2.

Ukazi, koordinate in izpis

Ukaze v logu delimo na *prave ukaze* in *funkcije*. Pravi ukaz opravi neko opravilo. Funkcijski ukaz pa vrne neko vrednost, ki mora biti podatek nekemu pravemu ukazu.

Na primer ukazna vrstica

```
PRINT SUM 3 4
```

zahteva izpis vsote števil 3 in 4. Pri tem je rezultat funkcijskega ukaza `SUM` za podatka 3 in 4 podatek za pravi ukaz `PRINT`.

Poskusite vnesti še ukaz

```
SUM 3 4
```

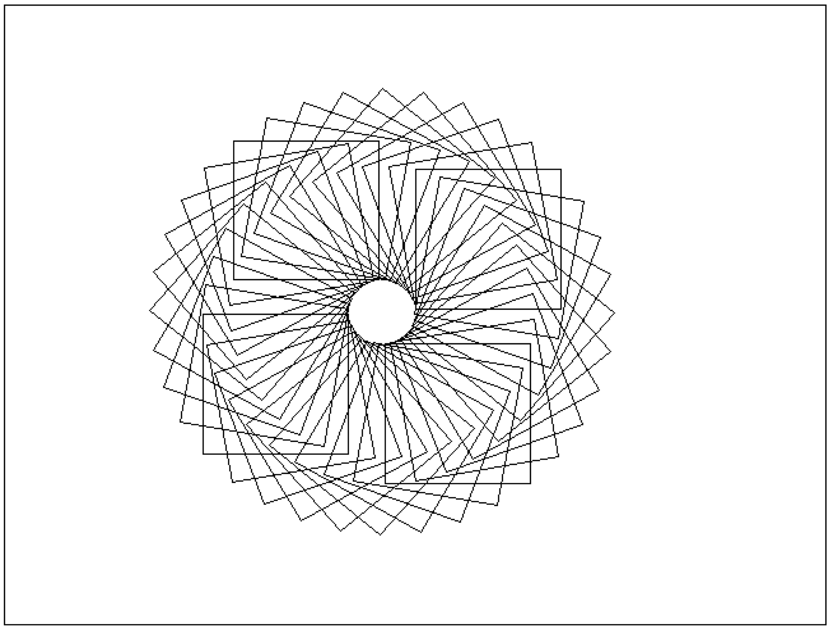
Kaj se bo zgodilo, je odvisno od tolmača za logo. Najverjetneje se bo tolmač pritožil, da dobljeni rezultat ni podatek za pravi ukaz; nekateri tolmači pa rezultat brez pripomb izpišejo.

Ukaze lahko razdelimo še na *vgrajene* ali *praukaze* in *uporabnikove* ali *lastne* ukaze. Vgrajene ukaze pozna logo že ob zagonu. Uporabnikove ukaze pa mora sestaviti uporabnik sam. Sestavljanju teh ukazov pravimo tudi *programiranje*.

2.1 Lastni ukazi

V običajnem, sprotnem, pogovornem načinu dela uvedemo lastni ukaz z ukazom `TO`. V večini narečij loga zahtevamo

`TO` *ime*



Slika 2.1: Roža

Na to zahtevo preide tolmač za logo v stanje, v katerem sprejema vrstice z opisom ukaza *ime*. Vnos zaključimo z vrstico, ki vsebuje ukaz `END`.

Podobno uvedemo nov ukaz, če ni predolg, tudi v narečju LOGOS, le da vse skupaj vnesemo kot eno vrstico

