

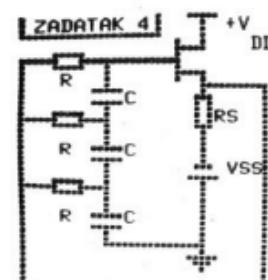
# Uputstvo za upotrebu računara

## REČNIK RAČUNARA "GALAKSIJA"

READY	BREAK	WHAT?	HOW?
SORRY	LIST	RUN	NEW
SAVE	OLD	EDIT	NEXT
INPUT	IF	GOTO	CALL
UNDOT	RET	TAKE	!
#	FOR	PRINT	DOT
ELSE	BYTE	WORD	ARR\$
STOP	HOME	RWD	MEM
KEY	BYTE	WORD	PTR
VAL	EQ	INT	&
USR	DOT	STEP	AT
X\$	Y\$	CHR\$	ELSE
TO			

TOČIMO  
I BEZ  
BONOVA

H  
0  
ME  
20 PRINT "ASCII KARAKTERI"  
30 FOR I=32 TO 192  
40 PRINT CHR\$(I);  
50 NEXT I



ZA RC OS-  
CILATOR  
NAĆI UČE-  
STANOST I  
NAJMANJE  
POJAČANJE  
SORS-FO-  
LOVERA DA  
BI DOŠLO  
DO OSCI-  
LOVANJA.

Dejan Ristanović  
"galaksija"

Računar „galaksija“ je, dakle, sastavljen i nalazi se pred vama. Teži i riskantniji deo posla je time završen — preostalo vam je da testirate kompjuter i — uživate u plodovima svoga rada! Vaš novi računar poznaje programski jezik bejzik i sa pravom isto to očekuje i od svog vlasnika. Da biste se, dakle, uspešno sporazumevali sa računarom, morate da uđete u tajne ovog jezika što, kao što ćete videti, nije mnogo težak posao. Ovo uputstvo za upotrebu će izložiti sve naredbe koje računar „glaksija“ razume i dati primere njihove upotrebe. Ono, ipak, nije pisano tako da bude potpuno samostalno — u okviru ovog specijalnog izdanja nalazi se osnovna škola bejzika, koju bi svaki vlasnik računara „galaksija“ koji nema iskustva sa programiranjem trebalo da pročita pre nego što nastavi sa ovim tekstrom. Primetićete da su neke naredbe računara „galaksija“ različite od onih koje su pomenute u školi bejzika. To ne treba da vas uplaši — bejzik nije standardizovan jezik i razlikuje se od kompjutera do kompjutera. No, kada se jednom potrudite i shvatite neki od njegovih dijalekata, prilagođavanje nekom drugom računaru je prava dečja igra.

Ovo uputstvo je koncipirano tako da vam omogući da pišete programe pre nego što ga potpuno pročitate i savladate. Zato su najvažniji oblici naredbi izloženi na njegovom početku, dok se deo za bolje poznavaoce računara nalazi na samom kraju. Onima koji imaju iskustva sa stonim računarima preporučujemo da jednostavno pogledaju spisak naredbi i memoriju mapu i počnu sa radom — na uputstvo će se vraćati kada osete potrebu.

## 1. Povezivanje sa televizorom

Računar „galaksija“ može da bude povezan sa bilo kojim televizorom koji prima UHF područje. Za to se koristi koaksijalni priključak koji se nalazi sa njegove zadnje strane. Stoga uzmite odgovarajući kabl i povežite ga sa antenskim ulazom u televizor. Zatim uključite televizor, prebacite ga na UHF područje i, laganim okretanjem birača kanala, pokušajte da na ekrantu dobijete poruku READY (READY znači da je računar spreman da primi vaše naloge). Kada je ugledate, možete da uživknete EUREKA — računar radi kako treba!

Čitaoci koji poseduju monitor ili prepravljeni televizor će poželeti da povežu računar sa njim i tako oslobođe kućni TV aparat. Za to treba koristiti monitorski izlaz i kabl sa priključcima koji zavise od samog monitora.

## 2. Tastatura

Računar „galaksija“ ima 54 tastera koji su raspoređeni poput dirki na pisaćoj mašini. Kada god pritisnete neki od njih, računar na ekrantu ispisuje odgovarajući simbol i to na mestu gde se prethodno nalazio kurzor (—). Čim se neki simbol pojavi na ekrantu, kurzor se pomera za jedno mesto udesno da bi vam pokazao gde će se pojaviti sledeći znak. Ukoliko otkucate toliko slova da se ispuni čitav red, računar će automatski preći u sledeći bez ikakve crticice koja bi to označila — računar ne deli reči na slogove niti obraća pažnju na to da li je neka reč neologično „presećena“.

Pritisnite, na primer, taster A dvadesetak puta. Zatim će vam postati jasno da veliki broj slova A ne znači ništa, pa ćete poželeti da ih obrišete. Za to vam nije potrebno nikakvo korekturno mastilo — dovoljno je da pritisnete taster ← i kurzor će se pomeriti za



jedno mesto umevo brišući automatski slovo na čije je mesto došao. Uzastopnim pritiscima na ovaj taster možete da obrišete sva otkucana slova — sve do početka reda.

Na tastaturi postoji, jasno, i tačka, ali čete se za koji trenutak uveriti da vaš računar tu tačku ne smatra krajem rečenice. Otkucajmo, na primer, RACUNAJ i stavimo tačku. Kompjuter ne reaguje, što znači da nije ni shvatio da mu je izdata neka naredba. U programiranju je, naime, uobičajeno da se na kraju rečenice, umesto tačke, pritisne taster koji je označen sa **RET**. Pritisnimo ga i na ekranu će se pojaviti poruka **WHAT?** iza koje će slediti uobičajeno **READY**. Šta se dogodilo? Pritiskom na **RET** stavili smo računaru do znanja da je izdata naredba koju on treba da izvrši. On je analizirao tekst koji smo otkucali i primetio da se među naredbama koje poznaje ne nalazi **RACUNAJ**. Zato je na ekranu ispisao **WHAT?**, a zatim i **READY**, stavljajući nam do znanja da je spreman za novu naredbu, koja, kako se sada, neće biti nerazumljiva.

Taster **RET** je, dakle, neobično značajan. Ne zaboravimo da ga pritisnemo kada god dovršimo neku naredbu ili rečenicu — ako to ne učinimo, računar će mirno čekati da nešto preduzmemo, ispoljavajući pri tom upornost koja daleko prevazilazi našu.

Ako nastavimo da pritisnemo **RET**, ekran će ubrzo biti popunjeno. Tada će se prvi redovi postepeno gubiti, a ono što kucamo pojavljivati u poslednjem. Stalno opštenje sa poslednjim redom može nekome da se učini ponizavajućim — u tom slučaju treba da naredimo računaru da obriše sadržaj čitavog ekrana i počne da piše od njegovog vrha. Jedan od načina je da isključimo i ponovo uključimo računar, ali ima i jedan bolji.

Upoznajmo, najpre, taster **SHIFT**. Ovaj napis ne vredi tražiti na tastaturi: odnosi se na dva neobezležena tastera. Istovremeni pritiskom na **SHIFT** i neki broj dobijamo na ekranu znak napisan iznad tog broja (pritisnimo, na primer, istovremeno **SHIFT** i 2. i na ekranu će da se pojavi znak navoda). Kod brojeva je, kao što vidimo, simbol napisan na samom tasteru ali postoje i neke dirke sa skrivenom šiftovanom funkcijom. Pritisnimo, na primer, istovremeno **SHIFT** i **DEL** (ukoliko smo, kao što je predviđeno, pri montaži predviđeli dva **SHIFT** tastera, možemo da koristimo bilo koji od njih) i ekran će biti obrisan. Kada nam, dakle, računarene poruke **WHAT?**, koje su izazvane našim početničkim greškama dosade, pritisnućemo **SHIFT DEL** i obrišis ekran.

Taster **SHIFT** će nam otkriti još neke male tajne računara „galaksija“. Pritisnimo, na primer, taster **SHIFT** i **C** i na ekranu će se pojaviti slovo **Č**. Sasvim slično, **SHIFT S** daje slovo **Š**, **SHIFT Z** slovo **Ž** a **SHIFT X** (zapamtiteće ga po tome što kapa iznad slova Č ima oblik koji podseća na polovinu lksa) — **Č**. Ostali tasteri nemaju šiftovane funkcije — **SHIFT A**, na primer, daje slovo **A**.

Ako ste, čitajući ovaj tekst, eksperimentisali sa kucanjem, verovatno ste, namerno ili slučajno, pritisnuli taster **REPT**. Tada se desilo nešto prilično čudno — računar je počeо da ispisuje gomile istih slova. Taster **REPT** dakle, ima funkciju koju programeri nazivaju „autorepeat“ (automatsko ponavljanje) — pritisak na njega nalaže računaru da ponavlja funkciju poslednjeg pritisnutog tastera. Namerno nismo rekli „da ponavlja poslednje slovo“: otkucajte red sa dosta slova a zatim pritisnite ←. Poslednje slovo je, kao što smo i očekivali, obrisano. No, pritisak na **REPT** će proizvesti daleko efektnije dejstvo: računar će „gutati“ slova sve dok ne obriše čitav red.

Osim tastera koje smo upoznali na tastaturi se nalaze još tri strelice kao i dirke sa tajanstvenim napisima **BRK** i **LIST/STOP** (u daljem tekstu **L/ST**). **BRK** je skraćenica od **BREAK** (u bukvalem prevodu „slomi“); pritisak na ovaj taster nalaže računaru da prekine sa izvršavanjem nekog programa i na ekranu ispiše **READY**. Taster **L/ST** nalaže računaru da na ekranu prikaže sadržaj programa koji se nalazi u njegovoj memoriji, a strelice nemaju neku posebnu funkciju — koriste se u igrama i računar može da kontroliše da li je neka od njih pritisnuta. Za sada ne umemo da pišemo programe pa su nam svi ovi tasteri prilično beskorisni, ali to neće trajati dugo.

Sa zadnje strane računara se nalazi dugme iznad koga bi trebalo da piše **RESET**. Ono predstavlja pojačanu verziju tastera **BRK**: ukoliko vam se ikada deši da se računar blokira i otkaze poslušnost (ovo se dešava, ako je računar ispravan, samo kod onih koji pišu programe na mašinskom jeziku ili pri čitanju programa sa kasete), pritisak na **RESET** će, u 99% slučajeva, izazvati ispisivanje uobičajenog **READY**, pri čemu memorija računara neće biti izbrisana. U preostalom procentu slučajeva moraćete da isključite i ponovo uključite računar, čime će programi biti izbrisani. Važno je zapamtiti da ni jednom naredbom ili grupom naredbi računar ne može da bude oštećen — isključivanje i ponovno uključivanje će ga uvek povratiti.

Pre nego što podemo dalje, moramo da izrekнемo i jedno upozorenje vezano za dugme **RESET**: ono nije namenjeno stalnoj upotrebi. Pokušajte, na primer, da brzo pritiskeTE **RESET** više puta, pa lako može da se desi da memorija računara bude kompletno izbrisana. **RESET**, dakle, koristite samo kao poslednju soluciju, kada nikako drugačije ne možete da „probudite“ vaš kompjuter.

### 3. Demo kasetu

Uz računar „galaksija“ ste, svakako, naručili i kasetu sa programima. Ona je namenjena testiranju računara, navikavanju na rad sa kasetofonom i, prevashodno, zabavi. Da biste je koristili, povežite džek na kome piše **EAR** (ili neki drugi napis, zavisno od kasetofona) sa računarem prema ranije prikazanoj šemci, stavite demo-kasetu, otkucajte **OLD**, pritisnite **RET** i startujte kasetofon. Ovo **OLD**, doduše, znači star, ali neće naterati računar da postane stara i istrošena mašina — ono mu jednostavno nalaže da sa kasete učite program koji je ranije snimljen i pripremi ga za izvršavanje. Ekran će se najpre potpuno zatamniti, da bi se, desetak sekundi dnočije, pojavilo **READY**. Ako se na ekranu i posle tridesetak sekundi ne pojavljuje ništa, računar nije primio program. Bez panike — to se dešava zbog toga što jačina zvuka ne odgovara potrebama kompjutera. Pritisnite zato **RESET**, premotajte kasetu na početak, ponovo otkucajte **OLD RET** i startujte kasetofon uz malo manipulisanje sa potenciometrom za regulisanje jačine zvuka. Posle kraćeg ili dužeg truda, program će biti učitan u memoriju računara i moći ćete da ga startujete. Startovanje se obavlja kucanjem naredbe **RUN** i pritiskanjem tastera **RET**. Bez brige: računar neće nikuda pobeći (**RUN=trči, beži**) nego će početi sa izvršavanjem upravo učitanog programa. Ne možemo da vam kažemo šta će se tada desiti. U okviru prvog programa računar će vas upoznati sa sadržajem demo-kasete i poučiti šta da radite.

Za učitavanje preostalih programa procedura je ista: otkucajte **OLD** pritisnite **RET**, startujete kasetofon i, kada se na ekranu pojavi **READY**, otkucajte **RUN** i pritisnite **RET**. Računar će, zatim, ispisati ime programa i upitati vas da li su potrebna uputstva za njegovu upotrebu. Odgovorite sa **D** (od **DA**) i pažljivo ih pročitajte.

## 4. Komandni i programski mod

Možda će se naći neko ko dalji tekst neće ni čitati — nekoliko igara sa demo-kasete će mu oduzeti toliko vremena da će smatrati da je računar ispunio svoju funkciju i opravdao uloženi novac. Ipak, učenje programiranje je lak i prijatan posao u koji se vredi upustiti.

Počemo na najjednostavniji mogući način: otkucaćemo, slovo po slovo, tekst **PRINT „RAČUNAR GALAKSIJA“** i pritisnuti **RET** (navodnici se dobijaju istovremenim pritiskom na tastere **SHIFT** i 2, a slovo **C** — pritiskom na **SHIFT X**). Tako smo naložili računaru da na ekran prenese tekst naveden između znakova navoda — u ovom slučaju **RAČUNAR GALAKSIJA**. Možemo da ponavljamo ovu naredbu veći broj puta umećući između znakova navoda bilo koji tekst proizvoljne dužine i računar će biti poslušni izvršilac naših želja. Svaka naredba se analizira i izvršava onoga momenta kada smo pritisnuli **RET**, pa se ovaj način rada naziva **komandnim** (na ovaj način računaru izdajemo komande kojima kontrolišemo njegov rad).

Osim komandnog treba da upoznamo **programske** režim rada. I kod njega računaru izdajemo komande, ali se one ne izvršavaju odmah — računar ih pamti i izvršava tek kada mu to bude naloženo. Da bi kompjuter razlikovao komande od naredbi koje treba da „pamtii“, uvedeni su brojevi programske linije. Broj programske linije (ili, kao što se ponekad nazivaju, linijski broj) ukazuje računaru na redosled kojim će, kada mu to bude naloženo, izvršavati „zapamćene“ instrukcije. Ovaj broj treba da se nalazi na samom početku linije.

Otkucajmo, na primer, **10 PRINT „RAČUNAR GALAKSIJA“** i pritisnimo **RET**. Računar nije izvršio naredbu; ona je zapamćena i dodeljen joj je linijski broj 10. Otkucajmo zatim **20 PRINT „JE MOJ PRVI KOMPJUTER“** i ponovo pritisnimo **RET**. Obzirom da je linijski broj druge naredbe (20) veći od linijskog broja prve (10), računar će najpre ispisati **RAČUNAR GALAKSIJA** a zatim **JE MOJ PRVI KOMPJUTER**. Da proverimo ovo rezonovanje, otkucaćemo **RUN** (ovo je naredba koju smo dobro upoznali igrajući se sa demo-kasetom — ona nalaze računaru da započne sa izvršavanjem programa) i pritisnuti neizbežno **RET**. Računar će, kao što se i očekivalo, izvršiti program i na ekranu ispisati:

```
RAČUNAR GALAKSIJA
JE MOJ PRVI KOMPJUTER
READY
```

Ono **READY** nije posledica neke od naredbi našeg programa već uobičajeni signal da je računar spremam da primi sledeću naredbu. To ponovo može da bude **RUN** — program će se izvršavati onoliko puta koliko poželimo.

