ΚΕΦΑΛΑΙΟ 5

ΕΝΑΣ ΔΙΑΦΟΡΕΤΙΚΟΣ ΤΡΟΠΟΣ ΚΑΤΑΣΚΕΥΗΣ FRACTALS

5.1 ΜΕΡΙΚΕΣ ΒΑΣΙΚΕΣ ΠΑΡΑΤΗΡΗΣΕΙΣ ΣΤΗΝ ΚΑΤΑΣΚΕΥΗ ΔΕΝΤΡΩΝ.

Εστω $t \to \infty$ το συνολικό μήκος και η γωνία είναι $w = 1=0$. Εάν είναι $w = 0$ τότε το δέντρο εκφυλίζεται σε μία ευθεία. Για λόγους ευκολίας παίρνουμε ότι αρχικά $s = 1$. Μεταβάλλοντας τον παράγοντα σμίκρυνσης f μας ενδιαφέρουν τα ακόλουθα ερωτιματικά.

a) Ποιό είναι το συνολικό μήκος γιά όλους τους κλάδους;

b) Μπορεί το δέντρο να περιοριστεί με μία πεπερασμένη επιφάνεια;

Ο αριθμός των κλάδων στο n-ιοστό επίπεδο δίνεται με την γεωμετρική ακολουθία $a_n = 2^n$. Ο συνολικός αριθμός όλων των κλάδων είναι το άθροισμα

$$A_k = \Sigma_{i=1}^{k} 2^n.$$

Τα μήκη ενός μόνου κλάδου μετά από την n διακλάδωση είναι $s_n = f^{n-1}$.

a) Συνολικό μήκος.

$$G = \lim_{k \to \infty} G_k = \Sigma_{i=1}^{\infty} 2^{n-1} f^n.$$

Εχουμε λοιπόν μία γεωμετρική σειρά με λόγο $q = 2 * f$. Εάν $q < 1$ τότε το G είναι πεπερασμένο. Δηλαδή εαν είναι ο $f < 1/2$. Στην περίπτωση που ο $f > 1/2$ δεν είναι πεπερασμένο το μήκος. Για $f = 1/2$ τότε το άθροισμα όλων των μηκών είναι το ίδιο σε κάθε επίπεδο διακλάδωσης.

b) Ο περιορισμός του σχήματος.

Η επέκταση ενός δέντρου μπορεί να εκτιμηθεί με βάση την απόσταση που έχουν τα φύλλα του δέντρου από το πρώτο επίπεδο διακλάδωσης. Τα μήκη ενός δρόμου διαμέσου του δέντρου μέχρι το φύλλο είναι

$$l = \lim_{k \to \infty} l_k = \Sigma_{i=1}^{\infty} s_k = \Sigma_{k=1}^{\infty} f^k.$$

Για $f > 1$ απειρίζεται, άρα το σχήμα επεκτείνεται πέρα από κάθε όριο. Για $f < 1$ ο δρόμος είναι πεπερασμένος. Υπάρχει πάλι μία γεωμετρική σειρά με $q = f$ και για το συνολικό μήκος ισχύει

$$l = \Sigma_{k=1}^{\infty} f^k = 1/(1- f), \quad (0 <= f < 1).$$

Ενας κύκλος με κέντρο στο πρώτο σημείο διακλάδωσης και ακτίνας $r(f) = 1/(1- f)$ περικλείει το δέντρο. Από εδώ προκύπτει για $0.5 < f < 1$ η κατάσταση όπου ένα αντικείμενο με άπειρο μήκος πλευρών μπορεί να περικλείεται από μία πεπερασμένη επιφάνεια.

Η απόσταση μπορεί να εκτιμηθεί με ακρίβεια. Γενικά η μέγιστη Ευκλείδεια απόσταση μίας κορυφής από το πρώτο σημείο διακλάδωσης βρίσκεται εάν οδεύουμε από το πρώτο σημείο διακλάδωσης εναλλακτικά μία αριστερά μία δεξιά για να φτάσουμε στο φύλλο. Αρα σε ένα άρτιο αριθμό αποστάσεων η ολική απόσταση βγαίνει ως εξής:

Αρα γενικά η απόσταση $A_{2n}\ A_{2*(n+1)}$ η μεγάλη πλευρά τριγώνου με μήκη πλευρών f^n, f^{2*n+1} και γωνία $180 - w$.

ΔΥΟ ΘΕΜΕΛΙΩΔΕΣ ΤΡΟΠΟΙ ΚΑΤΑΣΚΕΥΗΣ FRACTALS.

a) Υπάρχουν επιφάνειες (δύο διαστάσεων) με fractal διάσταση μεγαλύτερη του δύο:
b) Ποιά είναι η διάσταση πού έχουν οι fractals καμπύλες:

Οι fractal καμπύλες έχουν μία κλασματική διάσταση και μπορούν να παραχθούν με αναδρομή.

Για μία καλύτερη εντύπωση της καμπύλης μπορούμε να την μεγενθύνουμε κατά ένα παράγοντα v .Εάν κατά την μεγένθυση εμφανιστούν a νέα αντίγραφα της αρχικής εικόνας τότε η κλασματική διάσταση D της αρχικής καμπύλης είναι ο εκθέτης στο v για να φτάσουμε στο a .

$$v^D = a \Rightarrow D * \log v = \log a \Rightarrow D = \log a / \log v .$$

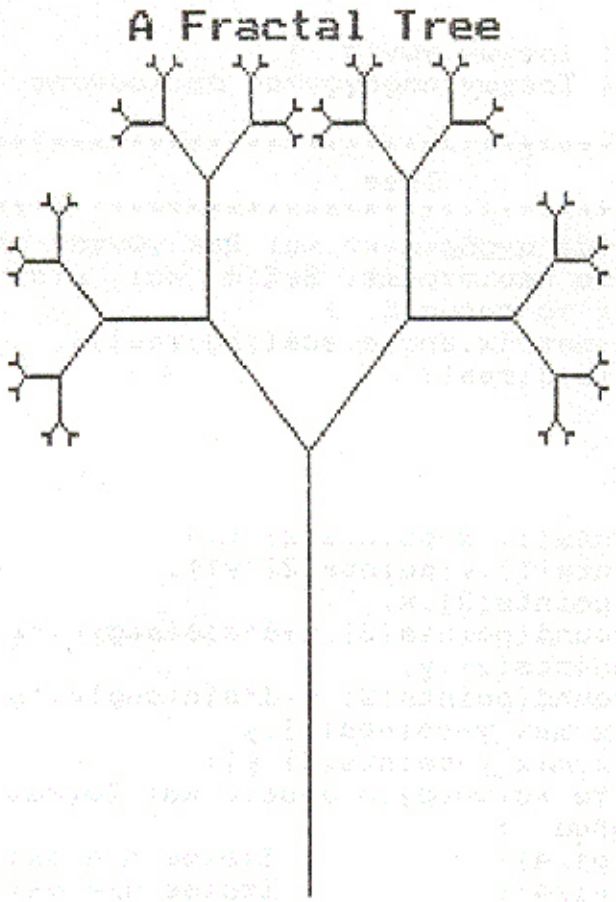
Αντίθετα από την αναδρομική σχέση, εάν το ευθύγραμμο τμήμα (initiator) αντικατασταθεί από μία βασική εικόνα που λέγεται εναρκτής (generator) με a πλευρές και f είναι ο παράγοντας συίκρυνσης για το μήκος των νέων a πλευρών τότε:

$$D = \log a / \log (1/ f).$$

Οσο ευκρινέστερα αποκλίνει η βασική φιγούρα δηλαδή ο εναρκτής από το αρχικό ευθύγραμμο τμήμα και όσο μεγαλύτερη τάση έχει να βγεί και να καταλάβει το επίπεδο τόσο μεγαλύτερη είναι η fractal διάσταση του.

```
program Fractal_Tree;

{ Το πρόγραμμα αυτό δημιουργεί ένα συμμετρικό με αναδρομή με το
  ακόλουθο τρόπο:
  Αρχίζει από ένα ευθύγραμμο τμήμα μήκους s και στην συνέχεια το
  περιστρέφει μία αριστερά και μία δεξιά έχωντας ελλαττώσει το
  μήκος του αρχικού τμήματος κατά ένα συντελεστή scaling. }

USES CRT, GRAPH;

const CharSize=8;
      pi=3.14;
      max_y=200;
      N=2;

type matrix = array [1..N] of PointType;
     { Περιέχει τα τρέχων δύο σημεία
       που    ορίζουν    το    τρέχων
       ευθύγραμμο τμήμα. }

var Gd,Gm:integer;
    points:matrix;
    angle,                { Τρέχων γωνία. }
    scaling:real;         { Τρέχων παράγοντας σμίκρυνσης. }

{*****************************************************************}
{*                          Tree                                 *}
{*****************************************************************}
{ Η procedure αυτή είναι αναδρομική και ξεκινώντας από ένα ευθύγραμμο
  τμήμα μήκους d μία το περιστρέφει δεξιά και μία αριστερά έχωντας
  κατάλληλα ελλαττώσει το τμήμα d. }
procedure tree(points:matrix;angle,scaling:real);
var angle1,angle2,scale,d:real;
    points1:matrix;
BEGIN
  If scaling>0.2 then
    begin
      d:=sqrt(sqr(points[1].x-points[2].x)+
              sqr(points[1].y-points[2].y));
       points1[1].x:=points[2].x;
      points1[2].x:=round(points[2].x+d*cos(angle)*scaling);
      points1[1].y:=points[2].y;
      points1[2].y:=round(points[2].y+d*sin(angle)*scaling);
      line(points[1].x,max_y-points[1].y,
           points[2].x,max_y-points[2].y);
      { Υπολογίζεται το καινούργιο σημείο και ζωγραφίζεται το
        ευθύγραμμο τμήμα. }
      angle1:=angle+(pi/4);          { Στρίψε π/4 ακτίνια αριστερά. }
      angle2:=angle-(pi/4);          { Στρίψε π/4 ακτίνια δεξιά. }
      scale:=scaling*0.85;           { Ελλαττώνουμε το παράγοντα
                                       σμίκρυνσης.}
      tree(points1,angle1,scale);    { Αριστερά αναδρομή. }
      tree(points1,angle2,scale)     { Δεξιά αναδρομή. }
    end
```

```
END;

BEGIN
  Gd:=Detect;                          { Αρχικοποίηση της οθόνης γραφικών. }
  InitGraph(Gd,Gm,'c:\pascal\tpu');
  if GraphResult<>grOK then Halt(1);
  ClearDevice;

  angle:=pi/2;                                  { Αρχική γωνία. }
  scaling:=1.0;
  points[1].x:=100;points[2].x:=100;     { Αρχικό ευθύγραμμο τμήμα. }
  points[1].y:=0;points[2].y:=50;
  tree(points,angle,scaling);
  OutTextXY(55,1,'A Factal Tree');
  repeat
  until keypressed;
  CloseGraph
END.
```

