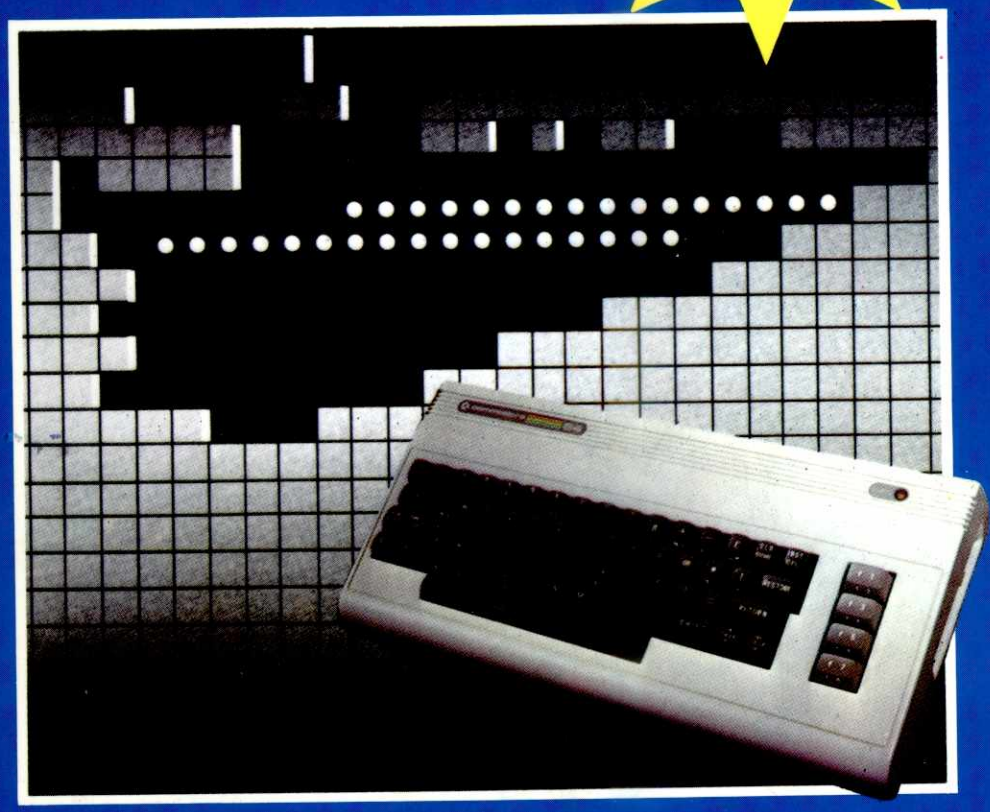
Commodore 64

Programiranje na lak način

Ian Stewart
Robin Jones



I. Stewart-R. Jones: Commodore 64 ◇ Programiranje na lak način

Ian Stewart — Robin Jones

Commodore 64

Programiranje na lak način

NIRO „Tehnička knjiga“ i Zavod za udžbenike i nastavna sredstva
— Beograd, 1986.

Naslov originala:

Easy Programming for the Commodore 64

Ian Stewart and Robin Jones

SHIVA PUBLISHING LIMITED
4 Church Lane, Nantwich, Cheshire CW5 5RQ, England

Prvi put objavljeno u Velikoj Britaniji 1983. g.

Ponovljena izdanja 1983, 1984 (dva puta)

Izdavači za SFRJ: NIRO "Tehnička knjiga", Beograd, 7. juli 26 i Zavod za udžbenike i nastavna sredstva, Beograd, Obilićev venac 5

Recenzent: dr Dušan Tošić

Prevod i stručna redakcija teksta: Adem Jakupović

Stručna redakcija teksta i adaptacija programa: mr Veselin Petrović

Za izdavača: Radomir Krivokapić, v. d. direktora i prof. dr Velimir Branković

Urednici: Radivoje Grbović i Ninoslav Čabrić

Grafički urednik: Jugoslav Bogdanović

Korektor: Mirjana Aćimović

Tiraž: 4.000 primeraka

Oslobođeno poreza na promet na osnovu mišljenja Republičkog komiteta za kulturu SRS

Sadržaj

Uvod

Ova knjiga namenjena je onima koji se prvi put sreću sa računarima kupivši ili nameravajući da kupe mikroračunar "Commodore 64". Ona bi se mogla pokazati korisnom i za iskusnije korisnike koji na "Commodore 64" prelaze sa drugog računara. "Commodore 64" je izuzetno pametna mašina sa mnogim mogućnostima kao što je: integrisano kolo za zvuk, grafika sa sprajtovima i velike mogućnosti za proširenje sistema. Cilj ove knjige je srazmerno skroman: da razumljivim rečima opiše glavne karakteristike mašine i njen programski jezik BASIC.

Kad kupite "Commodore 64", besplatno dobijate *Priručnik*[1]. Problem sa priručnicima je u tome što oni retko imaju dovoljno prostora da se bave detaljnim primerima. U stvari, pravi izvor informacija za "šezdesetčetvorku" nije Priručnik nego veliki *Vodič*[2] sa 486 strana koji uključuje sve do konstrukcije elektronskih i rasporeda nožica integrisanih kola. On je izvrstan za iskusnog programera, ali će masa sažetih informacija i žargon zbuniti početnika. Ova knjiga ima za cilj da premosti jaz između *Priručnika* i *Vodiča*.

U knjizi je dat temeljan, ali razumljiv opis osnova BASIC-a, standardnog višeg programskog jezika za "šezdesetčetvorku". Poglavlja su "skrojena" prema pojedinim "zamkama" i mogućnostima komodora i obuhvataju glavne komande koje će početniku najverovatnije biti korisne. U knjizi jednostavno nema dovoljno prostora da se obuhvati sve, ali ona daje solidnu osnovu na kojoj može lako dalje da se gradi.

Da bi se mogle koristiti rafinisanije mogućnosti "šezdesetčetvorke", neophodno je razumeti i određeni obim "teorije", posebno način korišćenja binarnih brojeva i način na koji je organizovana memorija računara. Malo detaljnije razmatraćemo naredbe PEEK i POKE, pošto su one nezamenljive ako od "šezdesetčetvorke" želite da dobijete sve one čudesne stvari za koje ste platili. „Šezdesetčetvorka" zahteva nešto više od nekih drugih mašina, ali smo se trudili da ideje predstavimo na što razumljiviji način i da ih učinimo lakim za korišćenje.

U tekstu se nalazi i veliki broj programa: počev od ispitnih programa za nove naredbe, do prilično dugačkih programa za igranje, grafičke prikaze, manipulisanje podacima na kaseti, sviranje i upravljanje sprajtovima. Većina poglavlja sadrži jednu

---

[1] Commodore 64 Microcomputer User Manual, isporučuje se uz računar.

[2] Commodore 64 Programmer's Reference Guide — može se kupiti u prodavnici koja prodaje Commodore.

ili više vežbi zamišljenih kao provera vašeg shvatanja i sugerisanje nekih ideja. Za sve vežbe navedena su, na kraju poglavlja, i rešenja koja se mogu koristiti kao dodatni izvor informacija.

Dva važna poglavlja pokrivaju *muziku* i *sprajtove*. "Šezdesetčetvorka" ima integrisano kolo za zvuk velikih mogućnosti. To kolo se naziva SID, a može da svira trozvučni akord na različitim instrumentima ili da proizvede zvučne efekte. Naš cilj je da učinimo SID dostupnim svakome ko može da sačuva bistru glavu. Sprajtovi su višenamenski pokretni grafički blokovi izuzetno pogodni za igre, animacije i grafičke prikaze. Njima upravlja integrisano kolo VIC koje je isto toliko moćno (i komplikovano) kao i SID. Naše razmatranje obuhvata pojednostavljen opis memorijskih registara kola VIC i kako se oni koriste za upravljanje sprajtovima. Dali smo i program za automatsko projektovanje sprajtova na ekranu i biblioteku potprograma za rukovanje sprajtovima (u Dodatku 3).

Drugi dodaci obuhvataju korisne stvari koje nisu obuhvaćene u dodacima *Priručnika:* npr. binarne brojeve (verovali ili ne, oni su vam potrebni za konstruisanje sprajtova i muziku), organizaciju memorije i korisne sistemske promenljive.

Važan (i prilično neuobičajen) deo ove knjige sastoji se od niza poglavlja posvećenih *pronalaženju i otklanjanju grešaka*. U tim poglavljima je opisan način utvrđivanja uzroka lošeg rada programa i način otklanjanja grešaka. Svako ko piše svoje sopstvene programe utvrdiće da su ta poglavlja izuzetno korisna. Dajemo i nekoliko sugestija za pisanje dobro strukturisanog programa, a posebno za dobro korišćenje *potprograma* čime se program razbija u manje celine.

Jedna mogućnost, koju *Priručnik* ne spominje, je grafika visokog razlaganja koja se može koristiti za crtanje krivih, grafova i crteža. U knjizi objašnjavamo kako se računar podešava za rad sa grafikom visokog razlaganja i kako se te mogućnosti koriste. Konačno, jedno umereno zamišljeno poglavlje objašnjava kako se koriste *datoteke* u kojima su sadržane informacije na kaseti. Organizacija knjige i izbor sadržaja zasnovani su na našem iskustvu na različitim mikroračunarima raznih proizvođača. Programske liste nisu zamišljene kao čuda usavršenog programiranja, nego smo pokušali da damo efikasne programe koji se mogu uneti u mašinu u nekom razumnom vremenu. Naš cilj je da prosvećujemo, a ne da impresioniramo. A naravno, postoji mnogo mogućnosti mašine koje uopšte nismo spomenuli, ili zbog toga što je za njih potrebno mnogo prethodnog znanja, ili zbog toga što nema dovoljno prostora. To je neizbežno (izuzev ako ne želite da platite velike sume novca na velike tomove), ali vam nudimo postepeno i temeljno uvođenje u "Commodore 64" u kojem se *objašnjava* šta se dešava (ne sviđaju nam se recepti iz kuvara) i u kojem su obuhvaćene glavne karakteristike sa kojima bi novi korisnici trebalo da se upoznaju. Na kraju, početak je uvek najteži.

NAPOMENA ZA STRUČNJAKE

Kao što smo upravo rekli, ovo je knjiga za korisnike koji se prvi put sreću sa računarom i nije pisana za stručnjake. Naravno, *najjednostavniji* put da se dođe do željenog rezultata nije uvek i *najefikasniji*. Zbog toga, ako vam se ponekad čini da postupamo nespretno, verovatno ste u pravu. Međutim, razumno je da prohodate pre nego što počnete da trčite i da naučite jednostavne mogućnosti pre nego što pređete na savršenije i moćnije.

Isto tako ne očekujte da programske liste budu savršene verzije sa grafikom arkadnih igara i brzim računanjem od kojeg zastaje dah. Cilj je bio da se daju proste

iste koje čitalac: (a) može da upiše u razumnom vremenu i (b) razradi i *razume*. Maštovitost dolazi kasnije.

NAPOMENA ZA GRAMATIČARE I PEDANTNE

Pre nekih pola tuceta knjiga odlučili smo da se sebi obraćamo u množini. To je možda malo neuobičajeno, ali je pogodno pošto se bavimo ličnim iskustvom. "Kao što rekosmo jutros našoj ženi. . ." — ne, to ne zvuči kako treba. Tako "ja" označava neku kombinaciju gramatičkih lica dok "mi" znači "ja i čitalac".

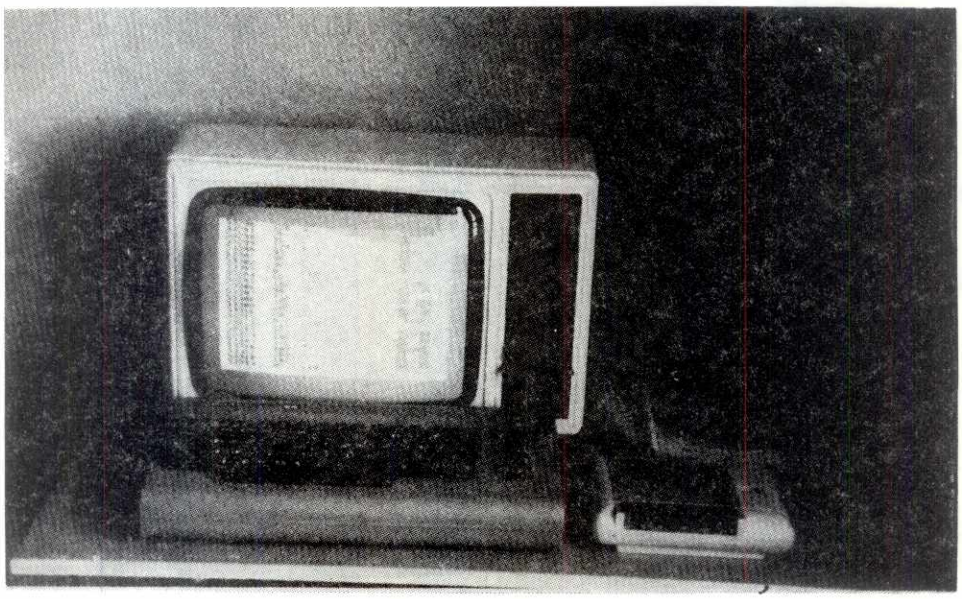
Prvo, kako da pripremimo hardver.
Verovatno ste već uspeli i sami, a ukoliko niste, nekoliko saveta može da vam pomogne.

1 Povezivanje i uključenje

Ako ste ikad povezivali TV-igru ili kućni računar sa televizijskim aparatom, nećete imati problema sa povezivanjem svoje "šezdesetčetvorke". Ako niste, možda ćete utvrditi da detaljan opis može da uštedi traćenje vremena.

Već ste izvadili "šezdesetčetvorku" iz kutije, pošto niko ko u sebi ima imalo krvi ne bi mogao da zamisli da je ostavi zapakovanu duže od trideset sekundi nakon donošenja kući. U kutiji ste, dakle, pronašli:

1. računar;
2. napojni sklop, četvrtastu sivu kutiju sa dva poduža kabla od kojih jedan ima sivu plastičnu stvarčicu sa dva reda tankih šiljaka u sebi;



Sl. 1.1 — Preporučuje se uredan i pristupačan raspored

3. kabl za priključenje na televizijski aparat dužine oko dva metra sa metalnim koaksijalnim konektorima na oba kraja;
4. *Priručnik* (poznat kao *Priručnik za mikroračunar "Commodore 64"*).

Na napojni sklop vežite tropolnu utičnicu (sa osiguračem od 3 ampera), ako već nije vezana.

Pre povezivanja dobro je da napravite pogodan raspored.

Ako ne koristite posebni televizijski aparat, "šezdesetčetvorku" ćete morati da koristite u blizini vašeg televizora. *Nije* dobro da je stavite na pod, rizikujete da nagazite na računar i oštetite ga. Što je još gore, završićete sa grčevima u želucu zbog savijenog položaja prilikom sedenja. Dakle, nabavite sto, nešto kao stočić za kafu, i stolicu i smestite ih pored televizijskog aparata.

Stavite računar na sto i priključite napojni sklop na konektor sa oznakom POWER (napajanje). Odmah do njega je crni prekidač sa oznakom ON/OFF (uključeno/isključeno). Stavite ga u položaj OFF (isključeno). Uključite utikač u utičnicu u zidu.

Uzmite kabl za povezivanje sa televizijskim aparatom i priključite konektor sa šiljkom u sredini na odgovarajući konektor sa zadnje strane računara. Gurnite ga tako da dobro nalegne. Drugi kraj priključite na televizijski aparat tamo gde se obično priključuje antena.

Uzgred, "šezdesetčetvorka" savršeno dobro radi i na crno-belom televizoru, ali, naravno, ne možete koristiti mogućnosti rada sa bojama.

Neki televizori imaju obrtni birač sa brojevima kanala. Ako je vaš televizor takav, podesite ga na kanal 36.

Ako nemate takav televizor, najverovatnije imate šest ili osam dugmadi za biranje kanala. Najčešće se primaju emisije najviše sa tri do četiri predajnika, tako da imate slobodno bar jedno dugme. Izaberite slobodno dugme i pritisnite ga.

Uključite televizor i računar (ne zaboravite na utičnice za mrežni napon u zidu).

Najverovatnije ćete na ekranu imati samo "sneg", pošto birač kanala nije podešen. Negde na televizoru imate šest (ili osam) malih obrtnih birača. Često su smešteni iza nekog poklopca. Ako do sada niste znali gde su, tražite, bićete iznenađeni. Ponekad se tablica sa nazivom proizvođača otvori kao fioka, a ponekad se samo otvori poklopac kad pritisnete na pravo mesto. Kad ste pronašli birače, podesite odgovarajući (onaj koji ste izabrali pritiskom na dugme) sve dok na ekranu ne dobijete tekst. Možda će biti potrebno da birač okrenete više puta ili da promenite smer okretanja, ako utvrdite da u jednom smeru nemate više šta da tražite.

Čak i ako imate birač sa brojevima i izabrali ste kanal 36, možda će biti potrebno dodatno fino podešavanje da biste dobili što bolju sliku. Možda će biti potrebno i da podesite osvetljenost i kontrast slike kao i boje (a ponekad i vertikalnu i horizontalnu sinhronizaciju). Ono što vidite na ekranu je plava pozadina i tekst:

**** COMMODORE 64 BASIC V2 ****

64K RAM SYSTEM 38911 BASIC BYTES FREE

Kad ste došli do ovoga, računar je spreman za dalji rad. Ako na ekranu niste dobili ništa, proverite da li je sve priključeno i povezano kako treba, kao i da li konektori dobro naležu, a zatim ponovo pokušajte da dobijete sliku na ekranu. Kad se prilikom podešavanja približite pravom mestu, videćete na ekranu trepćuće linije

koje se vide u "snegu". Ako još uvek ne možete da dobijete sliku na ekranu okretanjem birača sa jednog na drugi kraj, ili okrećete pogrešni birač, ili nešto drugo nije u redu. Ako ste baš spretni, proverite konektore na kablovima (da neka žica nije otkačena) i/ili proverite da vodovi nisu prekinuti pomoću baterije i sijalice. Ako rezultata još uvek nema, obratite se prodavnici u kojoj ste kupili računar. Računarski hardver je veoma pouzdan, ali se ponekad *može* pokvariti.

KASETOFONI

U kasnijoj fazi ćete želeti da priključite na vašu "šezdesetčetvorku" i kasetofon da biste upisali programe na kasetu. Trenutno nemojte to da činite. Ako baš želite da znate kako se to radi, pogledajte poglavlje 20.

JEDAN SAVET

Sasvim je moguće da neki članovi vašeg domaćinstva nisu zaljubljeni u "šezdesetčetvorku" kao vi. Oni će uspešno sakrivati svoja osećanja sve dok jednog dana ne uključe televizor da bi gledali "Dinastiju", pa na ekranu ugledaju višebojnu snežnu oluju. Dakle, kad završite rad sa računarom, *vratite televizor u prethodno stanje i proverite da li je dobro podešen.* Čak i računar treba da zna gde mu je mesto.

Čemu služe ti tasteri na računaru?
Ako ih pritiskate nasumice, ne preterujte.
Međutim, na taj način ćete barem steći dobar osećaj.

2 Tastatura

Tastatura "šezdesetčetvorke" u mnogome liči na tastaturu pisaće mašine, a slova su raspoređena po standardnom rasporedu QWERTY. U osnovi to je glup raspored, nastao u prvim danima pisaćih mašina kad je dolazilo do zaglavljivanja poluga sa slovima, ali je u poslednje vreme postao standard tako da svi treba da se priviknemo na njega.

Pored slova i brojeva tu su i tasteri sa oznakama CTRL, RUN/STOP, RESTORE, RETURN i drugim. Dva tastera sa oznakom CRSR imaju i strelice. Na donjem delu tastature nalazi se razmaknica, a sa desne strane četiri šira tastera sa oznakama f1, f3, f5 i f7. U donjem redu levo nalazi se taster sa simbolom koji liči na:

C=

koji ću ja u daljem tekstu zvati taster COMMODORE. Cilj ovog poglavlja je da vas upozna sa tastaturom, pa ukoliko ste stručnjak, možete da preskočite najveći deo ovog poglavlja.

ALFANUMERIČKI TASTERI

Alfanumerički tasteri su tasteri sa slovima i brojevima. Isto kao i pisaća mašina na papiru tako i oni na ekranu daju odgovarajući znak. Malo eksperimentišite popunjavajući ekran svim i svačim, ali pazite da ne pritisnete neki od drugih tastera sa smešnim natpisima.

Primetićete da trepćući kvadrat, poznat pod imenom *pokazivač* (kursor), određuje mesto na koje će se smestiti znak i da se u toku kucanja pokazivač automatski pomera za jedno mesto (i u novi red, po potrebi).

Hmmmmm. Divno, ali kako se osloboditi svega onoga što smo napisali na ekranu?

CLR/HOME

Ovaj taster se nalazi desno u gornjem redu. Ako ga pritisnete, utvrdićete da se pokazivač vratio u levi gornji ugao ekrana (polazni položaj). Ako pritisnete i taster SHIFT (prvo pritisnete SHIFT, pa dok je on pritisnut, i taster CLR/HOME), izbrisaćete (engl. clear) sve znakove sa ekrana.

SHIFT

Taster SHIFT može mnogo više od toga. U stvari, taster SHIFT menja efekat koji se postiže pritiskom na neki drugi taster. Ako taster na *gornjoj površini* ima dva različita simbola, tasterom SHIFT se bira gornji znak. Na primer:

SHIFT i taster 1 daju znak #
SHIFT i taster 8 daju znak (

Ako se radi o alfabetskim tasterima, taster SHIFT daje desni simbol od dva znaka nacrtana na *prednjoj strani* tastera. To je specijalni simbol "PET-grafike" i razlikuje se od tastera do tastera.

Postoje dva tastera SHIFT (da bi se olakšao rad). Taster SHIFT LOCK ima efekat da čini da se računar ponaša kao da je taster SHIFT stalno pritisnut. Da bi se računar vratio na prethodno stanje, treba ponovo pritisnuti taster SHIFT LOCK.

COMMODORE (C=)

Ovo je, u stvari, dodatni taster SHIFT. On bira *levi* grafički znak sa prednje strane tastera. On ima i nekoliko nejasnijih funkcija koje ću objasniti kasnije u tekstu, ali jednu treba navesti već sada:

Mala slova

"Šezdesetčetvorka" može, osim velikih slova ABCD. . . , da prikaže i mala slova abcd. . . Da biste dobili mala slova, pritisnite taster COMMODORE i SHIFT istovremeno. Odmah će sva velika slova na ekranu postati mala kao i sva slova koja dalje kucate. Grafički simboli se takođe menjaju. U stvari, "šezdesetčetvorka" može da bira između dva skupa znakova (skup znakova = svi znakovi koji mogu da se prikažu na ekranu) tako da od vašeg izbora zavisi koji skup ćete koristiti. Pritiskom na tastere COMMODORE i SHIFT vraćate računar u prethodno stanje (velika slova). Da biste videli koji se znak zamenjuje kojim, pogledajte Dodatak E.

TASTERI ZA POKAZIVAČ

Dva tastera sa oznakama CRSR omogućavaju pomeranje pokazivača na ekranu. Ako ne pritisnete taster SHIFT, levi taster CRSR pomera pokazivač za jedno

mesto dole, a desni za jedno mesto desno. Ako se drži i taster SHIFT, levi taster CRSR pomera pokazivač gore, a desni levo. Prilikom pomeranja levo ili desno pokazivač se pomera u novi red kad dođe do kraja ekrana. Kad se pokazivač pomera dole, pa dođe do kraja ekrana, tekst na ekranu *klizi* (tj. pomera se gore kao celina ostavljajući na dnu nove prazne redove). Kad se pokazivač pomera naviše, računar mu neće dozvoliti da izađe sa ekrana, ali ekran neće kliziti. Korišćenje tastera za pomeranje pokazivača omogućava da kucate znakove na ekranu gde god želite, a ne samo tamo gde vas vodi automatsko pomeranje pokazivača.

RUN/STOP

Bez tastera SHIFT ovaj taster ima značenje: STOP što znači da zaustavlja izvršenje programa. To je korisno ako u programu imate grešku, a ne želite da sačekate da se program izvrši do kraja, pa da ga onda ispravite. (U stvari, postoje neke veoma uobičajene greške koje sprečavaju da program uopšte završi rad.)

Efekat RUN (pozivanje programa sa kasete) postiže se jednovremenim pritiskom na taster SHIFT. Na ekranu ćete dobiti smešnu poruku:

LOAD (učitaj)

PRESS PLAY ON TAPE (pritisni taster za reprodukciju na kasetofonu)

što nas u ovom momentu ne interesuje *izuzev*, ako to greškom uradite pa računar više ne reaguje na tastaturu . . . . .

RESTORE

Iz ovoga možete da se izvučete tako što ćete jednovremeno pritisnuti tastere RUN/STOP i RESTORE (taster RESTORE ne funkcioniše zasebno). Ovim se računar vraća na stanje u kojem je bio prilikom uključenja. Ako ste u memoriji imali program, nećete ga izgubiti, ali se npr. boje vraćaju na početne. Kad god se nađete u nevolji, RUN/STOP + RESTORE treba da vas izvuku uz minimalan gubitak uloženog truda (alternativni način, isključenje napajanja, dovodi do gubitka svega što je bilo u memoriji). U važećem računarskom žargonu RESTORE vam omogućava "vrući restart".

INST/DEL

Bez tastera SHIFT ovo je taster za *brisanje* (engl. delete). Otkucajte nekoliko slova, a zatim pritisnite DEL. Primetićete da je poslednje slovo nestalo i da se pokazivač vratio za jedno mesto. Još jedan pritisak na DEL briše sledeće slovo itd. Možete da koristite tastere CRSR da biste namestili pokazivač tamo gde želite i *zatim* pritiskom na taster DEL obrišete znak *neposredno sa leve strane* pokazivača. Znakovi sa desne strane se automatski pomeraju tako da ne ostaje prazan prostor.

INST služi za *ubacivanje* (engl. insert). On ubacuje prazna polja desno od pokazivača i pomera ostatak reda. Da biste to postigli pritisnite tastere SHIFT i INST/ /DEL. Njegova glavna namena je pomoć u pisanju programa (vidi poglavlje 10).

CTRL

To je *upravljački* (engl. control) taster. On pomalo liči na taster SHIFT po tome što menja efekte ostalih tastera pritisnutih u isto vreme. On, međutim, važi samo za gornji red tastera i tastere CRSR.

Na primer, pritisnite taster CRTL i, istovremeno, taster sa oznakom 9. Utvrdićete da sve što dalje napišete ima *inverzne* boje (boja osnove je postala boja znaka i obratno). To je i naznačeno slovima RVS ON (reverse on = uključenje obratnih boja) na tasteru 9. Slično i RVS OFF (reverse off = isključenje obratnih boja) na tasteru 0 vraća normalne boje (naravno, ako je bio pritisnut i taster CTRL).

CTRL sa tasterima 1 — 8 menja boju znaka onako kako je označeno na prednjoj strani tastera: black (crna), white (bela), red (crvena), cyan (svetlo plava), purple (purpurna), green (zelena), blue (plava), yellow (žuta). Ovo se može koristiti za prikaz u bojama (mogu se menjati i boje osnove i ruba, ali ne tako jednostavno — vidi poglavlje 13).

RETURN

Ovaj taster kaže računaru: "Izvrši naredbu koja je upravo otkucana". Sve dok ne pritisnete taster RETURN, mašina je, u biti, pasivan uređaj za prikazivanje na ekranu kao pisaća mašina bez svojega "ja". Kad kucate program (poglavlje 5), morate da pritisnete taster RETURN posle svakog reda programa da bi se on zapamtio u memoriji.

Pokušajte ovo: otkucajte bilo šta, a zatim pritisnite taster RETURN. Dobićete poruku:

```
? SYNTAX ERROR
READY
```

To znači da je računar pokušao da izvrši naredbu, nije je razumeo i to vam javio. Nije ni čudo. Ako naredba ima smisla, mašina ne šalje poruku o grešci nego je izvršava. Npr. pokušajte sa naredbom

```
PRINT "NESTO" (+ RETURN)
```

i na ekranu će se pojaviti reč NESTO.

Ako se pokazivač nalazi u nekom redu ekrana, a vi slučajno pritisnete taster RETURN, računar će pokušati da izvrši naredbu u tom redu. Obično to neće imati smisla, pa ćete dobiti poruku da postoji greška u sintaksi. Nemojte, dakle, da vas iznenađuju slučajne poruke o grešci.

Neki odgovori su nejasniji. Otkucajte bilo šta, pritisnite RETURN i dobićete poruku o grešci. Pomerite sada pokazivač jedan red gore (u red u kojem piše READY) i ponovo pritisnite RETURN. Pogodite šta ćete dobiti? Poruka će biti:

```
? OUT OF DATA ERROR
READY
```

što može da vas zbuni. Glupa mašina je READY protumačila kao naredbu READ Y

koja joj kaže da potraži naredbu DATA. Pošto takve naredbe nema . . . eto u tome je problem. Ovaj bizarni efekt navodim samo zato što može da dovede do priličnog češkanja po glavi, ako ne shvatite šta se dešava.

RAZMAKNICA

Kao i na pisaćoj mašini dugačka prečka na dnu tastature ostavlja prazno mesto u tekstu. U ovoj knjizi ću za prazno mesto koristiti simbol:

▽

(ne tražite ▽ na tastaturi, nema ga. Zato sam ga i izabrao). Ako je sasvim očigledno gde treba da bude prazno mesto, biću toliko slobodan da izostavim znak ▽.

FUNKCIONALNI TASTERI

Široki tasteri su "funkcionalni programabilni tasteri". Ako ih pritisnete, odziv mašine može da bude bilo kakav što se postiže pogodnim programiranjem. Ovo je detaljnije objašnjeno u poglavlju 24.

DALJE?

Ovo nikako nije kraj priče. Tastatura "šezdesetčetvorke" može da čini i druge stvari. Ali na početku nema potrebe da brinete o tome, pa sam suptilnije stvari odložio za kasnije kad vam budu potrebne.

Nije neophodno da savladate sve kurseve programiranja da biste upoznali mogućnosti „šezdesetčetvorke".
Pred vama su tri kratka programa. . .

3 Da vam pobudi apetit

Za upisivanje programa nije obavezno razumevanje programiranja, uz uslov da programe dobijete od nekog drugog. Za početak to je dobra vežba za upoznavanje tastature. Međutim, nadam se da su vaše ambicije znatno veće nego što je izvršavanje programa koje stvaraju drugi.

Za svaki od tri dalje u tekstu navedena programa:

1. Upišite komandu NEW, a zatim pritisnite taster RETURN.
2. Pažljivo prepišite listu programa. Nakon svakog reda pritisnite taster RETURN. Ukoliko načinite grešku, a pri tome još niste pritisnuli RETURN tada ispravite red koristeći taster DEL; ukoliko ste pritisnuli taster RETURN tada ponovite ceo taj red.
3. Upišite komandu LIST, a zatim pritisnite taster RETURN da biste proverili da li se lista programa slaže sa listom na ekranu.
4. Otkucajte RUN, pritisnite taster RETURN i pažljivo posmatrajte šta se zbiva na ekranu.

Neka ovi primeri posluže za vežbu pošto će vam to koristiti.

Vežba 1: "Crvići"

```
10 FOR N=1 TO 1000
20 C=105 + INT (4*RND (0))
30 IF C=108 THEN C=117
40 PRINT CHR$(C);
50 NEXT N
```

Vežba 2: Treptanje

```
10 FOR N=1 TO 100
20 X=INT (16*RND (0))
30 POKE 53281,X
40 POKE 53280,15-X
50 FOR T=1 TO 100
60 NEXT T
70 NEXT N
```

Vežba 3: kakutani

Problem kakutani, još uvek nerešen, sastoji se u sledećem postupku. Izaberemo ceo broj (recimo 48). Ukoliko je on paran, deli se sa 2 (u našem slučaju daje 24). Ukoliko je neparan množi se sa 3 i dobijenom proizvodu dodaje 1. Ovaj postupak se beskonačno ponavlja. Da li se uvek kao rezultat dobija broj 1?
Npr. ovde imamo:

48 → (deljenjem sa 2) → 24 → (deljenjem sa 2) → 12
→ (deljenjem sa 2) → 6 → (deljenjem sa 2) → 3
→ (utrostručen + 1) → 10 → (deljenjem sa 2) → 5
→ (utrostručen + 1) → 16 → (deljenjem sa 2) → 8
→ (deljenjem sa 2) → 4 → (deljenjem sa 2) → 2
→ (deljenjem sa 2) → 1

i na kraju dobijamo 1. Sledeći program odabira početni broj na slučajan način i formira niz brojeva prema prethodno pomenutom pravilu. Program prekida rad ako je rezultat 1.

```
10 N=100 + INT (1000*RND (0))
20 PRINT N
30 M=N: L=INT (M/2)
40 IF M=2*L THEN N=L
50 IF M>1 AND M<>2*L THEN N=3*M+1
60 IF N>1 THEN 20
70 PRINT 1
```

IZVINJENJE

Kada jednom ovladate tastaturom, moći ćete da otkucate naredbu, ili kompletan program i da ga naterate da radi čak i ako ne razumete nijednu naredbu programa (to ste već učinili u vežbama 1, 2 i 3). Možete čak da "pozajmite" programske liste iz knjiga i časopisa. *To je sasvim normalno* i ne treba da se osećate krivim — svakom se ponekad žuri. Ovo pomaže da se zadrži interesovanje i poveća samopouzdanje. (Kao kod svih dobrih stvari ne treba preterivati. Ako se u toku godine informatičke tehnologije ne uradi ništa drugo nego stvori generacija ljudi koji znaju samo da kopiraju tuđi softver i ne pišu svoj, možda bi bilo bolje da se taj novac potroši na subvencionisanje televizijskog programa).

Ponekad, kad objašnjavamo neku naredbu BASIC-a, posebno u prvom delu dok još malo znamo, bilo bi zgodno koristiti neku naredbu koja *još nije objašnjena.* To je upravo ono što ću ja uraditi, ako mi se učini da tako treba.

Nemojte odmah da se užasavate i da gunđate: "Budala, što nam to nije ranije rekao!" Nego stisnite zube, otkucajte naredbu bez obzira što je ne poznajete i posmatrajte učinak! A učinka će biti i to za vaše dobro.

U svakom slučaju, ja se izvinjavam zbog toga.

Program je samo niz instrukcija, koji se pamti, da bi kasnije bio izvršen. Međutim, pre toga možete pokušati da date komande direktno sa tastature. Videćete kako „šezdesetčetvorka" pokazuje svoje aritmetičke mogućnosti.

4 Direktne komande

Kada sa tastature unesete instrukciju i neposredno nakon toga pritisnete taster RETURN tada koristite *direktni način rada*. ("Način" je termin u računarskom žargonu kojim se ukazuje na "stanje duha" računara). Odatle i potiče šala u nekom računarskom časopisu da je u izvesnoj televizijskoj emisiji voditelj izgledao kao da radi u "zaprepašćenom načinu rada".

Postoji i drugi, *odloženi način rada*, u kome se naredba smešta u memoriju računara da bi kasnije bila izvršena. U stvari, obično se pamti čitava grupa takvih naredbi koja predstavlja *program*. O programima će biti reči u sledećem poglavlju. Za sada nastavimo sa direktnim načinom rada.

"Šezdesetčetvorka" u direktnom načinu rada predstavlja neku vrstu kombinacije kalkulatora i pisaće mašine. Pokušajte:

PRINT 2+2

(a zatim RETURN). Na ekranu ćete videti odgovor: 4. Pa sad, možda ovo nije interesantno, ali ako unesete

PRINT 12345678 + 87654321

odmah ćete dobiti odgovor 99999999.

Vežba 1

Iskoristite računar za sračunavanje:

1. 7 + 4
2. 17 + 41
3. 5 + 16
4. 15123 + 97784

"Šezdesetčetvorka" može vršiti i oduzimanje:

PRINT 11 — 5

```
PRINT 77 — 3
PRINT 55555 — 22222
```

i tako dalje.

Za operaciju množenja koristi se zvezdica (*) umesto uobičajenog znaka × (da ne bi bilo zabune sa slovom X). Pokušajte:

```
PRINT 2*2
PRINT 2*3
PRINT 5*5
PRINT 99*77
```

Na kraju, operacija deljenja naznačava se simbolom / umesto sa ÷ Tako se količnik brojeva 24 i 3 može dobiti sa:

```
PRINT 24/3
```

a količnik brojeva 777 i 7 sa:

```
PRINT 777/7
```

Pored celih brojeva "šezdesetčetvorka" može obrađivati i decimalne brojeve kao što je 27.342, odnosno negativne brojeve kao što su —99 ili —27.342. Pored osnovnih matematičkih operacija sabiranja, oduzimanja, množenja i deljenja mogu se koristiti i različiti matematički izrazi.

U okviru ove knjige koristićemo samo elementarne matematičke operacije. Na kraju krajeva, nije svako ko želi da se bavi računarima i budući matematičar. A upravo zadivljuje koliko mnogo se može uraditi *bez* matematike.

STEPENOVANJE

Za one koji naginju matematici, ovde će biti izložena jedna osobina. Umesto uobičajene notacije:

X^N

za N-ti stepen broja X, u slučaju "šezdesetčetvorke" koristi se simbol strelica-gore (↑):

X↑N

Tako, npr., treći stepen broja 5, tj.:

$5^3 = 5*5*5 = 125$

se zapisuje u obliku

5↑3

U poglavlju 33 izložena su neka upozorenja koja se odnose na greške zaokruživanja koje nastaju pri korišćenju simbola strelica-gore (↑).

ODGOVORI

Vežba 1

1. PRINT 7 + 4	(rezultat 11)
2. PRINT 17 + 41	(rezultat 58)
3. PRINT 5 + 16	(rezultat 21)
4. PRINT 15123 + 97784	(rezultat 112907)

Način da naterate računar da radi ono što želite je da sastavite potrebne naredbe u određenom redosledu.

5 Programi

Program je niz instrukcija koje računar treba da izvrši. Baš kao recepti u kuvaru:

— uzmite dva jajeta,
— razbijte ih u činiju,
— mešajte 20 sekundi,
— dodajte dva kilograma šećera,
— dodajte 4 šoljice kokica,

i tako dalje. Međutim, računarski program mora da bude napisan veoma preciznim jezikom.

Evo jednog jednostavnog programa:

```
10 PRINT "ZDRAVO!";
20 GOTO 10
```

Ovo možete da otkucate u računar. Prvo otkucajte NEW i pritisnite RETURN — tako ćete izbrisati sve što je možda ostalo od prethodnog programa. *Pre kucanja novog programa UVEK postupajte na taj način.* Zatim otkucajte prvi red:

```
10 PRINT "ZDRAVO!";
```

pa pritisnite RETURN. Pošto naredba počinje *brojem*, u ovom slučaju 10, računar neće izvršavati naredbu odmah, nego će je smestiti u memoriju i izvršiti kad mu se to kaže. Sada otkucajte:

```
20 GOTO 10
```

i pritisnite taster RETURN. Za sada ne vodite računa o *značenju* ove "papazjanije".

Znači, smestili smo program u memoriju. Ali kako ćemo reći računaru da ga *izvrši?* (Tako se to kaže u žargonu). Otkucaćemo:

```
RUN
```

(Naravno, prestaću to da spominjem i pritisnuti taster RETURN, ali očekujem da pritisnete taster RETURN na kraju svakog "reda" programa i svake naredbe u direktnom načinu rada).

Uz pretpostavku da ste prepisali baš onako kako je štampano, na ekran će izbiti gomila pozdrava. Reč "ZDRAVO" će se ispisati po celom ekranu i izazvati brzo klizanje ekrana. I to će se ponavljati unedogled, ako ga ne zaustavite. Ali kako?

ZAUSTAVLJANJE MAŠINE

Tu pomaže taster STOP. Kad ga pritisnete, program se prekida ma šta u tom trenutku radio i šalje nam poruku o tome gde se nalazio. Ako sada pritisnete taster STOP, program će prestati da se izvršava ostavljajući najveći deo ekrana prekriven rečima "ZDRAVO!" Na dnu će se pojaviti poruka:

```
BREAK IN 10
READY
```

ili, možda:

```
BREAK IN 20
READY
```

Uvek možete da koristite taster STOP za zaustavljanje programa, ako vam se čini da je zaglavio ili da ne radi kako treba. Ako to ne funkcioniše, pokušajte RUN/STOP plus RESTORE kako je opisano u poglavlju 2.

CONT

Da biste nastavili izvršavanje nakon zaustavljanja tasterom STOP, možete da koristite naredbu:

```
CONT
```

(skraćeno od "continue" = nastavi). Pokušajte. I evo opet klizanja ekrana.

ŠTA SE DEŠAVA?

U programu pisanom u programskom jeziku BASIC svaka instrukcija ima svoj broj koji se zove *broj reda*. Ovde su brojevi redova 10 i 20. Normalno mašina izvršava naredbe po rastućem redosledu brojeva redova tako da će, u našem primeru, prvo izvršiti red 10, a zatim 20.

Međutim, neke naredbe imaju učinak promene sledećeg reda koji treba da se izvrši. Evo te naredbe u našem programu:

```
GOTO 10
```

Ona znači: "Ne obaziri se na sledeći red, čak i ako postoji, nego izvrši instrukciju u redu 10".

Da bismo razumeli šta program radi, potrebno je da znamo još nešto. Tačka-zarez (;) posle naredbe PRINT "ZDRAVO" kaže računaru da *sledeći* znak treba prikazati neposredno iza završetka teksta ZDRAVO! Ako se tačka-zarez izostavi, mašina će ispisivati tekst u sledećem redu ekrana.

Znači, kad otkucamo RUN, mašina radi sledeće:
Izvršava prvi red:

```
10 PRINT "ZDRAVO";
```

i na ekranu ispisuje:

```
ZDRAVO!
```

Pošto naredba *ne* sadrži ništa što bi menjalo redosled izvršenja, izvršava se *sledeći* red (po rastućem broju):

```
20 GOTO 10
```

Izvršenjem ove naredbe program se vraća na red 10. Znači, on sada ponovo ispisuje ZDRAVO! tako da na ekranu imamo:

```
ZDRAVO! ZDRAVO!
```

Program zatim prelazi na red 20 koji ga opet vraća na red 10:

```
ZDRAVO! ZDRAVO! ZDRAVO!
```

i nastavlja da ispisuje ZDRAVO! do beskonačnosti (ili dok ga ne zaustavi neki spoljni događaj kao taster STOP, kvar ili kraj vasione). Prilikom šestog ispisivanja doći će do kraja reda i nastaviti u sledećem, a nakon oko 140 ispisivanja doći će do donjeg reda, pa će *ekran početi da klizi*. U praksi se sve ovo dešava tako brzo da ne možete da vidite ništa drugo, nego brzo promicanje raznih ZDRAVO! dok ekran klizi.

Da biste ga usporili, dodajte još jedan red:

```
15 FOR I = 0 TO 200 : NEXT I
```

(Biće objašnjeno u poglavlju 6). Sada jasno može da se vidi kako računar ispisuje.

LISTANJE

Kad ste otkucali program, u nekoj fazi ćete možda hteti da ga vidite na ekranu. (Na primer, možda želite da izmenite neki red, a zaboravili ste mu broj). Da biste to postigli, u direktnom načinu rada otkucajte naredbu:

```
LIST
```

Pokušajte.

Programu možete da dodate nove redove samo kucanjem. Ne vodite računa o uređivanju redosleda, pošto će računar sam složiti program po redosledu. Ako otkucate:

```
20 PRINT "NIJE U REDOSLEDU"
10 PRINT "OVAJ PROGRAM"
```

a zatim LIST, dobićete:

```
10 PRINT "OVAJ PROGRAM"
20 PRINT "NIJE U REDOSLEDU"
```

Slično, možete da *izmenite* red tako da otkucate drugi sa istim brojem. Otkucajte:

```
20 PRINT "JE U REDOSLEDU"
```

i ponovo naredbu LIST.

Da biste izbrisali red, otkucajte samo njegov broj i ništa drugo (izuzev, naravno, RETURN). Tako:

```
20 (plus RETURN)
```

briše red 20 (računar ga tretira kao programski red bez stvarne naredbe, tj. samo kao broj i odmah ga ignoriše). Postoje i bolji načini da se izmeni red. (Vidi poglavlje 10). Ovo što sam ovde naveo predstavlja minimum sa kojim možete da nastavite i koji nam omogućava da nastavimo sa interesantnijim temama.

Da biste izlistali određene redove (što može da bude neophodno kod dužih programa zbog klizanja ekrana), koristite naredbu:

```
LIST 50—180
```

kojom ćete izlistati redove od 50 do 180.

NEW

Ovu naredbu sam već naveo u poglavlju 3, ali je vredno spomenuti je još jednom. Da biste izbrisali stari program iz memorije, otkucajte:

```
NEW
```

i pritisnite taster RETURN. Ako to ne uradite, ostaci prethodnog programa će se nalaziti u mašini i dovesti do velikog broja neobjašnjivih grešaka u programu koji ste pažljivo ukucali.

Naredbu NEW *možete* da koristite i u programu, ali sve što time postižete je samouništenje programa.

STOP i END

Obe naredbe:

```
STOP i
END
```

izazivaju prekid izvršenja programa. Razlika je u tome što posle naredbe STOP možete da nastavite izvršenje tako što ćete otkucati CONT. Posle naredbe END ne možete da nastavite izvršenje. Uredan program treba da završi naredbom STOP ili END, a STOP vam može zatrebati da sprečite izvršenje programa u pogrešnom opsegu brojeva redova.

BROJEVI REDOVA U BASIC-u

Ne koriste svi vaši programski jezici brojeve redova da bi mašini saopštili redosled izvršavanja naredbi, ali *određeni* redosled uvek postoji. Međutim, neke naredbe su predviđene za izmenu ovog prirodnog redosleda, pa upućuju mašinu na izvršenje neke ranije instrukcije (verovatno uz male izmene). Ono što računar čini tako fleksibilnim je mešavina preciznih instrukcija i promenljivih kao i redosleda u kojem se instrukcije izvršavaju.

U BASIC-u i nekim drugim jezicima svaka instrukcija ima broj. Na "šezdesetčetvorci" možete da koristite brojeve od 0 do 63999. *Ne* numerišite redove 1, 2, 3 itd. Uobičajenije je da se koristi numerisanje 10, 20, 30. . . Cilj je da se *ostavi prostor* za dodavanje novih redova, ako se kasnije ispostavi da je potrebno. Bilo bi veoma teško ubaciti red između redova 2 i 3 u nekom programu. Međutim, ne postoji neko pravilo o tome da brojevi treba da budu *okrugli*. Možete redove da numerišete i ovako: 17, 18, 25, 356, 999, 1000, 1003, 1010, 1020, 5033. Većina programa počinje sa urednim numerisanjem, a završava sa šarenilom brojeva kako se otklanjaju greške u programu.

REDOVI SA VIŠE NAREDBI

U jednom redu možete da napišete i više naredbi uz uslov da ih odvojite pomoću dve tačke (:). Na primer:

```
10 FOR T = 1 TO 200 : PRINT T; : PRINT "▽ PUTA 999 DAJE ▽"; :
   PRINT 999*T : NEXT T
```

Ovim se štedi na prostoru i vremenu kucanja, ali to nije uvek dobro rešenje jer:

1. program može da se grana samo na *početak* reda sa više naredbi.
2. Izmena reda sa greškom je teža.

Međutim, postoje slučajevi kad upotreba redova sa više naredbi štedi vreme, a ne izaziva nikakve probleme. To je slučaj, npr. kad se dodeljuju vrednosti promenljivima (vidi poglavlje 8):

```
10 A = 1 : B = 2 : C = 3 : D = 4 : E = 5
```

U nekima od mojih programa naći ćete redove sa više naredbi pošto sam smatrao da treba nešto i o tome da znate. Međutim, program se može pisati i bez njih, a sigurno je da ih ne treba koristiti suviše, jer program postaje nečitak. Jedan od poznatih izdavača obrazovnog softvera *zabranjuje* redove sa više naredbi upravo iz tog razloga.

REM: OBJAŠNJENJE REM-A

Postoji komanda koja omogućava da u program upišete komentare kao podsetnik (za vas i ostale). Ta komanda je:

```
REM
```

U bilo kom redu računar će ignorisati sve što se nađe iza naredbe REM. Znači možete da napišete ovako:

```
150 REM PROGRAM ZA PRACENJE SATELITA
160 TETA = 52: REM GEOGRAFSKA SIRINA
170 FI = 35: REM GEOGRAFSKA DUZINA
```

Primetimo da će u redu sa više naredbi sve naredbe posle REM biti ignorisane tako da će u slučaju:

```
200 X=365: REM DUZINA GODINE: Y=31: REM DUZINA MESECA
```

promenljivoj X biti dodeljena vrednost 365, a promenljivoj Y *neće* biti dodeljena vrednost 31.

Kad prvi put počnete da programirate verovatno će vam se učiniti da naredbe REM smetaju. A osim toga potrebno je vreme da se ukucaju. Možete da *izostavite* sve naredbe REM, *ali* možete izazvati problem, ako postoje naredbe GOTO ili GOSUB koje se pozivaju na broj reda sa naredbom REM. Najjednostavniji način da se ovo izbegne je da se kuca samo naredba REM bez komentara.

Kako vaše samopouzdanje bude raslo, utvrdićete da su te naredbe REM prilično korisne i počećete da ih stavljate svuda u programe. U ovoj knjizi pokušao sam da nađem kompromisno rešenje dajući nekoliko ključnih komentara i nastojeći da programi budu što je moguće kraći, i što razumljiviji. (Postoje različite tehnike sažimanja kojima se skraćuju programske liste, ali one dovode do toga da se liste teško čitaju, pa ih nisam koristio.)

SKRAĆENE KLJUČNE REČI

Većina ključnih reči BASIC-a (naredbe kao PRINT i GOTO) može da se skrati. Npr. GOTO se može skratiti sa:

G SHIFT/O

Dodatak D *Priručnika* daje listu ovih skraćenica. (*Upozorenje:* ima nekoliko štamparskih grešaka: "G" u GOTO i razni grafički znakovi u kućicama). Pošto ove skraćenice daju čudne liste ja ih u knjizi ne koristim, ali vi možete ako hoćete. One stvarno štede vreme.

ZADATAK PROGRAMERA

Sada bi trebalo da bude jasno šta je zadatak programera. Da bi se postigao određeni cilj, programer treba da sastavi niz instrukcija koje, kad ih računar *tačno* izvrši, daju željeni rezultat. Važno je shvatiti da računar nema pojma o tome šta je "svrha" programa. On jednostavno izvršava instrukcije, on je brzorazmišljajući i savršeno pedantan rob, pa ako mu kažete da uradi nešto glupo, on će to i učiniti.

To ćete otkriti vrlo brzo, čim počnete da pišete programe.

Osnova tehnike programiranja omogućava računaru ponavljanje izvršenja jednog zadatka više puta. Još bolje, moguće je u okviru zadatka izvršiti propisane promene. U primerima su navedene tablice množenja.

6 Petlja

U ovom poglavlju upoznaćete naredbu FOR ... NEXT (koju smo već koristili da bismo usporili rad računara) koja ukazuje računaru da naznačeni zadatak treba ponoviti više puta. Ovo nije od posebnog značaja ukoliko se radi o zadatku koji je nepromenljiv, ali zadatak koji se ponavlja može biti takav da se u svakom narednom koraku menja neki detalj. U tom slučaju ovo je bitno sa aspekta programera.

Osećajući se kao prosvetitelj savetujem vam da upišete sledeći program, a ja ću ga objasniti kad to uradite.

```
TABLICE MNOŽENJA

10 PRINT CHR$(147)          [brisanje ekrana. Vidi poglavlje 7]
20 PRINT "TABLICA MNOZENJA SA 7"
30 PRINT
40 FOR N=1 TO 12
50 PRINT N; "X▽7▽="; 7*N
60 NEXT N
```

Izvršavanjem ovog programa, ukoliko niste načinili neku grešku, dobićete:

```
TABLICA MNOZENJA SA 7

 1 X 7 = 7
 2 X 7 = 14
 3 X 7 = 21
 4 X 7 = 28
 5 X 7 = 35
 6 X 7 = 42
 7 X 7 = 49
 8 X 7 = 56
 9 X 7 = 63
```

10 X 7 = 70
11 X 7 = 77
12 X 7 = 84

KAKO RADI PETLJA

Petlja je postupak koji započinje naredbom FOR, naznačenom u redu 40, a završava se naredbom NEXT u redu 60. Pri tome broj N predstavlja *brojač*, koji služi da ukaže računaru na početnu vrednost (1) i krajnju vrednost (12) petlje.Ovo je dato u redu 40:

FOR	(oznaka početka petlje)
N	(brojač u petlji)
=1	(početna vrednost za N)
TO	(doseže do)
12	(krajnja vrednost za N)

Petlja se realizuje na sledeći način: kada računar prvi put naiđe na petlju, brojač N se postavlja na početnu vrednost (1) i izvršavaju se sve instrukcije, dok se ne naiđe na naredbu NEXT. Nakon toga vrši se upoređivanje vrednosti broja N sa krajnjom vrednošću (12). Ukoliko je N manje od pomenute vrednosti, tada se N povećava za 1 (tj. postavlja na 2), vraća se na početak petlje i ponovo započinje izvršavanje instrukcija. Pri sledećem susretu sa naredbom NEXT ponovo se vrši upoređivanje broja N i krajnje vrednosti, N se povećava na 3; zatim na 4, 5, 6, . . . sve dok N ne dobije vrednost 12. Nakon toga napušta se petlja i prelazi se na sledeći programski red (ukoliko postoji), ili se prekida izvršenje programa (ukoliko ne postoji naredni programski red).

Pre nego što pređemo na detaljno razmatranje programa recimo nešto o liniji 50. To je zapravo niz PRINT naredbi koje daju prikaz poput:

1 X 7 = 7

Razmaci ▽ služe za preglednije prikazivanje rezultata. Pojedinačni prikazi počinju vrednošću 1 za brojač N i pojavljuju se u obliku:

PRINT N	PRINT 1	1
;	bez pomeranja	
"X▽7▽="	PRINT "X▽7▽="	1 X 7 =
;	bez pomeranja	
7*N	PRINT 7*1	1 X 7 = 7
(nema simbola ;)	prelaz na sledeći red	

Napomenimo da se u naredbi PRINT N ne koriste znaci navoda. Ukoliko napišete PRINT "N", tada će biti ispisano slovo N. Za slučaj da nisu naznačeni znaci navoda biće ispisana numerička vrednost dodeljena N-u. Pošto broj N počinje vrednošću 1 i ide sve do vrednosti 12, naredbom PRINT N će biti ispisani brojevi 1, 2, 3, . . . , 12 u zavisnosti od stanja brojača u petlji.

Slično, 7*N uzima vrednosti 7*1=7; 7*2=14; . . . ; 7*12=84, pa će se ispisati i ovi brojevi.

Sada možemo preći na analiziranje programa i prikazivanje dobijenih rezultata.

10 PRINT CHR$(147)	Brisanje ekrana
20 PRINT "TABLICA MNOZENJA SA 7"	TABLICA MNOZENJA SA 7
30 PRINT	Ispisivanje praznog reda u cilju ostavljanja jednog praznog prostora ispod zaglavlja.
40 FOR N=1 TO 12	Postavljanje petlje pri čemu je N brojač koji uzima vrednosti od 1 (početna vrednost) do 12 (krajnja vrednost).
50 PRINT N; "X▽7▽="; 7*N	1 X 7 = 7
60 NEXT N	Ispitivanje da li je N=12, čime se obezbeđuje izlaz iz petlje. N se uvećava za 1 tj. dobija vrednost 2 i vrši se povratak na red 50.
50 PRINT N; "X▽7▽="; 7*N	2 X 7 = 14
60 NEXT N	Ispitivanje da li je N=12. U ovom trenutku to nije, jer je N=2. Brojač N se uvećava za 1 tj. dobija vrednost 3 i vrši se povratak na red 50. Dalje se ponavlja pomenuti postupak . . .
60 NEXT N	Brojač N je dobio vrednost 12 čime se obezbeđuje izlaz iz petlje. Pošto nema više komandi prekida se dalje izvršavanje programa.

Vežba 1

Izvršite izmene u redovima 20 i 50 tako da računar štampa:

1. Tablicu množenja brojem 5;
2. Tablicu množenja brojem 9;

a za one ambicioznije,

3. Tablicu množenja brojem 99.

(Uputstvo: zameni 7-icu sa 5, 9 ili 99)

VELIČINA KORAKA

Kada napišete naredbu poput:

FOR R=3 TO 14

računar *podrazumeva* brojanje po 1:

3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14

Veličinu koraka možete promeniti ukoliko koristite instrukciju STEP. Da bi se dobili iznosi koji su za dva veći jedan od drugog potrebno je upisati naredbu:

```
FOR R=3 TO 14 STEP 2
```

U ovom slučaju R dobija vrednosti:

3, 5, 7, 9, 11, 13

(u prethodnom nizu se ne može pojaviti broj 15, jer je veći od broja 14 koji je naznačen kao gornja granica u datoj naredbi. Međutim, ukoliko napišete:

```
FOR R=3 TO 15 STEP 2
```

tada će R dobiti vrednosti:

3, 5, 7, 9, 11, 13, 15

pošto je 15 najveća dozvoljena vrednost u naredbi FOR).

Vežba 2

U programu TABLICA MNOŽENJA BROJEM 7, izmenite red 40 tako da se štampaju samo proizvodi *parnih* brojeva sa brojem 7.

Vežba 3

Slično kao Vežba 2, ali je reč o *neparnim* brojevima. Šta više, moguće je štampati brojeve u opadajućem redosledu, koristeći za korak *negativnu* vrednost:

```
10 PRINT CHR$(147)
20 FOR I=10 TO 0 STEP -1
30 PRINT I
40 NEXT I
50 PRINT "UZLETELI SMO!"
```

PAUZA

Ponekad je potrebno privremeno zaustaviti izvršenje programa, npr. da bi se pročitao sadržaj ekrana. To se može realizovati korišćenjem "prazne petlje" poput ove:

```
500 FOR I=1 TO 100
510 NEXT I
```

Na ovaj način se troši vreme za prebrojavanje od 1 do 100. Ovo se može zapisati i sa:

```
FOR I=1 TO 100: NEXT I
```

čime se štedi memorija.