Pokušajmo da proširimo program koji smo sastavili i da se uverimo da računar izvršava naredbe po redosledu njihovih brojeva a ne po redu unošenja: otkucajmo 15 **PRINT „KONSTRUISAN SEPTEMBRA 1983“**, i pritisnimo **RET**. Po startovanju programa (**RUN** i **RET**), na ekranu će biti ispisani tekst:

```
RAČUNAR GALAKSIJA
KONSTRUISAN SEPTEMBRA 1983.
JE MOJ PRVI KOMPJUTER
READY
```

Što znači da je linija 15 umetnuta između linija 10 i 20 — baš kao što smo i očekivali.

Ako na ovaj način počnemo da umećemo naredbe, ubrzo ćemo zaboraviti kako program izgleda. To je lako proveriti: jednostavno pritisnemo taster na kome je ispisano **LIST** (**LIST** bi slobodno moglo da se prevede sa „prikaži program“). Dok taster **LIST** držimo pritisnut, računar će ispisivati program koji se nalazi u njegovoj memoriji, liniju po liniju. Čim pustimo ovaj taster, ekran će se „zamrznuti“, pa ćemo moći da analiziramo uneseni program. Ako želimo da se listanje nastavi, ponovo ćemo pritisnuti taster **LIST**, a ukoliko ne želimo pritisak na **BRK** će na ekranu prikazati uobičajeno **READY**.

Umesto da pritisnemo taster **LIST**, možemo da otkucamo reč **LIST** i pritisnemo **RET**. Sve dok je **RET** pritisnut, prikazivanje programa traje, a otpuštanje tastera privremeno prekida ovaj proces. Na prvi pogled nema smisla koristiti naredbu **LIST** kada je obezbeđen taster sa istim dejstvom. Ovakav zaključak je samo donekle tačan: ponekad ćemo poželjeti da prikazujemo program počevši od neke linije. U tom slučaju treba da

otkucamo LIST, a zatim navedemo broj te linije i pritisnemo RET (na primer, LIST 200). Na taj način ne moramo da posmatramo početak programa ako nam je potreban njegov kraj. Za početak će, doduše, naši programi biti toliko kratki da će bez problema stajati na ekran, pa naredbu LIST ostavljamo za blisku budućnost.

Ni izvršavanje programa ne mora da počne od početka: možemo, na primer, da otkucamo RUN 15 i pritisnemo RET. Računar će početi da izvršava program od linije 15 i ispisati na ekranu:

KONSTRUISAN SEPTEMBRA 1983.  
JE MOJ PRVI KOMPJUTER  
READY

Ova rečenica je, naravno, besmislena. No, za računar nije postojao nikakav razlog da odbije da je prenese na ekran, s obzirom da su sve instrukcije u programu bile valjane. U budućnosti ćemo videti da postoje programi koji ne mogu da se izvršavaju od proizvoljne linije bez katastrofalnih posledica (računajući i gubitak čitavog programa) ali nam je i ovo bila dobra škola — započinjanje programa od neke neplanirane tačke ne mora da izazove grešku ali su rezultati nepredvidljivi.

Unošenje programa i njegovo neprekidno izvršavanje je, bez sumnje, lepa zabava, ali će nam i ona dosaditi. Kada poželimo da izbrišemo otkucani program iz memorije, možemo da isključimo računar iz napajanja i trenutak docnije ga ponovo uključimo ili, što je sasvim isto, otkucamo PRINT USR (0) i pritisnemo RET. Na raspolažanju je i naredba NEW (posle nje se, naravno, pritiska RET) koja briše samo program, ali ne i sadržaj ekran i promenljivih.

## 5. „Galaksija“ kao kalkulator — PRINT, INT

Naredbe PRINT koje smo do sada koristili su nalagale računaru da tekst otkucan između navodnika jednostavno prenese na ekran. No, ova naredba može da obavi i mnogo aktivniju ulogu: Otkucajmo, na primer, PRINT 17+13 i pritisnimo RET. Na ekranu će se pojavitи broj 30, što znači da je postavljen zadatak izvršen! Umesto 17+13 možemo da otkucamo bilo koji izraz na način koji je uobičajen u matematiči: uz upotrebu zagrada, pri tome treba da pazimo da se množenje obeležava zvezdicom (\*) a deljenje kosom crtom (/) i da se množenje ne podrazumeva: PRINT (12+2)(17+3), bi izazvalo grešku dok bi se PRINT (17+2)\*(17+3) korektno izvršilo. Evo, na primer, kako bi izgledala naredba koja izračunava brojni izraz:

3 ·  $\frac{4.5 + 17.2 \cdot 19.45}{2.1 - 12.4/16.22}$

PRINT 3\*(4.5 + 17.2\*19.45)/(2.1 - 12.4/16.22)) RET

Izvršavanje ove naredbe će na ekranu dati rezultat 761.595, koji je tačan na šest cifara.

Videli smo da brojevi sa kojima računar „galaksija“ operiše ne moraju da budu ceši — dovoljno je da koristimo decimalnu tačku umesto kod nas uobičajenog zareza. Ipak, brojevi sa kojima možemo da radimo su ograničeni na raspon od -999999 do 999999. Brojevi veći ili manji od ovih limita se prikazuju na drugi način: pomoću stepena broja 10. Otkucajmo, na primer, PRINT 1000000 i računar će ispisati 1E+06 (podrazumevamo da ste zapamtili da se posle svake naredbe pritiska RET, pa ubuduće nećemo pisati odgovarajući komentar). Slovo E potiče od reči eksponent, a zapis se prevodi kao  $1 \cdot 10^6$ . Podsećanje na osnovačku matematičku će nas uveriti da je  $1 \cdot 10^6$  isto što i 1000000 — jedino je zapis drukčiji. Čak ni na ovaj način ne možemo da operišemo sa neograničeno velikim brojevima — ukoliko neki od argumenata ili rezultat neke operacije bude manji od  $10^{-38}$  ili veći od  $10^{37}$ , računar će na ekranu ispisati HOW? — tražili smo od njega nešto što prevazilazi njegove računske sposobnosti.

Krajnje je vreme da sastavimo i prvi program. Obzirom da tek počinjemo, to će biti jednostavan programčić koji računa vreme potrebno za putovanje od jednog do drugog

mesta. Da pojednostavimo problem, prepostavimo da je brzina kretanja konstantna na čitavom putu i da je unapred zadata (ovakvim zahtevima bi se verovatno pokoravao voz koji ne nalazi na semafore, krvine i slične prepreke). Ako put koji treba da se pređe označimo slovom **s**, brzinu slovom **v** a potrebno vreme slovom **t**, važiće relacija  $t = s/v$  (ovu relaciju osnovci uče u VI razredu a mladima bi trebala da bude očigledna: što je veći put, duže se putuje; što je veća brzina, kraće se putuje). No, pre nego što počnemo da pišemo program, pokušaćemo da iskoristimo naredbu PRINT i proverimo kako sve u praksi izgleda.

Od Beograda do Dubrovnika, na primer, ima 370 km u pravoj liniji, a avion „Caravelle“ se kreće brzinom od nekim 800 km/h. Da nađemo vreme (u časovima) otkućaćemo:

PRINT 370/800 i na ekranu će se pojaviti broj 0.4625. Olovka i papir (za sada još ne umemo da iskoristimo računar da obavi sve potrebne operacije) će pokazati da ovo predstavlja 27 minuta i 45 sekundi.

Koliko bi put trajao da se avion kretao brzinom od 900 km/h. Ili, šta da je putovao od Beograda do Zagreba? Mogli bismo, jasno, stalno da kucamo naredbe PRINT ali ćemo uvideti da od toga nema velikih vremenskih ušteda. Zato ćemo u program uneti instrukciju

### 30 PRINT S/V

30 je, kao što se sećamo, broj programske linije ali šta su S i V? Programeri ih nazivaju **promenljive** i koriste za smeštanje nekih podataka u memoriju računara. Svaka promenljiva je ime nekog podatka koji danije može da bude pozivan da bi se sa njim računalo. Njeno ime može da bude bilo koje slovo od A do Z (ne računajući slova Č, Č, Ž i Š) i pogodno je koristiti slova koja asociraju na podatak koji promenljiva čuva (mi smo upotrebili slova S i V). Ipak, program koji smo sastavili nije potpun. Ako otkucamo RUN na ekranu će da se pojavi broj 1 koji svakako ne predstavlja rešenje našeg problema. Razlog je jednostavan: računar „galaksija“ još nije naučio da čita naše misli i nema nikakav način da zna šta bi trebalo da sadrže promenljive S i V. U početku rada one imaju jednaku vrednost (**pažnja**: ta vrednost nije nula!) pa je rezultat deljenja jedan. Zato ćemo da dodamo naredbe 10 S=370 i 20 V=800 (nismo zaboravili da se posle svake linije pritiska RET?) i startovati program sa RUN. Ovoga puta sve je korektno radilo i računar je ispisao vreme putovanja od Beograda do Dubrovnika i to izraženo u časovima.

Ako želimo da ubrzamo zamišljeni avion otkućaćemo 20 V=900. Kako nova linija ima isti lininski broj kao i stara, jedna od njih mora da napravi mesto drugog. Računar uvek smatra da je linija koju smo poslednju otkucali merodavna i uklanja iz memorije liniju 20 V=800 da bi je zamenio sa 20 V=900. Program se ponovo startuje sa RUN i dobija se novi rezultat.

Prikazivanje rezultata koje smo primenili nije baš srećno izabrano: računar nam prikazuje vreme u časovima i to na način koji nam nije prirođan: navikli smo da se vreme izražava u časovima, minutima i sekundama. Da prikažemo rezultat na ovakav način potrebna nam je jedna nova funkcija — INT. INT je skraćenica od INTEGER i označava, u bukvalnom prevodu, ceo broj. U „galaksijinom“ bežizku INT je funkcija koja izračunava „ceo deo“ izraza koji je naveden iza nje. Ako, na primer, otkucamo PRINT INT (3.1415) i pritisnemo RET, na ekranu će se pojaviti broj 3 — računar je jednostavno odbacio decimalne broje u zagradi. U zagradi nije moral da se nalazi konstanta: mogli smo da napišemo PRINT INT (A/1000) ili čak PRINT INT (2,5 + (A \* 17,2 + C)).

Da bismo broj pretvorili u časove, minute i sekunde, moramo da pomnožimo vreme u časovima sa 60 i ceo deo rezultata proglašimo za potreban broj minuta; broj časova dobijamo deljenjem vremena sa šezdeset i uzimanjem celog dela rezultata (možete da uzmete papir i olovku i proverite ovo rezonovanje na primerima; nema nikakvog smisla da nastavljate sa čitanjem pre nego što vam do sada korišćene naredbe ne postanu kristalno jasne). Evo i programa:

```
10 S=370
20 V=800
30 T=S/V
40 PRINT INT(T)
```

```
50 PRINT „ČASOVA.“  
60 M=INT(M*60)  
70 PRINT M  
80 PRINT „MINUTA.“
```

Program, kao što se vidi, prikazuje samo časove i minute ali ćete, ako ste ga razumeli, bez problema povećati tačnost navođenjem sekundi!

## 6. Naredba INPUT

Da bismo u prethodnom programu izbegli neprekidno kucanje brojnog izraza koji računa s/v i kome prethodi naredba PRINT, koristili smo promenljive. Na žalost, ni na ovaj način kucanje nije bitno skraćeno: bolje bi bilo da nas, pri svakom izvršavanju programa, računar pita za pojedine vrednosti, a da mi samo kucamo potrebne brojeve. Ovaj problem rešava naredba INPUT, čije ćemo dejstvo upoznati pošto modifikujemo prethodni program:

```
10 PRINT „KOLIKO SU MESTA UDALJENA?“  
15 INPUT S  
20 PRINT „KOLIKA JE BRZINA?“  
25 INPUT V
```

Kada računar nađe na naredbu INPUT, privremeno će da prekine sa izvršavanjem programa i na ekranu ispiše znak pitanja. Od korisnika se očekuje da otkuca neki broj i, kada to učini, pritisne **RET**. Računar će otkucani broj dodeliti promenljivoj čije je ime navedeno iza INPUT i nastaviti sa radom.

Program se, kao i obično, startuje sa RUN. Računar ispisuje KOLIKO SU MESTA UDALJENA?

?\_\_

Otkucajmo broj 370 i pritisnimo **RET**. Računar ispisuje: KOLIKA JE BRZINA?

?\_\_

Čim otkucamo 800 i pritisnemo **RET**, na ekranu će pisati:

```
0  
ČASOVA,  
27  
MINUTA.  
READY
```

READY je znak da je računar spreman za dalji rad; možemo da otkucamo RUN i damo podatke za neki drugi grad ili brzinu. No, ubrzo će nam dosaditi stalno kucanje naredbe RUN. Zar ne bi bilo bolje naložiti računaru da se, po izvršavanju programa i izdavanju rezultata, vrati na početak programa i od nas zatraži nove podatke? Za to će nam dobro poslužiti naredbe bezuslovnog prelaska (skoka).

## 7. Naredbe bezuslovnog i uslovnog prelaska — *GOTO, IF.. ELSE, STOP*

Dodajmo našem programu naredbu 90 GOTO 10 (pažnja: u principu nije bitno da li u programu ostavljamo prazne prostore između GOTO (ili bilo koje druge naredbe) i broja 10 ali ne smemo da ostavljamo prostor unutar same naredbe: GO TO 10 će izazvati

poruku WHAT? i prekid rada programa). Smisao naredbe GOTO je sasvim jasan čak i za onoga ko nije imao nikakvih kontakta sa računarima: kada kompjuter nađe na nju, preći će na izvršavanje linije broj 10. Na ovaj način smo sastavili program koji je „opasan“: kada ga jednom startujemo on će se neprekidno izvršavati, stalno nas pitajući za podatke. Kada iscrpimo sve podatke, izvršavanje programa postaje bespredmetno i treba ga prekinuti. Jednostavno ćemo pritisnuti taster **BRK** i računar će ispisati BREAK 15 (što znači da je, u momentu kada smo ga zaustavili, izvršavao liniju broj 15) i uobičajeno READY. Zatim možemo da otkucamo NEW i tako obrišemo program, ali ćemo se za sada uzdržati od toga — program možemo da proširimo upoznajući dve nove naredbe.

Naredba GOTO se izvršava uvek na isti način — kada računar nađe na nju, prelazi na izvršavanje naredbe čiji je broj naveden iza GOTO (možemo da napišemo i GOTO A ako je promenljivoj A ranije dodeljen broj linije na koju računar treba da ide, ali ćete ovo indirektno adresiranje, bar u početku, ređe koristiti). No, postoji naredba koja izaziva različito dejstvo u zavisnosti od nekih parametara — to je naredba IF. Iza IF se nalazi neka pogodbena relacija čiju tačnost računar ispituje. Da bismo ovo razumeli, dodaćemo u naš program naredbu **18 IF S=0 STOP**. Treba, pre svega, da razumemo njenu sintaksu: STOP je naredba koju tek upoznajemo i koja jednostavno izaziva prekid rada. S=0 iza IF naredbe ne označava, kao do sada da se promenljivoj S dodeljuje vrednost nula već navodi računar da ispisat će li je vrednost promenljive S jednaka nuli. Ako jeste, izvršava se naredba STOP i računar prestaje sa radom, a ako nije, naredba STOP biva ignorisana i računar prelazi na izvršavanje sledeće naredbe.

Uz IF se često koristi i naredba ELSE. Možemo da unesemo novu naredbu 18 koja glasi:

18 IF S=0 STOP: ELSE PRINT „NASTAVLJAM SA RADOM“

Ukoliko je S zaista nula, računar će se zaustaviti. Ako nije, računar će izvršiti naredbu iza ELSE, a zatim nastaviti da izvršava program. Dve tačke ispred ELSE za sada morate da primite „zdravo za gotovo“ — njihova uloga će vam postati jasna nešto kasnije.

Kakav je smisao promena koje su izvršene? Korisniku je sada pružena mogućnost da, kada računar traži podatak o rastojanju gradova, otkuca 0 i pritisne **RET**. Računar će prestati sa radom i ispisati **READY**, eliminujući potrebu za pritiskanjem tastera **BRK**.

Umesto znaka = u pogodbenoj relaciji može da se nađe znak manje (na primer IF A<B PRINT „PRVI JE MANJI“: ELSE PRINT „DRUGI JE MANJI ILI SU JEDNAKI“) ili veće (na primer IF X>0 PRINT „BROJ JE POZITIVAN“: ELSE PRINT „BROJ JE NEGATIVAN“).

