4.

Funkcije in odločitveni stavek

4.1 Številske funkcije

Vsoto $a + b$ lahko v logu zapišemo tudi v obliki `SUM` a b ; in veččleno vsoto $a_1 + a_2 + ... + a_n$ v obliki

(`SUM` a_1 a_2 ... a_n)

Podobno lahko za zmnožek $a * b$ uporabimo obliko `PRODUCT` a b ; in za $a_1 * a_2 * ... * a_n$ izraz

(`PRODUCT` a_1 a_2 ... a_n)

Ukaza `SUM` in `PRODUCT` sta funkcijska ukaza, ker dane podatke predelata v nov podatek – zahteva `SUM 3 4` predela števili 3 in 4 v njuno vsoto 7. Tudi za ostale računske operacije pozna logo funkcijske ukaze:

`DIFFERENCE` a b

je enakovreden izrazu $a - b$ in izračuna razliko števil a in b;

`QUOTIENT` a b

je enakovreden izrazu a/b in izračuna kvocient števil a in b;

`DIV` a b

izračuna celoštevilski kvocient celih števil a in b;

`REMAINDER` a b

izračuna ostanek pri deljenju celega števila a z b. V zadnjih treh funkcijah mora biti vrednost b različna od 0.

Poskusite

```
PR QUOTIENT 91 19
PR DIV 91 19
PR REMAINDER 91 19
```

Zaporedoma dobimo `4.78947368`, `4` in `15`.

Poleg funkcij za posamezne računske operacije pozna večina narečij loga še naslednje funkcije:

`PI`	$\pi = 3.14159265$
`INT` n	celi del
`ROUND` n	najbližje celo število
`SQRT` n	kvadratni koren, $n \geq 0$
`ABS` n	absolutna vrednost
`RANDOM` n	naključno celo število med 0 in $n-1$
`SIN` n	sinus
`COS` n	kosinus
`TAN` n	tangens
`ARCTAN` n	ločni tangens
`EXP` n	eksponentna
`LOG` n	naravni logaritem, $n > 0$

Trigonometrijske funkcije `SIN`, `COS`, `TAN` in `ARCTAN` delajo s koti, izraženimi v stopinjah. V LOGOSu pa lahko uporabljamo tudi funkcije `SINR`, `COSR`, `TANR` in `ATANR`, ki delajo s koti, izraženimi v radianih.

V nekaterih narečjih loga lahko z ukazom `RANDOMIZE` zagotovimo, da bo zaporedje naključnih števil, ki jih daje funkcija `RANDOM`, pri ponovni izvedbi programa drugačno.

4.2 Lastne funkcije

Uporabnik lahko sestavi tudi lastne funkcijske ukaze. Opis funkcijskega ukaza ima običajno obliko

```
TO ime parametri
  izračun vrednosti
  OUTPUT vrednost
END
```

Ukaz `OUTPUT` *v*, ali krajše `OP` *v*, prekine izvajanje funkcijskega ukaza in vrne vrednost *v* kot njegovo vrednost.

Tako na primer funkciji

```
TO ENICE :n
  OP REMAINDER ABS :n 10
END
TO DESETICE :n
  OP ENICE DIV :n 10
END
```

vrneta enice oziroma desetice danega celega števila. Na zahtevo

```
PR ENICE -253 PR DESETICE -253
```

logo izpiše števili `3` in `5`.

4.3 Znaki

V računalniku so tudi znaki shranjeni kot števila. Vsak znak je predstavljen z njegovo zaporedno številko v tabeli znakov. Predpisu, ki vsakemu znaku priredi zaporedno številko in obratno, pravimo *koda*. V računalništvu se najpogosteje uporablja koda ASCII. ASCII pomeni *American Standard Code for Information Interchange* – Standardna ameriška koda za izmenjavo podatkov.

Tabela ASCII vsebuje 256 znakov s številkami od 0 do 255. Številkam 0 – 31 in 128 – 255 na različnih računalnikih ustrezajo različni znaki.

Zvezo, ki jo določa koda ASCII, povzemata v logu funkcijska ukaza `ASCII` in `CHAR`. Zahteva `ASCII` *znak* vrne zaporedno številko danega znaka; zahteva `CHAR` *n* pa vrne znak z zaporedno številko *n*. Tako

```
PR ASCII "a PR CHAR 65
```

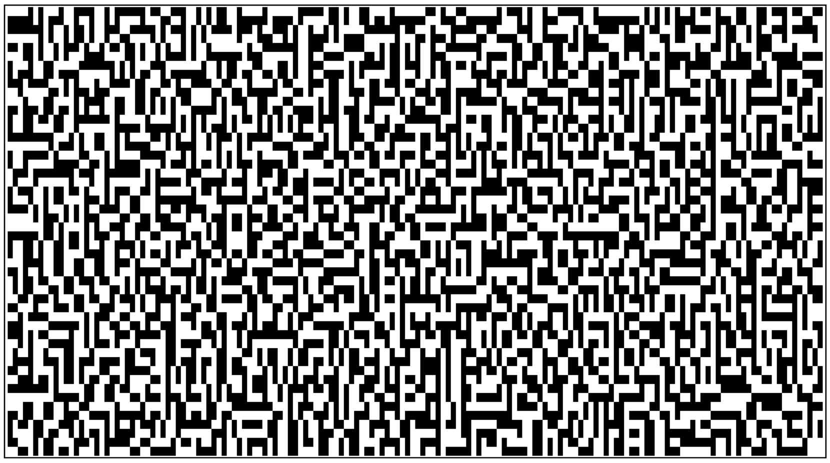
izpiše `97` in `A` .

