7.

Logo ni samo za otroke

Do sedaj smo videli, da lahko v logu počnemo celo vrsto stvari in ne samo podimo želvico po zaslonu. V tem razdelku bomo spoznali še nekaj ukazov, ki logo bistveno ločijo od programskih jezikov (pascal, C ...), katerih izvajanje temelji na prevajanju.

7.1 Podatki ukazi

Kot vemo, za izvajanje ukazov v logu skrbi tolmač (interpreter). V takih programskih jezikih meja med podatki in ukazi ni ostra. V logu omogočajo prehode med podatki in ukazi naslednji ukazi:

`RUN` *s*	izvede ukaze iz seznama *s*
`TEXT` *p*	vrne seznam z opisom ukaza *p*
`DEFINE` *p s*	vpelje ukaz z imenom *p* in opisom *s*

Poleg tega lahko seznam, kot vrednost izraza, podtaknemo tudi krmilnim stavkom `REPEAT` in `IF`.

Ker narečje LogoS ne upošteva vrstic v zapisu ukaza, seznam z opisom ukaza, ki ga vrne ukaz `TEXT`, v LCSI/LogoSu zgleda takole:

`[ [` *parametri* `] [` *ukazi* `] ]`

v ostalih narečjih loga pa takole:

`[ [` *parametri* `] [` *seznami_vrstice* `] ]`

Tako, na primer, za ukaz

```
TO ZNAK :x :y :z
  SETCUR (LIST :x :y)
  TYPE :z
END
```

na zahtevo

```
SHOW TEXT "ZNAK
```

LCSI/LogoS izpiše

```
[ [ :x :y :z ] [ SETCUR ( LIST :x :y ) TYPE :z ] ]
```

v drugih narečjih loga pa dobimo

```
[ [ :x :y :z ] [ [ SETCUR ( LIST :x :y ) ] [ TYPE :z ] ] ]
```

V nadaljevanju si bomo najprej podrobneje ogledali ukaz `RUN`.

Začnimo s preprostim primerom

```
MAKE "u [3 * 4 + 5] SHOW :u SHOW RUN :u
```

Logo izpiše `[ 3 * 4 + 5 ]` in 17.

Ukaz `RUN` omogoča *razširljivost* jezika logo – sestavljamo lahko lastne krmilne ukaze. Za primer sestavimo ukaz `ZAVSAK` e s u , ki vsak člen seznama s postavi v spremenljivko e in nad njo izvede ukaze u

```
TO ZAVSAK :e :s :u
  REPEAT COUNT :s
    [ MAKE :e FIRST :s MAKE "s BF :s RUN :u ]
END
```

Postavimo

```
MAKE "m [Maribor Celje Ljubljana Kranj Koper]
MAKE "a [0 -1 2 -3 4]
```

Tedaj zahtevi

```
ZAVSAK "c :m [PR :c]
ZAVSAK "c :m [PR FIRST :c]
```

izpišeta imena krajev in nato še začetnice imen krajev. Zahteva

```
ZAVSAK "x :a [PR SIN 30*:x]
```

pa izpiše sinuse kotov 0, -30, 60, 90 in 120 stopinj. Poskusimo še

```
ZAVSAK "x :a [PR SQRT :x]
```

Pri drugem členu logo prekine izvajanje s sporočilom, da funkcija kvadratni koren ni definirana za -1. No, logo omogoča nadzor tudi nad takimi primeri. Za to je namenjen ukaz `CATCH "`*oznaka ukazi* , ki prestreže oznako `"`*oznaka*, ki jo "vrže" z ukazom `THROW "`*oznaka* kakšen od stavkov iz seznama *ukazi*. Sam logo ob napaki vrže oznako `"ERROR` . Z ukazom `CATCH` lahko sestavimo nov funkcijski ukaz `KOREN`, ki vrne funkcijsko vrednost, kadar je ta definirana, sicer pa vrne besedo `Napaka`.

```
TO KOREN :t
  CATCH "ERROR [ OP SQRT :t ]
  OP "Napaka
END
```