`TO` *ime opis* `END`

Tako uvedemo ukaz `KVADRAT` z

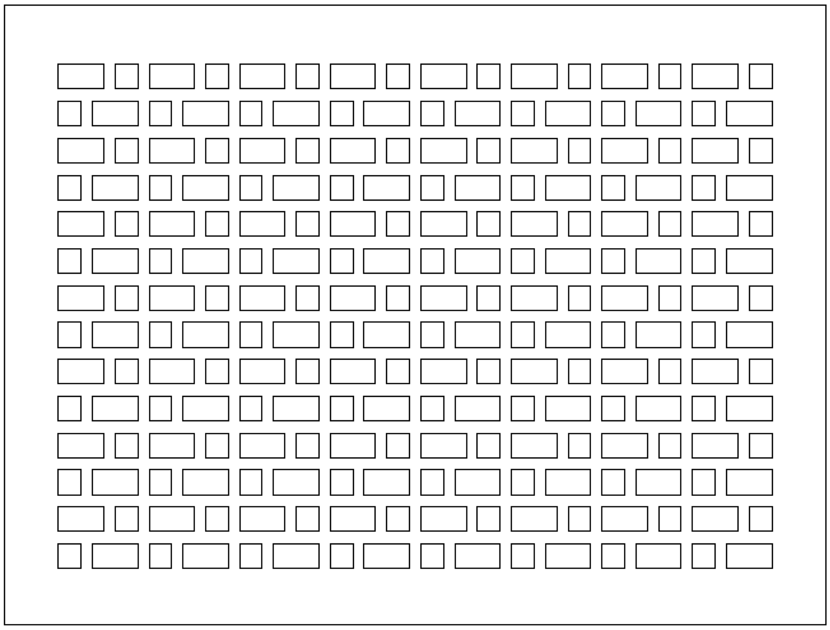
```
TO KVADRAT
  REPEAT 4 [ FD 100 RT 90 ]
END
```

in ga poženemo

```
CS KVADRAT
```

Ko je ukaz enkrat uveden, ga lahko uporabljamo enako kot praukaze. Na primer

```
CS REPEAT 36 [ PU FD 4 PD KVADRAT LT 10 ]
```



Slika 2.2: Zid

ki nariše "rožo" prikazano na sliki 2.1.

Poglejmo si še nekoliko zapletenejši primer, kjer kamne in opeke zlagamo v vrste, te pa nalagamo ene na druge in tako ustvarimo zid. Risanje zida zahtevamo z ukazom `SLIKA`.

```
TO KAMEN
  REPEAT 4 [ FD 16 RT 90 ]
END
TO OPEKA
  REPEAT 2 [ FD 32 RT 90 FD 16 RT 90 ]
END
TO VRSTA
  REPEAT 8 [ PU BK 24 PD KAMEN PU BK 40 PD OPEKA ]
END
TO ZID
  LT 90 REPEAT 7 [ VRSTA RT 90 PU FD 40 RT 90 FD 8
    VRSTA LT 90 PU FD 8 LT 90 FD 8 ]
END
TO SLIKA
  CS PU BK 165 LT 90 PU FD 260 RT 90 PD ZID
```

```
END
```

Zid je prikazan na sliki 2.2.

2.2 Delo z lastnimi ukazi

Imena vseh uporabniških ukazov izpiše na zaslon ukaz `POTS`. Ukaz *ime* izločimo (zbrišemo) z ukazom

`ERASE "`*ime*

Uporabimo lahko tudi okrajšavo `ER`.

Če želimo ukaz *ime* popravljati zahtevamo

`EDIT "`*ime*

Ta ukaz deluje na posameznih vrstah računalnikov zelo različno. Zato si podrobnosti preberite v priročniku ali pa, za Spectrum, v [4, 6] in, za C-64, v [5, 13].

V LCSI/LogoSu lahko v sprotnem načinu spremenimo opis uporabniškega ukaza tako, da njegovo novo definicijo začnemo z geslom `RETO`. Na primer, če želimo, da ima kvadrat dolžino stranice 120 namesto 100, bomo zahtevali

```
RETO KVADRAT REPEAT 4 [ FD 120 RT 90 ] END
```