Za izpis znaka, ki pripada dani številki, lahko sestavimo poseben ukaz

```
TO ZNAK :n
  (PR :n CHAR :n)
END
```

Na računalniku PC zahteva

```
ZNAK 201 ZNAK 205 ZNAK 187 ZNAK 200 ZNAK 188 ZNAK 186
```



Slika 4.1: Naključni vzorec

izpiše znake, iz katerih lahko sestavimo okvir.

Sestavimo še ukaz, ki izpiše vse znake s kodami od *od* do *do*:

```
TO Znaki :od :do
  TS CT MAKE "znak :od
  REPEAT :do - :od + 1 [
    (PR :znak CHAR :znak)
    MAKE "znak :znak + 1 ]
END
```

Poskusite `Znaki 31 50` . Oglejte si še druge znake na vašem računalniku.

Na računalniku PC kodam 220 – 223 ustrezajo pol beli, pol črni znaki. To je uporabljeno v naslednjem programu

```
TS CT
REPEAT 2000 [ TYPE CHAR 220 + RANDOM 4 ]
```

ki izpiše na znakovni zaslon naključni vzorec, kakršen je prikazan na sliki 4.1.

4.4 Pogoji

Logo pozna tudi logični vrednosti `"TRUE` (resnično) in `"FALSE` (lažno). Izrazom, ki dajo logično vrednost pravimo *pogoji.* Enostavni pogojni izrazi imajo obliko primerjav

$a < b$	manjši
$a = b$	enak
$a > b$	večji

Poskusite

```
MAKE "c 1  PR :c < 5  PR 0 = :c
```

Logo izpiše `TRUE` in `FALSE`.

Logično vrednost vračajo tudi nekatere vgrajene funkcije. Lahko pa tako funkcijo sestavi tudi uporabnik. Na primer, kakor vemo, je število n deljivo s številom d natanko takrat, ko je ostanek pri deljenju n z d enak 0. To izrazimo s funkcijo

```
TO DELJIV :n :d
  OP 0 = REMAINDER :n :d
END
```

Pogoje lahko povežemo v sestavljene pogoje z izjavnimi povezavami

`NOT` a	negacija; ni res a
`AND` a b	konjunkcija; a in b
`OR` a b	disjunkcija; a ali b

katerih delovanje je opisano s tabelama

a	`NOT` a
F	T
T	F

in

a	b	`AND` a b	`OR` a b
F	F	F	F
F	T	F	T
T	F	F	T
T	T	T	T

V tabelah T označuje `"TRUE`, F pa `"FALSE`. Uporabljamo lahko tudi obliki (`OR` a_1 a_2 ... a_n) in (`AND` a_1 a_2 ... a_n) .

Z uporabo izjavnih povezav lahko pogoj $a \leq x < b$ zapišemo v logu takole

```
AND OR :a=:x :a<:x :x<:b
```

4.5 Odločitveni stavek

Pogoje uporabljamo v ukazih oblike

`IF` *pogoj* `[` *res_ukazi* `]` `[` *laž_ukazi* `]`

ki preveri, ali je dani *pogoj* izpolnjen (resničen). Če je izpolnjen, opravi *res_ukaze*, sicer pa opravi *laž_ukaze*.

Tako na primer ukaz

```
TO VECJI :a :b
  IF :a > :b [ OP :a ] [ OP :b ]
END
```

vrne večje izmed števil a in b. Na zahtevo

```
PR VECJI -14 7  PR VECJI 8 5
```

logo izpiše 7 in 8.

Odločitveni stavek ima v različnih narečjih loga še druge oblike. Na primer:

`IF` *pogoj* `THEN` *res_ukazi*

ki jo lahko nadaljujemo z

`ELSE` *laž_ukazi*

Ker večina narečij loga ne dopušča, da bi se stavek raztezal čez več vrstic, poznajo stavke

`TEST` *pogoj*
`IFTRUE` *res_ukazi*
`IFFALSE` *laž_ukazi*

Prvi ukaz preveri dani *pogoj*; preostala dva pa povesta, kaj naj se opravi glede na možna izida preverjanja.

Pri pisanju lastnih ukazov nam bo prišel prav še ukaz `STOP`, ki zahteva prekinitev izvajanja ukaza, v opisu katerega se nahaja. Izvajanje se nadaljuje na ukazu, ki sledi klicu – zahtevi, ki je sprožila prekinjeni ukaz.