Vreme utrošeno na ovaj način zavisi od veličine petlje. Za duže pauze potrebno je broj 100 zameniti nekom većom vrednošću, za kraće pauze ovaj broj treba smanjiti. Ukoliko se petlja poveća za 1000 ciklusa tada se vreme za njeno izvršenje povećava za približno jednu sekundu. Shodno tome, za izvršenje jedne petlje veličine 100 utroši se približno jedna desetina sekunde.

Skraćeni zapis

U bilo kojoj naredbi NEXT brojač petlje može biti *izostavljen*. Stoga umesto NEXT I ili NEXT R možete pisati samo NEXT. Na ovaj način se štedi memorijski prostor, ali se gubi na preglednosti programa, jer se ne zna koja se NEXT naredba na koji brojač odnosi (računar, međutim, zna).

ODGOVORI

Vežba 1

1.
```
20 PRINT "TABLICA MNOZENJA BROJEM 5"
50 PRINT N; "X 5 ="; 5*N
```
2.
```
20 PRINT "TABLICA MNOZENJA BROJEM 9"
50 PRINT N; "X 9 ="; 9*N
```
3.
```
20 PRINT "TABLICA MNOZENJA BROJEM 99"
50 PRINT N; "X 99 ="; 99*N
```

Vežba 2

```
40 FOR N=2 TO 12 STEP 2
```

Vežba 3

```
40 FOR N=1 TO 11 STEP 2
```

ili:

```
40 FOR N=1 TO 12 STEP 2
```

(Oba reda daju isti rezultat. Nemojte zaboraviti da upišete i ostale redove programa).

Jedna od prvih stvari kojom treba ovladati je: kako ubediti računar da na ekranu televizora napiše ono što želite.

7 Izlaz na ekran

Računar nije od velike koristi ako samo stoji, zabavlja se sa samim sobom i odbija da komunicira sa spoljnim svetom. Već smo videli neke od komunikacija u radu: tastaturu (koja radi kao *ulazni* uređaj dovodeći informacije u mašinu) i ekran televizora (*izlazni* uređaj koji ih prikazuje). Ulaz je način na koji mi govorimo računaru, a izlaz način na koji se on obraća nama. U ovom poglavlju ću reći nešto više o izlazu na televizor, a o ulazu će biti govora u poglavlju 9.

BROJEVI I ZNAKOVNI NIZOVI

"Šezdesetčetvorka" može da obrađuje dva različita tipa informacija: *brojeve* i *znakovne nizove*. Brojevi se prikazuju u uobičajenom obliku, eventualno sa znakom minus i decimalnom tačkom kao:

```
25
—999
76.3332
```

i tako dalje. Znakovni nizovi su baš nizovi simbola ili *znakova* posmatrani kao zasebne celine. Da bi se to naglasilo oni se smeštaju između navodnika:

```
"MORTADELA"
"KVAKA 22"
"*** KRAJ ***"
"%&4999BCXXX/**DJUBRE+++>"
```

i slično.

PRINT

Ovo je osnovna izlazna naredba. Čak i ako imate štampač, nemojte naredbu shvatiti doslovno (engl. print = štampaj), ona daje izlaz na televizor, a ne odštampanu listu, Naredba PRINT radi na nešto različite načine zavisno od toga šta se

prikazuje. Za prikazivanje *broja* kao što je, npr. 42, upotrebićemo programski red, kao:

```
430 PRINT 42
```

Da bismo odštampali *znakovni niz*, koristimo navodnike:

```
440 PRINT "MORTADELA"
```

Ako isprobate ove "programe" (ne zaboravite RETURN) naredbom RUN, utvrdićete da na ekranu televizora prvi daje:

```
42
```

a drugi (kao što se moglo i očekivati):

```
MORTADELA
```

Razlika je u tome što se *navodnici oko znakovnog niza* ne ispisuju. U stvari,

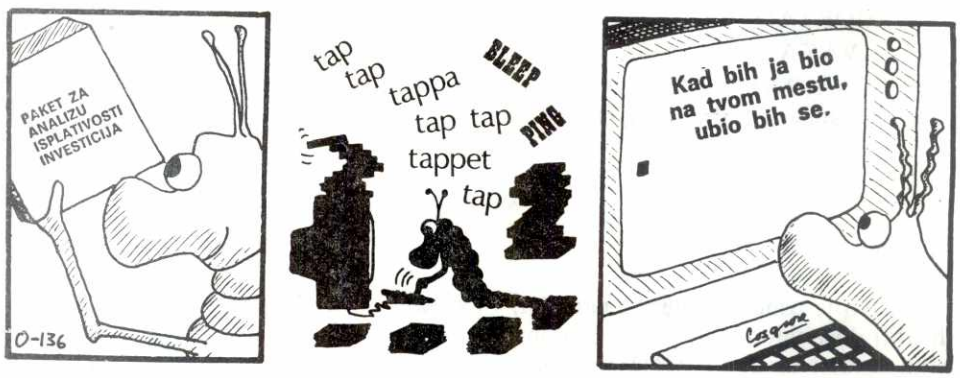
```
PRINT 42
PRINT "42"
```

daju isti rezultat. Ako vam se čini da je ceo posao besmislen, pokušajte sledeće:

Vežba 1

Uporedite rezultate sledećih programa od jednog reda:

1. 10 PRINT 6 * 7
   i
   10 PRINT "6 * 7"
2. 20 PRINT 40 + 2
   i
   20 PRINT "40 + 2"



3. 30 PRINT "ODGOVOR NA TO VELIKO PITANJE ZIVOTA, UNIVERZUMA I SVEGA"

i

30 PRINT ODGOVOR NA TO VELIKO PITANJE ZIVOTA, UNIVERZUMA I SVEGA

Razlika između ispisivanja brojeva i znakovnih nizova postaje još važnija kad želite da ispišete vrednosti *promenljivih*. (Vidi poglavlje 8.)

Skraćivanje

Umesto kucanja reči:

PRINT

možete da koristite upitnik:

?

Tako, npr. umesto PRINT "MENI", možete da otkucate:

? "MENI"

Ovim se štedi memorija i vreme kucanja nauštrb brzog razumevanja. Savetujem vam da prvo koristite PRINT, a tek kasnije da pređete na znak pitanja. Zbog lakšeg razumevanja u ovoj knjizi sam uvek koristio PRINT. "Šezdesetčetvorka" i onako raspolaže sa velikom količinom memorije.

ISPISIVANJE I INTERPUNKCIJA

Iza naredbe PRINT može da se koristi jedan od sledeća tri znaka:

1. tačka-zarez (;)
2. zarez (,)
3. ništa

Odgovarajući primeri bi bili:

1. 100 PRINT "OVO VEOMA USRECUJE POKORNI RACUNAR";
2. 110 PRINT "OVO VEOMA USRECUJE POKORNI RACUNAR",
3. 120 PRINT "OVO VEOMA USRECUJE POKORNI RACUNAR"

Znaci interpunkcije *umeću prazna mesta* iza onoga što je ispisano i tako se podešava položaj za sledeće ispisivanje. Oni različito utiču na brojeve i znakovne nizove (to je zbog pogodnosti rada, ali sam ja lično skeptičan):

	Brojevi	Znakovni nizovi
tačka zarez	— ostavlja jedno prazno mesto	ne ostavlja prazna mesta
zarez	— pomera se u prvu slobodnu kolonu od 0, 10, 20, 30	pomera se u prvu slobodnu kolonu od 1, 11, 21, 31
ništa	— pomera se u sledeći red	pomera se u sledeći red

Kad smo već tu, navešću da samo:

PRINT

ispisuje prazan red i prelazi u sledeći — kao vraćanje kolica i novi red kod pisaće mašine. U računarskom žargonu ova operacija zove se NEWLINE (novi red).

Razlog za uvođenje ovih znakova interpunkcije je u tome što pomoću njih možete ispisati različite oblike teksta na ekranu televizora. Glavna karakteristika je da se u jednoj naredbi PRINT može navesti više "stavki" uz uslov da su odvojene znakovima interpunkcije. Možete napisati, na primer:

```
500 PRINT "FRED", "LAURA"; 77, 4.22; "ROGER ▽ I ▽ KRAJ"
```

i na ekranu ćete dobiti:

```
FRED ▽▽▽▽▽▽ LAURA ▽ 77 ▽ 4.22 ROGER ▽ I ▽ KRAJ
```

Pa sad, ovo nije baš ne znam kako lepo, ali će sledeća vežba dati mnogo lepše rezultate.

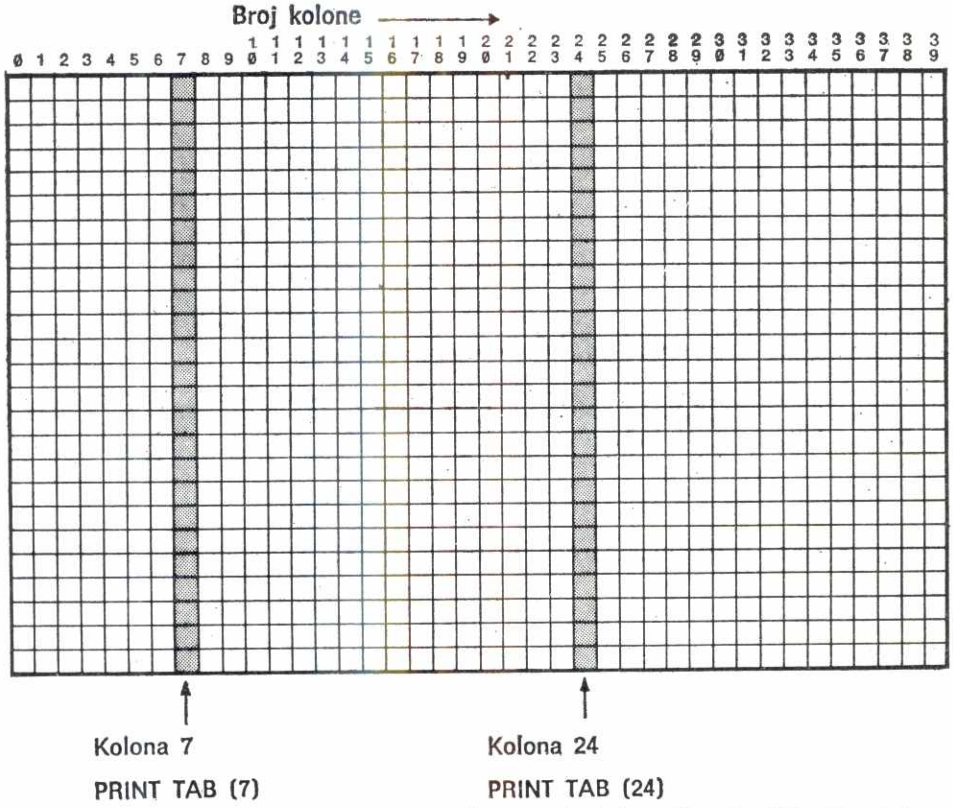
Vežba 2

Uporedite rezultate sledećih programa:

1. 10 PRINT 1
   20 PRINT 2
   30 PRINT 3
   40 PRINT 4
2. 10 PRINT 1, 2, 3, 4
3. 10 PRINT 1; 2; 3; 4
4. 10 PRINT 1,, 2,, 3,, 4
5. 10 PRINT 1,, 2,, 3
   20 PRINT 4,, 5,, 6
6. 10 PRINT 1,, 2,, 3,,
   20 PRINT 4,, 5,, 6
7. 10 PRINT 1,, 2,, 3
   20 PRINT
   30 PRINT 4,, 5,, 6
8. Probajte programe 1 — 7 još jednom, ali brojeve 1, 2 . . . 6 zamenite znakovnim nizovima "A", "B" . . . . . "F"
9. Kao (8), ali sad koristite znakovne nizove različitih dužina kao "MORTADELA", "*", "DR", "PAS", "KUPUS", itd.

TAB

Ova komanda se koristi za *tabulaciju* izlaza u određene kolone. Ekranski prikaz "šezdesetčetvorke" sastoji se od 25 *redova*, a svaki red sadrži 40 znakova. Znači, možete da zamislite da je ekran podeljen u 40 *kolona* (slika 7.1). Kolone su označene brojevima od 0 — 39.



Sl. 7.1 — Ekranski prikaz ima 25 redova, svaki sa po 40 kolona. Naredba TAB bira kolonu u kojoj se ispisuje

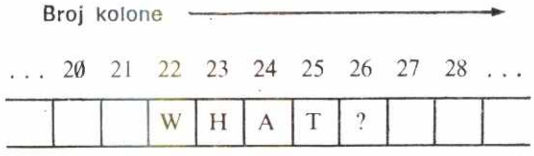
Naredba:

TAB (C);

u naredbi PRINT izazvaće ispisivanje koje počinje u sledećoj slobodnoj koloni C. (Ako ste već prošli kolonu C, ispisivanje će biti u sledećem redu — znači, ne uvek onako kako ste očekivali.) Na primer:

PRINT TAB (22); "WHAT?"

daće rezultat prikazan na slici 7.2.



Sl. 7.2 — Učinak naredbe PRINT TAB (22); WHAT?"

Tipična upotreba naredbe TAB data je u sledećem kratkom programu:

```
10 PRINT "BROJ"; TAB (10); "KVADRAT"; "TAB (20); "KUB"; TAB (30);
   "4.STEPEN"
20 PRINT
30 FOR T = 1 TO 20
40 PRINT T; TAB (10); T * T; TAB (20); T * T * T; TAB (30);
   T * T * T * T
50 NEXT T
```

UPRAVLJAČKI ZNAKOVI MEĐU NAVODNICIMA

Isprobajte ovaj program. Ne vodite računa o tome šta se prethodno nalazilo na ekranu. Otkucajte na ekranu bilo šta, a zatim:

```
10 PRINT"
```

a zatim pritisnite jednovremeno tastere SHIFT i CLR/HOME pa na kraju otkucajte:

"

pa pritisnite taster RETURN.

Sada otkucajte RUN.

Utvrdićete da se ekran izbrisao i da je pokazivač u polaznom položaju — baš kao da ste pritisnuli SHIFT + CLR/HOME kao direktnu naredbu.

Neke direktne naredbe, koje se na taj način mogu dobiti sa tastature, mogu se istim sredstvima izvršavati i iz programa. Na primer, ako pritisnete CTRL + 6 umesto SHIFT + CLR/HOME u navedenom primeru, utvrdićete da se posle izvršenja programa boja znakova promenila u zelenu.

Ova tehnika zove se "korišćenje upravljačkih znakova", ili *navodnički način rada* (quote mode). Upravljački znak se ne prikazuje nego *izvršava*.

Da bi vas podsetila da tu *ima* neki upravljački znak, "šezdesetčetvorka" *prikazuje* nešto u listi programa. Međutim, ono što ona prikazuje je samo proizvoljno izabran simbol pridružen upravljačkom znaku u cilju podsećanja. Za naredbu CLEAR (tj. SHIFT + CLR/HOME) ovaj znak je simbol herca iz špila karata prikazan *inverzno*. Za CTRL + 6 koristi se drugi znak — opet prikazan inverzno.

To je korisna tehnika, ali nepogodna za programske liste (pogledajte u časopisima — neke liste su skoro nečitke zbog toga). Zbog toga ću ja u ovoj knjizi *izbegavati* taj način rada, koliko god je to moguće, zamenjujući upravljačke znakove njihovim *kodovanim* oblikom:

```
CHR$ (K)
```

gde je K kodni broj, odnosno ASCII — kôd datog znaka (vidi poglavlje 18). Kodovi su izlistani u dodatku F *Priručnika* na stranama 135—137. Ako pogledamo listu videćemo, npr. da se CLR/HOME javlja dva puta, sa kodovima 19 i 147. Prvi kôd odgovara verziji bez SHIFT-a (pokazivač se vraća u polazni položaj), a drugi ver-

ziji uz jednovremeno pritiskanje tastera SHIFT (brisanje ekrana i vraćanje pokazivača u polazni položaj). Tako:

```
PRINT CHR$ (147)
```

ima isti učinak kao inverzni herc u navodnicima iz prethodnog primera. Slično se i zelena boja znakova može postići pomoću:

```
PRINT CHR$ (30)
```

Ne preporučujem vam da suviše razbijate glavu oko toga šta možete da učinite sa ovim o čemu je bilo reči, samo imajte na umu da se PRINT CHR$ (K) često koristi da bi se dobio neki sličan učinak iz programa.

PISANJE U IZABRANOM REDU

Upotreba upravljačkih znakova omogućava da naterate računar da ispiše znak na bilo kom mestu ekrana koje ste izabrali. Već sam pokazao kako se naredbom TAB može izabrati kolona. Ali kako izabrati red?

Evo jednog od načina. Prvo koriste CHR$ (19) da vratite pokazivač u polazni položaj, a da *ne* izbrišete ekran. Zatim koristite petlju sa nekoliko PRINT CHR$ (17) da biste pokazivač pomerili dole (to je kôd upravljačkog znaka za CURSOR DOWN = pokazivač dole). Da biste ispisali tekst na početku 15. reda, koristite:

```
500 PRINT CHR$ (19);
510 FOR T = 1 TO 15: PRINT CHR$ (17); : NEXT T
520 PRINT "OVO JE U 15. REDU"
```

Obratite pažnju na tačke-zareze. Bez njih bi bio preskočen mnogo veći broj redova. Ako želite da ispišete tekst počev od 8. kolone 15. reda, izmenite red 520 u:

```
520 PRINT TAB (8); "OVO JE RED 15 KOLONA 8"
```

Isti učinak se može dobiti kucanjem upravljačkih znakova među navodnicima u naredbi PRINT ". . .". Postoji i sasvim drugačiji pristup, koji uopšte ne koristi upravljačke znakove, a koji će biti opisan u poglavlju 19. Meni je on najdraži, ali vaš ukus može da bude drugačiji.

ODGOVORI

Vežba 1

1. 42
   6 * 7
2. 42
   40 + 2
3. ODGOVOR NA TO VELIKO PITANJE ZIVOTA, UNIVERZUMA I SVEGA
   ? SYNTAX ERROR

(Ovde je reč TO dovela do greške u sintaksi. ODGOVOR NA je smatran promenljivom)

Vežba 2

1. ▽1
   ▽2
   ▽3
   ▽4
2. ▽1▽▽▽▽▽▽▽▽▽2▽▽▽▽▽▽▽▽▽3▽▽▽▽▽▽▽▽▽4
3. ▽1▽▽2▽▽3▽▽4
4. ▽1▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽2
   ▽3▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽4
5. ▽1▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽2
   ▽3
   ▽4▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽5
   ▽6
6. ▽1▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽2
   ▽3▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽4
   ▽5▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽6
7. ▽1▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽2
   ▽3
   (prazan red)
   ▽4▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽5
   ▽6
8. Kao u prethodnom slučaju, ali za jedno mesto levo.
9. Kao pod (8) u pogledu početnih položaja znakovnih nizova.

U mnogim programima potrebno je da uzmete neke brojeve i obradite ih na odredeni način, ali i sami brojevi mogu da se promene. Vreme je, dakle, da predemo na:

8 Promenljive

Mnogi programi uključuju uobičajeno manipulisanje raznim podacima. Program, koji treba da analizira stabilnost nosećih mostova, sadržaće uglavnom numeričke podatke, a program za obradu teksta u kancelariji obrađivaće tekstualne podatke kao što su nizovi slova, razmaci i drugi simboli. Standardni način da se oni obrade BASIC-om su *promenljive*. Promenljiva se naziva nekim imenom koje bira programer, a služi za rezervisanje mesta za podatke. Stvarni podaci definišu *vrednosti* koje promenljiva poprima. Kao što ćemo videti, sve je mnogo jasnije u primeru nego u tekstu. Navodimo tipičan slučaj:

```
10 FOR T = 0 TO 1000
20 PRINT T; "NA KVADRAT JE", T * T
30 NEXT T
```

Ovaj program izračunava kvadrat T * T broja T za vrednosti 0 do 1000. Kažemo da je T numerička promenljiva. Ona ima nepromenljivo ime, u ovom slučaju T, i vrednost koja može da se menja u toku izvršavanja programa. U našem primeru vrednost počinje od 0, pa se povećava na 1 itd. sve dok ne dostigne krajnju vrednost 1000.

"Šezdesetčetvorka" radi sa dva glavna tipa promenljivih.

1. *Numeričke* promenljive koje sadrže brojeve kao svoje vrednosti.
2. *Znakovne* promenljive koje sadrže znakovne nizove.

Naziv promenljive mora da počne slovom, a može da se sastoji od slova i brojeva (grafički znakovi se ovde smatraju slovima). Tipični nazivi numeričkih promenljivih su:

A	B	C	ALFA	A5
TEZINA	VELICINA	PERA	JELENA	X233

i tako dalje. Znakovna promenljiva mora da ima znak "$" na kraju:

A$	B$	C$	ALFA$	A5$
TEZINA$	VELICINA$	PERA$	JELENA$	X233$

Prilikom izbora naziva promenljivih treba imati na umu dve stvari:

1. Računar prepoznaje samo prva dva znaka naziva (iako su dozvoljeni i duži nazivi — sve do oko 80 znakova). Tako se:

PR PRVI PROBA PRX82

interpretiraju kao da znače isto. To može da bude nezgodno, ako koristite nazive koji vas podsećaju na funkciju promenljive. Npr. u programu za obračun ličnog dohotka će:

IZNOS i
IZVOD

biti tretirani u računaru kao ista stvar (IZ). Međutim, programer u stvaralačkom žaru može ovo da zaboravi i da bude zbunjen dobijenim rezultatima.

2. Ako naziv promenljive bilo gde sadrži ključnu reč BASIC-a, kao što su PRINT, LIST, TO, FOR, itd. sistem neće dozvoliti upotrebu takve promenljive (iako ga interesuju samo prva dva znaka. Razlog za to je u tome što on prvo traži ključne reči). Znači da ne možete da koristite nijedan od sledećih naziva promenljivih, jer sadrže ključne reči (podvučeno):

TOTAL SEMAFOR SPORT$ PANDA STABILAN$

a ono što mi najviše smeta, moj stari omiljeni naziv:

FRED

je nevažeći, jer sadrži ključnu reč FRE koja kaže koliko je prostora preostalo u memoriji. (Vidi poglavlje 12.)

U nekim slučajevima sistem može pogrešno da interpretira ono što ste zamislili, a ne samo da prigovori. Pogodite šta će uraditi ovaj program:

```
10 LETVA$ = "A"
20 PRINT LETVA$
```

Dobićete poruku o grešci. Izmenite red 20 na:

```
20 PRINT VA$
```

pa će računar ispisati slovo A. Red 10 je interpretiran kao naredba:

```
10 LET VA$ = "A"
```

koja se poziva na promenljivu sa nazivom VA$. Veselo. Ali dokgod pažljivo izbegavate ključne reči BASIC-a, nema problema.

Vežba 1

Dva od navedenih naziva su važeći, a ostali nisu. Koji i zašto? (Savet: koristite naredbu PRINT)

#TRI	VALENTINA	RATNIK	SLON	221B PEKAR
ULICA	TREND	PECOS	BAGET	SPOSOBNOST
VERANDA	NOTA	PRORED	STIROPOR	PREMISA
DRNDANJE				

DODELA VREDNOSTI PROMENLJIVIM

Programer mora da kaže računaru koju vrednost da dodeli promenljivoj. (Ako ne znate, za numeričke promenljive on podrazumeva vrednost 0, a za znakovne nešto što se zove *prazan niz* odnosno znakovni niz koji ne sadrži nijedan znak.) Vrednosti se dodeljuju naredbom:

```
LET
```

kao, na primer:

```
100 LET TEZINA = 77
110 LET IME$ = "PANTA"
```

Da bi se uštedeo prostor, reč LET se može izostaviti (i obično se izostavlja):

```
100 TEZINA = 77
110 IME$ = "PANTA"
```

Obratite pažnju na navodnike oko reči PANTA. Navodnici su neophodni, ako dodeljujete vrednost *znakovnoj* promenljivoj. Navodnici nisu deo znakovne promenljive, nego samo ukazuju na to gde počinje i gde se završava znakovni niz. Za detaljnija objašnjenja znakovnih nizova pogledajte poglavlje 16.

Promenljivoj se vrednost može dodeliti i indirektno izražavajući vrednost kao kombinaciju već definisanih promenljivih:

```
10 A = 36
20 B = 5
30 K = 10 * A + B
```

dodeljuje promenljivoj K vrednost 10 * 36 + 5, tj. 365. Dajemo primer programa koji množi dva broja, 77 i 88:

```
10 A = 77
20 B = 88
30 K = A * B
40 PRINT K
```

Linije 10 i 20 možete da promenite tako da množite i neke druge brojeve. Naravno, postoji i kraći put da se postigne to isto:

```
10 K = 77 * 88
20 PRINT K
```

ili samo:

```
10 PRINT 77 * 88
```

ali to je samo zato što sam izabrao neuobičajeno prost primer.

Vežba 2

U samoposluzi je cena ulja 269 dinara, brašna 90 dinara, a lizalica za decu 6 dinara. Koristite tri promenljive ULJE, BRASNO i LIZ i napravite program koji će izračunati koliko staju 6 litara ulja, 5 kilograma brašna i 29 lizalica. Ovu promenljivu nazovite IZNOS.

Vežba 3

Promenom samo jednog reda u programu izračunajte koliko staje jedanaest litara ulja, četrnaest kilograma brašna i tri lizalice.

Vežba 4

Izmenom još tri reda u vežbi 2, izračunajte iznos, ako se cene promene na 386 za ulje, 106 za brašno i 3 za lizalice (istekao im rok upotrebe).

ODGOVORI

Vežba 1

Važeći nazivi su PEKAR i ULICA. Nazivi #TRI i 221B ne počinju slovom. Ostale sadrže ključne reči BASIC-a: VALENTINA, RATNIK, SLON, TREND, PECOS, BAGET, SPOSOBNOST, VERANDA, NOTA, PRORED, STIROPOR, PREMISA, DRNDANJE.

Vežba 2

```
10 PRINT CHR$ (147)
20 ULJE = 269
30 BRASNO = 90
40 LIZ = 6
50 IZNOS = 6 * ULJE + 5 * BRASNO + 29 * LIZ
60 PRINT "UKUPAN IZNOS JE"; IZNOS; "DINARA"
```

(U redu 60 nisam eksplicitno naveo gde treba da budu razmaci; od sada ću to činiti uvek kad je jasno gde treba da budu)

Vežba 3

Promenite red 50 u:

```
50 IZNOS = 11 * ULJE + 14 * BRASNO + 3 * LIZ
```

Vežba 4

Promenite redove 20 do 40 u:

```
20 ULJE = 386
30 BRASNO = 106
40 LIZ = 3
```

Ako želite da unosite informacije u računar samo mu recite da vas podseti, ukoliko mu je nešto potrebno.

9 Ulazi

Vežbe 2—4 u prethodnom poglavlju obavljaju posao. Međutim, zamorno je menjati programske redove uvek kada treba promeniti vrednost promenljive. Na sreću to nije neophodno zahvaljujući naredbi:

INPUT

koja omogućava dodeljivanje vrednosti sa tastature. Na primer:

```
10 PRINT "UPISITE BROJ"
20 INPUT N
30 PRINT "UPISANI BROJ JE ";N
```

Pri izvršavanju ovog programa na ekranu se pojavljuje poruka UPISITE BROJ i nakon toga znak pitanja (?). Ukoliko sada sa tastature unesete broj, a zatim pritisnete taster RETURN, tada će se izvršiti red 30 u programu tj. na ekranu će se pojaviti poruka UPISANI BROJ JE, a u njenom nastavku upisani broj.

VIŠESTRUKI ULAZI

U naredbi INPUT može se pojaviti više promenljivih koje su razdvojene zarezima, kao u slučaju instrukcije PRINT. Upišite:

```
10 INPUT A, B, C
20 PRINT A, B, C
```

Pri izvršavanju programa na ekranu se pojavljuje znak pitanja (?). Upišite neki broj, npr. 3. Nakon toga na ekranu se pojavljuje ?? i od vas se očekuje da upišete drugi broj. Upišite 5. Na kraju, na ekranu se ponovo pojavljuje simbol ??. Upišite 7 kao vrednost trećeg broja. "Šezdesetčetvorka" će vam na ekranu prikazati vrednosti 3, 5 i 7 koje predstavljaju vrednosti ulaznih veličina A, B i C. Pri izvršavanju programa ekran dobija sledeći izgled:

```
RUN
?3
??5
???7
3          5          7
READY
```

POGREŠNI UPISI

Ukoliko pri unošenju podataka načinite grešku, a pri tome još niste pritisnuli taster RETURN, tada je možete ispraviti koristeći taster INST/DEL. Međutim, ako ste pritisnuli RETURN, biće problema izuzev u jednom slučaju u kojem "šezdesetčetvorka" štiti sebe i vas od jednog čestog tipa greške. Ako pokušate da numeričkoj promenljivoj dodelite podatak koji nije broj, nastaje greška i pojavljuje se poruka:

```
REDO FROM START
```

koja vam ukazuje da unos treba ponoviti.

PRIPREMNE PORUKE

Moguće je ispisati poruku koja vas podseća šta se očekuje kao ulaz (pošto znak ? nije uvek od pomoći), kao na primer:

```
10 INPUT "UPISITE BROJ" ;N
```

koja ima isti efekat kao i redovi 10 i 20 programa koji je izložen u prethodnom poglavlju. Moguće je koristiti više ulaznih veličina sa istom pripremnom porukom.

Postoji bolji način za realizaciju Vežbi 2—4 iz prethodnog poglavlja, koji daje uopšteno rešenje:

```
10 PRINT CHR$ (147)
20 INPUT "CENE: ULJE, BRASNO, LIZALICA"; UL, BR, LI
30 INPUT "KOLICINE"; KU, KB, KL
40 CENA = UL * KU + BR * KB + LI * KL
50 PRINT "UKUPAN IZNOS JE"; CENA; "DIN."
```

Da bismo listu programa učinili kraćom možemo promenljive skratiti na dužinu od dva znaka UL, BR i LI umesto starih ULJE, BRASNO i LIZ, a imamo i tri nove promenljive za količinu svake od njih: KU, KB i KL.

UREĐIVANJE IZLAZA NA EKRANU

Pretpostavimo da želite da preko tastature unesete godine starosti i broj telefona. Pokušajte najpre sa programom:

```
10 PRINT CHR$ (147)
20 INPUT "KOLIKO IMATE GODINA"; A
30 PRINT A
40 INPUT "VAS BROJ TELEFONA"; T
50 PRINT T
```

Nakon izvršenja programa ekran ima sledeći izgled:

```
KOLIKO IMATE GODINA? 17
17
VAS BROJ TELEFONA? 361005
361005
READY
```

koji nije naročito dopadljiv. Ovo možemo izmeniti na sledeći način:

```
10 PRINT CHR$ (147)
20 INPUT "KOLIKO IMATE GODINA"; A
30 INPUT "VAS BROJ TELEFONA"; T
40 PRINT CHR$ (147)
50 PRINT A, T
```

Ponekad možda ne želite da čekate do kraja programa da biste sve prikazali na ekranu. Prikladnije je rasporediti poruke tako da budu prikazane pri vrhu ekrana, a zatim ih obrisati pre prikazivanja nekih drugih poruka.

Sada dolazi red na upravljačke znakove. Setimo se da PRINT CHR$ (19) vraća pokazivač na polazni položaj, ali sadržaj ekrana ostaje nedirnut. Na taj način možemo obezbediti ispisivanje pripremnih poruka u gornjih nekoliko redova. Evo primera sa višestrukim ulazom:

```
 10 PRINT CHR$ (147)
 20 RED = 4
 30 PRINT CHR$ (19);
 40 PRINT "          [30 razmaka]          "
 50 PRINT "          [30 razmaka]          ";: PRINT CHR$ (19);
 60 INPUT "KOLIKO IMATE GODINA"; A
 70 INPUT "VAS BROJ TELEFONA"; T
 80 PRINT CHR$ (19);
 90 FOR R = 1 TO RED: PRINT CHR$ (17);: NEXT R
100 PRINT A, T
110 RED = RED + 1
120 GOTO 30
```

Red 10 briše ekran, a red 20 postavlja promenljivu koja određuje red u koji će biti ispisan sledeći izlaz na listi. Redovima 30—50 brišu se dva prva reda ekrana za prihvatanje novih ulaznih veličina. Redovima 60—70 prihvataju se nove ulazne veličine. Redovi 80—90 formiraju izlaz u redu RED. Red 100 ispisuje elemente ta-

bele, red 110 vraća promenljivu RED na vrednost 1 tako da se može preći na sledeći red, a red 120 ponavlja ceo postupak.

Malo komplikovano, ali ne i nemoguće ako malo razmislite. . .

Uzgred, red 40 znači da treba da otkucate:

```
PRINT "▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽▽
```

a slično i red 50.

Vežba 1

Napišite program koji prihvata do 15 imena i telefonskih brojeva i ispisuje ih, ali tako da izgled ekrana bude uređen.

ODGOVORI

Vežba 1

```
 10 PRINT CHR$ (147);
 20 RED = 4
 30 FOR K = 1 TO 15
 40 PRINT CHR$ (19);
 50 PRINT "            [30 razmaka]            "
 60 PRINT "            [30 razmaka]            ";: PRINT CHR$ (19);
 70 INPUT "VASE IME"; IME$
 80 INPUT "VAS BROJ TELEFONA"; T
 90 PRINT CHR$ (19);
100 FOR R = 1 TO RED: PRINT CHR$ (17);: NEXT R
110 PRINT IME$, T
120 RED = RED + 1
130 NEXT K
```

Uzgred, redove 20 i 120 možete da izostavite, a red 100 zamenite sa

```
100 FOR R = 1 TO K + 3: PRINT CHR$ (17);: NEXT R
```

pošto vam nisu potrebna dva brojača RED i K.

Smatra se da živopisni izraz „isterivanje bubica" potiče iz ranih dana kompjuterizacije kad su insekti ulazili u mašinu i izazivali kratke spojeve. Ako danas računar pogreši, to je obično greška programera. Ali da stvari odmah postavimo na pravo mesto: još uvek treba da znate kako se otklanjaju „bube" (engl. bug = buba — označava programsku grešku, prim. prev.).

10 Pronalaženje i otklanjanje grešaka I

Ne smete biti obeshrabreni, ako program ne radi ispravno kad ga prvi put pustite u rad. Čak i profesionalnim programerima to teško polazi za rukom.

Zbog toga je važno da raspolažete tehnikom koja će vam pomoći da brzo i lako pronađete greške u programu. To je ono što se zove *pronalaženje i otklanjanje grešaka.*

SINTAKSNE GREŠKE

Za početak razmotrimo onu vrstu grešaka koja se prva javlja kad pustite program u rad. Pretpostavimo, npr., da ste negde u programu napisali:

```
50 FOR N = 1—7
```

zaboravivši da morate da koristite reč TO da biste razdvojili vrednosti u FOR-petlji. "Šezdesetčetvorka" će spremno prihvatiti naredbu dok je kucate, ali ćete prilikom izvršenja programa dobiti poruku:

```
SYNTAX ERROR IN 50
```

Drugim rečima, "šezdesetčetvorci" se ne sviđa gramatička konstrukcija naredbe u redu 50. To bi bilo kao kad bih ja rekao: "Šezdesetčetiri, on ne razumeti ova naredba." Vi biste imali primedbe na moju sintaksu. Jedina razlika je u tome što *vi* možete da shvatite šta sam ja hteo da kažem svojom rečenicom, a "šezdesetčetvorka" neće ni *pokušati* da shvati red 50. Ona će ga, kao što smo već videli, jednostavno odbaciti.

Posebno kad tek učite neki programski jezik postoji verovatnoća da napravite dosta ovakvih grešaka, a veoma je dosadno otkucati šezdeset redova programa sa desetak naredbi FOR samo da biste prilikom izvršenja bili obavešteni da ste loše zapamtili konstrukciju naredbe FOR i da treba sve da ih izmenite. Zbog toga treba da steknete običaj da otkucate RUN nakon svakih nekoliko redova programa. Naravno, nezavršeni programi se verovatno neće ponašati smisleno, pa čak možete da dobijete poruke o greškama koje će nestati kad napišete sledeći deo programa, ali u ovoj fazi nas interesuje samo pronalaženje sintaksnih grešaka pre nego što njima pretrpate program.

PROMENA POGREŠNOG REDA

Recimo da u programu imate već navedenu grešku i da želite da promenite naredbu. Najjednostavniji način da to postignete je da ponovo otkucate red (setite se, ako želite da potpuno izbacite red, dovoljno je da otkucate njegov broj i pritisnete taster RETURN). Problem je ovde, naravno, što red može da bude prilično dugačak, pa je čisto rasipanje vremena, ako ponavljate već otkucano uz eventualno samo jednu izmenu.

"Šezdesetčetvorka" ima *editor* (program za uređivanje upisanih znakova, prim. prev.) koji rešava ovaj problem. On je prilično moćan, pa kao i sve moćne stvari zahteva malo vežbe da biste mogli da ga koristite u potpunosti.

Prvo, obezbedite da se red koji želite da uredite, pojavi bilo gde na ekranu (nije važno gde). Ako red nije na ekranu, otkucajte:

LIST 50 (za naš primer).

Sada pomoću tastera za pomeranje pokazivača (desno u donjem redu tastature) pomerite pokazivač na broj 5 tako da dobijete:

[5] 0 FOR N = 1—7

Da biste to učinili pritisnite taster SHIFT, držite ga i u isto vreme pritisnite *unutrašnji* taster za pomeranje pokazivača. Ako samo pritisnete taster na sekund, videćete da se pokazivač pomerio za jedan red naviše. Ako taster držite duže pritisnut, proces će početi brzo da se ponavlja i, dok ne uvežbate, utvrdićete da često prebacujete metu. Nije važno, da biste pokazivač pomerili dole, samo pritisnite isti taster, ali bez pritiskanja tastera SHIFT.

Sad koristite drugi taster za pomeranje pokazivača (bez tastera SHIFT) da pokazivač dovedete na znak "—":

50 FOR N = 1 [—] 7

Otkucajte "T" i ono će zameniti znak "—":

50 FOR N = 1T [7]

Ako sada otkucate "0", ono će zameniti znak "7" što ne bi mnogo smetalo u ovom slučaju pošto ga možete ponovo otkucati, ali šta kad bi iza pokazivača bilo još 20 znakova? "Šezdesetčetvorka" nudi tajni način rešavanja problema tako što dozvoljava *produženje* reda umetanjem razmaka preko kojih se zatim može pisati. Desno na gornjem delu tastature postoji taster za *umetanje* (INST = insert). Primetićete da se natpis INST nalazi na gornjoj polovini tastera što znači da treba pritisnuti taster SHIFT (inače taster briše, DELete = brisanje).

Kad ste namestili pokazivač u položaj koji vam treba, pritisnite taster INST pa će se između znakova "T" i "7" pojaviti razmak:

50 FOR N = 1T □ 7

u koji možete da upišete "0":

50 FOR N = 1 TO 7

Na kraju pritisnite RETURN i posao je završen.

Naravno, duže traje ovo objašnjenje nego što je potrebno vreme da se to uradi, a posebno kad malo uvežbate.

Vežba 1

Evo nekoliko primera za vežbu. Svaki red ima sintaksnu grešku. Pokušajte da otkucate svaku naredbu, utvrdite u čemu je greška i zatim pomoću editora otklonite grešku. Konačno, otkucajte RUN da biste bili sigurni da više nema poruke? SYNTAX ERROR (možda ćete dobiti neku drugu poruku greške, ali u ovom trenutku obratite pažnju samo na sintaksne greške).

1. 20 IMPUT A
2. 30 B * C = F
3. 40 INPUT "UPISI VREDNOST", V
4. 50 IF P = 1 : PRINT "P = 1"
5. 60 A = 3; B = 7; C = 10
6. 70 TOP = 4

ODGOVORI

Vežba 1

1. Ova naredba treba da izgleda ovako:

20 INPUT A

Da biste je izmenili pomerite pokazivač na "M", otkucajte "N" i pritisnite RETURN. Lako!

2. Ova naredba je napisana obratno. Ona treba da glasi:

30 F = B * C

U ovakvim slučajevima verovatno je lakše ponovo otkucati red nego koristiti editor.

Međutim, radi vežbe uradite sledeće: Pomerite pokazivač tako da dođe na "B" i umetnite dva razmaka. U njih upišite "F = ". Sada pomerite pokazivač iza "F" i koristite taster DEL da izbrišete "F", a zatim i " = ". Pritisnite RETURN kao i obično.

3. Umesto zareza treba staviti tačku-zarez:

40 INPUT "UPISI VREDNOST"; V

Znači, u pitanju je najobičnija zamena kao u primeru 1.

4. Ova naredba treba da glasi:

50 IF P = 1 THEN PRINT "P = 1"

(vidi poglavlje 11) pa morate ":" da zamenite sa "T", umetnete tri razmaka i u njih upišete "HEN".

5. Umesto znakova ";" treba da budu ":" pa je u pitanju prosta zamena.

6. Ovaj primer vas je verovatno zbunio. Čini se da nema ničeg lošeg u tome da se "TOP" koristi kao naziv promenljive ali, na žalost, taj naziv sadrži ključnu reč "TO" (kao u naredbi FOR N = 1 TO 20) na početku (vidi poglavlje 8). Zbog toga se "šezdesetčetvorka" zbunjuje pošto pokušava da smisli naredbu koja počinje sa "TO", a računar očekuje da prvo naiđe na "FOR". Treba da promenite naziv promenljive u, npr. "VRH". Ovaj problem se javlja prilično često i treba biti svestan te mogućnosti, pošto inače može veoma da zbuni. Ako dobijete poruku o sintaksnoj grešci bez vidljivog razloga, pokušajte da promenite naziv promenljive.

Primedba: Kad koristite editor, pokazivač verovatno *neće* biti u praznom redu. Kad otkucate RUN, samo zamenjujete prva tri znaka u redu tako da će, ako je u redu pisalo, na primer:

READY

on sada biti:

RUNDY

pa kad pritisnete taster RETURN, BASIC će pokušati da razume RUNDY, pa pošto on to ne može, dobićete drugu poruku greške. Najsigurnije je da posle rada sa editorom izbrišete ekran.

Ponekad računar treba da izvršava razne zadatke u različitim uslovima. To se postiže grananjem.

11 Grananje

Već smo upoznali dve naredbe koje utiču na redosled po kojem računar izvršava program, a to su GOTO koja ga upućuje na određeni red i FOR . . . NEXT koja ga vodi kroz petlju. Ove naredbe su *suviše* pravilne za programe koji treba različito da reaguju u raznim okolnostima.

Jedan način da se program navede na *grananje* u skladu sa okolnostima je naredba:

IF . . . THEN . . .

Evo primera:

BANKOVNI SALDO

Ovo je veoma jednostavan primer programa "iz prakse". On će izračunavati bankovni saldo, ako navedete prethodni saldo i upišete uplate i isplate. Da bi program bio jednostavan, nisam vodio računa o mestu ulaznih poruka, ali ih vi možete malo urediti ako želite.

```
 10 PRINT CHR$(147)
 20 PRINT "PRETHODNI SALDO JE";
 30 INPUT B
 40 PRINT "UPISITE ISPLATE"
 50 INPUT D
 60 IF D<0 THEN 90
 70 LET B=B-D
 80 GOTO 50
 90 PRINT "UPISITE UPLATE"
100 INPUT D
110 IF D<0 THEN 140
120 LET B=B+D
130 GOTO 100
140 PRINT "TRENUTNI SALDO JE"; B
```

Ključni redovi su 60 i 110. Da bismo videli kako oni rade, potrebno je da znate da znak "<" znači *manje* od. Tako *uslov* D<0 znači "D je manje od nule". To nije broj, tako da nema numeričku vrednost nego logički izraz koji je ili *istinit* ili *lažan* (zavisno od toga da li je D *stvarno* manje od nule ili ne). Na primer:

$3 < 0$ je *lažno*
$-7 < 0$ je *istinito*

Tipična naredba IF . . . THEN ima oblik:

IF uslov THEN naredba

Ako je uslov istinit, naredba se izvršava. Međutim, ako je uslov lažan, *program prelazi na sledeći red.* Tako red 60, npr. upućuje program na red 90, ako je D negativno, a ako D nije negativno tada program nastavlja sa redom 70.

Da vidimo sada kako program radi. Trebalo bi da je do reda 50 sve dovoljno jasno. Pretpostavimo da imate tri isplate od 1850, 3260 i 820 dinara. Kad vas računar prvi put upita za iznos D, upisujete prvi iznos.

? 1850

Pošto uslov D<0 nije ispunjen, program izvršava redove 70 i 80 koji ga šalju *na ponovni ulaz* na red 50. Tako sada upisujemo sledeću isplatu:

? 3260

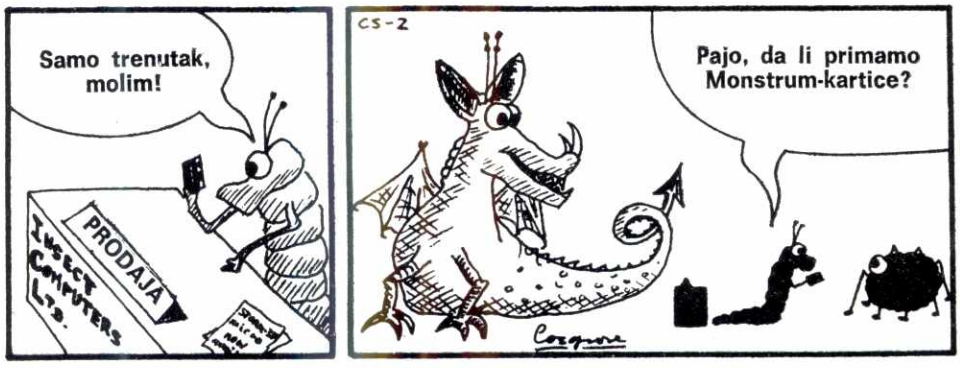
a zatim treću:

? 820

ali računar traži još:

?

tako da nešto treba da uradite. E to se rešava pomoću D<0. Upišite negativnu vred-



nost, recimo —1. Sada program prelazi na red 90, pošto je uslov D<0 *ispunjen*. Ceo postupak se sada ponavlja za uplate, a upisivanje podataka se završava kad ponovo upišete —1. Korišćenje besmislene vrednosti, kao što je —1, za kontrolu grananja je uobičajeni trik. Kažemo da —1 deluje kao *ograničavač*, pošto on mašini daje signal "kraj upisivanja".

U međuvremenu su redovi 70 i 120 oduzeli isplate i dodali uplate tako da smo dobili novi saldo.

Vežba 1

Napišite program za sabiranje iznosa koje ste utrošili u kupovini, a kao ograničavač koristite broj —1.

Vežba 2

Napišite program koji prihvata broj i ispisuje da li je manji od 100, jednak 100, ili veći od 100 (pomoć: ">" znači "veće od").

RELACIJE MEĐU BROJEVIMA

Postoji čitav niz simbola koji služe za upoređivanje dva broja. Evo potpune liste sa primerima:

=	jednako	(3=3 istinito; 3=4 lažno)
>	veće od	(3>2 istinito; 3>4 lažno)
<	manje od	(3<4 istinito; 3<2 lažno)
<>	različito	(3<>4 istinito; 3<>3 lažno)
>=	veće ili jednako	(3>=3, 3>=2 istinito; 3>=4 lažno)
<=	manje ili jednako	(3<=3, 3<=4 istinito; 3<=2 lažno)

LOGIČKI OPERATORI

Komandne reči:

AND
OR

mogu da se koriste za kombinovanje uslova. Ako sa c i d označimo uslove, onda je c AND d istinito uz uslov da su i c i d istiniti. Tako je:

X>0 AND X<3

istinito samo ako je X veće od nule, a istovremeno i manje od 3. Za cele brojeve to znači da X mora da bude 1 ili 2, a za decimalne (koji su već razmatrani) postoji više mogućnosti.

Na isti način c OR d je istinito uz uslov da je bar jedno (c ili d) istinito (a možda i oba). Tako:

X<5 OR X=5

ima isto značenje kao X<=5.

Konačno, trebalo bi da pomenem logičku komandu:

NOT

koja menja istinito u lažno i obrnuto.

USLOVNI SKOKOVI

Jedna od uobičajenih upotreba naredbe IF ... THEN je usmeravanje programa na novi red, na primer:

```
830 IF X=Y THEN 990
840 bilo šta ...
```

Ovo se zove *uslovni skok*.
Ovde postoji implicitno GOTO iza THEN. Naredba bi, u stvari, trebalo da glasi IF X=Y THEN GOTO 990, ali je GOTO izostavljeno radi kraćeg zapisa.

USLOVNE DODELE

Druga upotreba je dodeljivanje različitih vrednosti promenljivim zavisno od ispunjenosti uslova:

```
890 IF X=Y THEN K=77
900 IF X<>Y THEN LET K=99
```

što promenljivoj K dodeljuje vrednost 77 ako je X=Y, odnosno vrednost 99 ukoliko nije ispunjen pomenuti uslov. LET se, kao i obično, može izostaviti:

```
890 IF X=Y THEN K=77
900 IF X<>Y THEN K=99
```

Vežba 3

Porez na promet za neki artikl zavisi od šifre poreza i to:

šifra 1	(porez za školstvo)	2%
šifra 2	(porez za dečiju zaštitu)	5%
šifra 3	(savezni porez)	0%
šifra 4	(ostalo)	15%

Napišite program koji kao ulaz prihvata šifru i ispisuje procenat poreza.

ODGOVORI

Vežba 1

```
10 PRINT CHR$(147)
20 PRINT "IZDACI U KUPOVINI"
```

```
30 INPUT X
40 IF X<0 THEN 70
50 SUMA = SUMA + X
60 GOTO 30
70 PRINT "ZBIR JE"; SUMA
```

Vežba 2

```
10 PRINT CHR$(147)
20 INPUT N
30 IF N<100 THEN PRINT "MANJI OD 100"
40 IF N=100 THEN PRINT "JEDNAK 100"
50 IF N>100 THEN PRINT "VECI OD 100"
```

Vežba 3

```
10 PRINT CHR$(147)
20 INPUT "SIFRA POREZA"; TC
30 IF TC<1 OR TC>4 THEN 20
40 IF TC=1 THEN PRINT "2% POREZ ZA SKOLSTVO"
50 IF TC=2 THEN PRINT "5% POREZ ZA DECIJU ZASTITU"
60 IF TC=3 THEN PRINT "0% SAVEZNI POREZ"
70 IF TC=4 THEN PRINT "15% LOSA SRECA, DRUGAR"
```

U redu 30 uvedena je zaštita od upisivanja brojeva manjih od 1 ili većih od 4.

Računari ne obrađuju brojeve u našoj uobičajenoj decimalnoj notaciji. Oni ih obrađuju na način koji je bolje prilagoden elektronskim kolima. To je kao da brojite prste na nogama, a ne na rukama.

12 Binarni brojevi

Ne treba da budete stručnjak za matematiku da biste koristili binarne brojeve na "šezdesetčetvorci", ali je neophodno da znate nekoliko osnovnih stvari. Npr. ne možete da koristite sprajtove bez binarnih brojeva. Tako moramo da počnemo da ulazimo dublje u računar nešto ranije nego što smo hteli i da počnemo da vodimo računa o tome kako on stvarno *radi*.

Mi obično o brojevima razmišljamo u deseticama. Ako napišemo broj 3814 svi ćemo shvatiti da to znači:

$$3 \times 1000 + 8 \times 100 + 1 \times 10 + 4 \times 1$$

i svi znamo da se "mesna vrednost" od jedinice sa desne strane izračunava tako što za svako mesto ulevo prosto pomnožimo sa deset. Kažemo da takav broj ima *osnovu* deset.

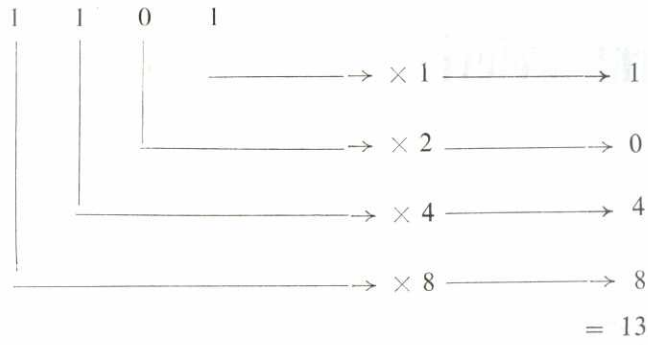
Pošto ovaj način koristimo od kad pamtimo, teško je shvatiti da postoje i drugi savršeno razumni načini za to isto. Razlog za korišćenje drugih osnova je u tome što se elektronski pojačavači ne ponašaju jednako za sve signale koji im se dovode. Npr. pojačavač koji treba da udvostruči ulazni signal može dobro da radi za ulaze od 1, 2, 3 i 4 jedinice, ali tada počinje da "zaravnjava" tako da ulaz 5 daje izlaz od samo 9,6 a ulaz 6 daje 10,8 tako da može da bude teško da se utvrdi razlika izlaza za ulaze 8 i 9.

Stavite kasetu sa snimljenom muzikom u neki jeftini kasetofon i pojačajte do kraja. Čujete li izobličenja? E, to je to. Prvi konstruktori računara nisu čuli izobličenja, nego su samo utvrdili da mašine povremeno ne mogu da razlikuju pojedine cifre, a to je obeshrabrivalo u pogledu pouzdanosti računara. Tako su morali da razmisle o predstavljanju brojeva i o tome šta je najpogodnije za elektronska kola.

Najjednostavnija stvar koju možete da uradite sa električnim signalom je da ga uključite ili isključite. Tako možete na zadovoljavajući način da predstavite cifre 0 (isključeno) i 1 (uključeno). Izobličenja više ne igraju nikakvu ulogu. Jasno je da li je signal prisutan, ili ne bez obzira koliko je izobličen. Ali, da li se može smisliti sistem koji koristi samo nule i jedinice? Da. U broju sa osnovom 10 najveća cifra je 9. Dodajte 1 na 9 i dobićete 10 — javlja se *prenos* (od jedinica u desetice). Možemo napisati bilo koji broj, po bilo kojoj drugoj osnovi koju izaberemo, ali će uvek najveća moguća cifra biti za 1 manja od osnove. Ako je osnova 2, najveća cifra je 1, pa broj sa osnovom 2 (ili *binarni* broj) sadrži samo nule i jedinice.

Šta sa mesnim vrednostima? Kod brojeva sa osnovom 10 mesne vrednosti smo dobili tako što smo krenuli od 1 (sa desne strane) i množili sa 10 svaki put kad smo se pomerili za jedno mesto levo. Za binarni broj opet počinjemo od 1, ali množimo sa 2 svaki put kad se pomerimo levo.

Tako, npr. binarni broj 1101 može da se prevede na osnovu 10 na sledeći način:



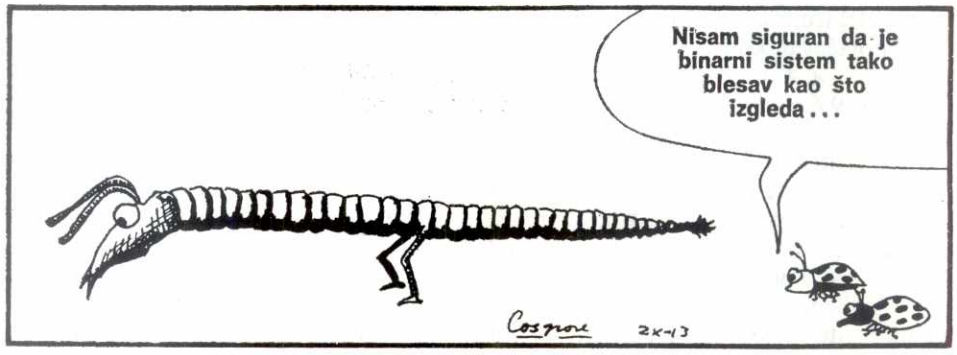
Prevođenje sa osnove 10 na osnovu 2 je isto tako prosto. Uzmimo za primer broj 25. Ako napišemo binarne mesne vrednosti:

32 16 8 4 2 1

i počnemo sa leve strane jasno je da nam treba 16, pa nam ostaje 9 koje se sastoji od 8 i 1, pa je tako broj 25 u binarnom obliku:

0 1 1 0 0 1

Svaka binarna cifra (0 ili 1) zove se *bit*.



BAJTOVI

Bajt je osmocifreni binarni broj, npr. 11000101 (= decimalno 197). Decimalne vrednosti bajta mogu da budu od 0 do 255. Memorijske lokacije "šezdesetčetvorke" mogu da sadrže tačno jedan bajt, pa otuda njihova važnost. (To je zbog toga što

je "šezdesetčetvorka" ono što zovemo *osmobitnim procesorom;* luksuznije mašine imaju veće memorijske lokacije, ali po znatno većoj ceni.) U Dodatku 1 navedeno je svih 256 mogućih vrednosti bajta. Sledeći program vam omogućava da upišete bajt i posmatrate kako se pretvara u decimalni broj.

```
10 DIM B(8)
20 SUMA = 0
30 PRINT CHR$(147)
40 INPUT "UPISI BROJ U BINARNOM OBLIKU"; B$
50 IF LEN(B$)<>8 THEN 40
60 PRINT CHR$(147)
70 PRINT "ZA PREVODJENJE BINARNOG BROJA▽"; B$
80 PRINT "U DECIMALNI KORISTI SE SLEDECI POSTUPAK:"
90 PRINT
100 FOR T=1 TO 8
110 S=2↑(8-T)
120 B(T) = VAL(MID$(B$, T, 1))
130 PRINT B(T); TAB(3);"X"; S; TAB(10);"=";
140 IF B(T)=0 THEN PRINT "▽▽▽▽▽0"
150 IF B(T)=1 AND T>1 THEN PRINT "▽";
160 IF B(T)=1 AND T>4 THEN PRINT "▽";
170 IF B(T)=1 THEN PRINT "▽▽"; S
180 SUMA = SUMA + B(T)*S
190 NEXT T
200 PRINT "(17 GRAFICKIH D + SHIFT)"
210 PRINT "▽UKUPNO▽▽ = ▽▽";
220 IF SUMA<100 THEN PRINT "▽";
230 IF SUMA<10 THEN PRINT "▽";
240 PRINT SUMA
250 PRINT "(17 GRAFICKIH D + SHIFT)"
260 PRINT
270 PRINT
```

BITOVI U BAJTU

Ponekad ćemo želeti da se pozovemo na, recimo, peti bit bajta 11010101. Ali da li znamo da li peti bit sa leve, ili peti bit sa desne strane? Nije svejedno, jedan je 0, a drugi 1. Da bi se izbegla zabuna, prihvaćeno je standardno numerisanje od 0 do 7 sa desnog kraja ka levom (ovo ima smisla iz matematičkih razloga) evo ovako:

Broj bita	7	6	5	4	3	2	1	0
Vrednost	1	1	0	1	0	1	0	1

Sada možemo da govorimo o "bitu 4 bajta 11010101" bez zabune.