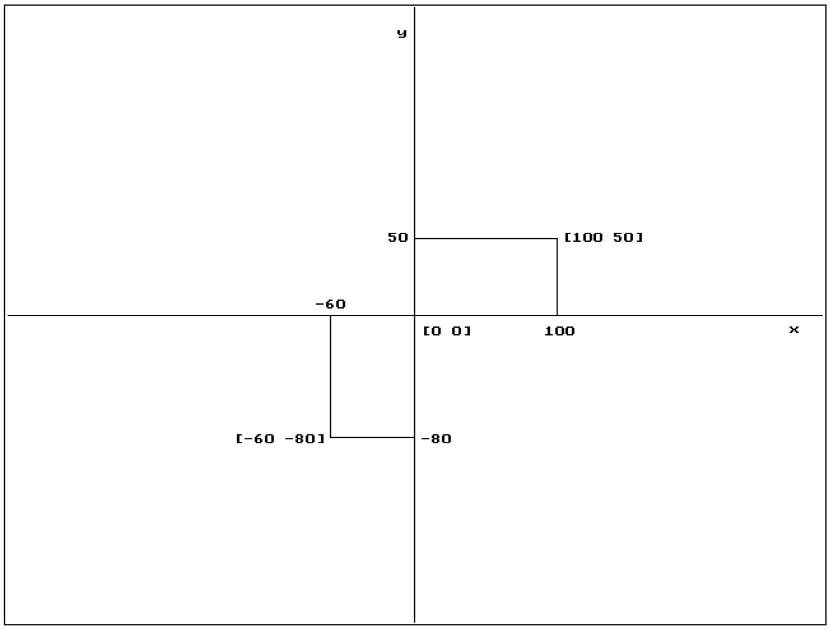
Za pisanje obsežnejših programov pa raje uporabimo urejevalnik, v katerega pridemo z ukazom `EDITOR`.

Večina narečij loga pozna še naslednje (ali njim podobne) ukaze:

`PO` *i*	izpiši definicijo ukaza *i*
`SAVE` *d s*	shrani ukaze iz seznama *s* na datoteko *d*
`SAVEALL` *d*	shrani vsebino delovnega prostora na datoteko *d*
`LOAD` *d*	vnesi ukaze z datoteke *d*
`CATALOG` *b*	izpiži seznam datotek na tekočem področju, katerih imena zadoščajo vzorcu *b*
`POFILE` *d*	izpiši vsebino datoteke *d*
`ERALL`	zbriši vsebino delovnega prostora

Najbrž ni potrebno posebej razlagati, kdaj in kako jih lahko uporabite.

Posamezna narečja loga imajo različne vgrajene podaljške za datoteke, ki vsebujejo ukaze v logu. Najpogosteje se uporablja podaljšek `.LOG`, s katerim pa nekaj programov označuje datoteke z dnevnikom obdelave. LCSI logo na PCjih uporablja podaljšek `.LF`, LCSI/LogoS pa podaljšek `.LG`.



Slika 2.3: Koordinatni sistem

2.3 Koordinate

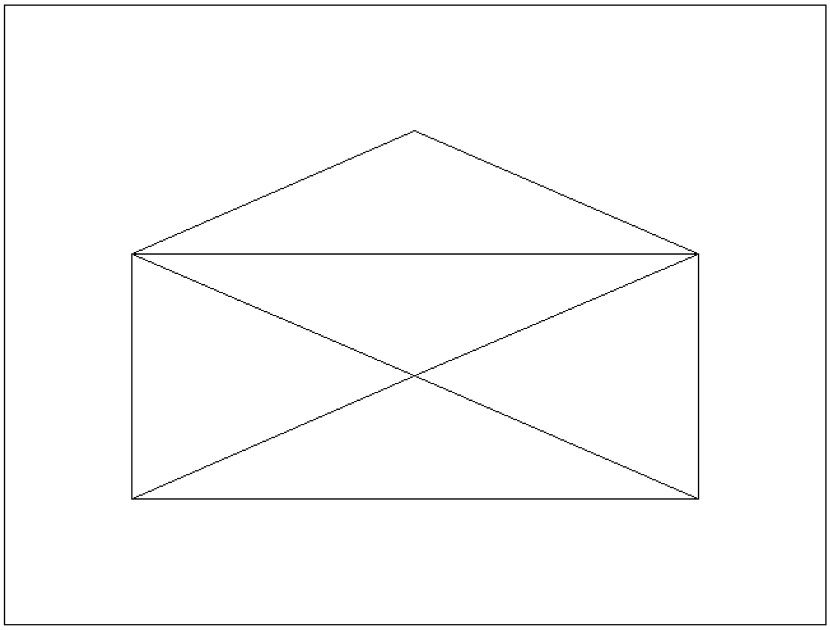
Na slikovni zaslon lahko postavimo tudi običajni koordinatni sistem, kakor smo ga navajeni v matematiki (glej sliko 2.3). Vsaka točka na zaslonu je določena z dvema številoma (x, y) – *koordinatama*, ki merita njen odmik od osi x in osi y. V logu točko (x, y) predstavimo s seznamom $[x\ y]$.