4.6 Rekurzija

Kot nekoliko zapletenejši primer si oglejmo funkcijski ukaz `VSOTA`, ki vrne kot vrednost vsoto števk (cifer) v desetiškem zapisu števila n. Postopek temelji na ugotovitvi, da je vsota vseh števk enaka vsoti enic in vsote števk števila brez enic

```
TO VSOTA :n
  IF :n = 0 [ OP 0 ]
    [ OP (REMAINDER :n 10) + VSOTA DIV :n 10 ]
END
```

Kaj se zgodi, ko zahtevamo

```
PR VSOTA 3958
```

Najprej se izračuna `VSOTA 3958`

`VSOTA 3958`	$\rightarrow$	`8 + VSOTA 395`
`VSOTA 395`	$\rightarrow$	`5 + VSOTA 39`
`VSOTA 39`	$\rightarrow$	`9 + VSOTA 3`
`VSOTA 3`	$\rightarrow$	`3 + VSOTA 0`
`VSOTA 0`	$\leftrightarrow$	`0`
`VSOTA 3`	$\leftarrow$	`3 + 0` = `3`
`VSOTA 39`	$\leftarrow$	`9 + 3` = `12`
`VSOTA 395`	$\leftarrow$	`5 + 12` = `17`
`VSOTA 3958`	$\leftarrow$	`8 + 17` = `25`

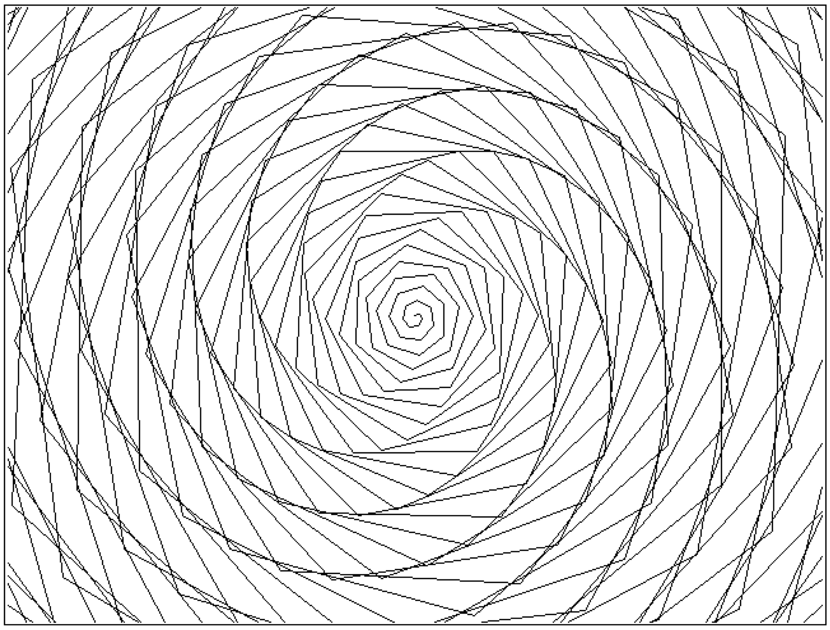
Torej `VSOTA` preda ukazu `PR` vrednost 25, ki jo le-ta izpiše.

Kot vidimo, ukaz `VSOTA` zahteva v svojem opisu ponovno svojo izvedbo. Takim ukazom, ki posredno ali neposredno zahtevajo svojo ponovno izvedbo, pravimo, da so *rekurzivni*. Za programiranje v logu je uporaba rekurzivnih ukazov značilna.

Sami preiskusite še ukaz

```
TO SPIRALA :d :s :a
  IF :d > 300 [ STOP ] [ ]
  FD :d RT :a
  SPIRALA (:d + :s) :s :a
END
```

recimo z zahtevama `SPIRALA 0 4 90` in `SPIRALA 0 3 178`. Na sliki 4.2 je prikazana `SPIRALA 0 1.2 53`.



Slika 4.2: Spirala

4.7 Naloga: Praštevilo

V logu sestavi ukaz `PRASTEVILO :n`, ki za dano naravno število n izpiše, ali je praštevilo ali ni.

Če ste pozabili, naravno število $n > 1$ je praštevilo natanko takrat, ko je deljivo samo z 1 in samim seboj – torej, ko ima natanko dva delitelja.

5.

Besede in seznami

5.1 Še nekaj o besedah

Pomudimo se najprej nekoliko pri samem zapisu besed. Kot vemo, besedo napovemo z dvojnimi navednicami `"`, ki jim sledi niz znakov, ki sestavlja besedo. No, v tem nizu ne smemo naravnost uporabiti nekaterih znakov. To so presledek ␣ in znaki

```
+ - * / ( ) [ ] < > = " : ; \
```

ter v narečju LogoS še znaki

```
$ % & , | ~ { } '
```

Nekatera narečja uporabljajo znak `#` za napoved zapisa števila v drugi osnovi. Tako so na primer `#B1011`, `#013` in `#H0B` zaporedoma zapisi števila 11 v dvojiškem, osmiškem in šestnajstiškem sestavu.

Če vseeno želimo katerega izmed omenjenih znakov vključiti v besedo, moramo predenj postaviti *ubežni znak* `\` (nekatera narečja loga poznajo druge ubežne znake) ali pa uprabiti ukaz `CHAR`. Tako ukaz

```
MAKE "kraj "Nova\ Gorica
```

shrani v spremenljivko *kraj* besedo `Nova␣Gorica`. Če pa bi ubežnemu znaku `\` sledila dva presledka, bi se logo pritožil, da ne pozna ukaza `Gorica`.

