PRIRUČNIK

za

TIM – 011 BASIC

TIM 011

IRO "GRADJEVINSKA KNJIGA"
BEOGRAD, 1988. godine

Tekst:
INSTITUT "MIHAJLO PUPIN", BEOGRAD
RJ RAČUNARSTVO

IRO "GRADJEVINSKA KNJIGA"
Beograd, Trg Marksa i Engelsa 8

MILAN VIŠNJIĆ, glavni urednik – direktor
MILICA DODIĆ, odgovorni urednik
OLGA VASILJEVIĆ, urednik

ISBN 86-395-0034-7

Typeset in TEX, ELEKTRONIKA 011, Beograd, Maršala Tolbuhina 53, tel. 437-437
Štampa: "BAKAR" – Bor

Sadržaj

1 Uvod

1.1 Napomene

U ovom uputstvu koriste se sledeće konvencije:

- [RET] (novi red) je taster kojim se završava unošenje komandnog ili programskog reda. Njegov kod je označen sa CR.
- Srednje zagrade ([ ]) označavaju da je izborno ono što se nalazi izmedju njih. tj. da se ne mora kucati.
- Izmedju znakova (< >), nalaze se informacije napisane malim slovima, koje korisnik treba da navede u BASIC komandi, instrukciji ili funkciji.
- Tri tačkice (...) služe da se označi ponavljanje.
- Sve što je napisano velikim slovima kuca se kako je napisano.
- Svi ostali znaci moraju se upotrebiti kako su prikazani.

Primer: **PRINT** [<broj>,]...

U ovom uputstvu se koriste srpskohrvatski izrazi: grupa znakova, znakovna konstanta, znakovna promenljiva, znakovni niz i znakovni izraz umesto odomaćenih engleskih:*string*, *string constant*, *string variable*, *string array* i *string expression*. Engleska reč *record* prevedena je našom rečju zapis. Engleske reči: *byte*, *buffer*, *port*, *microprocessor*, *sector* i *stack* nisu prevedene i pišu se kako se izgovaraju. Takodje se ne prevode ključne reči BASIC interpretatora.

1.2 Komanda pozivanja BASIC-a

GBASIC [<ime programa>] [**/F**:<broj datoteka>] [**/M**:<adresa>] [**/S**:<dužina zapisa>]
Unošenjem komande GBASIC bez opcija, BASIC interpretator se učitava u memoriju, a na ekranu se prikazuju zaglavlje verzije i poruka **Ok**. Ako se navede <ime programa> iza komande GBASIC, BASIC će po inicijalizaciji učitati i izvršiti navedeni program. Podrazumeva se da je nastavak imena programa .BAS, ako se ne navede drugačije. Ime programa može imati maksimalno 8 znakova, a nastavak 3 znaka. Tačka (.) služi za njihovo razdvajanje.

Opcijom **/F**:<broj datoteka> zadaje se broj datoteka koje mogu biti istovremeno otvorene. Ukoliko se ova opcija ne navede, podrazumeva se da se mogu otvoriti 3 datoteke istovremeno.

Opcija **/M**:<adresa> odredjuje memorijsku adresu do koje će BASIC smeštati svoje podatke. Iza te granice memorija je slobodna za korisničke namene. Ako se ova opcija ne navede, BASIC će zauzeti celokupni raspoloživi memorijski prostor.

Poslednjom opcijom **/S**:<dužina zapisa> postavlja se maksimalna dužina zapisa datoteka sa slučajnim pristupom. Dužina zapisa zadaje se u bajtovima. Ako se opcija ne navede, podrazumeva se 128 bajtova.

Svi brojevi mogu biti zadati u dekadnom, heksadekadnom (broju predhodi &H) i oktalnom (&O) brojnom sistemu.

Primeri pozivanja GBASIC interpretatora:

```
A>GBASIC
```

Učitava se BASIC interpretator.

```
A>GBASIC TEST/F:&O6
```

BASIC se učitava, izvršava program TEST.BAS a moguće je otvoriti maksimalno 6 datoteka istovremeno.

```
A>GBASIC /M:32768
```

Učitava se BASIC i koristi se prvih 32 kilobajta memorije.

```
A>GBASIC M0:TEST.ZZZ/F:2/M:&H9300
```

Učitava se BASIC, izvršava program TEST.ZZZ iz memorijskog diska, dozvoljeno je otvoriti 2 datoteke istovremeno i memorija je ograničena na oko 37Kb.