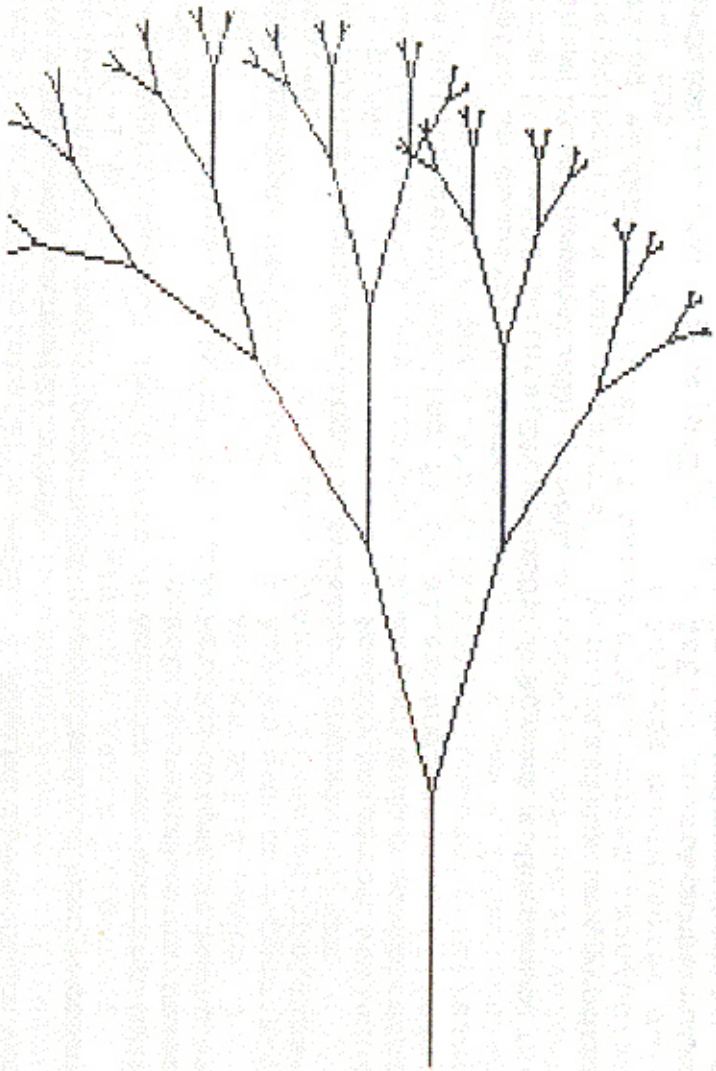


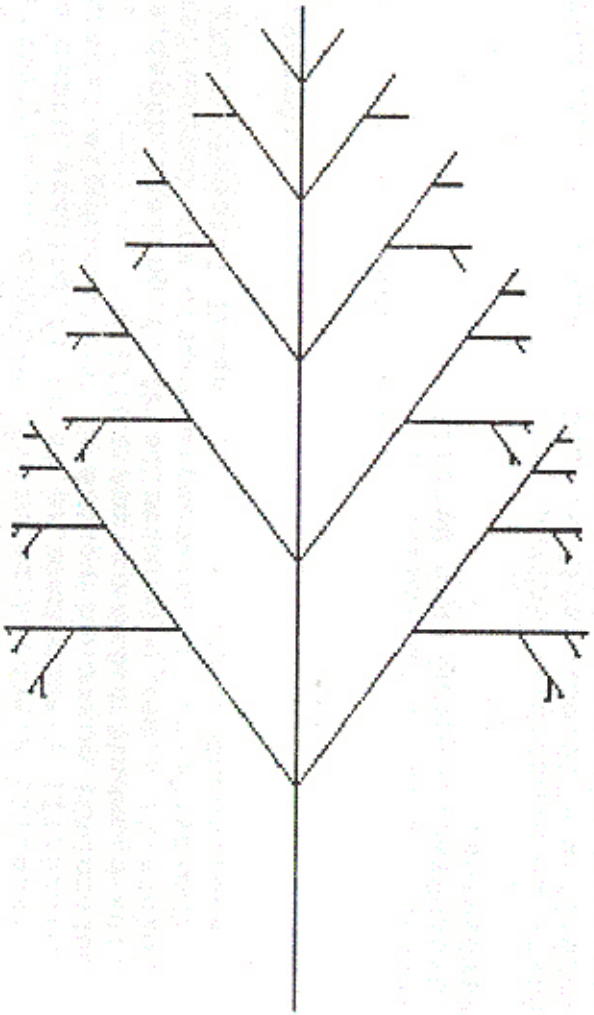
```
program Fractal_Tree;

{ Το πρόγραμμα αυτό δημιουργεί δύο μή συμετρικά και ένα συμμετρικό
  δέντρο με αναδρομή με το ακόλουθο τρόπο:
  Αρχίζει από ένα ευθύγραμμο τμήμα μήκους s και στην συνέχεια το
  περιστρέφει μία αριστερά και μία δεξιά έχωντας ελλαττώσει το
  μήκος του αρχικού τμήματος κατά ένα συντελεστή scaling.Στην
  περίπτωση του συμμετρικού η αριστερή γωνία είναι ίση με την
  δεξιά και ο αριστερός συντελεστής συστολής είναι ο ίδιος και
  για το αριστερό και για το δεξιό ευθύγραμμο τμήμα του
  κλάδου.Στην περίπτωση των μή συμμετρικών δέντρων όχι μόνο η
  αριστερή και η δεξιά γωνία δεν είναι ίσες αλλά και ο
  συντελεστής σμίκρυνσης του δεξιού τμήματος είναι διαφορετικός
  από τον συντελεστή σμίκρυνσης του αριστερού ευθύγραμμου
  τμήματος. }

USES CRT, GRAPH;

const CharSize=8;
      pi=3.14;
      max_y=300;
      N=2;

type matrix = array [1..N] of PointType;
    { Περιέχει τα τρέχων δύο σημεία
      που ορίζουν το τρέχων
      ευθύγραμμο τμήμα. }

var Gd,Gm,
    depth:integer;    { Βάθος της αναδρομής. }
    points:matrix;
    fl,               { Αριστερός συντελεστής σμίκρυνσης. }
    al,               { Αριστερή γωνία περιστροφής. }
    fr,               { Δεξιός συντελεστής σμίκρυνσης. }
    ar,               { Δεξιά γωνία περιστροφής. }
    scaling,          { Τρέχων συντελεστής σμίκρυνσης. }
    angle:real;       { Τρέχων γωνία. }

(*************************************************************)
(*                          Tree                             *)
(*************************************************************)
{ Η procedure αυτή είναι αναδρομική και ξεκινώντας από ένα ευθύγραμμο
  τμήμα μήκους d μία το περιστρέφει δεξιά και μία αριστερά έχωντας
  κατάλληλα ελλαττώσει το τμήμα d. }
procedure tree(points:matrix;depth:integer;scaling,
               fl,fr,al,ar:real;var angle:real);
var d:real;
    points1:matrix;
BEGIN
  If depth<>0 then
    begin
      angle:=angle+al;                { Στρίψε al ακτίνια αριστερά. }
      d:=sqrt(sqr(points[1].x-points[2].x)+
              sqr(points[1].y-points[2].y));
      points1[1].x:=points[2].x;
```