Tako lahko v večini narečij loga uporabljamo v besedah kot navadne znake le znake

```
. ! ? _ ^ @ `
```

Pogosto uporabljamo imena, ki vsebujejo znak `@` ali `.`, za poimenovanje "sistemskih" spremenljivk in ukazov.

V narečju LogoS lahko s tem, da del besede postavimo v enojne navednice `'`, zahtevamo, da ga upošteva dobesedno. Tako bi lahko prejšnji ukaz zapisali tudi v obliki:

```
MAKE "kraj "'Nova Gorica'
```

Do sedaj smo spoznali za delo z besedami funkcijske ukaze `ASCII`, `CHAR` in `WORD`. Poleg teh poznajo nekatera narečja loga še funkciji: `LOWERCASE` w, ki vse črke besede w predela v male; in `UPPERCASE` w, ki vse črke besede w predela v velike. Uporabimo lahko tudi okrajšavi `LC` in `UC`. Tako na primer na zahtevo

```
MAKE "b "LogoS PR :b PR UC :b PR LC :b
```

logo izpiše zaporedoma `LogoS`, `LOGOS` in `logos`.

Preberemo lahko tudi znak, ki ga uporabnik pritisne na tipkovnici. To nam omogoča ukaz `READCHAR`, ali krajše `RC`, ki počaka na pritisk na tipko in vrne ustrezni znak. Ali je pritisnjena kaka tipka, pa nam pove logična funkcija `KEYP`.

Uporabniki narečja LCSI/LogoS imajo na voljo ukaz `RdEKey`, ki omogoča branje tudi posebnih (smerne, funkcijske ...) tipk na tipkovnici. Vrne seznam kod pritisnjene tipke.

Ukaz `READCHAR` nam omogoča napisati ukaz, ki izpisuje kode ASCII znakov s tipk, na katere pritisnemo. Zapustimo ga s tipko $\boxed{\text{ENTER}}$, ki ima kodo ASCII 13.

```
TO TIPKE
  MAKE "@z RC IF 13 = ASCII :@z [ STOP ] []
  (TYPE "ASCII\( :@z "\)\ \=\ ) PR ASCII :@z
  TIPKE
END
```

5.2 Delo z besedami in seznami

Obstaja še nabor funkcij, ki delujejo tako nad besedami kot tudi nad seznami. Kadar je podatek lahko ali beseda ali seznam, mu bomo rekli objekt. Naštejmo jih:

`EMPTYP` *o*	ali je objekt *o* prazen
`COUNT` *o*	število členov v objektu *o*

`MEMBERP` x o	ali x pripada objektu o
`FIRST` o	prvi člen objekta o
`LAST` o	zadnji člen objekta o
`ITEM` n o	n–ti člen objekta o
`BUTFIRST` o	objekt o brez prvega člena; okrajšava `BF`
`BUTLAST` o	objekt o brez zadnjega člena; okrajšava `BL`
`FPUT` o s	dodaj objekt o na začetek seznama s
`LPUT` o s	dodaj objekt o na konec seznama s
`EQUALP` o_1 o_2	ali sta objekta o_1 in o_2 enaka

Sedaj pa si poglejmo njihove učinke. Ukazi

```
MAKE "a "Peter MAKE "b ["V "S "Z "J]
PR COUNT :a PR COUNT :b PR ITEM 3 :a
PR MEMBERP FIRST "JV :b PR FPUT FIRST :b BF :a
```

izpišejo zaporedoma `5`, `4`, `t`, `TRUE` in `Veter`.

Ukaz `READLIST`, ali krajše `RL`, prebere z zaslona vnešeni objekt. O tipu vsebine spremenljivke x pa lahko poizvemo z ukazi

`NAMEP` x	ali ima x vrednost
`NUMBERP` x	ali je x število
`WORDP` x	ali je x beseda
`LISTP` x	ali je x seznam

Večkrat pride prav tudi naslednji ukaz `ZNAKI`, ki iz dane besede b ustvari seznam znakov, ki jo sestavljajo:

```
TO ZNAKI :b
  IF EMPTYP :b [ OP [] ][]
  OP FPUT FIRST :b ZNAKI BF :b
END
```

Poskusite z zahtevo

```
SHOW ZNAKI "matematika
```

Logo izpiše seznam

```
[ m a t e m a t i k a ]
```

Ukaz `PRAFAKTORJI` pa ustvari seznam prafaktorjev danega števila n

```
TO @F :n :d
  IF :n<2 [ OP [] ] []
  IF :d*:d>:n [ OP (LIST :n) ] []
  IF 0=REMAINDER :n :d [ OP FPUT :d @F DIV :n :d :d] []
  OP @F :n :d+1
END
TO PRAFAKTORJI :n
  OP @F :n 2
END
```

Na zahtevo

```
SHOW PRAFAKTORJI 499905
```

dobimo

```
[3 3 3 5 7 23 23]
```

Poskusite še števila 499928, 500000 in 500094.