Ako eksperimentiramo sa IF naredbama poremetimo naš program koji će nam za dalji rad biti potreban, moraćemo da obrišemo suvišne linije. Brisanje linije broj 18, na primer, je vrlo jednostavno: otkucamo 18 i pritisnemo **RET**. Metoda je pogodna, ali stvara jedan artefakt: želimo, na primer, da izračunamo koliko je 31/17 i zaboravimo da otkucamo PRINT. Računar će shvatiti da smo uneli programsku liniju broj 31 i obrisati eventualnu naredbu našeg programa koja ima isti broj, zamenivši je besmislenom linijom /17.

## 8. Prijavljivanje grešaka

Već smo upoznali poruke WHAT? i HOW? koje bivaju ispisane na ekranu kada našem kompjuteru „nešto nije jasno“. Poruka WHAT? se uglavnom odnosi na sintaksne greške: upotrebili smo naredbu koju naš računar ne poznaje ili smo valjanu naredbu napisali na pogrešan način (PRINT AB, na primer, nema smisla jer se množenje ne podrazumeva). Poruku HOW? računar izdaje kada smo prekoračili opseg brojeva sa kojima se računa, kada smo naredbu GOTO (ili naredbu CALL koju tek treba da upoznamo) propratili linijskim brojem nepostojeće naredbe, kada smo pozvali previše potprograma i u sličnim situacijama.

Pored HOW? i WHAT?, postoji sistemska poruka sa kojom se nećemo sresti u skoroj budućnosti: SORRY. Ona nas obaveštava da je ono što smo želeli da učinimo sasvim moguće i ispravno napisano, ali da šest kilobajta (ili više ako smo proširili memoriju) nije dovoljno za smeštanje potrebnog programa ili podataka. Ako ovo uputstvo niste čitali „na preskok“ upoznavši tako matrice A i X\$, jedini način da na ekranu vidite

SORRY je da otkucate neki program koji ima više od pet kilobajta (računar „galaksija“ u osnovnoj verziji ima 6 kilobajta RAM memorije, ali se nešto više od jednog kilobajta koristi za neke interne potrebe računara).

Ukoliko računar prijavi grešku u okviru neke komande (otkucali smo, na primer, RAN umesto RUN), na ekranu će se pojaviti jednostavno WHAT?, HOW? ili SORRY. No, ako računar nađe na grešku u programu, on će, pored uobičajene poruke, ispisati čitavu liniju u kojoj je nastupila greška, stavljajući znak pitanja na ono mesto u liniji na kome je naišao na nerazumljiv simbol ili konstrukciju. Tako ne bi trebalo da bude teško da locirate grešku i, prekucavajući liniju ili koristeći komandu EDIT, dovedete program u ispravno stanje. U početku ćete verovatno mnogo grešiti kod oblika (sintakse) naredbi, ali će se docnije greške svesti na pogrešno kucanje i, na žalost, logičke greške (najopasnije su greške koje računar ne primećuje) — program se na prvi pogled korektno izvršava, ali su krajnji rezultati pogrešni.

## 9. Rad sa kasetofonom — SAVE, OLD i OLD?

Znanje koje smo stekli nam omogućava da pišemo relativno komplikovane programe. Međutim, svaki put kada isključimo računar ti programi bivaju obrisani pa ih, po ponovnom uključivanju, moramo kucati, što je veoma neprijatan posao. Na svu sreću, računar „galaksija“ može da se poveže sa kasetofonom i snima programe na standardne kasete. Programi snimljeni na kasetu nemaju imena pa može da se desi da, nekoliko meseci po snimanju, zaboravimo šta koji program radi. Zato je korisno na početku programa dodati liniju 1! OVO JE PROGRAM ZA RAČUNANJE ...

Uzvičnik na početku ove linije govori računaru da ona sadrži komentare koji su za njega nebitni: pri izvršavanju programa linije sa uzvičnicima bivaju ignorisane. Samo se po sebi razume da linija koja počinje uzvičnikom ne mora da bude prva u programu. Možemo, na primer, da pre svake grupe naredbi ubacimo komentar koji objašnjava njihovo dejstvo, ali treba imati u vidu da ovakve naredbe troše jedan bajt memorije po svakom upotrebljenom slovu što je, u uslovima ograničenog memorijskog prostora, često luksuz.

Kada pripremimo program za snimanje, jednostavno povezujemo odgovarajući džek sa priključkom MIC na kasetofonu, startujemo snimanje (dirke PLAY i REC), kucamo SAVE i pritiskamo RET (SAVE — sačuvaj). Ekran će se potpuno zamračiti i ovakvo stanje će potrajati izvesno vreme. Kada računar snimi program, ekran će oživeti sa novom porukom READY. Sačekajmo još nekoliko trenutaka i prekinimo snimanje.

Posle snimanja je vrlo korisno izvršiti verifikaciju. To što se na ekranu pojavit će READY još ne mora da znači da je program verno prenet na kasetu: moguće je da je traka bila oštećena (ovo oštećenje može da bude prisutno i kod potpuno nove kasete i ne mora biti uočljivo golim okom) ili da je kasetofon u nekom trenutku za moment promenio brzinu. Ukoliko je nastupila neka greška ovog tipa, možemo da imamo velikih problema: isključimo računar i, posle pokušaja da učitamo program koji smo razvijali satima ili danima, utvrđimo da je on nepovratno izgubljen. Ako izvršimo verifikaciju, bićemo upozoren na problem, pa ćemo moći da snimimo program na neku drugu kasetu.

Da bismo izvršili verifikaciju, moramo da premotamo kasetu na mesto koje prethodi početku snimljenog programa (veoma je preporučljivo posedovati kasetofon sa brojačem mada se i bez njega program prepoznaće po veoma karakterističnom zvuku; teško da ste ikada čuli neprijatljive zvukove) i zaustavimo ga. Povežimo utičnicu EAR na kasetofon sa odgovarajućim priključkom računara, postavimo nivo zvuka blizu maksimuma i otkucajmo OLD? RET. Zatim startujemo kasetofon (taster PLAY) i čekamo. Može da se dogodi jedna od tri stvari:

Prva i najpovoljnija je da računar, posle izvesnog vremena, aktivira ekran i napiše READY. To je znak da je program korektno snimljen i uspešno verifikovan.

Sledeća mogućnost je da se, u toku učitavanja programa, ekran osvetli sa porukom WHAT? i uobičajenim READY. To znači da je računar upravo primetio neslaganje onoga što je u memoriji i onoga što je na kaseti. Možemo da pokušamo još jednu verifikaciju povećavajući nivo zvuka ili ga smanjujući, ali je vrlo verovatno da ćemo morati ponovo da snimamo program.

Treća i najneprijatnija mogućnost je da se ne dogodi ništa — ekran je neprekidno crn iako je prošlo daleko više vremena nego što je trajalo snimanje programa (ili, ako imamo kasetofon sa brojačem, predeno je mesto na kome se nalazi kraj programa). To znači da je nivo zvuka bio preslab pa računar nije pronašao početak programa ili da, usled neke pogrešne manipulacije sa utičnicama, ništa nije ni snimljeno na traku. Pritisnimo dugme RESET koji nalazi sa zadnje strane računara i primetimo (kucanjem naredbe LIST) da je naš program i dalje u memoriji. Možemo da pokušamo novu verifikaciju ili, po potrebi, novo snimanje.

Treća opcija pri radu sa kasetofonom je unošenje programa sa trake i verovatno smo je savladali pri korišćenju demo-kasete. Povežimo računar sa kasetofonom na način opisan kod opcije OLD?, otkucajmo OLD RET i startujmo traku (taster PLAY). Posle izvesnog vremena, program će biti učitan i na ekranu će pisati READY. Ukoliko je to jedini prisutni komentar, program je vrlo verovatno ispravno upisan i možemo da ga startujemo sa RUN. No, ako se pre READY našlo WHAT?, računar je primetio grešku pri upisu. Možemo da otkucamo LIST (ne zaboravimo da listanje programa traje samo dok držimo pritisnut taster RET) i pogledamo na šta program liči: sasvim je moguće da su grupe slova promenjene ili ispuštenе. Najbolje je da u ovakvom slučaju pokušamo upisivanje još jedanput, menjajući jačinu tona u nadi da će sve biti u redu. Ukoliko ne uspemo, ostaje nam da pokušamo da ispravimo program što može da bude vrlo neprijatno, pogotovu ako nismo njegov autor. Iz iskustva možemo da kažemo da se problemi ovoga tipa ne javljaju kod verifikovanih programa, osim kada su snimani na jednom a učitavani sa drugog kasetofona. No, ako nam je neki program važan, želećemo da eliminisemo sve moguće izvore problema: u tom slučaju je korisno da snimimo program dva puta i to po mogućnosti na različite kasete. Obzirom da snimanje oduzima svega 2 minuta po kilobajtu, ova mera predostrožnosti neće predstavljati veliki gubitak vremena.

AT	4	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272	273	274	275	276	277	278	279	280	281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300	301	302	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319	320	321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336	337	338	339	340	341	342	343	344	345	346	347	348	349	350	351	352	353	354	355	356	357	358	359	360	361	362	363	364	365	366	367	368	369	370	371	372	373	374	375	376	377	378	379	380	381	382	383	384	385	386	387	388	389	390	391	392	393	394	395	396	397	398	399	400	401	402	403	404	405	406	407	408	409	410	411	412	413	414	415	416	417	418	419	420	421	422	423	424	425	426	427	428	429	430	431	432	433	434	435	436	437	438	439	440	441	442	443	444	445	446	447	448	449	450	451	452	453	454	455	456	457	458	459	460	461	462	463	464	465	466	467	468	469	470	471	472	473	474	475	476	477	478	479	480	481	482	483	484	485	486	487	488	489	490	491	492	493	494	495	496	497	498	499	500	501	502	503	504	505	506	507	508	509	510	511	512	513	514	515	516	517	518	519	520	521	522	523	524	525	526	527	528	529	530	531	532	533	534	535	536	537	538	539	540	541	542	543	544	545	546	547	548	549	550	551	552	553	554	555	556	557	558	559	560	561	562	563	564	565	566	567	568	569	570	571	572	573	574	575	576	577	578	579	580	581	582	583	584	585	586	587	588	589	590	591	592	593	594	595	596	597	598	599	600	601	602	603	604	605	606	607	608	609	610	611	612	613	614	615	616	617	618	619	620	621	622	623	624	625	626	627	628	629	630	631	632	633	634	635	636	637	638	639	640	641	642	643	644	645	646	647	648	649	650	651	652	653	654	655	656	657	658	659	660	661	662	663	664	665	666	667	668	669	670	671	672	673	674	675	676	677	678	679	680	681	682	683	684	685	686	687	688	689	690	691	692	693	694	695	696	697	698	699	700	701	702	703	704	705	706	707	708	709	710	711	712	713	714	715	716	717	718	719	720	721	722	723	724	725	726	727	728	729	730	731	732	733	734	735	736	737	738	739	740	741	742	743	744	745	746	747	748	749	750	751	752	753	754	755	756	757	758	759	760	761	762	763	764	765	766	767	768	769	770	771	772	773	774	775	776	777	778	779	780	781	782	783	784	785	786	787	788	789	790	791	792	793	794	795	796	797	798	799	800	801	802	803	804	805	806	807	808	809	8010	8011	8012	8013	8014	8015	8016	8017	8018	8019	8020	8021	8022	8023	8024	8025	8026	8027	8028	8029	8030	8031	8032	8033	8034	8035	8036	8037	8038	8039	8040	8041	8042	8043	8044	8045	8046	8047	8048	8049	8050	8051	8052	8053	8054	8055	8056	8057	8058	8059	8060	8061	8062	8063	8064	8065	8066	8067	8068	8069	8070	8071	8072	8073	8074	8075	8076	8077	8078	8079	8080	8081	8082	8083	8084	8085	8086	8087	8088	8089	8090	8091	8092	8093	8094	8095	8096	8097	8098	8099	80100	80101	80102	80103	80104	80105	80106	80107	80108	80109	80110	80111	80112	80113	80114	80115	80116	80117	80118	80119	80120	80121	80122	80123	80124	80125	80126	80127	80128	80129	80130	80131	80132	80133	80134	80135	80136	80137	80138	80139	80140	80141	80142	80143	80144	80145	80146	80147	80148	80149	80150	80151	80152	80153	80154	80155	80156	80157	80158	80159	80160	80161	80162	80163	80164	80165	80166	80167	80168	80169	80170	80171	80172	80173	80174	80175	80176	80177	80178	80179	80180	80181	80182	80183	80184	80185	80186	80187	80188	80189	80190	80191	80192	80193	80194	80195	80196	80197	80198	80199	80200	80201	80202	80203	80204	80205	80206	80207	80208	80209	80210	80211	80212	80213	80214	80215	80216	80217	80218	80219	80220	80221	80222	80223	80224	80225	80226	80227	80228	80229	80230	80231	80232	80233	80234	80235	80236	80237	80238	80239	80240	80241	80242	80243	80244	80245	80246	80247	80248	80249	80250	80251	80252	80253	80254	80255	80256	80257	80258	80259	80260	80261	80262	80263	80264	80265	80266	80267	80268	80269	80270	80271	80272	80273	80274	80275	80276	80277	80278	80279	80280	80281	80282	80283	80284	80285	80286	80287	80288	80289	80290	80291	80292	80293	80294	80295	80296	80297	80298	80299	80300	80301	80302	80303	80304	80305	80306	80307	80308	80309	80310	80311	80312	80313	80314	80315	80316	80317	80318	80319	80320	80321	80322	80323	80324	80325	80326	80327	80328	80329	80330	80331	80332	80333	80334	80335	80336	80337	80338	80339	80340	80341	80342	80343	80344	80345	80346	80347	80348	80349	80350	80351	80352	80353	80354	80355	80356	80357	80358	80359	80360	80361	80362	80363	80364	80365	80366	80367	80368	80369	80370	80371	80372	80373	80374	80375	80376	80377	80378	80379	80380	80381	80382	80383	80384	80385	80386	80387	80388	80389	80390	80391	80392	80393	80394	80395	80396	80397	80398	80399	80400	80401	80402	80403	80404	80405	80406	80407	80408	80409	80410	80411	80412	80413	80414	80415	80416	80417	80418	80419	80420	80421	80422	80423	80424	80425	80426	80427	80428	80429	80430	80431	80432	80433	80434	80435	80436	80437	80438	80439	80440	80441	80442	80443	80444	80445	80446	80447	80448	80449	80450	80451	80452	80453	80454	80455	80456	80457	80458	80459	80460	80461	80462	80463	80464	80465	80466	80467	80468	80469	80470	80471	80472	80473	80474	80475	80476	80477	80478	80479	80480	80481	80482	80483	80484	80485	80486	80487	80488	80489	80490	80491	80492	80493	80494	80495	80496	80497	80498	80499	80500	80501	80502	80503	80504	80505	80506	80507	80508	80509	80510	80511	80512	80513	80514	80515	80516	80517	80518	80519	80520	80521	80522	80523	80524	80525	80526	80527	80528	80529	80530	80531	80532	80533	80534	80535	805

## 10. Dalje mogućnosti naredbe PRINT, PRINT AT, HOME

Na samom početku smo rekli da će ovo uputstvo biti organizovano „po krugovima“: prvi krug predstavlja upoznavanje tastature i rad sa demo-kasetom, drugi osnove programiranja a treći, u koji upravo ulazimo, „lukusužnije“ naredbe i njihove verzije. Polazimo, naravno, ponovo od PRINT.

Do sada smo iza PRINT stavljali tekst pod znacima navoda ili neki brojni izraz koji je izračunavan i prikazivan na ekranu. Na taj način smo u jednom redu mogli da ispišemo samo jednu rečenicu što je, ako ništa drugo, neestetski. U okviru jedne naredbe PRINT može da se nađe više promenljivih, izraza i komentara koji će biti prikazani u istom redu. Računar, međutim, zahteva da ovi komentari budu odvojeni jedan od drugog znakom ; ili običnim zarezom. Ukoliko su dve stavke u naredbi PRINT odvojene simbolom ; računar će ih nadovezati jednu na drugu, a ako je upotrebljen zarez — razmaknuće ih tako da druga počinje od osmog, šesnaestog ili dvadeset četvrtog mesta u redu. Ovu naredbu ćemo najbolje shvatiti na primeru.

Vratimo se na program za izračunavanje vremena putovanje između dva mesta i obrišimo linije 40, 50, 70 i 80 a zatim otkucajmo:

70 PRINT INT(T); „ČASOVA“ ;M; „MINUTA.“

Izvršimo program i unesimo uobičajene podatke za rastojanje Beograd—Duborovnik i brzinu 800 km/h. Na ekranu će se pojaviti izveštaj:

0ČASOVA, 27MINUTA.