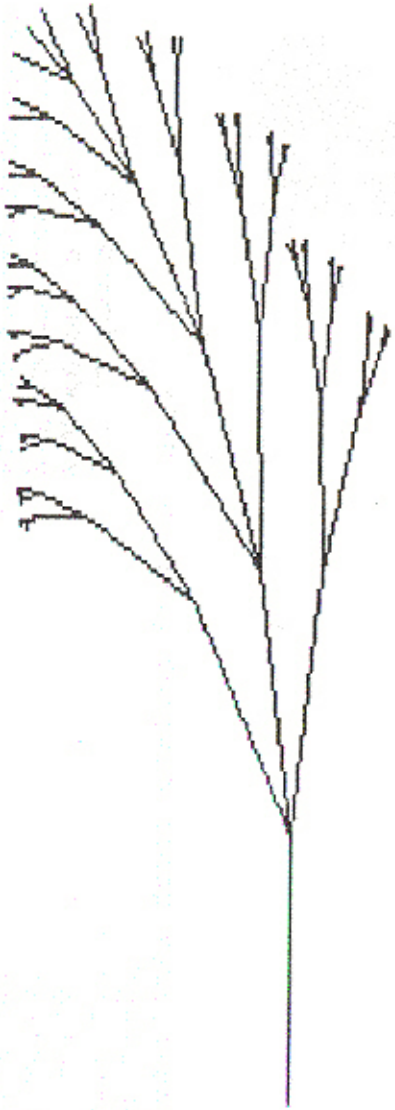
```
        points1[2].x:=round(points[2].x+d*cos(angle)*scaling);
        points1[1].y:=points[2].y;
        points1[2].y:=round(points[2].y+d*sin(angle)*scaling);
        line(points[1].x,max_y-points[1].y,
             points[2].x,max_y-points[2].y);
        { Υπολογίζεται το καινούργιο σημείο και ζωγραφίζεται το
          αριστερό ευθύγραμμο τμήμα. }
        tree(points1,depth-1,fl*scaling,fl,fr,al,ar,angle);

        angle:=angle-al-ar;                { Στρίψε al+ar ακτίνια δεξιά. }
        d:=sqrt(sqr(points[1].x-points[2].x)+
                sqr(points[1].y-points[2].y));
        points1[1].x:=points[2].x;
        points1[2].x:=round(points[2].x+d*cos(angle)*scaling);
        points1[1].y:=points[2].y;
        points1[2].y:=round(points[2].y+d*sin(angle)*scaling);
        line(points[1].x,max_y-points[1].y,
             points[2].x,max_y-points[2].y);
        { Υπολογίζεται το καινούργιο σημείο και ζωγραφίζεται το δεξιό
          ευθύγραμμο τμήμα. }
        tree(points1,depth-1,fr*scaling,fl,fr,al,ar,angle);

        angle:=angle+ar;                   { Στρίψε ar ακτίνια αριστερά. }
        d:=sqrt(sqr(points[1].x-points[2].x)+
                sqr(points[1].y-points[2].y));
        points1[1].x:=points[2].x;
        points1[2].x:=round(points[2].x+d*cos(angle)*scaling);
        points1[1].y:=points[2].y;
        points1[2].y:=round(points[2].y+d*sin(angle)*scaling);
        line(points[1].x,max_y-points[1].y,
        points[2].x,max_y-points[2].y)
        { Υπολογίζεται το καινούργιο σημείο και ζωγραφίζεται το
          ευθύγραμμο τμήμα. }
    end
END;

BEGIN
  Gd:=Detect;                          { Αρχικοποίηση της οθόνης γραφικών. }
  InitGraph(Gd,Gm,'c:\pl\pascal');
  if GraphResult<>grOK then Halt(1);
  ClearDevice;

  { Πρώτο μή συμμετρικό δέντρο. }
  points[1].x:=200;points[2].x:=200; { Αρχικό ευθύγραμμο τμήμα. }
  points[1].y:=20;points[2].y:=80;
  line(points[1].x,max_y-points[1].y,points[2].x,max_y-points[2].y);
  scaling:=1.0;
  fl:=0.89;                            { Αρχικός συντελεστής συμπύκνωσης. }
  fr:=0.735;                           { Δεξιός συντελεστής συμπύκνωσης. }
  al:=pi/9;                            { Αρχική αριστερή γωνία. }
  ar:=pi/9;                            { Αρχική δεξιά γωνία. }
  angle:=pi/2;                         { Αρχική γωνία. }
  depth:=7;
  tree(points,depth,scaling,fl,fr,al,ar,angle);
```

```
  { Δεύτερο συμμετρικό δέντρο. }
  points[1].x:=400;points[2].x:=400; { Αρχικό ευθύγραμμο τμήμα. }
  points[1].y:=0;points[2].y:=50;
  line(points[1].x,max_y-points[1].y,points[2].x,max_y-points[2].y);
  scaling:=1.0;
  fl:=0.65;                          { Αρχικός συντελεστής συίκρυνσης. }
  fr:=0.90;                          { Δεξιός συντελεστής συίκρυνσης. }
  al:=pi/4;                          { Αρχική αριστερή γωνία. }
  ar:=0;                             { Αρχική δεξιά γωνία. }
  angle:=pi/2;                       { Αρχική γωνία. }
  depth:=6;
  tree(points,depth,scaling,fl,fr,al,ar,angle);
  al:=-pi/4;                         { Αρχική αριστερή γωνία. }
  ar:=0;                             { Αρχική δεξιά γωνία. }
  tree(points,depth,scaling,fl,fr,al,ar,angle);

  { Τρίτο μή συμμετρικό δέντρο. }
  points[1].x:=600;points[2].x:=600; { Αρχικό ευθύγραμμο τμήμα. }
  points[1].y:=0;points[2].y:=60;
  line(points[1].x,max_y-points[1].y,points[2].x,max_y-points[2].y);
  scaling:=1.0;
  fl:=0.635;                         { Αρχικός συντελεστής συίκρυνσης. }
  fr:=0.89;                          { Δεξιός συντελεστής συίκρυνσης. }
  al:=pi/6;                          { Αρχική αριστερή γωνία. }
  ar:=-pi/18;                        { Αρχική δεξιά γωνία. }
  angle:=pi/2;                       { Αρχική γωνία. }
  depth:=6;
  tree(points,depth,scaling,fl,fr,al,ar,angle);
  al:=-pi/21;                        { Αρχική αριστερή γωνία. }
  ar:=-pi/18;                        { Αρχική δεξιά γωνία. }
  tree(points,depth,scaling,fl,fr,al,ar,angle);

  OutTextXY(270,1,'Fractal Trees');
  repeat
  until keypressed;
  CloseGraph
EN
```