Razlog za ovakvo numerisanje je u tome što najznačajniji bitovi, tj. oni koji u najvećoj meri doprinose decimalnoj vrednosti, imaju najveće brojeve. Jedinica u bitu 7 doprinosi 128 decimalnoj vrednosti, jedinica u bitu 6 doprinosi 64 itd. sve do bita 0 koji doprinosi samo 1. U stvari bit K doprinosi 2^K, tj. k-ti stepen broja 2 (u BASIC-u se to piše: 2 ↑ K).

VIŠI I NIŽI BAJT

Bajt može da sadrži vrednosti samo do 255, ali mnogi važni brojevi (npr. brojevi redova i adrese u mašini) mogu da budu veći. Sastavljanjem dva bajta možemo da idemo do 65535. Pretpostavimo da dva uzastopna bajta sadrže brojeve L i H. Tada je vrednost u ta dva bajta:

L + 256 * H

L zovemo *nižim bajtom*, a H *višim bajtom*. Npr. broj:

2500 = 196 + 9 * 256

biće smešten u memoriju kao:

	Niži bajt	Viši bajt
Binarno	11000100	00001001
Decimalno	196	9

U suštini ovde se broj tretira kao dvocifreni broj sa osnovom 256. Ovaj tip predstavljanja brojeva se koristi, npr. u zvučnom čipu SID (poglavlje 30). Za neki broj N između 0 i 65535 ova dva bajta su:

Viši bajt H = INT (N/256)
Niži bajt L = N — 256 * H = N — 256 * INT (N/256)

(INT znači "celobrojna vrednost od"). Na sličan način jedan bajt može da se koristi za memorisanje dva četvorobitna broja F i G (između 0 i 15) tako da sadrži 16 * F + G:

F	G	
1010	0101	= 165
10	5	

Ako je vrednost bajta X, onda imamo:

F = INT (X/16)
G = X — 16 * F = X — 16 * INT (X/16)

Izbor anvelope (ADSR) u SID-u koristi ovu metodu da ne troši gomilu polubajtova (polubajt se ponekad naziva nibl kao, npr. u knjizi *Reference Guide* = Vodič)

SLOBODNA MEMORIJA

Veličina memorije računara se obično izražava u *kilobajtima*, pri čemu jedan kilobajt ima 1024 (približno 1000). "Gola" "šezdesetčetvorka" ima 65536 bajtova, ili 64 kilobajta (64 K) memorije, ali oko 26 K zauzima sistem, tako da vama preostaje 38 K za igranje. Preciznije, ostaje vam 38911 bajtova.

Kad pišete program, slobodni prostor se smanjuje. Vrednosti promenljivih takođe troše prostor. Često je korisno da znamo koliko nam je memorije preostalo.

Da biste to utvrdili, koristite naredbu:

```
PRINT FRE (0)
```

Ako je rezultat pozitivan, to je broj slobodnih bajtova. Ako je negativan, pokušajte sa:

```
PRINT 65536 + FRE (0)
```

pa ćete dobiti tačnu vrednost. Ovo se dešava zbog čudnih osobina funkcije FRE, ali dovoljno je samo voditi računa o tome. U svakom slučaju, ako naredbom FRE dobijete negativnu vrednost, imate još najmanje 32 K memorije na raspolaganju. U programu možete da koristite:

```
1000 F = FRE (0) : IF F < 0 THEN F = F + 65536
1010 PRINT F; "▽ SLOBODNIH BAJTOVA"
```

LOGIKA BAJTOVA

Sa binarnim brojevima možete da vršite aritmetičke operacije, bilo direktno, bilo konvertujući ih u decimalne i obratno. Međutim, sa binarnim brojevima se mogu vršiti i *logičke* operacije AND i OR (I i ILI). Možda biste mogli da pogodite odgovor na:

```
185 OR 148
```

Verovali ili ne: 189. Evo objašnjenja: Postoji veza između logike bajtova i obične logike zasnovana na logičkoj interpretaciji binarnih cifara:

0 znači "lažno"

1 znači "istinito".

Za jednocifrene binarne brojeve logički operatori OR i AND daju sledeće rezultate:

0 AND 0 = 0	0 OR 0 = 0
0 AND 1 = 0	0 OR 1 = 1
1 AND 0 = 0	1 OR 0 = 1
1 AND 1 = 1	1 OR 1 = 1

Znači, p AND q je istinito (vrednost 1) samo kad su i p i q istiniti, dok je p OR q istinito uz uslov da je istinito p ili q ili oba. Ovo je u skladu sa uobičajenim logičkim pravilima. Ova ideja se može proširiti na niz binarnih cifara. Operacija se vrši na svakom bitu u nizu. Npr. da bismo izračunali:

10111001 OR 10010100

uzimamo cifre po redu, ovako:

		1. broj		2. broj		Odgovor
Bit 7	→	1	OR	1	=	1
Bit 6	→	0	OR	0	=	0
Bit 5	→	1	OR	0	=	1
Bit 4	→	1	OR	1	=	1
Bit 3	→	1	OR	0	=	1
Bit 2	→	0	OR	1	=	1
Bit 1	→	0	OR	0	=	0
Bit 0	→	1	OR	0	=	1

Sada pročitajmo rešenje:

10111001 OR 10010100 = 10111101

U decimalnoj notaciji ovo postaje:

185 OR 148 = 189

a to je ono što sam ranije tvrdio.

Na sličan način 185 AND 148 daje:

Bit 7	→	1	AND	1	=	1
Bit 6	→	0	AND	0	=	0
Bit 5	→	1	AND	0	=	0
Bit 4	→	1	AND	1	=	1
Bit 3	→	1	AND	0	=	0
Bit 2	→	0	AND	1	=	0
Bit 1	→	0	AND	0	=	0
Bit 0	→	1	AND	0	=	0

Ovde je odgovor 10010000 ili 144 decimalno.

Sledeći program može da vam pomogne da malo vežbate ove proračune i da vam približi osnovnu ideju.

```
10 PRINT CHR$(147)
20 INPUT "UPISITE PRVI BROJ";A
30 IF A<0 OR A>255 THEN 20
40 INPUT "UPISITE 'AND' ILI 'OR";X$
```

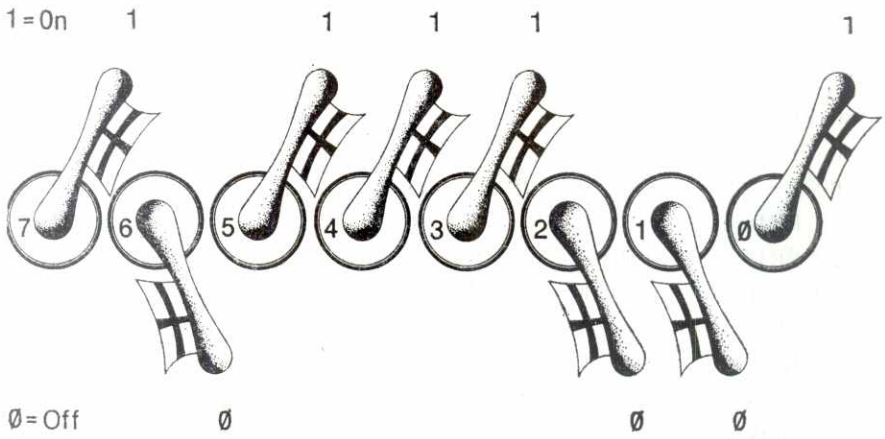
```
50 IF X$<>"AND" AND X$<>"OR" THEN 40
60 INPUT "UPISITE DRUGI BROJ";B
70 IF B<0 OR B>255 THEN 60
80 PRINT : PRINT
90 PRINT TAB(5);A;TAB(25);
100 C=A: GOSUB 500: PRINT K$
110 PRINT
120 PRINT TAB(5);B;TAB(25);
130 C=B: GOSUB 500: PRINT K$
140 PRINT TAB(25);"(8 GRAFICKIH * + SHIFT)"
150 IF X$="AND" THEN C=A AND B
160 IF X$="OR" THEN C=A OR B
170 GOSUB 500
180 PRINT TAB(5);A;X$;B;
190 PRINT TAB(25);K$
200 PRINT TAB(25);"(8 GRAFICKIH * + SHIFT)"
210 PRINT TAB(6);
220 PRINT "DECIMALNI OBLIK ▽▽▽▽▽▽▽▽";D: END
500 REM KONVERZIJA DECIMALNOG U BINARNI OBLIK: C U K$
510 K$="": D=C
520 FOR T=1 TO 8
530 CH=INT(C/2)
540 IF 2*CH=C THEN K$="0"+K$
550 IF 2*CH<>C THEN K$="1"+K$
560 C=CH
570 NEXT T
580 RETURN
```

Grafički znakovi u redovima 140 i 200 su 8 kopija znaka na tasteru * kad se pritisne i taster SHIFT. Naredba GOSUB još nije objašnjena — vidi poglavlje 14. Ona je slična naredbi GOTO, s tim što ima još jedan povratni GOTO na kraju, a koristi se za izvršavanje nekog zadatka — u našem primeru da konvertuje C u binarni broj zapisan u obliku znakovnog niza K$. Za sada ne vodite računa o tome.

Program ima tri ulaza: prvi broj, izbor AND ili OR i drugi broj.

PRIMENA SISTEMSKIH PROMENLJIVIH

Možda se pitate čemu sve ovo služi. To se primenjuje kad treba računar da premestimo u određeno "stanje duha" modifikovanjem sadržaja neke memorijske lokacije. Sistem ponekad koristi binarne cifre na određenim lokacijama (nazvanim sistemske promenljive) kao *indikatore* — tj. signale koji određuju da li se nešto koristi ili ne. Možete da posmatrate bajt kao niz od osam malih prekidača, pri čemu 0 znači da je prekidač "isključen", ili da je indikator (zastavica) "spušten", a 1 znači



Sl. 12.1 — Bajt je kao niz od osam prekidača ili indikatora od kojih svaki može da bude u jednom od dva stanja (1 = uključen, 0 = isključen)

"uključen", odnosno indikator "podignut". Na slici 12.1 je pokušaj jednovremenog prikazivanja prekidača i indikatora (zastavica).

Pretpostavimo da želimo da promenimo stanje bita 2 u "uključeno", bez obzira u kom se stanju trenutno nalazi, a da druge bitove želimo da ostavimo takve kakvi su. Ako znamo da je prekidač 2 *isključen*, stvar je jednostavna — dodaćemo 4 binarnoj vrednosti:

```
10111001 + 00000100 = 10111101
    ↑                     ↑
Bit 2 isključen      bit 2 uključen
```

Divno! Ali pretpostavimo da nismo znali u kom stanju se nalazi prekidač. Dodavanje broja 4 može da bude katastrofalno, jer:

```
10111101 + 00000100 = 11000001
    ↑                     ↑
Bit 2 uključen       Opaa! Bit 2 isključen
```

što je još gore, isključeni su i bitovi 5, 4 i 3, dok je bit 6 *uključen*. To je kao kad bi računar bio automobil, pa pokušamo da upalimo farove, pa ih isključimo iako su bili upaljeni, isključimo motor, parkirna svetla i poziciona svetla, a uključimo sirenu. Nije baš sjajno.

Problem je u ciframa koje se prenose prilikom sabiranja. Ono što je nama potrebno, je neka operacija koja utiče *samo* na bit 2 i postavlja ga na 0 ili 1 (u našem slučaju 1). Sad, ako nad bitom izvršite operaciju OR sa nulom, on ostaje kakav je i bio, ako izvršite operaciju OR sa jedinicom, on postaje 1 (proverite iz ranije navedene tabele). Tako, ako nad bajtom izvršimo operaciju OR sa 00000100, svi bitovi će ostati kakvi su i bili, izuzev bita 2 koji će dobiti vrednost 1. Ukratko:

```
pqrstuvw OR 00000100 = pqrst1vw
```

za sve vrednosti (0 ili 1) p, q, . . . w. Pošto možemo da damo naredbu OR za decimalne brojeve, to znači da:

X OR 4

daje rezultat jednak X, ali sa uključenim bitom 2.

Slično, da bi se bit 2 postavio na vrednost 0, a da svi ostali bitovi ostanu nepromenjeni, treba izvršiti operaciju AND sa 11111011 (decimalno 251). Ovo je zbog toga što je uvek p AND 1 = p, dok je p AND 0 = 0 (nije slučajno što je 251 = 255—4, ali samo matematičarima preporučujemo da utvrde vezu).

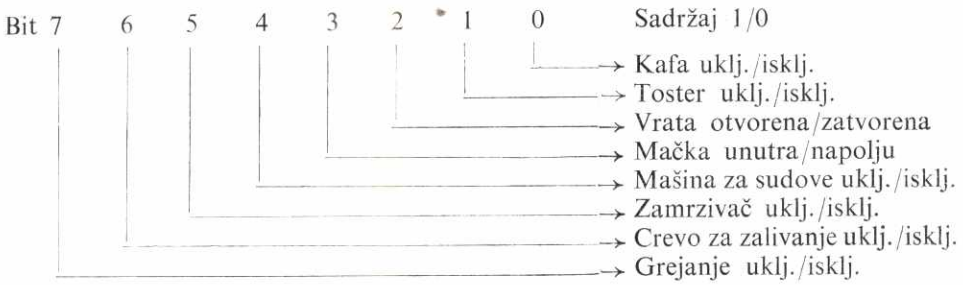
U daljem tekstu knjige povremeno ću koristiti ovu tehniku. Nije *neophodno* da je sada detaljno shvatite, da biste mogli da pratite naredbe u programima (*sada* nam to kaže. Grrr, uuhhh), ali pomaže u objašnjavanju onoga što bi inače bilo samo papazjanija. Posebno bih hteo da vas upozorim da programski red kao:

Y = X OR 4

ne znači da je korektor slabo obavio posao. Taj red ima određeni smisao. Možete se kladiti u deset prema jedan da taj red negde u memoriji uključuje bit 2.

Vežba 1

(Nemojte ovu vežbu shvatiti suviše ozbiljno, ali ona može da vam pomogne da shvatite upotrebu bitova kao indikatora za način rada sistema). Zamišljeni čuveni mikroračunar Roda-37, koristi promenljivu SWITCH za upravljanje nekim od mogućih dodataka na sledeći način:



Ne znate koje vrednosti imaju bitovi, ali biste hteli da mačku izbacite napolje. Koju komandu ćete upotrebiti?

ODGOVORI

Vežba 1

Za kontrolu mačke koristi se bit 3, a 11110111 je decimalno 247, pa naredba:

SWITCH = SWITCH AND 247

rešava problem. Jiiipiii!

Svaka lokacija u memoriji „šezdesetčetvorke" ima svoju adresu. Promenom sadržaja određene lokacije možete naterati mašinu da izvršava različite zadatke. Dve naredbe vam omogućavaju da vidite šta je smešteno na datoj adresi i da to izmenite:

13 PEEK i POKE

Kad programirate u BASIC-u, obično možete da prepustite mašini vođenje evidencije, pa programer ne mora da vodi računa o pitanjima kao: "Gde se u memoriji nalazi to i to?". Međutim, čim poželite da koristite šire mogućnosti "šezdesetčetvorke", kao što su boje i zvuk, (što ne spada u standardni BASIC), takva pitanja postaju važna. "Šezdesetčetvorka" nije toliko pristupačna početnicima kao neke druge mašine i zahteva da programer vodi računa o mnogim organizacionim detaljima. Za uloženi napor da ovladate nekim prostim idejama kompenzacija će biti u bržim reakcijama mašine na vaše naredbe. U stvari, možete samo da prepišete (iz programa u ovoj knjizi) one delove programa koji se bave ovim i koristite napredne mogućnosti, a da ne shvatite kako se to postiže. Ako vam to izgleda lakše, preskočite ovo poglavlje, pa ga pročitajte kasnije. Međutim, kao i u većini poslova, život je mnogo lakši, ako možete da shvatite *zašto*, i *kako* se nešto radi.

ORGANIZACIJA MEMORIJE

Memorija računara je izuzetno kruto i sistematski organizovana. Svaka memorijska lokacija ima određeni broj koji je njena *adresa*. U "šezdesetčetvorci" adrese su od 0 do 65535. Svaka adresa sadrži broj od 0 do 255 (jedan znak informacije). U stvari, taj broj je smešten u *binarnom* obliku kao niz od osam nula i jedinica, tako da svaka adresa sadrži jedan *bajt*. Za neke svrhe dovoljno je o tom broju razmišljati kao o običnom decimalnom broju, pa ću u ovom trenutku i ja postupiti tako.

Računar ima dva tipa memorije:

1. ROM (Read Only Memory — memorija čiji se sadržaj samo čita) koja sadrži operativni sistem odnosno ugrađene naredbe koje računaru omogućavaju rad.

2. RAM (Random Access Memory — memorija sa direktnim pristupom) u koju se smeštaju sadržaji koji mogu da se menjaju: BASIC-program, izgled ekrana, pomoćne tablice za proračune i sl.

Sadržaj RAM-a možete da menjate, ali ne i sadržaj ROM-a. Zbog toga: *najgore* što možete da uradite muvajući se po memoriji je to da mašinu dovedete u stanje koje zahteva da je isključite i ponovo uključite. Na taj način gubite sadržaj RAM-a, ali nema nikakvog oštećenja mašine. Dakle, možete slobodno da eksperimentišete i ne brinete da li ćete oštetiti mašinu.

Poslednjih deset bajtova ROM-a, npr. izgledaju kao u tabeli 13.1, a u "šezdesetčetvorci" ima još mnogo hiljada bajtova informacija. Kad računar radi, sve te male nule i jedinice jure u unutrašnjosti mašine kao mravi (pa sad, jedinice neka budu mravi a nule — ne-mravi), sve dok ne dođu tamo gde ste vi, kao programer (i ljudi koji su napisali program za ROM), odredili. Zar ne osećate strahopoštovanje kad upravljate životom i smrću tolikih mrava?

Tabela 13.1

Adresa	Sadržaj	
	binarno	decimalno
65526	01010010	82
65527	01010010	82
65528	01000010	66
65529	01011001	89
65530	01000011	67
65531	11111110	254
65532	11100010	226
65533	11111100	252
65534	01001000	72
65535	11111111	255

NAREDBA PEEK

Naredba, koja vam omogućava da vidite sadržaj memorijske lokacije (ali *ne* i da ga promenite), je:

PEEK

Na primer, da biste utvrdili sadržaj adrese 65535, otkucajte:

PRINT PEEK (65535)

Treba da dobijete broj 255 na ekranu. Ako ne dobijete taj broj, slagao sam vas.

Sledeći program (koji će u ovom poglavlju biti proširen, pa zbog toga ima čudne brojeve redova) omogućava vam da koristite PEEK u bilo kom delu memorije, po izboru:

```
 10 REM PEEKING ROUTINE
 20 INPUT "POCETNA ADRESA";AD: IF AD<0 OR AD>65535
  THEN 20
 30 H=0
 60 P=PEEK(AD)
 90 PRINT AD;TAB(8);P;TAB(16);
100 IF P<32 OR P>127 AND P<160 THEN PRINT "CTRL";
110 IF P>31 AND P<128 OR P>159 THEN PRINT CHR$(P);
120 PRINT TAB(24);
```

```
130 IF P>191 THEN PRINT "+";
140 PRINT
230 AD=AD+1: IF AD>65535 THEN STOP
240 H=H+1: IF H<20 THEN 60
250 GET A$: IF A$="" THEN 250
260 IF A$="S" THEN STOP
270 GOTO 30
```

Otkucajte RUN i upišite 65526 kad vas program upita za početnu adresu, pa ćete dobiti sadržaj poslednjih deset lokacija ROM-a kao u tabeli 13.1 (bez binarne verzije, ali sa dodatnom kolonom u kojoj će biti znak čiji je kôd broj dobijen naredbom PEEK). Tako sam ih i ja dobio.

Uopšte, možete upisati bilo koju početnu vrednost koju želite, program je napisan tako da obezbedi upisivanje adrese između 0 i 65535. Dobićete jednu "stranu" prikaza. Da biste prekinuli rad pritisnite taster "S", a da biste dobili sledeću stranu prikaza, bilo koji drugi taster. Obratite pažnju na brojeve u prve dve kolone. One daju adrese i sadržaje na tim adresama. Treću kolonu ću objasniti kasnije.

Prikazani brojevi vam možda ne znače mnogo. Ne bi ni trebalo — oni su pisani u mašinskom jeziku mikroprocesora 6510. Čak i za početak razumevanja trebalo bi da uzmete neku drugu knjigu (npr. *Vodič*). Međutim, ako nastavite da pritiskate tastere za novu stranu, shvatićete da se u vašem računaru nalazi gomila informacija. Neke od njih možete čak i da prepoznate.

Vežba 1

Pustite izvršenje navedenog programa (naredbom RUN) i upišite 41118, a zatim ga pustite ponovo i upišite 41374. Koji deo memorije vidite?

NARCIS SE PONOVO JAVLJA

Prema legendi Narcis se zaljubio u svoj sopstveni odraz u vodi i pretvorio se u cvet (osveta Artemide zbog toga što Narcis nije odgovorio na ljubav nimfe Eho). Izlažući se istoj opasnosti navešću program PEEKING ROUTINE da naredbu PEEK primeni na samog sebe.

Programi u BASIC-u se smeštaju u memoriji normalno počev od adrese 2048. (Ako ne promenite neke bitove u RAM-u tako da se promeni početak BASIC-programa, što pretpostavljam da niste učinili. Ukoliko znate kako da promenite memorijsko mesto početka programa, ionako nije potrebno da čitate ovo poglavlje.) Ako otkucate RUN, a zatim upišete 2048, dobićete prikaz dat u tabeli na str. 73.

(Striktno govoreći, poslednja dva reda će biti na drugoj strani prikaza, tako da treba da pritisnete neki taster da biste ih dobili, a uz njih i nastavak liste. Međutim, naveo sam ih zbog urednosti).

Treća kolona predstavlja znak koji se dobija konvertovanjem broja smeštenog u memorijskoj lokaciji (vidi *Priručnik*, Dodatak F). Neki znakovi imaju čudan učinak, ako pokušate da ih prikažete naredbom PRINT, tako da smo u programu predvideli da se umesto njih prikaže CTRL ("control"). Za kodove veće od 191 ispisaćemo "+" pošto daju iste znakove kao neki manji brojevi (primetićete, ako nastavite sa istraživanjem, da kôd 34 daje veoma čudan prikaz koji nećemo objašnjavati zbog nedostatka prostora. U svakom slučaju, bezopasan je).

Adresa	Sadržaj (decimalno)	Znak
2048	0	CTRL
2049	23	CTRL
2050	8	CTRL
2051	10	CTRL
2052	0	CTRL
2053	143	CTRL
2054	32	(prazno)
2055	80	P
2056	69	E
2057	69	E
2058	75	K
2059	73	I
2060	78	N
2061	71	G
2062	32	(prazno)
2063	82	R
2064	79	O
2065	85	U
2066	84	T
2067	73	I
2068	78	N
2069	69	E

U tabeli vam neće odmah sve biti jasno, ali veoma jasno možete da pročitate naslov PEEKING ROUTINE. Ako nastavite sa sledećim stranama prikaza, naići ćete na druge delove programa koji su odmah prepoznatljivi, a između njih neke zbrkane stvari. Zbrkane stvari su brojevi redova, u prilično nejasnom obliku, ili ključne reči BASIC-a komprimovane u 1 bajt da bi se uštedeo prostor u memoriji. Npr. 143 na lokaciji 2053 zamenjuje REM. Da narode, program se stvarno *smešta* u mašinu i možete ga i *videti*, ako ga tražite na pravom mestu.

Hmmmm .... ništa sjajno se još ne dešava. Međutim, prelaskom na sledeći odeljak možete postati pravi mađioničar.

NAREDBA POKE

Ne samo što možete da vidite program, nego možete i da ga menjate. Otkucajte ove direktne naredbe:

```
POKE 2058, 86
POKE 2060, 83
POKE 2061, 72
```

Sada izlistajte program (naredba LIST). Da li je isti? Ne baš sasvim. Sada naredba REM kaže:

PEEVISH ROUTINE (=džandrljivi program)

Ako opet otkucate RUN i upišete vrednost 2048, videćete šta se dešava: adrese 2058, 2060 i 2061 promenile su svoj sadržaj u V, S, H umesto K, N, G. Znači, naredba POKE omogućava da stavite određeni bajt na neku lokaciju čija adresa je data.

Vežba 2

Promenite PEEKING ROUTINE u PARKING ROUTINE pomoću dve naredbe POKE.

Sada sam stvarno otvorio Pandorinu kutiju. Ako možete da menjate sadržaje memorijskih lokacija po želji, možete da promenite bilo šta uz uslov da znate gde se nalazi i da nije bezbedno u ROM-u. Iz *Vodiča* možete da pronađete korisne adrese, a ja ću u kasnijim poglavljima neke i navesti. U svakom slučaju možete se zabavljati koristeći naredbu PEEK po celoj memoriji, a ukoliko vam se sviđa da koristite i naredbu POKE, ne oklevajte. Međutim, imajte na umu: slučajne promene sadržaja naredbom POKE retko daju dobar rezultat. Potrebno je da znate nešto o sistemu pre nego što počnete da se igrate s njim, baš kao i kod automobila. Razmišljam, da li je možda Pandorina kutija bila puna *buba* (engl. bug = buba, a označava i grešku u programu, prim. prev.).

BOJE EKRANA

Kao uvod u umeće korišćenja naredbe POKE, kojom se operativni sistem može koristiti spolja, pogledajmo boje ekrana. U bilo koje vreme na ekranu postoje tri osnovne boje: boja *ruba* ekrana, boja osnove (koju ponekad zovu i "boja papira") i boja prikazanog znaka (koja se zove i "boja mastila"). Boju znaka (mastila) možete da promenite pomoću tastera CTRL (vidi *Priručnik*, str. 10). Ali kako promeniti boje osnove (papira) i ruba?

Kodovi, koji upravljaju ovim bojama, smešteni su na dve lokacije:

53280	boja ruba
53281	boja osnove

a postoji lokacija i za boju znaka:

646 (tekuća boja znaka)

koju možete da koristite umesto tastera CTRL. Da biste izmenili boju, sve što treba je da naredbom POKE izmenite sadržaj na odgovarajućoj adresi. Kodovi boja su:

0 crna	8 narandžasta
1 bela	9 smeđa
2 crvena	10 svetlo crvena
3 cijan-plava	11 tamno siva
4 purpurna	12 srednje siva
5 zelena	13 svetlo zelena
6 plava	14 svetlo plava
7 žuta	15 svetlo siva

Tako, na primer:

POKE 53280, 9	daje smeđi rub
POKE 53281, 4	daje purpurnu osnovu
POKE 646, 11	daje tamno sivi znak

Promena boje osnove i ruba odnosi se na ceo ekran dok se promena boje znaka odnosi samo na nove znakove a stari ostaju one boje koje su i bili.

SKUP ZNAKOVA

Najlepše smo ostavili za kraj. Pokazaću vam gde "šezdesetčetvorka" memoriše informacije koje joj omogućavaju prikazivanje znakova. Međutim, ima nekoliko trikova koje treba primeniti da bi program funkcionisao.

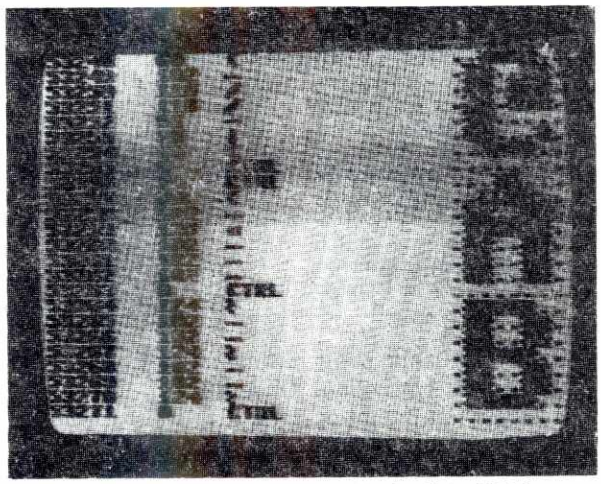
Prvo, podaci o znakovima smešteni su u memoriji u *binarnom* obliku pri čemu svaka binarna jedinica označava tačku u boji znaka, a nula tačku u boji osnove. Zbog toga ću dodati program za konverziju u binarni oblik iz poglavlja 12.

Druga začkoljica je to što je adrese za skup znakova lako naći — prema *Vodiču* one su od 53248 do 57343 — ali naredbom PEEK nećete sa tih adresa dobiti prave informacije. "Šezdesetčetvorka" je komplikovana životinjka i meša informacije u memoriji prema svojim potrebama. Ona meša i same adrese tako da data adresa može da ima više od jednog značenja zavisno od "stanja duha" u kojem se mašina nalazi.

U njenom normalnom stanju navedene adrese ne ukazuju uopšte na skup znakova u ROM-u nego na deo RAM-a koji se koristi za ulaz/izlaz. Međutim, "šezdesetčetvorka" se uz neznatno korišćenje naredbi POKE, može naterati da ove adrese poveže sa ROM-om. Potrebna je još jedna naredba POKE da se u toku rada onemogući prijem signala sa tastature, pošto bi inače moglo doći do svačega. Posle toga, naravno, mašinu treba vratiti u normalno stanje, i to odmah posle naredbe PEEK, tako da može da funkcioniše kao i do tada. Zvuči malo zbrkano, ali to je moj problem, a ne vaš, zar ne? A evo kako ćemo to uraditi.

U program PEEKING ROUTINE dodajte sledeće redove:

```
40 POKE 56334, PEEK (56334) AND 254
50 POKE 1, PEEK (1) AND 251
```



Sl. 13.1 — Podaci o znakovima "ukradeni" iz ROM-a

```
70 POKE 1, PEEK (1) OR 4
80 POKE 56334, PEEK (56334) OR 1
```

To je deo programa za preklapanje delova memorije. Red 40 onemogućava prekide (interrupt-e) sa tastature, red 50 dovodi informacije o skupu znakova na pravo mesto, red 70 ih oslobađa, a red 80 ponovo omogućava prekide. Zatim treba ubaciti konverziju u binarni oblik:

```
140 PRINT TAB(28);
150 B$=""
160 FOR S=1 TO 8
170 PH=INT(P/2)
180 IF P=2*PH THEN B$=CHR$(46)+B$
190 IF P<>2*PH THEN B$=CHR$(166)+B$
200 P=PH
210 NEXT S
220 PRINT B$
```

Primetimo da red 140 dolazi umesto starog reda sa istim brojem. Konačno, izmenite red 240 tako da glasi:

```
240 H=H+1 : IF H<20 THEN 40
```

Zatim otkucajte RUN i kao početnu adresu upišite 53248. Dobićete oblike slova @, A, B, C . . . . na desnoj strani ekrana. Umesto nula i jedinica pokušao sam da oblik prikažem tačkama i tačkastim kvadratima — znakovima sa kodovima 46 i 166, pa odatle u programu i redovi 180 i 190. Ako vam se više sviđaju nule i jedinice, promenite ove redove u:

```
180 IF P=2*PH THEN B$="0"+B$
190 IF P<>2*PH THEN B$="1"+B$
```

Različiti tipovi znakova su u memoriji smešteni počev od sledećih adresa:

53248	Velika slova
53760	Grafika
54272	Velika slova, inverzno (zamenjene boje osnove i znaka)
54784	Grafika, inverzno
55296	Mala slova
55808	Velika slova i grafika
56320	Mala slova, inverzno
56832	Velika slova i grafika, inverzno

Ima i izvesnog ponavljanja zbog načina na koji "šezdesetčetvorka" koristi dva različita skupa znakova (*Priručnik*, Dodatak E).

Informacije o znakovima mogu da se koriste u programima — npr. za prikazivanje znakova većih dimenzija. *Kad god to radite*, koristite redove 40, 50, 70, 80 iz našeg programa. Možete čak i da napravite svoj sopstveni skup znakova, ali to je za ovu knjigu suviše napredno pa pogledajte *Vodič*, str. 107—113.

ODGOVORI

Vežba 1

1. Tabela ključnih reči BASIC-a

2. Tabela poruka o grešci. Završni znak svake ključne reči ili poruke izmenjen je tako da bude oznaka koja računaru kaže gde reč završava. Pritisnite COMMODORE + SHIFT i ponovo pustite izvršenje programa. Sada su reči ispisane malim slovima, a oznake velikim. Upotreba ovih oznaka je lukav trik za uštedu memorije.

Vežba 2

```
POKE 2056, 65
POKE 2057, 82
```

Ako utvrdite da stalno pišete iste delove programa sa malim izmenama, ili da počinjete da zaboravljate šta neki delovi programa treba da rade, možda je to problem koji rešavaju:

14 Potprogrami

Često se određeni skup naredbi ponavlja nekoliko puta u istom programu. *Možda* ćete izbeći višestruko pisanje naredbi razumnom upotrebom naredbe GOTO, ali to nije uvek dovoljno dobro. Naredba

```
GOSUB
```

je kao GOTO koji "pamti odakle je došao". Kad naiđe na naredbu

```
RETURN
```

program se vraća na red *iza* reda sa naredbom GOSUB koja ga je poslala na ovaj deo programa.

Iza naredbe GOSUB obavezno se mora navesti broj reda, na primer:

```
GOSUB 1000
```

što program šalje na niz naredbi počev od reda 1000 (baš kao i GOTO 1000). Deo programa između reda 1000 i naredbe RETURN poznat je pod imenom *potprogram*, a kaže se da je *pozvan* naredbom GOSUB. Tako je potprogram neka vrsta "mini programa" koji se koristi za izvršenje određenog zadatka u nekom većem programu.

Postoje bar dva dobra razloga za korišćenje potprograma:

1. Njima se izbegava nepotrebno ponavljanje što dovodi do kratkih i efikasnih programa.

2. Oni programeru omogućavaju da organizuje program u lako razumljive delove što pojednostavljuje testiranje programa i pronalaženje i otklanjanje grešaka.

Vredi rano steći običaj pisanja potprograma — posebno na "šezdesetčetvorci" kod koje se mnoge standardne mogućnosti najbolje koriste efikasnim potprogramima.

Na primer, razmotrimo zadatak promene boje osnove (papira) ekrana. Mini program za to izgleda ovako:

```
3000 REM BOJA OSNOVE
3010 POKE 53281, PAPIR
```

Ovde je PAPIR promenljiva koja dobija vrednost 0 do 15 zavisno od toga koja boja se traži (vidi poglavlje 13). Da biste koristili ovaj potprogram, morate da dodate još jedan red:

```
3020 RETURN
```

Sada umesto naredbe POKE 53281, itd. možete da koristite naredbu GOSUB 3000 uz povećanu jasnoću programa. Npr. pretpostavimo da želite da menjate boje da biste povećali atraktivnost neke reklamne poruke na ekranu. Tada biste mogli da uradite nešto kao:

```
100 REM GLAVNI PROGRAM
110 PRINT CHR$(147)              [BRISANJE EKRANA/POKAZIVAČ
                                 U POLAZNI POLOŽAJ]
120 PRINT:PRINT:PRINT            [TRI PRAZNA REDA]
130 PRINT TAB(10); "TEHNICKA KNJIGA"
140 PRINT
150 PAPER=2: GOSUB 3000          [CRVENA OSNOVA]
160 PRINT TAB(5); "KATALOG RACUNARSKE LITERATURE
                                  NOV. 85"
170 PRINT
180 PAPER=4: GOSUB 3000          [PURPURNA OSNOVA]
190 PRINT "LAKO PROGRAMIRANJE ZA CRAY-1"
200 PRINT
210 PAPER=7: GOSUB 3000          [ŽUTA OSNOVA]
220 PRINT "IGRE NA VASEM VAX-U"
230 PRINT
240 PAPER=8: GOSUB 3000          [NARANDŽASTA OSNOVA]
250 PRINT "KOMPAJLER GENERATOR ZA PRAVE POCETNIKE"
260 PRINT
270 PAPER=13: GOSUB 3000         [SVETLO ZELENA OSNOVA]
280 GOTO 100                     [OPET SVE ISPOČETKA]
```

Glavno što ovde treba shvatiti je to da program, kad naiđe na naredbu GOSUB 3000 (recimo u redu 150), skače na relevantni red (ovde 3000) i izvršava naredbe dok ne naiđe na RETURN, a zatim se vraća na red *iza* 150. To znači, na red 160. Pa sad, *mogli* biste isti učinak postići i naredbom:

```
150 PAPIR = 2 : GOTO 3000
```

a da naredbu RETURN izmenite u:

```
3020 GOTO 160
```

Ali . . . kad dođe do reda 180, na sledećem GOSUB 3000 naredba RETURN vraća program na red 190, tako da modifikovan red 3020 ne bi radio ono što treba.

Slično će i naredba GOSUB u redu 210 biti izvršena do RETURN koji vraća program na red 220, za GOSUB u redu 240 RETURN vraća napred 250, a za GOSUB u redu 270 na red 280. To je bitna razlika između GOSUB i GOTO i ona čini da GOSUB bude mnogo moćnija naredba.

KRPLJENJE PROGRAMA

Ako stvarno pokušate da izvršite program iz primera, utvrdićete da se sve dešava za tren oka, pa je tekst skoro nečitljiv. Znači, program treba usporiti stavljanjem pauza na odgovarajuća mesta. Dobar način je da se doda još jedan potprogram:

```
2000 REM PAUZA U TRAJANJU OD OKO 2 SEC
2010 FOR T=1 TO 1500
2020 NEXT T
2030 RETURN
```

Sada ovaj potprogram možemo da pozovemo kad god nam je potrebna pauza:

```
145 GOSUB 2000
175 GOSUB 2000
205 GOSUB 2000
235 GOSUB 2000
265 GOSUB 2000
```

Ovaj način upotrebe potprograma za uvođenje dodatnih naredbi (kojih smo se naknadno setili) zove se *krpljenje* (engl. patch = zakrpa). To je primer još jednog dobrog načina korišćenja fleksibilnosti naredbe GOSUB.

STRUKTURA PROGRAMA

Osnovna struktura programa sada izgleda kao na slici 14.1. Uočavamo "glavnu liniju" toka programa sa ponavljanjem skretanja na potprograme.

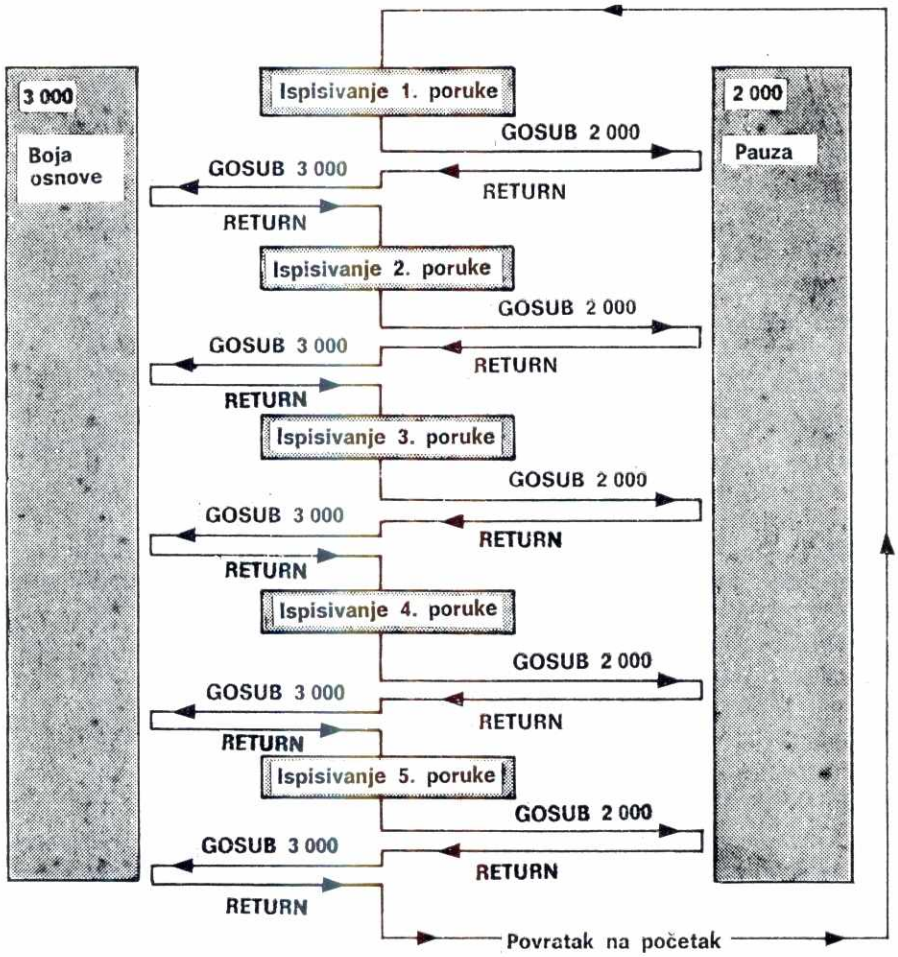
DAME IMAJU PRIVILEGIJU ...

... da mogu da se predomisle (kažu). Pretpostavimo da odlučite da promenite boju *ruba*, tako da bude ista kao boja osnove. Da ste pisali svaku promenu boje direktno pomoću POKE 53281, 2, itd. trebalo bi dodati još čitavu gomilu naredbi POKE 53280, ... da biste dobili to što tražite. Ovako, sa potprogramom, jedan jedini dodatni red rešava problem:

```
3015 POKE 53280, PAPIR
```

Vežba 1

Da bi sve bilo kako treba potrebno je da početna boja ruba bude tamno plava. Uradite to.



Sl. 14.1 — Tok izvršavanja programa za reklame. Uočavaju se ponovljeni pozivi dva potprograma i različite povratne tačke

PISANJE PROGRAMA "ODOZGO NADOLE"

Još jedna prednost potprograma je u tome što vam omogućavaju da pišete program "odozgo nadole", odnosno od globalnog izgleda programa, dok fini detalji mogu da se razrade i kasnije. "Tražite funte i ne brinite šta se dešava sa penijima" kako ne kaže izreka . . . ali je ipak to bolji savet u ovom slučaju. Uzmimo navedeni primer, pretpostavimo da ga nismo rešili na navedeni način i primenimo principe programiranja "odozgo nadole". Videćete da je rezultat primetno bolji.

Prvo: odredimo glavne podzadatke. Odmah mogu da se setim tri:

1. Prikaz poruke
2. Pauza
3. Promena boje osnove (i/ili ruba)

Ovi zadaci se u toku programa izvršavaju po pet puta. Zbog toga ćemo napisati tri potprograma koji će ih izvršavati:

```
1000 REM PRIKAZIVANJE PORUKE
(nešto što će dati prikaz poruke M$)
????? RETURN
```
} prvi potprogram

```
2000 REM PAUZA
(nešto što će izazvati pauzu od oko 2 sec.)
???? RETURN
```
} drugi potprogram

```
3000 REM BOJA OSNOVE
(nešto što će izazvati promenu boje
osnove u boju određenu promenljivom PAPIR)
?????? RETURN
```
} treći potprogram

Ovde smo sa ???? označili broj reda čija će se stvarna vrednost utvrditi kad budemo *pisali* potprograme. Ali za sada, jedino moramo da vodimo računa o promenljivima koje se u tim potprogramima koriste.

Trebaće da i računar dovedemo u stanje sa ispravnim početnim uslovima (brisanje ekrana, pomeranje pokazivača u gornji levi ugao, itd.):

4. Inicijalizovanje sistema

Biće nam, znači, potreban i četvrti potprogram:

```
4000 REM INICIJALIZOVANJE
(nešto što će dovesti računar u tražene uslove
????? RETURN
```
} četvrti potprogram

GLAVNI PROGRAM

Verzija 1

Glavni program sada izgleda ovako:

```
100 REM GLAVNI PROGRAM
110 GOSUB 4000                [INICIJALIZOVANJE]
120 M$="TEHNICKA KNJIGA"
130 GOSUB 1000                [ISPISIVANJE PRVE PORUKE]
140 GOSUB 2000                [PAUZA]
150 PAPER=2: GOSUB 3000       [CRVENA OSNOVA]
160 M$="KATALOG RACUNARSKE LITERATURE NOV. 85"
170 GOSUB 1000                [ISPISIVANJE DRUGE PORUKE]
180 GOSUB 2000                [PAUZA]
```

```
190 PAPER=4: GOSUB 3000      [PURPURNA OSNOVA]
200 M$= "LAKO PROGRAMIRANJE ZA CRAY—1"
210 GOSUB 1000               [ISPISIVANJE TREĆE PORUKE]
220 GOSUB 2000               [PAUZA]
230 PAPER=7: GOSUB 3000      [ŽUTA OSNOVA]
240 M$= "IGRE NA VASEM VAX-U"
250 GOSUB 1000               [ISPISIVANJE ČETVRTE PORUKE]
260 GOSUB 2000               [PAUZA]
270 PAPER=8: GOSUB 3000      [NARANDŽASTA OSNOVA]
280 M$= "KOMPAJLER GENERATOR ZA PRAVE POCETNIKE"
290 GOSUB 1000               [ISPISIVANJE PETE PORUKE]
300 GOSUB 2000               [PAUZA]
310 PAPER=13: GOSUB 3000     [SVETLO ZELENA OSNOVA]
320 GOTO 100                 [OPET SVE ISPOČETKA]
```

Verzija 2

Još nismo napisali potprograme, ali pre nego što to učinimo, već je jasno da glavni program nije baš sasvim zadovoljavajući. U njemu ima mnogo ponavljanja, isti proces se ponavlja pet puta, a menjaju se samo poruke i kodovi boja. Zar petlja ne bi bila efikasnija?

Bila bi — ali samo ako koristimo *matrice* (o kojima za sada ništa ne znamo sve do poglavlja 25) tako da pet promenljivih:

M$ (1) M$ (2) M$ (3) M$ (4) M$ (5)

sadrži poruke, a:

P (1) P (2) P (3) P (4) P (5)

sadrže boje osnove. Ako brojač petlje S uzima vrednosti od 1 do 5, u programu se na S-tu promenljivu možemo pozvati pomoću M$ (S) ili P (S). Naravno, stvarne vrednosti ovih promenljivih moraju se negde dodeliti. Pogodno mesto za to je potprogram za *inicijalizovanje*. Tako ćemo sada imati:

```
100 REM GLAVNI PROGRAM
110 GOSUB 4000               [INICIJALIZOVANJE]
120 FOR S=1 TO 5             [PETLJA]
130 M$=M$(S): GOSUB 1000     [ISPISIVANJE S-TE PORUKE]
140 GOSUB 2000               [PAUZA]
150 PAPER=P(S): GOSUB 3000   [OSNOVA S-TE BOJE]
160 NEXT S                   [KRAJ PETLJE]
170 GOSUB 2000               [PAUZA]
180 PRINT CHR$(147)          [BRISANJE EKRANA/POKAZIVAČ
                              U POLAZNI POLOŽAJ]
190 GOTO 120                 [OPET SVE ISPOČETKA]
```

Primetimo da nije potrebno *ponovo* inicijalizovanje posle reda 190, pa ne koristimo GOTO 100.

Sada imamo veoma jasnu predstavu strukture celog programa, ali nismo još *napisali* nijedan od potprograma. To je velika prednost pisanja "odozgo nadole". Da nismo koristili ovaj način pisanja programa, trebalo bi da napišemo i sve ostale delove programa, a sve bi verovatno bilo dosta zbrkano. Što je još gore, ceo program bi izgledao obeshrabrujuće i mogao bi da utiče na naše samopouzdanje. Kao što kaže Džonsov prvi zakon računarstva: "Nikad ne ostavljaj za sutra ono što možeš da ostaviš za prekosutra".

U tom smislu:

Vežba 2

Napišite potprograme koji počinju u redovima 1000, 2000, 3000 i 4000, a koji su još uvek neophodni da bi program za reklamiranje mogao da radi. Zatim isprobajte program.

Vežba 3

Izmislite sedam dodatnih naslova koje će izdavačka kuća "Tehnička knjiga" izdati 1987. godine i izmenite program tako da i njih dodate prikazu kataloga.

ODGOVORI

Vežba 1

Dodajte red:

```
115 POKE 53280, 6
```

Vežba 2

Potprogrami su:

```
1000 REM PRIKAZ PORUKE
1010 PRINT                          [PRAZAN RED]
1020 PRINT M$                       [PORUKA]
1030 RETURN
2000 REM PAUZA U TRAJANJU OD OKO 2 SEC
2010 FOR T=0 TO 1500          }
2020 NEXT T                   } "PRAZNA" PETLJA
2030 RETURN
3000 REM POSTAVLJANJE BOJE OSNOVE
3010 POKE 53281, PAPER
3020 RETURN
4000 REM POSTAVLJANJE POCETNIH USLOVA
```

```
4010 DIM M$(5): DIM P(5)          [DIMENZIONISANJE MATRICA,
                                  POGLAVLJE 25]
4020 M$(1)="TEHNICKA KNJIGA"
4030 M$(2)="KATALOG RACUNARSKE LITERATURE NOV. 85"
4040 M$(3)="LAKO PROGRAMIRANJE ZA VAS CRAY-1"
4050 M$(4)="IGRE NA VASEM VAX-U"
4060 M$(5)="KOMPAJLER GENERATOR ZA PRAVE POCETNIKE"
4200 P(1)=2
4210 P(2)=4
4220 P(3)=7
4230 P(4)=8
4240 P(5)=13
4400 PRINT CHR$(147)              [BRISANJE EKRANA/POKAZIVAČ
                                  U POLAZNI POLOŽAJ]
4410 POKE 646,0                   [CRNI ZNAKOVI]
4420 RETURN
```

Nedostajući brojevi redova u potprogramu 4000 izostavljeni su zbog toga da ostave mesta za treću vežbu i nisu od značaja.

Vežba 3

Izmenite dužinu petlje:

```
120 FOR S=1 TO 12
```

Promenite veličine matrica (za detalje vidi poglavlje 25):

```
4010 DIM M$ (12) : DIM P (12)
```

Dodajte dodatne naslove i boje:

```
4070 M$(6)="KONTROLA SVEMIRSKOG TAKSIJA U REALNOM
     VREMENU NA VIC-U 20"
4080 M$(7)="VODIC ZA OPERATIVNI SISTEM UNIX ZA UDAVACE"
4090 M$(8)="GRADNJA KOLA VRLO VISOKOG STEPENA INTE-
     GRACIJE OD STARIH SIBICA"
4100 M$(9)="NE BAS LAKO PROGRAMIRANJE ZA COMMODORE-
     -256"
4110 M$(10)="MASINSKI PROGRAMI ZA NUMERICKO KONTRO-
     LISANJE TOSTERA"
4120 M$(11)="VODIC ZA MREZU IGRACA"
4130 M$(12)="LOGO ZA SISTEM-ANALITICARE"
4250 P(6)=3
4260 P(7)=10
4270 P(8)=5
```

```
4280 P(9)=14
4290 P(10)=15
4300 P(11)=9
4310 P(12)=6
```

Uzgred, primetimo kako bi bilo lako menjati boje u programu koji je strukturisan na navedeni način. Samo bi trebalo izmeniti vrednosti promenljivih P (1) do P (12). Slično se mogu menjati i naslovi knjiga bez izmene glavnog programa.

Poglavlje „Pronalaženje i otklanjanje grešaka I" odnosilo se na „gramatičke" greške. Međutim, naredba može da bude gramatički savršena, a da u programu prouzrokuje besmislenosti.

15 Pronalaženje i otklanjanje grešaka II

GREŠKE U TOKU IZVRŠAVANJA PROGRAMA

Iako "šezdesetčetvorka" ne kaže ništa o sintaksnim greškama, dok ne otkucate RUN, ona bi, u principu, *mogla* to da radi pošto treba samo da pregleda naredbu i utvrdi ima li grešaka. Međutim, postoje i drugi tipovi grešaka koje se ne mogu otkriti, dok se program ne pusti u izvršenje. Te greške nazivaju se greškama u toku izvršenja.

Evo prostog primera:

```
10 FOR P=1 TO 20
20 N=5/(5-P)
30 PRINT N
40 NEXT P
```

Otkucajte RUN. Utvrdićete da je na početku sve u redu i da se dobijaju rezultati:

```
1.25
1.66666667
2.5
5
```

a zatim se pojavljuje poruka:

```
?DIVISION BY ZERO ERROR IN 20
```

Pa šta je pogrešno? Poruka nam kaže da je "šezdesetčetvorka" naišla na nešto sumnjivo u redu 20 koji glasi:

```
20 N=5/(5-P)
```

Sama naredba ne može da bude pogrešna pošto je izvršena već četiri puta i dala četiri navedena broja. Znači, mora da je nešto vezano za promenljivu P, koja

je jedina veličina koja se menja. Otkucajte PRINT P (ili, da bi bilo brže, samo ?P). Na ekranu će se ispisati "5".

Znači, mašina pokušava da izračuna:

$$\frac{5}{5-5} = \frac{5}{0}$$

a to ne može pošto bi rezultat bio veći nego bilo koji broj koji možete da zamislite, pa ma koliko bila velika memorija "šezdesetčetvorke" ovaj broj ne bi mogao da se smesti u nju. Zbog toga vas "šezdesetčetvorka", veoma razumno, obaveštava kad zahtevate da nešto podelite nulom i neće to ni pokušati, nego će vas samo obavestiti o tome.

Ova greška se može pojaviti u mnogo nejasnijim okolnostima nego što su ove. Recimo:

```
30 INPUT P, Q, R
40 A=(P+Q-R)/(5+(P-R)*(P-R)-2*Q)
```

Probajte sa vrednostima 7, 15 i 2 za P, Q i R pa ćete se uveriti.

Vežba 1

Koje vrednosti će izazvati poruku greške "deljenja nulom" u sledećim primerima:

1. A=7/(B-C)
2. R=P+Q/(2*P-Q)
3. M=R+2/(R*R+R*R*R)

ODGOVORI

Vežba 1

Možete da se izvučete tako što ćete svim promenljivim dodeliti vrednost 0 ali probajte i ovo:

1. Uzmite jednake vrednosti za B i C
2. Neka Q bude dvostruko veće od P; na primer Q=7, P=3.5
3. Uzmite R=-1

Uvek *možete* (i uvek treba) da sprečite deljenje nulom uvođenjem ispitivanja. Npr. u navedenom primeru možete napisati:

```
20 INPUT B, C
30 D=B-C
40 IF D=0 THEN PRINT "NIJE MOGUCE IZVRSITI DELJENJE":
   GOTO 20
50 A=7/D
```

Računari mogu da obrađuju i reči ili druge vrste simboličke notacije isto tako kao i brojeve.

16 Znakovni nizovi

Poštar je zakucao . . . ima pismo. Za *vas*. Veoma lično pismo: "Drug Malovrazić" kaže pismo "izabrani ste od svih stanovnika mesta Gornje Strnjike da besplatno dobijete . . .

Baš lepo. Ali prvi komšija, stari Krivokapić, dobio je pismo istog sadržaja. Ispostavlja se da je pismo dobio svaki stanovnik Gornjih Strnjika.

Evo kako se to postiže:

```
5 PRINT CHR$(147)
10 INPUT "VASE IME";N$
20 INPUT "GDE ZIVITE";T$
30 PRINT CHR$(147)
35 PRINT:PRINT:PRINT
40 PRINT "DRUG▽";N$
45 PRINT
50 PRINT "▽▽▽▽▽▽▽▽IZABRANI STE OD STANOVNIKA MESTA"
60 PRINT T$;"▽DA BESPLATNO DOBIJETE BASTENSKU"
65 PRINT "STAZU."
70 PRINT "NADAMO SE DA CETE BITI ZADOVOLJNI."
80 PRINT:PRINT
90 PRINT "▽▽▽▽▽▽▽▽▽▽▽▽S POSTOVANJEM"
100 PRINT "▽▽▽▽▽▽▽▽▽▽▽▽MILAN POPOVIC"
110 PRINT
120 PRINT "NAPOMENA: POSTARINA IZNOSI 9000000 DIN."
130 PRINT:PRINT:PRINT:PRINT:PRINT
```

Otkucajte RUN, a zatim upišite:

```
MALOVRAZIC
GORNJE STRNJIKE
```

ili:

```
REPONJIC
VELIKI POTOK
```

i tako dalje. Pokušajte i sa drugim imenima i mestima. Hmmmmm. . .

Sada zamislite da se imena i adrese automatski dovode iz neke banke podataka tako da se na sat štampa hiljade pisama.

Interesantno je, ma kako ceo primer bio "šašav", da nema apsolutno nikakvog *izračunavanja*. Samo memorija i nekoliko veoma jednostavnih manipulacija pisanim tekstom. Računar može da obrađuje i ovo, isto tako dobro kao i brojeve, pošto ima mogućnost memorisanja *znakovnih nizova*. Znakovne promenljive su u programu — imaju znak $ na kraju naziva promenljive.

O znakovnim nizovima i znakovnim promenljivima je već bilo reči u poglavlju 8. Sada ću vam pokazati kako se njima manipuliše.

OLANČAVANJE

Ovo je reč kojom se označava "spajanje". Da biste spojili dva znakovna niza u jedan, stavite znak "+" između njih. Na primer:

```
PRINT "PARA"+"PET"
```

daje na ekranu:

```
PARAPET
```

Primetimo da je redosled bitan: "PET"+"PARA" daje PETPARA. Primetimo i da navodnici nisu *deo* znakovnog niza. Kad se niz ispisuje, ili obrađuje na neki drugi način, navodnici služe samo za to da se naznače granice niza.