Nije baš sjajno — redovi više nisu odvojeni, ali je računar očigledno preterao sa zbijanjem. Ispred svakog broja je, doduše, ostavljao po jedno prazno mesto (to mesto, u stvari, nije prazno — na njega bi bio smešten znak minus da je broj bio negativan), ali nam je neophodan blanko simbol i iza njega. Jedan od načina da rešimo problem je da otkucamo:

70 PRINT INT(T), „ČASOVA“, ;M, „MINUTA.“

ali će tako rečenica biti nelogično mnogo izdeljena na polja. Pravo rešenje je naredba:

70 PRINT INT(T); „ČASOVA“; ;M; „MINUTA.“

Upoznavanjem simbola , i ; još nismo iscrpli sve mogućnosti naredbe PRINT: ona može da posluži i za pisanje teksta na proizvolnjem delu ekranu. Da bi ovakvo pisanje imalo nekog smisla, treba najpre da upoznamo naredbu za brisanje čitavog sadržaja ekranu. Ovaj efekat se u komandnom modu postiže istovremenim pritiskom na tastere [SHIFT] i [DEL], a u programskom — izvršavanjem naredbe HOME (HOME nalaze računaru da „vrati kurzor kući“ koja se nalazi u levom gornjem uglu ekranu. Pri tom se čitav sadržaj ekranu briše).

Na slici je prikazana mapa ekranu. Kao što se vidi, on je podeljen na 16 redova, a svaki od tih redova na 32 mesta. Mesta su numerisana brojevima od 0 (karakter u levom gornjem uglu ekranu) do 511 (karakter u desnom donjem uglu). Naredba PRINT AT može da počne da štampa od proizvoljnog mesta koje je navedeno iza AT; pun oblik naredbe je, dakle, PRINT AT 32, „RAČUNAR GALAKSIJA“ — u ovom primeru tekst „računar galaksija“ se pojavljuje na početku drugog reda, dok sadržaj prvog ostaje nepromjenjen.

Pri prikazivanju nekih komplikovanih tekstova treba posebno paziti na jednu stvar: ako najpre ispisujemo tekst u desnoj polovini neke linije, a zatim u levoj, neka PRINT AT naredba može lako da poremeti postojeći sadržaj nekog reda. Da se obezbedimo od toga, zavržavaćemo svaku naredbu PRINT znakom : (pišemo, na primer, PRINT AT 300, „RAČUNAR“;).

## 11. Crtanje po ekranu — DOT, UNDOT

Prethodno poglavlje nije razjasnilo sve oznake sa mape ekrana: tamo su redovi podjeleni na po tri dela, a kolone na po dva. Na ovaj način su dobijene takoe ili „pixeli“ kojih ima po 64 u svakom od 48 redova. Svaku ovu tačku (to je, zapravo, kvadratič) možemo slobodno da osvetlimo ili zatamnimo koristeći naredbe DOT i UNDOT (DOT znači tačka). Treba odmah da uočimo razliku između „tački“ i karaktera: tačka je osvetljena ili zatamnjena, dok su karakteri daleko raznovrsniji (sve slova, brojevi, interpunktivski znaci, prazno mesto ...); karakteri su adresirani brojevima od 0 do 511, a tačke na mnogo komforntniji način — za obraćanje svakoj navodimo red u kome se ona nalazi i njen položaj u tom redu; najzad, tačke su šest puta manje od karaktera, pa njima mogu da se dobijaju bolje slike na ekranu.

Naredbu DOT prate koordinate tačke koju treba osvetliti (DOT 0,0, na primer, osvetljava tačku u levom gornjem uglu, DOT 1,0 tačku desno od nje, DOT 31,23 tačku u sredini ekrana a DOT 63,47 tačku u desnom donjem uglu). Naredba UNDOT ima obrnuto značenje: ona zatamnjuje tačku čije su koordinate navedene na isti način. Program:

```
10 HOME  
20 DOT 31,23  
30 DOT 40,40  
40 UNDOT 31,23  
50 UNDOT 40,40  
60 GOTO 20
```

neprekidno uključuje i isključuje tačke čije su koordinate (31,23) i (40,40).

Naredba DOT može da se upotribe i unutar IF, ali tada ima sasvim drukčiji smisao: linije 100 IF DOT 10,10 GOTO 1000 ne osvetljava tačku 10,10 nego ispituje da li je ta tačka osvetljena i, ako jeste, prelazi na liniju broj 1000.

Ukoliko je neka tačka osvetljena, primena naredbe DOT na nju neće promeniti ovo stanje; slično tome, UNDOT neće promeniti status tačke koja je već zatamnjena. Naredbe DOT i UNDOT su veoma značajne i zasluzuju da ih upoznamo na dodatnim primerima. Ipak, sačekaćemo izvesno vreme obziru da je za ovakvu demonstraciju neophodno realizovati cikluse (petlje), a to ćemo biti u stanju da uradimo daleko bolje kada upoznamo grupu novih naredbi.

## 12. Ciklusi — FOR-TO-NEXT-STEP

Često je potrebno da se određeni skup instrukcija ponovi određen broj puta i da se, zatim, nastavi sa izvršavanjem programa. Pretpostavimo da nam je potreban program koji izračunava faktorijel nekog broja. Faktorijel se definše kao proizvod svih prirodnih brojeva zaključno sa brojem čiji se faktorijel traži (faktorijel broja 6 je, na primer,  $1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 = 720$ ). Potrebno je, dakle, da počnemo od broja 1 i množimo ga redom sa 2, 3, 4, ... sve do nekog n. Ovaj problem može, primenom naredbe IF, da se reši na sledeći način:

```
10 INPUT N  
20 Y=1  
30 I=1  
40 I=I+1  
50 Y=Y*I  
60 IF I<N GOTO 40: ELSE PRINT Y
```

Vidimo da je bilo neophodno uvesti naredbu koja inicijalizuje brojač (I=1), naredbu za povećavanje tog brojača kao i naredbu kojom ispitujemo da li je posao završen. Ovakvo rešenje, naravno, može da se primeni i u mnogo složenijim slučajevima, ali su takvi programi teži za sastavljanje (da smo, na primer, promenili redosled naredbi 40 i 50 u IF komandi bi morali da poređimo I sa N+1), manje pregledni i sporije rade. Naredba FOR

— TO i odgovarajuća naredba NEXT su pripremljene tako da olakšaju realizaciju ciklusa. Pre nego što predemo na njihov detaljan opis, videćemo kako se prethodni problem rešava njihovom primenom:

```
10 INPUT N
20 Y=1
30 FOR I=1 TO N
40 Y=Y*I
50 NEXT I
60 PRINT Y
```

Dobijeni program ima šest naredbi kao i prethodni ali, u celini gledano, zauzima manje memorije i brže se izvršava. Čak i ako vam je ovo prvi susret sa naredbama FOR—TO—NEXT, verovatno ste shvatili da se ciklus odvija između naredbi 30 i 50. Računar, po nailasku na naredbu 30, „inicira brojač“ odnosno dodeljuje promenljivoj I sadržaj 1 (da je naredba 30 glasila FOR I=2 TO N promenljivoj bi bio dodeljen sadržaj 2 i tome slično) i „pamti“ mesto na kome je ciklus počeo. Izvršavanje programa se nastavlja sasvim normalno sve dok računar ne nađe na naredbu NEXT I („sledeće I“). U tom trenutku on dodaje jedan promenljivoj I i ispituje da li se došlo do N. Ako nije, ciklus automatski počinje od početka. Ako jeste, naredba NEXT I se preskače i računar prelazi na izvršavanje naredbe 60.

U okviru naredbe FOR može da se nalazi i opcija STEP (korak). Da smo, umesto naredbe 30, otkucali FOR I=1 TO N STEP 2 računar bi, svaki put kada nađe na naredbu NEXT I, uvećavao sadržaj promenljive I za dva pa bi računao proizvod  $Y = 1 \cdot 3 \cdot 5 \cdot 7 \dots N$  (za slučaj da je N neparan broj ovo bi bila funkcija bez posebnog imena koja se obeležava sa  $N!!$ , a ako je N paran rezultat ne bi imao matematičko značenje. Da bi program računao proizvod  $2 \cdot 4 \cdot 6 \cdot 8 \dots N = N!!$  (za parne brojeve), naredbu 30 bi trebalo izmeniti tako da glasi FOR I=2 TO N STEP 2).

Nema nikakve prepreke da STEP bude negativan. Program:

```
10 FOR I=30 TO 0 STEP —1
20 PRINT I
30 NEXT I
```

će brojati od 30 do 0 i to unazad. Ipak, postoji jedno ograničenje vezano za naredbe FOR—NEXT—STEP koje treba imati u vidu ako u praksi verovatno neće izazvati nikakve probleme: konstante ili promenljive koje figurišu u ovim naredbama moraju da imaju vrednosti celih brojeva; ako nemaju, računar će sam „odseći“ decimalne. Naredba FOR I=0.5 TO 11.5 STEP 0.5 će se izvršavati nekorektno obzirom da će računar podrazumevati STEP 0.

## 13. Slučajni brojevi — RND

Svi rezultati koje računar daje su precizne funkcije ulaznih veličina. Računar „galaksija“ je opremljen jednom naredbom koja, na prvi pogled, odstupa od ove definicije: ona je namenjena generisanju slučajnih brojeva. Slučajni brojevi koje računar generiše su, u stvari, pseudoslučajni — rezultat su komplikovanih operacija nad sadržajima nekih sistemskih promenljivih i korisnik nema nikakvog načina da ih predviđi; za njega su, dakle, oni slučajni u dovoljnoj meri.

Otkucajmo NEW i unesimo jednostavni program:

```
10 PRINT RND
20 GOTO 10 i startujmo ga sa RUN. Na ekranu će velikom brzinom početi da se pojavljuju brojevi koje, istini za volju, nećemo stići ni da pročitamo. Došao je, dakle, pravi trenutak da otkrijemo poslednju tajnu tastature našeg računara. Pritisnimo taster DEL i držimo ga pritisnutog. Dogadaji na ekranu su se najednom „zamrzli“ — izgleda da je izvršavanje programa zaustavljeno. Prvi utisak ovde ne vara: izvršavanje programa je zainta zaustavljeno i taj prekid će trajati sve dok je taster DEL pritisnut; njegovim otpuštanjem program nastavlja da se izvršava. Treba odmah uočiti razliku između DEL i BRK: po
```

pritisku na **BRK** izvršavanje programa je trajno prekinuto i na ekranu piše READY, dok pritisak na **DEL** samo zaustavlja rad računara koji se nastavlja po otpuštanju tastera.

Kada znamo kako da prekinemo rad programa, možemo da se usresredimo na naredbu RND. Na ekranu se nalaze brojevi koje mi unapred ne možemo da predvidimo — oni su slučajan izbor vašeg računara. To su, u svakom slučaju, brojevi između 0 i 1, koji imaju po pet-šest decimala (ne treba se zbuniti ako neki od njih bude u eksponencijalnoj notaciji).

U praksi nam najčešće nisu potrebni decimalni brojevi između 0 i 1, nego celi brojevi između nekih a i b (ako, na primer, hoćemo da simuliramo kockicu za igru „čoveče, ne ljuti se“ trebaju nam slučajno izabrani brojevi od 1 do 6). Prava je prilika da se podsetimo naredbe INT i naučimo formulu  $R=INT(RND*(B-A)+A)$ . Njenom primenom računar generiše cele slučajne brojeve između A i B (ovi brojevi mogu da budu jednak A ali ne mogu da budu jednak B; matematičari bi rekli da se ovi brojevi nalaze u intervalu [A,B]).

Sastavimo program koji bi demonstrirao naredbe DOT i UNDOT. Najpre ćemo da obrišemo ekran, a zatim osvetljavati slučajne tačke. Ukoliko nađemo na tačku koja je već osvetljena, zatamnićemo je:

```
10 HOME
20 X=INT (RND*64)
30 Y=INT (RND*48)
40 IF DOT X, Y UNDOT X,Y: ELSE DOT X,Y
50 GOTO 20
```

Ako imate problema sa razumevanjem programa, moraćete da se vratite na poglavlje „crtanje po ekranu“ i „naredba IF“. Ako ga razumete, pokušajte da ga dopunite tako da se u desnom gornjem uglu ekrana neprekidno ispisuje broj osvetljenih tačaka. Ukoliko je generator slučajnih brojeva dobar, posle izvesnog vremena bi trebalo da se uspostavi „dinamička ravnoteža“ — stanje u kome se približno jednak broj tačaka osvetljava i zatamnjuje u nekom intervalu.

## 14. Potprogrami — CALL i RETURN

U mnogim programima se javljaju grupe instrukcija koje se ponavljaju. Računar „galaksija“, na primer, nije opremljen logaritamskom funkcijom, pa će mnogi poželeti da je dodaju. Oni će, dakle, koristeći neku numeričku formulu ili iterativni algoritam sastaviti program koji izračunava  $\log x$  za zadato  $x$  i poželeti da ga koriste u mnogim prilikama. Ako, međutim, navode čitav skup instrukcija svaki put kada im logaritam zatreba, neće daleko stići: izdaće ih strpljenje ili memorija računara. Zato se ovakav skup instrukcija odvaja u poseban program koji se iz glavnog programa poziva kao potprogram.

Potprogram se ne započinje ni na kakav poseban način — bitno je samo zapamtiti broj njegove prve linije. Na kraju potprograma mora da se nađe naredba RETURN (=vratiti se) koja nalaže računaru da nastavi sa izvršavanjem glavnog programa.

U glavnom programu se, kada je god potprogram potreban, piše naredba CALL koju prati broj prve linije potprograma (napr. CALL 1000). Sve će biti mnogo jasnije kada pogledamo primer. Sastavimo program koji pravom linijom spaja tačke čije su koordinate zadate, a zatim ćemo ga upotrebiti kao potprogram da uokvirimo ekran. Evo, najpre, potprograma koji je nešto pojednostavljen pretpostavkom da su dve tačke koje treba spajati nalaze na istoj horizontali ili vertikali. Ulazne veličine su brojevi X, Y, Z i T koji daju koordinate prve (X,Y) odnosno druge (Z,T) tačke; obzirom na uvedeno ograničenje mora da bude  $X=Z$  ili  $Y=T$ .

```
100 IF X=Z FOR I=Y TO T: DOT X,I:NEXT I: ELSE FOR I=X TO Z:
DOT I,Y:NEXT I
110 RETURN
```

U potprogramu, očigledno, ima i novina: napisana je jedna linija koja se sastoji od više naredbi razdvojenih znakom : Na ovaj način možemo da postupamo kada god želimo da štedimo memoriju ili kada nam se čini da grupu naredbi treba „spakovati“ u isti red pošto predstavljaju logičnu celinu. Postoje, ipak, dva razloga zbog kojih čitav program ne može

da bude smešten u jednu programsku liniju: maksimalna dužina linije koju računar „galaksija“ može da primi je četiri reda, a naredba koja se ne nalazi na početku neke linije ne može da se dosegne pomoću naredbi GOTO i CALL.

Evo, najzad, i programa koji će uokviriti ekran koristeći dati potprogram:

```
5 HOME
10 X=0; Y=0; Z=63; T=0; CALL 100
20 Y=47; T=47; CALL 100
30 X=63; Y=0; CALL 100
40 X=0; Y=0; Z=0; CALL 100
```

Vidi se da pre svakog poziva potprograma nisu postavljeni sadržaji svih promenljivih. Razlog je u tome što se ovi sadržaji u toku potprograma ne menjaju, pa nema potrebe ponavljati iste naredbe i razbacivati memoriju.

Testiranje programa (koji je, jasno, povezan sa potprogramom) u prvom trenutku prolazi povoljno: računar uokviruje ekran. No, posle toga nastupa greška i to u naredbi RETURN. Zašto? Računar je četiri puta pozvao i izvršio potprogram i tako završio naredbu 40. Posle toga je, jednostavno, počeo sa izvršavanjem naredbe 100 koja sledi iza nje, još jednom povukao istu liniju i našao na RETURN. Pošto potprogram nije pozivan, računar nije znao kuda da se vrati (RETURN = vrati se) pa je prijavio grešku.

Prva modifikacija u cilju rešavanja ovog problema je da se doda naredba 50 STOP (sećate li se naredbe STOP?). Solucija je dobra samo na prvi pogled: kada računar nađe na STOP, on prestaje sa izvršavanjem programa i izdaje poruku READY kvareći tako jedan deo teškom mukom nacrtanog okvira. Zato ćemo dodati naredbu 50 GOTO 50 koja navodi računar da se neprekidno vrti u „mrtooj petlji“ ne kvareći izgled ekranu (ovo je takozvano „dinamičko zaustavljanje“). Tako možemo da se divimo uokvirenom ekranu i pritisnemo taster **BRK** tek kada nam on dosadi.