Fractal Trees





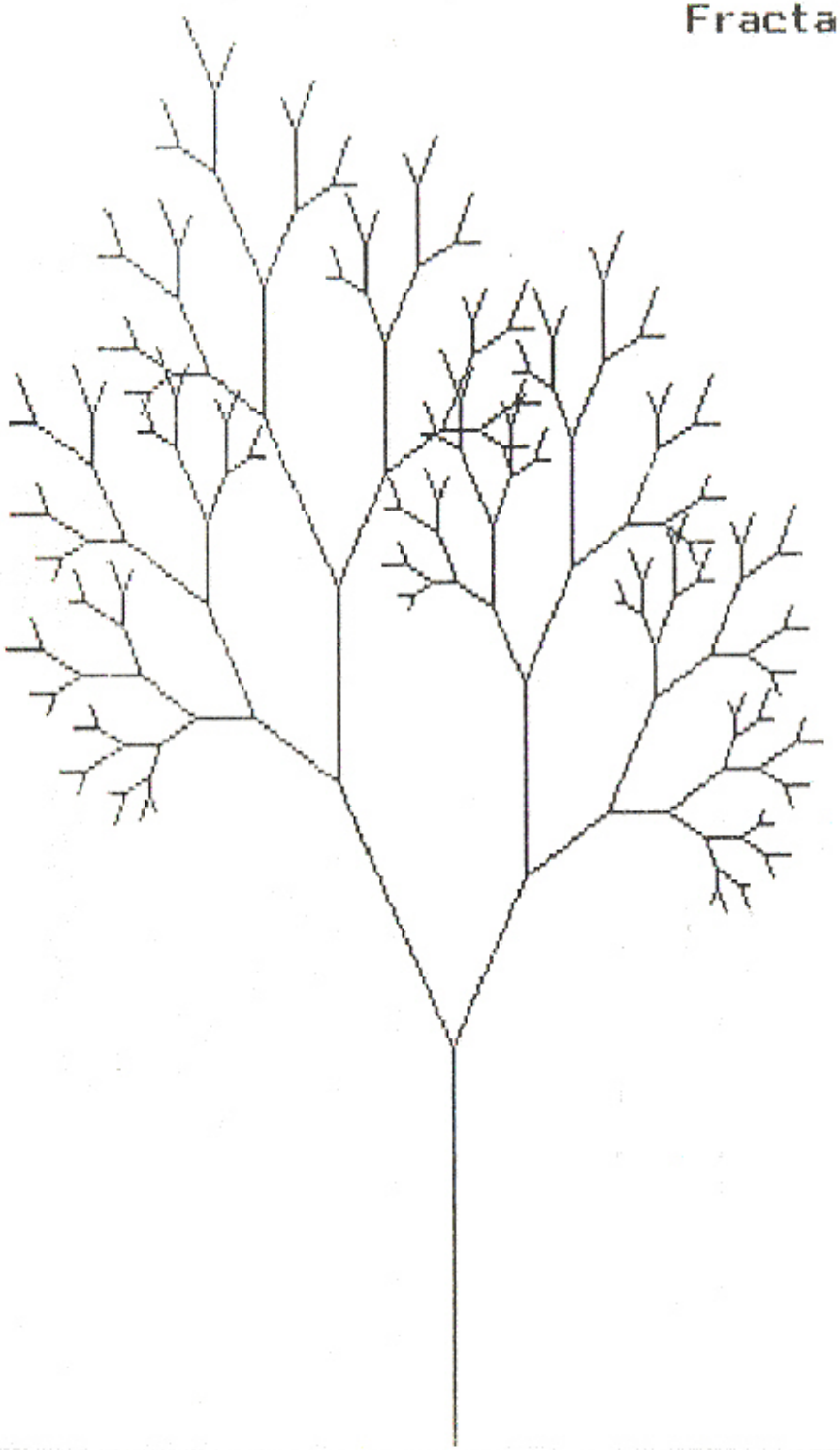


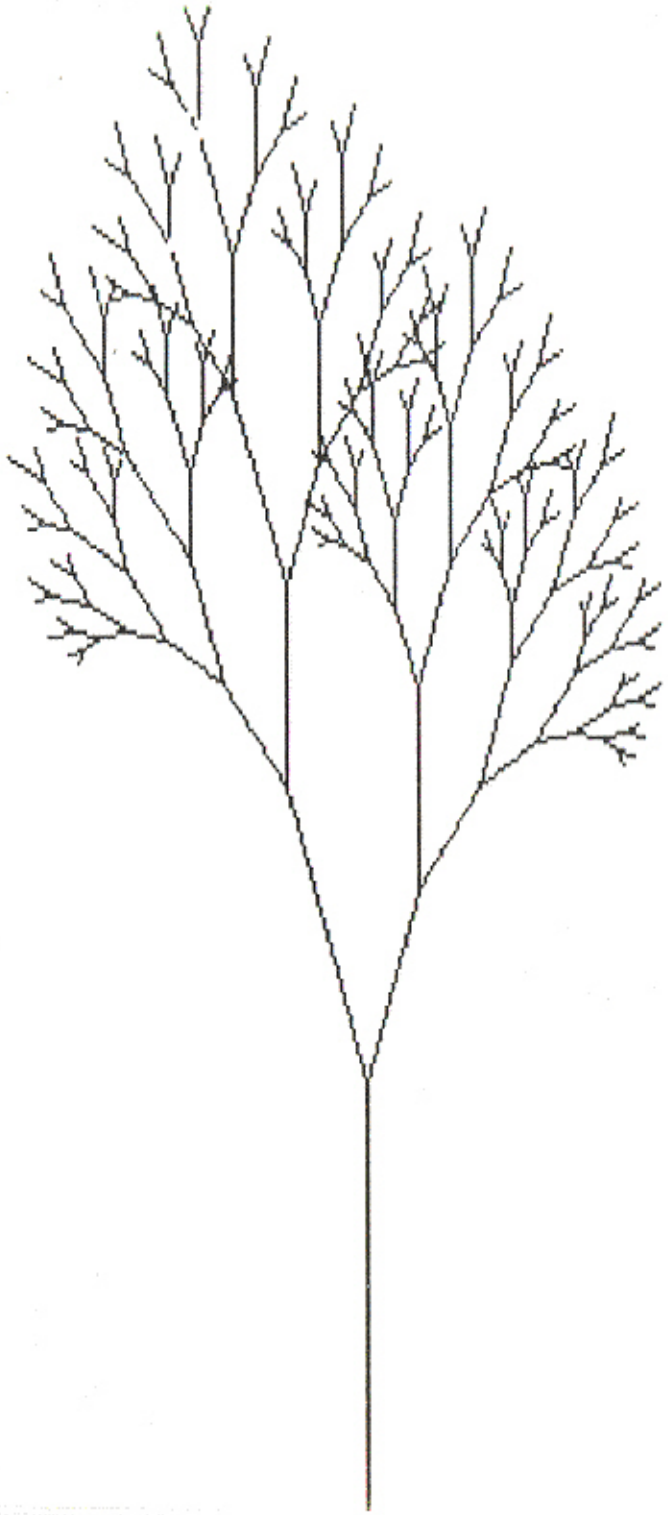
```
program Another_Fractal_Trees;

USES CRT, GRAPH;

const CharSize=8;
      pi=3.14;
      max_y=320;
      factor=0.65;              { Συντελεστής συστολής. }
      a1=pi/6;                  { Γωνία στροφής 30 μοιρών. }
      a2=pi/9;                  { Γωνία στροφής 20 μοιρών. }
      dinstance=70;             { Αρχικό ευθύγραμμο τμήμα. }
      max_depth=7;

var Gd,Gm,depth:integer;
    point:PointType;            { Το τρέχων σημείο. }
    f,                          { Συντελεστής συστολής. }
    a,                          { Γωνία στροφής. }
    d,                          { Ευθύγραμμο τμήμα. }
    angle:real;                 { Τρέχων γωνία. }

(*************************************************************************)
(*                              Draw                                     *)
(*************************************************************************)
{ Ζωγραφίζει ένα ευθύγραμμο τμήμα από το σημείο point στο σημείο  pnt.
  Η απόσταση του pnt από το point είναι s και το σημείο pnt βρίσκεται
  σε μία γωνία angle από το σημείο point. }
procedure draw(var point:PointType;s,angle:real);
var pnt:PointType;
BEGIN
     pnt.x:=round(point.x+s*cos(angle));
     pnt.y:=round(point.y+s*sin(angle));
     line(pnt.x,max_y-pnt.y,point.x,max_y-point.y);
     point:=pnt
END;

(*************************************************************************)
(*                              Left_tree                                *)
(*************************************************************************)
{ Σχεδιάζει το δέντρο. }
procedure left_tree(point:PointType;depth:integer;
                    f,a,d:real;var angle:real);forward;
(*************************************************************************)
(*                              Right_tree                               *)
(*************************************************************************)
procedure right_tree(point:PointType;depth:integer;
                     f,a,d:real;var angle:real);
var point1:PointType;
BEGIN
  if depth<>0 then
    BEGIN
      angle:=angle-a;                   { Στρίψε α ακτίνια δεξιά. }
      point1:=point;
      draw(point,d,angle);              { Ζωγράφισε ευθύγραμμο τμήμα
                                          μήκους d.}
      left_tree(point,depth-1,f,a,d,angle);
```

```
      point:=point1;                    { Επέστρεψε ένα βήμα πίσω. }
      angle:=angle+2*a;                 { Στρίψε 2*a ακτίνια αριστερά. }
      point1:=point;
      draw(point.d*f,angle);            { Ζωγράφισε ευθύγραμμο τμήμα
                                          μήκους d.}
      right_tree(point,depth-1,f,a,d*f,angle);

      point:=point1;
      angle:=angle-a                    { Στρίψε a ακτίνια δεξιά. }
    END
END;

(*************************************************************************)
(*                          Left_tree                                    *)
(*************************************************************************)
procedure left_tree;
var point1:PointType;
BEGIN
 If depth<>0 then
   BEGIN
     angle:=angle+a;                    { Στρίψε a ακτίνια αριστερά. }
     point1:=point;
     draw(point,d,angle);               { Ζωγράφισε ευθύγραμμο τμήμα
                                          μήκους d.}
     right_tree(point,depth-1,f,a,d*f,angle);

     point:=point1;
     angle:=angle-2*a;                  { Στρίψε 2*a ακτίνια δεξιά. }
     point1:=point;
     draw(point,d*f,angle);             { Ζωγράφισε ευθύγραμμο τμήμα
                                          μήκους d.}
     left_tree(point,depth-1,f,a,d*f,angle);

     point:=point1;
     angle:=angle+a;                    { Στρίψε a ακτίνια αριστερά. }
   END
END;


BEGIN
  Gd:=Detect;                       { Αρχικοποίηση της οθόνης γραφικών. }
  InitGraph(Gd,Gm,'c:\pl\pascal');
  if GraphResult<>grOK then Halt(1);
  ClearDevice;

  d:=dinstance;
  angle:=pi/2;
  f:=factor;
  depth:=max_depth;
  point.x:=200;point.y:=80;      { Πρώτο δέντρο με γωνία a1. }
  line(point.x,max_y-point.y+dinstance+25,point.x,max_y-point.y);
  a:=a1;
  left_tree(point,depth,f,a,d,angle);
```

```
  point.x:=500;point.y:=80;      { Δεύτερο δέντρο με γωνία a2. }
  a:=a2;
  left_tree(point,depth,f,a,d,angle);
  line(point.x,max_y-point.y+dinstance+25,point.x,max_y-point.y);
  OutTextXY(275,1,'Fractal Trees');
  repeat
  until keypressed;
  CloseGraph
EN
```