Logo bo sedaj zahtevo

```
ZAVSAK "x :a [PR KOREN :x]
```

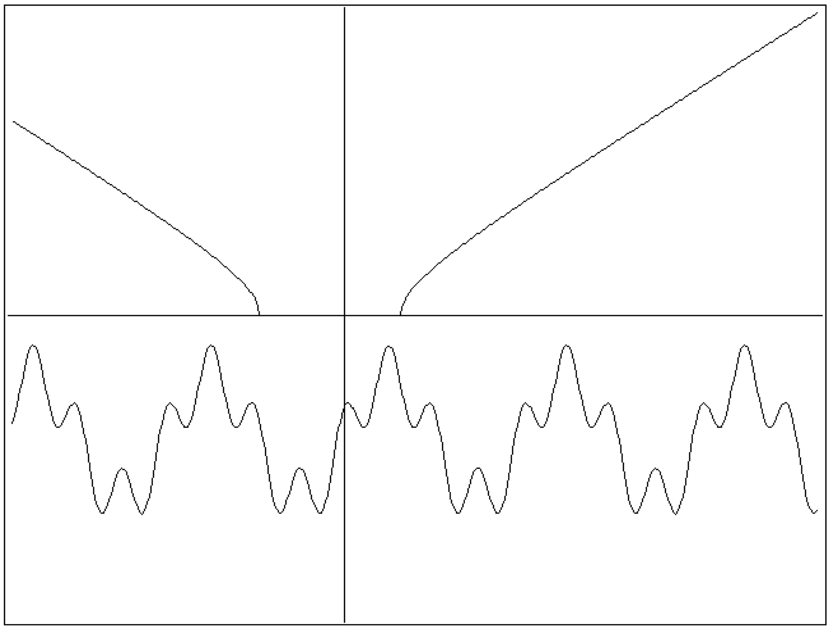
izvedel brez težav.

Poglejmo sedaj, kako lahko te ukaze uporabimo pri sestavljanju ukazov za risanje funkcij.

7.2 Risanje funkcij

V naslednjih vrsticah so podani ukazi za risanje funkcije F. Ukaz `OSI` nariše koordinatni sistem. Količini OX in OY določata mesto koordinatnega izhodišča na zaslonu, EX in EY pa dolžino enot na obeh oseh. Ukazi omogočajo tudi risanje funkcij, ki niso povsod definirane (glej sliko 7.1).

```
TO RISI
  CS HT SETPC 4
  MAKE "ox -50 MAKE "oy 0 MAKE "ex 20 MAKE "ey 20
  MAKE "dx :ox/:ex MAKE "dy :oy/:ey MAKE "h 1/:ex
  OSI :ox :oy :ex :ey
  MAKE "f [SQRT ((:x + 3)*(:x - 2))/3]
  SETPC 10 GRAF
  MAKE "f [ -4 + 2*(SIN(180*:x/PI)) + COS(720*:x/PI) ]
  SETPC 14 GRAF
  WAIT 0
```



Slika 7.1: Risanje funkcij

```
END
TO OSI :ox :oy :ex :ey
  PU SETPOS (LIST :ox -200) PD SETH 0 FD 400
  PU SETPOS (LIST -300 :oy) PD SETH 90 FD 600 PU
END
TO GRAF
  MAKE "i -285 PU
  REPEAT 570 [
    MAKE "x :i*:h - :dx  VREDNOST
    IF :def [ SETPOS (LIST :i :j) PD ] [ PU ]
    MAKE "i :i + 1 ]
END
TO VREDNOST
  CATCH "ERROR [ MAKE "j ((RUN :f) + :dy)*:ey
    MAKE "def "TRUE STOP ]
  MAKE "def "FALSE
END
```

Dani ukazi predstavljajo le okostje zaresnega sestava ukazov za risanje funkcij. V narečju LCSI/LOGOS je treba ukaz `RUN` v ukazu `VREDNOST` zamenjati z ukazom EVAL:

```
TO VREDNOST
  MAKE "j ((EVAL :f) + :dy)*:ey
  MAKE "def 0 = ERRORN
END
```

Funkcija ERRORN vrne številko zadnje napake in jo postavi zopet na 0.

7.3 Razvejitveni stavek CASE

V sodobnih programskih jezikih navadno najdemo tudi stavek, ki omogoča izbiro med več možnostmi. V logu bi izgledal recimo takole:

```
CASE izraz [
    v1  [ ukazi_1 ]
    ...
    vk  [ ukazi_k ]
    ...
  ELSE [ ukazi_* ]
]
```

Stavek CASE zahteva: če ima *izraz* vrednost v_k opravi *ukaze_k*; sicer opravi *ukaze_**. Člen ELSE in pripadajoči seznam ukazov lahko tudi izpustimo.