5.3 Naključni izbor

Razmeroma preprosto je sestaviti ukaz `IZBERI`, ki naključno izbere neki člen iz dane besede ali seznama

```
TO IZBERI :s
  OP ITEM (1+RANDOM COUNT :s) :s
END
```

Tako smo na zahtevo

```
REPEAT 20 [TYPE IZBERI "VSZJ]
```

dobili zaporedje znakov

```
JZVZSSVVSJJVSVJZSZVV
```

5.4 Iskanje

Napišimo še ukaz `KJE`, ki vrne zaporedno številko člena z v besedi / seznamu b; če ga ne najde, vrne številko 0

```
TO KJE :z :b
  MAKE "@k 1
  REPEAT COUNT :b [
    IF EQUALP :z ITEM :@k :b [ OP :@k ] []
    MAKE "@k :@k + 1 ]
  OP 0
END
```

Preiskusimo ga!

```
MAKE "s "AEIOU  PR KJE "I :s  PR KJE "B :s
```

Logo izpiše odgovora 3 in 0. LCSI/LogoS pozna ukaz `MEMBERN`, ki ima isti učinek kot ukaz `KJE`.

5.5 Ulomki in lokalne spremenljivke

Večkrat v opisu ukaza uporabimo spremenljivke, v katere shranimo kak začasni rezultat in jih po izteku ukaza ne potrebujemo več. Take spremenljivke lahko proglasimo za *lokalne spremenljivke* z ukazom `LOCAL` (glej primere v tem razdelku). S tem dosežemo dvoje:

- tolmač po izteku ukaza sprosti lokalnim spremenljivkam pripadajoči prostor;
- če v okolju, v katerem kličemo ukaz, obstaja spremenljivka z istim imenom kot lokalna spremenljivka, njene vrednosti ne pokvarimo nehote.

Poglejmo si še en primer uporabe seznamov. Ulomek $\frac{a}{b}$ lahko predstavimo s seznamom $[a\ b]$. Za delo z ulomki lahko tedaj sestavimo ustrezne ukaze. Ukaz `NAJDEL` določi največji skupni delitelj števil a in b (Evklidov algoritem)

```
TO NAJDEL :a :b
  IF :b=0 [OP ABS :a] [OP NAJDEL :b REMAINDER :a :b]
END
```

Ukaz `OBLIKA` vrne dani ulomek v okrajšani obliki, pri čemer je imenovalec pozitiven.

```
TO OBLIKA :a
  LOCAL "d
```

```
  IF 0 = LAST :a [ OP :a ] []
  MAKE "d NAJDEL FIRST :a LAST :a
  IF 0 < LAST :a
    [ OP (LIST DIV FIRST :a :d DIV LAST :a :d) ]
    [ OP (LIST (-DIV FIRST :a :d) (-DIV LAST :a :d)) ]
END
```

Sestavimo še ukaz SESTEJ, ki določi vsoto danih dveh ulomkov

```
TO SESTEJ :a :b
  (LOCAL "s "i)
  MAKE "s (FIRST :a)*(LAST :b) + (FIRST :b)*(LAST :a)
  MAKE "i (LAST :a)*(LAST :b)
  OP OBLIKA (LIST :s :i)
END
```

Ostale, manjkajoče ukaze za delo z ulomki boste sedaj najbrž zmogli dodati sami.

Preverimo ukaze

```
MAKE "u1 [6 -9] MAKE "u2 [-3 -4]
SHOW :u1 TYPE "\+ SHOW :u2 TYPE "\= SHOW SESTEJ :u1 :u2
```

Dobimo

```
[6 -9]+[-3 -4]=[1 12]
```

5.6 Naloga: Beseda

Vrednost posamezne črke je enaka njeni zaporedni številki v (naši) abecedi. Tako ima A vrednost 1, B ima vrednost 2, ...in Ž ima vrednost 25. Vrednost besede je enaka produktu vrednosti črk, ki jo sestavljajo. Na primer

$$v(\texttt{BABICA}) = 2 \cdot 1 \cdot 2 \cdot 10 \cdot 3 \cdot 1 = 120$$

$$v(\texttt{LOGOS}) = 13 \cdot 16 \cdot 8 \cdot 16 \cdot 19 = 505\,856$$

Poišči slovensko besedo, ki ima vrednost čim bližjo številu 500 000. Kako si si pri tem pomagal(a) z logom ?

6.

Zvok

Fizikalno je zvok valovanje. Če je to valovanje enakomerno – ima določljivo frekvenco, mu pravimo *ton*; sicer govorimo o *šumu*. Frekvenco ali, kakor pravijo glasbeniki, *višino* merimo v številu nihajev na sekundo – hertzih (Hz). Tako je 300 Hz = 300 nihajev na sekundo. Ljudje slišimo zvok v obsegu od 15–20000 Hz in lahko pojemo v obsegu 200–2000 Hz.

Ton sestavljajo štiri sestavine: višina, trajanje, moč (glasnost, amplituda nihanja) in barva. Barva je določena z ovojnico – krivuljo spreminjanja moči.