Fractal Trees



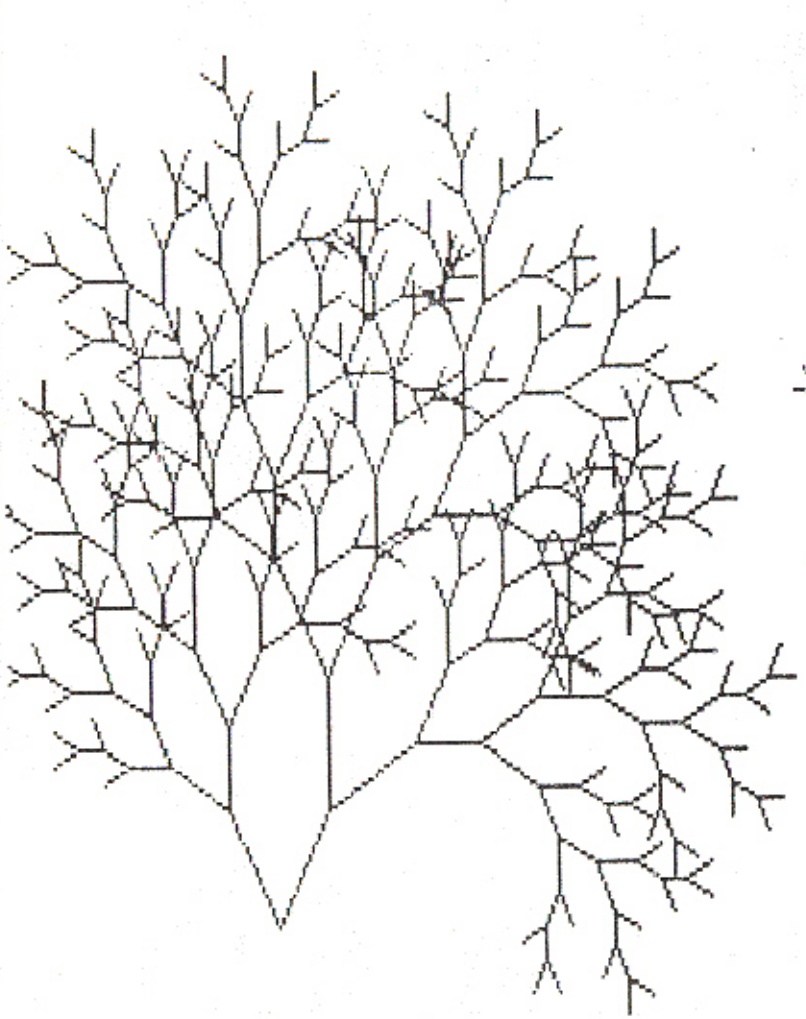


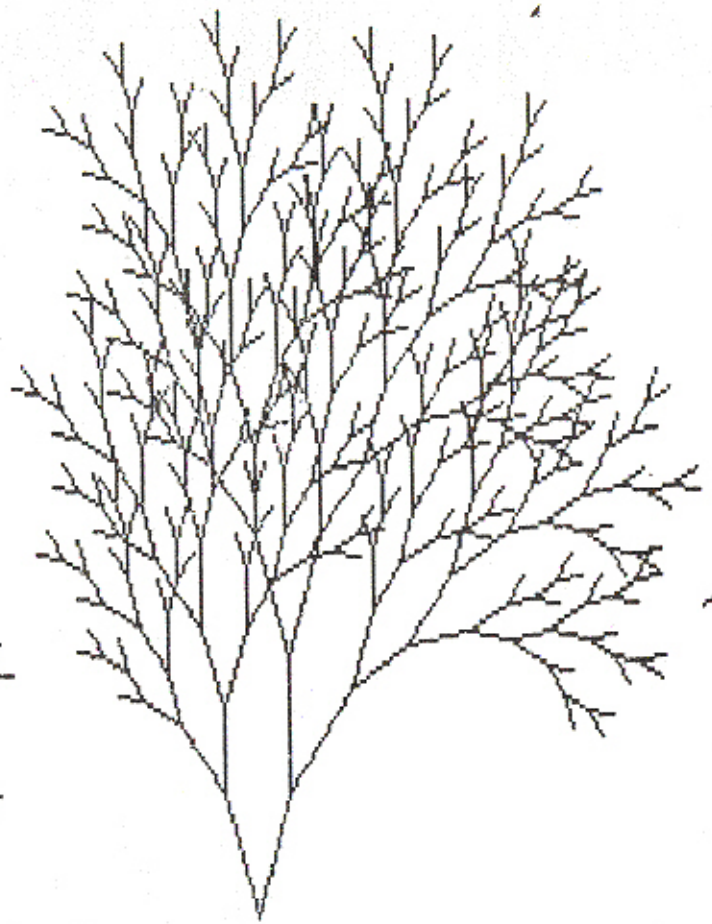
```
program Another_Fractal_Trees;

USES CRT, GRAPH;

const CharSize=8;
      pi=3.14;
      max_y=320;
      factor=0.65;              { Συντελεστής συστολής. }
      a1=pi/6;                  { Γωνία στροφής 30 μοιρών. }
      a2=pi/9;                  { Γωνία στροφής 20 μοιρών. }
      a3=pi/3;                  { Γωνία στροφής 60 μοιρών. }
      dinstance=30;             { Αρχικό ευθύγραμμο τμήμα. }
      max_depth=4;              { Μέγιστο βάθος αναδρομής. }

var Gd,Gm,depth:integer;
    point:PointType;            { Το τρέχων σημείο. }
    f,                          { Συντελεστής συστολής. }
    f1,                         { Σταθερά συντελεστού συστολής. }
    a,                          { Γωνία στροφής. }
    d,                          { Ευθύγραμμο τμήμα. }
    angle:real;                 { Τρέχων γωνία. }

(*************************************************************************)
(*                              Draw                                     *)
(*************************************************************************)
{  Ζωγραφίζει ένα ευθύγραμμο τμήμα από το σημείο point στο σημείο pnt.
Η απόσταση του pnt από το point είναι s και το σημείο pnt βρίσκεται σε
μία γωνία angle από το σημείο point. }
procedure draw(var point:PointType;s,angle:real);
var pnt:PointType;
BEGIN
     pnt.x:=round(point.x+s*cos(angle));
     pnt.y:=round(point.y+s*sin(angle));
     line(pnt.x,max_y-pnt.y,point.x,max_y-point.y);
     point:=pnt
END;

(*************************************************************************)
(*                              A_f_t                                    *)
(*************************************************************************)
{ Σχεδιάζει το δέντρο. }
procedure a_f_t(point:PointType;depth:integer;
                f,f1,a,d:real;var angle:real);
var point1,point2,point3:PointType;
BEGIN
 If depth<>0 then
   BEGIN
     angle:=angle+a;                { Στρίψε a ακτίνια αριστερά }
     point1:=point;
     draw(point,d*f,angle);         { Ζωγράφισε ευθύγραμμο τμήμα
                                      μήκους d*f.}
     a_f_t(point,depth-1,f1*f,f1,a,d,angle);

     point:=point1;                 { Επέστρεψε ένα βήμα πίσω. }
```

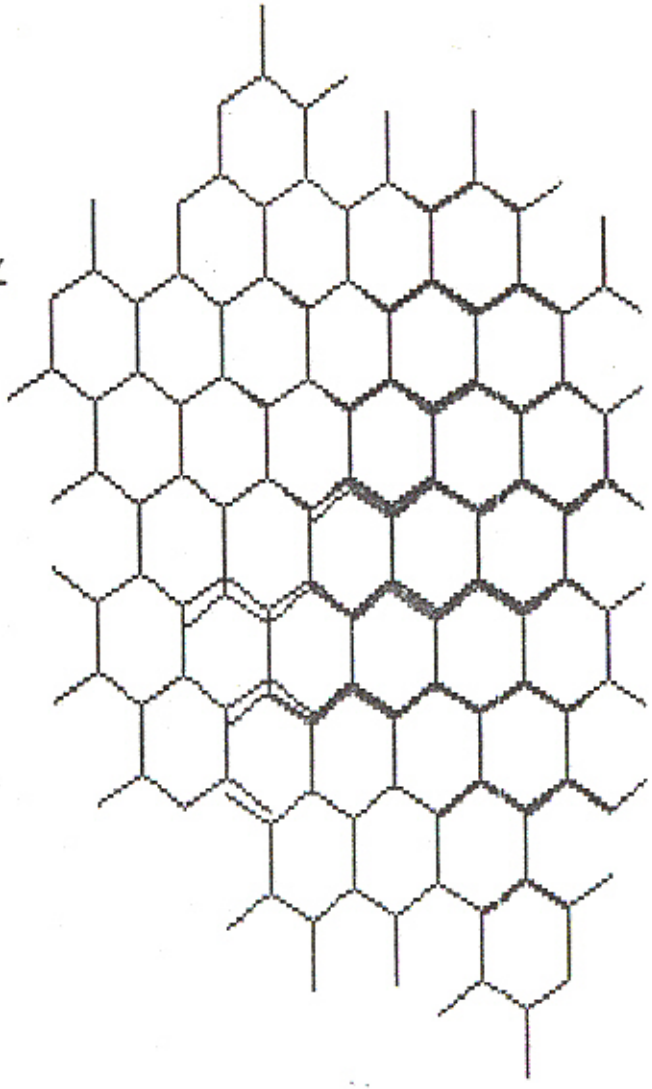
```
    angle:=angle-2*a;                    { Στρίψε 2*a ακτίνια δεξιά. }
    draw(point,d*f,angle);               { Ζωγράφισε ευθύγραμμο τμήμα
                                           μήκους d*f.}

    point2:=point;                       { Στρίψε a ακτίνια δεξιά. }
    angle:=angle-a;                      { Ζωγράφισε ευθύγραμμο τμήμα
    draw(point,d*f,angle);                 μήκους d*f.}

    a_f_t(point,depth-1,f1*f,f1,a,d,angle);

    point:=point2;                       { Επέστρεψε ένα βήμα πίσω. }
    angle:=angle+2*a;                    { Στρίψε 2*a ακτίνια αριστερά. }
    draw(point,d*f,angle);               { Ζωγράφισε ευθύγραμμο τμήμα
                                           μήκους d*f.}

    point3:=point;                       { Στρίψε a ακτίνια αριστερά. }
    angle:=angle+a;                      { Ζωγράφισε ευθύγραμμο τμήμα
    draw(point,d*f,angle);                 μήκους d*f.}

    a_f_t(point,depth-1,f1*f,f1,a,d,angle);

    point:=point3;                       { Επέστρεψε ένα βήμα πίσω. }
    angle:=angle-2*a;                    { Στρίψε 2*a ακτίνια δεξιά. }
    draw(point,d*f,angle);               { Ζωγράφισε ευθύγραμμο τμήμα
                                           μήκους d*f.}

    a_f_t(point,depth-1,f1*f,f1,a,d,angle);

    point:=point2;                       { Στρίψε a ακτίνια αριστερά. }
    angle:=angle+a;                      { Στρίψε a ακτίνια δεξιά. }
    angle:=angle-a;                      { Στρίψε a ακτίνια αριστερά. }
    angle:=angle+a
  END
END;

BEGIN
    Gd:=Detect;                          { Αρχικοποίηση της οθόνης γραφικών. }
    InitGraph(Gd,Gm,'c:\pl\pascal');
    if GraphResult<>grOK then Halt(1);
    ClearDevice;

    f:=1;
    d:=dinstance;
    angle:=pi/2;
    f1:=factor;
    depth:=max_depth;

    point.x:=100;point.y:=80;            { Πρώτο δέντρο με γωνία a1. }
    a:=a1;
    a_f_t(point,depth,f,f1,a,d,angle);

    point.x:=340;point.y:=80;            { Δεύτερο δέντρο με γωνία a2. }
    a:=a2;
    a_f_t(point,depth,f,f1,a,d,angle);
```

```
      f1:=1;
      d:=15;
      point.x:=600;point.y:=100;          { Τρίτο δέντρο με γωνία a3. }
      a:=a3;
      a_f_t(point.depth,f,f1,a,d,angle);

      OutTextXY(295,1,'Fractal Trees');
      repeat
      until keypressed;
      CloseGraph
END.
```