Možete da kombinujete i više nizova:

```
10 INPUT B$, C$
20 PRINT B$+B$+C$
```

Šta će biti ako upišete:

```
B$="C"              C$="M"
B$="KO"             C$="S"
B$="TIHO▽"          C$="SARLOTA"
```

Zašto?

Ako se određeni niz znakova (što *uključuje* i grafičke znakove) ponavlja više puta u toku programa, možda ćete utvrditi da se isplati da se ta vrednost dodeli nekoj znakovnoj promenljivoj.

DUŽINA ZNAKOVNOG NIZA

Naredba:

```
LEN
```

daje *dužinu* znakovnog niza — odnosno broj znakova od kojih je niz sačinjen. Na primer:

```
LEN ("FIDO")=4
LEN ("AAAAAAAAAAA")=11
LEN ("2+2=5")=5
LEN ("")=0
```

gde je "" *prazan niz* odnosno niz *bez* ijednog znaka. Uopšte, da biste dobili dužinu znakovnog niza K$, koristite:

```
LEN (K$)
```

Da biste ovo proverili, probajte sa ovim programom:

```
10 INPUT "ZNAKOVNI NIZ"; K$
20 PRINT K$; "▽ IMA DUZINU"; LEN (K$)
30 GOTO 10
```

Program je jasan sam po sebi.

OBRTANJE REČI

Sledeći program prihvata reč, slovo po slovo, i koristi olančavanje znakovnih nizova da bi dao ista slova, ali obrnutim redosledom (upotrebom naredbi kao MID$, LEFT$ i RIGHT$, moglo bi se postići da se odjednom upiše cela reč, ali ovde zbog jednostavnosti uzimamo slovo po slovo. Za detalje vidi sledeće poglavlje).

```
10 INPUT "PRVO SLOVO"; F$
20 INPUT "SLEDECE SLOVO"; N$
30 IF N$="0" THEN 60
40 F$=N$+F$
50 GOTO 20
60 PRINT CHR$ (147)
70 PRINT F$
```

Da biste naznačili kraj upisa, upišite "0" umesto slova.

Da bismo videli kako ovo funkcioniše, uzećemo reč OKSID. Po naredbi u redu 10 upisujemo prvo slovo "O", pa tako F$ dobija vrednost "O". Po naredbi u redu 20 upisujemo drugo slovo "K". Sada red 40 menja F$ u:

```
N$+F$="K"+"O"="KO"
```

a red 50 nas vraća na red 20 koji traži sledeće slovo koje je sada "S". Tako F$ postaje:

```
N$+F$="S"+"KO"="SKO"
```

i tako dalje:

Upisujemo "I": F$=N$+F$="I"+"SKO"="ISKO"
Upisujemo "D": F$=N$+F$="D"+"ISKO"="DISKO"
Upisujemo 0: program prelazi na red 60 i ispisuje: DISKO.

Ključna tačka je redosled po kojem se spajaju znakovni nizovi u redu 40. Šta će se desiti ako red 40 izmenite u:

```
40 F$=F$+N$
```

Vežba 1

Postoji igra reči u kojoj prvi igrač napiše jednostavnu rečenicu kao:

JUČE SAM VIDEO MAJMUNA

Sledeći igrač dodaje pridev koji opisuje majmuna:

JUČE SAM VIDEO PLAVOG MAJMUNA

Sledeći igrač dodaje još jedan pridev:

JUČE SAM VIDEO ŽESTOKOG PLAVOG MAJMUNA

i tako dalje, a rečenica postaje sve duža i duža (sve dok neko ne zaboravi koju reč treba dodati) tako da se na kraju dobija nešto kao:

JUČE SAM VIDEO SKRUPULOZNOG LENJOG LAKOVERNOG AROMATIČNOG OSVEŽAVAJUĆEG ENORMNOG POSEBNOG ARMIRANOG NEZADOVOLJNOG PLASTIČNOG SLAVNOG ŽESTOKOG PLAVOG MAJMUNA

ili nešto slično.

Napišite program koji će igračima omogućiti da prave takve rečenice dodavanjem po jednog prideva u svakoj fazi.

ODGOVORI

Vežba 1

```
10 Y$="JUCE SAM VIDEO▽"
20 B$="MAJMUNA"
30 A$=" "
40 PRINT Y$+A$+B$
50 INPUT "PRIDEV U AKUZATIVU"; I$
60 A$=I$+"▽"+A$
70 GOTO 40
```

Izvlačenjem delova znakovnog niza (čime se dobija podniz) možete da manipulišete rečima. Navedeni primeri uključuju računarske spunerizme i program koji će vam reći kog dana u nedelji ste se rodili.

17 Podnizovi

U prethodnom poglavlju uveo sam pojam *niza* znakova. Sada ću govoriti o naredbama:

LEFT$
RIGHT$
MID$

koje vam omogućavaju da izaberete deo niza — *podniz*. To su veoma korisne komande za opšte rukovanje nizovima.

LEVO, DESNO I U SREDINI

Da biste izabrali levi kraj znakovnog niza, koristite naredbu:

```
LEFT$ (X$, N)
```

koja daje N levih znakova znakovnog niza X$. Na primer:

```
10 LET X$ = "PRENOSNI AUTOMAT"
20 LET Y$ = LEFT$ (X$, 6)
30 PRINT Y$
```

daje PRENOS. Na sličan način se dobija i desnih N znakova naredbom:

```
RIGHT (X$, N)
```

pa:

```
10 LET X$ = "PRENOSNI AUTOMAT"
20 LET X$ = RIGHT$ (X$, 3)
30 PRINT X$
```

daje MAT (Uzgred, reči LET u naredbama nisu obavezne i mogu se izostaviti kao i kod dodele vrednosti brojnoj promenljivoj).

Na kraju, u smislu ovih ideja, postoji i naredba:

```
MID$ (X$, M, N)
```

koja daje N znakova niza X$ počev od znaka na poziciji M. U pogledu M postoji ograničenje: on mora biti veći od 0. Ako se izostavi N, uključuje se sve od pozicije M pa dalje. U navedenom primeru, ako naredbu 20 izmenimo u:

```
20 LET X$ = MID$ (X$, 10, 4)
```

dobićemo AUTO.

Tipična upotreba ove naredbe je ispisivanje dana u sedmici, ako je dat redni broj dana (1 do 7 počev od nedelje):

```
10 W$ = "NEDPONUTOSRECETPETSUB"
20 INPUT "KOJI DAN SEDMICE"; D
30 Y$ = MID$ (W$, 3 * D — 2, 3)
40 PRINT Y$
```

gde 3 * D — 2 daje tačan početni položaj u W$ (1, 4, 7, 10, 13, 16 i 19).

Vežba 1

Koristeći znakovni niz "JANFEBMAR . . . . DEC" napišite sličan program za ispisivanje meseca, ako je dat redni broj od 1 do 12.

ZNAKOVNI NIZOVI I BROJEVI

Znakovni niz "493" i broj 493 računar ne tretira na isti način. Možda to nećete primetiti, ako ih samo ispišete:

```
PRINT 493
PRINT "493"
```

daju isti rezultat. Pokušajte sada:

```
PRINT 493+7
PRINT "493"+7
PRINT "493"+"7"
```

utvrdićete da dobijate tri različita rezultata:

```
500
? TYPE MISMATCH ERROR
4937
```

U prvom slučaju dobijamo samo zbir brojeva.

U drugom slučaju, računar pokušava da sabere znakovni niz i broj i daje poruku o *neslaganju tipa* promenljivih (TYPE MISMATCH) što znači da jednostavno to ne može da uradi (šta bi dalo "KVAKA" + 22? Pa dobro, možda, na kraju krajeva, ovo nije baš najbolji primer . . .)

U trećem slučaju računar prosto olančava nizove "493" i "7" i dobija rezultat "4937" a zatim ga ispisuje bez navodnika. Ovo može da bude veoma korisno, pošto sa znakovnim nizovima možete da uradite stvari koje se ne mogu tako lako uraditi sa brojevima. Npr. da biste odredili prvu cifru broja 987654321, potrebno vam je samo LEFT$ ("987654321", 1). Aritmetičkim metodama to je znatno teže. *Međutim*, rezultat je *niz* "9" a ne broj 9, a možda ste hteli da nešto računate sa tom devetkom. Kako?

Znakovni niz u obliku broja (sa navodnicima sa obe strane) može da se pretvori u pravi broj naredbom:

VAL

(skraćeno od "value" = vrednost). Tako je:

VAL ("9")

broj 9. Pokušajte:

PRINT VAL ("493") + VAL ("7")

i videćete da to stvarno funkcioniše.

Postoji i slična naredba:

STR$

koja radi suprotno: ona konvertuje broj u znakovni niz. Na primer:

STR$ (7751) je "7751"

Za primere korišćenja vidi poglavlje 34, odgovori, vežba 1.

RAČUNARSKI SPUNERIZMI

Prečasni V. A. Spuner bio je poznat po svom običaju da zameni prve delove para reči što je često davalo komične efekte. On bi, npr. umesto:

MESECI DANA

rekao:

DESECI MANA

i tako dalje.

Manipulisanjem znakovnim nizovima moguće je postići da računar generiše spunerizme. Da li su oni smešni ili ne, stvar je u prvom redu ukusa, a zatim na korisniku je da odluči. Program o tome nema svoje mišljenje.

Najjednostavnije je programirati one spunerizme kod kojih se zamenjuju samo *prva slova* reči. Pa hajde da napišemo taj program.

```
10 INPUT "PRVA REC"; A$
20 INPUT "DRUGA REC"; B$
30 P$=LEFT$(A$, 1)
40 Q$=MID$(A$, 2)
50 R$=LEFT$(B$, 1)
60 S$=MID$(B$, 2)
70 PRINT A$+""+B$
80 PRINT "SPUNERIZOVANO DAJE:"
90 PRINT R$+Q$+""+P$+S$
100 GOTO 10
```

Otkucajte RUN, a zatim upišite "RUCNA" kao prvu, a "VAGA" kao drugu reč.

Naredba u redu 30 uzima prvi levi znak reči "RUCNA", pa je P$="R". U redu 40 uzima se sve od drugog znaka do kraja tako da je Q$="UCNA". Red 50 daje R$="V" a red 60 S$="AGA".

Redovi 70 i 80 prikazuju na ekranu originalne reči i odgovarajuću poruku a red 90 daje:

```
"V"+"UCNA"+"▽"+"R"+"AGA" = VUCNA RAGA
```

što i nije tako loše, u principu. Strava profa, što bi rekao Prečasni.

VEČITI KALENDAR

Ovaj program kao ulaz ima datum (dan D, mesec M i godinu Y) i na osnovu njega izračunava dan u sedmici.

```
10 A$="033614625035"
20 B$="NEDPONUTOSRECETPETSUB"
30 INPUT "DAN (1—31)"; D
40 INPUT "MESEC (1—12)"; M
50 INPUT "GODINA"; G
60 PRINT "DAN JE";
70 Z=G—1
80 C=INT(Z/4)—INT(Z/100)+INT(Z/400)
90 X=G+D+C+VAL(MID$(A$, M, 1))—1
100 IF M>2 AND (G=4*INT(G/4) AND G<>100*INT(G/400)) THEN
    X=X+1
```

```
110 X=X-7*INT(X/7)
120 PRINT MID$(B$, 3*X+1, 3)
```

Ovaj program prepišite *veoma* pažljivo, a posebno zagrade u redu 100. Otkucajte RUN i upišite (za proveru) 24 za D, 9 za M i 1945 za Y (odnosno 24. septembar 1945.). *Morate* da unesete celu godinu, a ne samo 45, jer ćete inače dobiti pogrešan odgovor. Trebalo bi da dobijete:

```
DAN JE PON
```

odnosno ponedeljak. Pokušajte sa današnjim datumom i svojim datumom rođenja. Pronađite kojeg datuma je počeo prvi svetski rat, pa probajte i sa njim.

Red 10 memoriše dvanaest "brojeva za korekciju meseca" u sažetom obliku kao jedan znakovni niz.

Red 20 formira znakovni niz "dana u sedmici" kao u već navedenom primeru.

Redovi 30—60 su ulazno/izlazne naredbe. Ništa posebno.

Redovi 80—100 vrše komplikovana izračunavanja koja u obzir uzimaju prestupne godine i "korekcije meseca". Primetimo upotrebu naredbe VAL i MID$ u redu 90. Tim naredbama se pronalazi M-ta cifra u A$ i konvertuje u broj.

Red 110 daje broj od 0—6 za dan u sedmici (ne 1—7 kao u ranijem primeru), a red 120 ispisuje dan koristeći naredbu MID$ za B$.

SAT

"Šezdesetčetvorka" ima ugrađeni sat — čak nekoliko. Iz BASIC-a možete da koristite dve standardne promenljive:

```
TI
TI$
```

Prva promenljiva je brojna, a njena vrednost je 60 puta veća od broja sekundi koje su protekle od kad je računar uključen.

Druga je znakovna promenljiva koja daje vreme u satima, minutima i sekundima. Na primer:

```
021143 = 2 sata, 11 minuta, 43 sekunda
150422 = 15 sati, 4 minuta, 22 sekunda
```

TI$ se može vratiti na nulu u bilo koje vreme naredbom:

```
TI$ = "000000"
```

pa će sat meriti vreme proteklo od tog trenutka. TI se ne može vratiti na nulu na ovaj način. Spominjem ih ovde jer je TI$ znakovna promenljiva, pa će nam je malo obrade nizova približiti.

```
10 PRINT CHR$(147)
20 A$=TI$
```

```
30 PRINT "CASOVI:"; LEFT$(A$, 2); "MINUTI:"; MID$(A$, 3, 2);
   "SEKUNDE:"; RIGHT$(A$, 2)
40 GOTO 20
```

Sad na ekranu imate sat. Ekran klizi kao lud, pa možda nije loše da dodate:

```
25 PRINT SHR$ (19)
```

Primetimo da se sekunde neće menjati u jednakim intervalima, pošto je u pitanju i vreme potrebno za ponavljanje programskog ciklusa. Neke vrednosti su preskočene, jer se sat ne očitava u pravom trenutku. Međutim, sat je tačan.

Koristan dodatak svakom programu je potprogram koji daje proteklo vreme. Na samom početku otkucajte:

```
1 TI$ = "000000"
```

Zatim otkucajte svoj program. Međutim, umesto naredbe STOP upišite (recimo) GOTO 10000, a zatim dodajte:

```
10000 A$=TI$
10005 PRINT "NASTAVAK → RETURN"
10010 GET B$: IF B$="" THEN 10010
10020 PRINT "UKUPNO PROTEKLO VREME JE"
10030 PRINT "CAS→"; LEFT$(A$, 2); "MIN→"; MID$(A$, 3, 2);
      "SEC→;"; RIGHT$(A$, 2)
10050 STOP
```

ODGOVORI

Vežba 1

```
10 M$="JANFEBMARAPRMAJJUNJULAVGSEPOKTNOVDEC"
20 INPUT "UNESI MESEC (1—12)"; M
30 Y$=MID$(M$, 3*M—2, 3)
40 PRINT "MESEC:";M;" →";Y$
```

Svaki znak ima svoj kôdni broj. Možete ga koristiti za ispitivanje o kojoj vrsti znaka se radi ili za konverziju iz jednog kôda u drugi. Dajemo primer za Morzeov kôd.

18 ASCII-kodovi

ASCII je skraćenica od "American Standard Code for Information Interchange" ("Američki standardni kôd za razmenu informacija") i predstavlja baš ono što mu ime kaže. Već neko vreme se koristi kao standardni sistem numeričkih kodova za znakove.

Da biste našli kôd znaka K$, tražite:

```
ASC(K$)
```

Evo i jednog programa za probu:

```
10 INPUT K$
20 PRINT ASC(K$)
30 GOTO 10
```

ASCII kodovi su dati u *Priručniku* na stranama 135—137. Znakovi 0—31 su *upravljački znakovi* koje koristi operativni sistem. Znakovi 96—127 su grafički znakovi.

Da biste utvrdili koji znak odgovara određenom kôdu, koristite naredbu koju sam ja već praktično koristio:

CHR$(C)

Evo analognog programa prethodnom programu:

```
10 INPUT C
20 PRINT CHR$(C)
30 GOTO 10
```

Probajte različite brojeve između 0 i 255. Dobićete zanimljive rezultate za brojeve 0—31 pošto ih sistem ne prikazuje na ekranu nego *sluša*.

Jedna od uobičajenih primena ASCII kodova je ispitivanje da li je znak datog tipa. Pretpostavimo da je znak K$. Tada:

```
IF ASC(K$)>47 AND ASC(K$)<58 ...
```

ispituje da li je znak cifra. Slično:

```
IF ASC(K$)>64 AND ASC(K$)<91 ...
```

ispituje da li je znak veliko slovo, a:

```
IF ASC(K$)>95 AND ASC(K$)<128 ...
```

da li je grafički znak. Moguće su i druge kombinacije.

Vežba 1

Napišite program koji prihvata niz znakova i ispisuje ASCII kodove svakog znaka.

GENERATOR MORZEOVOG KÔDA

Sledeći program će prihvatiti ulaznu poruku i ispisati je u Morzeovom kôdu.

```
10 DIM A$(26)
20 A$(1)=".—"
30 A$(2)="—..."
40 A$(3)="—.—."
50 A$(4)="—.."
60 A$(5)="."
70 A$(6)="..—."
80 A$(7)="— —."
90 A$(8)="...."
100 A$(9)=".."
110 A$(10)=".— — —"
120 A$(11)="—.—"
```

```
130 A$(12)=".—. ."
140 A$(13)="— —"
150 A$(14)="—."
160 A$(15)="— — —"
170 A$(16)=".— —."
180 A$(17)="— —.—"
190 A$(18)=".—."
200 A$(19)=". . ."
210 A$(20)="—"
220 A$(21)=". .—"
230 A$(22)=". . .—"
240 A$(23)=".— —"
250 A$(24)="—. .—"
260 A$(25)="—".— —"
270 A$(26)="— —. ."
```

Ovim se samo definišu kodovi za slova A—Z u obliku znakovne matrice. Verovatno možete da smislite i zgodniji način da se to uradi. A sada sam program:

```
300 INPUT "UNESITE PORUKU"; M$
310 PRINT CHR$(147)
320 FOR I=1 TO LEN(M$)
330 C=ASC(MID$(M$, I, 1))
340 IF C<32 OR C>32 AND C<65 OR C>90 THEN 380
350 C=C—64
360 IF C<0 THEN FOR J=1 TO 200:NEXT J:PRINT
370 IF C>0 THEN PRINT CHR$(C+64), A$(C)
380 NEXT I
390 STOP
```

Red 300 prihvata poruku koju treba kodirati. Redovi 320 i 380 uspostavljaju petlju koja pretražuje poruku znak po znak. Red 330 pronalazi I-ti znak. Red 340 proverava da li je u pitanju razmak ili slovo i ignoriše znak ako nije nijedno od toga. Red 350 oduzima 64 od ASCII kôda tako da za A dobijamo 1, za B 2 itd. (Primedba: za razmak dobijamo 32—64, odnosno negativan broj, vidi dalje u tekstu).

Red 360 obezbeđuje vremensku pauzu, ako je u pitanju razmak da bi se odvojile reči i ispisuje red.

Red 370 ispisuje slovo i njegov Morzeov kôd (iz matrice A$).

U ovom trenutku imamo nešto što je malo neuobičajeno: *nečujni* Morzeov kôd. Upotrebom zvučnog integrisanog kola SID (koje se nalazi u "šezdesetčetvorci") moguće je proizvesti i zvučni Morzeov kôd (vidi vežbu 2 u poglavlju 30).

ODGOVORI

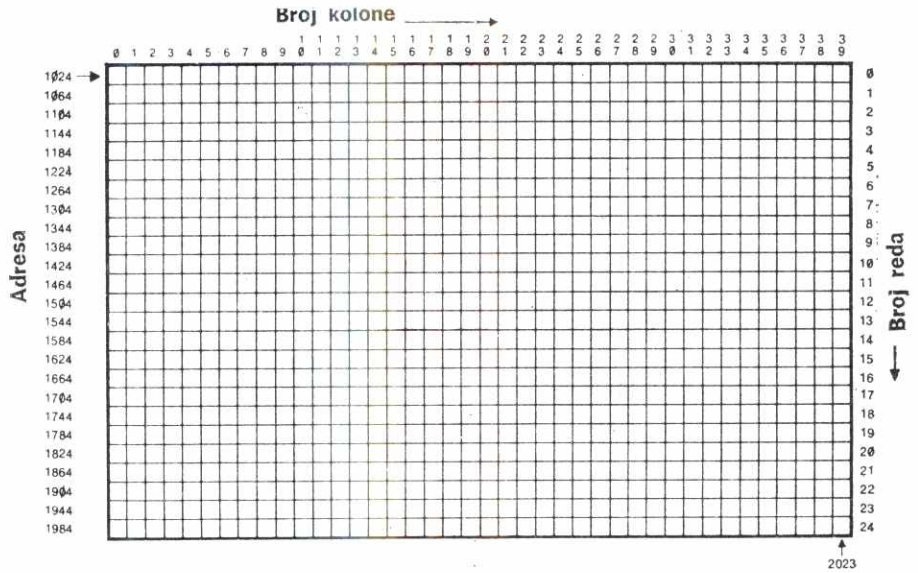
Vežba 1

```
10 INPUT "NIZ";S$
20 FOR I=1 TO LEN(S$)
30 K$=MID$(S$, I, 1)
40 PRINT ASC(K$)
50 NEXT I
```

Informacije potrebne za formiranje prikaza na ekranu nalaze se u dve memorijske zone. Njima programer može direktno da pristupi i iskoristi ih za upravljanje prikazom.

19 Ekranska memorija i memorija boja

U poglavlju 7 spomenuo sam da se prikaz na ekranu može podeliti u 25 redova sa po 40 znakova i da se redovi obeležavaju brojevima od 0—24, a kolone brojevima od 0—39. Na sl. 19.1 to je i prikazano. Primetimo da ekran sadrži tačno 1000 ($=25\times40$) znakova. Svaki znak se, svojim kôdom, može definisati kao jedan bajt informacije. Zbog toga ne iznenađuje kad utvrdite da u memoriji postoji zona sa 1000 adresa u kojoj se drže sve ove informacije.



Sl. 19.1 — Struktura ekranske memorije imitira strukturu samog ekrana

EKRANSKA MEMORIJA

Odgovarajuća memorijska zona počinje na adresi 1024 i završava na adresi 2023. Ova zona se naziva *ekranska memorija* ili *video RAM*. Računarska memorija

po svojoj prirodi nema pravougaonu strukturu kao na sl. 19.1. U njoj je sve u vidu dugačkog reda adresa. U ovom slučaju adrese se kreću po kolonama i prelaze u prvu kolonu sledećeg reda tek kada se red završi — baš kao čitanje knjige. Drugim rečima, gornji red ekranske memorije je na adresama:

1024 1025 1026 1027 ... ... 1062 1063

sledeći red (red 1) je na adresama:

1064 1065 1066 1067 ... ... 1102 1103

itd. tako da je red 24 na adresama:

1984 1985 1986 1987 ... ... 2022 2023

Opšte pravilo je da red R, kolona C odgovara adresi:

1024+40*R+C

Znači, da bismo prikazali određeni znak u redu R i koloni C, sve što je potrebno to je da na tu adresu naredbom POKE upišemo odgovarajući kôd:

POKE 1024+40*R+C, kod znaka

Postoje dva mala problema:

1. Potreban je kôd koji *nije* ASCII, nego kôd dat u dodatku E *Priručnika* na strani 132. Grubo uzevši to je ASCII kôd minus 64 za slova (i neke druge znakove), minus 32 za grafičke znakove i nezavisan za brojeve.

2. Postoje dva skupa znakova koji mogu da se prikažu. Skup 1 sadrži velika slova, a skup 2 mala. Da biste izabrali skup znakova treba promeniti sistemsku promenljivu na adresi 53272:

Skup 1: POKE 53272,29
Skup 2: POKE 53272,31

Ako do sada niste pokušali da menjate skup znakova, onda je to skup i sa velikim slovima. Preporučujem vam da ga se za sada držite dok ne ovladate osnovnim tehnikama.

EKRAN I NAREDBA POKE

Pretpostavimo da želimo da nacrtamo okruglu loptu u redu 12, a koloni 20 (skoro u samom centru ekrana). Adresa je:

1024+40*12+20=1524

a kôd za loptu je (dodatak E *Priručnika*):

81

Znači, potrebna vam je naredba:

POKE 1524,81

Pokušajte sa ovom naredbom u direktnom načinu rada. Pre toga pritisnite RUN/STOP+RESTORE.

Da li se nešto desilo?

Ne. Prilično čudno pošto bi prema *Priručniku* (strana 64 red 2) na ekranu trebalo da se pojavi lopta. *Priručniku* se prilično može verovati, ali ovoga puta je ispala greška. U stvari, na ekranu *jeste* lopta, ali je ne možete videti pošto je plava, a ne bela. Izmenite boju osnove:

POKE 53281,7

Ekran će postati žut, a lopta će se pojaviti.

Sada možete da eksperimentišete. Isprobajte:

POKE 1525,81

i videćete drugu (plavu) loptu.

Vežba 1

Koje naredbe POKE su potrebne da se prikaže:

1. Znak M u redu 7, koloni 9?
2. Znak karo (iz karata) u redu 20, koloni 32?
3. Znak "PI" u redu 11, koloni 8?

(pretpostavimo da koristimo skup znakova 1)

POTPROGRAM "PRINT AT"

Često poželim da na ekranu ispišem neki znak na određenom mestu. Neke verzije BASIC-a imaju naredbu PRINT AT, kojom se to lako rešava. Ona izgleda otprilike ovako:

PRINT AT 10, 15, "Z" (*)

Međutim, "šezdesetčetvorkina" verzija BASIC-a (koja je — rekao bih — slučajno malo rudimentarna) to ne omogućava. Rešenje je da se napiše standardni potprogram koji to radi. Smestiću ga u red 10000, uglavnom zato što je to znatno veći broj nego što ćete normalno koristiti, tako da neće smetati onome što pišete. Slično ću uraditi i sa drugim korisnim potprogramima, ali, naravno, sa drugim brojevima redova. Tako počinjemo da formiramo *biblioteku* korisnih potprograma. To će u velikoj meri poboljšati mogućnosti "šezdesetčetvorke" i uštedeti mnogo rutinskog rada.

Dobar princip je da potprogrami budu prilično *opšti*. Nema smisla, npr. napraviti potprogram koji će ispisati "X" u izabranom redu i koloni, ako se uz mali dodatni napor može ispisati bilo koji znak. Potprogramu su potrebna tri podatka:

RED = broj reda
KOL = broj kolone
CDE = kod znaka

Evo potprograma:

```
10000 REM PRINT AT RED, KOL, CDE
10010 POKE 1024+40*RED+KOL, CDE
10020 RETURN
```

Treba paziti da se nazivi promenljivih ne poklope sa nazivima iz programa.

Ovaj potprogram možete da koristite baš kao naredbu PRINT AT. Npr. pošto je za "Z" kod 26, učinak prethodno navedene naredbe (*) možete postići ovako:

```
RED=10: KOL=15: CDE=26: GOSUB 10000
```

što nije mnogo duže od naredbe PRINT AT.

Zapamtite kako se poziva potprogram. Najpre se promenljivim *dodele* vrednosti, a zatim *poziva* GOSUB. Promenljive koje se koriste u potprogramima često se nazivaju *prenosni parametri*. Uvek ih morate definisati (sa pravim vrednostima) *pre* pozivanja potprograma. Često se u potprogramu izračunavaju vrednosti drugih promenljivih. To su *povratni* parametri.

Na primer, potprogram za konverziju iz binarnog u decimalni oblik u redu 500 programa iz poglavlja 12 ima prenosni parametar C, a povratni K$.

Vodite računa o tome da nazive tih promenljivih ne koristite drugde u programu sa različitim značenjem. To može da izazove probleme.

POKRETNI ZNAK

Sada ću vam prikazati nešto iz računarske klasike, nešto što je osnova mnogih ranih arkadnih video-igara, a predstavlja odličan uvod u pokretnu grafiku i upotrebu naredbi za grananje.

Najpre upišite potprogram "PRINT AT" iz prethodnog odeljka, pa dodajte:

```
10 PRINT CHR$(147): POKE 53281,7
20 C=1: R=3
30 H=1: V=1
40 RED=R: KOL=C: CDE=35: GOSUB 10000
50 C=C+H: R=R+V
60 IF C=0 OR C=39 THEN H=-H
70 IF R=0 OR R=24 THEN V=-V
80 GOTO 40
```

Kada otkucate RUN, na ekranu ćete dobiti brzorastući red znakova "#", koji se "odbija" od ruba ekrana. Efekat "odbijanja" se postiže u redovima 60 i 70. Ideja je u tome da C i R daju tekući položaj pokretnog znaka, a H i V su promene tekućeg položaja (H — horizontalno, V — vertikalno). Na početku H i V imaju

vrednost i tako da se znak pomera dole i desno, ali svaki put kada dođe do ruba, bilo H bilo V menjaju smer. Primetimo upotrebu potprograma u redu 40.

Iluzija kretanja je prilično jaka, ali je kvari trag znakova koji ostaje na ekranu. Da bi se iluzija poboljšala, možemo da *izbrišemo* znak svaki put kad prikažemo sledeći. Prvo treba da *zapamtimo* njegov položaj:

```
45 C0=C: R0=R
```

Sada, kada se C i R promene (u redu 50), C0 i R0 se ne menjaju. Kad smo, u redu 40, prikazali novi znak, brišemo stari (prikazivanjem razmaka, kôd 32). Dobro mesto za to je red 75 (brisanje treba što više odložiti tako da slika što manje treperi):

```
75 RED=R0: KOL=C0: CDE=32: GOSUB 10000
```

Eto!

Umesto jednog znaka hajde da nacrtamo glistu koja se sastoji iz četiri znaka u redu. To se može postići tako što se u programu dodaje neka vrsta "kašnjenja" naredbe za brisanje, te se u svakom prolazu položaj brisanja pomera za jedan red sve dok se ne aktivira. Izbrišite red 45 i zamenite ga sa:

```
42 C0=C1: R0=R1
44 C1=C2: R1=R2
46 C2=C3: R2=R3
48 C3=C : R3=R
```

Ako možete da utvrdite kako ovo kašnjenje funkcioniše, nećete imati nikakvih problema sa Vežbom 2.

Vežba 2

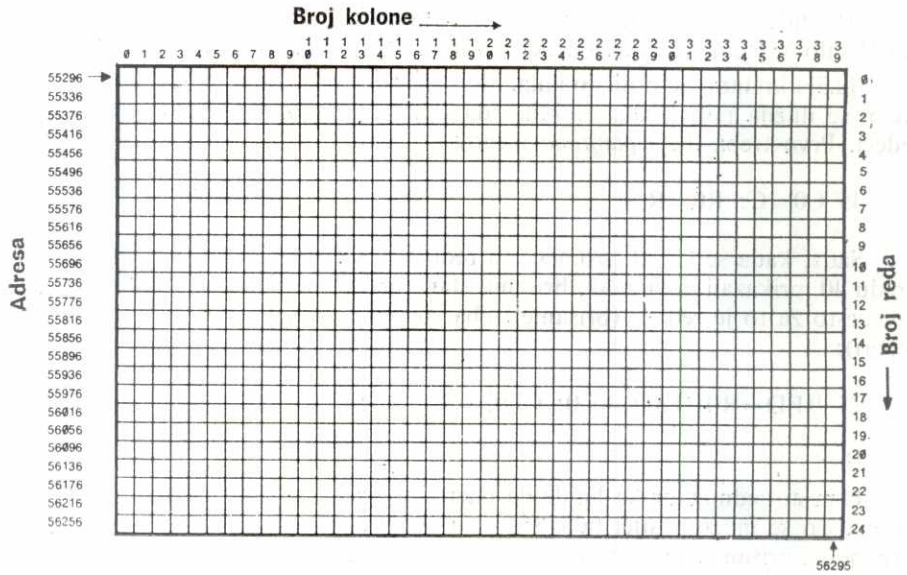
Zamenite redove 42—48 sa sedam redova koji daju glistu od 7 znakova.

MEMORIJA BOJA

Vratimo se sada na već pomenutu grešku u *Priručniku*. Ono što se desilo, to je da je znak *bio* prikazan, ali je bio pogrešne boje koja je slučajno bila boja osnove. To je jedna nezgodna karakteristika (karakteristika je greška koju ne možete ispraviti), ali možemo je zaobići tako što sami upravljamo bojama. Ja sam to rešio tako što sam promenio boju osnove, što ni u kom slučaju nije loše rešenje. Međutim, možete da upravljate bojom znakova tako što ćete naredbom POKE upisati odgovarajuće kodove u jednu drugu memorijsku zonu — *memoriju boja* ili *video-RAM*. Ova memorijska zona počinje na adresi 55296 i završava na adresi 56295 i odgovara ekranu na potpuno isti način na koji i ekranska memorija. Red 0 je na adresama:

```
55296  55297  55298  55299  ...  ...  ...  55334  55335
```

itd. (vidi sliku 19.2). Znači, kôd boje za red R i kolonu C nalazi se na adresi:



Sl. 19.2 — Struktura memorije boja je ista kao i struktura ekranske memorije, ali na drugim adresama

55296 + 40*R + C

Izbrišite ekran pa upišite:

POKE 1524,81

i videćete da se ništa nije desilo. Sada otkucajte:

POKE 55796,5

i videćete zelenu loptu. Kodovi boja su dati u poglavlju 13, pa možete da vidite da 5 znači "zeleno". Adresa koja odgovara redu 12 i koloni 20 je:

55296 + 40*12 + 20 = 55796

a nju smo i koristili.

Naravoučenije: Kad naredbu POKE koristite za ekransku memoriju treba da koristite POKE i za boju u memoriji boja.

Vežba 3

Kao vežba 1, ali sada neka boje budu:

1. Crvena
2. Svetlo zelena
3. Bela

Možete da izmenite potprogram "PRINT AT" tako da birate i boju. Samo dodajte promenljivu CR za kôd boje i red:

```
10015 POKE 55296+40*RED+KOL, CR
```

Ako zaboravite da promenljivoj CR dodelite vrednost, dobićete 0 (crnu boju). Evo primera upotrebe poboljšanog potprograma "PRINT AT":

```
10 PRINT CHR$(147)
20 FOR T=0 TO 15
30 RED=T+2:KOL=T+2:CDE=T+1:CR=T:GOSUB 10000
40 NEXT T
50 GOTO 50
```

Dodajte redove 10000, 10010, 10015, 10020 i otkucajte RUN. Da biste završili program, pritisnite STOP. Red 50 sprečava da neka poruka greške ne pokvari prikaz.

DINAMIČKO OBRTANJE REČI

Kao poduži primer dajemo program koji prihvata reč dužine do 21 znak i obrće je na ekranu pomerajući slova.

```
100 DIM X$(21)
110 PRINT CHR$(147)
120 PRINT "UPISI REC KOJU TREBA PREOKRENUTI"
130 PRINT
140 INPUT W$
150 WL=LEN(W$)
160 IF WL=2*INT(WL/2) THEN WL=WL+1: W$=W$+"▽"
170 WH=(WL-1)/2
180 FOR T=1 TO WL: X$(T)=MID$(W$, T, 1): NEXT T
200 REM PRIKAZIVANJE RECI
210 PRINT CHR$(147)
220 R=12
230 FOR T=1 TO WL
240 C=19-WH+T: C$=X$(T): GOSUB 2000
250 NEXT T
300 REM PRVI DEO ROTACIJE
310 FOR S=1 TO WH
320 FOR T=S TO WH
330 C=20+T: R=S+11: C$="▽": GOSUB 2000
340 R=R+1: C$=X$(C+WH-19): GOSUB 2000
350 C=20-T: R=13-S: C$="▽": GOSUB 2000
```

```
360 R=R-1: C$=X$(WH+C-19): GOSUB 2000
370 NEXT T: NEXT S
400 REM DRUGI DEO ROTACIJE
410 FOR S=1 TO WH
420 FOR T=1 TO S
430 C=19-WH+S: R=11-WH+T: C$="▽": GOSUB 2000
440 C=C+1: C$=X$(T): GOSUB 2000
450 C=21+WH-S: R=13+WH-T: C$="▽": GOSUB 2000
460 C=C-1: C$=X$(WL+1-T): GOSUB 2000
470 NEXT T: NEXT S
500 REM TRECI DEO ROTACIJE
510 FOR S=1 TO WH
520 FOR T=1 TO WH+1-S
530 R=11-WH+T: C=19+S: C$="▽": GOSUB 2000
540 C=C+1: C$=X$(T): GOSUB 2000
550 C=21-S: R=13+WH-T: C$="▽": GOSUB 2000
560 C=C-1: C$=X$(WL+1-T): GOSUB 2000
570 NEXT T: NEXT S
600 REM CETVRTI DEO ROTACIJE
610 FOR S=1 TO WH
620 FOR T=1 TO S
630 R=13+WH-S: C=19-WH+T: C$="▽": GOSUB 2000
640 R=R-1: C$=X$(WL+1-T): GOSUB 2000
650 R=11-WH+S: C=21+WH-T: C$="▽": GOSUB 2000
660 R=R+1: C$=X$(T): GOSUB 2000
670 NEXT T: NEXT S
680 STOP


2000 REM PRINT AT R, C, C$
2010 CD=ASC(C$)
2020 IF CD>64 THEN CD=CD-64
2030 POKE 1024+40*R+C, CD
2040 POKE 55296+40*R+C, 7
2050 RETURN
```

Neću da detaljno objašnjavam kako ovaj program radi, pošto u njemu ima nešto komplikovanijih proračuna mesta na koje se neki znak pomera. Primetimo da se potprogram često koristi. On je jedna varijacija našeg uobičajeno korišćenog potprograma "PRINT AT" koji direktno prihvata znakove i konvertuje kod. Za detaljnije objašnjenje naredbe ASC vidi poglavlje 18. Da biste dobili jednu interesantnu varijaciju, izbrišite redove 330, 350, 430, 450, 530, 550, 630, 650.

ODGOVORI

Vežba 1

1. POKE 1024+40*7+9,13 ili POKE 1313,13
2. POKE 1024+40*20+32,88 ili POKE 1696,88
3. POKE 1024+40*11,94 ili POKE 1472,94

Vežba 2

Izbrišite redove 42—48 i dodajte:

```
41 C0=C1: R0=R1
42 C1=C2: R1=R2
43 C2=C3: R2=R3
44 C3=C4: R3=R4
45 C4=C5: R4=R5
46 C5=C6: R5=R6
47 C6=C: R6=R
```

Očigledno je da mora da postoji lepši način da se ovo uradi. Pročitajte o matricama (poglavlje 25) i pokušajte da poboljšate program.

Vežba 3

Dodajte naredbama POKE iz vežbe 1:

1. POKE 55296+40*7+9,2 ili POKE 55582,2
2. POKE 55296+40*20+32,13 ili POKE 55968,13
3. POKE 55296+40*11+8,1 ili POKE 55744,1

Korišćenjem specijalnog kasetofona možete da sačuvate programe na kaseti i da ih učitate kasnije kad želite da ih koristite. Evo kako.

20 Kasete

Kad napišete neki program koji radi nešto lepo i kad ste njime posebno zadovoljni (i samim sobom) sigurno je da ne želite da ga kucate svaki put kad želite nekome da ga pokažete ili sami da ga koristite. Međutim, kad isključite "šezdesetčetvorku", sadržaj memorije se gubi (to se dešava sa svim kućnim računarima, pošto oni koriste tzv. dinamički RAM kojem je potrebna električna energija da bi radio).

Ili, ako vam ponestane vremena, dok pišete neki poduži program, ili vam je privremeno dosta rada na pisanju programa, sigurno je da ne želite sledeći put ponovo da ga kucate.

Teško da računar možete da ostavite stalno uključen na struju. Umesto toga prenesite program na traku komandom:

SAVE (engl. sačuvati, spasiti)

Postoji i druga naredba:

LOAD (engl. napuniti, učitati u računar)

koja će stvar skinuti sa trake i vratiti u "šezdesetčetvorku". Naravno, ima još nešto pa je zbog toga i napisano ovo poglavlje.

KASETOFON

Biće vam potreban specijalni kasetofon. Običan kućni kasetofon *neće* biti dobar — nedostaju mu neki neophodni priključci i komande. Kasetofon je tipa C2N, a možete ga kupiti u istoj prodavnici gde i "Commodore". On se može koristiti sa "šezdesetčetvorkom" kao i sa računarima VIC, PET i CBM. *Priručnik* ga naziva malo nespretno *datassette unit*. Kasetofon se priključuje na zadnju stranu "šezdesetčetvorke". Konektor ima urez tako da ne može da se priključi naopako.

Upozorenje: Pre priključivanja kasetofona isključite napajanje, jer inače može doći do oštećenja računara. Isto tako ne izvlačite konektor kasetofona, dok je napajanje uključeno.

Kasetofon, koji sam kupio, imao je metalnu foliju za uzemljenje. Izgleda da ona nije neophodna, ali proverite sa vašim prodavcem ako niste sigurni.

"Šezdesetčetvorka" će upravljati većinom funkcija kasetofona, ali ćete vi sami morati da pritisnete neku dugmad. Možda treba u grubim crtama objasniti šta se dešava pošto će tako neke naredbe zvučati razumnije.

Osnovna ideja je da računar znakove programa pretvori u niz zvučnih signala kao čujni Morzeov kod. Ovi signali se snimaju na traku baš kao neko muzičko delo. Kad program želite ponovo da učitate, ovi signali se šalju u računar.

Primetimo da ne možete da snimite program na vodeći plastični deo trake, da ne možete da ga učitate u računar, ako nije snimljen ili je na pogrešnom mestu i da možete greškom da izbrišete već snimljen program, ako na isti deo trake snimite drugi. Dakle, budite pažljivi!

MERE PREDOSTROŽNOSTI

Da bi rezultati bili što bolji i da bi vam život bio što lakši:

1. Proverite da li ste sve dobro povezali.

2. Proverite da li je sve priključeno na mrežni napon i da li je uključeno. Proverite da li se kasetofon okreće.

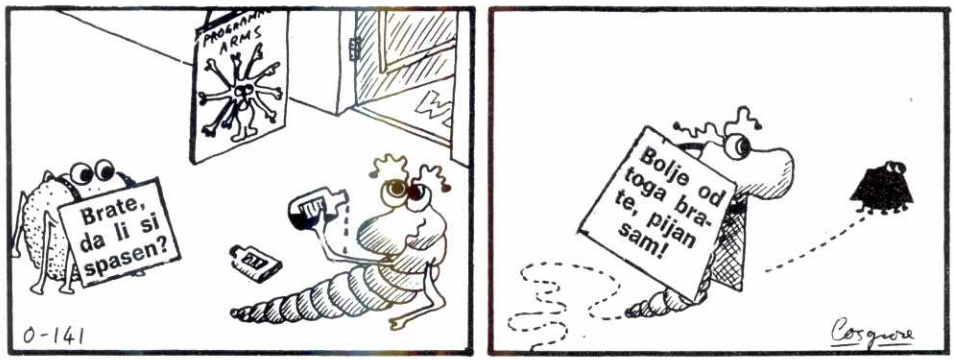
3. *Očistite* i *demagnetizujte* glave za snimanje i brisanje na kasetofonu. To su dva mala metalna bloka preko kojih klizi traka. Magnetne čestice na njima mogu da smanje kvalitet snimanja i da dovedu do teškoća.

4. Koristite traku dobrog kvaliteta. Računarske kasete C12 i C15 su idealne, ali će biti dobra i traka C30 srednjeg kvaliteta. Jeftine trake će *možda* funkcionisati, ali neće dugo trajati, a imaju tendenciju ostavljanja sitnih magnetnih čestica iza sebe pa je potrebno češće čišćenje.

5. *Ne* koristite trake duže od 30 minuta pošto na taj način možete da oštetite kasetofon.

6. Premotajte traku na početak tako da znate gde ste. Koristite trake na kojima ništa nije snimljeno (tako će vam biti lakše da čujete šta se dešava).

7. Premotajte traku dok plastični vodeći deo trake ne pređe preko glave. Nije dobro da pokušate snimanje na vodećem delu — to ne funkcioniše.



TESTIRANJE KOMANDE SAVE

Otkucajte jednostavan ispitni program kao:

```
10 PRINT "JEDNOSTAVNI ISPITNI PROGRAM"
20 GOTO 10
```

ili bilo šta drugo što vam se sviđa. Sada otkucajte:

```
SAVE "FRED"
```

i pritisnite RETURN. Pojaviće se poruka:

```
PRESS RECORD & PLAY ON TAPE
```

Na kasetofonu pritisnite dugmad RECORD i PLAY (budite čvrsti: neka dugmad dobro uskoče na mesto pre nego što oslabite pritisak ili ćete morati ispočetka).

Sada će se ekran izbrisati na nekoliko sekundi. Kasetofon će sam od sebe početi da se okreće, jedno vreme će vrteti traku, a zatim će se zaustaviti. Pojaviće se poruka:

```
SAVING FRED
READY
```

To je to! Vaš program je sad na traci pod nazivom FRED.

TESTIRANJE KOMANDE LOAD

Sada premotajte traku na početak i otkucajte NEW da biste iz memorije izbrisali program. Ovo je potrebno samo zbog toga da biste se uverili da je program stvarno na traci.

Otkucajte:

```
LOAD "FRED"
```

i pritisnite RETURN. Pojaviće se poruka:

```
PRESS PLAY ON TAPE
```

Pritisnite dugme PLAY na kasetofonu. Traka će početi da se kreće, a nakon nekog vremena (budite strpljivi) zaustaviće se uz poruku:

```
OK
SEARCHING FOR FRED
FOUND FRED
```

Nakon pauze (koju možete skratiti pritiskom na taster COMMODORE) traka će nastaviti da se kreće, i ukoliko je sve u redu, zaustaviće se uz poruku:

```
LOADING
READY
```

Sada možete da listate program naredbom LIST da biste utvrdili da li je zaista ponovo u memoriji i možete ga na uobičajeni način izvršiti naredbom RUN.

NAZIVI PROGRAMA

Zašto SAVE "FRED" i LOAD "FRED"? Šta jadni stari Fred ima sa svim tim?

Programu na traci se može dati neki naziv da bi se razlikovao od ostalih programa na traci. Naziv može da bude bilo šta do dužine od 16 znakova. Tako su ispravni nazivi programa:

```
FRED
ANDJA
PROG1
++ #### A <> DD7
```

Komanda SAVE bez naziva samo će upisati program na traku. SAVE iza kojeg je naziv u navodnicima upisaće i "blok zaglavlja" koji sadrži naziv i može se kasnije pronaći. Komanda LOAD bez naziva će jednostavno učitati prvi program koji nađe na traci. Komanda LOAD iza koje stoji naziv u navodnicima traže dok ne nađe odgovarajući blok zaglavlja, a zatim će učitati program.

Nazivi se koriste zato što možete da snimite više programa sa različitim nazivima na istu traku, a zatim da učitate onaj koji ste izabrali. Da biste videli kako to funkcioniše, naredbom SAVE snimite nekoliko ispitnih programa jedan za drugim (pazite da ne snimite jedan preko drugog) pod različitim nazivima:

```
SAVE "TEST1"
SAVE "TEST2"
SAVE "TEST3"
```

(Programi mogu da budu isti ako želite. Međutim, lakše ćete se uveriti da sve funkcioniše, ako po nečemu možete da ih razlikujete).

Premotajte traku i otkucajte:

```
LOAD "TEST3"
```

Pojaviće se uobičajena poruka:

```
PRESS PLAY ON TAPE
```

Pritisnite dugme PLAY. Sačekajte. Traka će početi da se kreće, ekran će se izbrisati, zatim će se traka zaustaviti, a na ekranu će se pojaviti poruka:

```
SEARCHING FOR TEST3
FOUND TEST1
```

Posle duže pauze (koja se može skratiti pritiskom na taster COMMODORE) traka kreće, ekran se briše . . . ovog puta poruka je:

SEARCHING FOR TEST3
FOUND TEST1
FOUND TEST2

Ako se pronađe TEST3, biće učitan na uobičajeni način.

KOMANDA VERIFY

Ova naredba funkcioniše kao komanda LOAD izuzev što ne učitava program. Ona samo proverava da li je on jednak onome što se već nalazi u memoriji. Poruke i redosled operacija su u suštini isti. Glavna upotreba komande VERIFY je provera da li je program dobro upisan na traku. Program treba upisati komandom SAVE, zatim treba premotati traku i proveriti upis komandom VERIFY. Kasetofon ima ugrađen brojač. Možete ga koristiti da utvrdite do kog mesta na traci ste došli. Veoma je preporučljiva ideja da se vodi evidencija o programima i odgovarajućem stanju brojača na kartici stavljenoj u kutiju kasete.

Međutim, šta ako ste zaboravili da to uradite, a želite da utvrdite šta se nalazi na traci, a da ne pokvarite program koji je trenutno u računaru? Nije baš preporučljivo da koristite naredbu LOAD. Možda ćete uspeti da učitate nešto i tako izgubite ono iz memorije.

Trik je u sledećem:

VERIFY "NEMA"

gde je NEMA bilo koji naziv koji ne koristite za naziv programa. Računar će tražiti NEMA i na ekranu listati sve programe sa trake.

SEARCHING FOR NEMA
FOUND FRED
FOUND ANDJA
FOUND SIMULATOR
FOUND JAMB

itd.

Možete da zapišete stanje brojača u svakoj fazi.

KOMERCIJALNI SOFTVER

Možete da kupite i programe već snimljene na traci. Postupak učitavanja za njih je isti kao i za programe koje ste sami upisali na traku. Premotajte traku na početak i otkucajte LOAD. (Ako program ima naziv, trebalo bi da bude naveden u dokumentaciji koja ide uz njega. U dokumentaciji treba da budu navedena i uputstva za eventualno nestandardno učitavanje. Međutim, obično LOAD bi trebalo uvek da funkcioniše.) Budite strpljivi, pošto proizvođači često ostavljaju prazan poduži deo trake pre programa.

ZAŠTITA OD BRISANJA

Ljubitelji hi-fija znaju sve o tome, ali ljudi koji su kupili kasetofon samo za računar možda i ne znaju. Kasetu možete zaštititi od brisanja tako što ćete odlomiti plastični jezičak koji se nalazi sa suprotne strane trake. Kad neku stranu okrenete gore, levi jezičak štiti tu stranu. Odlomite ga odvijačem ili nečim sličnim.

Da biste snimili na zaštićenu traku, jednostavno zalepite komad lepljive trake preko otvora u kojem je bio jezičak. Komercijalne trake obično već imaju otvore umesto jezička.

OLANČAVANJE PROGRAMA

Kad se program učita komandom LOAD, da biste ga inicirali, treba da otkucate RUN. Međutim, komandu LOAD možete da koristite i *iz* programa. U tom slučaju učitani program se izvršava automatski.

Tako na traku možete da upištete skup programa kao, na primer:

```
PROG1:      10
            20
            . . .
          5000 LOAD "PROG2"
PROG2:      10
            20
            . . .
          7000 LOAD "PROG3"
PROG3:      10
            20
            . . .
          8962 LOAD "PROG4"
```

i tako dalje. Prvi program treba učitati ručno i otkucati RUN. Ostalo će se izvršavati automatski. Npr. možda se igrate video-igrom golf. PROG1 obrađuje prvu rupu, PROG2 drugu itd. Kad dođete do reda sa naredbom LOAD, znači da ste lopticu uterali u rupu. Ovim se šire granice mogućnosti za maštovitu grafiku koja zauzima mnogo prostora u memoriji i za čitav niz programa koji bi se inače odjednom našli u memoriji.

Za druge načine korišćenja trake vidi poglavlje 34.

Olovka i papir se još uvek koriste.

21 Pronalaženje i otklanjanje grešaka III

TABELE TESTIRANJA "NA SUVO"

Greške u izvršavanju ne pronalaze se uvek tako lako kao u poglavlju 15. Ako ne možete da vidite šta je pogrešno, potrebno je da pažljivo i sistematski pregledate program. To može da bude skoro laboratorijski postupak, međutim, ako ga se u potpunosti pridržavate, skoro garantovano se isplati.

Objasniću tehniku testiranja "na suvo" na jednom primeru. Zamislimo da ste u časopisu pronašli sledeći program:

```
10 INPUT "UPISITE SLEDECI BROJ"; N
20 IF N>0 THEN S=S+N : C=C+I : GOTO 10
30 PRINT "PROSEK JE"; S/C
40 INPUT "IMA LI JOS PODATAKA (DA/NE)"; Q$
50 IF Q$="DA" THEN 10
```

a u propratnom tekstu je objašnjenje da će program izračunati prosek članova skupa pozitivnih brojeva koji se završava nulom. Drugim rečima, trebalo bi da program radi otprilike ovako:

```
UPISITE SLEDECI BROJ? 3
UPISITE SLEDECI BROJ? 4
UPISITE SLEDECI BROJ? 5
UPISITE SLEDECI BROJ? 0
```

a nakon toga trebalo bi da dobijete poruku:

```
PROSEK JE 4
```

(nula ne spada u podatke, nego je znak da više nema podataka, vidi poglavlje 11). Zatim bi program trebalo da pita korisnika da li želi da obradi drugi skup podataka:

```
IMA LI JOS PODATAKA (DA/NE)?
```

Ako kao odgovor otkucate "DA", program treba da počne od početka i da traži novi skup brojeva.

Ako otkucate program i pustite da se izvršava sa brojevima 3, 4, 5 i 0, utvrdićete da ne radi. On će prihvatiti brojeve, kako ste i očekivali, ali kad upišete nulu, program će se prekinuti uz poruku:

?DIVISION BY ZERO ERROR IN 30

Da biste utvrdili šta je "glupa" mašina uradila, pravite se sada da ste *vi* mašina i ropski izvršavajte programske naredbe kao što to radi mašina. Treba da prikažete kako se razne vrednosti u programu menjaju i to u tabeli kao što je ova:

Broj reda	S	N	C	I	Grananje

U ovom programu postoje samo promenljive S, N, C i I. Kako se koji programski red izvršava upisuju se nove vrednosti S, N, C i I (ako su se u tom redu promenile), a ako je u pitanju naredba IF . . . THEN tada se u koloni "Grananje" znakom "+" obeležava ako je došlo do grananja, a ako nije, koristi se znak "X". Pre nego što počne izvršenje programskog reda, postoji još jedna stvar koju će "šezdesetčetvorka" učiniti čim otkucate RUN, a to je da svim promenljivima dodeli vrednost nula tako da tabela izgleda ovako:

Broj reda	S	N	C	I	Grananje
	0	0	0	0	

Pretpostavimo da program testiramo brojevima 3, 4, 5 i 0. Kad se red 10 izvrši prvi put, promenljivoj N će se dodeliti vrednost 3 tako da ćemo imati:

Broj reda	S	N	C	I	Grananje
	0	0	0	0	
10		3			

U redu 20 N je veće od nule tako da se 3 dodaje na S, vrednost I (nula) dodaje se na C i vrši se skok na red 10:

Broj reda	S	N	C	I	Grananje
	0	0	0	0	
10		3			
20	3		0		+

Ako nastavimo postupak, dobićemo:

Broj reda	S	N	C	I	Grananje
	0	0	0	0	
10		3			
20	3		0		+
10		4			
20	7		0		+
10		5			
20	12		0		+
10		0			
20					X
30					← pokušaj deljenja 12/0

Sada možemo da vidimo kako je došlo do greške. Tabela nam služi da utvrdimo kako se koja promenljiva menja. Pogledajmo prvo kolonu N. Vidimo da se javljaju vrednosti 3, 4, 5 i 0 što smo i očekivali. Interesantnije su vrednosti S. Možemo da vidimo postepeno formiranje zbira svih vrednosti N. Prvo je tu vrednost 3, zatim 7 (=3+4) pa 12 (=7+5). Ali šta je sa C? Tu se ne dešava apsolutno ništa. Pošto smo upisali tri broja, prosek bi trebalo da bude 12/3=4 tako da bi C trebalo da bude 3. U stvari, razlog zašto C ostaje nula je u tome što mu se dodaje I, a I je stalno nula. Pretpostavimo da je greškom I štampano umesto broja 1. Tada bi tabela izgledala ovako:

Broj reda	S	N	C	Grananje
	0	0	0	
10		3		
20	3		1	+
10		4		
20	7		2	+
10		5		
20	12		3	+
10		0		
20				X
30				Ispis 12/3=4

Sad stvari već izgledaju bolje. Izmenite u redu 20 slovo "I" i broj "1" i otkucajte RUN. Upišite 3, 4, 5 i 0, pa ćete dobiti rezultat 4. Do sada je dobro. Sada upišite "DA" kad vas program upita da li ima još podataka i probajte sa 10, 20, 30 i 0. Program će ispisati odgovor 12 što nije tačno, jer je prosek za brojeve 10, 20 i 30 jednak 20.

LOGIČKE GREŠKE

To je primer jednog novog tipa greške — logičke. Program nešto radi i izvršava zadatak, a da "šezdesetčetvorka" ne daje nikakvu poruku da nešto nije u redu, ali ono što je program uradio *nije* izračunavanje prosečne vrednosti brojeva 10, 20 i 30.

Pa da nastavimo sa našim testiranjem "na suvo" da bismo utvrdili šta je pogrešno. Ne treba ponavljati ono što smo već uradili, pa ćemo to samo zapisati da

je, kad dođemo do reda 40, S=12, N=0 i C=3. Potrebna nam je i znakovna promenljiva Q$:

Broj reda	S	N	C	Q$	Grananje
40	12	0	3	DA	
50					+
10		10			
20	22		4		

Mislim da nije potrebno da idemo dalje da bismo utvrdili u čemu je problem. Pošto S počinje od 12, ono dobija vrednost 22 kad bi trebalo da bude samo 10, a C postaje 4 kad bi trebalo da bude 1. Drugim rečima, "šezdesetčetvorka" nije zaboravila prvi skup vrednosti kad obrađuje drugi tako da će izračunati:

(3+4+5+10+20+30)/6=12

Sada je lako. Dodajmo red 5:

```
5       S=0 : C=0
```

i izmenimo red 50 u:

```
50 IF Q$="DA" THEN 5
```

Koliko su verovatni tipovi grešaka koje smo videli? Pa, štamparske greške zamene I i 1 i 2 i Z su prilično uobičajene (ne u ovoj knjizi, nadam se), a čak i kad nema greške u štampi, prilično je moguće da loše prepišete, a da to i ne primetite kad kucate program. Čak i kad sami pišete program, bićete iznenađeni koliko se često dešava da zaboravite naziv promenljive i napišete drugi, ili čak da koristite istu promenljivu za dve različite stvari. Zbog toga treba da budete pažljivi i zabeležite naziv promenljive i njenu funkciju čim je prvi put upotrebite. Ali ljudska priroda je ljudska priroda . . .

