```
> restart:
  with(Statistics):



  delta:=evalf(0.1):


  #          ,          0 . 1   0 . 0 5
  0.01  0.005
  ChZnPoZap:=4:
  N:=10^ChZnPoZap:
  ShagT:=evalf[1](1/10^ChZnPoZap);

  T[0]:=0:
  for k from 1 to N do
            T[k] := T[k-1]+ShagT;
  end do:

  TaTau:=proc(t,n)
            add(dig(t,k)/2^(k-2*n),k=2*n+2..30,2):
  end proc:

  g:=proc(a,b)
  (-1)^a*(a-b)^2:
  end proc:


  dig:= proc (t, n)
            local s,p,i:
            s:=t:
            for i from 1 to n do
                      p:=trunc(s*2):
                      s:=s*2-p:
            end do:
            p;
  end proc:


  Koch:=proc(n,t)
            local c, alpha;
            alpha:=evalf(Pi/3):
            c:=1/(2+2*cos(alpha)):
            c^n*TaTau(t,n)*exp(I*alpha*add(g(dig(t,2*k-1),dig(t,2*
  k)),k=1..n))+add(c^(j-1)*(dig(t,2*j-1)*(dig(t,2*j-1)+dig(t,2*j))
  /2+(-1)^dig(t,2*j-1)*dig(t,2*j)*c+dig(t,2*j-1)*(1-dig(t,2*j))*c*
  sin(alpha)*I)*exp(I*alpha*add(g(dig(t,2*k-1),dig(t,2*k)),k=1..
  j-1)),j=1..n):
  end proc:



  SaveFractalNotWar:=[[0,0,0],seq([evalc(Re(Koch(5,T[k]))),evalc(Im
  (Koch(5,T[k]))),T[k]],k=1..N-1),[1,0,1]]:
  ChisloUzlovFractala:=nops(SaveFractalNotWar)-1:
```

```
KoxaFunction:= proc(t)
          option remember;
          local i:
          global ChisloUzlovFractala, SaveFractalNotWar:
          for i from 0 to ChisloUzlovFractala do
                    if t=i/ChisloUzlovFractala then
                    RETURN([SaveFractalNotWar[i+1,1],
SaveFractalNotWar[i+1,2]]);
                    fi:
          od:
NULL:
end proc:
```