Fractal Trees





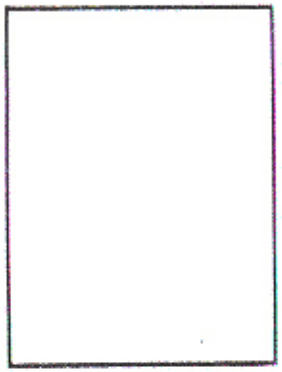


```
program Schnee_Curve;

{ Η fractal αυτή καμπύλη έχει διάσταση D = log s / log t =>
  D = -log 4/ log (1/3) => D = 1.262.Το αρχικό δηλαδή σχήμα
  (ευθύγραμμο τμήμα) αντικαταστήται από ένα εναρκτή με s = 4 το
  πλήθος πλευρές και κάθε πλευρά είναι το 1/3 του αρχικού
  ευθύγραμμου τμήματος. }

USES CRT, GRAPH;

const CharSize=8;
      pi=3.14;
      st_x=50;
      st_y=95;
      trn_x=225;
      trn_y=180;
      max_depth=4;
      dinstance=125;

var Gd,Gm,start_x,start_y,sign:integer;

{*****************************************************************}
{*                          Schnee                               *}
{*****************************************************************}
{ Η procedure αυτή είναι αναδρομική και καλεί το εαυτό της για να
  αντικαταστήσει το τρέχων ευθύγραμμο τμήμα με το εναρκτή μέχρι το
  βάθος της αναδρομής να γίνει μηδέν.Οταν γίνει το βάθος της αναδρομής
  μηδέν τότε ζωγραφίζει ένα ευθύγραμμο τμήμα από το σημείο point στο
  σημείο n_point.Η απόσταση του n_point από το point είναι s και το
  σημείο n_point βρίσκεται σε μια γωνία angle από το σημείο point. }
procedure schnee(var point:PointType;s:real;
                 var angle:real;sign,depth:integer);
var n_point:PointType;
BEGIN
  If depth<>0 then
    begin
      schnee(point,s/3,angle,sign,depth-1);
      angle:=angle+sign*(pi/3);          { Στρίψε π/3 ακτίνια αριστερά }
      schnee(point,s/3,angle,sign,depth-1);
      angle:=angle-sign*2*(pi/3);        { Στρίψε 2*π/3 ακτίνια δεξιά. }
      schnee(point,s/3,angle,sign,depth-1);
      angle:=angle+sign*(pi/3);          { Στρίψε π/3 ακτίνια αριστερά }
      schnee(point,s/3,angle,sign,depth-1)
    end
   else begin
          n_point.x:=round(point.x+s*cos(angle));
          n_point.y:=round(point.y+s*sin(angle));
          line(start_x+n_point.x,start_y+n_point.y
               start_x+point.x,start_y+point.y);
          point:=n_point
        end
END;

{*****************************************************************}
{*                          Choice                               *}
```

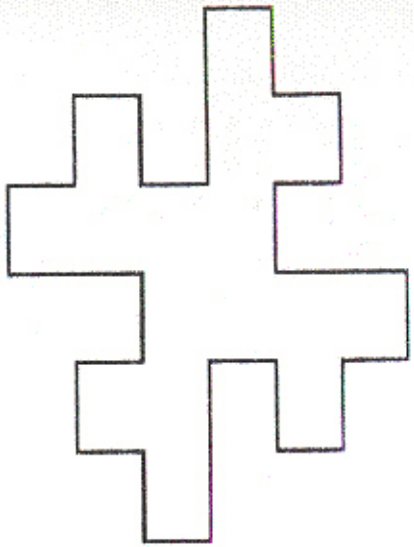
```
(*************************************************************************)
{ Η procedure αυτή που δέχεται σαν όρισμα την μεταβλητή sign  δείχνει
  πρός τα που θα στρίψει η καμπύλη  κατά  θετική  ή  αρνητική  γωνία.
  Δηλαδή εάν θα στρίψει αριστερά ή δεξιά. }
procedure choice(sign:integer);
var depth:integer;              { Τρέχων βάθος της αναδρομής. }
    point:PointType;            { Το τρέχων σημείο. }
    string_1:string[1];
    d,                          { Ευθύγραμμο τμήμα. }
    angle:real;                 { Τρέχων γωνία. }
BEGIN
  ClearDevice;
  start_x:=st_x;
  start_y:=st_y;
  d:=dinstance;
  for depth:=0 to max_depth do
     begin
       if (depth mod 3 = 0) and (depth<>0) then
         BEGIN
           start_x:=st_x;
           start_y:=start_y+trn_y
         END;
       angle:=0.0;
       point.x:=10;point.y:=17;          { Αρχικό σημείο. }
       { Τρείς πλευρές του τριγώνου. }
       schnee(point,d,angle,sign,depth);
       angle:=angle-2*(pi/3);            { Στρίψε 2*π/3 ακτίνια δεξιά. }
       schnee(point,d,angle,sign,depth);
       angle:=angle-2*(pi/3);            { Στρίψε 2*π/3 ακτίνια δεξιά. }
       schnee(point,d,angle,sign,depth);
       case sign of
       1:BEGIN
           OutTextXY(start_x,
                     start_y+7*CharSize,'First Schnee Curve i =');
           Str(depth,string_1);
           OutTextXY(start_x+8*23,start_y+7*CharSize,string_1)
         END;
       -1:BEGIN
            OutTextXY(start_x,
                      start_y+7*CharSize,'Second Schnee Curve i =');
            Str(depth,string_1);
            OutTextXY(start_x+8*24,start_y+7*CharSize,string_1)
          END
       END;
       start_x:=start_x+trn_x
     end;readln
END;

BEGIN
  Gd:=Detect;                       { Αρχικοποίηση της οθόνης γραφικών. }
  InitGraph(Gd,Gm,'c:\pl\pascal');
  if GraphResult<>grOK then Halt(1);

  choice(1);                        { Μία καμπύλη όπου ο παραγωγός στρίβει }
  choice(-1);                       { πρός τα αριτερά και μία όπου στρίβει }
  CloseGraph                        { κατά δεξιά. }
EN
```

```
program First_Koch_Curve;

{ Η fractal αυτή καμπύλη έχει διάσταση D = log a / log f  =>
  D = -log 8/  log (1/4) => D = 3/2.Το αρχικό δηλάδη σχήμα
  (ευθύγραμμο τμήμα) αντικαταστήται από ένα εναρκτή με a το πλήθος
  πλευρές και κάθε πλευρά είναι το 1/4 του αρχικού ευθύγραμμου
  τμήματος. }

USES CRT, GRAPH;

const CharSize=8;
      pi=3.14;
      st_x=50;
      st_y=15;
      trn_x=350;
      trn_y=180;
      max_depth=3;
      dinstance=80;

var Gd,Gm,start_x,start_y,
    depth:integer;              { Τρέχων βάθος της αναδρομής. }
    point:PointType;            { Το τρέχων σημείο. }
    string_1:string[1];
    d,                          { Ευθύγραμμο τμήμα. }
    angle:real;                 { Τρέχων γωνία. }

(*************************************************************)
(*                          Koch                             *)
(*************************************************************)
{ Η procedure αυτή είναι αναδρομική και καλεί το ευατό της για να
  αντικαταστήσει το τρέχων ευθύγραμμο τμήμα με το εναρκτή μέχρι το
  βάθος της αναδρομής να γίνει μηδέν.Οταν γίνει το βάθος της αναδοομής
  μηδέν τότε ζωγραφίζει ένα ευθύγραμμο τμήμα από το σημείο point στο
  σημείο n_point.Η απόσταση του n_point από το point είναι s και το
  σημείο n_point βρίσκεται σε μία γωνία angle από το σημείο point. }
procedure koch(var point:PointType;s:real;
               var angle:real;depth:integer);
var n_point:PointType;
BEGIN
  If depth<>0 then
    begin
      koch(point,s/4,angle,depth-1);
      angle:=angle+(pi/2);              { Στρίψε π/2 ακτίνια αριστερά. }
      koch(point,s/4,angle,depth-1);
      angle:=angle-(pi/2);              { Στρίψε π/2 ακτίνια δεξιά. }
      koch(point,s/4,angle,depth-1);
      angle:=angle-(pi/2);              { Στρίψε π/2 ακτίνια δεξιά. }
      koch(point,s/4,angle,depth-1);
      koch(point,s/4,angle,depth-1);
      angle:=angle+(pi/2);              { Στρίψε π/2 ακτίνια αριστερά. }
      koch(point,s/4,angle,depth-1);
      angle:=angle+(pi/2);              { Στρίψε π/2 ακτίνια αριστερά. }
      koch(point,s/4,angle,depth-1);
      angle:=angle-(pi/2);              { Στρίψε π/2 ακτίνια δεξιά. }
      koch(point,s/4,angle,depth-1)
```