U okviru potprograma može da se pozove sledeći potprogram, baš kao što u okviru jednog ciklusa (petlje) može da se započne drugi, ali je broj ovakvih poziva ograničen na 13. U praksi se sigurno neće dogadati da vam je potrebno više od trinaest nivoa potprograma, ali morate da pazite na početničke greške poput ove:

```
10 CALL 100
...
100 A= ...
150 GOTO 10
```

Na ovaj način računar neće nikada nailaziti na naredbu RETURN, pa će biti u situaciji da stalno poziva nove i nove potprograme sve dok ne dode do greške. Ovakvu grešku je često teško otkriti: program dobro radi jedno vreme a onda, u nepravilnim intervalima, na ekranu ugledamo poruku HOW?

## 15. Ispravljanje programa — EDIT

Jedini način za ispravljanje grešaka u programu koji smo do sada upoznali bio je prekucavanje linije što je neekonomična i neobično nervirajuća solucija (u to ste se i sami uverili, pogotovu ako ste pisali neki program koji koristi naredbu PRINT AT). Računar „galaksija“ poseduje i specijalnu komandu EDIT koja je namenjena ispravkama. Ona ne sme da se nađe u programu (blisko je pameti da programi mogu da se edituju samo u komandnom modu) i treba da je prati linijski broj naredbi koja se edituje. Pritisom na **RET** računar briše ekran i u njegovim prvim redovima ispisuje liniju koju smo poželeli da menjamo. Kurzor se nalazi na njenom kraju. Koristeći tastere ← i →, možemo da ga pokrećemo u okviru naredbe sve dok ne dodemo do mesta na kome treba da ispravimo grešku. Ako želimo da izbacimo slovo koje se nalazi desno od kurzora, treba jednostavno da pritisnemo taster **DEL**. Ukoliko želimo da ubacimo novi deo programske linije, jednostavno ga otkucamo i on će da bude umetnut počevši od tekuće pozicije kurzora. Pri pomeranju kurzora i brisanju pojedinih delova programske linije, taster **REPT** (autorepeat) dolazi do punog izražaja.

Kada smo završili sa ispravkama neke linije (nema nikakve prepreke da, u toku ispravki, nekoliko puta menjamo isti deo linije), možemo da pritisnemo **RET** i tako se saglasimo sa izvršenim izmenama ili da pritisnemo **BRK** i tako naložimo računaru da ignoriše izmene i ostavi liniju u stanju u kom je bila pre nego što je editovanje započeto.

Pri editovanju se pojavljuje i jedan artefakt: nema prepreke da promenimo i sam broj programske linije pod uslovom da u okviru njega ne ostavljamo razmake (broj linije 10 02 nije dopušten). Kada pritisnemo **RET**, računar će u memoriju uneti novu programsku liniju dok stara neće biti ni promenjena ni izbrisana. Ovo pruža mogućnost „umnožavaњa“ jedne naredbe u okviru programa bez potrebe da se ona svaki put iznova kuca.

## 16. Numerički niz A (I)

Do sada smo upoznali promenljive A, B, C, ... Y i Z pomoću kojih smeštamo podatke u memoriju računara. Ovi podaci su međusobno logički odvojeni: promenljiva V, na primer, može da čuva podatak o brzini nekog objekta, F o sili koja deluje na njega, T o vremenu i slično. Ponekad smo, međutim, suočeni sa potrebom da obradimo grupu međusobno srodnih podataka. Želimo, na primer, da sortiramo podatke po veličini. Besmisleno je da koristimo promenljive A, B, ... obzirom da bismo u programu morali da navedemo veliku grupu naredbi koja bi poredila sadržaj svake promenljive sa sadržajima svih ostalih. Osim toga, šta da radimo ako treba da sortiramo više od 26 podataka?

Računar „galaksija“ može da radi sa numeričkim nizom A(I) (ne treba ga mešati sa promenljivom A). Elementi ovog niza su A(0), A(1), A(2) itd — njihov broj je ograničen jedino veličinom memorije. Iako ćemo mapu memorije objaviti tek docije, potrebno je da već sada u grubim crtama objasnimo razmeštaj elemenata niza A(I) da bi se shvatila njegova promenljiva veličina. Zamislimo da se od početka memorije prema kraju smešta BASIC program a od kraja memorije prema početku elementi niza A(I). Svaki element (A(0), A(1), ...) zauzima po četiri bajta. Jasno je da se, negde na sredini memorije, niz A(I) i BASIC program „susreću“. Računar smatra da je BASIC program značajniji i ograničava niz A(I) tako da popunjava samo slobodan deo memorije. Ukoliko pokušamo da menjamo vrednost nekog elementa niza A(I) koji bi promenio bežik program, računar će prekinuti sa radom i izdati poruku **SORRY**.

Kako da saznamo sa koliko elemenata niza A(I) raspolažemo? Jedan od načina je da otkucamo jednostavan program:

```
10 I=0
20 A(I)=A(I)
30 I=I+1
```

40 GOTO 20 i startujemo ga sa RUN. Kada se na ekranu pojavi poruka **SORRY**, otkucamo PRINT I i saznamo broj prvog nepostojecog elementa niza. Ovaj način nije naročito dobar, ponavljajući zato što u postojeći bežik program treba da dodajemo nove naredbe koje ga nepotrebno produžavaju i tako smanjuju broj članova niza. Zato je dizajnirana naredba nazvana **MEM**. PRINT MEM, na primer, prikazuje na ekranu broj bajtova memorije koji su slobodni da prime niz A(I). Kako svaki element ovog niza zauzima po četiri bajta, PRINT MEM/4 će dati maksimalan broj elemenata koji nam stoji na raspolaaganju.

Evo, na posletku, i programa koji će olakšati razumevanje rada sa nizom A(I). Program uređuje grupu brojeva u rastući niz i prikazuje dobijene rezultate. Korišćen je umereno dobar algoritam (postoje, naravno, i daleko bolji, ali su manje razumljivi za početnike), pa program unekoliko ilustruje brzinu rada računara „galaksija“.

```
10 PRINT „KOLIKO IMA BROJEVA“: INPUT N
20 FOR I=1 TO N: INPUT A(I): NEXT I
30 FOR I=1 TO N-1: FOR J=I+1 TO N
40 IF A(I)<A(J) GOTO 60
50 P=A(I):A(I)=A(J): A(J)=P
60 NEXT J: NEXT I
70 FOR I=1 TO N: PRINT A(I),A(I): NEXT I
```

## 17. Rad sa alfanumericima — CHR\$, EQ, VAL, PTR ARR\$

Do sada smo radili isključivo sa brojevima, dok smo komentare pisane standardnim srpskohrvatskim jezikom umetali isključivo između navodnika u PRINT naredbama i iza znakal. Računar „galaksija“, međutim, poseduje i mogućnost rada sa nizovima znakova (pod znakom ovde podrazumevamo slovo, broj, interpunkcijski znak i prazan prostor) ili, kratko rečeno, alfanumericima.

Poput brojeva, za smeštanje reči koristimo promenljive koje se nazivaju alfanumeričkim. U programu se prepoznaju po tome što se iza imena svake nalazi oznaka za dolar (\$). U svaku alfanumeričku promenljivu može da se smesti tekst koji ima najviše 16 znakova, računajući i razmake, interpunkcijske znakove i slično.

Postoje samo dve fiksne anumeričke promenljive koje se nazivaju X\$ i Y\$. Njima možemo da dodeljujemo vrednosti korišćenjem standardnih bejzik naredbi, pri čemu ne smemo da zaboravimo da sadržaj uvek (osim kod izvršavanja naredbe INPUT) stavljamo između znakova navoda. Možemo, na primer, da napišemo: X\$=„ALFANUMERIK“, a zatim PRINT X\$: na ekranu će da se pojavi reč alfanumerik.

Pored uobičajenog prenošenja sadržaja jedne promenljive u drugu (naredba X\$=Y\$), raspolažemo i funkcijom CHR\$. Ovu funkciju prati argument u zagradi koji je jednostavno neki broj između 0 i 255. Ovaj broj dobijamo iz tablice koja je data na sledećoj slici. Tablicu sličnu ovoj možete da dobijete na ekranu i ako izvršite sledeći jednostavni BASIC program:

**TABLICA ASCII KARAKTERA**

32	43	+	54	6	65	A	76	L	87	W	
33	44	,	55	7	66	B	77	M	88	X	
34	45	-	56	8	67	C	78	N	89	Y	
35	46	.	57	9	68	D	79	O	90	Z	
36	47	/	58	:	69	E	80	P	91	Ć	
37	48	0	59	:	70	F	81	Q	92	Ć	
38	49	1	60	<	71	G	82	R	93	Ž	
39	spec	50	2	61	=	72	H	83	S	94	Š
40	(	51	3	62	>	73	I	84	T	95	-
41	)	52	4	63	?	74	J	85	U		
42	*	53	5	64	spec	75	K	86	V		

```
10 FOR I=32 TO 255
20 PRINT I;";CHR$(I),
30 NEXT I
```

Pogledajmo kako to izgleda na primeru. Naredba PRINT CHR\$(65) će, na primer, pokazati slovo A na ekranu. Funkciju CHR\$ koristimo pri formiraju složenih alfanumerika koje se obavlja sabiranjem.

Alfanumerici, jasno, nisu brojevi koje ima smisla sabirati. No, iako se za sabiranje alfanumerika koristi isti znak kao i za sabiranje brojeva (plus), pojam ima sasvim drukčiji smisao: alfanumerici se nadovezuju. Ako, na primer, napišemo X\$=„GALAKSIJA“ + CHR\$(75) + „SIJA“ a zatim PRINT X\$, na ekranu će se pojavit reč GALAKSIJA obzirom da CHR\$(75) odgovara slovu K.

Na prvi pogled bi se reklo da naredbe poput oornie nemaju nekoj velikog smisla: zar nije jednostavnije napisati X\$=„GALAKSIJA“? No, posmatrajmo naredbu X\$=CHR\$(34) + „GALAKSIJA“ + CHR\$(34). Po njenom izvršavanju PRINT X\$ daje reč „GALAKSIJA“

koje se nikako drukčije nije mogla dobiti obzirom da znaci navoda ne mogu da budu deo nekog teksta pošto označavaju njegov početak i kraj.

U okviru naredbe  $X\$ = \dots$  može da se nalazi promenljiva  $Y\$$  pa čak i sama promenljiva  $X\$$ , ali uz jedno ograničenje: ona mora da bude sasvim na početku izraza. Dakle, naredba  $X\$ = X\$ + Y\$$  će nadovezati sadržaj promenljive  $Y\$$  na promenljivu  $X\$$  ali će  $X\$ = Y\$ + X\$$  izazvati neželjeno ponalaženje (probajtel). Slično, može da se napiše  $Y\$ = Y\$ + CHR\$60\$ + X\$$  ali ne i  $Y\$ = CHR\$99\$ + Y\$ + X\$$ .

Ako nam dve alfanočničke promenljive nisu dovoljne, možemo da definišemo alfanumerički niz koji se zove  $X\$()$  (ne treba ga, ponovo, mešati sa promenljivom  $X\$$ ). Na početku programa treba da stavimo naredbu  $ARR\$()$  gde je  $n$  broj elemenata niza  $X\$()$  koji nam je potreban (svaki od tih elemenata može da se upotribe za čuvanje 16 znakova), a zatim možemo da postavljamo i menjamo sadržaje nekih od tih elemenata, koristeći ranije izložena pravila za sabiranje (nadovezivanje) alfanumeričkih funkcija  $CHR\$$ .

Ponekad može da nam zatraže da uporedimo neka dva aumerika da bi saznali da li su oni jednak. Afanumerici se smatraju jednakim ako sadrže iste znakove na istim mestima i ako su jednake dužine. U tom smislu je „ABC“ = „ABC“ ali nije „ABC“ = „ABC“. Za poređenje alfanumerika koristimo naredbu  $EQ$ .

Možemo, na primer, da napišemo:

100 IF EQ  $Y\$, X\$(10)$  GOTO 1000: ELSE PRINT „RAZLIČITI SU“ i računar će, ako su sadržaji promenljivih  $X\$(10)$  i  $Y\$$  jednaki, preći na naredbu 1000, a ako nisu prikazati na ekranu RAZLIČITI SU i nastaviti sa radom. Možemo, jasno, da poređimo pomežljive  $X\$$  i  $Y\$$ , bilo koja dva elementa alfanumeričkog niza  $X\$()$  ili bilo koju kombinaciju.

Za svakog početnika je prilično teško da pravi razliku između numeričkih i alfanumeričkih promenljivih. Naredba  $X\$ = „123“$  je, na primer, sasvim različita od naredbe  $X = 123$ . U prvom slučaju smo dobili niz znakova koji ne podleže aritmetičkoj obradi (ako izuzmemo pomenuto nadovezivanje), a u drugom operativni broj 123. Promenljive  $X\$$  i  $X$  su sasvim različite i računar ih čuva na raznim mestima u memoriji.

U nekim slučajevima može da bude korisno da se neki alfanumerički koji se sastoji od cifara pretvoriti u broj. Za to nam služi naredba  $VAL$ . Otkucajmo itavan program: jednoetavan program:

```
10 INPUT X$  
20 PRINT VAL(PTR X$)+1 i startujmo ga sa RUN. Kada računar zatraži od nas da otkucamo  $X\$$ , odgovorimo sa 25 (posle toga, naravno, pritiskamo RET). Trenutak docnije na ekranu se pojavi broj 26. Računar je, naime, našao na naredbu PRINT, našao VAL (vrednost) afanumeričke promenljive  $X\$$  i dodao joj jedan, dobivši rezultat 26. Napominjemo da sintaksa naredbe  $VAL$  koja je ovde korišćena ne može da bude jasna čitaocu koji nije čitao „na preskok“ jer se u njoj koristi naredba PTR koju još nismo obradili. Za sada je, ipak, primite onakvu kakva jeste, docnije će sve postati jasnije.
```

Kada unosimo  $X\$$ , ne moramo da se ograničimo samo na brojeve: možemo kucamo čitav izraz, koristeći imena promenljivih i pazeći da on nema više od 16 simbola. Naredba  $VAL$  (PTR  $X\$$ ) će izračunati vrednoet ovog izraza i pretvoriti ga u broj.

## 18. Časovnik — DOT \*, UNDOT \*

Računar „galaksija“ je opremljen vrlo preciznim kvarcnim časovnikom koji će, po svoj prilici, naći upotrebu kao štoperica. Ovaj časovnik može, ako mu to naredimo, da prikazuje samo sate i minute, ali može da bude i toliko precizan da prikazuje i pedesetinke sekunde. Za rad sa njim je potrebna promenljiva  $Y\$$  i naredbe DOT\* i UNDOT\*.

Najpre treba da postavimo tekuće vreme. Napišemo (u komandnom modu ili u programu)  $Y\$ = „21:45:20:50“$  i pritisnimo RET (ako želimo da časovnik pokazuje samo sate i minute, napisaćemo  $Y\$ = „21:45“$  a ako želimo sate, minute i sekunde:  $Y\$ = „21:45:20“$ ; ukoliko želimo da merimo vreme, napisaćemo  $Y\$ = „00:00:00:00“$ ). Računar će smatrati da je sada tačno 21 čas, 45 minuta, 20 sekundi i 50 stotinki, ali njegov časovnik još neće započeti da radi: treba da ga startujemo sa DOT\*. Ako poželimo da zaustavimo časovnik, upotrebimo naredbu UNDOT\*. Evo, na primer, programa koji u desnom gornjem uglu ekranu neprekidno prikazuje tekuće vreme:

```
10 HOME:Y$=,,10:15:20:15" (ili bilo koje drugo vreme)
20 DOT*
30 PRINT AT 20;Y$;
40 GOTO 30
```

Ovako rešen časovnik ima tri mane: prva je što se isključivanjem računara briše tekuće vreme, druga što časovnik ne radi dok se program snima ili učitava sa kasete, a treća što, ako je u programu korišćena naredba DOT\*, moramo da pazimo da li su dve tačke treći simbol promenljive Y\$. Ako jesu, računar će smatrati da Y\$ sadrži tekuće vreme, pa će menjati njen sadržaj što može da proizvede neželjene posledice.