Želvo lahko usmerjamo tudi glede na ta koordinatni sistem. Na voljo imamo ukaze:

`SETPOS` $[x\ y]$	premik želve v točko (x, y);
`SETX` n	premik želve do koordinate $x = n$;
`SETY` n	premik želve do koordinate $y = n$;
`TOWARDS` $[x\ y]$	vrne smer želve proti točki (x, y);
`SETHEADING` n	postavi smer želve na n; okrajšava `SETH`;
`DOT` $[x\ y]$	nariše piko na (x, y), želvo pusti tam, kjer je.

Prav bosta včasih prišla tudi ukaza:

`SETBG` n	postavi barvo podlage na n;
`PENREVERSE`	odslej želva riše, tam kjer ni sledi, in briše (dele) sledi, čez katere gre; okrajšava `PX`.



Slika 2.4: Pismo

Na voljo imamo tudi nekaj funkcijskih ukazov, ki vrnejo posamezne zanimive količine:

`POSITION`	položaj (mesto) želve; okrajšava `POS`;
`XCOR`	koordinata x (abscisa) želve;
`YCOR`	koordinata y (ordinata) želve;
`HEADING`	smer želve;
`PENCOLOR`	barva sledi; okrajšava `PC`;
`BACKGROUND`	barva podlage; okrajšava `BG`.

Ti ukazi bodo prišli prav tudi pri reševanju naslednje naloge.

2.4 Naloga: Pismo

Sestavi zaporedje ukazov v logu, ki na zaslonu nariše pismo (glej sliko 2.4) v eni potezi – ne da bi dvignili pero – pri čemer nobene črte ne riše dvakrat.

2.5 Besede in izpis

Znake, ki jih najdemo na tipkovnici računalnika delimo na tri skupine: velike in male *črke* `A`, `B`, `C`, ..., `Z`, `a`, `b`, `c`, ..., `z`; *števke* ali *cifre* `0`, `1`, `2`, ..., `9` in *posebne znake* `+`, `-`, `*`, `(`, `:`, `?`, ...

Iz znakov lahko sestavljamo nize znakov ali besede. *Besedo* začnemo z dvojnim narekovajem `"` . Ta ne pripada besedi, temveč le pove, da zaporedje znakov, ki mu sledi, sestavlja besedo. V besedi ne smemo naravnost uporabiti presledka in znakov `[]()+-*/<=>` . Če jih vseeno želimo uporabiti, moramo pred vsak tak znak zapisati znak `\`. Primeri besed so: `"Ljubljana`, `"!!napaka` in `"2\+2` . Logo pozna tudi prazno besedo `"`, ki ne vsebuje nobenega znaka.

Količino p lahko izpišemo na zaslon z ukazi:

`PRINT` p	izpiše podatek p in gre v novo vrstico;
`TYPE` p	samo izpiše podatek p;
`SHOW` p	izpiše podatek p, če je seznam tudi z zunanjimi oklepaji, in gre v novo vrstico.

Če želimo naenkrat izpisati več podatkov $p_1,\ p_2,\ \ldots\ p_n$, zahtevamo

(`PRINT` $p_1\ p_2\ \ldots\ p_n$)

Namesto `PRINT` lahko uporabimo tudi okrajšavo `PR`.

Za izpis podatkov navadno raje uporabimo *znakovni zaslon*, na katerega preidemo s slikovnega z ukazom `TEXTSCREEN` ali `TS`, pobrišemo pa ga z ukazom `CLEARTEXT` ali `CT`.

Poskusite

```
TS CT PR "Ljubljana
TYPE 26 TYPE "marec
PR 1991 SHOW "Konec
```

V narečju LCSI/LOGOS lahko določimo število decimalnih mest pri izpisu števil z ukazom `SETNUMPREC` n. Torej, če delamo s celimi števili, pred izpisom postavimo `SETNUMPREC 0`.