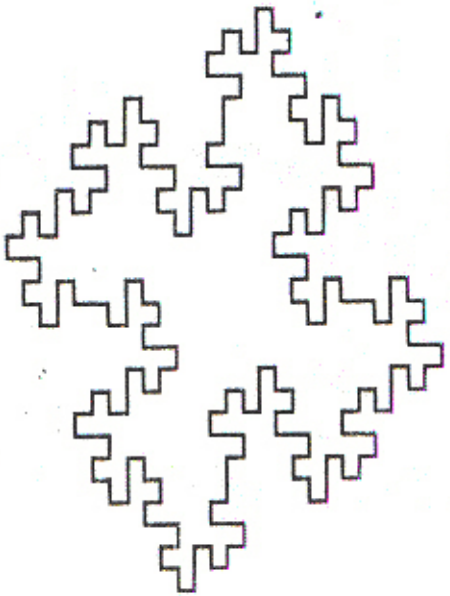
```
    end
  else begin
         n_point.x:=round(point.x+s*cos(angle));
         n_point.y:=round(point.y+s*sin(angle));
         line(start_x+n_point.x,start_y+n_point.y,
              start_x+point.x,start_y+point.y);
         point:=n_point
       end
END;

BEGIN
  Gd:=Detect;                          { Αρχικοποίηση της οθόνης γραφικών. }
  InitGraph(Gd,Gm,'c:\pl\pascal');
  if GraphResult<>grOK then Halt(1);

  start_x:=st_x;
  start_y:=st_y;
  angle:=0.0;
  d:=dinstance;
  for depth:=0 to max_depth do
     begin
       if (depth mod 2 = 0) and (depth<>0) then
         BEGIN
           start_x:=st_x;
           start_y:=start_y+trn_y
         END;
       point.x:=10;point.y:=10;          { Αρχικό σημείο. }
       koch(point,d,angle,depth);
       angle:=angle+(pi/2);              { Στρίψε π/2 ακτίνια αριστερά. }
       koch(point,d,angle,depth);
       angle:=angle+(pi/2);              { Στρίψε π/2 ακτίνια αριστερά. }
       koch(point,d,angle,depth);
       angle:=angle+(pi/2);              { Στρίψε π/2 ακτίνια αριστερά. }
       koch(point,d,angle,depth);
       angle:=angle+(pi/2);              { Στρίψε π/2 ακτίνια αριστερά. }
       OutTextXY(start_x-5*CharSize,
                 start_y+15*CharSize,'First Koch Curve for i =');
       Str(depth,string_1);
       OutTextXY(start_x+8*20,start_y+15*CharSize,string_1);
       start_x:=start_x+trn_x
     end;readln;
  CloseGraph
EN
```



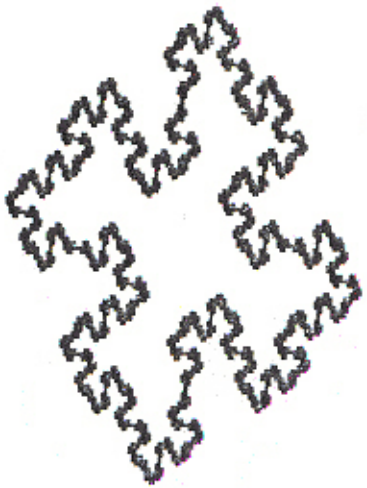
First Koch Curve for i = 0



First Koch Curve for i = 1



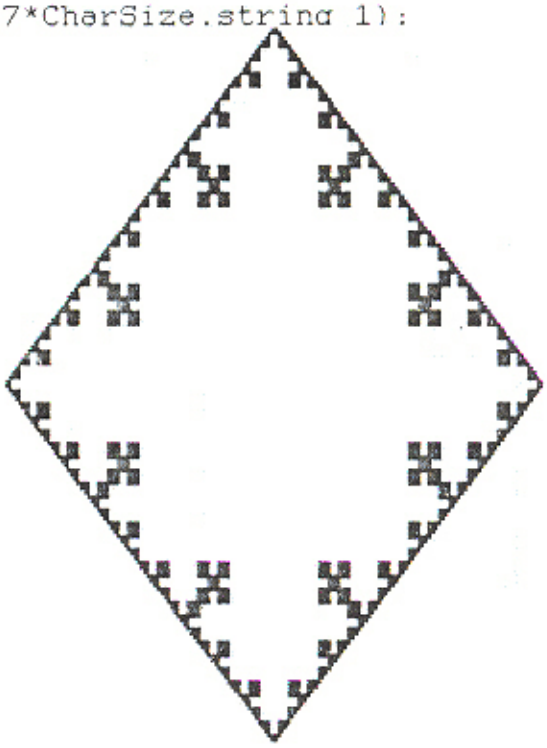
First Koch Curve for i = 2



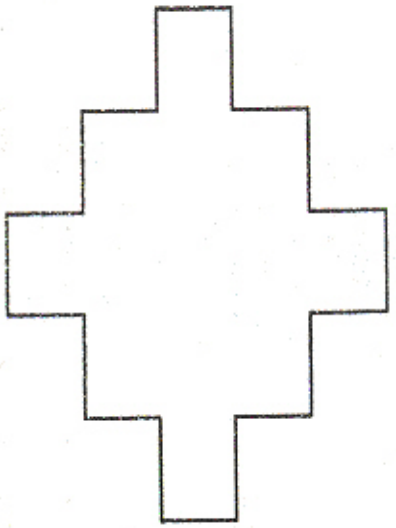
First Koch Curve for i = 3

```
program Second_Koch_Curve;

{ Η fractal αυτή καμπύλη έχει διάσταση D = log a / log f  =>
  D = -log 5/  log (1/3) => D = 1.465.Το αρχικό δηλάδη σχήμα
  (ευθύγραμμο τμήμα) αντικαταστήται από ένα εναρκτή με a = 5 το
  πλήθος πλευρές και κάθε πλευρά είναι το 1/3 του αρχικού
  ευθύγραμμου τμήματος. }

USES CRT, GRAPH;

const CharSize=8;
      pi=3.14;
      max_y=300;
      st_x=50;
      st_y=30;
      trn_x=350;
      trn_y=170;
      max_depth=4;
      dinstance=70;

var Gd,Gm,start_x,start_y,
    depth:integer;              { Τρέχων βάθος της αναδρομής. }
    point:PointType;            { Το τρέχων σημείο. }
    string_1:string[1];
    d,                          { Ευθύγραμμο τμήμα. }
    angle:real;                 { Τρέχων γωνία. }

{**************************************************************}
{*                          Koch                              *}
{**************************************************************}
{ Η procedure αυτή είναι αναδρομική και καλεί το ευατό της για να
  αντικαταστήσει το τρέχων ευθύγραμμο τμήμα με το εναρκτή μέχρι το
  βάθος της αναδρομής να γίνει μηδέν.Οταν γίνει το βάθος της αναδρομής
  μηδέν τότε ζωγραφίζει ένα ευθύγραμμο τμήμα από το σημείο point στο
  σημείο n_point.Η απόσταση του n_point από το point είναι s και το
  σημείο n_point βρίσκεται σε μία γωνία angle από το σημείο point. }
procedure koch(var point:PointType;s,angle:real;depth:integer);
var n_point:PointType;
BEGIN
  If depth<>0 then
    begin
      koch(point,s/3,angle,depth-1);
      angle:=angle-(pi/2);              { Στρίψε π/2 ακτίνια δεξιά. }
      koch(point,s/3,angle,depth-1);
      angle:=angle+(pi/2);              { Στρίψε π/2 ακτίνια αριστερά. }
      koch(point,s/3,angle,depth-1);
      angle:=angle+(pi/2);              { Στρίψε π/2 ακτίνια αριστερά. }
      koch(point,s/3,angle,depth-1);
      angle:=angle-(pi/2);              { Στρίψε π/2 ακτίνια δεξιά. }
      koch(point,s/3,angle,depth-1)
    end
   else begin
          n_point.x:=round(point.x+s*cos(angle));
          n_point.y:=round(point.y+s*sin(angle));
          line(start_x+n_point.x,start_y+n_point.y,
```