## 19. Alternativni metod primanja podataka — TAKE i #

Do sada smo primali podatke isključivo pomoću naredbe INPUT, a dodeljivali sadržaje pomenljivima i pomoću komandi tipa A=123. Često nam je, međutim, u početku programa neophodno da dodelimo vrednosti mnogobrojnim promenljivima i elementima matrice A. U takvim slučajevima naredba TAKE može da bude neobično praktična.

Iza TAKE se navodi lista promenljivih kojima treba dodeliti vrednosti. To mogu da budu numeričke promenljive, elementi matrice A, alfanumeričke promenljive ili elementi niza XS (ako je ovaj prethodno definisan sa ARR\$). Elementi ove liste se odvajaju zarezima. U početku programa će se, dakle, često naći naredba:

```
10 TAKE A,B,C,X,Z
```

Ali, odakle računar može da uzme (TAKE=uzmi) ove podatke? U okviru programa mora na proizvoljnom mestu (pre ili posle naredbe TAKE, po želji) da se nađe jedna ili više linija koje počinju simbolom #. Ovaj simbol obaveštava računar da se u linijama koje ga sadrže nalazi tablica podataka koji će biti pročitani sa TAKE. Brojevi i reči u ovim tabelama treba da budu razdvojeni zarezima.

Ako se, na primer, u programu nalazi naredba:

```
100#10, 20, 30, -11, 79, 779, a računar izvrši ranije navedenu TAKE naredbu, promenljiva A će dobiti vrednost 10, promenljiva B dvadeset itd. promenljiva Z će dobiti vrednost 79. Računar će „zapamtitи“ dokle je došao sa čitanjem podataka pa će, ako nađe na naredbu 50 TAKE L, promenljivoj L dodeliti broj 779.
```

Ako se u listi iza # nalazi manje podataka nego što se u TAKE naredbi traži, računar će prijaviti grešku. Obrnuto, ukoliko se ne učitaju svi podaci iz liste, računar neće reagovati, što je i prirodno.

Ponekad može da bude potrebno da se u okviru programa ponovo uzimaju isti podaci (potrebne su, na primer, početne vrednosti za neku igru, a želimo da se, po završetku svake partije, automatski započne nova sa istim početnim parametrima). U tom slučaju, pre prve naredbe TAKE koja se nalazi u programu treba da stavimo TAKE 0 (nula a ne slovo o). Na taj način će računar započeti uzimanje podataka od samog početka prve # liste. Umesto nule, iza TAKE možemo da stavimo neki drugi broj. U tom slučaju računar pronađe liniju sa tim brojem i priprema se da u sledećoj TAKE naredbi uzme podatke iz # liste koja se nalazi u toj liniji. Ukoliko u navedenoj naredbi nema # liste, računar će pronaći prvu sledeću takvu listu, odakle vidimo da je TAKE 0 samo specijalni slučaj naredbe TAKE n. Naredba RUN automatski izvršava TAKE 0.

## 20. Naredba KEY

U nekim igrama su potrebne brze reakcije igrača i naredba INPUT postaje sasvim nepogodna. INPUT A, na primer, prekida rad programa sve dok korisnik ne unese neki

broj i ne pritisne RET, što znači da je vreme za razmišljanje neograničeno. Osim toga, naredba INPUT ispisuje na ekranu znake koje korisnik kuca i tako kvari sliku koja je na njemu prikazana. Zato računar „galaksija“ poseduje naredbu KEY.

#### NAREDBA KEY (n)

TASTER N	TASTER N	TASTER N	TASTER N	TASTER N	TASTER N	TASTER N	TASTER N
A 1	I 9	Q 17	Y 25	1 33	9 41		
B 2	J 10	R 18	Z 26	2 34	:	42	RPT 50
C 3	K 11	S 19	‡ 27	3 35	:	43	
D 4	L 12	T 20	↔ 28	4 36	+	44	LIST 52
E 5	M 13	U 21	↔ 29	5 37	=	45	SHIFT 53
F 6	N 14	V 22	→ 30	6 38	+	46	
G 7	O 15	W 23	BLANKO 31	7 39	/	47	
H 8	P 16	X 24	Ø 32	B 40	ENTER	48	

Iza KEY se, u zagradi, nalazi broj koji se određuje iz tablice koja je data na slici. Vidi se da svakom tasteru odgovara neki broj, pa postaje jasno za šta KEY (n) služi: ova naredba omogućava ispitivanje da li je neki taster pritisnut ili nije. Posmatrajmo, na primer, program:

```
10 IF KEY (1) PRINT „PRITISNUO SI A“: ELSE GOTO 10
```

Program se neprekidno izvršava sve dok korisnik ne pritisne taster A, zatim se na ekranu pojavljuje odgovarajući komentar, posle čega računar prestaje sa radom.

Na demo kaseti je naredba KEY izuzetno mnogoeksploitalisana, naročito za ispitivanje pritisnutosti neke od strelica (u okviru naredbe INPUT čak se i ne može uneti strelica kao karakter!), pa će analiza programa sa nje pružiti neiskusnom korisniku predstavu o značaju ove naredbe.

KEY(0) ima funkciju koja je u mnogo čemu različita od funkcije do sada opisane naredbe KEY. KEY (0), naime, daje kod znaka koji je pridružen tasteru koji korisnik pritisne (kod se računa prema tabeli koja je data u diskusiji naredbe CHR\$). Naredba IF KEY(0) se, dakle, neće pravilno izvršavati, ali zato može da se napiše A=KEY(0). Kada računar, u toku izvršavanja programa, nađe na ovu naredbu, on će čekati sve dok korisnik ne pritisne neki taster, a zatim će vrednost kôda (broj 80 ako korisnik pritisne taster P, na primer) znaka koji je isписан na tom tasteru pridružiti promenljivoj A.

10 HOME

20 PRINT CHR\$(KEY(0))::GOTO 20, na primer, prenosi na ekran slova koja korisnik kuca pa se tako, na neki način, imitira „normalan“ rad računara. Međutim, program 10 X\$="\*\*"

20 X\$=X\$+CHR\$(KEY(0))

30 GOTO 20 na izvestan način imitira naredbu INPUT X\$, s tim što se uneseni tekst smešta u X\$ bez kvarenja izgleda ekranu.

## *21. Skraćivanje naredbi*

Ovim poglavljem započinjemo poslednji krug: to je deo uputstva za upotrebu koji je namenjen onima koji su solidno savladali osnove programiranja i upoznali sve do sada izložene karakteristike računara „galaksija“. Njim započinje deo koji će, silom prilika, biti manje jasan i zahtevati više napora za savladavanje. Za to postoje jaki razlozi: programiranje na mašinskom jeziku, koje želimo da upoznamo, predstavlja viši stupanj i neku vrstu nadgradnje bežikja i nije namenjeno neiskusnim korisnicima. Ovim, naravno, ne želimo nikoga da plašimo: mašinsko programiranje nije teško; ono zahteva samo malo više rada i nema smisla upuštati se u njega dok nismo savladali bežik.

Sa mašinskim programiranjem, ipak, ne možemo da počnemo baš odmah. Potrebno je da upoznamo neke tajne našeg računara i shvatimo kako je organizovana njegova memorija, kako možemo slobodno da menjamo njen sadržaj i kakvim se opasnostima pri tome izlažemo. Najpre ćemo pomenuti jednu karakteristiku koja može u mnogome da nam olakša manipulisanje sa računaram: mogućnosti skraćivanja naredbi.

Svaka naredba koju računar „galaksija“ poznaje i koja ima više od dva slova može da se skrati. Umesto, na primer, da pišemo PRINT 10\*2, možemo da otkucamo jednoetavno P. 10\*2. Tačka iza P govori računaru da je pred njim skraćena naredba PRINT. Ovako skraćena, naredba PRINT zauzima dva bajta memorije (PRINT zauzima pet), brže se izvršava i lakše kuca. Na žalost, skraćivanjem naredbi degradiramo izgled bežjik programa i činimo program znatno nerazumljivijim za početnika.

Bežjik kojim je opremljen računar „galaksija“ je tako koncipiran da svaka naredba može da se skrati što eliminiše potrebu da navedemo tabelu skraćenih oblika! Ovo, na prvi pogled, izgleda nemoguće: već smo, na primer, upoznali naredbe INPUT i INT, NEW i NEXT i slične koje počinje istim slovom. Verovali ili ne, računar „galaksija“ u ovoj prilici može ograničeno da čita vaše misli — skratite bilo koju naredbu na jedno slovo i on će pogoditi šta ste želeli da učinite! Objasnjenje je jednostavno: INPUT predstavlja naredbu koja počinje od početka neke linije bežjika ili od oznake : a INT funkciju koja se nalazi samo desno od znaka jednakosti gde naredba INPUT svakako ne može da se nađe. Isto tako RETURN nemamisla u komandnom, a RUN u programskom modu, pa svaka od njih može da se zameni sa R. O svemu ovome ne morate mnogo da mislite: jednostavno skratite svaku naredbu koja ima više od dva slova na jedno slovo i stavite tačku, a računar se nikada neće zbuniti.

## 22. Računanje sa logičkim iskazima

Obzirom da se u IF naredbi mogu naći samo logički izkazi u kojima se koriste oznake <,>, i=, svi oni koji su pisali iole složene programe susretali su se sa problemima tipa: potrebno mi je da računar ode na liniju broj 100 ako je X manje ili jednak Y. Ukoliko ste se dosetili trika koji ćemo da izložimo, zaslужujete sve komplimente i imate sve šanse da postanete dobar programer:

10 IF X>Y ELSE GOTO 100 (ili, s obzirom na prethodno poglavlje:  
10 IFX>YE.G.100)

Šta se desiš? Ako je X veće ili jednak Y, odgovor na pitanje „da li je X manje od Y“ je negativan, pa računar izvršava ono što se nalazi iza ELSE. Ukoliko je X manje od Y, ELSE biva ignorisano i računar prelazi na sledeću liniju.

Kako smo sli da izostavimo dve tačke ispred ELSE? To u ovom slučaju sme da se uradi obzirom da ne postoji nikakva dvoznačnost i da se tako štedi jedan bajt RAM-a. U radu sa računaram „galaksija“ ćete primetiti da u mnogim slučajevima neke naredbe mogu da se iskoriste u pojednostavljenom obliku koji nije izložen da bi se izbegla konfuzija kod početnika. U našem primeru dve tačke ispred ELSE ne bi nikako zasmetale računaru!

Ovaj trik je ipak nedovoljan: često želimo da se neka naredba izvrši ako je ieno nekoliko logičkih iskaza (ako je, na primer, X veće od Y i Z veće od 2). Računar „galaksija“ omogućava i korišćenje logičkih operacija I odnosno ILI, ali na donekle indirektan način.

Svakom logičkom iskazu koji je tačan dodeljuje se vrednost jedan, a onome koji je netačan — nula. Otkucajmo, na primer, PRINT 2=2 i na ekranu će se pojaviti broj jedan obzirom da je dva jednako dva. Možemo, isto tako, da napišemo PRINT (B=2)‡(A=3). Ukoliko je B različito od dva a A tri, na ekranu će se pojaviti broj 2 (1+1=2). Ukoliko je B jednak dva ali je A različito od tri ili ako je B različito od dva a A jednak tri, na ekranu će se pojaviti broj 1 (1+0=0+1=1). Ako je, najzad, A različito od tri, a B od dva, na ekranu će se pojaviti nula (0+0=0).

Naredba IF izračunava vrednost izraza koji je naveden iza nje i smatra da je on netačan ako je jednak nuli, a tačan ako je različit. U tom smislu je IF S=0 ELSE GOTO 100 potpuno ekvivalentno sa IF S GOTO 100, pri čemu je druga naredba kraća i brže se izvršava.

Koristeći upravo stečena znanja, logičko i čemo lako zamjenjivati množenjem, a logičko ILI — sabiranjem. Primer naveden tri pasusa ranije, na primer, može da se sažme u naredbu:

10 IF (X>Y)\*(Z>2) PRINT „USLOV JE ISPUNJEN”: ELSE PRINT „USLOV NIJE ISPUNJEN”

## 23. Dalje mogućnosti naredbe HOME

Naredba HOME, kao što smo rekli, nalaže računara da obriše sadržaj ekrana i o njoj se, izgleda, ne mora više razgovarati. No iza HOME može da se nađe i neki broj između 1 i 512 koji uneškoliko menja smisao ove naredbe.

Kada računar ispunji čitav ekran, prva linija nestaje, sve ostale se pomeraju za jedan red nagore, a novi tekst se pojavljuje u poslednjem (šesnaestom) redu ekrana. To je prirođen tok stvari, ali ponekad može da bude korisno da se jedan deo ekrana zaštitи od automatskog brisanja. Na tom delu ekrana može da se nađe neki tekst koji u toku čitavog rada programa treba da bude vidljiv (na primer, sažeto uputstvo za upotrebu kod neke igre). Da bismo, na primer, zaštitili prva tri reda ekrana, izvršićemo sledeći jednostavni program:

```
10 HOME
20 PRINT „OVI REDOVI”: PRINT „SU”: PRINT „ZAŠTIĆENI”
30 HOME 96
```

Zatim kućajmo neke naredbe i, kada ostatak ekrana bude ispunjen, primetimo da je „pokretan” samo deo ispod trećeg reda.

Zaštitu prvih redova nije potpuna: pomoću PRINT AT možemo da menjamo njihov sadržaj, pa čak i da ga obrišemo pritisnuvši SHIFT DEL. Međutim, zona na ekranu ostaje zaštićena čak i ako promenimo njen sadržaj — automatski scroll (pomeranje redova) je ne remeti.

Postoji jedno ograničenje vezano za zaštitu ekrana: kada je ona aktivirana, ubočajeni fini scroll (pomeranje redova nije naglo nego prijatno za oči) je isključen. To znači da će listanje programa biti oko tri puta brže nego obično, ali da će ga biti teže pratiti obzirom da redovi prebrzo „preleću” ekran. Ako želimo samo da isključimo fini scroll bez zaštite dela ekrana, otkućaćemo HOME 512. Ovo stanje, kao i zaštita bilo kog dela ekrana, opoziva se sa HOME.

## 24. Heksadecimalni sistem — oznaka &

Za rad sa mašinskim jezikom je pogodno koristiti heksadecimalni brojni sistem. U njemu se, uz cifre 0, 1, 2, 3, 4, 5, 6, 7, 8 i 9 koriste i „cifre” A, B, C, D, E i F. Tako, na primer, ubočajeni broj 10 se u heksadekadnom sistemu piše kao OA ili, jednostavno A, 11 je OB, 12 — OC, 13 — OD, 14 — CE, 15 — OF, a 16 se piše kao 10. Ako u broju nema slova A, B, C, D, E i F, lako može da dođe do zabune u kom je sistemu pisani. Problem se u programiranju (ponekad i u matematici) rešava tako što se ispred neksadekadnog broja stavlja oznaka & (na primer &10, &5A i slično). Ova oznaka je poznata i računaru „galaksija”; otkućajmo, na primer, PRINT &000A (ili samo PRINT &A) i pritisnimo RET — na ekranu će se pojaviti broj 10 koji predstavlja prevod heksadekadnog broja 000A u ubočajeni dekadni sistem.

Kada god u nekoj naredbi treba da napišemo neki dekadni broj, možemo da ga zamjenimo heksadekadnim pri čemu ispred njega stavljamo znak &. Tako, na primer, možemo da zamjenimo naredbu HOME 32 naredbom HOME &20, znaajući da se u svakom redu nalazi po &20 (&20=32) karaktera. Treba voditi računa o činjenici da se oznakom & mogu dobiti i negativni brojevi: &8001 nije 32769 nego — 32767.

Rad sa heksadekadnim sistemom postaje posebno značajan tek sa naredbama BYTE I WORD.

## 25.

### Organizacija memorije — naredba BYTE

Naš računar je opremljen ROM-om od četiri kilobajta i istim tolikim RAM-om (dodavanjem čipova, ROM može da se proširi do 8 Kb, a RAM do 6Kb; dodavanjem posebne pločice RAM bi mogao da se proširi do 54 kilobajta). ROM i RAM možemo da zamislimo kao skup ćelija koje su poredane u red i numerisane brojevima od &000 do &FFFF. U svaku ćeliju može da se smesti bilo koji broj između &00 i &FFFF ili, da budemo precizniji, u nekim od ćelija već se nalaze fiksirani brojevi. Ove ćelije se nalaze na početku memorije (adrese &000 do &0FFF) i čine ROM — deo memorije u koj je upisan sistemski program koji omogućava ispravno funkcionisanje našeg računara. U ostale ćelije možemo slobodno da upisujemo sadržaj koji će biti interpretiran kao mašinski program koji može da se izvršava.