2.6 Spremenljivke

Računalnik si lahko posamezne količine tudi zapomni. Shrani jih v svoj *pomnilnik*. Uporabnik si lahko pomnilnik predstavlja kot omaro s predalčki. Posamezni predalček lahko: opremimo z imenom; vanj shranimo količino; po potrebi pogledamo, kaj je v njem shranjeno; ali njegovo vsebino zamenjamo z novo.

V programskih jezikih pravimo takim predalčkom spremenljivke. *Spremenljivko* sestavljajo: ime, tip, prostor in vrednost.

Če hočemo v programu s spremenljivko kaj početi, uporabljamo za to njeno *ime*. Ime spremenljivke prevajalnik ali tolmač poveže z njej pripadajočim *prostorom* v pomnilniku. Vsebina tega prostora je *vrednost* spremenljivke. *Tip* spremenljivke pa določa: velikost spremenljivki pripadajočega prostora; vrednosti, ki jih lahko vsebuje; kako naj prevajalnik razume vsebino prostora; in kaj lahko s spremenljivko počnemo. Kot smo že omenili, pozna logo tri tipe: števila, besede in sezname.

V logu je ime spremenljivke beseda. Na primer: `"PETER`, `"NOVA.CENA`. Do vsebine spremenljivke `"ime` pridemo z ukazom `THING "ime`, kar lahko krajše zapišemo `:ime`. Tako na primer namesto `THING "DATUM` raje pišemo `:DATUM` .

Količino *vrednost* shranimo v spremenljivko *ime* z ukazom

`MAKE "`*ime vrednost*

Na primer

```
MAKE "DAN 23 MAKE "MESEC "marec MAKE "LETO 1991
(PR :DAN :MESEC :LETO)
```

V programskih jezikih, ki temelje na prevajanju, lahko spremenljivka vsebuje le vrednosti enega tipa; v logu pa lahko vanjo shranimo katero koli vrednost. Na primer, brez ugovora bo sedaj sprejel tudi zahtevo

```
MAKE "LETO "MCMXCI SHOW :LETO
```

Poskusite še

```
MAKE "A "KONEC MAKE "B "A SHOW :B SHOW THING :B
```

2.7 Ukazi s parametri

Kako sestavimo ukaz za risanje kvadrata, smo se že naučili. Če želimo risati kvadrate s spremenljivo velikostjo stranice, moramo ukaz opremiti s podatkom, ki mu bo to povedal. Tudi v opisu ukaza moramo povedati, da je ukaz odvisen od podatka. V ta namen vpeljemo posebno sestavino – *parametre*. Ime parametra ima obliko vrednosti spremenljivke – sestavlja ga dvopičje, ki mu sledi zaporedje znakov. Uvedba ukaza s parametri *ime* ima obliko

TO *ime parametri*
  opis
END

Kadar želimo v opisu ukaza uporabiti vrednost kakega parametra, zapišemo njegovo ime. Pri uporabi ali, kakor pravimo v računalništvu, *klicu* ukaza navedemo njegovo ime in za njim zaporedje vrednosti parametrov.

Recimo, da označuje dolžino stranice kvadrata parameter `:a` . Tedaj lahko sestavimo ukaz

```
TO KVA :a
  REPEAT 4 [ FD :a RT 90 ]
END
```

ki ga preizkusimo z zahtevo

```
CS KVA 30 KVA 50 KVA 70 KVA 100
```

Uporabniški ukazi so zelo pomembna sestavina vsakega programskega jezika, saj, če jih pravilno uporabljamo:

- povečajo preglednost in razumljivost programov;
- približajo programski jezik področju uporabe;
- omogočajo pisanje krajših programov;
- strokovnjaki lahko pripravijo *knjižnice* ukazov. Tako lahko tudi nestrokovnjaki uporabljajo zahtevna "orodja".