Druga greška pokazuje dve važne stvari. Prvo, iako BASIC pokušava da pomogne postavljajući početne vrednosti promenljivih na nulu, a da vas i ne pita, to ne mora uvek da bude pogodno. Dobro je uobičajiti da se svim promenljivima na početku programa dodele početne vrednosti, čak i ako ne morate, pošto tako izbegavate ovu vrstu greške. Neću da kažem da ćete činiti tako trivijalne greške kao što je ova, ali ako ste napisali program koji obrađuje jedan skup podataka, pa ga kasnije menjate da obrađuje više skupova, videćete kako se može zaboraviti inicijalizovanje.

Drugo, ovo ilustruje kako treba biti pažljiv prilikom testiranja programa. Ne treba zaključiti da će program raditi dobro sa bilo kojim skupom podataka samo zato što dobro radi sa jednim skupom. O tome će biti više reči u poglavljima 26 i 29.

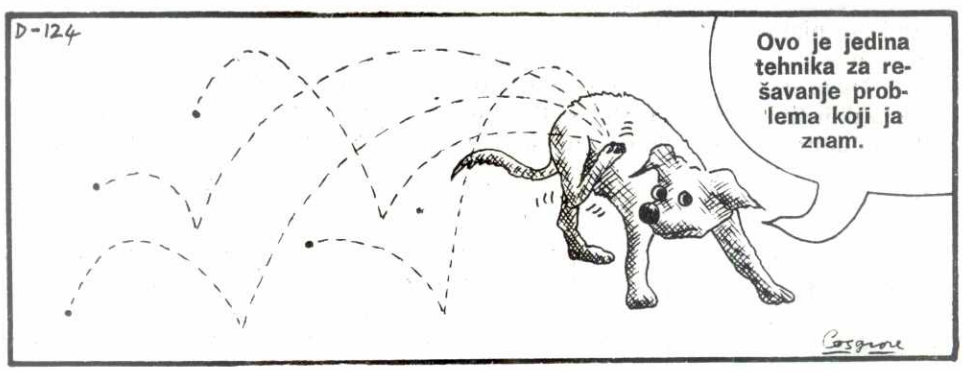
Vežba 1

Evo primera na kojem možete da isprobate svoju veštinu. Sledeći program trebalo bi da prihvata niz pozitivnih brojeva (od 0 do 1000) i da, na kraju, ispiše najmanju i najveću vrednost koja je upisana. Kao kraj upisa uzima se negativni broj. Npr. ako su podaci 8, 4, 3, 9, —1, program bi trebalo da na ekranu ispiše:

9 3

Odaću vam tajnu — u programu su dve greške.

```
10 MAX=0 : MIN=0
20 INPUT N
30 IF N<MIN THEN MIN=N
40 IF N>MAX THEN MAX=N
50 IF N<0 THEN PRINT MAX, MIN : END
60 GOTO 20
```



ODGOVORI

Vežba 1

Za dati primer tablica "suvog testiranja" izgleda ovako:

Broj reda	MAX	MIN	N	Grananje	
	0	0	0		
10	0	0			
20			8		
30					
40	8				
50					
60				+	
20			4		
30					
40					
50					
60				+	
20			3		
30					
40					
50					
60				+	
20			9		
30					
40	9				
50					
60				+	
20			−1		
30		−1			
40					
50					ispisivanje 9, −1

Znači, *najveća* vrednost je ispravno ispisana, ali najmanja nije. Ako pogledamo kolonu MAX, videćemo kako je zamišljeno da program radi. Svaki put kad je vrednost N veća od vrednosti koju ima promenljiva MAX, vrednost N se dodeljuje promenljivoj MAX. Promenljiva MIN bi trebalo, naravno, da se ponaša na isti način, tako da bi trebalo da ima vrednosti 8, 4 a zatim 3. Međutim, vrednost promenljive MIN se ne menja sve dok se ne upiše −1, što ne bi trebalo da utiče na rezultat pošto negativan broj samo označava kraj skupa podataka.

Razlog za to što se vrednost promenljive MIN ne menja do kraja programa je u tome što ona već ima najmanju moguću vrednost (nula). Početna vrednost promenljive MIN treba da bude dovoljno velika, tako da je zameni prva upisana vrednost koja će *morati* da bude manja. Pošto je poznato da je najveća moguća vrednost 1000, odgovaraće *svaka* vrednost veća od 1000. Tako red 10 sada postaje:

```
10 MAX=0 : MIN=1001
```

Sada treba sprečiti da ograničavajuća vrednost −1 bude dodeljena promenljivoj MIN. To je lako. Samo treba da pitamo da li je kraj podataka pre nego što promenimo vrednosti promenljivih MAX i MIN. Znači, red 50 treba brisati i napisati ga kao red 25.

22 Slučajni brojevi

U nekim programima želite da se računar ponaša na nepredvidljiv način. "Šezdesetčetvorka" ima naredbu kojom se dobijaju "slučajni" brojevi, pa možete da ih koristite za nešto što ne želite unapred da znate. To je posebno korisno u igrama. Šta mislite, koliko ima igara u kojima je potrebno baciti kocku ili izvući neku kartu?

Naredba za slučajne brojeve je:

```
RND
```

Iza naredbe dolazi broj u zagradi.

Da biste videli šta ova naredba radi, otkucajte ovaj program, a zatim naredbu RUN:

```
10 INPUT N
20 FOR T=1 TO 10
30 PRINT RND (N)
40 NEXT T
50 GOTO 10
```

Prvo za N upišite 1. Dobićete deset brojeva između 0 i 1 bez neke vidljive zakonitosti. Pokušajte opet sa 1. Dobićete novih deset brojeva opet bez zakonitosti.

Pokušajte sada sa —2. Dobićete isti broj ponovljen deset puta, ali različit od bilo kog broja koji ste dobili ranije.

U stvari, opšti učinak naredbe RND (N) je ovaj:

1. Ako je N pozitivan broj, RND (N) daje slučajan broj između 0 i 1 (vrednost 0 može da se pojavi, a vrednost 1 ne).

Međutim, ovako dobijeni nizovi slučajnih brojeva su ponovljivi.

2. Ako je N nula, RND (0) daje slučajni broj određen ukupno proteklim vremenom od trenutka uključenja računara. Na neki način to je "najslučajniji" mogući broj.

3. Ako je N negativan broj, RND (N) daje *određeni* "slučajni" broj koji zavisi od N.

Preporučujem da učinak (3) ignorišete, pošto se on koristi uglavnom za pronalaženje i otklanjanje grešaka vezanih za postupak generisanja "slučajnog" broja. (Oni nisu stvarno slučajni u punom smislu reči, ali je postupak generisanja pomoću polaznog broja (engl. seed = seme) takav da za većinu praktičnih primena daje rezultate bez zakonitosti). U stvari, najverovatnije ćete uvek koristiti naredbu:

RND (0)

pa ću od sada i ja koristiti samo nju.

KOCKE, KARTE I KOCKARSKI REKVIZITI

Ostanite na vezi, neko kuca na vrata ..... Ne, druže milicioneru, nemam dozvolu za kockanje. Pa, pošto insistirate, promeniću naslov u:

SLUČAJNI DOGAĐAJI

Hmmm! E pa, hajde na posao. Kad bacite kocku, dobićete broj između 1 i 6 i to na sasvim slučajan način. "Šezdesetčetvorkina" "kocka" daje decimalne brojeve između 0 i 1. Znači, potrebno je malo matematičkog žongliranja.

Evo tipičnih slučajnih brojeva "šezdesetčetvorke" (u prvoj koloni) sa onim što dobijete kad ih množite sa 6.

RND (0)	6* RND (0)
0,131137465	0,78682479
0,80924873	4,85549238
0,846447204	5,07868323
0,591965711	3,55179427
0,26800113	1,6080678

Vidimo da množenje sa 6 proširuje opseg brojeva sa 0—1 na 0—6. To je prvi korak. Sledeći je da se oslobodimo decimalnih mesta. Naredba:

INT

daje *celobrojni deo* broja odnosno najveći ceo broj koji nije veći od datog broja. Za pozitivne brojeve on je jednak broju *ispred* decimalnog zareza dok je za negativne za 1 manji. Na primer:

INT (4.85549238)=4
INT (—3.141592)= —4

Primetimo da se u primeru za decimalni broj koristi tačka, a ne zarez.

To je zbog toga što svi viši programski jezici za osnovu imaju engleski jezik i što se na govornom području engleskog jezika za odvajanje decimalnih brojeva koristi tačka, a ne zarez (prim. pev.).

Ako pre decimalne tačke nema ništa, naredba INT daje 0 (za pozitivne brojeve). Ako primenimo naredbu INT na desnu kolonu tabele, dobićemo sledeće brojeve:

0 4 5 3 1

što skoro odgovara bacanju kocke. Jedina razlika je u tome što su ovi brojevi od 0 do 5 umesto od 1 do 6. Dodajmo zato 1:

1 5 6 4 2

pa ćemo dobiti ono što nam treba. Ako zapišemo sve u jednom redu, dobićemo da:

INT (6 * RND (0)) + 1

daje slučajan ceo broj u opsegu 1—6 kao i kocka.

Vežba 1

Koje naredbe treba da koristite da biste dobili slučajan ceo broj koji odgovara:

1. Paketu od 52 karte
2. Jednoj boji sa 13 karata
3. Jednoj domini izvučenoj iz kompleta od 28 domina.
4. Rođendanu u godini koja nije prestupna
5. Broju između 10 i 99 (uključujući oba)?

TABLICA UČESTALOSTI POJAVLJIVANJA

Evo programa sa relativno ozbiljnom namenom. On baca kocku 150 puta i pamti koliko puta se koji broj pojavio.

```
10 PRINT CHR$(147)
20 POKE 53281,0
30 POKE 53280,0
40 POKE 646,1
50 FOR R=1 TO 6
60 RED=3*R: KOL=0: CDE=48+R: CR=1: GOSUB 10000
70 NEXT R
80 FOR T=1 TO 150
90 D=INT(6*RND(0)+1)
100 C(D)=C(D)+1
110 IF C(D)>38 THEN STOP
120 RED=3*D: KOL=C(D)+2: CDE=160: CR=D+1: GOSUB 10000
130 NEXT T
140 GOTO 140
```

```
10000 REM PRINT AT
10010 POKE 1024+40*RED+KOL, CDE
10015 POKE 55296+40*RED+KOL, CR
10020 RETURN
```

Redovi 10—40 brišu ekran i biraju boje. Broj pojavljivanja brojeva 1, 2, 3, 4, 5, i 6 pamti se u šest promenljivih C (1), C (2), C (3), C (4), C (5) i C (6). Malo sam varao pošto je ova lista promenljivih ono što zovemo *matricom*, a o matricama će biti reči tek u poglavlju 25. Nije važno, mislim da je ideja dovoljno jasna.

Redovi 50—70 ispisuju brojeve 1—6 u koloni. 48+R samo generiše znakove 1—6 čiji su ekranski kodovi (vidi Dodatak E u *Priručniku*) 49 do 54. Primetimo upotrebu potprograma u redu 1000 za ispisivanje znaka u određenom redu i koloni.

Redovi 80 i 130 bacaju kocku 150 puta.

Red 100 povećava odgovarajući brojač broja pojavljivanja nekog od brojeva.

Red 120 ispisuje na ekranu obojeni blok u redu D. Red 110 služi samo za to da spreči prelazak u novi red, ako se u ispisivanju stigne do desnog ruba ekrana.

Videćete kako obojene trake rastu na ekranu. Iako za neka izvršenja programa neki broj pobedi, na duže staze se pobede brojeva izjednačavaju. Kocka je *fer*.

ODGOVORI

Vežba 1

1. INT (52 * RND (0)) + 1
2. INT (13 * RND (0)) + 1
3. INT (28 * RND (0)) + 1
4. INT (365 * RND (0)) + 1
5. INT (90 * RND (0)) + 10 (pošto 90 * RND (0) daje brojeve od 0 do 89,

dodavanje 10 daje opseg 10—99. Broj 90 se dobija izračunavanjem dužine opsega, 99—10=89 plus 1 za kraj, a zatim se 10 koristi za pomeranje opsega).

Možda ste razmišljali o tome da problem sa dominama (3) rešite upotrebom INT (7 * RND (0)) da biste dobili tačke 0 do 6 pri čemu je 0 prazno polje ("golo"). Uradite to dva puta, ali zadržite samo one parove M, N za koje je M<=N pošto ćete inače domine sa različitim brojevima kao |3|6| dobiti dva puta, jednom kao |3|6|, a jednom kao |6|3| i dvostruke domine |5|5| samo jedanput, a to menja verovatnoću.

„Šezdesetčetvorka" je opremljena širokim spektrom grafičkih znakova koji se mogu koristiti za crtanje slika na ekranu.

23 PET-grafika

Teško da ste propustili da primetite da "šezdesetčetvorka" može da ispisuje i grafičke znakove, a ne samo slova i brojeve. Njima se može pristupiti sa tastature, a prikazani su na *prednjem delu* odgovarajućeg tastera. Znak sa desne strane dobija se pomoću tastera SHIFT, a znak sa leve strane pomoću tastera COMMODORE. Ovi znakovi, iako loše izabrani, prilično su raznoliki. Oni su preuzeti sa Commodore-ovog računara PET (PET, engl. = maza, kućni ljubimac), pa odatle i naziv ovog poglavlja. Što se računara tiče, grafički znakovi su znakovi kao i svaki drugi i mogu se prikazati naredbom PRINT ili naredbom POKE sa odgovarajućim ASCII-kodom.

Za autora knjige grafički znakovi predstavljaju problem pošto ih nema u standardnim štamparskim znakovima. U svakom slučaju, teško ih je razlikovati (kao što možete da vidite u programskim listama u časopisima). Zbog toga nam je potrebna neka konvencija da bismo mogli da ih razmatramo.

Razmotrimo taster A. Sa leve strane je grafički znak pravog ugla, a sa desne znak pik. U tekstu ću se na njih pozivati na sledeći način:

gAc (grafički A + taster COMMODORE)
gAs (grafički A + taster SHIFT)

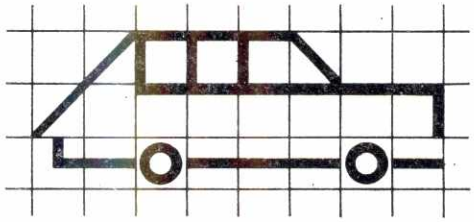
Slično važi i za ostale tastere.

PROJEKTOVANJE GRAFIKE NA MREŽI

Od grafičkih znakova možete na mreži linija da napravite crtež. Npr. sl. 23.1 prikazuje pokušaj crtanja automobila.

Koristili smo znakove:

▽	gNs	gOs	gOs	gOs	gMs	▽	▽
gNs	▽	gYc	gYc	gYc	gYc	gYc	gPs
gZc	gCs	gWs	gCs	gCs	gCs	gWs	gCs



Sl. 23.1 — Crtež automobila pomoću PET-grafike

(▽ ima uobičajeno značenje). Znači, na ekranu bismo mogli da prikažemo automobil programom kao:

```
10 PRINT CHR$ (147)
20 PRINT: PRINT: PRINT
30 PRINT TAB (15); "▽ gNs gOs gOs gOs gMs ▽ ▽"
40 PRINT TAB (15); "gNs ▽ gYc gYc gYc gYc gPs"
50 PRINT TAB (15); "gZc gCs gWs gCs gCs gCs gWs gCs"
60 GOTO 60
```

Vežba 1

Nacrtajte voz sa jednim vagonom. Prikažite ga na ekranu.

POMERANJE GRAFIKE

Različitim vrednostima za TAB može se postići da se grafika pomera na ekranu. Nešto kao ovo:

```
10 PRINT CHR$ (147)
20 FOR C=0 TO 30
30 PRINT CHR$ (19)
```

```
40 PRINT: PRINT: PRINT
50 PRINT TAB (C); "▽   gNs  gOs  gOs  gOs  gMs  ▽   ▽"
60 PRINT TAB (C);  "gNs  ▽   gYc  gYc  gYc  gYc  gYc  gPs"
70 PRINT TAB (C);  "gZc  gCs  gWs  gCs  gCs  gCs  gWs  gCs"
80 NEXT C
```

Probajte. Nešto nije u redu. Ostaju tragovi zadnjeg dela automobila.

Problem je u tome što smo zaboravili da izbrišemo prethodni automobil kad smo crtali novi.

Mogli biste izmeniti red 30 u:

```
30 PRINT CHR$ (147)
```

ali vam ne savetujem. Ekran suviše trepće.

Bolja tehnika je da se automobil opremi "nevidljivim zadnjim krajem" koji briše zadnji kraj prethodnog automobila. Sve što treba da uradite je da u redovima 50 do 70 dodate još jedan razmak na početku svakog niza grafičkih znakova:

```
50 PRINT TAB (C);  "▽▽ gNs  gOs  gOs  gOs  gMs  ▽ ▽"
60 PRINT TAB (C);  "▽   gNs   ▽   gYc  gYc  gYc  gYc  gYc
                    gPs"
70 PRINT TAB (C);  "▽   gZc  gCs  gWs  gCs  gCs  gCs  gWs
                    gCs"
```

Sada je sve u redu.

Vežba 2

Promenite program tako da se automobil kreće od kolone 30 do kolone 0 (savet: biće vam potreban "nevidljivi prednji deo").

KORIŠĆENJE ZNAKOVNIH PROMENLJIVIH

Ako imate potrebu da češće ispisujete niz grafičkih znakova, mnogo je bolje da ih definišemo kao znakovne promenljive. Npr. evo programa koji simulira trku dva automobila:

```
10 PRINT CHR$ (147)
20 C1 = 0 : C2 = 0 : C = 0
30 A$ = "▽ ▽   gNs  gOs  gOs  gOs  gMs"
40 B$ = " ▽    gNs   ▽   gYc  gYc  gYc  gYc  gYc  pGs"
50 C$ = " ▽    gZc  gCs  gWs  gCs  gCs  gCs  gWs  gCs"
60 R = 8: GOSUB 1000
70 R = 16: GOSUB 1000
80 N = INT (1+2 * RND (0))
90 IF N=1 THEN C1=C1+1 : R=8 : C=C1
```

```
100 IF N=2 THEN C2=C2+1 : R=16 : C=C2
110 GOSUB 1000
120 IF C1=30 THEN PRINT CHR$ (19); "POBEDA   KOLA 1": STOP
130 IF C2=30 THEN PRINT CHR$ (19); "POBEDA   KOLA 2": STOP
140 GOTO 80
1000 PRINT CHR$ (19);
1010 FOR T=1 TO R: PRINT: NEXT T
1020 PRINT TAB (C); A$
1030 PRINT TAB (C); B$
1040 PRINT TAB (C); C$
1050 RETURN
```

U redu 1000 je potprogram koji crta automobil u redu R i koloni C. Oba automobila kreću iz kolone 0 (C1 i C2 su 0), a nalaze se u redovima 8 i 16. Red 80 na slučajan način odlučuje koji automobil će se pomeriti, redovi 90 do 110 ga pomeraju, a redovi 120 i 130 ispituju koji automobil je pobedio.

Vežba 3

Izmenite program tako da prvi automobil bude crven, a drugi zelen (uputstvo: PRINT CHR$ (28) daje crvenu boju znakova, a PRINT CHR$ (30) zelenu).

ODGOVORI

Vežba 1

```
10 PRINT CHR$ (147)
20 A$ = "gAc    gCs  gCs  gCs  gSc   ▽   gOs  gYc
        g—s▽▽▽gNc  g—s"
30 B$ = "g—s    g+c   ▽   g+c  g—s   ▽   gPs  gEc  g—s
        g—s    g—s  gZc  gIs"
40 C$ = "g—s    G  W  R   gQc  gRc  gOs  A M T R A K  g—s"
50 D$ = "gZc    gWs  gCs  gWs  gXc   ▽   gCs  gQs  gCs
        gCs    gQs  gCs  gQs  gXc"
60 FOR T=1 TO 7: PRINT: NEXT T
70 PRINT TAB (8); A$
80 PRINT TAB (8); B$
90 PRINT TAB (8); C$
100 PRINT TAB (8); D$
110 GOTO 110
```

Primetimo način na koji su tekstualni znakovi (G, W, R, itd.) pomešani sa grafičkim znakovima u redu 40.

Vežba 2

```
10 PRINT CHR$ (147)
20 FOR C=30 TO 0 STEP -1
30 PRINT CHR$ (19)
40 PRINT: PRINT: PRINT
50 PRINT TAB (C); "▽  gNs gOs gOs gOs gMs  ▽   ▽ ▽"
60 PRINT TAB (C); "gNs ▽  gYc gYc gYc gYc gYc gPs  ▽"
70 PRINT TAB (C); "gZc gCs gWs gCs gCs gCs gWs gCs ▽"
80 NEXT C
```

U redu 50 na desnom kraju dovoljan je samo jedan razmak. Zašto?

Vežba 3

Dodajte red:

```
1015 PRINT CHR$ (26+2*N)
```

i izmenite sledeće redove

```
60 R=8 : N=1 : GOSUB 1000
70 R=16 : N=2 : GOSUB 1000
```

Računaru se može narediti da očitava tastaturu i da na različite načine reaguje na ono što je očitao. Jedna od primena je upravljanje pokretnom grafikom.

24 Upravljanje sa tastature

Problem sa naredbom INPUT je u tome što ona zadržava sve što se otkuca dok se ulaz ne završi. Zamislite video-igru u kojoj bi se cela akcija zamrznula dok vi ne pritisnete odgovarajuće tastere... Nije baš veselo, a? Naredba:

GET

upućuje računar na to da očitava tastaturu i utvrdi koji taster je pritisnut. Rezultat se memoriše u nekoj znakovnoj promenljivoj koju sami izaberete. Ako želite da promenljiva bude A$, naredba će biti:

GET A$

Evo jednostavnog ispitnog programa:

```
10 PRINT CHR$ (147)
20 GET A$
30 PRINT "PRITISNULI STE TASTER"; A$
40 GOTO 20
```

Videćete gomilu poruka kako se brzo kreće po ekranu. Ako pritisnete neki taster, recimo "Z", videćete i to Z kako brzo prolazi ekranom.

Razlog za sve to je u tome što, kad ne pritisnete nijedan taster, računar promenljivoj A$ dodeljuje vrednost praznog niza " " i nastavlja dalje. Da biste to izbegli, dodajte:

```
25 IF A$="" THEN 20.
```

Program se sada vrti u maloj petlji čekajući da pritisnete neki taster. Eksperimentišite! Probajte sa tasterom RETURN (on će izazvati preskakanje reda pošto se RETURN interpretira kao prelaz u novi red). Probajte i taster CTRL, npr. CTRL+5.

POKRETNA GRAFIKA

Hajde da malo proširimo ideju i da koristimo tastere za upravljanje grafikom. Uzmimo naš automobil iz poglavlja 23. Recimo da želimo da ga pomeramo jedno mesto napred pritiskom na taster "F" odnosno jedno mesto nazad pritiskom na taster "B". Evo šta ćemo uraditi:

```
10 PRINT CHR$ (147)
20 C=15
30 A$ ="▽    ▽  gNs gOs gOs gOs gMs ▽  ▽"
40 B$ =  "▽    gNs  ▽  gYc gYc gYc gYc gYc gPs  ▽"
50 C$ =  "▽    gZc gCs gWs gCs gCs gCs gWs gCs  ▽"
60 R=12 : GOSUB 1000
70 GET G$
80 IF G$="F" THEN C=C+1
90 IF G$="B" THEN C=C-1
100 IF C<0 THEN C=0
110 IF C>29 THEN C=29
120 GOTO 60
1000 PRINT CHR$ (19);
1010 FOR T=1 TO R : PRINT : NEXT T
1020 PRINT TAB (C); A$
1030 PRINT TAB (C); B$
1040 PRINT TAB (C); C$
1050 RETURN
```

Broj kolone u kojoj se crta automobil nalazi se u promenljivoj C. Redovi 80 i 90 koriguju C prema tome koji taster je pritisnut. Redovi 100 i 110 sprečavaju da automobil izađe sa ekrana. Primetili ste da postoji nevidljivi prednji kao i nevidljivi zadnji kraj automobila (redovi 30—50). To je zato što se kreće u oba smera.

Vežba 1

Izmenite program tako da se automobil pomera gore ako pritisnete taster "U" odnosno dole, ako pritisnete taster "D". Ako želite, umesto automobila nacrtajte helikopter.

"ŠAŠAVI" BAFER

Postoji i jedna začkoljica vezana uz naredbu GET, a to je način na koji se očitava tastatura. U računaru postoji *bafer tastature* koji sadrži do 10 znakova. Kad god se pritisne neki taster, rezultat se memoriše u baferu (ako ima mesta). Rezultat toga je da, ako pritisnete *nekoliko* tastera (što je veoma moguće u žaru igre), svi oni se memorišu u baferu, pa ih računar kasnije *obrađuje*.

Da biste to videli, usporite pomeranje automobila naredbom:

```
115 FORT T=1 TO 1000: NEXT T
```

Sada otkucajte RUN i *brzo* nekoliko puta pritisnite taster „F". Videćete da se automobil pomera, pa se zatim pomeri još jednom, pa još jednom i tako sve dok se bafer ne isprazni. *Brzo* pritisnite FBFBFB i sačekajte. Videćete kako se automobil pomera levo-desno.

Da biste prevazišli ovu karakteristiku mašine, možete da kontrolišete dužinu bafera tastature naredbom:

```
POKE 649, 1
```

koja bafer ograničava na jedan znak (za N znakova koristite

```
POKE 649, (N)
```

Programu dodajte red:

```
5 POKE 649, 1
```

Sada će pritiskanje više tastera izazvati samo *jedno* pomeranje — ono izazvano pritiskanjem prvog tastera nakon naredbe GET.

AUTOMATSKO PONAVLJANJE

Primetićete i da, ako držite taster pritisnut, naredba GET ga više ne primećuje. Ona očita taster, a zatim ga ignoriše sve dok ga ne pustite. Znači, ne možete da brzo pomerate kola držeći taster "F" u pritisnutom položaju. Često biste želeli da postignete baš to.

Tajna je u sistemskoj promenljivoj na adresi 197 koja sadrži (čudno kodovan) trenutno pritisnut taster. Isprobajte ovaj mali program:

```
2000 PRINT PEEK (197)
2010 GOTO 2000
```

Sada otkucajte RUN 2000. Na ekranu ćete dobiti kolonu brojeva 64. Pritisnite neki taster, recimo "A" i držite ga u pritisnutom položaju. Broj se menja u 10. Probajte sa ostalim tasterima. Videćete da svaki daje drugačiji broj. Ovi brojevi dati su u Dodatku 7. Oni nisu ni ASCII — kodovi ni kodovi za ekran nego nešto sasvim drugo.

Kôd za "F" je 21, a za "R" kôd je 17. Znači, da bismo dobili automatsko ponavljanje upravljanja automobilom, uradićemo sledeće:

Izbrisaćemo red 115, a redove 70—90 promeniti u:

```
70 A=PEEK (197)
80 IF A=21 THEN C=C+1
90 IF A=17 THEN C=C—1
```

Ponovo otkucajte RUN, pritisnite taster F ili R i posmatrajte kako je odziv trenutan.

FUNKCIONALNI TASTERI

Možda ste se pitali čemu služe četiri široka odvojena tastera na desnoj strani tastature. *Priručnik* ih zove "funkcionalnim tasterima koji se mogu programirati", a posle toga o njima ne kaže ama baš ništa.

U stvari, postoji osam funkcija. Bez tastera SHIFT:

f1 f3 f5 f7

a sa tasterom SHIFT:

f2 f4 f6 f8

To su, u stvari, samo "slepi" tasteri koji se mogu očitati naredbom GET. Pravi taster "koji se može programirati" trebalo bi odmah nakon pritiskanja, da reaguje nekim standardnim postupkom koji je korisnik programirao. Ovi tasteri reaguju, međutim, samo ako u programu imate naredbu GET. Na žalost, komodorovim BASIC-om nećete moći da dobijete pravu mogućnost programiranja funkcionalnih tastera. Ipak, funkcionalni tasteri vam omogućavaju bar to da koristite tastere na standardnim mestima što može da uštedi mnoge zabune kao: "Da li za uništenje neprijateljske flote koja je napala Kraljevstvo treba pritisnuti Z ili K?".

ASCII-kodovi za funkcionalne tastere (vidi poglavlje 18) su:

f1	133
f2	137
f3	134
f4	138
f5	135
f6	139
f7	136
f8	140

Tako je jedan od načina za korišćenje, recimo, tastera f5 ovaj:

```
150 GET AS : IF A$ = CHR$ (135) THEN bilo šta ....
```

Drugi način je da se koriste navodnici (poglavlje 7):

```
150 GET A$ : IF A$ = "(pritisnite taster f5)" THEN ...
```

Npr. pretpostavimo da želite da izbrišete ekran kad pritisnete taster f5. Tada biste mogli da napišete ovako:

```
150 GET A$ : IF A$ = "(taster f5)" THEN PRINT CHR$ (147)
```

Otkucajte RUN i pritisnite f5. Šta se desilo? Ništa, izuzev, ako ste veoma brzi pošto f5 može da funkcioniše *samo* dok se izvršava naredba GET. Dodajte:

```
160 GOTO 150
```

i već je bolje.

Prikladan način korišćenja ovih tastera je da se napišu programi kao:

```
15000 GET AS : IF A$ = "(taster f5)" THEN PRINT CHR$ (147)
15010 RETURN
```

Zatim postavite naredbe GOSUB 15000 na strategijska mesta u programu tako da se taster f5 stalno ispituje. Npr. stavite takav GOSUB u unutrašnju petlju sistema sa više petlji.

Ovo je odlična tehnika utoliko što su funkcionalni tasteri veliki i lako se pronalaze, ali treba imati u vidu da taster f5 ne radi ništa što ne biste mogli postići bilo kojim drugim tasterom, recimo "Z", koristeći istu metodu.

Vežba 2

Napišite program koji ispisuje tabelu kvadrata brojeva od 1 do milion, a menja boju osnove od 0—15 (za jedan korak) svaki put kad pritisnete taster f7.

ODGOVORI

Vežba 1

Prvo automobilu dodajte gore i dole nevidljivi rub:

```
25 D$="(10 razmaka)"
55 E$="(10 razmaka)"
```

Sada ih prikažite na ekranu:

```
1015 PRINT TAB (C); D$
1045 PRINT TAB (C); E$
```

Zatim povećajte broj tastera koji se ispituju:

```
92 IF G$="U" THEN R=R-1
93 IF G$="D" THEN R=R+1
```

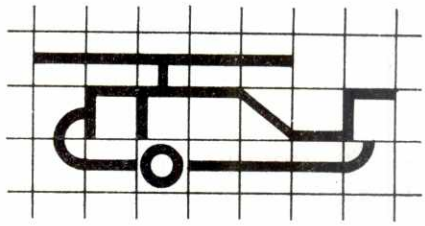
i zaštitite:

```
95 IF R<0 THEN R=0
96 IF R>18 THEN R=18
```

Sprečite da se R vrati na 12 u samoj petlji:

```
20 C=15 : R=12
60 GOSUB 1000
```

Za prikaz helikoptera dajemo ideju na slici 24.1 Ovo daje:



Sl. 24.1 — Crtež helikoptera pomoću PET-grafike

```
30 A$ = "▽   gCs  gCs  gRc  gCs  gCs  ▽ ▽"
40 B$ = "▽   gUs  gOs  gOs  gYc  gMs  gPc  gOs  ▽"
50 C$ = "▽   gJs  gCs  gWs  gCs  gCs  gCs  gKs  ▽"
```

Vežba 2

```
10 PRINT CHR$(147)
20 CL=0
30 FOR T=1 TO 1000000
40 PRINT T, T*T
50 GOSUB 5000
60 NEXT T
70 STOP


5000 GET A$
5010 IF A$<>"[TASTER F7]" THEN RETURN
5020 POKE 53281, CL
5030 CL=CL+1:IF CL=16 THEN CL=0
5040 RETURN
```

Računarski programi često zahtevaju čitave liste informacija, brojeva ili reči. Jedan način za memorisanje ovih lista su:

25 Matrice

Matrica je, u suštini, numerisana lista kao, npr. lista veša za pranje:

1. čarape
2. suknja
3. jastučnica
4. pulover

Ako neko želi treći elemenat liste, može da pogleda i vidi koji je (u našem slučaju: jastučnica). Postoje dve vrste matrica: *numeričke* i *znakovne* matrice.

Pretpostavimo da u računaru želimo da memorišemo listu dvanaest meseci u godini i to po redu. Nazivi meseca su znakovni nizovi, pa koristimo znakovnu matricu.

(Pravila za nazive matrica su ista kao i za nazive promenljivih. Računaju se samo prva dva znaka).

Računaru prvo treba reći *kolika će lista biti*. Ovo je poznato pod nazivom "dimenzionisanje matrice", a postiže se naredbom:

```
DIM
```

kao, na primer:

```
10 DIM MESEC$ (12)
```

Striktno govoreći, mogli biste izaći na kraj i sa 11 umesto 12, zbog toga što su (iz razloga koje ovde neću spominjati) matrice u "šezdesetčetvorci" numerisane od 0, pa tako matrica dimenzionisana sa 12, kao u primeru, ima elemente numerisane 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 — ukupno trinaest. Najjednostavniji način izbegavanja zabune je ignorisanje nultog elementa. Sada računaru treba reći šta sadrži 12 elemenata:

```
20 MESEC$ (1) = "JANUAR"
30 MESEC$ (2) = "FEBRUAR"
.....
130 MESEC$ (12) = "DECEMBAR"
```

Ja sam koristio tačkice, ali vi ćete morati da upišete nazive svih dvanaest meseci. Loša sreća. Sada možete da se pozovete na bilo koji elemenat matrice tako da napišete naziv matrice, a iza njega broj elementa u zagradama. Na primer:

```
MESEC$ (7)
```

daje "JULI", itd.
Da biste ispisali naziv nekog meseca, dodajte:

```
140 INPUT "REDNI BROJ MESECA"; N
170 PRINT MESEC$ (N)
```

Slično možete formirati i numeričke matrice:

```
10 DIM ANDJA (500)
```

sa elementima

ANDJA (0)
ANDJA (1) ANDJA (2) . . . ANDJA (500)

(najbolje ignorisati, ali ne uvek)

Matrice lako zauzimaju veliku količinu memorije. Ako imate velike matrice čiji su elementi *celi brojevi* možete uštedeti prostor tako da koristite *celobrojne promenljive* (detaljnije u poglavlju 30). Celobrojna promenljiva je kao obična numerička promenljiva, ali iza naziva treba da bude znak "%":

```
DIM ANDJA%(500)
```

VIŠEDIMENZIONE MATRICE

Višedimenzione matrice funkcionišu na sličan način samo što elementi formiraju neku vrstu tabele. Za, recimo, dve dimenzije:

```
DIM ANDJA (7, 4)
```

tabela ima sedam redova i četiri kolone. Pa sad, u stvari, *osam* redova i *pet* kolona pošto postoje dodatni red 0 i dodatna kolona 0 kao u slučaju matrice sa jednom dimenzijom:

ANDJA (0, 0)	ANDJA (0, 1)	ANDJA (0, 2)	ANDJA (0, 3)	ANDJA (0, 4)
ANDJA (1, 0)	ANDJA (1, 1)	ANDJA (1, 2)	ANDJA (1, 3)	ANDJA (1, 4)
ANDJA (2, 0)	ANDJA (2, 1)	ANDJA (2, 2)	ANDJA (2, 3)	ANDJA (2, 4)
ANDJA (3, 0)	ANDJA (3, 1)	ANDJA (3, 2)	ANDJA (3, 3)	ANDJA (3, 4)
ANDJA (4, 0)	ANDJA (4, 1)	ANDJA (4, 2)	ANDJA (4, 3)	ANDJA (4, 4)
ANDJA (5, 0)	ANDJA (5, 1)	ANDJA (5, 2)	ANDJA (5, 3)	ANDJA (5, 4)
ANDJA (6, 0)	ANDJA (6, 1)	ANDJA (6, 2)	ANDJA (6, 3)	ANDJA (6, 4)
ANDJA (7, 0)	ANDJA (7, 1)	ANDJA (7, 2)	ANDJA (7, 3)	ANDJA (7, 4)

Linije samo odvajaju elemente iz reda 0 i kolone 0 koje je često najbolje ignorisati (zavisi. . .). Elemenat u redu RED i koloni KOLONA je:

ANDJA (RED, KOLONA)

Možete imati i dvodimenzionu znakovnu matricu:

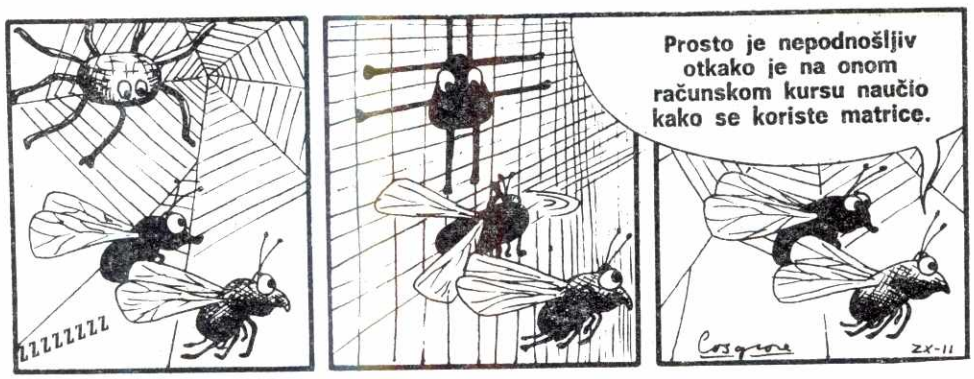
DIM ANDJA$ (7, 4)

čiji elementi su znakovni nizovi, a ne brojevi.

Korišćenje matrica je veoma korisno pošto se mnogo informacija, po prirodi, javlja u obliku tabele. Setite se samo telefonskog imenika:

Ime	Adresa	Mesto	Telefon
Pera Perić	Donjogradska 5	Novi Sad	555555
Janko Popović	P. Preradovića 18	Beograd	212212
Sima Babac	Trogirska 8	Split	28643
Ante Rukavina	Beogradska 13/II	Zagreb	764467

Ovako izdeljen telefonski imenik može da se zamisli kao znakovna matrica sa četiri kolone (telefonski brojevi se takođe smatraju znakovnim promenljivim, pošto u istoj matrici ne mogu da se jave i znakovni i brojni podaci).



CRTANJE KARATA ZA IGRANJE

Ilustrovaću korisnost matrica tako što ću napisati program koji na ekranu prikazuje karte dobijene u jednom deljenju pokera. Da bi programska lista bila kraća, koristićemo neka pojednostavljenja:

1. Dozvoljene su karte samo od asa do 10, a slike nisu (da se dozvole i slike, u principu nije problem samo ih treba nacrtati PET-grafikom. To je malo duže, pa ćemo to izostaviti, jer samo može da zamagli osnovnu ideju).

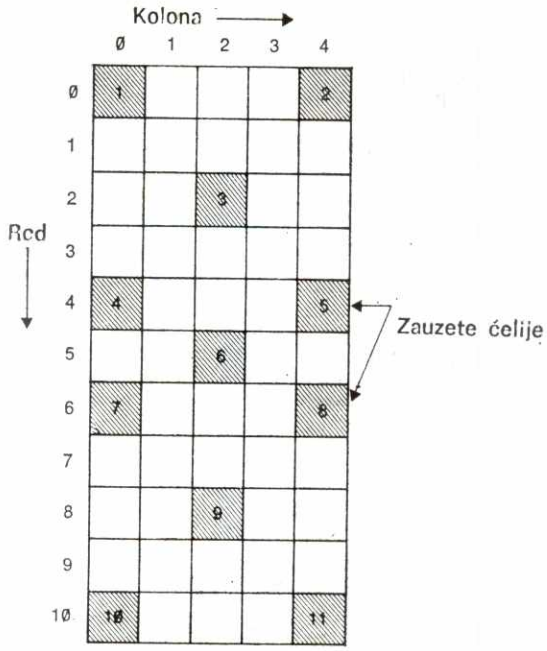
2. Svaka karta se izvlači slučajno i ne kontroliše se da li je ista karta već izvučena. Za rešavanje ovog vidi vežbu 1. U redu. Izgleda teško, pa počnimo odozgo nadole. Osnovni koraci će biti:

— slučajan izbor karte,
— slučajan izbor boja,
— crtanje karte na odgovarajućem mestu na ekranu;
— ponavljanje postupka za svih 5 karata.

Jasno je da će glavni problem biti crtanje *karata* pa ću se trenutno zadržati na njemu.

Biće nam potrebni simboli za herc, tref, karo i pik. Tu nema problema — PET-grafika. Rub karte se takođe može nacrtati PET-grafikom. Ono što će biti problem je *pozicioniranje* simbola. Cilj mi je da rezultat bude zadovoljavajući, ali ne baš tako lep kao prava igraća karta.

Nakon što sam na papiru isprobao nekoliko ideja, odlučio sam se za standardnu mrežu simbola kao na slici 25.1.



Sl. 25.1 — Standardna mreža karata za igranje

Ima 11 mogućih mesta na koje će se smestiti simbol, a na slici su ta mesta obeležena brojevima. Npr. dobra trojka pik može se dobiti, ako se znak pik nacrta na položajima 3, 6, 9. Znači, za svaku vrednost karte od 1—10 potrebno je definisati *koja* od ćelija je zauzeta. Za vrednost karte koristiću promenljivu KARTA čija vrednost može da bude od 1—10. Uzimajući za uzor logiku bajtova mogu svakoj vrednosti promenljive KARTA da dodelim niz od 11 nula i jedinica pri čemu nula na K-tom mestu znači da ćelija sa brojem K na slici 25.1 ostaje prazna, dok 1 znači da u ćeliju treba staviti simbol boje. Tako vrednosti KARTA = 3 odgovara niz 00100100100 koji ima jedinice samo na mestima 3, 6 i 9. Jeste li shvatili? To je dobra ideja, pošto svih 10 nizova mogu da smestim u znakovnu matricu C$.

Pošto je konačna lista prilično velika, daću je deo po deo, a vi možete da je otkucate odjednom, ako želite, ali ja vam savetujem da i vi unosite deo po deo. Do sada, moja ideja je dovela do:

```
100 DIM C$(10)
110 C$(1)="00000100000"
120 C$(2)="00100000100"
130 C$(3)="00100100100"
140 C$(4)="11000000011"
150 C$(5)="11000100011"
160 C$(6)="11011000011"
170 C$(7)="11100011011"
180 C$(8)="11100011111"
190 C$(9)="11011111011"
200 C$(10)="11111011111"
```

POLOŽAJ NA KOJEM SE ISCRTAVA

Do sada je dobro, ali ćelije su prilično nepravilno raspoređene. Ovo se može rešiti tako da se definiše red i kolona za svaku ćeliju K u K-tom elementu numeričkih matrica R i C. Sa sl. 25.1. imamo:

```
250 DIM R(11):DIM C(11)
260 R(1)=0:C(1)=0
270 R(2)=0:C(2)=4
280 R(3)=2:C(3)=2
290 R(4)=4:C(4)=0
300 R(5)=4:C(5)=4
310 R(6)=5:C(6)=2
320 R(7)=6:C(7)=0
330 R(8)=6:C(8)=4
340 R(9)=8:C(9)=2
350 R(10)=10:C(10)=0
360 R(11)=10:C(11)=4
```

BOJE

Boje ću obeležiti sledećim brojevima:

1. Herc
2. Tref
3. Karo
4. Pik

A sada vrednosti koje naredbom POKE treba upisati u ekransku memoriju. To su 83, 88, 90 i 65. Opet bez nekog pravila. Uzećemo dakle, opet, numeričku matricu (nazovimo je S) da memorišemo ove podatke:

```
400 S (1)=83 : S (2)=88 : S (3)=90 : S (4)=65
```

(S ima samo četiri elementa, pa neću da je dimenzionišem. Ako se matrica sa jednom dimenzijom ne dimenzioniše, BASIC podrazumeva da je dimenzionisana brojem 10).

Herc i karo su *crvene* boje, a pik i tref *crne*. Koristićemo kodove za boje: 0 (crno) i 2 (crveno) i memorisaćemo ih u matrici K:

```
410 K (1)=2 : K (2)=0 : K (3)=2 : K (4)=0.
```

GLAVNI PROGRAM

A sada na posao! Da bismo olakšali posao, hajde da vidimo kako ćemo nacrtati *jednu* kartu čija se rešetka simbola (slika 25.1) iscrtava tako da gornji levi ugao bude u redu RED i koloni KOL. (Razmišljamo uopšteno, setite se.) Trenutno ću RED postaviti na 5, a KOL na 7. To ćemo izmeniti kasnije kad napravimo petlju koja će dati deljenje sa *pet* karata.

Prvo izaberimo slučajne vrednosti za promenljive KARTA i BOJA:

```
500 KARTA=1+INT (10 * RND (0))
510 BOJA=1+INT (4 * RND (0))
```

Zatim postavljamo vrednosti promenljivih RED i KOL:

```
520 RED = 5
530 KOL = 7
```

Zatim utvrđujemo kodove simbola i boje:

```
540 SCD=S (BOJA) : KR=K (BOJA)
```

Posle toga crtamo kartu. Pošto još ne znamo kako ćemo to uraditi, pozvaću potprogram:

```
550 GOSUB 2000 : REM CRTANJE KARTE
```

i (u ovom trenutku):

```
560 STOP.
```

POTPROGRAM ZA CRTANJE KARTE

Na sl. 25.1 imamo 11 ćelija od kojih svaka odgovara mogućoj "šari" na karti. Znači, potrebno je da pretražimo svako od 11 mesta u nizu C$ (KARTA) pa ako je na tom mestu, crtamo šaru na karti.

```
2000 REM TRAZENJE SARE
2010 FOR X=1 TO 11
2020 U$=MID$(C$(KARTA), X, 1)
```

```
2030 IF U$="0" THEN 2050
2040 GOSUB 3000:REM CRTANJE SARE
2050 NEXT X
2060 RETURN
```

Već se pojavljuje... Sada da nacrtamo šaru:

```
3000 REM CRTANJE SARE
3010 RC=RED+R(X):CC=KOL+C(X)
3020 POKE 1024+40*RC+CC, SCD
3030 POKE 55296+40*RC+CC, KR
3040 RETURN
```

Ako otkucate RUN, primetićete da je trebalo da brišemo ekran i da izaberemo boju. A osim toga karta nema rub.
Inicijalizovanje je lako:

```
10 POKE 53281, 7
20 PRINT CHR$ (147)
```

POBOLJŠANJE CRTEŽA

Da bismo nacrtali rub karte i da bismo ispisali njenu vrednost i boju, potreban nam je još jedan potprogram:

```
545 GOSUB 4000 : REM RUB KARTE
```

Ne mogu dalje da odlažem. Evo:

```
4000 REM RUB KARTE
4010 PRINT CHR$ (19);
4020 FOR Y=1 TO RED—3 : PRINT : NEXT Y
4030 C0=KOL—4
4040 PRINT TAB (C0); "gUs (11 puta g * s) gIs"
4050 FOR Y=1 TO 15
4060 PRINT TAB (C0); "g—s (11 razmaka) g—s"
4070 NEXT Y
4080 PRINT TAB (C0); "gJs (11 puta g * s) gKs"
```

Ovim smo dobili rubove. Primetimo da je g * s grafičko * plus SHIFT, a da je g—s grafičko — plus SHIFT.
A sada boja i broj:

```
4100 PRINT CHR$(19);
4110 FOR Y=1 TO RED—2:PRINT:NEXT Y
```

```
4120 PRINT TAB(KOL—3);KARTA
4130 PRINT TAB(KOL—3);CHR$(SCD+32)
4140 RETURN
```

SCD+32 služi za konverziju ekranskog koda (POKE) u ASCII — kod. Razni izrazi KOL—4, KOL—3 itd. pronađeni su metodom pokušaja i grešaka (mnogo grešaka). Nemojte misliti da se uvek sve rešava tako lako kao što može izgledati na osnovu ove knjige.

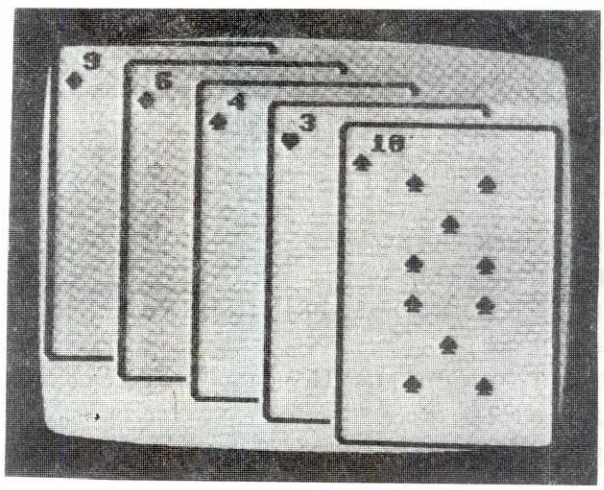
KOMPLETNO DELJENJE

Kad smo uspeli da otklonimo sve greške i pustili program da radi, možemo da pređemo na crtanje svih pet karata.

Izbrišite redove 520, 530 i 560. Dodajte:

```
490 FOR Z=1 TO 5
520 RED=5+Z:KOL=5+4*Z
560 NEXT Z
570 GOTO 570
```

Eto!



Sl. 25.2 — Konačni rezultat: karte dobijene u pokeru

Vežba 1

Dodajte deo programa koji proverava svako slučajno izvlačenje karte i boje da bi se obezbedilo da ista karta nije već izvučena i ako jeste da izvlači ponovo (dva asa pika u igri ne bi baš oraspoložila protivnika u pokeru).

Neću dati drugu vežbu da ne biste gubili vreme koje vam je potrebno za odgovor na prvu vežbu. Ipak: razmislite kako biste nacrtali slike (u kartama).

ODGOVORI

Vežba 1

Potrebna nam je matrica za ispitivanje svake nove karte i boje. Evo jednog načina:

```
511 FOR W=1 TO 5
512 IF F(W)=KARTA AND G(W)=BOJA THEN 500
513 NEXT W
514 F(Z)=KARTA:G(Z)=BOJA
```

Ovim se vrednosti KARTA memorišu u matrici F, a vrednosti BOJA u matrici G. Pošto one imaju samo po 5 elemenata, nije ih potrebno dimenzionisati.

Striktno govoreći, iskakanje iz petlje je loš običaj. Zbog toga nam je potreban indikator koji će nakon završetka petlje aktivirati grananje:

```
512 IF F(W)=KARTA AND G(W)=BOJA THEN FLAG=1
515 IF FLAG=1 THEN FLAG=0:GOTO 500
516 FLAG=0
```

Međutim, ovde je to možda malo suviše pedantno.

26 Pronalaženje i otklanjanje grešaka IV

U poglavlju "Pronalaženje i otklanjanje grešaka III" govorio sam o *suvom testiranju* programa gde se ručno prati tok izvršenja naredbi. Glavni nedostatak ove tehnike je u tome što korisnik — a to ste vi — mora sam da obavi ceo posao. "Šezdesetčetvorka" samo samozadovoljno čeka i kaže vam da nema pojma o čemu pričate, ili što je još gore, daje samo pogrešne odgovore. Vreme je da je nateramo da zaradi za život.

Ona se može naterati da sama formira tabele suvog testiranja. Npr. pogledajmo opet vežbu 1. iz tog poglavlja:

```
10 MAX=0:MIN=0
20 INPUT N
30 IF N<MIN THEN MIN=N
40 IF N>MAX THEN MAX=N
50 IF N<0 THEN PRINT MAX, MIN:END
60 GOTO 20
```

Program treba da pronađe minimum i maksimum od niza upisanih brojeva, ali on to ne radi. Evo kako ćemo utvrditi gde su greške. Dodajte red 5 koji ispisuje zaglavlje tabele, a zatim iza svakog reda ispisujte brojeve redova i vrednosti promenljivih koje se u tom redu menjaju u odgovarajućim kolonama:

```
5 PRINT "LN MAX MIN N GRANANJE"
10 MAX=0:MIN=0:PRINT "10";TAB(5);MAX;TAB(10);MIN
20 INPUT N:PRINT "20";TAB(15);N
30 IF N<MIN THEN MIN=N:PRINT "30";TAB(10);MIN
40 IF N>MAX THEN MAX=N:PRINT "40";TAB(5);MAX
50 IF N<0 THEN PRINT MAX, MIN:END
60 PRINT "60";TAB(21);"+":GOTO 20
```

Naredba TAB samo izaziva ispisivanje u odgovarajućim kolonama ekrana. Oblik tabele se nešto razlikuje od tabele koju smo crtali rukom. Redovi u kojima

nema izmena ne prikazuju se, ali zato se lakše prati. Ozbiljniji problem je to što pripremne poruke ulaza i upisane vrednosti presecaju tabelu. Idealno bi bilo kad bi se dve vrste izlaza mogle slati na različita mesta. Npr. tabela "suvog testiranja" mogla bi da se šalje na štampač, ili ako ga nemate, na kasetu. Da biste to mogli da uradite, potrebno je da budete upućeni u "šezdesetčetvorkin" sistem datoteka, pa ću se na ovaj problem vratiti kad budemo razmatrali datoteke u poglavlju 34. Ako nemate štampač, možda ćete hteti da ekonomišete sa ispisanim vrednostima. Npr. u ovom slučaju, verovatno će biti dovoljno da se znaju vrednosti MAX i MIN na kraju svake petlje. Tako bi moglo da bude dovoljno da se u originalni program samo doda red 55:

```
55 PRINT MAX, MIN
```

Možda biste hteli da izmenite promenljive da biste probali razne varijante. Dobar trik je da se u red sa naredbom PRINT upiše naredba REM koja tako deaktivira naredbu PRINT:

```
55 REM PRINT MAX, MIN
```

Pošto je sada red 55 u stvari komentar, računar će ga ignorisati. Ako vam kasnije bude potreban, samo izbacite REM. Tako štedite vreme za ponovno kucanje.

Tako nam "šezdesetčetvorka" može reći kako se menjaju promenljive. Postoje još dve korisne stvari koje ona može da uradi:

1. Može da nam kaže koje redove izvršava i kojim redom.
2. Može da nam kaže koliko često izvršava određeni red ili deo programa.

PRAĆENJE IZVRŠENJA

Najjednostavniji način praćenja izvršenja programa je da se u svaki red doda naredba PRINT koja jednostavno prikazuje broj reda koji se izvršava. Sa tim dodatkom program za minimum i maksimum izgledao bi ovako:

```
10 PRINT "<10>":MAX=0:MIN=0
20 PRINT "<20>":INPUT N
30 PRINT "<30>":IF N<MIN THEN MIN=N
40 PRINT "<40>":IF N>MAX THEN MAX=N
50 PRINT "<50>":IF N<0 THEN PRINT MAX,MIN:END
60 PRINT "<60>":GOTO 20
```

Brojeve redova sam stavio u zagrade (< >) tako da ih ne pobrkate sa onim što treba da ispiše program.

Primetimo da se deo za praćenje nalazi uvek na početku reda. Obično redosled nije bitan, ali da sam red 30 napisao ovako:

```
30 IF N<MIN THEN MIN=N : PRINT "<30>"
```

naredba PRINT bi se izvršavala *samo* kad je N manje od MIN, pa bi praćenje bilo nekompletno.

Praćenje može da bude više zbunjujuće nego korisno, ako se pretera. U bilo kojem postupku pronalaženja grešaka treba biti oprezan i postavljati pitanja koja imaju smisla, a zatim mašinu navesti da obezbedi odgovore *samo* na ta pitanja. Nema smisla vršiti kompletno "suvo testiranje", ako želite da znate da li određena promenljiva ikada ima vrednost 25. Što više podataka tražite od mašine, to vam više vremena treba da te podatke analizirate. Isto važi i za praćenje. Ako ste sigurni da se neki deo programa pravilno izvršava, što da se mučite i da ga pratite. Uopšte, program treba pratiti u blizini mesta grananja.

Evo primera. Napišimo deo programa koji prihvata dan u mesecu (podatak koji se koristi negde u programu). Dobro je uobičajiti da se kontroliše upisana vrednost da bi se obezbedilo da bude u opsegu 1 — 31. *Mogli* bismo napraviti i komplikovaniju kontrolu koja bi proveravala da dan nije veći od 29, ako je mesec februar i sl. ali ostanimo, za sada, kod najjednostavnije kontrole. Naš prvi pokušaj mogao bi da bude:

```
50 INPUT "UPISITE DAN U MESECU"; D
60 IF D>0 OR D<32 THEN 200
70 PRINT "NEMOGUC DAN"
80 GOTO 50
: : : : : : : : : : :
200 REM OVDE SE PRELAZI AKO JE DAN VAZECI
```

Program se ne ponaša ispravno pa možemo da uradimo sledeće:

```
60 PRINT "<60>" : IF D>0 OR D<32 THEN 200
80 PRINT "<80>" : GOTO 50
200 PRINT "<200>" : REM OVDE SE PRELAZI AKO JE DAN VAZECI
```

tako da se prati samo onaj deo programa na koji se sumnja. Kad pustimo program u izvršenje, utvrdićemo da, bez obzira koju vrednost upišemo za D, praćenje daje uvek isti rezultat:

```
<60>
<200>
```

Program ne može da dođe do reda 70. Red 60 *bi trebalo* da bude ovakav:

```
60 IF D>0 AND D<32 THEN 200
```

Ako malo razmislite, *bilo koji* broj je ili veći od 0 ili manji od 32.

Problem se javlja prilično često pošto smo dosta neobazrivi kad koristimo reči AND i OR ("i" i "ili") u običnom govoru, pa se ta neobazrivost može preneti i u naše programiranje. U konverzaciji slušalac prilično dobro razume o čemu govornik govori, ponekad čak i pre nego što završi rečenicu, tako da su dozvoljene male netačnosti. Računari ne opraštaju tako lako.