Za opisani stavek CASE je razmeroma preprosto sestaviti ustrezni ukaz v logu:

```
TO CASE :i :s
  LOCAL "@is
  MAKE "@is MEMBERN :i :s
  IF 0 < :@is
    [ RUN ITEM 1 + :@is :s ]
    [ MAKE "@is MEMBERN CM "ELSE :s
      IF 0 < :@is
        [ RUN ITEM 1 + :@is :s ] [ ]
    ]
END
```

Opis ukaza CASE smo zapisali v LCSI/LogoSu. Pri tem smo uporabili ukaza

MEMBERN x o	zaporedna številka prve pojavitve člena x v objektu o; če je ni, je vrednost 0
CM b	ukaz (klic ukaza) z imenom b

Ukaz MEMBERN je v bistvu enakovreden ukazu KJE, ki smo ga sestavili na strani 43. Namesto ukaza CM "ELSE, ki besedo "ELSE predela v ukaz ELSE, pa lahko napišemo, na primer, FIRST [ELSE] .

Delovanje ukaza CASE preverimo z naslednjim ukazom

```
TO CaseTest
  TS CT
  REPEAT 800 [
    CASE 1 + RANDOM 6 [
      1   [ TYPE "'. ' ]
      3   [ TYPE "'+ ' ]
      5   [ TYPE "'- ' ]
     ELSE [ TYPE "'* ' ]
    ]
  ]
END
```

7.4 Rekurzija

S pojmom rekurzije smo se že srečali. V logu je nepogrešljiva. Z uporabo rekurzije lahko sestavimo zelo kratke opise krivulj, kakršna je zmajnica (glej sliko 7.2).

```
TO Zmaj :n :a :h
  IF :n < 1 [ FD :h STOP ] []
  Zmaj :n - 1 90 :h RT :a  Zmaj :n - 1 -90 :h
END
```

Poskusite z zahtevo

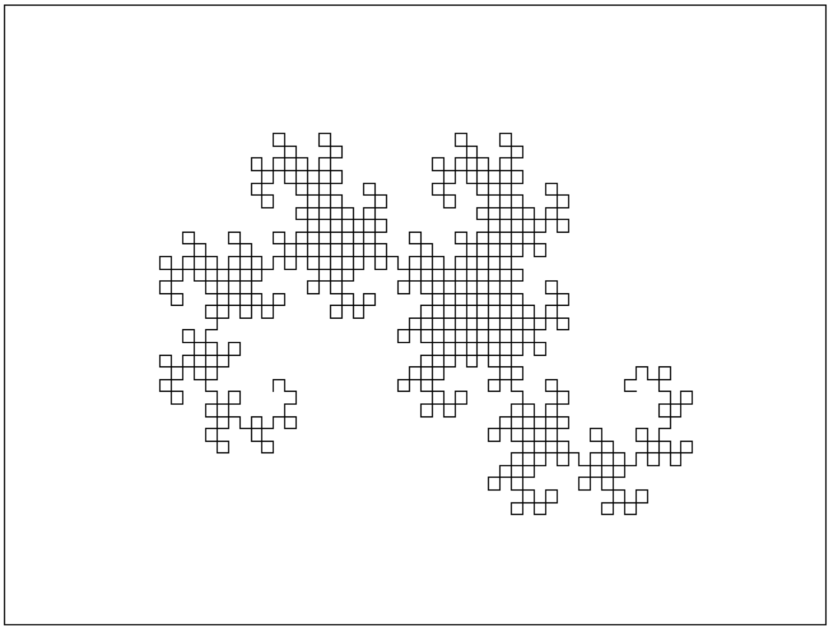
```
CS HT PU SETPOS [-100 -50] PD Zmaj 10 90 8
```

V naslednjih vrsticah bomo spoznali še dva primera uporabe rekurzije.