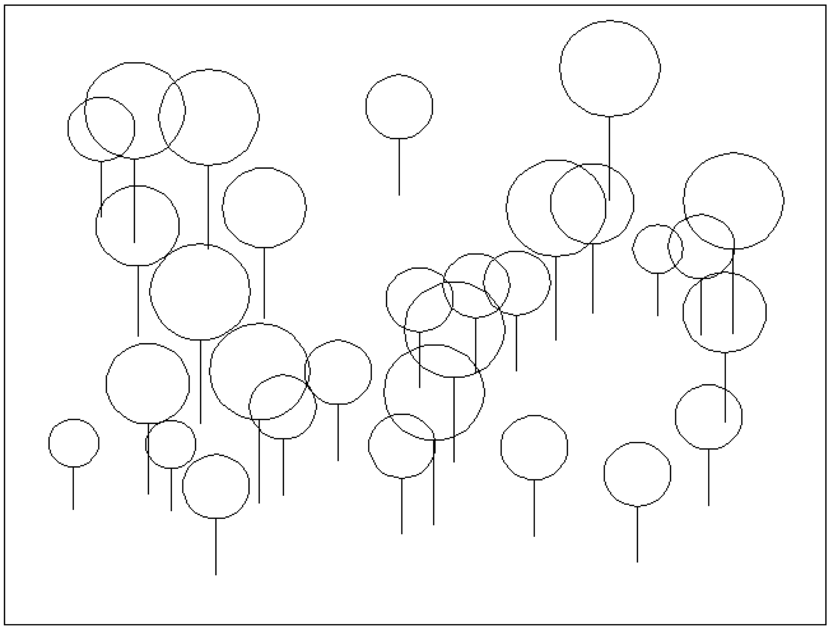
Za nas je pomembna še ena lastnost: ker lahko za imena uporabljamo slovenske besede, lahko logo "naučimo" slovenščine. Na primer:

```
TO NAPREJ :k
  FD :k
END
TO DESNO :k
  RT :k
END
```

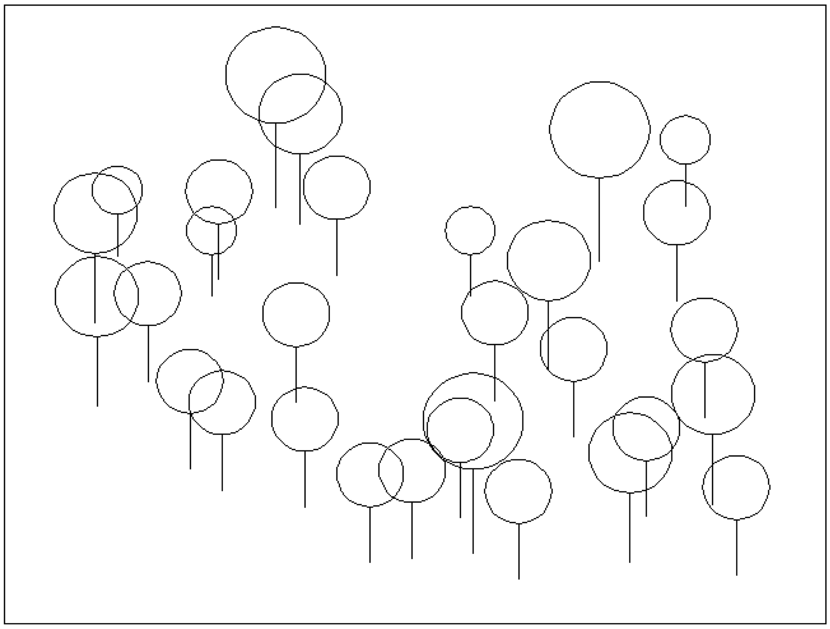
Sedaj bo logo razumel tudi zahteve

```
NAPREJ 100 DESNO 90 NAPREJ 50
```

Za nekoliko zahtevnejši primer narišimo gozd. Najprej drevo s koreninami v točki t in korakom velikosti a



Slika 2.5: Gozd



Slika 2.6: Še en gozd

```
TO DREVO :t :a
  SETH 0 PU SETPOS :t PD
  REPEAT 10 [FD :a] LT 85
  REPEAT 36 [FD :a RT 10]
END
```

več dreves sestavlja gozd

```
DREVO [-90 -50] 5 DREVO [10 30] 3 DREVO [110 -30] 4
```

Na slikah 2.5 in 2.6 je prikazan gozd iz 30 naključno velikih in postavljenih dreves.