6.1 Zvok na računalniku

Večina mikroračunalnikov lahko ustvarja zvok. Vendar so razlike med njimi precejšne. V najpreprostejših izvedbah zvoka (Spectrum, PC) lahko spreminjamo le višino in trajanje. V ostalih mikroračunalnikih (C-64, Atari, Amiga, Spectrum +2) je ustvarjanje zvoka prepuščeno posebnemu čipu, ki lahko nadzira vse štiri sestavine tona. Navadno lahko tudi upravlja z nekaj vzporednimi viri zvoka. Na nekaterih mikroračunalnikih lahko računalnik s posebnim vmesnikom MIDI (Musical Instrument Digital Interface) povežemo z elektronskimi zvočnimi napravami (sintesizerji, zvočniki, mikrofoni ...). Seveda se vse te zmožnosti odražajo tudi v naboru ukazov, s katerimi krmilimo zvok. Tudi zvočne zmogljivosti računalnika PC lahko dopolnimo z vgraditvijo posebne zvočne kartice. Zadnji čas se posebej uveljavlja kartica *Sound Blaster*.

Pri našem seznanjanju z zvokom v logu se bomo držali planinskih načel in se prilagodili najšibkejšemu členu v verigi. Predpostavili bomo, da imamo na voljo ukaz `BEEP` v t , ki ustvari $t/50$ sekunde trajajoči ton višine v.

V narečju LCSI/LOGOS je ukazu BEEP enakovreden ukaz TONE v t; lahko pa ga tudi sestavimo iz še osnovnejših ukazov

SOUND v	začni pisk višine v;
NOSOUND	prekini piskanje;
WAIT t	počakaj $t/50$ sekunde.

takole

```
TO BEEP :v :t
  SOUND :v WAIT :t NOSOUND
END
```

Če za programiranje zvoka uporabljamo ukaz SOUND, lahko posamezni ukaz WAIT nadomestimo s kakim kratkim koristnim opravilom.

Atarijev logo pozna ukaz TOOT c v g t , kjer je c vir zvoka (0 ali 1) in g (0 do 15) moč tona. Z njim lahko izrazimo ukaz BEEP takole:

```
TO BEEP :v :t
  TOOT 0 :v 15 :t
END
```

Kako je z zvokom na računalniku C-64, je opisano v knjigi [13, str. 128-132]. Bralec naj sam sestavi ukaz BEEP z uporabo Commodorejevega ukaza SOUND.

6.2 Zvočni učinki

Ukaz BEEP lahko uporabimo za ustvarjanje najrazličnejših zvočnih učinkov. Tako na primer ukaz

```
REPEAT 10 [ BEEP 267 40 BEEP 200 30 ]
```

posnema sireno reševalnega avtomobila. Pozvonimo pa z

```
REPEAT 8 [ REPEAT 2 [
  REPEAT 12 [ BEEP 700 1 ] WAIT 10 ] WAIT 70 ]
```

Ukaz

```
TO ZMES :n :od :do :t
  REPEAT :n [ BEEP :od + RANDOM (:do - :od) :t ]
END
```

zaigra n naključnih tonov z višinami med *od* in *do* ter trajanjem t. Poskusite

```
ZMES 50 100 500 10
```

Zaporedno igranje tonov nam omogoča ukaz

```
TO TONI :od :do :k :t
  MAKE "v :od
  REPEAT ABS DIV (:do - :od) :k
    [ BEEP :v :t MAKE "v :v + :k ]
END
```

ki zaigra vse tone od višine *od* do višine *do* s korakom k in trajanjem t. Najprej poskusite

```
TONI 0 1000 25 20
```

Zanimiv učinek pa ustvari tudi zahteva

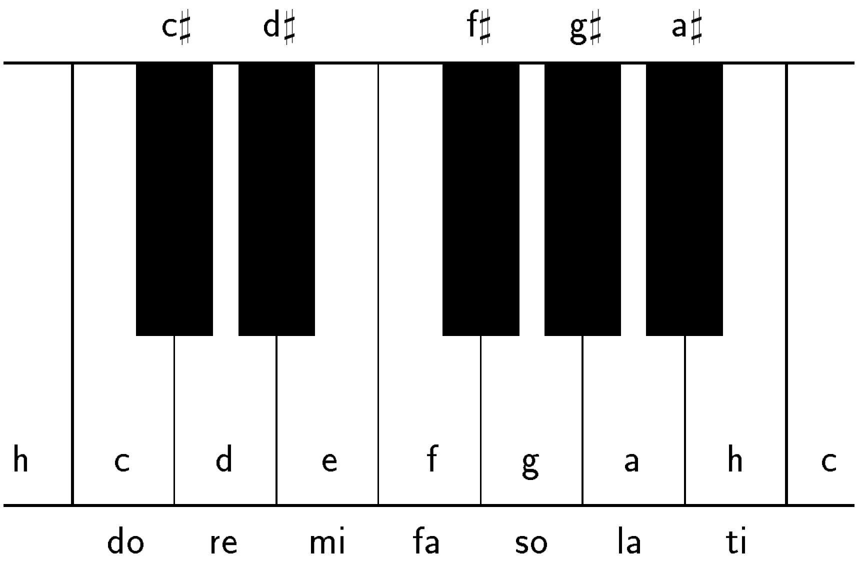
```
TONI 6000 1000 -25 1  BEEP 50 10
```

6.3 Glasba

V glasbi praviloma uporabljamo le tone izbranih višin, ki sestavljajo *tonski sistem*. Za označevanje osnovnih tonov uporabljamo v evropski glasbi običajno črke abecede *c, d, e, f, g, a, h* (angleži uporabljajo namesto črke *h* črko *b*) ali pa solmizacijske zloge *do, re, mi, fa, so (sol), la, ti (si)*. V tonskem sistemu se zaporedje osnovnih tonov večkrat ponovi. Del tonskega sistema med zaporednima ponovitvama oznake tona imenujemo *oktava* (glej sliko 6.1). Glede na osnovno *malo* oktavo označujemo tone v nižjih oktavah z velikimi črkami in odmikom, npr. D_2; tone v višjih oktavah pa z malo črko in odmikom, npr a^1.