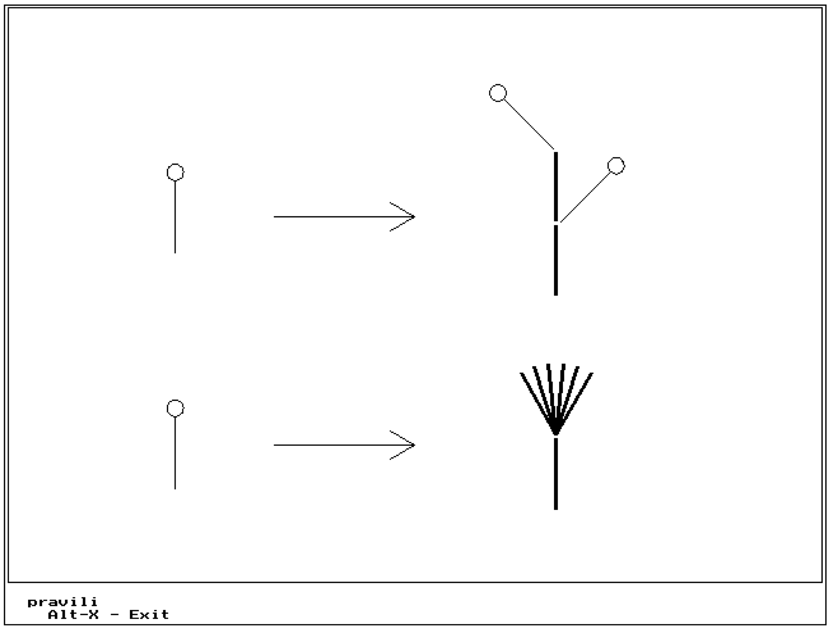
7.5 Drevo

Drevo raste po pravilih prikazanih na sliki 7.3, kjer debele črte predstavljajo olesenele dele. Pravili rasti lahko prepišemo v naslednja ukaza:

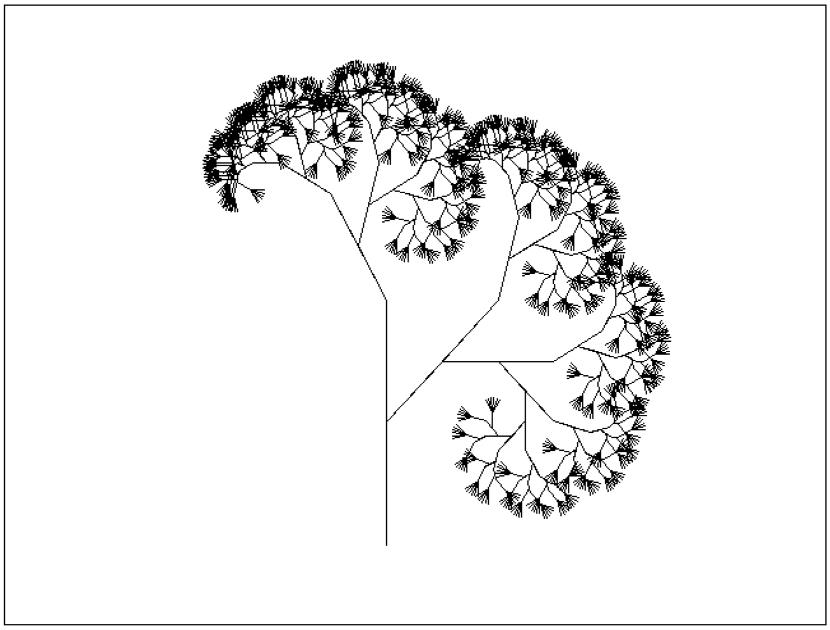
```
TO Iglice
  FD 10 LT 30
```



Slika 7.2: Zmajnica



Slika 7.3: Pravila rasti drevesa



Slika 7.4: Drevo

```
  REPEAT 6 [ FD 10 BK 10 RT 12 ]
  LT 42 BK 10
END
TO Deblo :n :d
  IF :n = 0 [ Iglice STOP ] []
  FD :d RT 45  Deblo (:n - 1) :d * 0.7
  LT 45 FD :d LT 30  Deblo (:n - 1) :d * 0.5
  RT 30 BK :d * 2
END
```

ki na zahtevo `Deblo 8 80` ustvarita drevo prikazano na sliki 7.4.

7.6 Sto

Med sosednji števki (cifri) v zaporedju

```
9 8 7 6 5 4 3 2 1
```

ali na začetek zaporedja vstavimo plus ali minus ali pa ju staknemo. Nato izračunamo vrednost tako dobljenega izraza. Na primer

```
98-76+54+3+21 = 100
```

Ali obstaja še kak tak izraz z vrednostjo 100 ? Vse odgovore na to vprašanje nam dasta na zahtevo `TS Izraz 8 [ 9 ]` ukaza

```
TO PREVERI :s
  IF 100 = RUN :s [ PR :s ] []
  MAKE "t FPUT (- FIRST :s ) BF :s
  IF 100 = RUN :t [ PR :t ] []
END
TO Izraz :n :s
  IF :n = 0 [ PREVERI :s STOP ] []
  Izraz :n - 1 LPUT :n (SE :s [ + ])
  Izraz :n - 1 LPUT :n (SE :s [ - ])
  Izraz :n - 1 LPUT 10*(LAST :s) + :n BL :s
END
```

V narečju LCSI/LogoS je treba ukaz `RUN` nadomestiti z ukazom `EVAL`. Kaj pa, če bi vrstni red števk obrnili 1,2,3,...,9 ?