2.8 Znakovni zaslon

Kot smo že povedali, poleg *slikovnega* zaslona, na katerem rišemo slike, pozna logo še *znakovni* zaslon, ki je namenjen za izpis besedil. Nanj nas postavi ukaz `TEXTSCREEN` ali krajše `TS`. Na znakovni zaslon pišemo z ukazi `PRINT`, `SHOW` in `TYPE`.

Znakovni zaslon je razdeljen na enako velika pravokotna "okenca". V vsako tako okence lahko napišemo po en znak. Zaporedna okenca v vodoravni smeri sestavljajo *vrstice*, v navpični pa *stolpce*. Vrstice in stolpci so oštevilčeni. Šteti začnemo v levem zgornjem vogalu zaslona. Na računalniku PC ima znakovni zaslon navadno 25 vrstic in 80 stolpcev. Koliko vrstic in stolpcev je na vašem zaslonu?

Z ukazom

`SETCURSOR [` s v `]`

se postavimo na okence v s-tem stolpcu in v-ti vrstici. Ime ukaza lahko okrajšamo s `SETCUR`. Kje je tekoče okence izvemo z ukazom `CURSOR`, ki vrne seznam `[` s v `]`. Tako na primer

```
SETCUR [14 11] TYPE "Logo SHOW CURSOR SHOW CURSOR
```

izpiše:

```
Logo[18 11][25 11]
```

Če imamo na voljo barvni zaslon, lahko z ukazom

`SETTC [` c p `]`

zahtevamo, da naj bodo odslej izpisani znaki barve *c* (črnilo), pripadajoča okenca pa barve *p* (podlaga, papir). Barve so oštevilčene. Obe barvi podamo tako, da napišemo njuni številki. Na posameznih računalnikih (in celo zaslonih) so na voljo različni nabori barv. Seznam, ki vsebuje tekoči barvi papirja in črnila, vrne ukaz `TEXTCOLOR` ali `TC`.

Za barvo papirja velja na PC samo prvih 8 barv, nadaljnje vrednosti povzročijo utripajoči izpis. Z ukazom `SETTC` lahko na PC tudi na črno-belem zaslonu dosežemo posamezne učinke (utripanje, jakost izpisa, inverzni izpis, podčrtanje) ... Poskusite

```
TS CT
SETTC [  0 1 ] PR "XXXXX00000
SETTC [  0 7 ] PR "XXXXX00000
SETTC [  1 0 ] PR "XXXXX00000
SETTC [  8 5 ] PR "XXXXX00000
SETTC [  9 0 ] PR "XXXXX00000
SETTC [ 24 3 ] PR "XXXXX00000
SETTC [ 16 7 ] PR "XXXXX00000
```

3.

Računanje in seznami

3.1 Računanje

V logu lahko tudi računamo. Za začetek se bomo zadovoljili z osnovnimi računskimi operacijami: seštevanjem $+$, odštevanjem $-$, množenjem $*$ in deljenjem /. Poskusite

```
TS CT
PR 3 + 4
PR 17 - 8
PR 8 * 7
PR 7 / 3
```

zaporedni odgovori so 7, 9, 56, 2.333333 . Uporabimo lahko tudi zapletenejše izraze. Vrstni red izračuna določimo z oklepaji. Tako

```
PR (5+6)*(12-7)
```

izračuna in izpiše vrednost 55.

Če vrstni red ni določen z oklepaji, ga določata naslednja dogovora:

- *prednostni vrstni red*: tako kot je v navadi v matematiki, množenje in deljenje močneje vežeta kot seštevanje in odštevanje;
- *izračun z leve proti desni*: operacije, ki vežejo enako močno, se opravljajo z leve proti desni.

Tako je zahteva

```
PR 3 - 4 * 5 + 7
```

enakovredna zahtevi

```
PR (3 - (4 * 5)) + 7
```

Zakaj ? Preizkusite !