PROFIL PROGRAMA

Profil programa pokazuje koliko puta se izvršava određeni red. Kao i obično ne treba preterivati, pa treba da budemo selektivni u pogledu toga za koji deo programa želimo profil. To se radi prilično lako. Pretpostavimo da želimo da utvrdimo koliko puta se izvršava red 420 u nekom programu. Na početku programa brojač treba postaviti na nulu i povećati ga za 1 svaki put kad se izvrši red 420:

```
5 PC=0
.....
420 A=A*(P—1)
421 PC=PC+1
.....
809 PRINT PC
810 STOP
```

Hajde da pogledamo konkretan primer. Program u primeru treba da prihvati najviše 20 vrednosti (upisivanje se završava nulom) i da ih složi po rastućem redosledu. Znači, ako se upišu vrednosti:

```
3
8
1
4
2
0
```

program bi trebalo da ispiše:

```
1
2
3
4
8
```

Nula ne bi trebalo da se pojavi, jer je samo ograničavač.

```
10 DIM A(20)
20 FOR P=1 TO 20
30 INPUT A(P)
40 IF A(P)=0 THEN 60
50 NEXT P
60 N=P
65 F=0
70 FOR P=1 TO N
```

```
80 IF A(P)<A(P+1) THEN 130
90 T=A(P)
100 A(P)=A(P+1)
110 A(P+1)=T
120 F=1
130 NEXT P
140 IF F=1 THEN 65
150 FOR P=1 TO N
160 PRINT A(P)
170 NEXT P
```

Program ne radi ono što treba (otkucajte ga i probajte). U stvari, on ulazi u beskonačnu petlju.

Pa gde da počnemo da gledamo? Prva petlja (20—50) izgleda dovoljno bezopasno, a završna (150—170) samo ispisuje sadržaj matrice. Izgleda razumno da se skoncentrišemo na petlju od 70 do 130. Iz reda 80 se jasno vidi da se nekad izvršavaju sve naredbe petlje, a nekad se redovi od 90 do 120 ignorišu. Pa dobro, da uzmemo dva brojača C1 i C2 koji će brojati koliko puta je program ušao u petlju i koliko puta je izvršen završni deo petlje.

Ovo možemo postići sa:

```
67 C1=0
68 C2=0
75 C1=C1+1
125 C2=C2+1
132 PRINT C1, C2
```

Kad smo već kod toga, možda bismo mogli da ispišemo sadržaj matrice na kraju svake petlje, pošto je jasno da se brojevi unutar nje pomeraju, a da se osim toga koristi veoma malo promenljivih. Evo:

```
134 FOR Q=1 TO N
135 PRINT A(Q);
136 NEXT Q
137 PRINT
138 FOR X=1 TO 200:NEXT X
```

Da probamo sa nekoliko skupova podataka, da vidimo šta će se desiti. Ako upišemo:

```
3
6
1
8
5
0
```

dobićemo:

```
6          4
316500
6          4
135006
6          2
130056
6          2
100356
6          2
001356
6          1
001356
6          1
001356
```

itd. do beskonačnosti.

Pa, izgleda da se vrednosti slažu po redu, ali smo negde izgubili "8", a nije jasno ni odakle dve nule. Osim toga, glavnu petlju program stalno izvršava 6 puta, a broj izvršavanja potpetlje se stalno smanjuje dok se ne ustali na 1.

Jedna od nula je očigledno ograničavač skupa, a druga je elemenat tabele koji nije dobio određenu vrednost u toku izvršenja nego ga je sistem inicijalizovao na nulu. Drugim rečima, program obrađuje dve vrednosti suviše. Pa hajde da izmenimo red 60:

```
60 N=P—2
```

i pokušajmo ponovo. Nada je večna. Dobićemo:

```
4          2
3165
4          2
1356
4          0
1356
1
3
5
6
```

Na neki način, to je napredak — oslobodili smo se nula. Međutim, još uvek nam nema osmice.

Teško se može videti kako je nestala. Možda je još uvek tu, ali se ne ispisuje. Gde koristimo naredbu PRINT? U redovima 150—170. Mora da je opseg 1 TO N suviše mali. Povećajmo ga za 1:

```
150 FOR P=1 TO N+1
```

A, naravno, kad smo već tu, može se pretpostaviti da i u redu 134 imamo isti problem. Pa hajde da i to ispravimo:

```
134 FOR Q=1 TO N+1
```

Hmmmmm. Još jednom u napad, dragi prijatelji, još jednom .....
Ovog puta ćemo (za iste podatke) dobiti:

```
4          2
31658
4          2
13568
4          0
13568
1
3
5
6
8
```

Divno! Sredili smo ga. Radi savršeno. Da li? Probajmo:

```
3
5
2
1
5
0
```

Sada dobijamo:

```
4          3
32155
4          3
21355
4          2
12355
4          1
12355
4          1
12355
4          1
12355
itd.
```

Dolazi do dobrog rezultata, ali nikad ne izlazi iz petlje. Primećujemo da se C2 nikad ne spušta na nulu, a možete se kladiti da je to ono što završava program.

Šta odlučuje da li program ulazi u potpetlju ili ne? Red 80:

```
80 IF A (P)<A (P+1) THEN 130
```

Razlika između dva skupa podataka je u tome što drugi skup sadrži dve identične vrednosti. Pošto 5 nije manje od 5, potpetlja će se izvršavati svaki put kad program naiđe na dve petice. To je razlog zašto program uvek jednom uđe u potpetlju. Možda bi trebalo napisati:

```
80 IF A (P)<=A (P+1) THEN 130
```

Sada sve funkcioniše.

```
4           2
32155
4           2
21355
4           1
12355
4           0
12355
1
2
3
5
5
```

Sada radi kako treba, pa možemo da izbrišemo redove za testiranje.

Nadam se da sam ovim ilustrovao dve važne tačke. Prvo, nije trebalo da tačno znamo kako funkcioniše postupak. Ako ste ovo pažljivo proradili, verovatno vam je sad prilično jasan, a nekoliko "suvih prolaza" bi vas verovatno ubedilo da ga i razumete. (Suvi prolazi su dobar način za razumevanje postupka računara. Često mi je bilo potrebno i tuce prolaza na nekom nejasnom delu programa — nečijeg, ne mog, naravno — pre nego što mi je bilo stvarno jasno o čemu se radi). Drugo, uvek postoji sklonost da se poveruje kako je sve u redu kad se program prvi put uspešno izvrši i da sada možete da svratite na čašicu. Kao što smo videli, posao *nije* završen pošto mogu postojati skupovi podataka za koje program ne radi kako treba. Uostalom, kafana je zatvorena još pre sat i po, ako i vama vreme prolazi tako brzo kao meni kad pišem program.

Mogu postojati i efikasniji načini definisanja velikog broja promenljivih nego što je to kucanje velikog broja sličnih programskih redova.

27 Liste podataka

Ako prelistate stranice ove knjige, primetićete da dobar deo programa uključuje poveće količine brojeva i znakovnih nizova. Oni se često memorišu u matricama. U poglavlju 25 je jedan ekstreman primer. Ako se ima u vidu da je jedna od pretpostavljenih jakih strana računara njegova mogućnost da izvršava slične naredbe mnogo puta *bez* intervencije čoveka, sigurno mora da postoji bolji put nego da programer stalno ponavlja skoro identične redove. Brojevi i znakovni nizovi se ipak na *neki* način moraju uneti u memoriju, pošto "šezdesetčetvorka" nema telepatske moći. Način postoji. Za formiranje liste podataka koristi se naredba:

```
DATA
```

a za izvlačenje pojedinih podataka iz liste naredba:

```
READ
```

Podaci u naredbi DATA mogu da budu znakovni nizovi ili brojevi, a mogu se i mešati (odnosno u jednoj naredbi DATA mogu se pojaviti i brojevi i znakovni nizovi). Međutim, programer mora da obezbedi da naredba READ koristi ispravan *tip* promenljive. Ako naredba:

```
READ X
```

naiđe na znakovni niz, neće funkcionisati, pošto za znakovni niz treba koristiti znakovnu promenljivu:

```
READ X$
```

Pogledajmo, npr. program koji ispisuje gradove i broj njihovih stanovnika:

```
10 PRINT CHR$(147)
20 DATA BEOGRAD, 1250000, ZAGREB, 850000, LJUBLJANA, 320000,
   SKOPLJE, 400000
30 FOR T=1 TO 4
```

```
40 READ GRAD$, BRSTA
50 PRINT GRAD$, BRSTA
60 NEXT T
```

Znakovni nizovi mogu biti među navodnicima, ali ne moraju. Za naredbu DATA svejedno je da li napišete BEOGRAD ili "BEOGRAD". Međutim, postoji izuzetak: znakovni niz koji sadrži zarez, mora biti među navodnicima.

Vežba 1

Evo spiska prvih pet predsednika SAD sa godinama u kojima su predsednikovali. Napišite program koji koristi listu DATA da ispiše te informacije.

George Washington	1789—1797
John Adams	1797—1801
Thomas Jefferson	1801—1809
James Madison	1809—1817
James Monroe	1817—1825

TRAŽENJE PLANETA

Listama DATA se može postići više nego samo ispisivanje podataka. Npr. možete ih pretraživati dok ne nađete odgovarajući podatak.

```
10 DATA MERKUR, 58, VENERA, 108
20 DATA ZEMLJA, 150, MARS, 228
30 DATA JUPITER, 778, SATURN, 1427
40 DATA URAN, 2870, NEPTUN, 4997, PLUTON, 5900
50 INPUT "KOJA PLANETA"; PLANETA$
60 FOR G=1 TO 9
70 READ X$, Y
80 IF X$=PLANETA$ THEN 100
90 NEXT G
100 PRINT X$; "JE UDALJEN/UDALJENA"
110 PRINT Y; "MILIONA KILOMETARA";
120 PRINT "OD SUNCA"
```

Redovi 10—40 su liste podataka — naziv planeta i njegovo rastojanje od Sunca u milionima kilometara. Redovi 60—90 čine petlju koja *pretražuje* liste podataka tražeći dati planet upisan prilikom izvršavanja reda 50. Kad ga pronađe, program prelazi na redove 100—120 koji ispisuje tražene podatke.

(Primedba: PLANET$ treba uneti *baš* onako kako je napisano u listi, jer će inače program izaći iz petlje, a da ne pronađe ono što ste tražili, a zatim će prikazati podatke za Pluton (zašto Pluton?)).

Kad dođete u putničku agenciju, ili na proveru za ulazak u avion, a službenik upiše vaše ime da bi utvrdio da li ste rezervisali mesto za taj let, on koristi jednu (znatno usavršeniju) verziju ovog tipa pretraživanja podataka. Naravno, lista podataka je *enormna* i veći deo programskog umeća utroši se na to da se tolikim podacima manipuliše bez greške.

NAREDBA RESTORE

Ponekad ćete možda hteti da koristite iste liste podataka više puta u istom programu. Daću jedan veoma jednostavan primer. Šta će se desiti?

```
10 DATA FRED
20 READ X$
30 PRINT X$
40 GOTO 20
```

Dobićete prvo ispisivanje imena FRED, a zatim poruku OUT OF DATA. Računar ne zna da ste vi hteli da ponovo koristite istu listu podataka.
Da biste mu to naznačili, koristite reč:

```
RESTORE
```

Pokušajte da dodate red:

```
35 RESTORE
```

pa ćete videti. Naredba RESTORE vraća na početak liste "ukazivač" na tekući položaj u listi podataka.

ODGOVORI

Vežba 1

```
10 DATA GEORGE WASHINGTON, 1789, 1797, JOHN ADAMS, 1797,
   1801
20 DATA THOMAS JEFFERSON, 1801, 1809, JAMES MADISON, 1809,
   1817
30 DATA JAMES MONROE, 1817, 1825
40 PRINT CHR$(147)
50 FOR N=1 TO 5
60 READ IME$, D1, D2
70 PRINT IME$, D1;"—"; D2
80 NEXT N
```

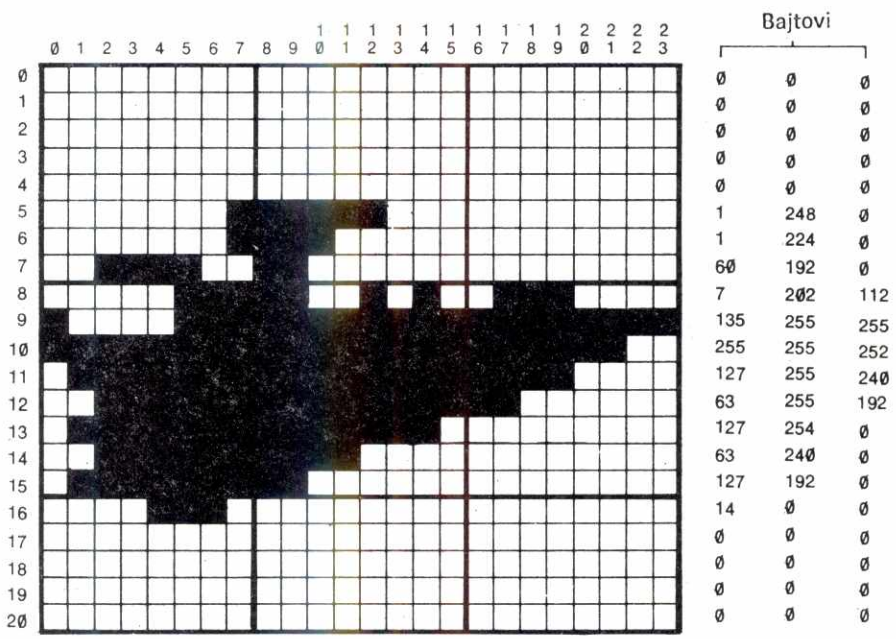
Jedna interesantna i neuobičajena karakteristika „šezdesetčetvorke" je mogućnost crtanja i pomeranja velikih grafičkih blokova po ekranu, preklapanje tih blokova i kontrola sudara.

28 Sprajtovi

Sprajtovi ili MOB-ovi (Moveable Object Block = pokretni blok) su umereno veliki grafički crteži kojima upravlja posebno integrisano kolo VIC, a mogu se kretati po ekranu po želji programera. Oni mogu biti osnova za mnoge atraktivne igre i prikaze. Njima se ne može rukovati baš jednostavno, pa je cilj ovog poglavlja da vam prikaže nekoliko osnovnih stvari — dovoljno da sami možete da koristite sprajtove.

CRTANJE SPRAJTA

Podaci koji definišu sprajt sastoje se od mreže sa 21 × 24 ćelije koje su prazne ili popunjene. Sl. 28.1 prikazuje primer "svemirske krstarice".



Sl. 28.1 — Svemirska krstarica na mreži sprajta i odgovarajući podaci konvertovani iz binarnog u decimalni oblik

Prazne i popunjene ćelije prikazuju se nizom brojeva koji se memoriše na odgovarajućem mestu (vidi dalje u tekstu). Da biste formirali taj niz brojeva, svaku praznu ćeliju označite nulom, a svaku popunjenu jedinicom kao na sl. 28.1. Uzmite zatim svaki red od 24 cifre i podelite ga na tri niza od po 8 cifara. Za naš primer red 8 sa slike će dati:

00000111 11001010 01110000

Ovo liči na binarne bajtove . . . pa u tome i jeste stvar. Ako se konvertuju u decimalni oblik, dobićemo:

7 202 112

Znači, svaki red sprajta može se posmatrati kao niz od tri decimalna broja (između 0 i 255). Brojevi za ceo sprajt izlistani su sa strane na sl. 28.1. Oni se čitaju od levog gornjeg ugla do desnog donjeg, tj. prvo se čitaju tri bajta za red 0, pa tri bajta za red 1 i tako sve do reda 20. To čini ukupno 63 broja. Svoj sprajt *možete* da nacrtate na karo-papiru i da izračunate decimalne vrednosti prema tabeli iz Dodatka 1, ali zar ne bi bilo lepše kad bi računar uradio teži deo posla?

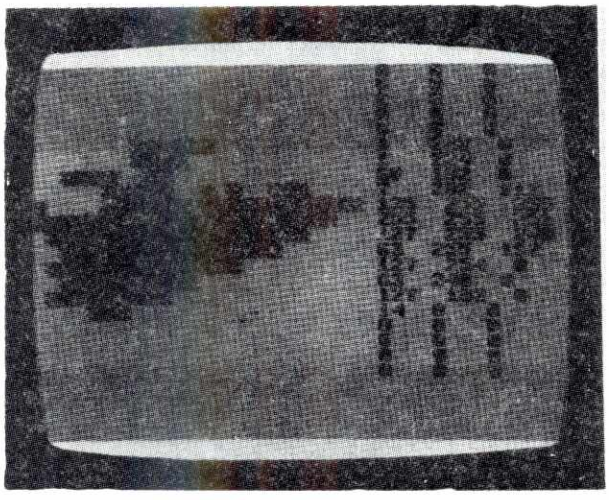
CRTANJE SPRAJTA POMOĆU RAČUNARA

Dajemo primer prilično jednostavnog programa koji vam omogućava da na ekranu nacrtate sprajt i generišete listu podataka. Da bi lista ostala dovoljno kratka, izostavljena su različita moguća poboljšanja. Ako želite da poboljšate program, samo napred!

```
 10 POKE 53280,4
 20 PRINT CHR$(147)
 30 FOR S=0 TO 20
 40 IF S=8*INT(S/8) THEN PRINT
    "_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _"
    [24 ZNAKA —]
 50 IF S<>8*INT(S/8) THEN PRINT
    ". . . . . :. . . . . :. . . . . :"
 60 NEXT S
 80 PRINT : PRINT : PRINT
100 DIM S(20, 23)
110 FOR LIN=0 TO 20
120 FOR KOL=0 TO 23
130 CDE=63 : GOSUB 1000
140 GET A$
150 IF A$<>"0" AND A$<>"1" THEN 140
160 IF A$="0" THEN S(LIN, KOL)=0 : CDE=32 : GOSUB 1000
170 IF A$="1" THEN S(LIN, KOL)=1 : CDE=102 : GOSUB 1000
175 IF A$="2" GOTO 210
```

```
180 NEXT KOL
190 NEXT LIN
200 POKE 53280,3
210 GET A$ : IF A$<>"N" AND A$<>"D" THEN 210
220 IF A$="N" THEN POKE 53280,4 : GOTO 110
250 PRINT CHR$(19);
260 FOR LIN=0 TO 20
270 FOR X=0 TO 16 STEP 8
280 V=0
290 FOR KOL=0 TO 7
300 IF S(LIN, X+KOL)=1 THEN V=V+2↑(7-KOL)
310 NEXT KOL
320 PRINT TAB(24+X/2) STR$(V);
330 NEXT X
340 PRINT
350 NEXT LIN
360 GOTO 360
1000 REM PRINT AT LIN, KOL, CDE
1001 REM "PRINT AT" POTPROGRAM IZ POGLAVLJA 19
1010 POKE 1024+40*LIN+KOL, CDE
1020 RETURN
```

Otkucajte RUN. Rub postaje purpuran iz razloga koji će odmah biti jasni. Dobićete mrežu od 21×24 ćelije nacrtanu tačkama i crticama i izdeljenu na delove 8×8 ćelija. U gornjem levom uglu nalazi se znak "?". Ako pritisnete "1", biće zame-



Sl. 28.2 — Program za crtanje sprajtova u radu

njen tačkastim blokom, a ako pritisnete "0", razmakom. Zatim se znak pomera za jedno mesto. Dalje nastavljate na isti način upisujući blokove ili razmake sve dok potpuno ne popunite mrežu.

U tom momentu rub postaje svetlo-plav da bi vas podsetio da treba da pritisnete neki taster (nema prostora za poruku, pa je ovo zgodno rešenje). Ako upišete "D" (znači "da"), program nastavlja rad, a ako upišete "N" (znači "ne"), program će shvatiti da ste pogrešili i da hoćete da ponovite od početka (kad idete od početka treba upisati sve nule i jedinice, a ne samo one koje menjate, pa je tu moguće poboljšanje programa).

Računar zatim automatski ispisuje podatke za redove na desnoj strani ekrana. Prepišite ih na papir (ili ih odštampajte na štampač ili upišite u datoteku na kaseti, vidi poglavlje 34).

REGISTRI ZA SPRAJTOVE

Za sprajtove su rezervisane posebne zone u memoriji. Adrese počinju od 53248 (koji ću radi lakšeg korišćenja od sada u daljem tekstu zvati V) i završavaju na 53294. Za korisnika nisu upotrebljive sve te adrese, pa ću zbog toga ignorisati one nejasnije. Osim toga, na adresama 2040—2047 ima nekoliko *ukazivača* (engl. pointer) koji računaru kažu gde da traži 63 bajta grafike koji definišu svaki sprajt. Ove ukazivače ću malo kasnije detaljnije opisati, a sada ćemo dati kratak pregled važnijih adresa.

Položaji sprajtova: adrese V do V+15 sadrže broj kolone (ili X-koordinatu u visokom razlaganju) i broj reda (Y-koordinatu) za svaki od 8 sprajtova. Ovi brojevi su od 0—255. Svaki se pamti kao jedan bajt na jednoj adresi.

Indikator korekcije: osam bitova bajta na adresi V+16 definišu korekciju X-koordinate na desnoj strani ekrana. Ako je vrednost bita K sastavljena na 1, broju kolone dodaje se 256. To je neophodno da bi sprajt mogao da dođe do desnog ruba ekrana.

Uključenje/isključenje: osam bitova bajta na adresi V+21 uključuju sprajt K, ako bit K ima vrednost 1, odnosno isključuju sprajt K, ako bit K ima vrednost 0.

Vertikalno širenje: osam bitova bajta na adresi V+23 šire sprajt K na dvostruku visinu, ako bit K ima vrednost 1.

Horizontalno širenje: osam bitova bajta na adresi V+29 šire sprajt K na dvostruku širinu, ako bit K ima vrednost 1.

Indikator sudara: ako se dva sprajta "sudare", odgovarajući bitovi ovog registra postavljaju se na vrednost 1.

Boje: Adrese od V+39 do V+46 sadrže kôd boje (0—15 kao u poglavlju 13) za svaki sprajt.

Ukazivači za podatke: adrese 2040 do 2047 (gornji kraj memorije boja) sadrže ukazivače na početne adrese podataka za sprajtove 0—7. Ako ukazivač K ima vrednost PTR, početna adresa podataka je 64 * PTR. Ovo ćemo zvati PTR-tim *blokom* memorije (od 64 * PTR do 64 * PTR+63). Ovo vam omogućava da sprajt definišete bilo gde u prvih 16348 bajta RAM-a. Postoje načini da se koristi i preostalih 49152 bajta, ali su oni komplikovani (vidi *Vodič* str. 101 i 133). *Međutim*, ne možete prosto ubaciti sprajt na bilo koju adresu, pošto će BASIC uništiti te podatke. (Preporučene adrese navedene su dalje u tekstu.)

Tabela 28.1 — Ukazivači za podatke o sprajtovima

Adresa	Sadržaj
2040	ukazivač za sprajt 0
2041	ukazivač za sprajt 1
2042	ukazivač za sprajt 2
2043	ukazivač za sprajt 3
2044	ukazivač za sprajt 4
2045	ukazivač za sprajt 5
2046	ukazivač za sprajt 6
2047	ukazivač za sprajt 7

Tabela 28.2 — Registri za sprajtove
V = 53248 = početna adresa zone registra

Adresa	Sadržaj								Funkcija
V+ 0	broj kolone sprajta 0								Položaji sprajtova
V+ 1	broj reda sprajta 0								
V+ 2	broj kolone sprajta 1								
V+ 3	broj reda sprajta 1								
V+ 4	broj kolone sprajta 2								
V+ 5	broj reda sprajta 2								
V+ 6	broj kolone sprajta 3								
V+ 7	broj reda sprajta 3								
V+ 8	broj kolone sprajta 4								
V+ 9	broj reda sprajta 4								
V+10	broj kolone sprajta 5								
V+11	broj reda sprajta 5								
V+12	broj kolone sprajta 6								
V+13	broj reda sprajta 6								
V+14	broj kolone sprajta 7								
V+15	broj reda sprajta 7								
V+16	Sp7	Sp6	Sp5	Sp4	Sp3	Sp2	Sp1	Sp0	indikator korekcije
V+21	Sp7	Sp6	Sp5	Sp4	Sp3	Sp2	Sp1	Sp0	uključ./isključenje
V+23	Sp7	Sp6	Sp5	Sp4	Sp3	Sp2	Sp1	Sp0	vertikalno širenje
V+29	Sp7	Sp6	Sp5	Sp4	Sp3	Sp2	Sp1	Sp0	horizont. širenje
V+30	Sp7	Sp6	Sp5	Sp4	Sp3	Sp2	Sp1	Sp0	indikator sudara
V+39	kôd boje sprajta 0								Boje
V+40	kôd boje sprajta 1								
V+41	kôd boje sprajta 2								
V+42	kôd boje sprajta 3								
V+43	kôd boje sprajta 4								
V+44	kôd boje sprajta 5								
V+45	kôd boje sprajta 6								
V+46	kôd boje sprajta 7								

Adrese za upravljanje sprajtovima navedene su u tabelama 28.1 i 28.2, a ponovljene su i u Dodatku 2. Za značenje izostavljenih adresa vidi *Vodič* str. 131—181. To je 50 strana. Zar vam nisam rekao da sprajtovi nisu sasvim jednostavni?

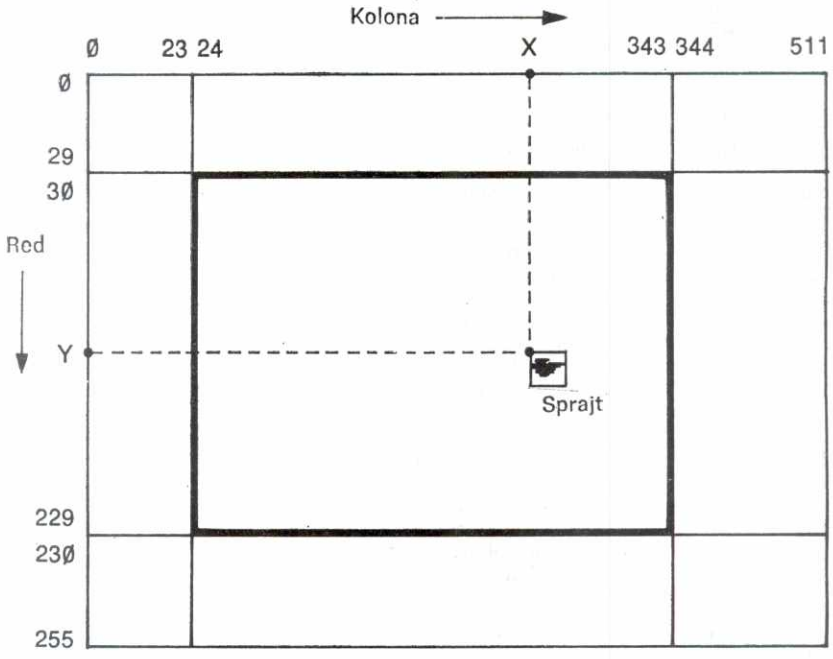
KOORDINATE SPRAJTOVA

Brojevi redova i kolona u kojima se sprajt nalazi mogu da budu u sledećim opsezima:

Red: 0—255

Kolona: 0—511 (uključujući indikator korekcije)

Ovo je znatno više nego zona ekrana koja sadrži 200 redova i 320 kolona. Ovo predstavlja prednost, pošto sprajt možete da smestite van ekrana i da ga zatim postepeno pomerate u vidljivu zonu. Odnos koordinata sprajtova i ekrana prikazan je na sl. 28.3.



Sl. 28.3 — Koordinate sprajtova

Treba imati u vidu da se pod brojem reda i brojem kolone sprajta podrazumeva položaj gornjeg levog ugla mreže 21×24 koja ga definiše. Prilikom pisanja programa o tome treba voditi računa. I kad se sprajt proširi opet se radi o gornjem levom uglu mreže.

FORMIRANJE SPRAJTA

Da bismo na ekranu dobili sprajt, moramo:

1. Definisati podatke za njegov oblik.
2. Postaviti ukazivač na adresu podataka.
3. Naredbom POKE upisati podatke na te adrese.
4. Uključiti sprajt.
5. Definisati boju sprajta.
6. Odrediti broj kolone i reda za sprajt.

Uzmimo sprajt svemirske krstarice iz našeg primera i definišimo ga kao sprajt 1. Evo kako:

```
10 V=53248
100 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
110 DATA 1,248,0,1,224,0,60,192,0,7,202,112,135,255,255
120 DATA 255,255,252,127,255,240,63,255,192,127,254,0
130 DATA 63,240,0,127,192,0,14,0,0,0,0,0,0,0,0
140 DATA 0,0,0,0,0,0
200 POKE 2041,13          [UKAZIVAČ ZA SPRAJT 1 UKAZUJE NA
                          13. BLOK]
210 FOR G=0 TO 62
220 READ H                [UČITAVANJE PODATAKA]
230 POKE 832+G, H         [UPIS U BLOK; 832=64*13]
240 NEXT G
250 POKE V+21,2           [UKLJUČENJE SPRAJTA 1]
260 POKE V+40,7           [SPRAJT 1 ŽUTE BOJE]
270 POKE V+2, 100         [SPRAJT 1 U KOLONI 100]
280 POKE V+3, 100         [SPRAJT 1 U REDU 100]
```

Pažljivo upišite program i otkucajte RUN. Videćete žutu svemirsku krstaricu kako ste i tražili.

Možete eksperimentisati menjajući položaj sprajta direktnim naredbama:

```
POKE V+2, 110
```

pomera sprajt desno

```
POKE V+3, 90
```

pomera sprajt gore

```
POKE V+40, 5
```

menja boju sprajta u zelenu. Probajte i sa drugim vrednostima da vidite šta će se desiti.

KRETANJE

Da biste pokrenuli sprajt, ne treba da učinite ništa drugo nego da stalno menjate brojeve redova i kolona. Npr. *izbrišite* redove 270 i 280 iz primera i dodajte petlju:

```
1000 X=0 : Y=70
1010 POKE V+2, X : POKE V+3, Y
1020 X=X+3 : IF X>255 THEN X=0
1100 GOTO 1010
```

Krstarica će se sada kretati po ekranu sleva udesno. Primetimo da ne prelazi ceo put. To je zbog toga što bi za to trebalo da broj kolone bude veći od 255. Pošto tu ima jedna manja komplikacija, pogledajte dalje tekst.

ŠIRENJE

Sprajt je ipak malo kratak i nezgrapan za svemirsku krstaricu. Ne sekirajte se. Možemo ga proširiti po horizontali dodavanjem reda:

```
290 POKE V+29, 2
```

Ovim se širi sprajt 1, ali ne i ostali pošto je 2=00000010 binarno, tako da samo bit 1 ima vrednost 1. To je već bolje, zar ne?

Da biste sprajt proširili i vertikalno, dodajte:

```
300 POKE V+23, 2
```

Širenje udvostručava dimenziju u datom smeru. Primećujete da je grafika grublja kad se sprajt širi. To je zbog toga što se koriste isti podaci samo je blok 2×1, 1×2 ili 2×2 tačke.

Ne sviđaju mi se glomazne svemirske krstarice, pa izbrišimo red 300.

Samo da vidimo kakve efekte možete da koristite, u petlju za kretanje dodajte:

```
1030 POKE V+29, 2 * INT (2 * RND (0))
1040 POKE V+23, 2 * INT (2 * RND (0))
```

Sada krstarica brzo menja oblik. Da bismo promenili i boju, dodajmo:

```
1050 POKE V+40, 16 * RND (0)
```

Veselo, ali umara, pa izbrišimo redove 1030—1050.

UPRAVLJANJE SA TASTATURE

Ako koristimo petlju sa naredbom GET, možemo kretanjem upravljati sa tastature.

```
1030 GET A$

1040 IF A$="U" THEN Y=Y—4
1050 IF A$="D" THEN Y=Y+4
```

Sada će se krstarica pomerati gore, ako pritisnete taster U, odnosno dole, ako pritisnete taster D. Jasno je da se mogu vršiti i komplikovanije promene položaja, ali ovim smo hteli samo da ilustrujemo mogućnosti.

Da bi kretanje bilo brže zamenite X+3 u redu 1020 sa X+6 ili X+10 (brže kretanje smeta):

```
1020 X=X+6 : IF X>255 THEN X=0
```

KRETANJE PO CELOM EKRANU

Vreme je da se vratimo problemu pomeranja sprajta iza kolone 255 na desnom kraju ekrana.

Sada dolazi na red indikator korekcije u V+16. Ako bit K ima vrednost 1, broj kolone sprajta K povećava se za 256. Zbog toga red 1010 izmenimo u:

```
1010 POKE V+3, Y
1011 IF X>255 THEN OF=1 : POKE V+16, (PEEK(V+16)) OR 2 :
     GOTO 1013
1012 POKE V+16, (PEEK(V+16)) AND 253
1013 POKE V+2, X-OF*256
1014 OF=0
```

Kad ste to uradili, izmenite red 1020 u:

```
1020 X=X+6 : IF X>350 THEN X=0
```

Sada krstarica u svom prolazu pokriva celokupnu širinu ekrana.

PRIORITET SPRAJTOVA

Ako se dva sprajta preklope, onaj sa manjim brojem biće preko onoga sa većim brojem. Donji sprajt će se videti kroz sve "rupe" gornjeg sprajta baš kao što se i očekuje.

Da bismo ovo isprobali, uzmimo još jedan sprajt. Koristićemo isti postupak (u ovom momentu to je dobro za vežbu, ali za kasnije preporučujem bolji pristup, ako želite da koristite *mnogo* sprajtova).

```
400 DATA 0,255,0,3,255,192,15,195,240,63,0,252,255,0,255
410 DATA 63,0,252,127,195,254,31,255,248,3,255,192
420 DATA 0,255,0,0,195,0,1,129,128,3,0,192,6,0,96
430 DATA 15,0,240,15,0,240,7,129,224,3,195,192
440 DATA 1,231,128,0,0,0,31,255,248
500 POKE 2040, 14          [UKAZIVAČ SPRAJTA 0 UKAZUJE NA
                            14. BLOK]
510 FOR G=0 TO 62
520 READ H
530 POKE 896+G, H          [BLOK 14: 896=64*14]
540 NEXT G
```

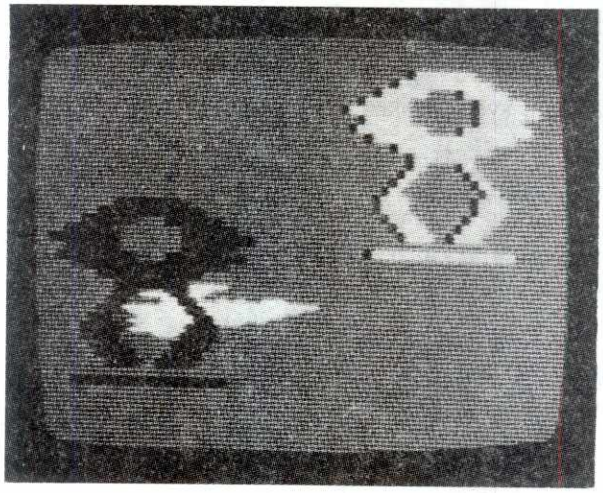
Da biste uključili sprajt 0 i sprajt 1, red 250 treba izmeniti u:

```
250 POKE V+21, 3
```

pošto je 3=00000011 binarno, pa su tako bitovima 0 i 1 dodeljene vrednosti 1. Sada nastavljamo:

```
560 POKE V+39,5      [SPRAJT 0 ZELENE BOJE]
570 POKE V, 120      [SPRAJT 0 U KOLONI 120]
580 POKE V+1,95      [SPRAJT 0 U REDU 95]
590 POKE V+29,3      [ŠIRENJE SPRAJTA 0 I 1 HORIZONTALNO]
600 POKE V+23,1      [ŠIRENJE SPRAJTA 0 VERTIKALNO]
```

Otkucajte RUN i tasterima "U" i "D" navedite krstaricu tako da prođe preko novog sprajta. Vidite li kako prolazi *iza* njega? Tu je zbog toga što "zelena stvar" (sprajt 0) ima veći prioritet od krstarice (sprajt 1).



Sl. 28.4 — Prioritet sprajtova: jedan je iznad drugog

Pretpostavimo da hoćemo da promenimo prioritet. Možemo to da uradimo tako da "zelena stvar" bude sprajt 2, a ne sprajt 0. To podrazumeva sledeće izmene:

```
500 POKE 2042,14
550 POKE V+21,6
560 POKE V+41,5
570 POKE V+4,120
580 POKE V+5,95
590 POKE V+29,6
600 POKE V+23,4
```

Pokušajte sada. Krstarica prolazi ispred, a ne iza "zelene stvari".

KORIŠĆENJE ISTIH PODATAKA ZA VIŠE SPRAJTOVA

Iste podatke možemo da dodelimo ne samo jednom sprajtu, nego nekolicini njih, tako da učinimo da dva ili više ukazivača budu jednaki. Pretpostavimo da smo dobili sprajtove 1 i 2 u navedenom primeru, ali sada želimo da sprajt 0 bude "crna stvar" (takođe dvostruke veličine) na nekom drugom mestu. To možemo da učinimo. Uključimo sva tri sprajta menjajući opet red 250:

```
250 POKE V+21, 7          (7=00000111)
```

Formirajmo sada sprajt 0:

```
700 POKE 2040, 14         [PODACI ZA SPRAJT 0 IZ ISTOG 14. BLOKA]
760 POKE V+39, 0          [SPRAJT 0 CRNE BOJE]
770 POKE V, 70            [SPRAJT 0 U KOLONI 70]
780 POKE V+1, 124         [SPRAJT 0 U REDU 124]
790 POKE V+29, 7          [SVA TRI SPRAJTA PROŠIRENA HORIZONTALNO]
800 POKE V+23, 5          [A SAMO 0 I 2 VERTIKALNO]
```

Ako otkucate RUN, imaćete dve "stvari" i krstaricu.

Sl. 28.5 — Korišćenje istih podataka za više sprajtova

Vežba 1

Izmenite program tako da formirate sprajt 3 kao drugu krstaricu istog oblika, ali crvenu. Neka krstari paralelno sa žutom, ali 20 redova niže.

DETEKTOVANJE SUDARA

Pokušajmo sada da otkrijemo da li se sprajt 2 sudara sa nekim drugim sprajtom. Sudari se detektuju po indikatorima u registru V+30. Ako se dva sprajta sudare, dva odgovarajuća bita dobijaju vrednost 1. Npr. ako se sudare sprajtovi 1 i 2, V+30 će imati sledeći sadržaj:

00000110=6

Vrednost se ažurira pri svakom sudaru. Ako naredbom PEEK pročitate sadržaj registra, sadržaj se automatski vraća na nulu. Da biste videli da li se sprajt 2 sudario sa nekim drugim sprajtom, morate ispitati bit 2 tog registra. To se postiže naredbom:

IF (PEEK (V+30) AND 4) = 4 THEN (bio je sudar. . .)

Ispitivanje sudara možemo staviti u petlju koja počinje u redu 1000 na sledeći način:

```
1025 GOSUB 2000
2000 REM ISPITIVANJE SUDARA SPRAJTA 2
2010 IF (PEEK (V+30) AND 4) = 4 THEN POKE 53280, x—7 * INT
     (x/7)
2020 RETURN
```

Ovim se postiže treptanje ruba, ako dođe do sudara. (Kako?)

Otkucajte RUN i proverite, da li rub trepće svaki put kad se sprajt 1 sudari sa sprajtom 2.

Možda ćete čak utvrditi da je zabavno da stalno očitavate vrednost registra V+30 naredbom PEEK da biste videli šta se dešava:

```
2015 PRINT PEEK (V+30)
```

Ovo pomaže da utvrdite još jednu važnu stvar: prikaz sprajta nezavisan je od uobičajenog pisanja teksta na ekranu, tako da ih na ekranu možete imati u isto vreme. Videćete kolonu brojeva, obično 0, ali kad god se dva sprajta dodirnu, broj se menja. Proverite da li raspored bitova odgovara pravom paru sprajtova:

3=00000011 ako sprajt 1 pogodi sprajt 0
6=00000110 ako sprajt 1 pogodi sprajt 2

Vežba 2

Koji broj će se pojaviti u V+30 ako se:

1. Sprajt 5 sudari sa sprajtom 7?
2. Sprajt 2 sudari sa sprajtom 4?
3. Sprajt 6 sudari sa sprajtom 3?

Vežba 3

Pokušajte da V+30 u redovima 2010 i 2015 izmenite u V+31. Rub još ponekad trepće. Kada? Šta se dešava?

OVO JE SAMO POČETAK

Ovo je bilo dugačko poglavlje, a samo smo zagrebali površinu. Možete, npr. imati višebojne sprajtove. Međutim, nema više prostora, a ja se nadam da ste i ovako naučili dosta da imate čime da se zanimate. Kad jednom savladate ovo što sam vam rekao o rukovanju sprajtovima, možete pogledati i *Vodič* da biste koristii druge mogućnosti.

POBOLJŠANJA

Umesto da sprajtove formirate "iz ruke" možda će vam se više svideti da oformite niz potprograma za rukovanje sprajtovima i da ih koristite po potrebi. U Dodatku 3 navedeno je nekoliko osnovnih.

GDE MEMORISATI PODATKE O SPRAJTOVIMA

Za tri sprajta ili manje možete da koristite blokove 13, 14 i 15. Tu se, u stvari, nalazi *bafer za kasetofon*, pa se ta zona memorije koristi samo kad kasetofon radi. Zbog toga je to sigurno mesto za sprajtove. Na žalost, bafer nije dovoljno veliki da se u njega smesti svih osam blokova od po 64 bajta. Moraćete da pokušate negde drugde. Ako vaš program u BASIC-u nije suviše velik, *Vodič* sugeriše blokove 192—199. Opet, ako želite da saznate više, konsultujte *Vodič*.

GRAND PRIX

Ovo je sve bila teorija. Evo jednog razumno jednostavnog, ali kompletnog programa za igru koji koristi sprajtove. To je jedan od standardnih programa: kola se nalaze na trkačkoj stazi, a vi treba da izbegnete sudare sa ivicom druma. Kola stoje na istoj visini, a staza prolazi. Evo i liste. U programu se koriste neki trikovi koje smo naučili u prethodnim poglavljima.

```
10 PRINT CHR$(147)
20 PRINT "GRAND PRIX"
30 GOSUB 1000: REM POSTAVLJANJE SPRAJTA
40 PRINT
50 INPUT "NIVO: 1—5"; D
60 IF D<1 OR D>5 THEN 50
70 L=9+D : R=31—D
80 PRINT CHR$(147)
85 FOR T=1 TO 2000 : NEXT T
```

```
90 FOR T=1 TO 15
100 PRINT TAB(L); "gWc g+c"; TAB(R); "g+c gQc"
    [IVICA DRUMA]
110 NEXT T
120 TI$="000000"
    [PODEŠAVANJE SATA]
130 POKE V+31, 0
200 PRINT TAB(L); "gWc g+c"; TAB(R); "g+c gQc"
    [NOVA IVICA DRUMA]
210 Q=INT(3*RND(0))-1
220 IF L+Q<0 OR R+Q>38 THEN Q=0
230 L=L+Q : R=R+Q
240 GOSUB 2000 : REM UCITAVANJE SA TASTATURE
250 GOTO 200
1000 REM SPRAJT
1010 V=53248
1020 DATA 0, 126, 0, 0, 255, 0, 49, 255, 140, 49, 255, 140
1030 DATA 63, 255, 252, 49, 255, 140, 49, 255, 140, 3, 255, 192
1040 DATA 3, 255, 192, 3, 255, 192, 3, 255, 192, 3, 255, 192
1050 DATA 3, 255, 192, 3, 255, 192, 49, 255, 140, 49, 255, 140
1060 DATA 63, 255, 252, 49, 255, 140, 48, 255, 12, 0, 126, 0
1070 DATA 0, 24, 0
1080 POKE 2041, 13              [SPRAJT 1 U BLOKU 13]
1090 FOR G=0 TO 62
1100 READ H                     } [FORMIRANJE SPRAJTA U BLOKU 13]
1110 POKE 832+G, H
1120 NEXT G
1130 POKE V+21, 2               [UKLJUČIVANJE SPRAJTA 1]
1140 POKE V+40, 7               [POSTAVLJANJE ŽUTE BOJE ZA
                                SPRAJT]
1150 POKE V+23, 2               [VERTIKALNO ŠIRENJE SPRAJTA]
1160 POKE V+29, 2               [HORIZONTALNO ŠIRENJE SPRAJTA]
1170 X=168                      [HORIZONTALNA KOORDINATA]
1180 POKE V+2, X : POKE
     V+3, 100                   [POZICIONIRANJE SPRAJTA]
1190 RETURN
2000 REM UCITAVANJE SA TASTATURE
2010 P=PEEK(197)                [POSLEDNJE PRITISNUTI TASTER]
2020 IF P=47 THEN
     X=X-3                      [POMERANJE U LEVO AKO JE PRITI-
                                SNUT TASTER <]
2030 IF P=44 THEN
     X=X+3                      [POMERANJE U DESNO AKO JE PRI-
                                TISNUT TASTER >]
```

```
2040 GOSUB 3000 : REM POMERANJE SPRAJTA
2050 IF(PEEK(V+31)AND2)=2 THEN 4000  [ISPITIVANJE SUDARA]
2060 RETURN
3000 REM POMERANJE SPRAJTA
3010 IF X>255 THEN OF=1
3020 POKE V+2, X-256*OF
3030 IF OF=0 THEN POKE V+16, 0
3040 IF OF=1 THEN POKE V+16, 2
3050 OF=0
3060 RETURN
4000 REM KRAJ
4010 M$=TI$                    [OČITAVANJE SATA]
4020 FOR N=1 TO 25             } TREPTANJE EKRANA
4030 POKE 53281, 15*RND(0)     }
4040 NEXT N
4050 POKE 53281, 6             [VRAĆANJE EKRANA NA PLAVO]
4060 PRINT CHR$(19)
4070 PRINT:PRINT
4080 PRINT "gTc [12 PUTA]"
4090 PRINT "UDES NAKON"
4100 PRINT VAL(LEFT$(M$,2));"SATI"          }
4110 PRINT VAL(MID$(M$,3,2));"MINUTA"       } UTROŠENO VREME
4120 PRINT VAL(RIGHT$(M$,2));"SEKUNDE"      }
4130 PRINT "g@c [12 PUTA]"
4140 STOP
```

Kad pustite program u rad, pitaće vas koji nivo težine želite, 1 je lako, a 5 je teško. Ako ne znate, počnite sa 1 i uvežbavajte. Igra počinje odmah kad otkucate nivo i pritisnete RETURN. Ako vam to smeta, dodajte red:

```
85 FOR T=1 TO 2000 : NEXT T
```

pa ćete dobiti nešto vremena za razmišljanje.

Taster [<,] pomera levo, a taster [>.] desno. Pokušajte da izbegnete rub druma. Program koristi detekciju sudara sprajta i pozadine (vidi odgovor na vežbu 3).

ODGOVORI

Vežba 1

U programu dodajte sledeće redove:

```
850 POKE 2043, 13
860 POKE V+42, 2
```

čime se definiše ukazivač i boja. Izmenite prethodne redove da biste uključili sva četiri sprajta i da ih ispravno proširite:

```
250 POKE V+21, 15
790 POKE V+29, 15
```

Sada izmenite petlju:

```
1010 POKE V+3, Y : POKE V+7, Y+20
1013 POKE V+2, X — OF * 256 : POKE V+6, X — OF * 256
```

Vežba 2

1. 10100000 = 160
2. 00010100 = 20
3. 01001000 = 72

Vežba 3

Rub trepće kad se sprajt sudari sa *tekstom*. Registar V+31 obrađuje sudare sprajt-tekst tako da bit K postavi na 1 kad sprajt K dodirne tekst. Strana 158 *Priručnika* ovo opisuje kao "sudar sprajta i pozadine" što je zagonetna definicija. Međutim, to je štamparska greška, a trebalo bi da piše "sudar sprajta i *prednjeg plana*". Slično bi i registar V+27 ("prioritet sprajta i pozadine" u *Priručniku*) trebalo da bude "prioritet sprajta i prednjeg plana". Bit K određuje da li će sprajt K biti iznad ili ispod teksta.

29 Pronalaženje i otklanjanje grešaka V

Kako se konačno dokazuje da program radi baš onako kako je zamišljeno? Ne želim da otvorim komplikovanu "filozofsku" diskusiju (tako smo počeli), ali grubo uzevši, to je kao kad biste pitali astronoma da li će sutra izaći sunce. Ako je veoma pedantan može da vam odgovori da se Zemlja okreće oko Sunca već toliko dugo, a da zakoni fizike govore da će ona i nastaviti da se okreće oko Sunca na dosadašnji način i da bi se na to moglo skoro sigurno kladiti, ali će dodati da nema načina da se utvrdi da li su naši zakoni fizike ispravni i da to što posmatramo hiljadama godina može biti samo jedna od manifestacija nekog kompleksnijeg zakona čiji bi efekat mogao sutra da izazove promenu smera obrtanja Zemlje, ili čak njen potpuni izlazak iz orbite.

SPAVAJUĆE GREŠKE

Po analogiji, to što se program ponaša ispravno za prvu hiljadu skupova podataka, nije apsolutna garancija da će raditi i za hiljadu i prvi skup. U stvari, često se greške pojavljuju tek nakon što je naizgled uspešno završeni program mesecima ili čak godinama radio bez problema stotinama i hiljadama puta. To nije sasvim iznenađujuće. Na kraju, najverovatnije je da programer previdi baš one okolnosti koje se najređe javljaju.

Daćemo jedan primer:

Za elektrodistribuciju pišemo niz programa koji će obrađivati račune za potrošače. Objasnili su nam da postoje dve tarife, A i B. Po tarifi A potrošač plaća kvartalno fiksni iznos od 6.000 dinara, a utrošenu energiju plaća 4 dinara po utrošenoj jedinici. Po tarifi B potrošač ne plaća fiksni iznos, a za utrošenu jedinicu energije plaća 7 dinara.

Prema tim zahtevima pišemo deo programa, kao:

```
100 INPUT "TARIFA (A/B)"; T$
105 INPUT "KOLICINA"; KOLICINA
110 IF T$="A" THEN 300
120 IF T$="B" THEN 140
```

```
130 GOTO 5000
140 CENA=7*KOLICINA/100
150 PRINT CENA
160 GOTO 100
300 CENA=15+4*KOLICINA/100
310 PRINT CENA
320 GOTO 100
    . . .
5000 PRINT "NEKOREKTNA TARIFA"
5010 STOP
```

U redu. Znam da bi program mogao da bude bolji i da su nam potrebne i dodatne informacije kao što je ime potrošača i broj računa, ali dali smo osnovni kostur programa.

Ispitivanje ovog dela programa pokazalo je da dobro radi, pa odlazimo gunđajući da je to što su nam dali da pišemo tako jednostavan program traćenje našeg velikog talenta. I program stvarno radi dobro. Godinama. A onda jednog dana ispisuje račun na 0,00 dinara. To naravno niko ne primećuje pošto je to jedan od hiljada računa, a verovatno se i kovertiraju automatski. Primalac je začuđen i verovatno se dobro zabavlja pošto račun pokazuje koliko računari mogu da budu glupi. On ne preduzima ništa nego baca račun. Na nesreću, napisali smo još jedan program, u istom nizu, koji pamti datum kada je račun poslan, a ako u roku od 28 dana ne dobije potvrdu da je račun plaćen, on piše poslednju opomenu. Ovog puta primalac je više ljut nego što mu je zabavno, ali opomenu kao i ranije baca u korpu za otpatke. Program koji proverava vreme proteklo između slanja računa i uplate, posle 60 dana šalje tehničkoj službi nalog za isključenje.

Šta se desilo? Potrošač je stariji penzioner koji koristi jedan od jeftinih višenedeljnih zimskih turističkih aranžmana koje turističke agencije nude starijim građanima. Tako se desilo da je bio u inostranstvu duže od tri meseca i tokom čitavog obračunskog perioda nije trošio električnu energiju. On spada i u one štedljive koji plaćaju po tarifi B. To je razlog što je sistem ispisao zahtev za uplatu od nula dinara, naravno, to nije čest slučaj pošto je veoma malo ljudi odsutno toliko dugo, a korisnici tarife B izgleda da su čvrsto vezani za zemlju. Da bi se javio problem, potrebno je da potrošač ispuni oba uslova.

Kad se jednom utvrdi, greška se lako otklanja:

```
145 IF RACUN = 0 THEN 100
```

tako da se preskoči naredba PRINT. Pretpostavlja se da se ovakav problem desio na jednom od prvih računara, ali ja se ne bih složio da je to samo priča. U svakom slučaju smatram da ovo lepo ilustruje kako greška može da bude neotkrivena skoro beskonačno.

Naravoučenije: kad izmišljate podatke za testiranje programa, nemojte ih uzimati slučajno. Izaberite vrednosti bliske i jednake vrednostima koje izazivaju grananje. Ako je naredba:

```
305 IF U<30 THEN 400
```

program testirajte vrednostima U=29,999, U=30 i U=30,001. Možda ste hteli da napišete:

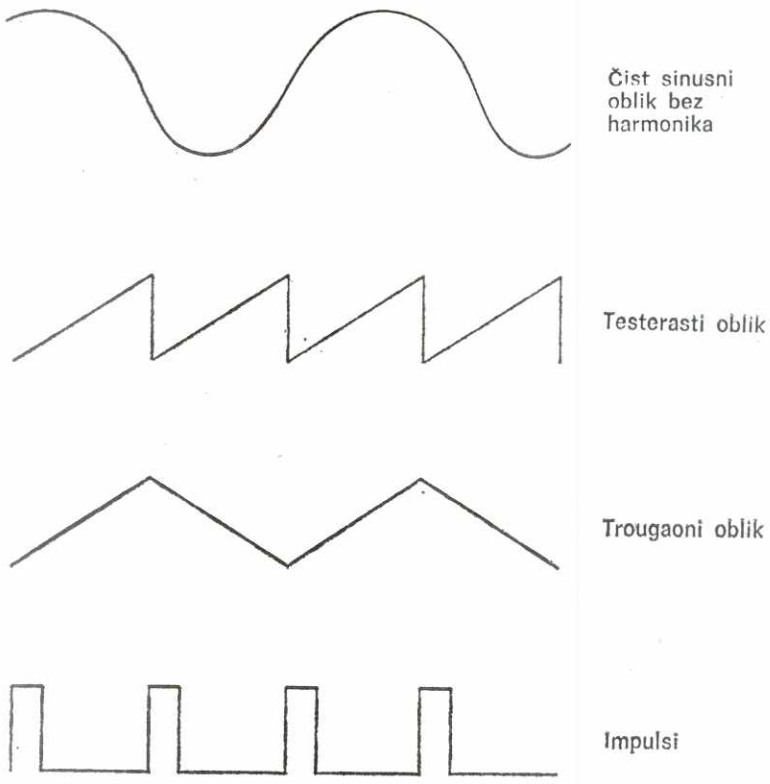
```
305 IF U<=30 THEN 400
```

Ako ispitujete samo sa vrednostima U=15 i U=160 nećete primetiti grešku. Podatke izaberite tako da se obavezno izvrši svaki deo programa. I naravno, treba da budete sigurni u to kakav odgovor treba da dobijete za svaki skup podataka.

„Commodore" tvrdi da skraćenica za zvučno integrisano kolo (SID) znači „Sound Interface Device" (uređaj za zvučni interfejs). Moja teorija je da je to pogrešno zapisana skraćenica za: Sidnejska operska kuća.

30 Zvuk i muzika

Već ste upoznali VIC koji, kao što smo videli, može da učini mnoge stvari sa ekranom. Sada ću vas upoznati sa integrisanim kolom SID koje, da ne preterujemo, može "šezdesetčetvorku" da pretvori u muzički instrument, pa čak i u podnošljiv sintisajzer.



Sl. 30.1 — Četiri osnovna talasna oblika

Reći kolu SID šta treba da radi slično je načinu programiranja kola VIC. To znači da SID ima niz registara u koje se naredbom POKE upisuju razne vrednosti. Svaki registar određuje neku od karakteristika zvuka koji se generiše.

Počnimo od razmišljanja o tome koje osobine muzičke note čine da ta nota bude karakteristična za određeni instrument. Npr. lako možemo da odredimo da li je nota C odsvirana na gitari ili orguljama. Zašto?

U svakoj noti nalazi se skup različitih učestanosti. Ne želim da se udubljujem u fizičke zakone po kojima nastaje muzika, ali ukratko, svaki put kad se odsvira nota, generiše se *osnovna* učestalost zajedno sa nekim *harmonicima* koji su umnošci osnovne učestalosti. Broj i relativna glasnost harmonika (u odnosu na osnovnu učestalost) daje noti boju karakterističnu za instrument. Ako ove učestalosti smatramo kombinacijom, rezultat je određeni talasni oblik koji bi na osciloskopu mogao izgledati kao na slici 30.1.

Međutim, ako kolu SID kažete samo kako da formira *učestalosti* u noti gitare, rezultat neće mnogo ličiti na gitaru. To je zbog toga što postoji još jedna karakteristika note koju treba razmotriti. Glasnost note se u toku sviranja menja. Kod orgulja, npr. zvuk počinje tiho pošto u cevi nema mnogo vazduha, a kasnije kad se nakupi dovoljna količina vazduha zvuk postaje jači. Kod gitare se dešava suprotno. Zvuk je najjači odmah posle puštanja žice, a zatim zvuk postepeno opada, ako ga gitarista namerno ne priguši. Saksofonista, kad jednom počne da svira notu, može da održava istu jačinu zvuka dogod ima vazduha u plućima. Takva analiza može da se uradi za sve instrumente.

Iz svega ovoga jasno je da ima više faza u formiranju pravog skupa glasnosti (ili anvelope) za neku notu.

SID prepoznaje četiri faze:

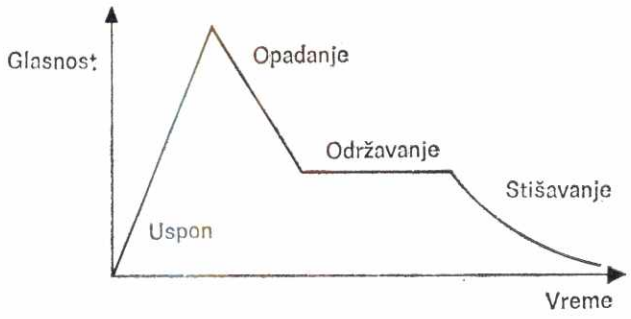
1. *Uspon* (attack): brzina kojom se postiže najveća jačina zvuka od početka sviranja note.