7.7 Virusi v logu

V svojem prispevku na konferenci EUROLOGO 91 [12] je Gerald Futschek pokazal, da lahko v logu sestavimo tudi ukaze-viruse, ki okužijo druge uporabnikove ukaze. V tem razdelku je podana izvedba takega virusa v narečju LCSI/LogoS. Z manjšimi spremembami ga je mogoče predelati tudi za druga narečja loga.

Za prenašanje virusa na zdrave ukaze skrbi ukaz

```
TO Okuzi :Zrtve :Virus
  IF EMPTYP :Zrtve [ STOP ] [ ]
  LOCAL "Zrtev
  MAKE "Zrtev FIRST :Zrtve TYPE :Zrtev
  IF NOT (OR EQUALP :Zrtev "Okuzi
             EQUALP "@ ITEM 2 LAST TEXT :Zrtev)
    [ PR "' okuzen'
      DEFINE :Zrtev
        (LIST FIRST TEXT :Zrtev
              (SE :virus LAST TEXT :Zrtev))
    ] [ PR " ]
  Okuzi BF :Zrtve :Virus
END
```

ki na začetek opisa vsakega še neokuženega ukaza iz seznama *žrtve* doda *virus*ne ukaze. Sam virus je določen z ukazom

```
TO Sprozi
  LOCAL "@
  MAKE "@ [ Okuzi PROCEDURES
              (SE LPUT :@ [ LOCAL "@ MAKE "@ ] [ RUN :@ ])
             PR "'... Nasvidenje' ]
  RUN :@
  PR "'... Dober dan'
END
```

ki ob klicu povzroči širjenje virusa na ostale ukaze v delovnem prostoru. Funkcijski ukaz `PROCEDURES` vrne seznam vseh uporabniških ukazov v delovnem prostoru. Ukaza `PRINT` v ukazu `Sprozi` lahko izpustimo ali nadomestimo z drugimi bolj ali manj prijaznimi ukazi.

Za preizkus delovanja virusa sestavimo še dva ukaza

```
TO Abc
  PR "'... Abc'
END
TO Preveri
  TS CT SHOW TEXT "Abc PR "
  Sprozi SHOW TEXT "Abc PR "  WAIT 0
  CT Abc WAIT 0
  CT LOAD "roze Abc WAIT 0
END
```

in zahtevamo `Preveri`.

7.8 Naloga: Rekurzivna risba

Sestavi ukaze v logu, ki narišejo neko rekurzivno risbo (drevo, krivuljo ...).

8.

Rešitve nalog

Rešitve so pripravljene za LCSI/LogoS za računalnik PC. Če na vašem računalniku slika sili čez rob zaslona, razpolovite vrednosti v ukazih `FD` in `BK` – npr. namesto `FD 70` napišite `FD 35`.

8.1 Olimpijski krogi

Objavljamo nekoliko predelano rešitev, ki jo je poslala Urška Lesjak iz Kopra:

```
TO KROG
  REPEAT 360 [ FD 70 PD FD 7 PU BK 77 LT 1]
END
TO KROGI
  CS HT PU FD 50 RT 90
  FD 165 LT 90 SETPC 4 KROG
  LT 90 FD 165 SETPC 8 KROG
  FD 165 SETPC 1 KROG
  RT 40 BK 110 RT 140 SETPC 14 KROG
  FD 165 SETPC 2 KROG
END
```

Risanje krogov sprožimo z zahtevo `KROGI`.

8.2 Pismo

Že v prvi polovici 18. stoletja je znani matematik Leonhard Euler (1707–1783) dokazal: Risbo lahko narišemo v eni potezi natanko takrat, ko sta izpolnjena pogoja:

- po risbi lahko pridemo iz vsakega križišča v vsako drugo križišče;
- na risbi sta največ dve križišči, v katerih se stika liho število črt.

Če ni nobenega lihega križišča, lahko začnemo risanje v katerem koli križišču; če pa sta dve lihi križišči, moramo risanje začeti v enem izmed njiju in v drugem končati.