Razmiki med toni so večji ali manjši. Manjšemu pravimo *polton*, večjemu pa *celi ton*. Dva poltona sestavljata celi ton. Z višajem $\sharp$ označujemo zvišanje tona za polton, z nižajem $\flat$ pa znižanje tona za polton. Podrobnosti si lahko preberete v učbeniku [21].

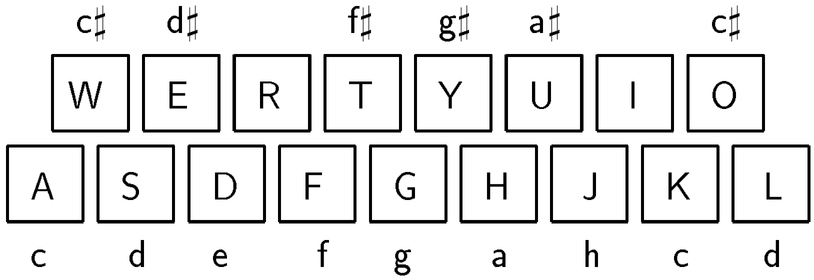
V računalništvu navadno uporabljamo c-durovo lestvico in tonski sistem, ki temelji na komornem tonu $a^1 = 440$ Hz. V tem sistemu sestavljajo toni geometrijsko zaporedje. Razmerje med višinama zaporednih tonov je enako $\sqrt[12]{2} = 1.05946$. To zagotavlja, da ima posamezni ton v nižji (višji) oktavi polovično (dvojno) višino.



Slika 6.1: Oktava

ton	O	o	o^1	o^2	o^3
c	65	131	262	523	1046
$c\sharp(d\flat)$	69	139	277	554	1109
d	73	147	294	587	1175
$d\sharp(e\flat)$	78	156	311	622	1244
e	82	165	330	659	1318
f	87	175	349	698	1397
$f\sharp(g\flat)$	92	185	370	740	1480
g	98	196	392	784	1568
$g\sharp(a\flat)$	104	208	415	831	1661
a	110	220	440	880	1760
$a\sharp(h\flat)$	117	233	466	932	1865
h	123	247	494	988	1976

Tabela 6.1: Tonski sistem



Slika 6.2: Tipkovnica – klavir

6.4 Tipkovnica – klavir

Za primer sestavimo ukaz `KLAVIR`, ki tipkovnico pretvori v klaviaturo, tako kot je prikazano na sliki 6.2. Izvajanje ukaza prekinemo s pritiskom na piko.

```
TO KLAVIR
  TS
  MAKE "s [262 2 3 330 311 349 392 440 9 494 523
  587 13 14 554 16 17 18 294 370 466 22 277 24 415 ]
  REPEAT 10000 [
    MAKE "c LC RC
    IF EQUALP :c ". [ STOP ] []
    IF MEMBERP :c "asdfghjklwetyuo
      [ MAKE "t ITEM ((ASCII :c) - 96) :s BEEP :t 20 ]
      [ ]
  ]
END
```

6.5 Skladbe

V programskih jezikih so navadno toni izbranega sistema oštevilčeni. Na primer takole:

...	$A\sharp$	H	c	$c\sharp$	d	...	h	c^1	$c^1\sharp$	...
...	-2	-1	0	1	2	...	11	12	13	...

Namesto ukaza BEEP tedaj uporabljamo ukaz oblike NOTA n t , ki zaigra ton s številko n in trajanjem t.

Pogosto želimo zaigrati več tonov z enakim trajanjem. To krajše zapišemo z ukazom IGRAJ

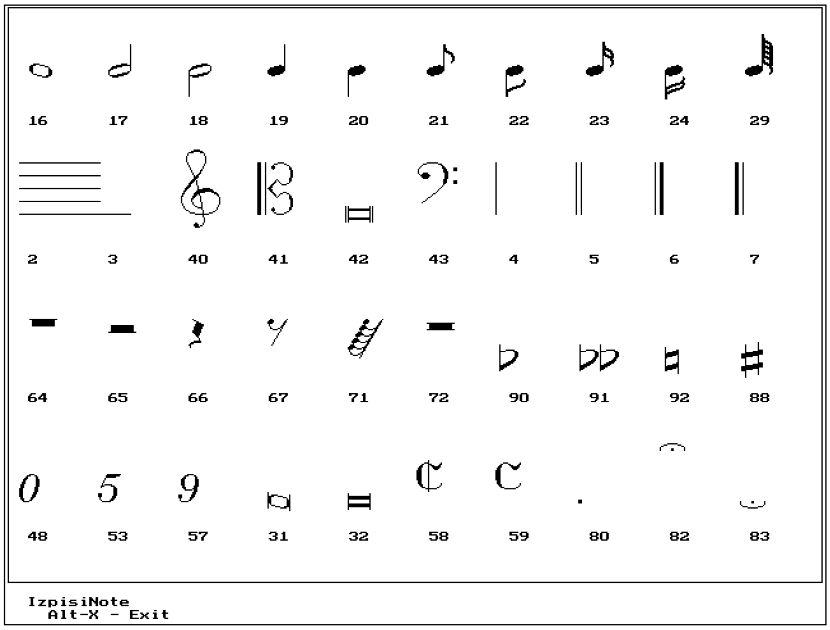
```
TO IGRAJ :s :t
  IF EMPTYP :s [ STOP ] []
  NOTA FIRST :s :t  IGRAJ BF :s :t
END
```