Adrese &1000 do &1FFF odgovaraju ROM-u koji se ne nalazi u vašem računaru — to je proširenje koje će jednoga dana biti na raspolaganju ako za njega bude interesa. Adrese &2000 do &27FF obuhvataju ćelije koje, u sadašnjoj situaciji, za nas nisu od interesa: računar ih interno koristi za rad sa tastaturom. Informacije sadržane u ćelijama &2800 do &2C39 su veoma značajne: tu se nalaze sistemske promenljive, video-memorija (o sistemskim promenljivima i video memoriji čemo govoriti docnije), aritmetički akumulatori i stek. Počev od &2C3A, u memoriju se smešta bežik program (ukoliko sadržaj jedne sistemске promenljive nije promenjen, ali i o tome docnije) dok je kraj memorije rezervisan za nizove: od &37FF (ili &3FFF ako imate RAM od 6 kilobajta) se, unazad prema bežik programu, smešta alfanumerički niz X\$(I), a ispod njega numerički niz A(I). Ukoliko se bežik program i numerički niz „susretnu“ na sredini memorije, računar ispisuje SORRY i odbija da dodaje nove elemente numeričkom nizu.

Koristeći naredbu BYTE možemo da proverimo i, po potrebi, promenimo sadržaj bilo kog bajta (ćelije) memorije. Otkucajmo, na primer, PRINT BYTE (&2BA8) i pritisnimo uobičajeno RET — na ekranu će se pojaviti broj 11. To znači da ćelija čija je adresa &2BA8 sadrži broj 11 (dekadno). Otkucajmo, dalje, BYTE &2BA8, 12 i primetimo da se slika na našem monitoru pomerila udesno. PRINT BYTE (32BA8) će nas uveriti da je prethodna naredba naložila računaru da promeni sadržaj ćelije &2BA8 i u nju smesti broj 12 umesto dosadašnjeg 11. Samo pomeranje slike nije direktno izazvano naredbom BYTE — ćelija &2BA8 kazuje računaru gde počinje slika, pa se promene njenog sadržaja trenutno primećuju. Pouka je da je vrlo opasno menjati sadržaje memorije pomoću BYTE ukoliko tačno ne znamo šta radimo — greška može da izazove brisanje programa pa čak i blokiranje računara posle koga ni RESET neće pomoći — moraćemo da isključimo i ponovo uključimo kompjuter, što je, kao što znamo, lek za sve probleme. U sadašnjem slučaju dovoljno je da otkucamo BYTE &2BA8, 11 i sve će se vratiti u normalno stanje.

Evo još jednog primera upotrebe naredbe BYTE: želimo, na primer da osvetlimo sve tačke na ekranu. Ako to radimo pomoću DOT, posao će potrajati prilično dugo. Ako koristimo PRINT, nećemo nikako moći da osvetlimo poslednji karakter. Na svu sreću, program:

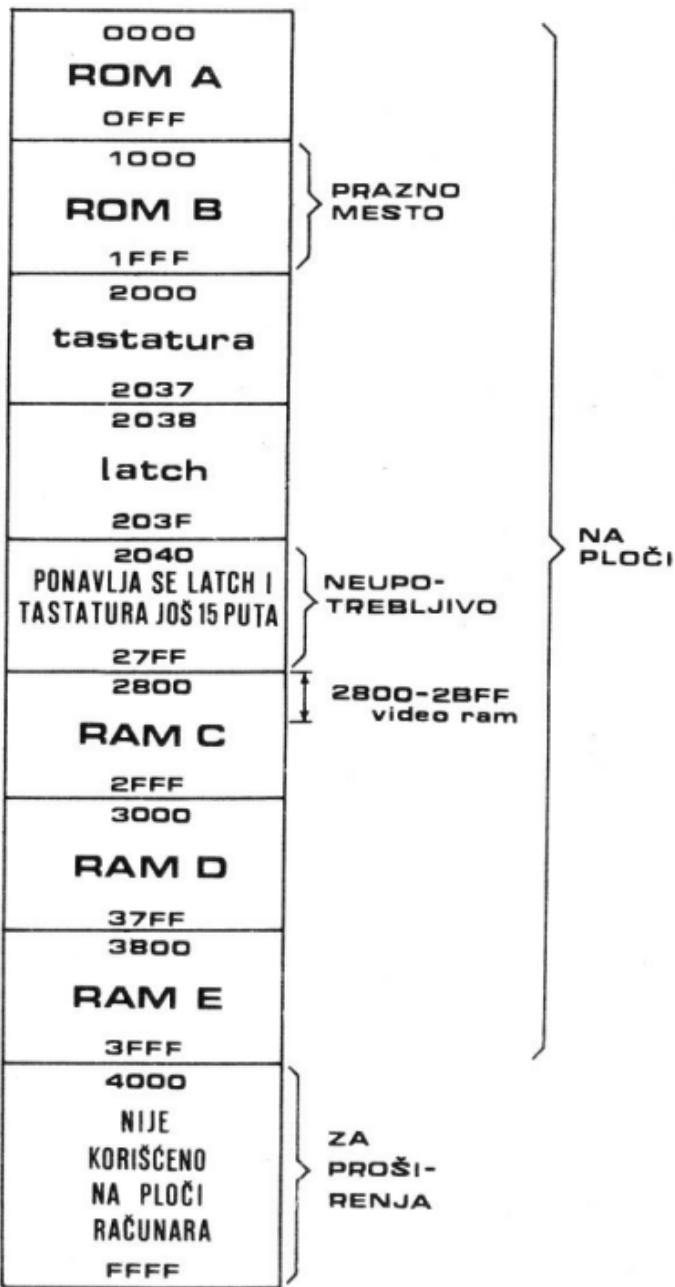
```
10 FOR I=&2800 TO &29FF
20 BYTE I, 255: NEXT I
```

Rešiće problem u prihvativljivom vremenu. Možemo da izmerimo vreme potrebno za ovu jednostavnu operaciju (ako ste dobro upoznati računar „galaksija“, povećajte preciznost merenja koristeći računarev časovnik) obzirom da ćemo isti problem docnije rešiti i na mašinskom jeziku.

## 26.

### Nove mogućnosti rada sa alfanumericima — naredba PTR

Izdvajanje slova iz neke alfanumeričke promenljive je do sada bilo skopčano sa velikim problemima (praktično nemoguće što je onima koji su pisali programe moralo da priredi



dosta neprijatnih trenutaka. Da bi se problemi rešili, treba upoznati naredbu PTR. Izvršimo, na primer, PRINT PTR X\$ i na ekranu će se pojaviti broj 10864. To znači da je promenljiva X\$ u memoriji smeštena počevši od ćelije 10864 (ili, što je isto, &2A70). Znajući da alfanumerička promenljivā prima 16 slova, zaključujemo da će ćelije &2A70 do &2A7F koriste za promenljivu X\$. Unesimo sledeći jednostavni program:

```
10 INPUT X$  
20 Y$=""  
30 FOR I=0 TO 2  
40 BYTE PTR Y$+I, BYTE (PTR X$+I)  
50 NEXT I: BYTE (PTR Y$+3), 0
```

Startujmo ovaj program i unesimo bilo koju reč koja ima manje od 16 slova. Trenutak docnije na ekranu će se pojaviti prva tri slova reči. Čitaoci ovih redova koji su razumeli rad sa alfanumericima će primeti da bi naredbe 40 i 50 mogle da se zameni sa:

```
40 Y$=Y$+CHR$ (BYTE (PTR X$+I)): NEXT I
```

Naredbu PTR možemo da koristimo i za pronaalaženje elemenata numeričkog ili alfanumeričkog niza ili fiksnih promenljivih. Kao vežbu pokušajte da sastavite program koji će sortirati reči po abecednom redu. Problem postaje bitno složeniji kada posmatramo reči koje sadrže i slova Š, Ž, Č i Č.

## 27.

### Sistemske promenljive — naredba WORD

Tabela na slici prikazuje mapu sistemskih promenljivih i drugih značajnih memorijskih adresa. Sistemske promenljive označavaju memorijske ćelije u koje računar smešta podatke od značaja za njegovo normalno funkcionisanje. Jednu od sistemskih promenljivih smo već upoznali — ćelija čija je adresa &2BA8 odgovorna za poziciju slike na ekranu.

#### SISTEMSKIE ADRESE

ADRESA	DATA	INICIJALNO	ŠABOZAJ
2800	512	20	VIDEO MEMORIJA
2A00	104	00	NUMERICKE VARIJABLE A-Z
2A68	2	2800	POZICIJA KURZORA U MEMORIJI
2A6A	2	3800	KRAJ MEMORIJE
2A6C	2	00	BLOKIRANJE DELA VIDEO-MEMORIJE
2A6E	2	00	STOK REGISTAR ZA FOR-NEXT
2A70	16	00	X\$
2A80	16	00	Y\$
2A91	2	00	STEP ZA AKTIVNU PETLJU
2A93	2	00	POZICIJA TEKUCE LINIJE (ZA VREME FOR-NEXT)
2A95	2	00	BASIC POINTER (ZA VREME CALL I FOR-NEXT)
2A99	2	00	16*ARR#+16
2A9B	2	00	ADRESA NEXT VARIJABLE
2A9D	2	2C3C	TAKE POINTER
2A9F	2	00	POZICIJA TEKUCE LINIJE
2AA1	2	00	REGISTAR ZA AKTIVNI FOR-NEXT
2AA3	2	00	SP PRIVREMEHO ZA AKTIVNI CALL
2AA5	2	00	DIFERENCIJATOR ZA TASTATURU
2AA7	3	00	RND
2AAC	124	00	ARITMETICKI AKUMULATORI (IX)
2BAB	1	11	HORIZONTALNA POZICIJA TEKSTA
2BA9	3	C9	LINK ZA NAREDBE
2BAC	3	C9	LINK ZA VIDEO
2BAF	1	00	AKO JE BIT 7=1, SAT RADJ
2BB0	1	00	BROJAC ZA POMERANJE SLIKE
2BB1	1	00	FLAG ZA POMERANJE SLIKE
2BB1	1	00	REGISTAR ZA REPT
2BB5	1	00	FLAG ZA STAMPAC
2BB6	125	00	BAFER
2C36	2	2C3A	POINTER POČETKA BASICA
2C38	2	2C3A	POINTER KRAJA BASICA
2C3A	??	00	BASIC PROGRAM

Međutim, većina drugih sistemskih promenljivih zauzima dva bajta obzirom da se u njima nalaze adrese nekih drugih delova memorije (sistemska promenljiva koja zauzima celije &2C36 i &2C37, na primer, daje adresu početka bežik programa). To znači da nam naredba BYTE pomaže samo posredno: možemo da ispitamo sadržaje obe celije koje čine neku sistemsku promenljivu i ih kombinujemo u jedan broj (u datom primeru PRINT 256\*BYTE (&2C37)+BYTE (&2C36) daje 11322 (isto što i &2C3A) — adresu početka bežik programa koja se slaže sa mapom). Ovakav način, posebno kada treba promeniti sadržaj neke sistemske promenljive, nije baš prijatan, pa je „galaksija“ opremljena, naredbom WORD koja je namenjena baš radu sa sistemskim promenljivima.

Naredba WORD je neobično slična naredbi BYTE osim u jednoj sitnici: ona operiše sa dva bajta memorije. Otkucajmo, na primer, PRINT WORD (&2C36) i na ekranu će se pojaviti broj 11322 — nikakva konverzija nije bila potrebna.

Otkucajmo, dalje, WORD &2C36, &2D3A: WORD &2C38, &2D3A i pritisnimo RET. Time smo promenili sadržaj sistemskih promenljivih u kojima računar čuva informacije o početku i kraju bežik programa, pa će svi bežik programi koji od tog momenta kucamo počinjati 256 bajta više (počevši od &2D3A). Tako dobijenih 256 bajta može, na primer, da se koristi za smeštanje mašinskih programa.

## 28. Mašinski programi — naredba USR

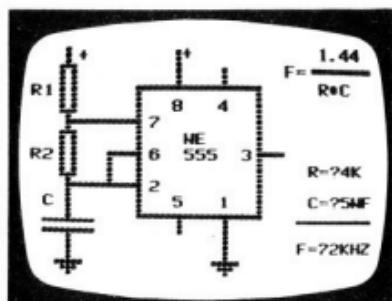
Raspoloživi prostor nam, na žalost, ne dopušta da se ozbiljno bavimo mašinskim programiranjem, ali ćemo izložiti neke podatke vezane za smeštanje, snimanje i startovanje mašinskih programa na računaru „galaksija“ i dati jedan primer. Za praćenje daljeg teksta je vrlo korisno posedovati spisak instrukcija procesora Z80A.

Postoje dva mesta koja su „prirodno određena“ za mašinske programe: jedno je već spomenuto (ispred bežika), a drugo na vrhu memorije, tamo gde se normalno nalazi niz X\$(l). Prva ideja je, bar po našem mišljenju, pogodnija, jer se mašinski programi snimaju zajedno sa bežikom (o tome docnije), pa ćemo se zadržati na njoj.

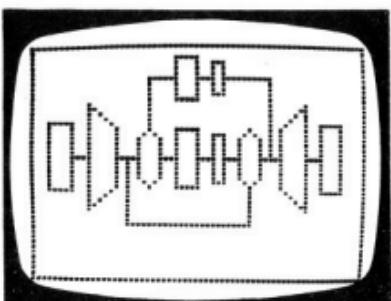
Da bi za mašinski program rezervisali nnnn bajtova, treba da pomerimo početak bežika. To možemo da uradimo pomoću naredbe WORD (kao u prethodnom poglavljiju), ali možemo da upotrebimo i novu opciju: NEW nnnn. Primenom ove naredbe briše se postojeći bežik program iz memorije i početak svih budućih bežik programa pomera za nnnn bajtova. Mašinski program se, u nedostaku programa koji bi to olakšao, unosi pomoću naredbe BYTE, bajt po bajt počevši od &2C3A. Da vidimo kako to izgleda na primeru.

Sastavljemo program koji popunjava ekran punim karakterima i videti koliko je mašinski jezik brz. Evo programa uz korišćenje standardne mnemonike procesora:

2C3A 01 00 02	LD BC, 200H	; treba „osvetliti“ &200 bajtova.
2C3D 21 00 28	LD HL,2800H	; adresa početka ekranu
2C40 36 FF CIKL	LD (HL),FFH	; FF je „pun karakter“
2C42 23	INC HL	; HL se uvećava
2C43 0B	DEC BC	; brojač je umanjen
2C44 CB 78	BIT 7,B	; da li je BC negativan?
2C46 28 F8	JR Z,CIKL	; ako nije, ponavlja se ciklus
2C48 C9	RET	; povratak u BASIC.



Pomoć konstruktoru: računar projektuje astabilni multivibrator



Sledeći potez je vaš: Popunite ovu blokšemu

Mašinski program se unosi tako što se najpre otkuca NEW 15 (program ima 15 bajtova), a zatim: BYTE&2C3A, &01 RET BYTE&2C3B, &00 RET BYTE&2C3C, &02 RET BYTE&2C3D, &21 RET itd. Poslednje se kuca BYTE&2C47, &F8 RET BYTE &2C48,&C9 RET čime je unošenje mašinskog programa završeno. Da bismo ga isprobali, otkucajmo: 10 A=USR (&2C3A)  
20 GOTO 20

USR je nova naredba. Ona nalaže računaru da izvrši mašinski program čiji se početak nalazi na adresi koja je navedena u zagradi. Po izvršavanju programa (mašinsko RET, C9) promenljiva A će dobiti vrednost iz HL registra, što u datom primeru nije mnogo bitno. Naredba 20, jednostavno, sprečava da se izvršavanje programa prekine i da izgled ekranu tako bude pokvaren.

Otkucajmo RUN, pritisnimo RET i, ako nije bilo greški u kucanju, ekran će biti popunjeno tačkama. Uporedimo ovo sa „brzinom“ (posle ovakve demonstracije navodnici su neophodni) BEZIK programi koji smo nedavno sastavili pa će nam postati jasno zašto se bolji programeri okreću mašinskom jeziku. Za one koji ne mogu da se sete kako da izmere vreme izvršavanja mašinskog programa da kažemo da je ono oko 0,02 sekunde što, u poređenju sa bežikom, predstavlja ubrzanje od oko 580 puta.