V primeru pisma sta lihi križišči v levem in desnem spodnjem vogalu. Sedaj ni več težko najti rešitev in jo zapisati v logu:

```
TO PISMO
  CS PU SETPOS [ -200 -120 ] PD
  SETPOS [    0  -40 ]  WAIT 50
  SETPOS [ -200   40 ]  WAIT 50
  SETPOS [    0  120 ]  WAIT 50
  SETPOS [  200   40 ]  WAIT 50
  SETPOS [  200 -120 ]  WAIT 50
  SETPOS [ -200 -120 ]  WAIT 50
  SETPOS [ -200   40 ]  WAIT 50
  SETPOS [  200   40 ]  WAIT 50
  SETPOS [    0  -40 ]  WAIT 50
  SETPOS [  200 -120 ]
END
```

Risanje pisma sprožimo z zahtevo `PISMO`. V rešitev so dodani ukazi `WAIT 50`, ki po vsaki narisani črti zadržijo izvajanje za eno sekundo in tako omogočijo preveritev pravilnosti rešitve.

8.3 Poštevanka do 100

Nalogo je bilo mogoče rešiti na več načinov. Najpreprostejša, čeprav nekoliko daljša rešitev je napisati 11 ukazov oblike

```
PR [ 3 3 6 9 12 15 18 21 24 27 30 ]
```

Ta ukaz izpiše vrstico poštevanke, ki pripada številu 3. Stolpci v tako dobljeni tabeli niso poravnani. To lahko dosežemo, če enomestne številke dopolnimo spredaj s presledkom in izpišemo kot nize – na primer številko 6 v seznamu nadomestimo z nizom `"\ 6`.

Tisti, ki bolje poznajo LOGO in prisegajo na to, da je lenoba gibalo razvoja, pa bi se najbrž odločili za krajšo rešitev:

```
TS CT  MAKE "a 1  MAKE "b 0  MAKE "y 1
REPEAT 10 [
  SETCUR (LIST 16 + 4*:a :y) TYPE :a  MAKE "a :a + 1 ]
REPEAT 10 [
  MAKE "a 1 MAKE "b :b + 1 MAKE "y :y + 2
  SETCUR (LIST 16 :y) TYPE :b
  REPEAT 10 [
    SETCUR (LIST 16 + 4*:a :y) TYPE :a*:b
    MAKE "a :a + 1 ]
]
```

8.4 Praštevilo

Nalogo je bilo enostavno rešiti z uporabo ukaza `PRAFAKTORJI` s strani 42 – število je praštevilo natanko takrat, ko ima en sam prafaktor. Poglejmo si še premo rešitev:

V rešitvi naloge je upoštevano, da ima naravno število n, ki ni praštevilo, vsaj en delitelj, ki ne presega $\sqrt{n}$. Tako dobimo razmeroma učinkovit postopek preverjanja praštevilskosti:

```
TO PRA :n
  LOCAL "k MAKE "k 2
  REPEAT (ROUND SQRT :n) - 1 [
    IF 0 = REMAINDER :n :k [ OP "FALSE ]
      [ MAKE "k :k + 1 ] ]
  OP "TRUE
END
TO PRASTEVILO :n
  (TYPE "stevilo\  :n "\ )
  IF PRA :n [TYPE "je][TYPE "ni] PR "\ prastevilo
END
```

8.5 Beseda

Kot zadovoljivo rešitev naloge bomo šteli vsako besedo, ki za manj kot 10 odstotkov odstopa od ciljne vrednosti 500000.

Najprej napišimo tabelo vrednosti naših črk:

A	1	E	6	J	11	O	16	T	21
B	2	F	7	K	12	P	17	U	22
C	3	G	8	L	13	R	18	V	23
Č	4	H	9	M	14	S	19	Z	24
D	5	I	10	N	15	Š	20	Ž	25

Pri reševanju naloge je bilo mogoče uporabiti logo na različne načine. Najpreprosteje za računanje produktov – na primer za besedo MUREN

```
PRINT 14*22*18*6*15
```

Če smo leni in se nam ne ljubi samim iskati vrednosti črk, lahko to zaupamo logu

```
TO KJE :z :b
  LOCAL "k MAKE "k 1
  REPEAT COUNT :b
    [IF EQUALP :z ITEM :k :b [OP :k][MAKE "k :k+1]]
  OP 0
END
TO V :b
  IF EMPTYP :b [OP 1] []
  OP (KJE FIRST :b "abcxdefghijklmnoprsytuvzw ) * V BF :b
END
```

pri tem smo naše črke **č**, **š** in **ž** nadomestili zaporedoma s črkami **x**, **y** in **w**.