V izrazih lahko uporabljamo tudi spremenljivke in izračunano vrednost shranimo v neko spremenljivko z ukazi oblike

MAKE "*i izraz*

ki zahteva, da se izračuna vrednost danega izraza in nato shrani v spremenljivko i. Enak učinek ima tudi ukaz

NAME *izraz* "*i*

Poglejmo, kaj opravita ukaza:

```
MAKE "k 7
MAKE "k :k + 1
```

Prvi ukaz shrani v spremenljivko k vrednost 7. Drugi ukaz zahteva, da se najprej izračuna vrednost izraza `:k + 1` in nato shrani v spremenljivko k. Izraz `:k + 1` zahteva, da vzamemo vrednost spremenljivke k, to je 7, in ji prištejemo 1. Dobimo 8. To vrednost shranimo nazaj v spremenljivko k kot njeno novo vrednost. Torej, če posplošimo, je pomen drugega ukaza: povečaj vrednost spremenljivke k za 1.

Poizkusite še

```
MAKE "a "le MAKE "b "ska TYPE :a PR :b
MAKE "c :a MAKE "a :b MAKE "b :c TYPE :a PR :b
```

Vidimo, da stavki `MAKE` med stavkoma `PR` in `TYPE` premenjajo vrednosti spremenljivk a in b. Spremenljivko c smo uporabili za začasno hranjenje začetne vrednosti spremenljivke a.

Kaj pa izpišejo ukazi:

```
TS CT MAKE "a 19 MAKE "b 91 (PR :a :b)
MAKE "a :a + :b MAKE "b :a - :b MAKE "a :a - :b
(PR :a :b)
```

in

```
MAKE "a 1 MAKE "b 1 PR :a PR :b
REPEAT 10 [ MAKE "a :a + :b MAKE "b :a + :b PR :a PR :b ]
```

Za zahtevnejše bralce bo najbrž poučen tudi naslednji zgled:

```
TO povecaj :stevec
  MAKE :stevec (THING :stevec) + 1
END

MAKE "n 0   (PR [:n =] :n)
povecaj "n  (PR [:n =] :n)
```

3.2 Seznami

Če so količine $a_1 a_2 \ldots a_n$ znane, dobimo *seznam* tako, da zaporedje količin obdamo z oglatima oklepajema `[` in `]`. Količine, ki nastopajo v seznamu, so lahko tudi same seznami. Na primer:

```
[ 1 2 3 ]
[ [ 1 ] 4 [ 2 3 ] "aha x ]
```

Seznamu `[ ]`, ki vsebuje prazno zaporedje, pravimo *prazni seznam.* S seznami lahko opišemo najrazličnejše vrste podatkov.

Kadar količine dobimo kot vrednosti spremenljivk ali kot rezultat izračuna, jih združimo v seznam z ukazom `LIST`

(`LIST` a_1 $a_2 \ldots a_n$)

Ukazu `LIST` je soroden ukaz `SENTENCE`, ali krajše `SE`, oblike

(`SE` s_1 $s_2 \ldots s_n$)

ki elemente seznamov $s_1, s_2, ..., s_n$ združi v skupni seznam. Na primer, naj bo

```
MAKE "p [1 2] MAKE "q [3 4]
```

Tedaj ukaz

```
SHOW (LIST :p :q)
```

izpiše seznam `[[1 2][3 4]]`; ukaz

```
SHOW (SE :p :q)
```

pa seznam `[1 2 3 4]`. Podobno ukaz

(`WORD` b_1 $b_2 \ldots b_n$)

združi dane besede v eno besedo.

Poglejmo si še zgled, ki združuje več sestavin, ki smo jih spoznali v tem razdelku. Ukazi

```
TS CT MAKE "v 1
REPEAT 20 [
  SETCUR (LIST 2*:v :v) TYPE :v MAKE "v :v + 1 ]
```

izpišejo "diagonalno" po znakovnem zaslonu prvih 20 naravnih števil.

Tako, sedaj pa že poznate vse, kar je potrebno za rešitev naslednje naloge.

3.3 Naloga: Poštevanka do 100

Sestavi zaporedje ukazov v logu, ki na znakovnem zaslonu izpiše poštevanko do 100.

	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	12	14	16	18	20
3	3	6	9	12	15	18	21	24	27	30
4	4	8	12	16	20	24	28	32	36	40
5	5	10	15	20	25	30	35	40	45	50
6	6	12	18	24	30	36	42	48	54	60
7	7	14	21	28	35	42	49	56	63	70
8	8	16	24	32	40	48	56	64	72	80
9	9	18	27	36	45	54	63	72	81	90
10	10	20	30	40	50	60	70	80	90	100