2 Opšte karakteristike

2.1 Komandna linija

Kada BASIC interpretator ispiše na ekranu poruku `Ok` spreman je da prima komande ili program. Linija u kojoj se kuca zove se komandna linija. Ona je dugačka maksimalno 255 slovnih znakova. Pošto je ekranski red kraći, linija duža od 80 znakova se ispisuje u više redova.

U komandnoj liniji može se otkucati komanda, niz komandi i programski red. Komande su engleske reči iza kojih može da sledi odredjeni parametar (na primer: `RUN`, `LIST`, `LIST 100-` ...). Komande se izvršavaju odmah i po izvršavanju zaboravljaju. Ovaj način rada zove se direktni i u njemu je pogodno obavljati odredjene manje operacije (prikaz vrednosti promenljivih, manja izračunavanja itd.).

2.2 Program / Programski red

Program u BASICu je sastavljen iz niza programskih redova. Programski red ima sledeći oblik:

<nnnnn> <BASIC instrukcija> [:<BASIC instrukcija>]...

<nnnnn> je broj programskog reda i kreće se u rasponu od 0 do 65529. BASIC instrukcija je postojeća ključna reč koja vrši odredjenu funkciju i može imati iza sebe odredjene parametre. Programski red se može sastojati iz više instrukcija. Tada instrukcije odvajamo jedne od drugih znakom dve tačke (:). Programski red se, dakle, razlikuje od komandnog po tome što se ispred komande ili instrukcije nalazi broj reda. Program (skup programskih redova) se može kucati preko reda (15. red, 2. red, 16. red...), a BASIC će ih sam urediti tako da njihovi brojevi budu u rastućem redosledu. Programski redovi se izvršavaju rastućim redosledom njihovih brojeva. Ako otkucamo programski red sa brojem koji je prethodno već upotrebljen, novootkucani će zameniti postojeći programski red. Ova mogućnost se koristi i za brisanje. Otkuca se broj reda i [RET], što znači: zameni stari red praznim. Programski redovi se čuvaju u memoriji dok se ne izda komanda za njihovo brisanje ili isključi računar. Program se može sačuvati na disku pre isključivanja ili brisanja (vidi komandu SAVE). U memoriji se čuva i izvršava samo jedan BASIC program.

2.3 Kontrolni znaci

BASIC prihvata sve znake abecede (A-Z,a-z), specijalne znake (znake iznad brojeva i na desnoj strani tastature) i odredjene kontrolne znake. Kontrolni znaci se dobijaju pritiskom na odredjene tastere ili kombinacijom [CTRL] (kontrol tastera) i nekog slova. BASIC prepoznaje sledeće kontrolne znake :

ZNAK		TASTER	OBJAŠNJENJE / FUNKCIJA
[CTRL]	A		Menjanje teksta komandne linije
[CTRL]	C	[BRK]	Prekida rad programa i vraća se u komandnu liniju
[CTRL]	H	[BS]	Pomera kursor jedno mesto ulevo i briše znak pod njim
[CTRL]	I	[TAB]	Tabulator (pomera kursor za 8 znakovnih mesta udesno)
[CTRL]	J	[LF]	Pomera kursor u sledeću fizičku liniju ekrana
[CTRL]	M	[RET]	Kraj komandi, kraj programskog reda, novi red. [RET] se u ASCII kodu označava sa CR
[CTRL]	O		Isključuje štampanje dok program radi (naredno pritiskanje uključuje štampanje.)
[CTRL]	R		Ponovo ispisuje liniju koja se kuca
[CTRL]	S		Zaustavlja program
[CTRL]	Q		Nastavlja izvršavanje programa zaustavljenog sa [CTRL] S
[CTRL]	U		Briše liniju koja se kuca
[CTRL]	[	[ESC]	Izlaz iz izmene teksta programske linije
del		[DEL]	Briše poslednji otkucani znak i stavlja ga izmedju kosih crta

3 Sastav BASIC-a

BASIC interpretator prepoznaje sledeće objekte:

- konstante
- promenljive
- komande
- instrukcije
- komandnu liniju
- program

Komandna linija je veza izmedju čoveka i BASICa i sastoji se iz komandi i njihovih parametara. Parametri su konstante ili promenljive. Program se sastoji iz programskih redova, a ovi dalje od instrukcija, konstanti i promenljivih.