ki zaigra tone iz danega seznama s s trajanjem t. S tem ukazom lahko zapišemo skladbo *Čukova ženitev* (glej [25]) takole:

```
TO CUK :t
  IGRAJ [ 0 2 4 5 ] :t  IGRAJ [ 7 7 9 9 ] 2*:t
  IGRAJ [ 7 ] 4*:t  IGRAJ [ 9 9 ] 2*:t
  REPEAT 2 [ IGRAJ [ 7 ] 4*:t
    IGRAJ [ 5 5 5 5 ] :t IGRAJ [ 4 4 2 2 ] 2*:t ]
  IGRAJ [ 0 ] 4*:t
END
```

V naslednjih ukazih je pokazano, kako lahko sami sprogramiramo ukaz NOTA z uporabo ukaza BEEP.

```
TO VOS :n :os
  IF :n = 0 [ OP :os ] []
  IF :n > 12 [ OP VOS (:n - 12) :os*2 ]
    [ IF :n < 0 [ OP VOS (:n + 12) :os/2 ]
      [ OP VOS (:n - 1) :os*1.05946 ] ]
END
TO VISINA :n
  OP ROUND VOS :n 110
END
TO NOTA :n :t
  BEEP ITEM 13+:n :note :t
END
TO TEST
  MAKE "note [] MAKE "k -9
  REPEAT 48 [ MAKE "note
    LPUT VISINA :k :note MAKE "k :k + 1]
  TS PR :note  CUK 10  WAIT 25  CUK 15
END
```



Slika 6.3: Notna pisava

Ukaz `VISINA` določi višino tona s številko n. Uporabimo ga v ukazu `TEST`. Ta najprej ustvari seznam *note*, ki vsebuje višine tonov s številkami od -12 do 35. Iz tega seznama dobi ukaz `NOTA` podatek o višini tona. Nato ukaz `TEST` še izpiše seznam *note* in dvakrat zaigra skladbo *Čukova ženitev*; drugič nekoliko počasneje.

6.6 Izpis not v LogoSu

Ker je treba zapolniti še eno stran, da pride novo poglavje na liho stran, pokažimo, kako lahko note izpišemo v LogoSu. Ta možnost je dostopna tudi v LCSI/LogoSu.

Nabor znakov z notno pisavo je pripravljen na datoteki `NOTE.CHR` in ima vlogo uporabniškega nabora znakov (`"UserFont`). Vgrajeni LogoSovi nabori znakov pa so `"DefaultFont`, `"TriplexFont`, `SmallFont`, `"GothicFont`, `"SansSerifFont`, `"ScriptFont` in `"LComFont`.

Ukaz `PrintNote` izpiše na tekoče mesto k-ti znak iz notne pisave in ga opremi z njegovo številko. Uporabljen je v ukazu `PrintNotes`, ki izpiše znake določene z danim seznamom. Izpis znakov na sliki 6.3 je dobljen z ukazom `IzpisiNote`.

```
TO PrintNote :k
  (LOCAL "v "s)
  MAKE "v DIV :n 10 MAKE "s REMAINDER :n 10
  SetTextFont "UserFont SETCUR (LIST 9*:s+2 10.5*:v+5)
  TYPE CHAR :k SETCUR (LIST 2.4*:s+1.5 2.1*:v+1.8)
  SetTextFont "DefaultFont TYPE :k MAKE "n :n + 1
END
TO PrintNotes :s
  IF EMPTYP :s [ STOP ] [ ]
  PrintNote FIRST :s  PrintNotes BF :s
END
TO IzpisiNote
  CS HT
  SetTextUserFont "note  SetTextFont "UserFont
  SetTextSize 1 1  SetTextColor Yellow
  MAKE "n 0
  PrintNotes [ 16 17 18 19 20 21 22 23 24 29 ]
  PrintNotes [  2  3 40 41 42 43  4  5  6  7 ]
  PrintNotes [ 64 65 66 67 71 72 90 91 92 88 ]
  PrintNotes [ 48 53 57 31 32 58 59 80 82 83 ]
  WAIT 0
END
```

6.7 Naloga: Zvok

Sestavi zaporedje ukazov v logu, ki zaigra melodijo, ki si si jo izbral (na primer *Naša četica koraka*), ali pa ustvari kak zvočni učinek (na primer tiktakanje ure).