$$ShagT := 0.0001 \quad (1)$$

```
> dig(0.1,4)
```

$$1 \quad (2)$$

```
> #

SubdivizionTrue[1]:=0:
i:=1:

#
while SubdivizionTrue[i]<1 do
          #
          SubdivizionTrue[i+1]:=evalf[2](SubdivizionTrue[i]+delta/4):
          i:=i+1:
od:
N:=i:print(%):



k:=0:
while k<300 do
          i:='i':
          j:='j':
          SubdivizionShtrih1:=0:
          #
          #
          x1:=evalf(Sample(RandomVariable(Uniform(0, 1)), 1)[1]):
          y1:=evalf(Sample(RandomVariable(Uniform(0, 1)), 1)[1]):

          #
          while evalf[5](x1)=evalf[5](y1) do
                    y1:=evalf(Sample(RandomVariable(Uniform(0, 1)
), 1)[1]):
          od:

          y:=max(x1,y1); x:=min(x1,y1);
          #
          j:=1:
          for i from 1 to N do
                    if SubdivizionTrue[i]>=x and SubdivizionTrue
[i]<=y then
                              SubdivizionShtrih1[j]:=
SubdivizionTrue[i]:
```

```
                                j:=j+1:
                                #

                                if j=2 then
                                        iStart:=i:
                                fi:

                        fi:
                od:
                #   ,
                m:=j-3:
                #

                iFinish:=iStart+m+1:

                keyOnlyOneWay:=0:

                #             3

                if m>1 then
                #-----------------------------------------------------
- - - - - - - - - - - - - - -
                        RandomVariableForProbability:=Sample
(RandomVariable(Uniform(0, 1)), 1)[1]:
                        pC:=min(1,delta/(y-x));
                        #     ,

                        if RandomVariableForProbability<evalf(pC) and
keyOnlyOneWay=0 then
                                #       (
)
                                key:=-1:
                                while key<>SubdivizionShtrih1[m+2] do
                                        SubdivizionShtrih1Shift:=
0:
                                        SubdivizionShtrih1Shift
[1]:=SubdivizionShtrih1[1]:
                                        RandomVariableForShift:=
Sample(RandomVariable(Uniform(-delta/10, delta/10)), m+5):
                                        for i from 2 to m+1 do

SubdivizionShtrih1Shift[i]:=abs(evalf[5](trunc(10^ChZnPoZap*
(SubdivizionShtrih1[i]+
                                RandomVariableForShift[i]))
/10^ChZnPoZap)):
                                        od:
                                        SubdivizionShtrih1Shift
[m+2]:=SubdivizionShtrih1[m+2]:
                                        SubdivizionShtrih1Shift:=
sort([seq(SubdivizionShtrih1Shift[pp],pp=1..m+2)]);
                                        key:=
SubdivizionShtrih1Shift[m+2]:
                                od:
                                sssss:=0:
                                #
          ,
                                diamShtrih1Shift:=evalf(max(seq
```

```
(SubdivizionShtrih1Shift[ii]-SubdivizionShtrih1Shift[ii-1],ii=2..
m+2))):                          if diamShtrih1Shift<=delta then
                                          for i from 1 to m+2 do
                                          SubdivizionShtrih2[i]:=
SubdivizionShtrih1Shift[i]:
                                          od:

                                 #

                                          for i from 1 to iStart do
                                                    Subdivizion1[i]
:=SubdivizionTrue[i]:
                                          od:
                                          j:=2:
                                          for i from iStart+1 to
iFinish-1 do
                                                    Subdivizion1[i]
:=SubdivizionShtrih2[j]:
                                                    j:=j+1:
                                          od:

                                          for i from iFinish to N
do
                                                    Subdivizion1[i]
:=SubdivizionTrue[i]:
                                          od:

                                          #

                                          sigma1:=add(evalf((sqrt(
(KoxaFunction(Subdivizion1[jj+1])[1]-KoxaFunction(Subdivizion1
[jj])[1])^2+
(KoxaFunction(Subdivizion1[jj+1])[2]-KoxaFunction(Subdivizion1
[jj])[2])^2))^(ln(4)/ln(3))),jj=1..
          N-1):
                                          sigma:=add(evalf((sqrt(
(KoxaFunction(SubdivizionTrue[jj+1])[1]-KoxaFunction
(SubdivizionTrue[jj])
[1])^2+(KoxaFunction(SubdivizionTrue[jj+1])[2]-KoxaFunction
(SubdivizionTrue[jj])[2])^2))^(ln(4)/ln
          (3))),jj=1..N-1):

                                 #

  k
                                          if sigma1<sigma then
                                                    k:=k+1:

                                                    for i from 1 to
N do

SubdivizionTrue[i]:=Subdivizion1[i]:
                                                    od:
                                                    p r i n t ( "
      " ) :
                                                    print(sigma1):
                                                    p r i n t ( "
```

```
" ) :
                                                print(k):
                                                keyOnlyOneWay:=0:
                                        fi:
                                        keyOnlyOneWay:=0:
                                fi:
                        fi:

                        #----------------------------------------
------------------------------ B
                        #

                        RandomVariableForProbability:=Sample
(RandomVariable(Uniform(0, 1)), 1)[1]:
                        pD:=min(1,delta/(y-x));
                        #     ,

                        if RandomVariableForProbability<evalf(pD) and
keyOnlyOneWay=0 then
                                #       (
)
                                SubdivizionShtrih1Delete[1]:=
SubdivizionShtrih1[1]:
                                SubdivizionShtrih1Delete[2]:=
SubdivizionShtrih1[m+2]:

                                #
            ,

                                diamShtrih1Delete:=evalf
(SubdivizionShtrih1Delete[2]-SubdivizionShtrih1Delete[1]):
                                if diamShtrih1Delete<=delta then
                                        SubdivizionShtrih2[1]:=
SubdivizionShtrih1Delete[1]:
                                        SubdivizionShtrih2[2]:=
SubdivizionShtrih1Delete[2]:


                                        #

                                        for i from 1 to iStart-1
do
                                                Subdivizion1[i]
:=SubdivizionTrue[i]:
                                        od:

                                        Subdivizion1[iStart]:=
SubdivizionShtrih2[1]:
                                        Subdivizion1[iStart+1]:=
SubdivizionShtrih2[2]:


                                        jjj:=iStart+2:
                                        for i from iFinish+1 to N do
                                                Subdivizion1
[jjj]:=SubdivizionTrue[i]:

                                                jjj:=jjj+1:
                                        od:
```

```
                                        jjj:=jjj-1:



                                        #

                                        sigma1:=add(evalf((sqrt(
(KoxaFunction(Subdivizion1[jj+1])[1]-KoxaFunction(Subdivizion1
[jj])[1])^2+
(KoxaFunction(Subdivizion1[jj+1])[2]-KoxaFunction(Subdivizion1
[jj])[2])^2))^(ln(4)/ln(3))),jj=1..
          jjj-1):
                                        sigma:=add(evalf((sqrt(
(KoxaFunction(SubdivizionTrue[jj+1])[1]-KoxaFunction
(SubdivizionTrue[jj])
[1])^2+(KoxaFunction(SubdivizionTrue[jj+1])[2]-KoxaFunction
(SubdivizionTrue[jj])[2])^2))^(ln(4)/ln
          (3))),jj=1..N-1):
                                        #sigma1:=1:
                                        #

    k
                                        if sigma1<sigma then
                                                k:=k+1:
                                                SubdivizionTrue:=0:
                                                for i from 1 to
jjj do

SubdivizionTrue[i]:=Subdivizion1[i]: #print(%);
                                                od:#1 to j
                                                print("
      " ) :
                                                print(sigma1):
                                                N:=jjj: print(N):
                                                print("
" ) :
                                                print(k):
                                                keyOnlyOneWay:=0:
                                        fi:#sigma1<sigma
                                fi: #diamShtrih1Delete<=delta

                                keyOnlyOneWay:=0:
                        fi:#RandomVariableForProbability<evalf(pD)

                        #----------------------------------------
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -   (
                                )
                        #

                        RandomVariableForProbability:=Sample
(RandomVariable(Uniform(0, 1)), 1)[1]:
                        #RandomVariableForProbability:=0:
                        pI:=min(1,delta/(y-x));
                        #     ,

                        if RandomVariableForProbability<evalf(pI) and
keyOnlyOneWay=0 then
```

```
                                #
                                i:=1:
                                j:=1:
                                while j<=m do
                                        if SubdivizionShtrih1
[j+1]-SubdivizionShtrih1[j]>delta/10 then

SubdivizionShtrih1Insert[i]:=SubdivizionShtrih1[j]:
                                                i:=i+1:

SubdivizionShtrih1Insert[i]:=SubdivizionShtrih1Insert[i-1]:
                                                while abs
(SubdivizionShtrih1Insert[i]-SubdivizionShtrih1Insert[i-1])=0 do

SubdivizionShtrih1Insert[i]:=evalf[5](trunc(10^ChZnPoZap*(Sample
(RandomVariable(Uniform
                        (SubdivizionShtrih1[j], SubdivizionShtrih1
[j+1])), 1)[1]))/10^ChZnPoZap):
                                                od:
                                                i:=i+1:
                                        else

SubdivizionShtrih1Insert[i]:=SubdivizionShtrih1[j]:
                                                i:=i+1:
                                        fi:
                                        j:=j+1:
                                od:
                                SubdivizionShtrih1Insert[i]:=
SubdivizionShtrih1[j]:
                                NN:=i;

                                #
          ,
                                diamShtrih1Insert:=evalf(max(seq
(SubdivizionShtrih1Insert[ii]-SubdivizionShtrih1Insert[ii-1],ii=
2..NN))):
                                if diamShtrih1Insert<=delta then
                                        for i from 1 to NN do

SubdivizionShtrih2[i]:=SubdivizionShtrih1Insert[i]:
                                        end do:

                                        #

                                        for i from 1 to iStart do
                                                Subdivizion1[i]
:=SubdivizionTrue[i]:
                                        od:
                                        j:=iStart:
                                        for i from 1 to NN do
                                                Subdivizion1[j]
:=SubdivizionShtrih2[i]:
                                                j:=j+1:
                                        end do:

                                        for i from iFinish to N
do
```

```
                                                    Subdivizion1[j]
:=SubdivizionTrue[i]:
                                                    j:=j+1:
                                               od:
                                               j:=j-1:

                                               #

                                               sigma1:=add(evalf((sqrt(
(KoxaFunction(Subdivizion1[jj+1])[1]-KoxaFunction(Subdivizion1
[jj])[1])^2+
(KoxaFunction(Subdivizion1[jj+1])[2]-KoxaFunction(Subdivizion1
[jj])[2])^2))^(ln(4)/ln(3))),jj=1..
          j-1):
                                               sigma:=add(evalf((sqrt(
(KoxaFunction(SubdivizionTrue[jj+1])[1]-KoxaFunction
(SubdivizionTrue[jj])
[1])^2+(KoxaFunction(SubdivizionTrue[jj+1])[2]-KoxaFunction
(SubdivizionTrue[jj])[2])^2))^(ln(4)/ln
          (3))),jj=1..N-1):

                                               #

    k
                                               if sigma1<sigma then
                                                    k:=k+1:
                                                    SubdivizionTrue:=0:
                                                    for i from 1 to
j do

SubdivizionTrue[i]:=Subdivizion1[i]: #print(%);
                                                    od:#1 to j
                                                    p r i n t ( "
     C " ) :
                                                    print(sigma1):
                                                    N:=j:print(N):
                                                    p r i n t ( "
" ) :
                                                    print(k):
                                                    keyOnlyOneWay:=0:
                                               fi:#sigma1<sigma

                                               keyOnlyOneWay:=1:
                                          fi: #diamShtrih1Insert<=delta
                                fi:#RandomVariableForProbability<evalf(pI)
          fi:#m>3
          keyOnlyOneWay:=0:

          if SubdivizionTrue[N]<0.98 then
                    p r i n t ( " ! " ,   SubdivizionTrue[N]):
                    k:=100:
          fi:

od:# k<...
```



```
               "      "
```