3.1 Konstante

Konstante su nepromenljive vrednosti koje se nalaze u parametrima komandi i instrukcija. Postoje sledeće konstante:

- numeričke
- znakovne

Numeričke konstante se dele na:

- Celobrojne konstante (*integer*)
- Konstante sa fiksnim decimalnim zarezom (*fixed point*)
- Konstante sa pokretnim decimalnim zarezom (*floating point*) koje mogu biti: jednostruke preciznosti (*single precision*) i dvostruke preciznosti (*double precision*)
- Heksadekadne konstante (*hex*)
- Oktalne konstante (*octal*)

Napomena: U anglosaksonskim zemljama koristi se decimalna tačka *(decimal point)* umesto zareza koji je uobičajen kod nas. U BASIC jeziku se, dakle, koristi decimalna tačka umesto zareza.

Celobrojne numeričke konstante su celi brojevi bez decimalne tačke u opsegu od -32768 do 32767.

Pozitivne konstante sa decimalnom tačkom su konstante sa fiksnom decimalnom tačkom.

Konstante sa pokretnom decimalnom tačkom se zapisuju u notaciji mantisa i eksponent, pri čemu vrednosti eksponenta predhodi E ili D u zavisnosti od toga da li je konstanta jednostruke ili dvostruke preciznosti, respektivno.

Konstanta je jednostruke preciznosti ako:

ima sedam ili manje cifara ili

ima eksponent zapisan sa E ili

iza cifara stoji znak uzvika !.

primeri: `46.9 -1.345E-06 64535.  2341.5!`

Konstanta je dvostruke preciznosti ako:

ima osam ili više cifara ili je

eksponent zapisan sa D ili

iza cifara stoji znak #.

primeri: `4582374589 -1.34533D-06 64535.0# 374618273647346.2312`

Konstante kojima prethode znaci &H su u heksadekadnoj notaciji (na primer `&H1A9F`), a u oktalnoj su notaciji ukoliko im prethode znaci `&` (na primer `&123`) ili `&O` (na primer `&03732`).

3.2 Znakovne konstante

Znakovne konstante su nizovi znakova ograničeni znacima navoda (") i mogu biti dugačke do 255 znakova.

primer:

```
"Zdravo" "broj 1233" "18974,3434"
```

3.3 Promenljive

Za razliku od konstanti čija je vrednost nepromenljiva, vrednost promenljive se može menjati dodeljivanjem nove vrednosti. Promenljiva mora imati ime različito od ključnih reči (komandi, instrukcija i funkcija). Imena promenljivih mogu biti bilo koje dužine, pri čemu BASIC radi sa prvih 40 znakova imena. Promenljive koje se razlikuju od 40 -og znaka imena nadalje smatraju se istim. Ime se može sastojati iz slova, brojeva i tačke, a mora počinjati slovom.

Promenljive mogu biti:

- numeričke
- znakovne
- numerički nizovi
- znakovni nizovi

Numeričke promenljive se dele na:

- Celobrojne (integer)
- Promenljive sa pokretnim decimalnim zarezom (floating point) koje mogu biti:
  - jednostruke preciznosti (single precision) i
  - dvostruke preciznosti (double precision)

3.4 Tipovi promenljivih

Promenljiva uz čije ime stoji znak % je celobrojna promenljiva. Znak ! stoji uz ime promenljive jednostruke preciznosti, a znak # uz ime promenljive dvostruke preciznosti. Znakovne promenljive imaju uz ime znak $. Promenljive bez ovih znakova iza imena smatraju se (ukoliko drugačije nije definisano) promenljivama jednostruke preciznosti. Tip promenljive može biti unapred definisan instrukcijama: DEFINT, DEFSNG, DEFSTR i DEFDBL (videti u poglavlju BASIC instrukcije).

primer:

`PI#`	dvostruka preciznost
`MINIMUM!`	jednostruka preciznost
`MAKSIMUM`	jednostruka preciznost
`LIMIT%`	celobrojna promenljiva
`IME$`	znakovna promenljiva

3.5 Nizovi

Nizovi su skup promenljivih za koje je potrebno rezervisati memorijski prostor (vidi DIM instrukciju) čija veličina zavisi od broja elemenata niza i tipa promenljivih. Imena nizova su istog oblika kao imena promenljivih. Ime elementa niza sastoji se iz imena niza i dimenzije zatvorene u male zagrade (na primer A(10)). Nizovi mogu biti višedimenzionalni pri čemu broj dimenzija može ići do 255, a broj članova po dimenziji do 32767. Na primer, `Z1313(12,44,1,5,1)` je element sa koordinatama 12 44 1 5 i 1 petodimenzionalnog niza `Z1313` jednostruke preciznosti.