Želeli bismo da maksimalno stimulišemo pisanje programa na mašinskom jeziku. Računar „galaksija“ je dobar poligon za njih: dugme RESET će popraviti posledice najvećeg broja grešaka, tako da nećete morati stalno da unosite program kada računar „krahira“. I pored toga, biće neophodno da mašinske programe snimate na kasetu. U tome će vam pomoći sledeće poglavlje.

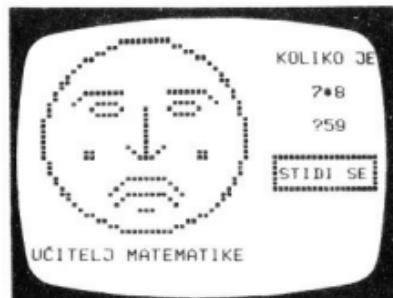
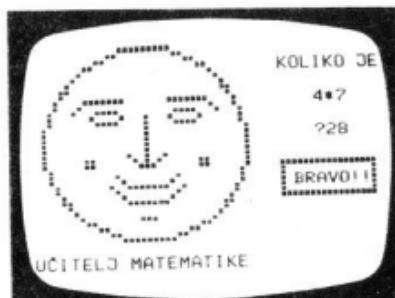
## 29.

### Nove mogućnosti naredbi SAVE i OLD

Ako mašinski program unesemo na način izložen u prethodnom poglavljiju, obično SAVE će, zajedno sa bežikom, snimiti i sve što se nalazi u memoriji, počevši od adrese &2C3A. Ponekad je, ipak, potrebno da na kasetu snimimo proizvoljan segment memorije. Za to će nam poslužiti naredba SAVE mmmm, nnnn.

Da bismo, na primer, snimili mašinski program koji se nalazi u čelijama memorije čije su adrese &3C30 do &3FFF, otkucaćemo SAVE &3C30, &3FFF, startovati snimanje na kasetofonu i pritisnuti RET. Posle toga, ukoliko nam je program važan, možemo da izvršimo verifikaciju na uobičajeni način.

Za učitavanje koristimo obično OLD: računar će učitane bajtove smestiti na iste adrese na kojima su se nalazili pre snimanja. Ponekad ćemo, međutim, poželeti da učitavanje programa započнемo određen broj bajtova pre ili posle mesta sa koga su snimljeni (ovo retko ima smisla, jer će sve JP i CALL mašinske naredbe neispravno funkcionisati posle pomeranja). Ako, na primer, želimo da mašinski program bude pomeran na više za 1024 bajta, otkucaćemo OLD 1024 (ili, što je isto, OLD &400), pritisnuti RET i startovati kasetofon. Da je pomeranje trebalo da bude negativno (prema početku memorije) otkučali bismo OLD — 1024. Obzirom da je ovakvo pomeranje relativno, svaki put kada snimamo neki mašinski program pomoću SAVE mmmm, nnnn, treba da zapišemo adresu njegovog početka i kraja.



# 30. Potprogrami iz ROM-a

Čak i dok niste započeli da pišete programe, u vašem računaru se nalazio jedan program i to toliko veliki i moćan da čete sličan moći da napišete tek kad postanete daleko iskusniji programer. To je sistemski program od četiri kilobajta koji omogućava računaru „galaksiju“ da radi ono što mu vi nalažećete; taj program omogućava prevođenje bežički naredbi koje unosite na oblik pristupačan mašini, izdavanje odgovora na ekranu, rad sa kasetofonom...

U jednom velikom programu, kao u metrou, ima mnogo stranica na kojima možete da se „ukrcate“, „otputujete negde“ i „izadete“. Poznavanje adresa tih „stanica“ je dovoljno da iskoristite naredbu **USR** i dodele do njih. Vreme koje nam je bilo na raspolaganju za pripremanje ovog uputstva kao i njegov obim nam, na žalost, nisu dopustili da se pozabavimo nabranjem interesantnih mašinskih potprograma. Odabrali smo samo jedan primer, koji će čitaoca podstaći na dalja istraživanja i, istovremeno, povećati mogućnosti računara „galaksija“.

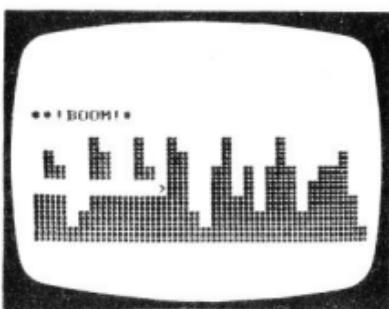
U toku rada (osim pri snimanju programa na kasetu) računar na ekranu neprekidno prikazuje sliku, na šta se, prirodno, gubi određeno vreme. Ponekad nam je potrebno da izvršimo neki komplikovan proračun koji odnosi dosta vremena, a slika nam nije bitna; zar ne bi bilo dobro „isključiti“ sliku i smanjiti vreme izvršavanja programa tri-četiri puta? Za ovakav ubrzani režim rada poslužiće nam naredba **A=USR(14)** (umesto A možemo da stavimo bilo koju promenljivu, napišemo **PRINT USR(14)** i slično). Povratak u normalan režim rada se postiže naredbom **A=USR(22)** ili automatski nastupa po završetku rada programa (taj završetak može da bude posledica naredbe **STOP**, pritiska na BRK ili **RESET** i slično). Evo i primera:

```
10 A=USR(14)
20 FOR I=1 TO 10000: NEXT I
```

Startujmo program sa **RUN**, a zatim sa **RUN 20** (time ubrzani režim nije iniciran) i primetimo razlike u vremenu izvršavanja. Za to ne možemo da koristimo računarev interni časovnik jer je on „isključen“ u toku ubrzanog režima rada.

```
5 PRINT
10 FOR A=32 TO 95
20 PRINT CHR$(A);
30 NEXT A
40 FOR B=128 TO 191
50 PRINT CHR$(B);
60 NEXT B
80 GOTO 80
END
>RUN
```

1. "B92%6 ( )\*+, -./0123456789;:;<>?
EABCDEFGHJKLMNOPQRSTUVWXYZWXY22C25
2. "B92%6 ( )\*+, -./0123456789;:;<>?
EABCDEFGHJKLMNOPQRSTUVWXYZWXY22C25
3. "B92%6 ( )\*+, -./0123456789;:;<>?
EABCDEFGHJKLMNOPQRSTUVWXYZWXY22C25



Kompjuterska abeceda: slova koja je računar „galaksija“ „naučio“.

Imate li dobre refleksije: Jedna od mnogobrojnih akcionalih igara prilagođena računaru „galaksija“.

```
10 INPUT X: CALL 3000:PRINT 9
20 GOTO 10

30001 I KVADRATNI KOREN
30010 IF X<0 Y=0: RETURN
30020 IF X<0 PRINT "KOREN NIJE
DEFINISAN": STOP
30030 Y=X^.5: Z=0
30040 W=(X-Y)*.5
30050 IF (W=0) + (Z=2) RETURN
30060 Y=Y+H: Z=H: GOTO 30040
END
>READY
??
1. 41421
255_
```

Za matematički nastojene: Potprogrami za kvadratni koren

```
10 INPUT X: CALL 31000
20 PRINT"LN(X)";L,"LOG(X)";X
30 GOTO 10
31000 I LOGARITAM
31010 E=0: IF X<0 PRINT "FUNKCI
JA NIJE DEFINISANA": STOP
31020 IF X>1 ELSE X=X/2: E=E+1:
GOTO 31020
31040 IF X<0.5 X=X+X: E=E-1: G0
TO 31060: ELSE X=(X-0.287107)/(X
+0.287107): L=X*X
31050 L=((0.598979*L+0.961421)*
L+2.88539)*X+E-0.5)*0.693147
31060 IF (L<1E-6)*(L>-1E-6)L=0
31070 X=L*0.4342945: RETURN
```

i logaritam

## 31. Saradujmo i dalje

Ovim smo stigli do kraja uputstva za upotrebu računara „galaksija“. Ostalo je, čini nam se, još dosta stvari koje nismo pomenuli, „caku“ koje ni mi ne pozajemo i, naravno, bezbroj ideja za programe. Ako ste sastavili računar „galaksija“ i naučili da se služite njime, pošaljite nam bar kratko pismo u kom ćete nas obavestiti o tim činjenicama. „Galaksija“ planira da otvori svoje stranice i posveti redovne rubrike temama vezanim za računar „galaksija“, kao i da u postojeći katalog bežik programa uvrsti što više dobrih programa za njega, koji će tako biti na raspolaganju svim korisnicima. Čitava ova akcija zavisi u mnogome i od svakog čitaoca ovih redova: vi ste ti koji treba da napišu programe i da ih pošalju na našu adresu. Samo udruženim naporima računar „galaksija“ može da bude dopunjeno onim što mu trenutno jedino nedostaje — programskom podrškom.

90 C.3000  
100 C.2100  
170 B.P.43:IFKEY(27)D=-32  
180 IFKEY(28)D=32  
190 IFKEY(30)D=1  
200 IFKEY(29)D=-1  
210 IFKEY(48)G.1000  
300 B.P.32:A#P+D  
310 G.300\*B.(A)  
332 P=A:G.170  
336 G.555  
342 H.:P.AT224,"UDARILI STE U Z  
VEZDU I STRADALI!";:G.1380  
396 G.555  
412 B.A.36:IFN=2G.170:E.N=N-1:I  
F1-(N=0)G.170  
414 U.\*:H.:P.AT202,"B R A V  
O"  
416 P.AT416,"VREME: ";Y#;  
418 G.1380  
555 D=-D16.170  
1000 Y=P  
1010 IFKEY(31)G.170  
1020 IFKEY(27)X=-32:G.1070  
1030 IFKEY(28)X=32:G.1070  
1040 IFKEY(30)X=1:G.1070  
1050 IFKEY(29)X=-1:G.1070  
1060 G.1010  
1070 IFB.(Y)=46B.Y.32  
1080 Y=0:X=G.B.(Y)+1100  
1132 B.Y.46:G.1070  
1136 IFM=2N=N-1:IFN=0G.414  
1138 B.Y.32:G.170  
1142 H.:P.AT224,"POGODILI STE Z  
VEZDU I STRADALI!";:G.1380  
1196 G.1138  
1212 H.:P.AT229,"POGODILI STE Z  
IVU METU";AT267,"I STRADALI!";:G.  
1380  
1355 H.:P.AT226,"POGODILI STE Z  
ID I STRADALI!";  
1380 P.AT500,"JOS JEDNOM?";  
1382 IFKEY(48).1390  
1386 IF1-KEY(14)G.1382  
1388 IFKEY(14)G.1388:E.H.:S.  
1390 IFKEY(4)G.1390:E.G.100  
1400 #25,112,12,42,8,96,35,36  
1500 IFKEY(14)G.1500E.R.  
2100 H.:P."NIVO IGRE (1 ILI 2)"  
;:I.M1 IF1-(M=1)-(M=2)G.2100:E.IF  
M=2C.6000  
2105 H.:T.B.F.I=10240T010271:B.  
1,255:B.I=480,255:N.I  
2110 F.I=10272T0107205.32:B.I,2  
55:B.I+31,255:N.I  
2120 F.J=1T04:T.K,C.F.L=1TOK  
2130 I=INT(480\*RND+10272):IF1-(  
B.(I)=32)G.2130  
2140 B.I,C:N.L:N.J  
2150 P=INT(480\*RND+10272):IF1-(  
B.(P)=32)G.2150  
2160 F.I=53T0495.-1:B.P,I:F.J=1  
T0120:N.J:N.I:D=0:N=25:IFM=2N=60  
2170 Y#="00:00:00":D.\*:R.  
3000 H.  
3010 P.AT166,"\*\*\*\*\*  
\*\*"  
3020 P.AT198,"\* W O N D E R E R  
\*\*"  
3030 P.AT230,"\*\*\*\*\*  
\*\*"  
3040 F.I=0T063:D.I,0:D.I,47:N.I  
3050 F.I=1T046:D.0,I:D.63,I:N.I  
3060 P.AT469,"UPUTSTVA?";  
3070 IFKEY(14)G.1500  
3080 IF1-KEY(4)G.3070  
3090 H.:P." U OVOJ AKCIONOJ I  
GRI TREBA"  
3100 P."DA PRETVORITE SVE ZIVE  
METE U"  
3110 P."MRTVE. ZIVA META POSTAJ  
E MRTVA"  
3120 P."SAMO AKO UDARITE U NJU.  
PORED"  
3130 P."ZIVIH I MRTVIH META NA  
EKRANU"  
3140 P."CETE VIDETI ZVEZDE I P  
REPREKE"  
3150 P."(NA EKRANU JE SAKRIVEN  
I NES."  
3160 P."TO NEVIDLJIVIH PREPREKA  
)MORA"  
3170 P."TE DA FAZITE DA NE UD  
ARITE U"  
3180 P."ZVEZDU DOK SE OD PREPRE  
KA ODBI"  
3190 P."JATE BEZ POSLEDICA."  
3200 P." OPREMLJENI STE ORUZZ  
EM KOJIM"  
3210 P."MOZETE DA UNISTAVATE PR  
EPREKE"  
3220 P."(VIDLJIVE I NEVIDLJIVE)  
ALI KO"  
3230 P."JE NE SMETE DA UPERITE  
U DRUGE"  
3240 P."OBJEKTE.";:C.4000  
3260 H.:P."OBJEKTI SU OBELEZENI  
OVAKOM:"  
3270 P.:P.:# JE PREPREKA I  
LI MRTVA META"  
3280 P.:P.0 JE ZIVA META"  
3290 P.:P.\* JE, NARAVNO, ZVE  
ZDA"  
3310 P.:P.+" JE VAS POLOZAJ":  
C.4000  
3350 H.:P."NA RASPOLAGANJU SU V  
AM KOMANDE!"  
3360 P.:P.:P."STRELICE ZA POME  
RANJE"  
3370 P.:P."RET ZA PUNJENJ  
E ORUZZA"  
3380 P." (ISPALJUJETE  
GA PRI"  
3390 P." TISKOM NA STR  
ELICU)"  
3400 P.:P."SPACE AKO NAPUNI  
TE ORUZZE"  
3410 P." PA SE PREDOMI  
SLITE."  
3420 P.:P.:" M N O G O S  
R E C E !"  
3430 C.4000:RET  
4000 P.AT497,"PRITISNI";  
4002 P.AT506,"RET";  
4004 IFKEY(48)G.4020  
4006 FORI=0T060:N.I  
4008 P.AT506,"";  
4010 IFKEY(48)G.4020  
4012 FORI=0T060:N.I:G.4002  
4020 IFKEY(48)G.4020  
4030 R.  
6000 H.:P." NA OVOM NIVOU MOR  
ATE DA UNI!"  
6010 P."STITE SVE STO SE NALAZI  
NA EX-"  
6020 P."RANU OSIM ZVEZDA I (EVE  
NTUALNO)"  
6030 P."NEVIDLJIVIH PREPREKA!"  
C.4000:RET

## SPISAK NAREDBI SA PRIMERIMA

ARR\$	ARR\$(20)
BYTE	PRINT BYTE(11176) BYTE 11176,12
CALL	CALL 100 CALL 10*A+100
CHR\$	PRINT CHR\$(34)
DOT	DOT 20,10 DOT * IF DOT 20,10 PRINT "BELO"
EDIT	EDIT 20
ELSE	IF B<17 X=1: ELSE GOTO 100
EQ	IF EQ Y\$,X\$(5) GOTO 100
FOR	FOR I=A TO 100
GOTO	GOTO 100 GOTO 10*A+100
HOME	HOME HOME 64 HOME 512
IF	IF A>5 GOTO 100
INPUT	INPUT A INPUT X\$
INT	A=INT(B/C)
KEY	IF KEY(1) GOTO 100 A=KEY(0)
LIST	LIST LIST 500
MEM	PRINT MEM
NEW	NEW NEW 256
NEXT	NEXT I
OLD	OLD OLD -50
PTR	PRINT PTR X\$
PRINT	PRINT A+2*B PRINT A;"*";B;"=";C PRINT A,B,C PRINT "GALAKSIJA"
RETURN	RETURN
RND	A=RND PRINT INT(7*RND+1)
RUN	RUN RUN 100
SAVE	SAVE SAVE 15728,16251
STEP	FOR I=100 TO 0 STEP -5
STOP	STOP
TAKE	TAKE A,100,B,C,250,X\$
UNDOT	UNDOT20,10 UNDOT *
USR	A=USR(14)
VAL	PRINT VAL(PTR X\$)
WORD	WORD 10905,0 PRINT WORD(10905)
!	! PROGRAM ZA SUMIRANJE
#	#10,30,"JANUAR"
&	PRINT &2CBA