2. *Opadanje* (decay): brzina kojom jačina zvuka opada do nivoa održavanja.

3. *Održavanje* (sustain): nivo jačine koji se održava u toku sviranja note.

4. *Stišavanje* (release): brzina kojom se nota "gasi" kad prestane da se svira.

Sve ovo vam omogućava da uradite razne stvari. Međutim, kao i obično, što je sredstvo moćnije to pažljivije treba njime rukovati i više znati.



Sl. 30.2 — Četiri faze anvelope ADSR (ATTACK, DECAY, SUSTAIN, RELEASE)

Zbog toga ćemo početi sa veoma ograničenim ciljem, a zatim ćemo postepeno širiti horizonte.

Prvi posao je da samo formiramo notu, a zatim da povežemo note u melodiju. U ovom trenutku nije važno da li rezultat zvuči kao klavir ili kao orgulje.

Očigledno je da bi bilo pogodno razmišljati na (grubo) klasičan način (govoreći muzički). Da bismo se približili numeričkoj prirodi BASIC-a, definišimo svaku notu kao par brojeva (jedan za notu i jedan za oktavu). Srednja oktava neka bude oktava 0, a nota C neka bude nota 0. Tako ćemo srednju oktavu obeležiti kao u tabeli 30.1.

Tabela 30.1

Nota	Oktava	Muzička notacija
0	0	C
1	0	C# (ili Db)
2	0	D
3	0	D# (ili Eb)
4	0	E
5	0	F
6	0	F# (ili Gb)
7	0	G
8	0	G# (ili Ab)
9	0	A
10	0	A# (ili Hb)
11	0	H
0	1	C (jedna oktava više)

Prema ovim oznakama na raspolaganju su nam oktave od —4 (najniža) do 3 (najviša).

Sada nam je potrebna veza između ovog načina obeležavanja i učestalosti koje naredbom POKE treba upisati u odgovarajuće registre kola SID.

Intervali učestalosti u zapadnoj muzici zasnivaju se na *dijatonskoj* skali ("dijatonska" znači "u intervalima tonova"). Na žalost, odnosi učestalosti između uzastopnih nota nisu jednaki. Dobro je što su kompozitori u 18. veku otkrili da to znači da ne mogu da menjaju ključ, pa su pronašli hromatsku skalu koja je približna aproksimacija.

Sve ovo funkcioniše ovako: da bi se sa jedne note neke oktave prešlo na istu notu sledeće oktave, učestalost treba pomnožiti sa dva (to važi za obe skale). Pošto u oktavi ima dvanaest nota, da bi odnosi učestalosti uzastopnih nota bili jednaki, da bismo sa jedne note prešli na sledeću, treba učestalost pomnožiti sa 2 ↑ (1/12).

Sada treba reći da učestalost srednjeg C iznosi 4291 Hz. Znači, trebaće nam potprogram koji prihvata vrednost note i oktave (promenljive NTE i OCT) diže broj 4291 na neki stepen i vraća vrednost koju treba naredbom POKE upisati u registre kola SID. Registre? Zašto više od jednog registra? Pa zato što su učestalosti veće od 256, pa su za definisanje učestalosti potrebna dva bajta koja ću zvati H% (za viši bajt) i L% (za niži bajt). Pa, tu smo:

```
1000 F=4291*2↑OCT
1010 IF NTE>0 THEN F=F*2↑.0833333333: NTE=NTE—1: GOTO 1010
1020 FI=INT(F+0.5)
1030 H%=FI/256: L%=FI—256*H%
1040 RETURN
```

Znakovi % u nazivima promenljivih H% i L% mogu da vas zbune.

Postoje dva načina memorisanja brojeva u računaru: brojevi sa *pokretnim decimalnim zarezom* (floating point, kao što je, npr. broj 7,443) i *celi brojevi* (integer, kao broj 7). Celobrojne promenljive zauzimaju manje mesta u memoriji. Dodavanjem znaka % nazivu promenljive ili matrice računar se obaveštava da taj broj tretira kao ceo. Vodite računa o tome da nazivi moraju da budu isti u celom programu. Mašina smatra H i H% dvema različitim promenljivim.

A sada nešto o glavnom programu. Uvek zaboravljam adrese registara u kolu SID, pa ću one koje mi trebaju dodeliti promenljivima:

```
10 VOL=54296 : FH=54273 : FL=54272 : WHM=54276 : AD=54277:
   SR=54278
```

"VOL" određuje jačinu zvuka celog integrisanog kola a može da bude od 0 do 15. HF i FL su viši i niži bajt učestalosti, WFM je talasni oblik, a AD i SR su delovi AD (uspon i opadanje) i SR (održavanje i stišavanje) anvelope.

Nije baš pogodno vršiti proračune dok muzika svira, to bi suviše dugo trajalo. Zbog toga ćemo formirati matrice, izračunati potrebne vrednosti, memorisati ih u matricama i na kraju odsvirati.

```
20 DIM H% (200), L% (200), D (200)
```

H% i L% su vrednosti višeg i nižeg bajta učestalosti, a D je trajanje note. Ovim je dozvoljeno najviše 201 nota, ali vi naravno, možete dodeliti više prostora.

Pretpostavićemo da imamo listu nota definisanih naredbama DATA, pri čemu je svaka nota definisana mestom u okviru oktave, oktavom i trajanjem. Trajanje 0 biće uzeto kao kraj podataka.

Formirajmo petlju za učitavanje nota, izvršimo potrebne konverzije i napunimo matrice:

```
40 P=0
50 READ NTE, OCT, D
55 IF D=0 THEN 100
60 GOSUB 1000: REM PRIHVATANJE H%, L%
70 H%(P)=H%: L%(P)=L%: D(P)=D
80 P=P+1
90 GOTO 50
```

A sada da odsviramo ton. Prvo ćemo definisati tempo melodije koja će se svirati. Tempo se upisuje tako da može da se menja. Zatim ćemo podatke upisati u registre.

```
100 INPUT "TEMPO"; TE
102 INPUT "AD, SR"; A, S
104 POKE AD, A: POKE SR, S
106 INPUT "TALASNI OBLIK"; W
108 POKE WFM, W
110 POKE VOL, 15
```

Ovde je redosled događaja bitan. Posebno je važno da se u registar WFM vrednost upiše *posle* upisivanja vrednosti u AD i SR iz razloga koje ću navesti kasnije.

Sada samo da protrčimo kroz petlju koja će naredbom POKE upisati vrednosti u registre učestalosti i odložiti sviranje sledeće note dok na nju ne dođe red.

```
115 P=0
120 POKE FH, H%(P) : POKE FL, L%(P)
130 FOR N=1 TO D(P)*TE : NEXT
140 P=P+1
150 IF D(P)=0 THEN POKE VOL, 0: END
160 GOTO 120
```

Primetimo da kašnjenje u redu 130 zavisi od TE. Što je TE veće, to se ton sporije svira. Zbog toga nije važno koji su brojevi upotrebljeni za D dok između njih postoji odgovarajući odnos. Uvek možete da promenite TE tako da dobijete odgovarajuće trajanje. Primetimo i da smo na kraju promenljivoj VOL dodelili vrednost nula. Da to nismo učinili, poslednja nota bi se svirala do beskonačnosti.

Probajte sada program. Otkucajte:

```
2000 DATA 1, 0, 2
2010 DATA 0, 0, 0
```

a zatim RUN.

Probajte sa sledećim parametrima:

```
TEMPO = 200
AD, SR = 0, 240
TALASNI OBLIK = 17
```

Trebalo bi da dobijete C u osnovnoj oktavi sa zvukom sličnim orguljama. Ostavite red 2000 pa dodajte:

```
2010 DATA 1, 0, 2
2020 DATA 5, 0, 2
2030 DATA 5, 0, 2
2040 DATA 3, 0, 2
2050 DATA 1, 0, 2
2060 DATA 6, 0, 2
2070 DATA 6, 0, 2
2080 DATA 5, 0, 2
2090 DATA 3, 0, 2
2100 DATA 5, 0, 2
2110 DATA 8, 0, 2
2120 DATA 8, 0, 2
2130 DATA 7, 0, 2
2140 DATA 8, 0, 4
```

```
2150 DATA 5, 0, 2
2160 DATA 10, 0, 3
2170 DATA 8, 0, 1
2180 DATA 6, 0, 2
2190 DATA 5, 0, 2
2200 DATA 3, 0, 2
2210 DATA 1, 0, 2
2220 DATA 0, 0, 2
2230 DATA 5, 0, 2
2240 DATA 3, 0, 2
2250 DATA 1, 0, 2
2260 DATA 1, 0, 2
2270 DATA 0, 0, 2
2280 DATA 1, 0, 4
2290 DATA 0, 0, 0
```

Otkucajte RUN pa koristite vrednosti kao i ranije. Tema odgovara ako "šezdesetčetvorku" koristite na Božić.

Međutim, ja sam tu melodiju izabrao zbog toga što većina nota ima jednako trajanje i u istoj su oktavi, pa se može lako programirati. Mogli ste sve prepisati sa nota za pesmu "Dok su pastiri čuvali stada" (a to je bilo *to*), ali po sluhu i metodom pokušaja i grešaka. (Šta kažete sada?)

Vežba 1

Promenite listu DATA u programu tako da svira hor iz "Ne plači za mnom, Argentino".

HARMONIJA

Kolo SID je "mnogo pametnije" nego što sam ja to do sada pokazao. Za početak recimo da možemo da imamo više od jednog glasa (do tri, u stvari) i da se oni mogu ponašati potpuno nezavisno jedan od drugog.

Da bismo ovo ilustrovali, probajmo dvozvučnu harmoniju. Pošto će se u isto vreme svirati dve note, matrice M% i L% treba da imaju po dve kolone pa izmenimo red 20:

```
20 DIM H% (200,2), L% (200,2), D (200)
```

(Zbog jednostavnosti pretpostavićemo da obe note imaju isto trajanje). Sada će svaka naredba DATA imati pet vrednosti:

nota (glas 1), oktava (glas 1), trajanje, nota (glas 2), oktava (glas 2).

U redu 50 učitavamo prve tri vrednosti, pa ga ne menjamo kao ni redove 55 i 60, ali u redu 70 rezultat upisujemo u *prve* kolone matrica:

```
70 H% (P, 1)=H% : L% (P, 1)=L% : D(P)=D
```

Sada treba da dođemo do druge note i da i za nju ponovimo izračunavanje, a rezultat smestimo u drugu kolonu matrice:

```
75 READ NTE, OCT
76 GOSUB 1000
77 H% (P, 2)=H% : L% (P, 2)=L%
```

Svi registri kojima je ranije definisan glas 1 treba sada da se definišu i za glas 2. Pogodno je što svi registri za glas 2 imaju adrese za sedam veće od adresa odgovarajućih registara za glas 1. Uzgred, registri za glas 3 imaju adrese za 14 veće od adresa za glas 1. Kratak pregled registara kola SID dat je u dodatku 4.

Tako deo programa koji definiše vrednosti registara sada izgleda ovako:

```
100 INPUT "TEMPO"; TE
102 INPUT "AD, SR(1)"; A, S
104 POKE AD, A: POKE SR, S
105 INPUT "AD, SR(2)"; A, S: POKE AD+7, A: POKE SR+7, S
106 INPUT "TALASNI OBLIK(1)"; W1: POKE WFM, W1
108 INPUT "TALASNI OBLIK(2)"; W2: POKE WFM+7, W2
```

Red 120 postaje:

```
120 POKE FH, H% (P, 1) : POKE FL, L% (P, 1)
```

a potreban nam je još i red 125:

```
125 POKE FH+7, H% (P, 2) : POKE FL+7, L% (P, 2)
```

Ostali deo programa se ne menja. Naravno, menjaju se svi redovi DATA:

```
2000 DATA 5, 0, 1, 5, 0
2010 DATA 1, 0, 1, 5, 0
2020 DATA 3, 0, 1, 5, 0
2030 DATA 5, 0, 1, 5, 0
2040 DATA 8, 0, 1, 5, 0
2050 DATA 6, 0, 1, 5, 0
2060 DATA 6, 0, 1, 6, 0
2070 DATA 10, 0, 1, 6, 0
2080 DATA 8, 0, 1, 6, 0
2090 DATA 8, 0, 1, 8, 0
2100 DATA 1, 1, 1, 8, 0
2110 DATA 0, 1, 1, 8, 0
2120 DATA 1, 1, 1, 8, 0
2130 DATA 8, 0, 1, 8, 0
2140 DATA 5, 0, 1, 8, 0
2150 DATA 1, 0, 1, 6, 0
```

```
2160 DATA 3, 0, 1, 6, 0
2170 DATA 5, 0, 1, 6, 0
2180 DATA 6, 0, 1, 5, 0
2190 DATA 8, 0, 1, 5, 0
2200 DATA 10, 0, 1, 5, 0
2210 DATA 8, 0, 1, 5, 0
2220 DATA 6, 0, 1, 5, 0
2230 DATA 5, 0, 1, 5, 0
2240 DATA 3, 0, 1, 1, 0
2250 DATA 5, 0, 1, 1, 0
2260 DATA 1, 0, 1, 1, 0
2270 DATA 0, 0, 1, 0, 0
2280 DATA 1, 0, 1, 0, 0
2290 DATA 3, 0, 1, 0, 0
2300 DATA 8, —1, 1, 0, 0
2310 DATA 0, 0, 1, 0, 0
2320 DATA 3, 0, 1, 0, 0
2330 DATA 6, 0, 1, 0, 0
2340 DATA 5, 0, 1, 0, 0
2350 DATA 3, 0, 1, 0, 0
2360 DATA 5, 0, 1, 5, 0
2370 DATA 1, 0, 1, 5, 0
2380 DATA 3, 0, 1, 5, 0
2390 DATA 5, 0, 1, 5, 0
2400 DATA 8, 0, 1, 5, 0
2410 DATA 6, 0, 1, 5, 0
2420 DATA 6, 0, 1, 6, 0
2430 DATA 10, 0, 1, 6, 0
2440 DATA 8, 0, 1, 6, 0
2450 DATA 8, 0, 1, 8, 0
2460 DATA 1, 1, 1, 8, 0
2470 DATA 0, 1, 1, 8, 0
2480 DATA 1, 1, 1, 8, 0
2490 DATA 8, 0, 1, 8, 0
2500 DATA 5, 0, 1, 8, 0
2510 DATA 1, 0, 1, 6, 0
2520 DATA 3, 0, 1, 6, 0
2530 DATA 5, 0, 1, 6, 0
2540 DATA 10, —1, 1, 5, 0
2550 DATA 8, 0, 1, 5, 0
2560 DATA 6, 0, 1, 5, 0
2570 DATA 5, 0, 1, 5, 0
```

```
2580 DATA 3, 0, 1, 5, 0
2590 DATA 1, 0, 1, 5, 0
2600 DATA 8, —1, 1, 0, 0
2610 DATA 1, 0, 1, 0, 0
2620 DATA 0, 0, 1, 0, 0
2630 DATA 1, 0, 6, 1, 0
2640 DATA 0, 0, 0, 0, 0
```

Otkucajte RUN i izaberite iste parametre kao i ranije i to za oba glasa. Trebalo bi da čujete "Hrist, radost čovekove želje" od J.S. Baha (To nije Johan Sebastijan nego njegov malo poznati sin Jova Stević koji je note pisao tako da harmonija bude malo pogrešna).

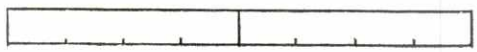
Da biste zvuk malo "razvukli" napišite red 75 ovako:

```
75 READ NTE, OCT : OCT=OCT—1
```

Sada je harmonija jednu oktavu ispod melodije. Zvuči sasvim dobro, zar ne?

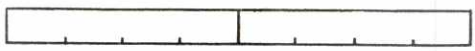
Zatim, izmenimo talasni oblik za melodiju (glas 1). Umesto 17 upišite 33. Razlika u karakteru zvuka je prilično uočljiva.

Do sada sam lenčario što se tiče vrednosti ADSR, ali sada već imamo dovoljno znanja da možemo i njima da upravljamo dovoljno pouzdano. Vrednosti uspona i opadanja drže se u dve polovine osmobitnog registra (vidi poglavlje 12) na ovaj način:



Što je veći broj u odgovarajućoj polovini registra to je veće trajanje uspona odnosno opadanja. Najduži uspon je 1111 (=15 decimalno). Pošto je to leva polovina bajta, ova vrednost je 16 puta veća (= 240). Prilično brz uspon treba da je 0011, a decimalni ekvivalent je 243.

Vrednosti održavanja i stišavanja organizovane su na sličan način:



Najveća vrednost perioda održavanja je 240 (decimalno).

Vrednosti, koje sam do sada koristio daju najbrži uspon, najbrže opadanje do najglasnije vrednosti održavanja, a na kraju je najbrže stišavanje. U stvari, nema nikakvog opadanja pošto vrednost održavanja kaže da se održava maksimalna jačina što se može potvrditi i slušanjem.

Ono što više začuđuje je činjenica da ne čujete da se note stišavaju. Postoji samo blag prelaz sa jedne na drugu notu. Razlog za to je što se nivo održavanja ne menja sve dok se bit 0 u kontrolnom registru talasnog oblika (WFM ne vrati na vrednost 0. Do sada smo koristili talasne oblike 17 (00010001 binarno) i 33 (00100001 binarno), a oba imaju bit 0 sa vrednošću 1. *To je razlog* zašto je važno da se u registar ADSR upiše vrednost pre upisivanja vrednosti u registar WFM. Kad se jednom postavi vrednost bit 0 u registru WFM, sa registrom ADSR možete

da radite šta god hoćete, ali neće biti nikakvog efekta. Sve to znači da je jedini način da otpočnete fazu stišavanja ciklusa taj da u WFM naredbom POKE upišete vrednost nula. Zatim treba naredbom POKE ponovo upisati željeni talasni oblik pre sviranja nove note. Dakle, da bismo ovo isprobali, ostavimo red za harmoniju onako kako jeste, a red 120 izmenimo u:

```
120 POKE WFM, W1 : POKE FH, H% (P, 1) : POKE FL, L% (P, 1)
```

Želimo da odgovarajuća harmonijska nota počne u istom trenutku, (ili bar onoliko blizu koliko je to moguće) tako da red 125 ne menjamo, ali sada nakon odgovarajućeg kašnjenja treba da isključimo melodiju:

```
127 FOR N=1 TO D (P) * MD : NEXT : POKE WFM, 0
```

Ovo, naravno, podrazumeva da je vrednost MD (kašnjenje) već obezbeđena. Pošto verovatno ne želimo da se zamajavamo posebno kašnjenjem i tempom, ima smisla da ih upišemo zajedno, pa tako red 100 postaje:

```
100 INPUT "TEMPO"; TE : INPUT "KASNJENJE"; MD
```

Kašnjenje u redu 130 je sada suviše veliko. Petlja bi trebalo da se izvrši TE — MD puta:

```
130 FOR N=1 TO D (P) * (TE — MD) : NEXT
```

Otkucajte RUN i pokušajte sa sledećim vrednostima:

TEMPO? 180

MELODIJA? 180 (tako da obe note jednako traju)

AD, SR (1)? 9, 0

AD, SR (2)? 0, 240

TALASNI OBLIK (1)? 33

TALASNI OBLIK (2)? 17

Sada imate klavsen koji svira glavnu melodiju, a u pratnji su orgulje (pa dobro, potrebno je i *malo* mašte).

DRUGI INSTRUMENT

Možete se igrati vrednostima ADSR tako da ih izabirate slučajno, pa da dobijete neke dobre početne rezultate. Međutim, ako želite da modelirate određeni instrument, potrebno je da na umu imate neke stvari.

Prvo, talasni oblik 17 (trougaoni) ima prilično čist ton koji podseća na orgulje, flautu ili zvono.

Talasni oblik 33 (testerasti) više zvuči kao žičani instrument. Zbog toga se i koristi za klavsen.

Ima još jedan talasni oblik o kojem uopšte nismo govorili. Njegova vrednost je 65, a daje *impulse* čija se širina može menjati upisivanjem vrednosti u registar od dva bajta. Za glas 1 niži bajt je 54274, a viši bajt 54275. Pokušajte ovo:

```
POKE 54274,255 : POKE  54275, 0
```

Pustite program ponovo, pa za talasni oblik 1 upišite 65, a za ADSR 1 upišite 9, 0.

To još više zvuči kao žičani instrument, zar ne? Priručnik tvrdi da je to klavir ali ja mislim da vam je potrebna "kingsajz" mašta da utvrdite *takvu* vezu. Međutim, gitara je još jedan žičani instrument. Da li možemo da odredimo kakva treba da bude anvelopa za gitaru?

Lako je ako malo razmislimo. Žica se otpusti, a odmah iza toga ton je najglasniji. Znači, uspon je brz (0). Sada nota postepeno postaje sve tiša, pa je opadanje prilično dugo, recimo 10. Tako je vrednost AD jednaka 0 * 16+10=10. Održavanja nema, a stišavanje može da bude brzo (nema nikakve razlike), pa je SR=0.

Probajte. Nadam se da ćete se složiti da zvuči zadovoljavajuće. Smanjite malo vrednost AD (6 ili 4). Utvrdićete da je nota, kao što ste i očekivali, "prigušena" i da je zvuk kao bendžo ili ukulele.

A kako se dobija zvono? Po izgledu anvelope liči na gitaru, pošto zvuk postepeno opada od najveće jačine koja se javlja odmah posle udarca. Međutim, zvuk je čistiji. Probajmo zato talasni oblik 17 sa AD=10 i SR=0. Rezultat liči na zvono, ali možda to nećete primetiti, ako ne izaberete veoma spori tempo da biste dozvolili zvuku da se ugasi.

Vežba 2

Program za Morzeov kôd iz poglavlja 18 samo ispisuje tačke i crtice. Bilo bi mnogo interesantnije da se osim toga daju i odgovarajući zvuci. Izmenite program tako da radi i to. (Koristite istu notu i za tačke i za crte, ali tako da crte traju tri puta duže od tačaka).

ODGOVORI

Vežba 1

To ćete postići sledećim naredbama DATA (moguće su i druge, ako izaberete drugi ključ).

```
2000 DATA 7,0,4,
2010 DATA 7,0,2
2020 DATA 7,0,2
2030 DATA 7,0,4
2040 DATA 8,0,2
2050 DATA 10,0,2
2060 DATA 0,1,2
2070 DATA 10,0,8
2080 DATA 10,0,2
```

```
2090 DATA 0,1,4
2100 DATA 0,1,2
2110 DATA 10,0,2
2120 DATA 3,1,6
2130 DATA 10,0,2
2140 DATA 8,0,4
2150 DATA 7,0,4
2160 DATA 7,0,3
2170 DATA 8,0,3
2180 DATA 10,0,2
2190 DATA 5,0,8
2200 DATA 5,0,3
2210 DATA 7,0,3
2220 DATA 8,0,3
2230 DATA 3,0,10
2240 DATA 3,0,2
2250 DATA 5,0,2
2260 DATA 3,0,2
2270 DATA 7,0,4
2280 DATA 10,0,6
2290 DATA 10,—1,2
2300 DATA 10,—1,2
2310 DATA 10,—1,2
2320 DATA 0,0,4
2330 DATA 3,0,6
2340 DATA 0,0,0
```

Ovde postoji mali problem. Pošto su neke od uzastopnih nota jednake, ne možete da čujete kraj jedne i početak druge note — samo jednu kontinualnu notu.

Kad pročitate deo o harmoniji, videćete način da svaku notu završite kad hoćete tako da zvuče odvojeno (treba da izmenite redove 120 i 130).

Vežba 2

Prvo inicijalizujte registre kola SID kao i obično:

```
5 VOL=54296 : FH=54273 : FL=54272 : WFM=54276:AD=54277 :
  SR=54278.
```

Sam program se ne menja uz dva izuzetka:

1. Dodajte redove 280, 290 i 295:

```
280 INPUT "UPISITE NOTU, OKTAVU"; NTE, OCT
290 GOSUB 1000
295 POKE FH,H%: POKE FL,L%: POKE VOL,15: POKE AD,0: POKE
    SR,240
```

Potprogram u redu 1000 će, naravno, biti onaj koji izračunava učestalost na osnovu ulaznih muzičkih veličina. Prepišite ga iz ovog poglavlja.

2. U redu 370 pozovite potprogram koji svira notu:

```
370 IF C>0 THEN PRINT CHR$ (C+64), A$ (C) : GOSUB 5000
```

Ovo koristi novi potprogram koji izgleda ovako:

```
5000 K$=A$(C)
5010 T=1
5020 L$=MID$(K$,T,1)
5030 IF L$="." THEN POKE WFM,17: FOR D=0 TO 100: NEXT D:
     POKE WFM,0
5040 IF L$="—" THEN POKE WFM,17: FOR D=0 TO 300: NEXT D:
     POKE WFM,0
5050 IF L$="▽" OR T=4 THEN RETURN
5060 T=T+1: GOTO 5020
```

Kad projektujete program često je mnogo lakše, ako izaberete pravi način za predstavljanje informacija u mašini. Ilustrujmo ovo programom koji igra „kružiće i krstiće".

31 Projektovanje programa

U poglavlju 14 videli smo program u kojem je bilo korisno da se u upotrebljenim potprogramima koristi pravilna struktura. A sada pogledajmo jedan program koji ima ne samo strukturisane potprograme, nego i strukturisane podatke.

Problem je u tome da se mašina programira tako da igra logičku igru "kružića i krstića" (naziv u Americi = ticktacktoe) protiv čoveka kao drugog igrača. Izabrao sam ovu igru zato što su pravila veoma jednostavna: igrači naizmenično upisuju 0 ili X na tablu 3×3, a pobeđuje prvi koji dobije tri znaka u redu. Lako je zastraniti pitajući se o strategiji igre i baciti se na projektovanje komplikovanih delova programa za izbor najboljih poteza, ali malo razmišljanja se kasnije u radu isplati. Veoma važan prvi zadatak je kako predstaviti tablu i stanje igre.

TABLA

Tipičan položaj na tabli je nešto kao:

X	O	
X		X
0		

Jedan od načina je da se polja na tabli numerišu ovako:

1	2	3
4	5	6
7	8	9

a zatim napravi lista onoga šta se na svakom polju nalazi:

1	X
2	O
3	—
4	X
5	—

6	X
7	O
8	—
9	—

Crtica znači da je polje prazno. Zgodno je da se simboli "X", "O" i "—" zamene numeričkim kodom:

zamenimo X sa 1
zamenimo O sa −1
zamenimo — sa 0

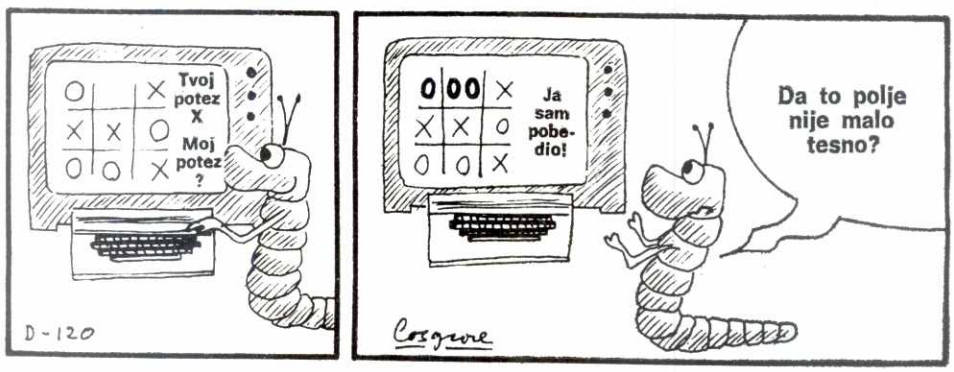
(Kasnije ćete videti zašto sam izabrao ove brojeve, a u ovom trenutku shvatite to kao rezultat osećaja za simetriju).

Tako se stanje na tabeli može prikazati *matricom* BD ovako:

	BD
1	1
2	−1
3	0
4	1
5	0
6	1
7	−1
8	0
9	0

Kad god se izmeni stanje na tabli, menjaju se odgovarajuće vrednosti u BD tako da, npr. BD (5) uvek sadrži stanje srednjeg polja tabele.

Možda se pitate zašto je ovo bolje od neke matrice BD$ koja bi sadržala znakove O, X i — kao znakovne nizove. Budite strpljivi, sve će se razjasniti.



PROCENA POLOŽAJA

Sledeći problem je: Kako naterati mašinu da vuče smislene poteze? Nije dobro ostaviti je da poteze bira slučajno, nego mora da blokira kad u redu ima dva znaka X, npr. (u ovom poglavlju ću podrazumevati da "šezdesetčetvorka" igra znakovima O). Zbog toga nam je potreban neki način za *procenu stanja* igre odnosno položaja na tabli, odnosno nešto što će nam pomoći da odmerimo moguće poteze.

Uzećemo drugu matricu koju ćemo zvati EV i koja će sadržati informacije o trenutnom stanju svakog reda, kolone i dijagonale. Ukupno ih ima osam:

```
7   4   5   6   8
↘   ↓   ↓   ↓   ↙
1→  1 | 2 | 3
    ---------
2→  4 | 5 | 6
    ---------
3→  7 | 8 | 9
```

Koristeći ovaj način obeležavanja redova, kolona i dijagonala, formiraćemo matricu EV ovako:

	EV		
1	0	= BD(1) + BD(2) + BD(3)	gornji red
2	2	= BD(4) + BD(5) + BD(6)	srednji red
3	—1	= BD(7) + BD(8) + BD(9)	donji red
4	1	= BD(1) + BD(4) + BD(7)	leva kolona
5	—1	= BD(2) + BD(5) + BD(8)	srednja kolona
6	1	= BD(3) + BD(6) + BD(9)	desna kolona
7	1	= BD(1) + BD(5) + BD(9)	leva dijagonala
8	—1	= BD(3) + BD(5) + BD(7)	desna dijagonala

Prikazane vrednosti EV odgovaraju ranije prikazanom položaju na tabli. Videćemo da nam one daju neke indicije za smislene poteze. Primetimo da je svaka vrednost u matrici EV samo *zbir* vrednosti BD za odgovarajući red, kolonu i dijagonalu. To je jedan od razloga zašto se položaj ne čuva u *znakovnoj* matrici. Ali zašto koristimo zbir?

Prvo, pogledajmo EV(2) čija je vrednost 2. To se može desiti samo ako u tom redu ima dva znaka X, a znakova O uopšte *nema*. Naravno, jer bi dva znaka X *i* jedan znak O dali zbir 1+1+(—1)=1 a ne 2. Vidi EV(4) gde se dešava upravo to. Znači, rezultat je sledeći: ako bilo gde u EV postoji vrednost +2, igrač X može da pobedi (ako je red na njega). Slično i vrednost —2 znači da igrač O može da pobedi. Vrednost —3 znači da je igrač O upravo pobedio, a vrednost +3 da je to učinio igrač X.

Preostaju još slučajevi kad su vrednosti EV jednake 1, —1 i O, ali ćemo videti da o njima ne treba brinuti pošto one ne odgovaraju "ključnim" tačkama u taktici igre.

STRUKTURA PROGRAMA

Opšta struktura programa će biti:

Funkcija	Prvi red programa
Inicijalizovanje table	1000
Prikaz table	2000
→ Potez čoveka	3000
Prikaz table	opet 2000
Ispitivanje da li je kraj	4000
Potez računara	5000
Prikaz table	opet 2000
Ispitivanje da li je kraj	opet 4000

Neke od ovih delova programa treba podeliti u manje delove. Npr. "ispitivanje da li je kraj" treba da generiše matricu EV, a zatim traži vrednosti 3 ili —3. Ako ih nema mora da utvrdi da li je igra završena nerešeno, tako što će proveriti da li u BD ima neka nula. Ako nema nule, svaki kvadrat je zauzet.

Ispitivanje da li je kraj

Procena stanja na tabli (6000)
Ako u EV postoji vrednost —3, pobedio je računar: END
Ako u EV postoji vrednost 3, pobedio je čovek: END
Ako u BD nema nula, partija je nerešena: END
RETURN

(Setite se da "šezdesetčetvorka" uvek igra O. Nije teško modifikovati ideju tako da se dozvoli da igra X ili O, po želji, ali to dovodi do komplikacija koje odvode od glavne teme razmatranja).

A sada da popričamo o potprogramu "potez računara". Da bi mogao da igra smisleno, računar mora da zna kakvo je stanje igre. Zbog toga će potprogram pozvati drugi potprogram — "procena stanja". Zatim u EV traži vrednost —2. Ako pronađe neku vrednost —2, on zna da u sledećem potezu može da pobedi i dalje ne treba da traži. Ako nema vrednosti —2, on traži vrednost 2, pošto ona predstavlja opasnost koju može blokirati. Izuzev, naravno, ako postoje *dve* vrednosti 2 u kojem slučaju X pobeđuje bez obzira šta O uradi (ovo nije baš sasvim tačno. *Moguće* je da se dva reda X-ova preklapaju tako da O može da blokira jednim potezom. Međutim, to može da se desi samo, ako je X već imao dva znaka u redu tako da je računar to blokirao, ili je već izgubio igru potez ranije. Drugim rečima, strategija računara obezbeđuje da se ova mogućnost u stvarnosti nikad ne dešava).

Ako nema pobedonosnog poteza i nikakve pretnje koju treba blokirati, računar igra *slučajan* potez (drugim rečima gleda samo jedan potez unapred. Mogu se formirati i strategije sa sagledavanjem više poteza, ali za ovu igru one su jedva potrebne).

Potez računara (5000)

Proračun table (6000)
Ako ima —2 u EV, završni udarac (7000): RETURN

Ako u EV ima dva puta 2, predaja: END
Ako u EV ima jedna dvojka, blokiranje (8000): RETURN
Igranje slučajnog poteza (9000)
RETURN

Sada smo izmislili još nekoliko potprograma na sledećem nivou. Proračun table je veoma prost i o njemu neću reći ništa sve dok ne napišem potprogram.

Završni udarac (7000)

Odigravanje slučajnog poteza (9000)
Proračun table (6000)
Ako EV sadrži —3, RETURN
Vraćanje poteza (10000)

Ideja je u tome da, kad smo već *stigli* do završnog udarca, *mora* da postoji i pobedonosni potez. *Mogli* bismo da napišemo potprogram koji bi ga pronašao direktno, ali je jednostavnije, a u praksi isto toliko brzo, samo odigrati slučajan potez i ponoviti proračun table. Ako u EV postoji vrednost —3, računar je pobedio. Ako vrednosti —3 nema, potez je bio pogrešan. Zbog toga "vraćamo potez" (odnosno menjamo BD u prethodno stanje i pokušavamo ponovo).

"Blokiranje" je veoma slično.

Blokiranje (8000)

Odigravanje slučajnog poteza (9000)
Proračun table (6000)
Ako EV ne sadrži 2, RETURN
Vraćanje poteza (10000)

Svi ostali potprogrami pišu se direktno bez pozivanja daljih potprograma. Eto sada već možemo da napišemo nešto u BASIC-u.

PROGRAM

Glavni program proizlazi sam po sebi:

```
5 DIM BD(9):DIM EV(8)
10 GOSUB 1000: REM INICIJALIZACIJA TABLE
20 GOSUB 2000: REM PRIKAZ
30 GOSUB 3000: REM POTEZ
40 GOSUB 2000: REM PRIKAZ
50 GOSUB 4000: REM ISPITIVANJE KRAJA
60 GOSUB 5000: REM POTEZ RACUNARA
70 GOSUB 2000: REM PRIKAZ
80 GOSUB 4000: REM ISPITIVANJE KRAJA
90 GOTO 30
```

Potprogram za *inicijalizovanje* je jednostavan:

```
1000 FOR P=1 TO 9
1010 BD(P)=0
1020 NEXT P
1030 RETURN
```

Možda ćete pitati: "Zašto sve to radimo kad naredba DIM ionako sve vrednosti postavlja na nulu?" To je sigurno dobro pitanje. Međutim, postoji i odgovor: *bezbednost*. Među vežbama postoje i neke koje obuhvataju igranje više partija zaredom, a između partija treba BD vratiti na nulu. To se lako zaboravlja kad se bakće sa programom, pa je dobro uobičajiti da se promenljive inicijalizuju, *a da se ne* pretpostavlja ništa o prethodnim vrednostima.

Evo potprograma za *prikazivanje:*

```
2000 FOR P=1 TO 9
2010 IF BD(P)=1 THEN PRINT "X";
2020 IF BD(P)=-1 THEN PRINT "O";
2030 IF BD(P)=0 THEN PRINT " ";
2040 IF INT(P/3)=P/3 THEN PRINT
2050 NEXT P
2060 PRINT:PRINT
2070 RETURN
```

I ovo zaslužuje komentar. Matrica BD se pretražuje i ispituje da li je vrednost 1, —1 ili 0, pa se te vrednosti pretvaraju u X, O i tačke. Pošto želimo da prva tri znaka budu u istom redu, na kraju svake naredbe PRINT nalazi se tačka-zarez (;) da bi se sprečilo prelaženje u novi red. Međutim, kad je P=3, 6 ili 9, potrebno je da pređemo u novi red, jer bi inače izgled table bio ovakav:

```
XO.X.XO.,
```

a ne:

```
X O .
X . X
O . .
```

To se postiže redom 2040. Ako je P=3 onda je INT (P/3)=1 a to je jednako P/3. Slično važi i za P=6 i P=9. Međutim, ako je, recimo, P=4 tada je INT (P/3)=1 a P/3=1,3333. Samo kad je P umnožak od 3, P/3 i INT (P/3) su jednaki (vidi poglavlje 22). Tako nove redove imamo samo kada je P jednako 3, 6 ili 9.

Naravno, mogao sam napisati i kukavički:

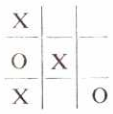
```
2040 IF P=3 OR P=6 OR P=9 THEN PRINT
```

ali je trik sa naredbom INT mnogo šire primenljiv.

Možda ste primetili da je potprogram malo primitivan. Dobićete ovakav prikaz:

```
X . .
O X .
O . O
```

a ne:



a prikaz će biti priljubljen uz levi rub ekrana.

Kad vam ceo program bude radio zadovoljavajuće, možda ćete hteti da zamenite ovaj potprogram nekim koji daje lepši prikaz. Mnogo će vam pomoći neki od PET-grafičkih znakova. Evo to je jedna od lepih mogućnosti koje daje strukturisanje programa u potprograme. Iz programa možete da izvadite potprogram i zamenite ga drugim (poboljšanom ili alternativnom verzijom) i možete da budete (prilično) sigurni da neće biti dramatičnih uticaja na neki drugi deo programa pošto nema naredbi GOTO koje utiču na više od jednog potprograma (naravno, još uvek morate da pazite na to da su nazivi promenljivih usaglašeni).

Upisivanje poteza čoveka-igrača je lako:

```
3000 INPUT "UPISITE POTEZ"; P
3010 IF BD(P)<>0 THEN PRINT "TO POLJE JE ZAUZETO": GOTO
     3000
3020 BD(P)=1
3030 RETURN
```

Potez se upisuje kao broj od 1 do 9. Prvo se ispituje odgovarajući element BD. Ako on nije nula, tamo već ima nečeg pa dajemo takvu poruku i zahtevamo važeći potez. Kad je sve u redu element BD dobija vrednost 1 (= znak X) pošto je to potez koji vuče čovek.

Ispitivanje da li je kraj ima oblik skeleta navedenog ranije:

```
4000 GOSUB 6000
4010 FOR P=1 TO 8
4020 IF EV(P)=-3 THEN PRINT "JA SAM POBEDIO": END
4030 IF EV(P)=3 THEN PRINT "VI STE POBEDILI": END
4040 NEXT P
4050 FOR P=1 TO 9
4060 IF BD(P)=0 THEN RETURN
4070 NEXT P
4080 PRINT "TO JE PRAVI POTEZ": END
```

Isto važi i za potprogram *"potez računara"*:

```
5000 GOSUB 6000
5010 MT=0
5020 FOR P=1 TO 8
5030 IF EV(P)=-2 THEN GOSUB 7000: RETURN
5040 IF EV(P)=2 THEN MT=MT+1
5050 NEXT P
5060 IF MT=2 THEN PRINT "UREDU, PAMETNJAKOVICU"!: END
5070 IF MT=1 THEN GOSUB 8000: RETURN
5080 GOSUB 9000
5090 RETURN
```

I *proračun table* je lak:

```
6000 EV(1)=BD(1)+BD(2)+BD(3)
6010 EV(2)=BD(4)+BD(5)+BD(6)
6020 EV(3)=BD(7)+BD(8)+BD(9)
6030 EV(4)=BD(1)+BD(4)+BD(7)
6040 EV(5)=BD(2)+BD(5)+BD(8)
6050 EV(6)=BD(3)+BD(6)+BD(9)
6060 EV(7)=BD(1)+BD(5)+BD(9)
6070 EV(8)=BD(3)+BD(5)+BD(7)
6080 RETURN
```

Očigledno je da ovde postoje mogućnosti da se proračun vrši u petljama, ili da se koriste pogodne liste DATA, ali stvarno nije vredno truda.

O *završnom udarcu* ne treba mnogo reći:

```
7000 GOSUB 9000
7010 GOSUB 6000
7020 FOR P=1 TO 8
7030 IF EV(P)=-3 THEN RETURN
7040 NEXT P
7050 GOSUB 10000
7060 GOTO 7000
```

Blokiranje je prilično slično izuzev što koristi indikator F da utvrdi da li je još uvek ostala vrednost 2:

```
8000 GOSUB 9000
8010 GOSUB 6000
8020 F=0
8030 FOR P=1 TO 8
8040 IF EV(P)=2 THEN F=1
8050 NEXT P
```

```
8060 IF F=0 THEN RETURN
8070 GOSUB 10000
8080 GOTO 8000
```

Primetimo da je naredba RETURN u ovom potprogramu u redu 8060. Pošto znamo da blokirajući potez postoji, računar će ga pronaći, ostaviti u indikatoru vrednost 0 i naredbom RETURN izaći iz potprograma kao što treba.

Evo potprograma za *odigravanje slučajnog poteza:*

```
9000 CM=INT(RND(1)*9)+1
9010 IF BD(CM)<>0 THEN 9000
9020 BD(CM)=-1
9030 RETURN
```

Zašto slučajan potez? Zašto ne samo menjati P od 1 do 9? Mogli biste, ali bi to učinilo da odgovor računara bude apsolutno predvidljiv. U igri je to najbolje izbegavati.

Vraćanje poteza je najtrivijalniji mogući potprogram:

```
10000 BD(CM)=0
10010 RETURN
```

Eto to je to! Program se napisao skoro sam od sebe — nakon što je najteži deo posla obavljen prilikom definisanja strukture. Ono što je moglo da bude komplikovana zbrka nerazumljivih hijeroglifa svelo se samo na šačicu kratkih i jasnih potprograma i jednog još jasnijeg dela koji ih povezuje. Tako ste dobili prilično inteligentnog igrača "kružića i krstića".

Međutim, program *može* da se poboljša (svaki program se može poboljšati i to je razlog zašto skoro svaki program i skoro svaki računar ima, recimo, verziju 3.7 ili nešto slično. Originalna verzija se poboljšava pošto problem shvatite u potpunosti — kao i zašto vaš odgovor nije ovako dobar kako bi mogao da bude — tek kad sa njim izvesno vreme radite). Već sam predložio da se poboljša prikaz na ekranu. Evo još nešto što treba pokušati:

Vežba 1

Igrač O (računar) uvek je na početku u lošijem položaju, jer igra drugi (može se pokazati da je u *ovoj* igri onaj koji igra drugi uvek u lošijem položaju; u nekim igrama to nije slučaj). Zbog toga njegova odbrana treba da bude što bolja. Posebno treba obratiti pažnju na to da on uvek zauzme srednje polje (broj 5) ako može. Znači, potprogram "potez računara" treba izmeniti tako da to radi (uz uslov da nema pretnje ili dobitka u poziciji) ako je moguće. Uradite to.

Vežba 2

Sada se može igrati samo jedna partija, a zatim treba ponovo da otkucate RUN. Izmenite program tako da se igra više partija, a da se rezultat pamti.

Vežba 3

Izmenite program tako da X i O naizmenično povlače prve poteze. Ako pobedi O, u sledećoj partiji X povlači prvi potez i obratno.

ODGOVORI

Vežba 1

Kako stvari sada stoje, potprogram "potez računara" (5000), kad utvrdi da nema nikakav potez koji pobeđuje i da ne mora da blokira, povlači slučajan potez u:

```
5080 GOSUB 9000
```

Znači, neposredno *pre* toga treba ubaciti ispitivanje da li je kvadrat 5 slobodan, pa ako jeste, igrati taj potez:

```
5075 IF BD(5)=0 THEN BD(5)=—1: RETURN
```

Vežba 2

Biće vam potrebne dve nove promenljive XSCRE i OSCRE (i DR za brojanje nerešenih rezultata, ako želite da prikažete i njih) koje na početku treba da imaju vrednost 0. U isto vreme treba postaviti indikator "kraj partije" (EOG):

```
4 XSCRE=0 : OSCRE=0 : DR=0 : EOG=0
```

(Važi ista primedba o bezbednosti. Mogli biste izostaviti inicijalizovanje ovih promenljivih ali. . .).

Sada je potrebno da izmenite sve naredbe END. Ako je pobedio računar, OSCRE se povećava za 1, a EOG se postavlja na vrednost 1. Ako je pobedio čovek, XSCRE se povećava za 1, a EOG dobija vrednost 1. Ako je rezultat nerešen, DR se povećava, a u EOG se upisuje 1.

Tako redovi 4020, 4030 i 4080 postaju:

```
4020 IF EV(P)=—3 THEN PRINT "JA SAM POBEDIO":OSCRE=OSCRE
     +1:EOG=1:RETURN
4030 IF EV(P)=3 THEN PRINT "VI STE POBEDILI": XSCRE=XSCRE+
     1:EOG=1:RETURN
4080 PRINT "TO JE PRAVI POTEZ":DR=DR+1:EOG=1:RETURN
```

U stvari, red 4030 se nikad ne izvršava (ni u originalu ni u ovoj izmeni). Zašto?

Postoji još jedno mesto gde partija može da se završi, u potprogramu "blokiranje" gde računar shvata da je izgubio i predaje se (ne baš elegantno). Red 5060 izmenite u:

```
5060 IF MT=2 THEN PRINT "U REDU, PAMETNJAKOVICU":
     XSCRE=XSCR+1:EOG=1:RETURN
```

Sada ispitajmo EOG u glavnom programu:

```
55 IF EOG=1 THEN GOSUB 11000
85 IF EOG=1 THEN GOSUB 11000 : GOSUB 2000
```

a zatim na 11000 napišite potprogram koji pita da li igrač želi da igra drugu partiju, pa ako ne želi ispisuje konačni rezultat:

```
11000 INPUT "PONOVNA IGRA (DA/NE)▽";Q$
11010 IF Q$="DA" THEN EOG=0:GOSUB 1000:RETURN

11020 PRINT "KONACNI REZULTAT"
11030 PRINT "▽X▽:▽▽";XSCRE;"▽O▽: ";OSCRE;"▽POTEZA▽:▽";
      DR
11040 END
```

Ne zaboravite da u redu 1010 indikator kraja partije vratite na nulu i da ponovo inicijalizujete tablu. . .

Ako isprobate program, utvrdićete da postoji mala (pa . . . prilično mala) greška. Ako pobedi X, završni izgled table prikazuje se dva puta.

Razlog za to je što sam bio lenj da napišem još jedan deo programa. *Svi* načini završavanja partije treba da se obrađuju u potprogramu u redu 4000. Ovakav program kakav je sada, ima dodatno obrađivanje kraja u redu 5060. Trebalo je da se tu postavi indikator koji bi se ispitivao u potprogramu u redu 4000. Međutim, kad sam vas već uvukao u ovo zamešateljstvo, najbrži način da se iz toga izvučemo je da se onemogući *prikazivanje* ako je EOG=1:

```
2000 IF EOG=1 THEN RETURN
2005 FOR P=1 TO 9
```

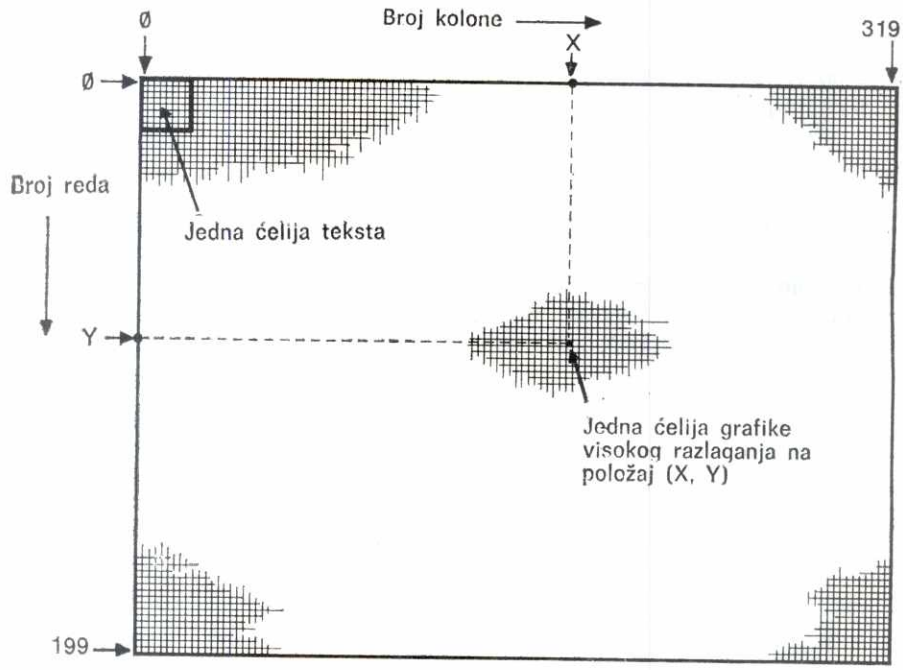
Vežba 3

Možda ste primetili da se to već rešilo. Pošto X i O naizmenično povlače poteze, a nova partija ulazi u petlju tamo gde je prethodna izašla, pa ako je poslednji potez odigrao X, sledeći potez će povući O. To znači da počeci niza nerešenih partija *jesu* naizmenični (pošto svaka nerešena partija ima 9 poteza), pa O počinje, ako je poslednju partiju dobio X i obratno.

Mada u Priručniku nije pomenuto, na „šezdesetčetvorki" možete koristiti finu grafiku, kao i grubu PET-grafiku. Ona zahteva nešto rada na računaru, ali se isplati.

32 Grafika u visokom razlaganju

Svaki znak koji se prikazuje na ekranu sastoji se od kvadrata sa 8×8 ćelija ili tačkica (engleski pixel) koje služe za formiranje znaka (hardverski). Direktnim pristupom tim ćelijama moguće je ostvariti na ekranu grafiku sa visokim razlaganjem (Hi-res grafiku). To znači da možete prikazati $25 \times 8 = 200$ redova i $40 \times 8 = 320$ kolona. To je skoro isti brojni sistem koji se koristi i za sprajtove, ali ograničen isključivo na područje ekrana. Vidi sl. 32.1.



Sl. 32.1 — Struktura ekrana u visokom razlaganju

NAČIN RADA SA GRAFIKOM VISOKOG RAZLAGANJA

Pošto je Hi-res grafika teška za početnike izložiću vam tri korisna potprograma koje možete kopirati i koristiti u BASIC programima. Oni omogućuju:

1. Uključivanje u Hi-res način rada i određivanje u memoriji ekranske memorijske zone.
2. Crtanje tačke na izabranom mestu.
3. Crtanje duži između dve zadate tačke.

Da bi se računar doveo u Hi-res način rada potrebno je postaviti peti bit sa adrese 53265 na jedinicu (pogledaj *Vodič*, strana 123). Ovo se može realizovati naredbom:

```
POKE 53265,PEEK(53265) OR 32
```

koja radi kako je to objašnjeno u poglavlju 12. Takođe, neophodno je rezervisati prostor u memoriji za podatke koji će biti prikazani na ekranu, obrisati ovaj deo memorije i utvrditi boju ekrana. Sledeći potprogram obezbeđuje područje u memoriji, počev od adrese 8192, koje će biti dodeljeno ekranu.

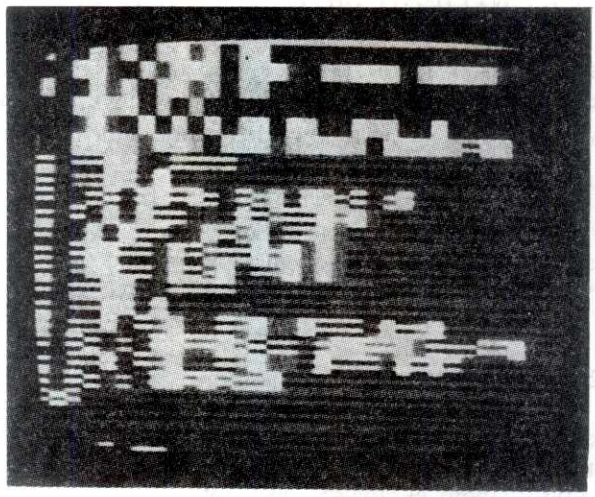
```
11000 REM HI-RES INICIJALIZACIJA
11010 POKE 53265,PEEK(53265) OR 32
11020 POKE 53272,PEEK(53272) OR 8
11030 BM=8192
11040 FOR U=BM TO BM+7999
11050 POKE U,0
11060 NEXT U
11070 FOR U=1024 TO 2023
11080 POKE U,13
11090 NEXT U
11100 RETURN
```

Otkucajte RUN. Najpre ćete dobiti zbrkan ekran, zatim se ekran briše u uglavnom crnu pozadinu sa nekim šarenim mrljama na mestima gde je postojao tekst. Nakon toga se ekran prazni i postaje svetlo zelen (promenite broj 13 u redu 11080 na vrednost 16*BOJA—CRTEZA + BOJA—OSNOVE, pri čemu su BOJA—CRTEZA i BOJA—OSNOVE kodovi boja koje želite). Ovaj program daje crni crtež na svetlozelenoj osnovi.

Napomenimo da je brisanje ekrana prilično sporo: oko 20 sekundi u okviru BASIC programa. Za slučaj mašinskog programa ovo je gotovo trenutno, ali to je druga priča!

CRTANJE

Hi-res kolona i red definišu na ekranu koordinatni sistem, kao što je to prikazano na slici 32.1. Jedan od osnovnih zadataka je pisanje potprograma koji crta jednu tačku (pixel) na poziciji (X, Y), pri čemu X predstavlja red, a Y kolonu. Korišćenjem ovog potprograma moguće je crtati prave, krive ili čitave slike.



Sl. 32.2 — Pri prvom postavljanju HI-res ekrana on ima izgled kao na slici. Ekran mora biti obrisan pre nego što se koristi za grafiku

Sledeći potprogram omogućuje crtanje jedne tačke u redu X i koloni Y. Podrazumeva se da je Y brojna vrednost između 0 i 199, a X između 0 i 320.

```
12000 REM PLOT X,Y
12010 BY=BM + 320*INT(Y/8) + 8*INT(X/8) + (Y AND 7)
12020 BT=7 — (X AND 7)
12030 POKE BY,PEEK(BY) OR (2↑BT)
12040 RETURN
```

Vežba 1

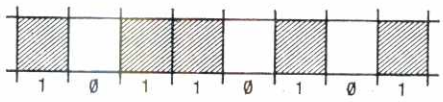
Obezbedite da ovaj program prihvata isključivo ispravne vrednosti za koordinate X i Y.

KAKO OVO RADI?

Ovaj odeljak razmatra tehničke aspekte, tako da ga možete preskočiti ukoliko to želite, a eventualno se na njega vratiti kasnije.

Svaki bajt u memoriji ekrana sadrži podatke za 8×1 red tačaka u visokom razlaganju. Binarna vrednost 0 znači "tačka se ne crta", a vrednost 1 znači "tačka se crta". Tako, na primer, bajt 10110101 ima značenje kao što je to prikazano na slici 32.3.

Kad sistemsku promenljivu na adresi 53265 izmenite tako da pređete na grafiku sa visokim razlaganjem, operativni sistem nalaže računaru da interpretira podatke na pomenuti način. Ovo se naziva *grafik definisan bitovima.*



Sl. 32.3 — Tačka definisana bitovima

Adrese naše ekranske memorije odgovaraju pozicijama ekrana kako je to prikazano u tabeli 32.1

Tabela 32.1

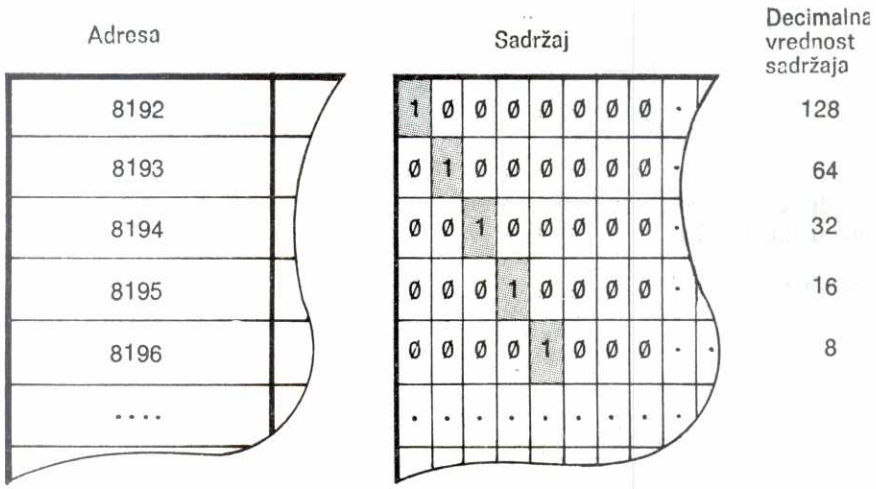
	0	1	2	. . .	39	← broj kolone niskog razlaganja
0	8192	8200	8208	. . .	8504	
1	8193	8201	8209	. . .	8505	
2	8194	8202	8210	. . .	8506	
3	8195	8203	8211	. . .	8507	
4	8196	8204	8212	. . .	8508	0 broj reda niskog razlaganja
5	8197	8205	8213	. . .	8509	
6	8198	8206	8214	. . .	8510	
7	8199	8207	8215	. . .	8511	
Broj reda visokog razlaganja 8	8512	8520	. . .	. . .	. . .	
9	8513	8521	. . .	. . .	. . .	
10	8514	8522	. . .	. . .	. . .	
11	8515	8523	. . .	. . .	. . .	1
12	8516	8524	. . .	. . .	. . .	
13	8517	8525	. . .	. . .	. . .	
14	8518	8526	. . .	. . .	. . .	
15	8519	8527	. . .	. . .	. . .	
.	. . .	. . .	. . .	. . .	. . .	
.	. . .	. . .	. . .	. . .	. . .	
.	. . .	. . .	. . .	. . .	. . .	