3.6 Memorijski prostor

Promenljive u zavisnosti od tipa zauzimaju sledeći memorijski prostor:

TIP PROMENLJIVE	BROJ BAJTOVA
celobrojna	2
jednostruke preciznosti	4
dvostruke preciznosti	8
celobrojni niz	2 po elementu
niz jednostruke preciznosti	4 po elementu
niz dvostruke preciznosti	8 po elementu
znakovna	broj znakova + 3

3.7 Automatsko pretvaranje tipova promenljivih

BASIC će u odredjenim slučajevima automatski pretvoriti jedan tip vrednosti u drugi. To se dogadja u sledećim slučajevima:

- Ako se numerička konstanta dodeljuje numeričkoj promenljivoj koja je drugog tipa, onda će se izvršiti pretvaranje konstante u tip promenljive. Ovo ne važi za znakovne i numeričke tipove. Ako je potrebno da se numeričkoj promenljivoj dodeli znakovna vrednost ili obrnuto. BASIC će javiti `Type mismatch`. odnosno (zamena tipa).

  primer:

```
10 A% = 23.42   dodeli promenljivoj A% 23.42
20 PRINT A%     štampaj A%
RUN             izvrši program
 23
```

- Prilikom rešavanja aritmetičkog izraza svi operandi u aritmetičkim i logičkim operacijama se postavljaju na isti nivo preciznosti tj. na preciznost najpreciznijeg tipa. Rezultat izraza takodje ima isti nivo preciznosti.

primer:

```
10 D#=6#/7
20 PRINT D#
RUN
 0.8571428571428571
```

- Logički operatori (vidi: logički operatori) pretvaraju svoje operande u celobrojni tip i vraćaju takodje celobrojni tip kao rezultat. Operandi moraju biti u opsegu -32768 do 32767 inače BASIC javlja `Overflow` (greška prekoračenja).

- Kada se vrednost u pokretnom zarezu dodeljuje celobrojnom tipu, onda se razlomljeni deo zaokružuje na ceo broj.

primer:

```
10 C%=5.567E01
20 PRINT C%
RUN
 56
```

- Ako se vrednost jednostruke preciznosti dodeljuje promenljivoj dvostruke preciznosti, samo prvih sedam cifara rezultata biće tačno. Ovo se dešava zato što je tačnost jednostruke preciznosti ograničena na sedam cifara. Apsolutna greška prilikom ove konverzije je manja od 6.3E-8.

primer:

```
10 A=2.04
20 B#=A
30 PRINT A,B#
RUN
 2.04         2.039999961853027
```

3.8 Izrazi i operatori

Izraz može biti konstanta, promenljiva ili konstrukcija konstanti i promenljivih sa operatorima izmedju njih, tako da za rešenje ima jednu vrednost. Operatori su matematičke i logičke operacije nad izrazima. Operatori u BASICu dele se na četiri grupe:

- Aritmetički
- Relacijski
- Logički
- Funkcijski

3.9 Aritmetički operatori

Aritmetički operatori, poredjani po prioritetu, su:

OPERATOR	OPERACIJA	PRIMER
^	stepenovanje	x^2
-	negacija	-x
* /	množenje, deljenje	x*y x/y
+ -	sabiranje, oduzimanje	x+y x-y

Za menjanje prioriteta koriste se male zagrade. Prvo se izračunava ono što je u njima po uobičajenom prioritetu.

primer:

ARITMETIČKI IZRAZ	BASIC ARITMETIČKI IZRAZ
$x + 2y$	x+2*y
$x - \frac{y}{z}$	x-y/z
$\frac{xy}{z}$	x*y/z
$\frac{x+y}{z}$	(x+y)/z
$(x^2)^n$	(x^2)^n
$x(-y)$	x*(-y)

