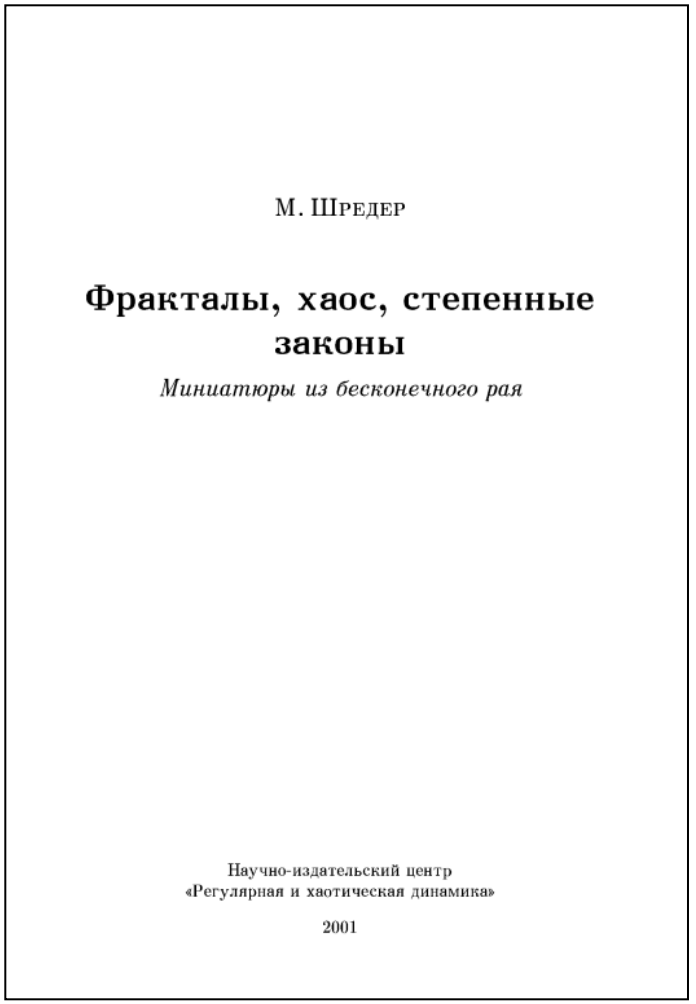


# Моделирование растений. Скобочные *L*-системы

```
//Растение

   Axiom(LS, 'X');
   AddRule(LS, 'F', 'FF');
   AddRule(LS, 'X', 'F-[[X]+X]+F[+FX]-X');
   Angle(LS, 22.5);

   ScaleDraw(LS, 6);
```






https://ru.wikipedia.org/wiki/L-система