Vrednost najboljše, a nekoliko vprašljive, rešitve ŽABA DIŠI dobimo tedaj z zahtevo

```
PRINT V "wabadiyi
```

Z uporabo nekoliko predelanega ukaza `PRAFAKTORJI` s strani 42 pa lahko določimo števila okrog 500000, ki so mogoče vrednosti besed – nimajo nobenega prafaktorja, večjega od 23. To so: 500000, 499928, 500094, 499905, 499851, 500175, 499800, 500346, 499681, 500480, ...

Naštejmo še nekaj rešitev iz odgovorov reševalcev:

500000	ŽABA DIŠI
500480	POVOD
499392	POPER
499200	ČOLNIČ, GNILOBA, NABODALO
498960	BATERIJA, MUREN, RUMENA, KUHATI

8.6 Zvok

Če uporabimo ukaza `NOTA` in `IGRAJ` s strani 50, lahko zapišemo "Naša četica koraka" takole:

```
TO CETICA :t
  NOTA 7 3*:t  NOTA 5 :t
  IGRAJ [ 4 5 7 9 7 7 9 9 9 9 ] 2*:t
  IGRAJ [ 7 7 ] 4*:t  NOTA 5 3*:t  NOTA 9 :t
  IGRAJ [ 7 5 4 7 5 4 2 5 4 2 4 5 ] 2*:t
  NOTA 7 4*:t  NOTA 5 3*:t  NOTA 9 :t
  IGRAJ [ 7 5 4 7 5 4 2 5 4 2 ] 2*:t
  NOTA 0 4*:t  NOTA 0 :t
END
```

Igranje sprožimo z zahtevo `CETICA 10`.

8.7 Rekurzivna risba

Za primer napišimo ukaz `Hilbert`, ki nariše Hilbertovo krivuljo (slika 8.1):

```
TO Hilbert :n :a :h
  IF :n = 0 [ STOP ] []
  RT :a Hilbert :n - 1 (-:a) :h FD :h
  LT :a Hilbert :n - 1 :a :h FD :h
  Hilbert :n - 1 :a :h LT :a FD :h
  Hilbert :n - 1 (-:a) :h RT :a
END
```

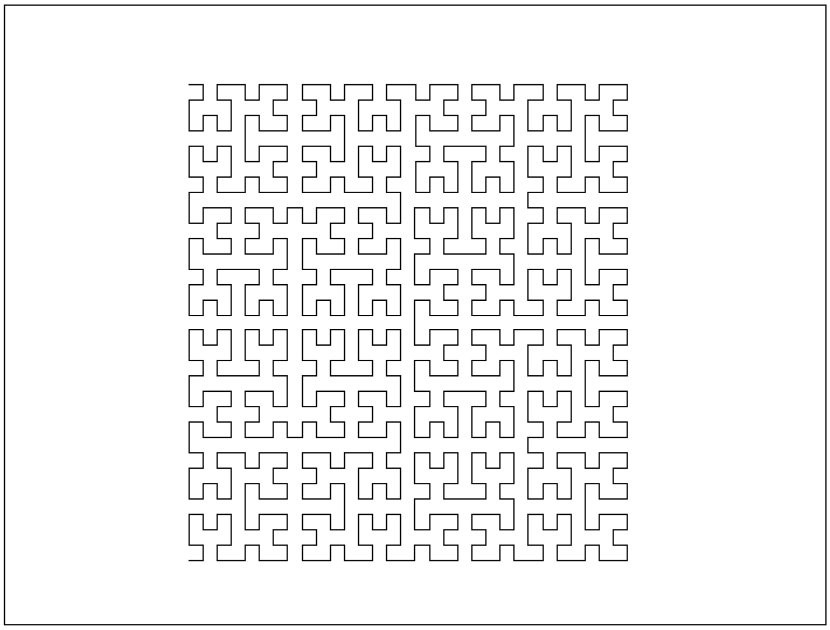
Risanje sprožimo z zahtevo:

```
CS HT PU SETPOS [ -160 -160 ] PD Hilbert 5 90 10
```

8.8 Kako smo žrebali

Povejmo še, kako smo žrebali nagrajence. Vsakič smo pregledali odgovore na nagradno nalogo in zadovoljive oštevilčili z zaporednimi številkami. Nato smo z zahtevo `ZREB` sprožili ukaza:

```
TO IZBERI :p :r
  IF :p = 0 [ STOP ] []
```



Slika 8.1: Hilbertova krivulja

```
  MAKE "k 1 + RANDOM :r PR :k
  IF MEMBERP :k :s [ IZBERI :p :r ]
    [ MAKE "s LPUT :k :s IZBERI (:p - 1) :r ]
END
TO ZREB
  RANDOMIZE TS CT MAKE "s [ ]
  TYPE [ resitev = ]  MAKE "r RL
  TYPE [ nagrad = ]  MAKE "n RL
  IZBERI :n :r  SHOW :s  WAIT 0
END
```

Kot podatka smo vnesli število zadovoljivih rešitev in število nagrad (4). Na zaslonu smo dobili seznam štirih naključnih številk, ki so predstavljale številke nagrajenih rešitev.