3.10 Celobrojni operatori

U BASICu postoji još par aritmetičkih operatora: celobrojno deljenje (*integer division*) i ostatak pri deljenju (modulus). Celobrojno deljenje je, na primer, 10\4 = 2. Znak za celobrojno deljenje je obrnuta kosa crta (\). Vrednosti koje učestvuju u celobrojnom deljenju mogu biti brojevi čija se zaokružena vrednost nalazi u opsegu [-32768,32767], dakle u celobrojnom deljenju mogu učestvovati i razlomljeni brojevi, na primer, 25.68\6.99 = 3. Po prioritetu je celobrojno deljenje odmah iza množenja i običnog deljenja.

Ostatak pri deljenju se označava operatorom MOD i daje ostatak pri celobrojnom deljenju. Na primer:

10.4 MOD 4 = 2 (10 \ 4 = 2 i ostatak 2)
+ 25.68 MOD 6.99 = 5 (26 \ 7 = 3 i ostatak 5)

Prioritet MOD operatora je odmah iza celobrojnog deljenja. Najčešća upotreba ova dva operatora je razdvajanje reči u bajtove: xxxxx \ 256 = viši bajt i xxxxx MOD 256 = niži bajt u reči dužine dva bajta.

3.11 Prekoračenje i deljenje nulom

Ukoliko se pri izračunavanju aritmetičkog izraza pojavi u imeniocu nula (deli se nulom), ispisuje se poruka `Division by zero` (deljenje nulom). Rezultat deljenja je mašinska beskonačnost (maksimalna vrednost broja u BASICu) sa korektnim predznakom. Izvršavanje programa se nastavlja. Dizanje nule na negativni stepen izaziva istu grešku.

Ako se desi da dodje do prekoračenja, ispisuje se poruka `Overflow` (prekoračenje), a rezultat je mašinska beskonačnost sa odgovarajućim predznakom (u zavisnosti od operacije i operanada). Izvršavanje programa se nastavlja.

3.12 Relacijski operatori

Pored aritmetičkih operatora u BASIC jeziku postoje i relacijski operatori. Upotrebljavaju se za poredjenje dve vrednosti. Rezultat poredjenja može biti tačno (true, vrednost različita od 0) i netačno (false, vrednost 0). Ovaj rezultat može biti upotrebljen za kontrolu toka izvršavanja programa (vidi instrukciju IF..THEN..ELSE) ili u aritmetičkim izrazima (na primer A=B=C što će dodeliti promenljivoj A vrednost 0 ako B i C nisu isti, ili -1 ako su isti). Relacijski operatori su:

OPERATOR	RELACIJA	IZRAZ
=	Jednakost	`X = Y`
<>	Nejednakost	`X <> Y`
<	Manje od	`X < Y`
>	Veće od	`X > Y`
<=	Manje ili jednako	`X <= Y`
>=	Veće ili jednako	`X >= Y`

Relacijski operatori imaju niži prioritet od aritmetičkih, pa će se izvršavati poslednji. Na primer, izraz:

```
X+Y < (T-1) / Z
```

je istinit ukoliko je vrednost `X+Y` manja od `T-1` podeljeno sa `Z`.

3.13 Logički operatori

Logički operatori omogućavaju testiranje višestrukih relacija, manipulisanje bitovima ili formiranje logičkih izraza. Logički izrazi vraćaju rezultat koji se smatra tačnim (različit od nule) ili netačnim (nula). Po prioritetu izvršavanja logički izrazi su ispod aritmetičkih i relacijskih.

Logički operatori poredjani po prioritetu:

OPERATOR	OPERAND(I)	REZULTAT
NOT	X	NOT X
	0	1
	1	0
AND	X Y	X AND Y
	0 0	0
	0 1	0
	1 0	0
	1 1	1
OR	X Y	X OR Y
	0 0	0
	0 1	1
	1 0	1
	1 1	1
XOR	X Y	X XOR Y
	0 0	0
	0 1	1
	1 0	1
	1 1	0
IMP	X Y	X IMP Y
	0 0	1
	0 1	1
	1 0	0
	1 1	1
EQV	X Y	X EQV Y
	0 0	1
	0 1	0
	1 0	0
	1 1	1

Logički operatori mogu povezati dva ili više relacijskih izraza.