Tako svakoj ćeliji znaka kojoj kada se koristi u niskom razlaganju odgovara *jedna* adresa u ekranskoj memoriji, sada odgovara osam adresa, odnosno blok-memorije od osam bajtova. Blokovi su raspoređeni u istom redosledu kao i ćelije u ekranskoj memoriji niskog razlaganja. Prvo se ide po redovima, a u novi red se prelazi posle kolone 39.

Pretpostavimo da želite da nacrtate duž po dijagonali, počev od levog gornjeg ugla ekrana i da je ona dužine 5 tačaka. Adrese i sadržaji imaju oblik kao na slici 32.4.

Postavljeni problem se može rešiti sledećim programom:

```
10 GOSUB 11000  [upiši potprogram za grafiku visokog razlaganja]
20 POKE 8192,128
30 POKE 8193,64
```



Sl. 32.4 — Grupisanje tačaka za dobijanje dijagonalne linije

```
40 POKE 8194,32
50 POKE 8195,16
60 POKE 8196,8
70 GOTO 70
```

Isprobajte pa ćete se uveriti.

Isti pristup se može koristiti i u opštem slučaju:

1. Pronađite odgovarajuću adresu.

2. Naredbom POKE upišite potrebnu vrednost da biste dobili željeni prikaz na ekranu.

Pošto ne želite da obrišete ekran, moraćemo da pretpostavimo da adresa može da sadrži i vrednost različitu od nule. To zahteva da koristite operator OR za postojeći sadržaj i novu vrednost, kao u poglavlju 12.

Red 12010 izračunava tačnu adresu.

Red 12020 izračunava vrednost koju treba upisati naredbom POKE da bi se nacrtala nova tačka.

Red 12030 vrši operaciju OR nad postojećim sadržajem i upisuje dobijeni rezultat.

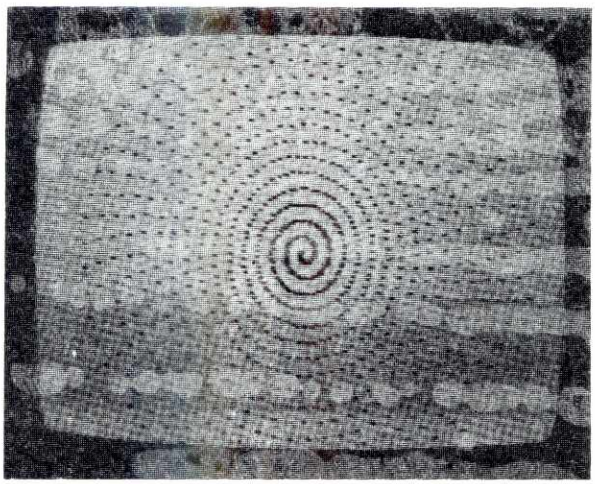
Za više detalja vidite *Vodič*, strana 125.

SPIRALA

Evo jednog primera koji ilustruje upotrebu pomenutih potprograma:

```
10 GOSUB 11000
20 FOR T=1 TO 1000
30 X=160+T*SIN(T/10)/10
```

```
40 Y=100+T*COS(T/10)/10
50 GOSUB 12000
60 NEXT T
70 GOTO 70
```



Sl. 32.5 — Spirala nacrtana pomoću hiljadu bitova

Nemojte zaboraviti dva ranije pomenuta potprograma. U ovom slučaju korišćene su trigonometrijske funkcije SIN i COS za crtanje spirale. Nakon dobijanja grafika pritisnite tastere RUN/STOP + RESTORE.

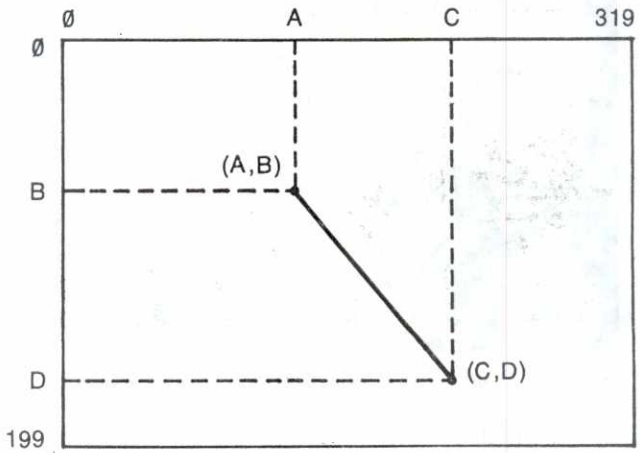
DUŽ

Sledeći program crta duž čiji krajevi imaju koordinate (A, B) i (C, D) kao na slici 32.6.

Ne ulazeći mnogo u detalje dolazimo do rešenja:

```
13000 REM DUZ (A,B)—(C,D)
13010 IF A>C THEN A0=C: C=A: A=A0: B0=D: D=B: B=B0
13020 CA=C—A: DB=D—B
13030 IF CA=0 THEN 13100
13040 IF DB<=CA AND DB>=—CA THEN 13200
13050 S=SGN(DB)
13060 FOR V=0 TO DB STEP S
13070 X=V*CA/DB+A: Y=V+B: GOSUB 12000
13080 NEXT V
13090 RETURN
13100 IF DB=0 THEN RETURN
13110 S=SGN(DB)
```

```
13120 FOR V=B TO D STEP S
13130 X=A: Y=V: GOSUB 12000
13140 NEXT V
13150 RETURN
13200 FOR V=0 TO CA
13210 Y=V*DB/CA+B: X=V+A: GOSUB 12000
13220 NEXT V
13230 RETURN
```



Sl. 32.6 — Crtanje duži između dve zadate tačke

SGN u redovima 13050 i 13110 predstavlja *sign* funkciju. SGN(J) ima vrednost 1 ukoliko je J>0, 0 ukoliko je J=0, a —1 ako je J<0. Ostali deo programa određuje koordinate.

ZVEZDASTA ŠARA

Ovaj program crta zvezdasto raspoređene duži.

```
10 GOSUB 11000
20 FOR T=1 TO 30
30 A=160: B=100
40 C=160+90*COS(PI*T/15)
50 D=100+90*SIN(PI*T/15)
60 GOSUB 13000
70 NEXT T
80 GOTO 80
```

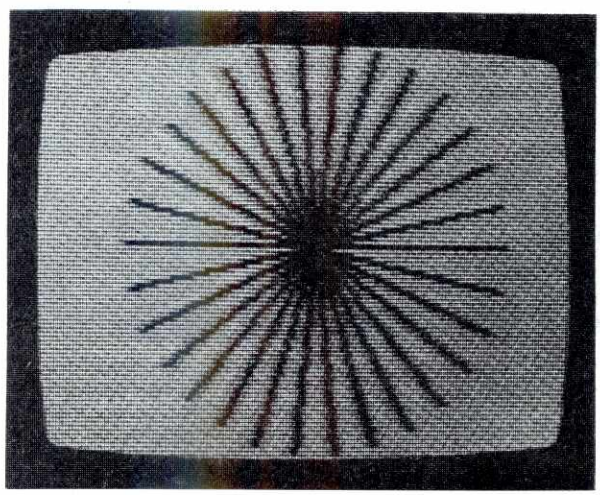
Ako vrednost 30 u redu 20 zamenite sa 20 ili 40 dobićete manje ili više radijalnih linija. Ne zaboravite da otkucate i standardne potprograme u redovima 11000, 12000 i 13000.

POLIGONI

Evo još jednog programa:

```
10 INPUT "UPISI BROJ STRANICA";N
20 GOSUB 11000
30 FOR T=0 TO N-1
40 U=T+1
50 A=160+90*COS(2*PI*T/N)
60 B=100+90*SIN(2*PI*T/N)
70 C=160+90*COS(2*PI*U/N)
80 D=100+90*SIN(2*PI*U/N)
90 GOSUB 13000
100 NEXT T
110 GOTO 110
```

Pokušajte sa N=5,6,7, . . . itd. Svaki put pritisnite tastere RUN/STOP + RESTORE.



Sl. 32.7 — Zvezdasta šara

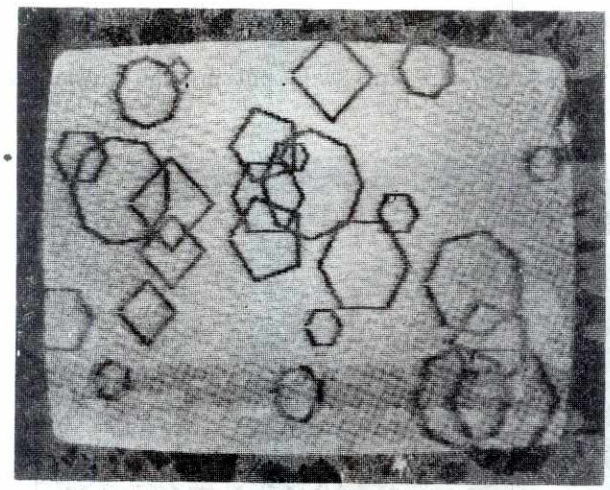
RAČUNARSKA UMETNOST

Na kraju, evo programa koji crta slučajno raspoređene poligone, sve dok mu pritiskom na taster S ne kažete da prestane sa radom.

```
10 GOSUB 11000
20 CX=30+260*RND(0): CY=30+140*RND(0)
30 RAD=20*RND(0)+5
50 N=INT(4+6*RND(0))
```

```
 60 FOR T=0 TO N-1
 70 U=T+1
 80 A=CX+RAD*COS(2*PI*T/N)
 90 B=CY+RAD*SIN(2*PI*T/N)
100 C=CX+RAD*COS(2*PI*U/N)
110 D=CY+RAD*SIN(2*PI*U/N)
120 GOSUB 13000
130 NEXT T
140 GET A$: IF A$<>"S" THEN 20
150 GOTO 150
```

Još uvek će vam trebati RUN/STOP + RESTORE za zaustavljanje programa.



Sl. 32.8 — Umetnost računara korišćenjem slučajno nacrtanih poligona

Vežba 2

Napišite program koji crta pravougaonik ako su dati brojevi kolona XL i XR za levu i desnu stranu, kao i dva broja YT i YB za gornju i donju stranu.

ODGOVORI

Vežba 1

Dodajte redove:

```
12002 IF X<0 OR X>319 THEN FLAG=1: RETURN
12004 IF Y<0 OR Y>199 THEN FLAG=1: RETURN
```

Sada u glavnom programu dodajte redove koji koriste indikator (tzv. flag), ispisuju poruku greške (recimo) i redefinišu X i Y, kao na primer:

```
764 IF FLAG=1 THEN PRINT "OOOPA! IZADJE IZ EKRANA":
    FLAG=0: GOTO
```

(na red u kome se zadaju vrednosti za X i Y) itd. u zavisnosti od samog programa.

Vežba 2

```
 10 INPUT "LEVA KOLONA"; XL
 20 INPUT "DESNA KOLONA"; XR
 30 INPUT "GORNJI RED"; YT
 40 INPUT "DONJI RED"; YB
 50 GOSUB 11000
 60 A=XL: B=YT: C=XR: D=YT: GOSUB 13000
 70 A=XR: B=YB: C=XR: D=YT: GOSUB 13000
 80 A=XR: B=YB: C=XL: D=YB: GOSUB 13000
 90 A=XL: B=YT: C=XL: D=YB: GOSUB 13000
100 GOTO 100
```

Nisam se mučio oko zaštite INPUT-a, ako hoćete ona se veoma lako dodaje po potrebi.

Ponekad brojevi koji izgledaju jednako nisu jednaki.

33 Pronalaženje i otklanjanje grešaka VI

GREŠKE U ZAOKRUŽIVANJU

Greške koje smo do sada razmatrali su naše greške i bilo ih je lako otkloniti kad smo ih pronašli. Postoji i druga vrsta grešaka koja proizlazi iz same konstrukcije mašine. Nije u pitanju konstrukciona greška, nego posledica načina organizacije svih računara. Stvar je u preciznosti kojom računari pamte brojeve. Ako razmislimo o bilo kojem uobičajenom načinu "držanja" brojeva, očigledno je da je broj cifara koje se "drže" ograničen. Na primer, brojač pređenih kilometara u automobilu može imati samo 5 cifara pošto ima samo 5 "prozora". Isti je slučaj i sa računarom. Svaki broj može da zauzme najviše onoliko koliko ima "prozora". Međutim, svaki "prozor" ne predstavlja decimalnu cifru. Interni mašinski kôd za brojeve je sasvim različit od načina na koji mi zamišljamo brojeve, a ja vas neću zamarati svim tim detaljima. Činjenica je da postoji netačnost koja proizlazi iz tog načina prikazivanja brojeva, a neophodna konverzija brojeva iz tog načina prikazivanja u način jasan korisniku znači da spoljni prikaz broja (onako kako je prikazan na ekranu) može da ne bude isto što i unutrašnji prikaz. Kao primer za ovo o čemu pričam navešću logaritme koji se uče u srednjoj školi. Ako pomoću logaritama pomnožite 2 i 2 dobićete:

Broj	logaritam
2	0,3010
2	0,3010
3,999	0,6020

odnosno $2 \times 2 = 3{,}999$

Kombinacija činjenica da su ovi logaritmi tačni samo do četiri cifre (tj. zauzimaju samo 4 "prozora") i da dolazi do konvertovanja (broj u logaritam, a zatim logaritam u broj) dovodi do netačnosti.

Evo programa koji izaziva istu vrstu problema:

```
10 FOR P=1 TO 10
20 S=SQR(P)
```

```
30 Q=S↑2
40 IF P<>Q THEN PRINT P,Q
50 NEXT P
```

Uzmimo slučaj kad je P=9. U redu 20 vadi se koren tako da je S=3. Zatim se u redu 30 promenljiva S diže na kvadrat, tako da je Q=9 odnosno jednako P. Naravno, P će uvek biti jednako Q, pošto korenovanje, pa kvadriranje, dovodi do onoga od čega smo počeli. Znači, red 40 nema smisla: pošto se P nikad ne razlikuje od Q, na ekranu se neće ispisati ništa. Da li je tako? Pustite program da se izvršava. Dobićete:

```
3          3
5          5.00000001
6          6.00000001
7          7.00000001
9          9.00000001
10         10
```

To je stvarno veoma čudan rezultat, pošto mašina ne samo što ispisuje vrednosti za koje tvrdi da su različite, nego ih ispisuje i kad su jednake. Desilo se da su komplikovani matematički postupci doveli do malih netačnosti u unutrašnjem predstavljanju brojeva, što se odrazilo kao razlika između P i Q. Međutim, postoje i netačnosti u prevođenju internog oblika u decimalni prikaz na ekranu, tako da oni *izgledaju* identični iako mašina tvrdoglavo tvrdi da nisu. Za neke vrednosti interni kodovi *jesu* jednaki (za 8, npr.). Ovaj tip greške može izuzetno da zbuni pa je ponekad jedini izlaz da se u naredbi IF dozvoli mala greška tako da ćemo imati:

```
IF ABS(P—Q) < 0.0001 THEN ...
```

Funkcija ABS je neophodna, pošto Q može da bude veće od P. U tom slučaju bi rezultat bio manji od nule, pa samim tim manji od 0,0001, iako njegova apsolutna vrednost može biti i velika (—30 je manje od 0,0001). Funkcija ABS "odseca" znak minus.

STEPENOVANJE

Greške zaokruživanja su posebno uobičajene ako koristite:

```
T↑N
```

Za N-ti stepen broja T. Probajte:

```
10 FOR T=1 TO 10
20 PRINT T*T, T↑2
30 NEXT T
```

Nešto čudno se desilo za T=7 i 9, odnosno došlo je do greške zaokruživanja. Ako želite, recimo, peti stepen *celog* broja T, tačnije je ako koristite:

```
T*T*T*T*T
```

a ne:

```
T ↑ 5
```

U stvari, T ↑ N se izračunava kao:

```
EXP (N * LOG (T))
```

(za one kojima su stepenovanje i logaritmovanje bliski).

Na primer, pretpostavimo da tražite celobrojna rešenja A, B, C jednačine: A2+B2=C2

Mogli biste da probate nešto kao ovo:

```
10 FOR A=1 TO 10
20 FOR B=1 TO 10
30 C=SQR(A↑2+B↑2)
40 IF C=INT(C) THEN PRINT A,B,C
50 NEXT B
60 NEXT A
```

Pokušaj neslavno propada bez rešenja, iako su brojevi 3, 4 i 5, (na primer) rešenje u opsegu u kojem se rešenja traže. Krivac je greška zaokruživanja. Greška zbog ↑ može se izbeći ako se A ↑ 2 i B ↑ 2 zamene sa A * A i B * B. Naredba SQR izaziva više problema, ali jedan izlaz je:

```
10 FOR A=1 TO 10
20 FOR B=1 TO 10
30 C=INT(SQR(A*A+B*B))
40 IF A*A+B*B=C*C THEN PRINT A,B,C
50 NEXT B
60 NEXT A
```

Ovo funkcioniše, delimično i zbog toga što, izgleda, SQR daje nešto veće vrednosti. Zbog veće sigurnosti dodajte:

```
45 IF A*A+B*B=(C+1) * (C+1) THEN PRINT A, B, C+1
```

u slučaju da INT vršimo nad brojem nešto malo manjim od celog broja.

(Ovo je izuzetno neefikasan pristup samom problemu, gledano matematički, ali je to toliko uobičajena greška da zaslužuje da se spomene).

Celobrojne promenljive A%, B% i C% mogu takođe da poboljšaju izglede na uspeh, ali ne za greške koje daju *manju* vrednost.

Iako memorija „šezdesetčetvorke" izgleda prilično velika za većinu zadataka, nije uvek i dovoljna. Da bi se obradile velike količine podataka, ili male količine, ali u specijalne svrhe, potreban je i dodatni tip memorije.

34 Datoteke

Jedna od stvari u kojima su računari posebno dobri je memorisanje i ispitivanje velikog broja podataka. Kako oni to rade? Na kraju krajeva, "šezdesetčetvorkina" memorija ima samo oko 39000 bajtova (znakova), a to nije mnogo ako želite da memorišete dela Čarlsa Dikensa, ili da formirate katalog Britanskog muzeja.

Ova strana ima oko 600 reči, a svaka reč sadrži u proseku 5 znakova (uključujući i razmake). Znači, "šezdesetčetvorka" bi mogla da "zapamti" oko 13 strana ove knjige. Još nismo oduzeli deo memorije koji zauzima program za analizu tih podataka, tako da stvar stoji još gore.

Postoji i drugi problem. Podaci su u memoriji samo dok je mašina uključena, tako da bi u "šezdesetčetvorku" mogli da upišemo samo 13 strana "Travničke hronike" (ili neke druge knjige), s tim da je nikad ne isključujemo.

Ovo je očigledno smešno. Mora da postoji bolji način. Potrebna nam je *nepromenljiva* memorija (tj. ona koja ne gubi svoj sadržaj kad nema dovoda električne energije). Već smo se upoznali sa jednim takvim sistemom za memorisanje: kasetofonom. Jedina razlika je u tome što sad želimo da ga koristimo za memorisanje podataka, dok smo ga do sada koristili za memorisanje programa.

DATOTEKE NA KASETAMA

Skup podataka koji se na traci čuva kao jedinstvena celina zove se *datoteka*. Ako želite možete i program smatrati datotekom. Kao što programima dodeljujemo nazive, tako isto nazivamo i datoteke.

Program upisujemo na traku ovako:

```
SAVE "FRED"
```

Ovo ima dva efekta. Prvo se na traku upisuje naziv datoteke "FRED", a zatim se upisuje program.

I kad na datoteku na traci želite da upišete podatke, potrebna su ista ta dva koraka. Drugim rečima, prvo morate da date naredbu koja datoteci daje naziv, a zatim drugu koja upisuje podatke.

Naziv dajete datoteci naredbom kao što je:

OPEN 1, 1, 1, "FRED"

Deo "1, 1, 1" treba malo objasniti. Prvo, "1" je broj datoteke, (ili broj kanala) i za njega možete uzeti vrednost kakvu god želite. Ja sam uzeo "1", jer je to prva datoteka koju sam naveo. Drugo, "1" definiše uređaj koji se koristi. Za kasetofon to je "1", za štampač "4", a za disk "8" itd. Ove vrednosti se ne mogu menjati. One su fiksno definisane u sistemu. Treće, "1" kaže sistemu da li želimo da pišemo na uređaj ili da čitamo sa njega. (Ova vrednost ima različita značenja za razne uređaje. Pošto ovde govorimo samo o kasetofonu, neću da komplikujem navodeći značenja za ostale periferijske jedinice.) Znači, *samo* za kasete, opšti oblik naredbe OPEN je:

```
OPEN   CH,   1,   U/I,   "NAZIV"
       ↑     ↑    ↑        ↑
       |     |    |        └──────── naziv datoteke
       |     |    ├───────────────── 0: ulazna datoteka (čita se)
       |     |    └───────────────── 1: izlazna datoteka (piše se)
       |     └────────────────────── definiše kasetofon
       └──────────────────────────── proizvoljni broj kanala
```

U redu. Isprobajte sada ovo:

```
10 CH=5
20 OPEN CH,1,1,"FRED"
30 FOR N=1 TO 10
40 PRINT#CH,N
50 PRINT N
60 NEXT N
70 CLOSE CH
```

Red 20 dodeljuje datoteci naziv "FRED" i pridružuje joj izlazni kanal 5. Red 40 ispisuje brojeve od 1 do 10 u kanal, a time i na datoteku (FRED) koja je tom kanalu pridružena. Primetimo oblik ove naredbe. Ona izgleda kao obična naredba PRINT izuzev što ima "# CH", čime usmerava izlaz na kanal 5. Sledeći red (50) ispisuje brojeve na ekran. Posle kraja petlje datoteka se *zatvara* naredbom CLOSE (red 70). Primetimo da se naziv datoteke spominje samo u naredbi "OPEN". U svim drugim naredbama navodi se samo broj kanala.

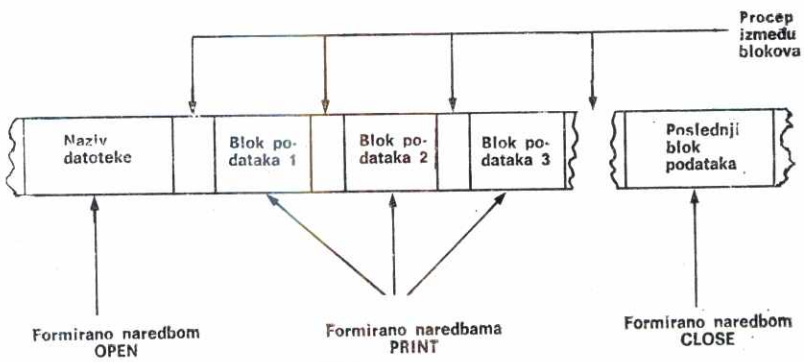
Kad pustite da se program izvršava, dobićete poruku:

PRESS RECORD & PLAY ON TAPE

Proverite da li ste stavili traku i pritisnite (jednovremeno) tastere RECORD i PLAY. Pažljivo posmatrajte ekran. Videćete kako se on briše, a traka automatski počinje da se kreće. U toku tog perioda mašina izvršava red 20 i ispisuje naziv datoteke u "blok zaglavlja" na traci. Nakon toga na ekranu se za kratko vreme ponovo pojavi prikaz i ispisuju se brojevi od 1 do 10. Primetićete da u tom kratkom periodu traka miruje. Ekran se ponovo briše, traka se pokreće i na kraju se na ekranu ponovo pojavljuju znakovi, a traka se zaustavlja. Ako sada razmislite o tome shvatićete da je čudno. Naredbe za izlaz na traku izvršavaju se naizmenično sa naredbama za ispisivanje na ekranu, pa biste mogli očekivati da se traka kreće *dok* se brojevi prikazuju na ekranu. Razlog za ovo čudno ponašanje je u tome što mašina ne posluša odmah kad joj kažete da na datoteku nešto upiše. Ona to privremeno smešta u deo memorije koji se zove *bafer* dok se ne nakupi dovoljno podataka da se isplati prenos na traku (tj. dok se bafer ne napuni). Brojevi od 1 do 10 ne pune bafer do kraja. Zato, ako ne uradite nešto, na traku neće biti upisan nijedan podatak. Eto čemu služi naredba CLOSE u redu 70. Ona "šezdesetčetvorki" saopštava da se na datoteku pridruženu kanalu CH neće više ništa pisati i da treba upisati bafer na traku ma šta bilo u njemu u tom trenutku. To se zove "pražnjenje bafera".

Izmenite red 30 u:

30 FOR N=1 TO 200



Sl. 34.1 — Struktura datoteke na traci

i ponovo otkucajte RUN. Sada ćete jasno videti organizaciju bafera pošto ima više bafera podataka za prenos, a između prenosa blokova na ekranu ćete moći da vidite prikaz.

Znači, traka izgleda kao na sl. 34.1.

Za vas kao korisnika nijedan od ovih detalja nije važan, dok imate na umu da uvek posle završetka izlaza na traku izdate naredbu CLOSE. Zapamtite da ne očekujete da se traka pokrene odmah iza svake naredbe PRINT. Međutim, ako znate šta se stvarno događa, manje ćete grešiti ili — u najgorem slučaju — moći ćete da objasnite grešku kad je napravite (to malo liči na upravljanje automobilom. *Ne morate* da znate kako funkcioniše kvačilo kad menjate brzine, ali ako znate onda vam je jasno zašto se isplati menjanje brzina i zašto se čuje užasno struganje, ako ne pritisnete papučicu kvačila do kraja).

ČITANJE DATOTEKE

Da vidimo sada kako možemo doći do podataka na traci. Dodajte ove redove i otkucajte RUN:

```
80 END
100 OPEN CH, 1, 0, "FRED"      (otvaranje kanala 5 za čitanje sa datoteke "FRED")
110 INPUT #CH, A
120 PRINT A
130 IF A<>200 THEN 110
```

Premotajte traku na početak i otkucajte GOTO 100. (*Nemojte* upisivati RUN 100, jer ćete izgubiti sadržaj promenljive CH.) Mašina će vas obavestiti da treba da pritisnete taster PLAY na kasetofonu, pa će zatim tražiti zaglavlje "FRED" na traci. Kad ga pronađe, odmah će dobiti zahtev da pročita podatak sa trake (red 110), pa će odmah učitati i drugi blok. Kad ispiše prvi blok podataka, pročitaće sledeći blok, itd. sve dok ne dođe do broja 200 gde se program završava.

Sugerisao sam vam da u memoriji imate i program za pisanje i program za čitanje, tako da možete lako da eksperimentišete i da lakše prevaziđete početne probleme. Na primer, možda ćete utvrditi da ste pokušali da pišete po vodećoj plastičnoj traci, ako ste zaboravili da traku namotate do magnetnog sloja. Ako je taj slučaj, niste upisali blok zaglavlja pa program za čitanje neće moći da ga nađe, znači, moraćete da ponovite postupak.

PRIMENA: ROĐENDANI

To stvarno fascinira, ali da li je i korisno? Svakako, i to na više načina. Na primer, pretpostavimo da želite da računar izvrši neke komplikovane matematičke proračune koji traju satima. Očigledno je da ne želite da sedite pored računara i čekate da se pojavi rešenje, ali ukoliko nemate štampač, to ćete morati da učinite (bar u slučaju da postoji verovatnoća da vrednosti "otkližu" sa ekrana zbog toga što ima suviše mnogo izlaza da bi stalo na jedan ekran). Alternativa je da rezultate upišete u datoteku koju možete da pročitate kad vam je zgodno. Čak i ako *imate* štampač ova tehnika može biti pogodna kad noću izvršavate poduži program. Može biti veoma neugodno kad štampač svakih 20 minuta između ponoći i 6 sati glasno ispisuje po jedan red.

Uobičajenija upotreba datoteka je organizovano memorisanje alfabetskih i numeričkih podataka. Na primer, možda hoćete da imate na računaru adrese svih prijatelja i poslovnih partnera — neku vrstu kompjuterizovanog adresara, ako vam se više sviđa taj izraz. Očigledno je da morate da memorišete ime, adresu i telefonski broj, ali bilo bi dobro da imate još neke informacije kao, na primer: da li određenom licu šaljete čestitku ili poklon za rođendan ili za Novu godinu. Ako šaljete rođendanske poklone, bilo bi dobro da na datoteci imate i datume rođenja. Na taj način mogli biste da napišete programe za pretraživanje datoteke koji bi mogli da utvrde kome treba poslati čestitku ili poklon u martu, na primer. Rođendane ne treba zaboravljati. Ali o tome kasnije. Naš prvi posao je *formiranje* datoteke. Za svako lice imaćemo sledeće podatke:

Ime
Adresa
Datum rođenja
Podaci o čestitkama i poklonima

Ovaj skup podataka zovemo *slog* (da bi stvari bile jednostavnije, zaboravimo telefonski broj). Svaku stavku sloga (npr. adresu) zovemo *polje*.

Prva dva polja nisu problem. Za datum rođenja treba definisati standardni oblik i pridržavati se toga. Na kraju, nije baš zgodno reći računaru da traži NOV, ako su meseci zabeleženi kao brojevi (u tom slučaju treba da traži 11).

Najlakše je koristiti brojeve tako da:

020960

znači:

02 drugi
09 septembar (9. mesec)
60 1960 (izostavimo 19)

Potreban nam je i pogodan način da kodujemo podatke o čestitkama i poklonima. Postoji mnogo načina da se to uradi. Svaku mogućnost obeležimo brojem:

rođendanska čestitka	1
rođendanski poklon	2
novogodišnja čestitka	3
novogodišnji poklon	4

tako da je, ako nekome šaljete rođendansku i novogodišnju čestitku (bez poklona), kôd jednak 13.

Ovakav program formira datoteku:

```
5 CH=1:CAS=1:IN=0:OUT=1
10 OPEN CH,CAS,OUT,"MLIST"
20 INPUT "IME";N$
30 PRINT#CH,N$
40 INPUT "ADRESA";AD$
50 PRINT#CH,AD$
60 INPUT "DATUM RODJENJA";BD$
70 PRINT#CH,BD$
80 INPUT "KODOVI CESTITKI";G$
90 PRINT#CH,G$
100 INPUT "IMA LI JOS PODATAKA (DA/NE)";Q$
110 IF Q$="DA" THEN 20
```

```
115 PRINT#CH,"****"
120 CLOSE CAS
130 END
```

Treba primetiti neke sitnice uprkos očiglednoj jednostavnosti programa. Prvo, zvuči primamljivo da se unesu *svi* podaci, a zatim upišu:

```
PRINT #CH, N$, AD$, BD$, G$
```

Problem je ovde što PRINT# daje u datoteci *baš isti* oblik kao naredba PRINT na ekranu. Znači, ako je N$="PERA",AD $="BEOGRAD HILENDARSKA 8", BD$="110737", a G$="3", u datoteci ćete imati:

```
PERA BEOGRAD HILENDARSKA 8 110737 3
```

(i možda još neke razmake). Ako to pokušate da pročitate naredbom:

```
INPUT#CH, N$, AD$, BD$, G$
```

neće biti dobro. *Ceo* slog će biti učitan u N$, pošto nema zareza koji odvajaju polja, a što naredba INPUT očekuje.

Mogli biste i namerno da upišete zareze:

```
PRINT# CH, N$; ","; AD$; ","; BD$; ","; G$
```

ali ja mislim da je to zbunjujuće i više volim da koristim posebne naredbe PRINT#.

Drugo, iz istog razloga, u adresama ne sme biti zareza.

Treće, primećujete, iako su poslednja dva polja brojevi, da sam ih tretirao kao znakovne nizove. Kasnije ćete videti zašto.

Četvrto, pogledajte red 115. On upisuje granični podatak (* * * *) na kraju datoteke u polje "ime" sloga koji dolazi iza poslednjeg sloga. To nam je potrebno da bismo utvrdili da li je pročitana čitava datoteka kad je pretražujemo.

Konačno, pogledajmo red 5. Koristio sam ga da navedem skraćenice umesto brojeva (za rad sa datotekama). Na primer, sad mogu da napišem "OUT" umesto "1" u naredbi OPEN, da bi naznačio da želim da pišem u datoteku. Smatram da se naredba "Open a channel to cassette for output" (engl. otvori kanal ka kaseti za izlaz) mnogo bolje ogleda u "OPEN CH,CAS,OUT" nego u "OPEN 1, 1, 1".

Vežba 1

Kako sada stvari stoje, korisnik mora da pamti kodove za čestitke i poklone. Izmenite red 80 u:

```
80×GOSUB 1000
```

i u redu 1000 napišite potprogram koji korisniku postavlja niz pitanja kao:

```
DA LI SALJETE RODJENDANSKU CESTITKU?
```

i automatski formira G$.

PRETRAŽIVANJE DATOTEKE

Možda bismo hteli da dobijemo odgovor na pitanje: "Čijih rođendana treba da se setim u junu i šta treba da pošaljem?". Pa napišimo program koji će to raditi. On će ukazati i na to kako da ga kasnije više generalizujemo.

Prvo treba da otvorimo datoteku:

```
5 CH=1 : CAS=1 : IN=0 : OUT=1
10 OPEN CH, CAS, IN, "MLIST"
```

a zatim pročitamo slog:

```
20 INPUT#CH,N$
25 IF N$="****" THEN END
30 INPUT#CH,AD$
40 INPUT#CH,BD$
50 INPUT#CH,G$
```

Sada pogledajmo koji nas slog interesuje:

```
60 IF MID$ (BD$, 3, 2) = "06" THEN GOSUB 500: REM PRONADJEN
```

Drugim rečima ispitujemo da li je mesec rođenja juni. U ovom redu koristim MID$ (vidi poglavlje 17) da izdvojim dva srednja znaka BD$ i uporedim sa "06". Zbog toga se datum rođenja upisuje u datoteku kao *znakovni niz*. Mnogo je lakše izdvojiti znakove iz znakovnog niza, nego nekoliko cifara iz broja. A znakovni niz je uvek moguće pretvoriti u broj naredbom VAL (poglavlje 17), ako nam je to potrebno. Ako smo pronašli nekog ko je rođen u junu, prelazimo na potprogram u redu 500, ali o njemu ćemo kasnije.

Pročitajmo sledeći slog:

```
70 GOTO 20
```

POTPROGRAM "SLOG JE PRONAĐEN"

A sada o potprogramu. Potrebno je da pregledamo G$, da utvrdimo da li ima jedinica (čestitka), ili dvojka (poklon), ili oboje. Evo kako:

```
500 FLAG=0
510 FOR P=1 TO 4
520 IF MID$(G$,P,1)="1" THEN FLAG=1:PRINT "CESTITKA"
530 IF MID$(G$,P,1)="2" THEN FLAG=1:PRINT "POKLON"
540 NEXT P
550 IF FLAG=0 THEN RETURN
560 PRINT
570 PRINT N$
```

```
580 PRINT AD$
590 PRINT:PRINT
600 RETURN
```

Pretražujemo G$ tražeći "1" ili "2". Ako nađemo "1", ispisujemo reč CESTITKA, a ako nađemo "2", reč POKLON. U oba slučaja *indikator* (FLAG) dobija vrednost 1. To nam omogućava da posle završetka petlje znamo da ne treba slati čestitku ili poklon, ako je indikator jednak nuli, pa nema smisla ispisivati ni ime i adresu Zbog toga je i napisan red 550. Međutim, ako je vrednost indikatora 1, nešto šaljemo (čestitku ili poklon), pa program ispisuje ime i adresu pre izlaska iz potprograma.

Vežba 2

Izmenite program tako da se može koristiti za bilo koji mesec.

Vežba 3

Napišite program koji obrađuje novogodišnje poklone i čestitke i daje informaciju o tome koliko čestitki treba da kupite.

PROMENA IZLAZNE DATOTEKE

Normalno kad izdate naredbu PRINT, sistem smatra da to znači da izlaz hoćete da šaljete na ekran. Kao što smo videli, izmenom oblika naredbe PRINT možemo da usmerimo podatke na bilo koji uređaj. Međutim, do sada još nisam objasnio da se podaci mogu preusmeriti čak i da se naredbe PRINT ne menjaju.

To se postiže naredbom CMD čiji je oblik:

```
CMD kanal
```

Ako napišem:

```
CMD 5
```

to izlaz, umesto na ekran, usmerava na kanal 5. Naravno, kanal 5 treba prethodno da bude definisan naredbom OPEN kao na primer:

```
OPEN 5, CAS, OUT, "EKRAN"        (pretpostavljamo da su CAS i OUT
                                  jednaki 1 kao i ranije)
```

Sada se svi podaci, koji bi inače išli na ekran, upisuju na kasetu. Jedini izuzeci su poruke greške koje i dalje izlaze na ekranu.

Da bi se sistem vratio u uobičajeno stanje (tj. da izlaz bude na ekran), prvo treba ispisati prazan red (odnosno samo novi red):

```
PRINT# 5
```

a zatim zatvoriti datoteku:

CLOSE 5

Ovde postoji jedna mala nelogičnost. Moglo bi se očekivati da se može napisati PRINT, a ne PRINT# 5, pošto su podaci ionako usmereni na kanal 5. To, međutim, ne pomaže. Ne pitajte mene zašto.

Vežba 4

Setimo se praćenja programa iz poglavlja "Pronalaženje i otklanjanje grešaka IV". Postoji način da koristite praćenje da biste dobili profil programa.

Evo kako se to radi: napišite program koji uključuje naredbu za praćenje kao u poglavlju "Pronalaženje i otklanjanje grešaka IV", ali rezultat praćenja uputite na kasetu.Ne koristite znakove "<" i ">" oko broja reda. Oni nisu potrebni pošto rezultat praćenja i normalan izlaz ne idu na isto mesto, a osim toga to bi malo iskomplikovalo sledeću fazu.

U sledećoj fazi se napiše program koji traži da se upiše broj reda, a zatim čita upravo formiranu datoteku i broji koliko puta se traženi broj reda pojavljuje i na kraju ispisuje broj pojavljivanja tako da znate koliko puta se ta naredba izvršila.

ODGOVORI

Vežba 1

```
1000 G$=" ":REM G$ JE NA POCETKU PRAZAN NIZ
1010 INPUT "SALJETE LI RODJENDANSKU CESTITKU (DA/NE)▽";
     Q$
1020 IF Q$="DA" THEN G$=G$+"1"
1030 INPUT "SALJETE LI RODJENDANSKI POKLON (DA/NE)▽";Q$
1040 IF Q$="DA" THEN G$=G$+"2"
1050 INPUT "SALJETE LI NOVOGODISNJU CESTITKU (DA/NE)▽";
     Q$
1060 IF Q$="DA" THEN G$=G$+"3"
1070 INPUT "SALJETE LI NOVOGODISNJI POKLON (DA/NE)▽";Q$
1080 IF Q$="DA" THEN G$=G$+"4"
1090 RETURN
```

Vidite li kako se G$ postepeno formira?
Sličnost pitanja i uticaj na G$ sugerišu alternativni pristup.
Pretpostavimo da na početku programa formiramo znakovnu matricu:

BC$

1	RODJENDANSKU CESTITKU
2	RODJENDANSKI POKLON
3	NOVOGODISNJU CESTITKU
4	NOVOGODISNJI POKLON

U tom slučaju mogli bismo da napišemo:

```
1000 G$=" "
1010 FOR P=1 TO 4
1020 PRINT "SALJETE LI▽";BC$(P);
1030 INPUT Q$
1040 IF Q$="DA" THEN G$=G$+STR$(P)
1050 NEXT P
1060 RETURN
```

Primetimo da je potrebno da P u redu 1040 pretvorimo u njegov znakovni ekvivalent naredbom STR$.

U ovom slučaju stvarno nije bilo vredno truda, pošto tri ušteđena reda izazivaju pisanje više od tri reda za formiranje matrice B$. Međutim, ako ima još nekoliko mogućnosti, ovo može da bude pogodna tehnika rešavanja problema.

Vežba 2

Najjednostavnije je dodati red:

```
15 INPUT "UPISI MESEC KAO DVOCIFRENI BROJ": MTH$
```

a red 60 izmeniti u:

```
60 IF MID$ (BD$, 3, 2)=MTH$ THEN GOSUB 500
```

Još bolje je obezbediti se od slučajnog upisivanja samo jedne cifre (5 umesto 05, npr.) ovako:

```
16 IF LEN (MTH$)=1 THEN MTH$="0"+MTH$
```

Vežba 3

Ovde nam neće biti potrebni podaci o datumu, pa red 60 postaje:

```
60 GOSUB 500
```

a redovi 520 i 530 se menjaju u:

```
520 IF MID$ (G$, P, 1)="3" THEN NC=NC+1 : FLAG=1 : PRINT
    "CESTITKA"
530 IF MID$ (G$, P, 1)="4" THEN FLAG=1 : PRINT "POKLON"
```

Svaki put kad je potrebna novogodišnja čestitka, sadržaj promenljive NC se povećava za 1, pa bi radi sigurnosti trebalo na početku u tu promenljivu upisati vrednost 0:

```
6 NC=0
```

a na kraju je samo ispišemo:

```
65 PRINT "BROJ POTREBNIH CESTITKI"; NC
```

Vežba 4

U originalnom programu treba dodati datoteku:

```
10 OPEN 3, CAS, OUT, "TRACE"
```

(pretpostavlja se da CAS i OUT imaju vrednosti kao i ranije)

a zatim u svaki red koji se prati dodati PRINT# 3:

```
150 PRINT# 3, "150" : REM OVU NAREDBU TREBA PRATITI
```

Zatim na kraju programa, ali pre zatvaranja datoteke, treba upisati u datoteku pogodnu graničnu vrednost:

```
800 PRINT# 3, "-1"
810 CLOSE 3
820 END
```

Za dobijanje profila određenog reda možete da napišete:

```
10 IN=0:CAS=1:COUNT=0
20 OPEN 5,CAS,IN,"TRACE"
30 INPUT "BROJ TRAZENOG REDA";L
40 INPUT#5,N
50 IF N<0 THEN PRINT COUNT:END
60 IF L=N THEN COUNT=COUNT+1
70 GOTO 40
```

Lako i zgodno.

Ovo biste mogli i da proširite tako da dobijete profil niza redova, a da ne morate da premotavate traku. Biće vam potrebna matrica da biste memorisali sve brojeve redova koje hoćete da pratite, a umesto reda 60 da pozovete potprogram koji pretražuje matricu.

Programiranje prepuštam vama.

Dodatak 1: Binarno-decimalna konverzija bajta

Binarno	Decimalno	Binarno	Decimalno	Binarno	Decimalno	Binarno	Decimalno
00000000	0	01000000	64	10000000	128	11000000	192
00000001	1	01000001	65	10000001	129	11000001	193
00000010	2	01000010	66	10000010	130	11000010	194
00000011	3	01000011	67	10000011	131	11000011	195
00000100	4	01000100	68	10000100	132	11000100	196
00000101	5	01000101	69	10000101	133	11000101	197
00000110	6	01000110	70	10000110	134	11000110	198
00000111	7	01000111	71	10000111	135	11000111	199
00001000	8	01001000	72	10001000	136	11001000	200
00001001	9	01001001	73	10001001	137	11001001	201
00001010	10	01001010	74	10001010	138	11001010	202
00001011	11	01001011	75	10001011	139	11001011	203
00001100	12	01001100	76	10001100	140	11001100	204
00001101	13	01001101	77	10001101	141	11001101	205
00001110	14	01001110	78	10001110	142	11001110	206
00001111	15	01001111	79	10001111	143	11001111	207
00010000	16	01010000	80	10010000	144	11010000	208
00010001	17	01010001	81	10010001	145	11010001	209
00010010	18	01010010	82	10010010	146	11010010	210
00010011	19	01010011	83	10010011	147	11010011	211
00010100	20	01010100	84	10010100	148	11010100	212
00010101	21	01010101	85	10010101	149	11010101	213
00010110	22	01010110	86	10010110	150	11010110	214
00010111	23	01010111	87	10010111	151	11010111	215
00011000	24	01011000	88	10011000	152	11011000	216
00011001	25	01011001	89	10011001	153	11011001	217
00011010	26	01011010	90	10011010	154	11011010	218
00011011	27	01011011	91	10011011	155	11011011	219
00011100	28	01011100	92	10011100	156	11011100	220
00011101	29	01011101	93	10011101	157	11011101	221
00011110	30	01011110	94	10011110	158	11011110	222
00011111	31	01011111	95	10011111	159	11011111	223
00100000	32	01100000	96	10100000	160	11100000	224
00100001	33	01100001	97	10100001	161	11100001	225
00100010	34	01100010	98	10100010	162	11100010	226
00100011	35	01100011	99	10100011	163	11100011	227
00100100	36	01100100	100	10100100	164	11100100	228
00100101	37	01100101	101	10100101	165	11100101	229
00100110	38	01100110	102	10100110	166	11100110	230
00100111	39	01100111	103	10100111	167	11100111	231
00101000	40	01101000	104	10101000	168	11101000	232
00101001	41	01101001	105	10101001	169	11101001	233
00101010	42	01101010	106	10101010	170	11101010	234
00101011	43	01101011	107	10101011	171	11101011	235
00101100	44	01101100	108	10101100	172	11101100	236
00101101	45	01101101	109	10101101	173	11101101	237
00101110	46	01101110	110	10101110	174	11101110	238
00101111	47	01101111	111	10101111	175	11011111	239
00110000	48	01110000	112	10110000	176	11110000	240
00110001	49	01110001	113	10110001	177	11110001	241
00110010	50	01110010	114	10110010	178	11110010	242

Binarno	Decimalno	Binarno	Decimalno	Binarno	Decimalno	Binarno	Decimalno
00110011	51	01110011	115	10110011	179	11110011	243
00110100	52	01110100	116	10110100	180	11110100	244
00110101	53	01110101	117	10110101	181	11110101	245
00110110	54	01110110	118	10110110	182	11110110	246
00110111	55	01110111	119	10110111	183	11110111	247
00111000	56	01111000	120	10111000	184	11111000	248
00111001	57	01111001	121	10111001	185	11111001	249
00111010	58	01111010	122	10111010	186	11111010	250
00111011	59	01111011	123	10111011	187	11111011	251
00111100	60	01111100	124	10111100	188	11111100	252
00111101	61	01111101	125	10111101	189	11111101	253
00111110	62	01111110	126	10111110	190	11111110	254
00111111	63	01111111	127	10111111	191	11111111	255

Dodatak 2: Pregled registara za sprajtove

V = 53248 = početna adresa zone registara

Adresa	Sadržaj								Funkcija
V+ 0	Broj kolone sprajta 0								Položaj sprajta
V+ 1	Broj reda sprajta 0								
V+ 2	Broj kolone sprajta 1								
V+ 3	Broj reda sprajta 1								
V+ 4	Broj kolone sprajta 2								
V+ 5	Broj reda sprajta 2								
V+ 6	Broj kolone sprajta 3								
V+ 7	Broj reda sprajta 3								
V+ 8	Broj kolone sprajta 4								
V+ 9	Broj reda sprajta 4								
V+10	Broj kolone sprajta 5								
V+11	Broj reda sprajta 5								
V+12	Broj kolone sprajta 6								
V+13	Broj reda sprajta 6								
V+14	Broj kolone sprajta 7								
V+15	Broj reda sprajta 7								
V+16	Sp7	Sp6	Sp5	Sp4	Sp3	Sp2	Sp1	Sp0	Indikator korekcije
V+21	Sp7	Sp6	Sp5	Sp4	Sp3	Sp2	Sp1	Sp0	uključ./isključe ,je
V+23	Sp7	Sp6	Sp5	Sp4	Sp3	Sp2	Sp1	Sp0	vertikalno širenje
V+29	Sp7	Sp6	Sp5	Sp4	Sp3	Sp2	Sp1	Sp0	horizontalno širenje
V+30	Sp7	Sp6	Sp5	Sp4	Sp3	Sp2	Sp1	Sp0	indikatori sudara
V+39	Kôd boje sprajta 0								Boje
V+40	Kôd boje sprajta 1								
V+41	Kôd boje sprajta 2								
V+42	Kôd boje sprajta 3								
V+43	Kôd boje sprajta 4								
V+44	Kôd boje sprajta 5								
V+45	Kôd boje sprajta 6								
V+46	Kôd boje sprajta 7								
2040	Ukazivač na podatke sprajta 0								Ukazivači
2041	Ukazivač na podatke sprajta 1								
2042	Ukazivač na podatke sprajta 2								
2043	Ukazivač na podatke sprajta 3								
2044	Ukazivač na podatke sprajta 4								
2045	Ukazivač na podatke sprajta 5								
2046	Ukazivač na podatke sprajta 6								
2047	Ukazivač na podatke sprajta 7								

Dodatak 3: Biblioteka potprograma za sprajtove

Ova biblioteka potprograma ima za cilj da vam olakša korišćenje sprajtova. Svaki od potprograma može se pozvati u glavni program korišćenjem naredbe GOSUB, pri čemu je potrebno prethodno postaviti odgovarajuće parametre. Kao i uvek, V=53248 predstavlja početnu adresu zone u memoriji koja je rezervisana za sprajtove.

Postavljanje ukazivača za sprajt K na blok PTR

```
20000 POKE 2040+K,PTR
20010 RETURN
```

Smeštanje podataka za sprajt K u blok PTR

```
20100 FOR G=0 TO 62
20110 READ H
20120 POKE 64*PTR+G,H
20130 NEXT G
20140 RETURN
```

(Ovaj potprogram podrazumeva da se naredba DATA, koja je neophodna, nalazi u glavnom programu.)

Uključivanje sprajta K

```
20200 POKE V+21,(PEEK(V+21)) OR 2↑K
20210 RETURN
```

Isključivanje sprajta K

```
20300 POKE V+21,(PEEK(V+21)) AND (255—2↑K)
20310 RETURN
```

Postavljanje CR boje za sprajt K

```
20400 POKE V+39+K,CR
20410 RETURN
```

Postavljanje sprajta K u red Y *i kolonu* X

```
20500 POKE V+2*K+1,Y
20510 IF X>255 THEN OF=1
20520 POKE V+2*K,X—256*OF
```

```
20530 IF OF=1 THEN POKE V+16,PEEK(V+16) OR 2↑K
20540 IF OF=0 THEN POKE V+16,PEEK (V+16) AND (255—2↑K)
20550 OF=0
20560 RETURN
```

Horizontalno širenje sprajta K

```
20600 POKE V+29,PEEK(V+29) OR 2↑K
20610 RETURN
```

Horizontalno skupljanje sprajta K

```
20700 POKE V+29,PEEK(V+29) AND (255—2↑K)
20710 RETURN
```

Vertikalno širenje sprajta K

```
20800 POKE V+23,PEEK(V+23) OR 2↑K
20810 RETURN
```

Vertikalno skupljanje sprajta K

```
20900 POKE V+23,PEEK(V+23) AND (255—2↑K)
20910 RETURN
```

Detektovanje sudara sprajta K sa bilo kojim drugim sprajtom

```
21000 IF (PEEK(V+30) AND 2↑K) = 2↑K THEN GOSUB 30000
21010 RETURN
```

(Na 30000 se nalazi potprogram koji se izvršava ukoliko nastupi sudar.)

Detektovanje sudara sprajtova K i L

```
21100 IF(PEEK(V+30) AND (2↑K+2↑L)) = 2↑K+2↑L THEN 30000
21110 RETURN
```

Detektovanje sudara između sprajta K i teksta

```
21200 IF(PEEK(V+31) AND 2↑K) = 2↑K THEN 30000
21210 RETURN
```

Dodatak 4: Registri integrisanog kola za zvuk

S = 54222

Adresa			Sadržaj		Funkcija
glas 1	glas 2	glas 3			
S+0	S+7	S+14	Niži bajt učestanosti		Definiše visinu tona
S+1	S+8	S+15	Viši bajt učestanosti		
S+2	S+9	S+16	Niži bajt impulsa (samo za impulse)		Definisanje tona impulsnog talasnog oblika
S+3	S+10	S+17	Viši bajt impulsa (samo za impulse)		
S+4	S+11	S+18	Kôd talasnog oblika*		Izbor "instrumenta"
S+5	S+12	S+19	Uspon (4 bita)	opadanje (4 bita)	uspon/opadanje
S+6	S+13	S+20	održavanje (4 bita)	stišavanje (4 bita)	održavanje/stišavanje
S+24			jačina tona (0—15)		glasnost

* Kodovi talasnih oblika:
17 — trougaoni oblik
33 — testerasti oblik
65 — impulsni oblik
129 — šum

Dodatak 5: Memorijska mapa računara „Commodore 64"

Ovo je jednostavni vodič za glavne zone "šezdesetčetvorkine" memorije. Za više detalja vidi Dodatak 6 i *Vodič*. Primetimo da neke zone imaju više od jedne namene zavisno od konteksta i toga kako su neke sistemske promenljive definisane.

Adrese	Sadržaj
0—827	Sistemski deo
828—1023	Bafer za kasetofon + slobodan prostor
1024—2023	Ekranska memorija (video RAM)
2040—2047	Ukazivači na podatke o sprajtovima
2048—40959	Normalno prostor za program u BASIC-u
40960—49151	BASIC ROM (ako se koristi PEEK) *ili* 8K RAM-a
49152—53247	4K RAM-a
53248—57343	Ulazno/izlazni uređaji i memorija za boje *ili* ROM za znakove (vidi poglavlje 13) *ili* 4K RAM-a
53248—54271	Kolo VIC (Sprajtovi + ekranski prikaz)
54272—55295	Kolo SID (zvuk)
55296—56319	Memorija za boje
53620—56575	Kolo CIA broj 1 (interfejs za dodatke)
56576—56831	Kolo CIA broj 2
56832—57343	Rezervisano za U/I proširenja u budućnosti
57344—65535	KERNAL ROM* (ako se koristi PEEK) *ili* 8K RAM-a

* KERNAL je centralni operativni sistem projektovan tako da napisani programi budu kompatibilni sa budućim verzijama računara.

Dodatak 6: Neke korisne sistematske promenljive

Ovo je izbor iz adresa koje koristi operativni sistem, a koje se mogu koristiti pomoću naredbi PEEK i POKE. Za kompletan opis vidi *Vodič*.

Naziv	Adresa	Funkcija
TXTTAB	43—44	Ukazivač na početak zone BASIC-teksta
VARTAB	45—46	Ukazivač na početak promenljivih u BASIC-u
ARYTAB	47—48	Ukazivač na početak matrica u BASIC-u
STREND	49—50	Ukazivač na kraj matrica + 1
FRETOP	51—52	Ukazivač na početak memorije nizova
CURLIN	57—58	Broj tekućeg reda BASIC-a
OLDLIN	59—60	Broj prethodno izvršenog reda BASIC-a
DATLIN	63—64	Broj reda tekuće naredbe DATA
FBUFPT	113—114	Ukazivač na bafer kasetofona
RNDX	139—143	Vrednost na osnovu koje RND izračunava slučajan broj
LSTX	197	Kôd trenutno pritisnutog tastera*
USER	243—244	Ukazivač na memoriju boja
COLOR	646	Trenutna boja znakova
XMAX	649	Veličina bafera tastature
	53265 (bit 5)	Uključenje grafike visokog razlaganja
	53280	Boja ruba
	53281	Boja osnove

* Za kodove vidi dodatak 7.

Primedba: Većina promenljivih ima po 2 bajta. Ako su njihove adrese X X+1, vrednost u promenljivoj je:

PEEK (X)+256 * PEEK (X+1)

(prvo niži pa viši bajt). Npr. adresa početka zone BASIC-teksta je:

PEEK (43)+256 * PEEK (44)

Dodatak 7: Kodovi očitavanja tastature

U ovom dodatku izlistani su sadržaji adrese 197 kad je pritisnut dati taster. Naredbom PEEK(197) može se, bez pozivanja na bafer tastature, detektovati koji je taster pritisnut.

Taster	Kôd	Taster	Kôd	Taster	Kôd
(ni jedan)	64		46	T	22
*	49	A	10	U	30
+	40	B	28	V	31
,	47	C	20	W	9
—	43	D	18	X	23
.	44	E	14	Y	25
/	55	F	21	Z	12
0	35	G	26	RETURN	1
1	56	H	29	CLR/HOME	51
2	59	I	33	INST/DEL	0
3	8	J	34	CRSR ↓ ↑	7
4	11	K	37	CRSR → ←	2
5	16	L	42	←	57
6	19	M	36	f1	4
7	24	N	39	f3	5
8	27	O	38	f5	6
9	32	P	41	f7	3
:	45	Q	62	£	48
;	50	R	17		
=	53	S	13		

Ovo je uvod u BASIC Commodora 64 za one koji se prvi put sreću sa računarima i one koji prelaze sa drugog računara. Videćete šta su:

— znakovni nizovi
— matrice
— zvuk i muzika
— sprajtovi
— grafika visokog razlaganja

Uključene su liste više programa koji odmah mogu da se izvršavaju. Primeri:

— pretraživanje planeta
— spunerizmi
— grand prix
— Morzeov kôd

Nizom poglavlja o osnovnim tehnikama pronalaženja i otklanjanja grešaka pokriveno je ozbiljno područje programiranja. Obrađeno je i rukovanje datotekama i projektovanje programa. Opširni dodaci daju informacije o binarno-decimalnoj konverziji, registrima sprajtova, registrima integrisanog kola za zvuk i memorijskoj mapi. Na kraju svakog poglavlja dat je niz vežbi sa odgovorima.

Ova najnovija knjiga tima *Stewart i Jones* pomoći će vam da uspešno koristite vaš Commodore.

YU ISBN 86-325-0020-1