Literatura

[1] Abelson H., di Sessa A.: *Turtle Geometry; The Computer as a Medium for Exploring Mathematics.* The MIT Press, Cambridge Massachusetts 1981.

[2] Allan B.: *Introducing LOGO.* Granada, London 1984.

[3] Batagelj V.: *LOGO-S, osnutek.* Ljubljana, oktober 1989.

[4] Batagelj V., Horvat B., Pisanski T.: *Logo na računalniku Spectrum - SOLI/LCSI Sinclair Logo.* Presek 14(1986-87)3, 162-169.

[5] Batagelj V., Horvat B., Pisanski T.: *Logo na računalniku Commodore 64.* Presek 14(1986-87)4, 232-239.

[6] Batagelj V. in sodelavci: *Z računalnikom v matematiko.* Računalniška obzorja, DZS, Ljubljana 1987.

[7] Bratley P., Millo J.: *Self-Reproducing Programs.* Software – Practice and Experience, **2**(1972), 397-400.

[8] *Bilten 6. republiškega tekmovanja iz znanja računalništva za osnovnošolce.* Gimnazija Bežigrad, Ljubljana 11. maja 1991. ZOTKS, Ljubljana 1991.

[9] *Byte, the small systems journal; Logo.* 7(August 1982)8.

[10] *CICIBAN 1988.* Šlenc T., Rovtar B.: več prispevkov o logu.

[11] *EUROLOGO 91.* (E. Calabrese, Ed.) Proceedings, Third European Logo Conference, Parma, Italy, 27-30 August 1991. A.S.I., Parma 1991.

[12] Futschek G.: *Experiments with Self-reproductive Logo-programs; Viruses for Logo-programs and Countermeasures.* v EUROLOGO 91. (E. Calabrese, Ed.) Proceedings, Third European Logo Conference, Parma, Italy, 27-30 August 1991. A.S.I., Parma 1991, 311-327.

[13] Gerlič I.: *Prvi koraki v logo.* ZOTKS, Ljubljana 1988.

[14] Grigorieff V.: *Devenir champion de LOGO PLUS.* Marabout, Alleur (Belgija) 1989.

[15] Grobelnik M., Zupanič D.: *LOGO-S, verzija 1.0, tehnični priročnik.* Ljubljana, januar 1991.

[16] Harvey B.: *Computer Science Logo Style*, Volume 1: *Intermediate Programming*; Volume 2: *Projects, Styles, and Techniques.* The MIT Press, Cambridge, Massachusetts 1985, 1986.

[17] Haas E.F.: *Schüler programmieren in Logo.* Schöningh, Paderborn 1988.

[18] *Logo v osnovni šoli, program dela krožka.* Projekt 1988/90. Zavod SR Slovenije za šolstvo, Ljubljana, februar 1988.

[19] Papert S.: *Mindstorms: Children, Computers and Powerfull Ideas.* Basic Books, New York 1980.

[20] *PC Logo. Reference Manual, Version 3.0.* Harvard Associates, Inc., Cambridge, MA 1988.

[21] Prek S.: *Teorija glasbe.* DZS, Ljubljana 1990.

[22] Ross P.: *Logo Programming for the IBM PC.* Addison-Wesley, Wokingham, England 1985.

[23] Sauer G.: *Das große LOGO Buch zum ATARI ST.* Data Becker, Düsseldorf 1986.

[24] Solomon C., Minsky M., Harvey B.: *Logo Works, Challenging Programs in Logo.* McGraw-Hill, New York 1986.

[25] Voglar M., Nograšek M.: *Majhna sem bila.* DZS, Ljubljana 1990.

Izdal in založil: Zavod Republike Slovenije za šolstvo in šport
Predstavnik: dr. Srečko Zakrajšek
Urednica: Nada Razpet

Računalniško stavil: avtor
Opremila: avtor in Nataša Prosenc
Razmnožil: DOMUS
Naklada: 800 izvodov, prvi natis

Ljubljana, 1991