0.769855538847

" "

1

" C "

0.763146598250

" "

2

" C "

0.760270989759

" "

3

" "

0.760154350

" "

4

" C "

0.757916065166

" "

5

" "

0.753971319

" "

6

" C "

0.751648493274

" "

7

" "

0.745082108

" "

8

" C "

0.740241077885

" "

9

" "

0.7344752786

" "

10

" "

0.7319659571

82

" "

11

" "

0.7252036911

75

" "

12

" C "

0.7251146765

78

" "

13

" C "

0.7236921579

83

" "

14

" "

0.7232492375

" "

15

" "

0.7143526497

80

" "

16

" "

0.7119108869

" "

17

" "

0.7074739943

" "

18

" "

0.7028217839

" "
19
" "
0.7027657921
" "
20
" "
0.7027208586
78
" "
21
" C "
0.7016526046
79
" "
22
" "
0.7001955976
77
" "
23
" C "
0.6985251018
78
" "
24
" "
0.6976328603
" "
25
" "
0.6907304380
71
" "
26
" C "
0.6904882022
72
" "
27
" "
0.6877887846

" "
28
" "
0.6855077920
" "
29
" C "
0.6832475446
74
" "
30
" "
0.6831195556
" "
31
" "
0.6803698550
" "
32
" C "
0.6706868935
82
" "
33
" C "
0.6702581083
83
" "
34
" "
0.6697474519
" "
35
" C "
0.6663714750
86
" "
36
" "
0.6555989317
79
" "

37
" C "
0.6532367075
86
" "
38
" "
0.6528343689
" "
39
" C "
0.6513555976
87
" "
40
" C "
0.6513449276
88
" "
41
" "
0.6436963398
86
" "
42
" C "
0.6164953577
88
" "
43
"попался!", 0.96
"попался!", 0.96
"попался!", 0.96
"попался!", 0.96
"попался!", 0.96
"попался!", 0.96
"попался!", 0.96
"попался!", 0.96
"попался!", 0.96
"попался!", 0.96
"попался!", 0.96
"попался!", 0.96

"попался!", 0.96
"попался!", 0.96
" C "
0.6159782464
90
" "
101
"попался!", 0.96
"попался!", 0.96
"попался!", 0.96
"попался!", 0.96
"попался!", 0.96
"попался!", 0.96
"попался!", 0.96
"попался!", 0.96
"попался!", 0.96
"попался!", 0.96
" "
0.6141345060
" "
101
"попался!", 0.96
"попался!", 0.96
"попался!", 0.96
"попался!", 0.96
"попался!", 0.96
"попался!", 0.96
"попался!", 0.96
"попался!", 0.96
"попался!", 0.96
"попался!", 0.96
"попался!", 0.96
"попался!", 0.96
"попался!", 0.96
"попался!", 0.96
"попался!", 0.96
"попался!", 0.96
"попался!", 0.96
"попался!", 0.96
" C "
0.6118448190
92
" "

101

"попался!", 0.96

"попался!", 0.96

"попался!", 0.96

"попался!", 0.96

"попался!", 0.96

"попался!", 0.96

"попался!", 0.96

" "

0.6098340018

77

" "

101

"попался!", 0.96

"попался!", 0.96

" "

0.6094705458

" "

101

"попался!", 0.96

"попался!", 0.96

"попался!", 0.96

"попался!", 0.96

"попался!", 0.96

"попался!", 0.96

"попался!", 0.96

" C "

0.6077324467

78

" "

101

"попался!", 0.96

" "

0.6076359184

" "

101

" "

0.6056502501

Warning, computation interrupted

``

$EV(rg)$

$EV(rg)$

Warning, computation interrupted

```
> ChisloUzlovFractala
```

999 **(3)**

```
>
```