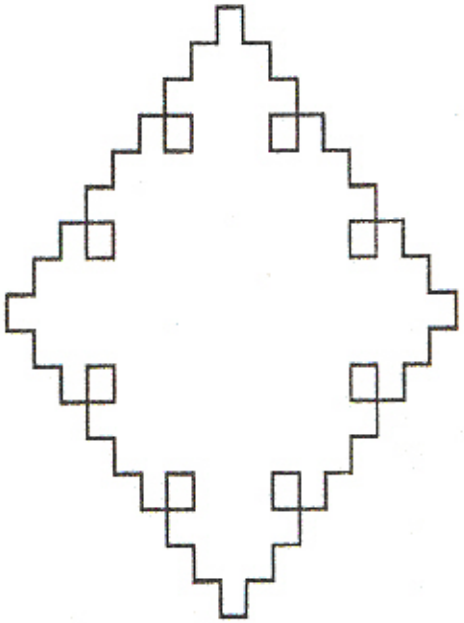
```
          line(start_x+n_point.x,start_y+n_point.y,
               start_x+point.x,start_y+point.y);
          point:=n_point
        end
END;

BEGIN
  Gd:=Detect;                    { Αρχικοποίηση της οθόνης γραφικών. }
  InitGraph(Gd,Gm,'c:\pl\pascal');
  if GraphResult<>grOK then Halt(1);

  start_x:=st_x;
  start_y:=st_y;
  angle:=0.0;
  d:=dinstance;
  for depth:=0 to max_depth do
     begin
       if (depth mod 2 = 0) and (depth<>0) then
         BEGIN
           start_x:=st_x;
           start_y:=start_y+trn_y
         END;
       if depth = 4 then
         BEGIN
           readln;
           start_y:=st_y;
           ClearDevice
         END;
       point.x:=25;point.y:=10;    { Αρχικό σημείο. }
       koch(point,d,angle,depth);
       angle:=angle+(pi/2);          { Στρίψε π/2 ακτίνια αριστερά. }
       koch(point,d,angle,depth);
       angle:=angle+(pi/2);          { Στρίψε π/2 ακτίνια αριστερά. }
       koch(point,d,angle,depth);
       angle:=angle+(pi/2);          { Στρίψε π/2 ακτίνια αριστερά. }
       koch(point,d,angle,depth);
       angle:=angle+(pi/2);          { Στρίψε π/2 ακτίνια αριστερά. }
       OutTextXY(start_x-3*CharSize,
                 start_y+17*CharSize,'Second Koch Curve for i =');
       Str(depth,string_1);
       OutTextXY(start_x+8*23,start_y+17*CharSize,string_1);
       start_x:=start_x+trn_x
     end;readln;
  CloseGraph
EN
```



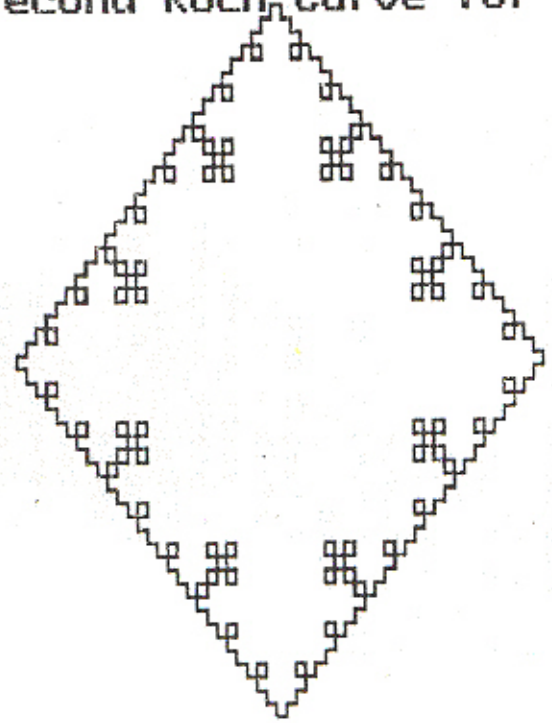
Second Koch Curve for i = 0



Second Koch Curve for i = 1



Second Koch Curve for i = 2

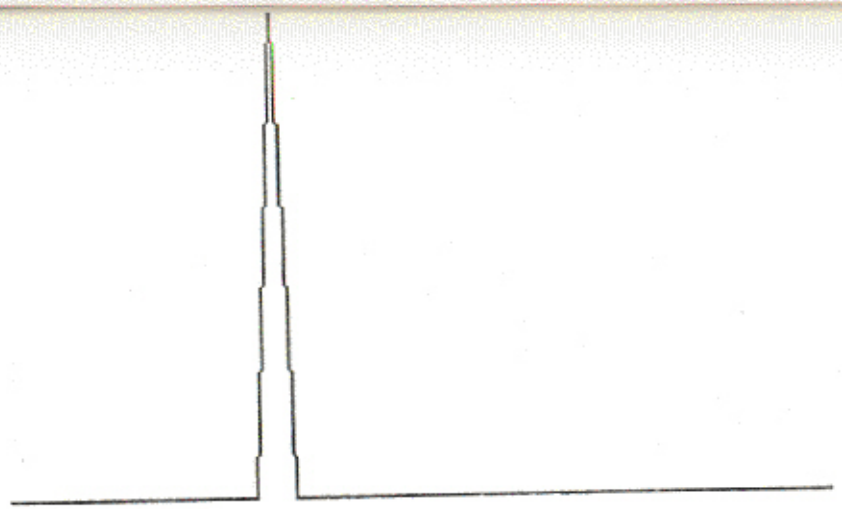


Second Koch Curve for i = 3

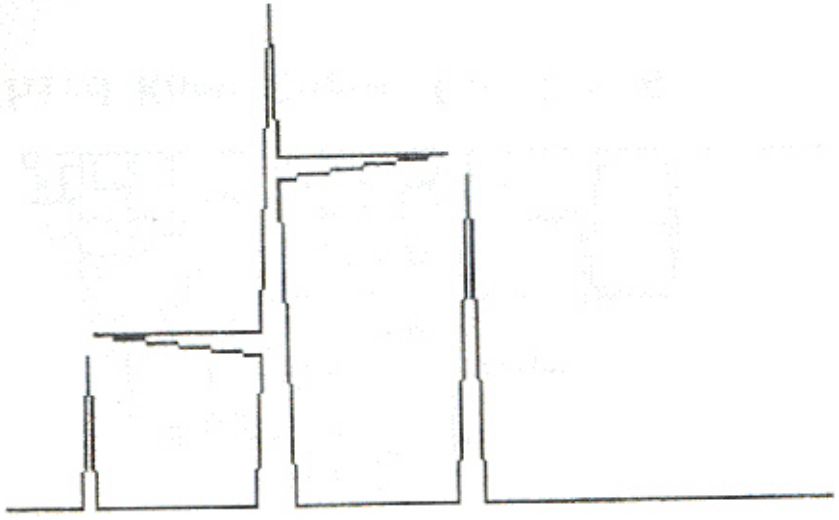
```
program Third_Koch_Curve;

USES CRT, GRAPH;

const CharSize=8;
      pi=3.14;
      max_y=100;
      st_x=10;
      st_y=35;
      trn_x=350;
      trn_y=180;
      max_depth=5;
      dinstance=250;

var Gd,Gm,start_x,start_y,
    depth:integer;              { Τρέχων βάθος της αναδρομής. }
    point:PointType;            { Το τρέχων σημείο. }
    string_1:string[1];
    d,                          { Ευθύγραμμο τμήμα. }
    angle:real;                 { Τρέχων γωνία. }


(*************************************************************************)
(*                              Koch                                     *)
(*************************************************************************)
{ Η procedure αυτή είναι αναδρομική και καλεί το ευατό της για να
  αντικαταστήσει το τρέχων ευθύγραμμο τμήμα με το εναρκτή μέχρι το
  βάθος της αναδρομής να γίνει μηδέν.Οταν γίνει το βάθος της αναδρομής
  μηδέν τότε ζωγραφίζει ένα ευθύγραμμο τμήμα από το σημείο point στο
  σημείο n_point.Η απόσταση του n_point από το point είναι s και το
  σημείο n_point βρίσκεται σε μία γωνία angle από το σημείο point. }
procedure koch(var point:PointType;s:real;
               var angle:real;depth:integer);
var n_point:PointType;
BEGIN
  If depth<>0 then
    begin
      koch(point,6*(s/19.942),angle,depth-1);
      angle:=angle+87*(pi/180);         { Στρίψε 87 μοίρες αριστερά. }
      koch(point,9*(s/19.942),angle,depth-1);
      angle:=angle-174*(pi/180);        { Στρίψε 174 μοίρες δεξιά. }
      koch(point,9*(s/19.942),angle,depth-1);
      angle:=angle+87*(pi/180);         { Στρίψε 87 μοίρες αριστερά. }
      koch(point,13*(s/19.942),angle,depth-1)
    end
   else begin
          n_point.x:=round(point.x+s*cos(angle));
          n_point.y:=round(point.y+s*sin(angle));
          line(start_x+n_point.x,start_y+max_y-n_point.y,
               start_x+point.x,start_y+max_y-point.y);
          point:=n_point
        end
END;

BEGIN
```

```
  Gd:=Detect;                          { Αρχικοποίηση της οθόνης γραφικών. }
  InitGraph(Gd,Gm,'c:\pl\pascal');
  if GraphResult<>grOK then Halt(1);

  start_x:=st_x;
  start_y:=st_y;
  angle:=0.0;
  d:=dinstance;
  for depth:=0 to max_depth do
     begin
       if (depth mod 2 = 0) and (depth<>0) then
         BEGIN
           start_x:=st_x;
           start_y:=start_y+trn_y
         END;
       if depth = 4 then
         BEGIN
           readln;
           start_y:=st_y;
           ClearDevice
         END;
       point.x:=10;point.y:=15;     { Αρχικό σημείο. }
       koch(point,d,angle,depth);
       OutTextXY(start_x,
                 start_y+12*CharSize,'Third Koch Curve for i =');
       Str(depth,string_1);
       OutTextXY(start_x+8*25,start_y+12*CharSize,string_1);
       start_x:=start_x+trn_x
     end;readln;
  CloseGraph
EN
```

Third Koch Curve for i = 0



Third Koch Curve for i = 1



Third Koch Curve for i = 2

Third Koch Curve for i = 3