Na primer:

```
IF A>31 AND A<127 THEN PRINT CHR$(A)
IF A<32 OR A>126 THEN PRINT "." :IF NOT A THEN 90
```

Logički operatori mogu biti upotrebljeni za testiranje i postavljanje pojedinačnih bitova u mašinskim rečima.

AND operator se obično koristi za izdvajanja odredjene grupe bitova radi testiranja ili postavljanje bitova na nulu.

OR operator obično služi za postavljanje bitova na jedinicu ili za sastavljanje grupa bitova u celinu.

Pogledajte sledeće primere:

63 AND 16 ⇒ 16	63	binarno	111111
	16	binarno	10000
	16	rezultat	10000
11 AND 14 ⇒ 10	11	binarno	1011
	14	binarno	1110
	10	rezultat	1010
4 OR 2 ⇒ 6	4	binarno	100
	2	binarno	10
	6	rezultat	110
10 OR 10 ⇒ 10	10	binarno	1010
	10	binarno	1010
	10	rezutat	1010
−1 OR −2 ⇒ −1	-1	binarno	1111111111111111
	-2	binarno	1111111111111110
	-1	rezultat	1111111111111111

Napomena: svi logički operatori rade sa 16 bitova, a negativne vrednosti se predstavljaju u tzv. drugom komplementu. Drugi komplement se dobija na sledeći način: broj se predstavi u binarnom obliku, zatim se bitovi komplementiraju (jedinica se zameni nulom, a nula jedinicom) i na kraju, dobijenom rezultatu se doda 1.

```
NOT X ⇔ -(X+1)
```

3.14 Funkcije

Funkcije vraćaju jednu vrednost izrazu u kome se nalaze. Vrednost zavisi od argumenata funkcije. Na primer, SQR(X) vraća kvadratni koren vrednosti X ili recimo. SIN(X) izračunava sinX. Detaljan opis funkcija videti u poglavlju BASIC funkcije.

Pored funkcija ugradjenih u BASIC interpreter postoji način da se definišu nove funkcije i upotrebljavaju istovetno kao ugradjene (vidi instrukciju DEFFN).

3.15 Operacije nad znakovnim konstantama i promenljivama

Grupe znakova se mogu spajati operatorom plus (+), na primer:

```
10 A$="IME ": B$="DATOTEKE"
20 PRINT A$ + B$
30 C$="NOVO "+A$+B$
40 PRINT C$
RUN
IME DATOTEKE
NOVO IME DATOTEKE
```

Kao i numeričke vrednosti, grupe znakova se mogu porediti relacijskim operatorima . Grupe znakova se porede po dužini i vrednostima. Vidi vrednosti znakova u dodatku ASCII KOD. U poredjenju učestvuju vodeći i prateći blanko znakovi.

Za poredjenje se koriste relacijski operatori: =, <>, <, >, <=, >=

Primeri poredjenja:

```
"AA" < "AB"
"IME PROGRAMA" = "IME PROGRAMA"
"X&" > "X#"
"CL " > "CL"
"KG" < "kg"
D$ < "9/12/87"  gde je D$="8/12/87"
```

Sve znakovne konstante moraju biti zatvorene u navodnike.

3.16 BASIC greške

BASIC ispisuje poruke o greškama u tekstualnom obliku na engleskom jeziku. Poruke o greškama su lako razumljive. Više o tome videti u dodatku PORUKE O GREŠKAMA.

4 BASIC komande i instrukcije

Sve BASIC komande i instrukcije su opisane u ovom poglavlju na sledeći način:

- Format (prikazuje kako se zadaje instrukcija)
- Namena (objašnjava se namena instrukcije)
- Objašnjenje / Primedbe (govori se o detaljima vezanim za instrukciju)
- Primer (navodi jednostavne primere upotrebe instrukcije)

Za opis formata važe NAPOMENE (poglavlje 1.1).

4.1 AUTO

Format: AUTO [<broj linije>[,<uvećanje>]]
Namena: Automatsko zadavanje broja programske linije pri unošenju programa.

Objašnjenje / Primedbe

Komanda AUTO zadaje brojeve programskih linija od <broja linije> i uvećava broj sledeće linije za <uvećanje>. Kada se parametri ne navedu, dodeljuje im se vrednost 10. Ako je naveden <broj linije> i otkucan zarez, a <uvećanje> izostavljeno, uvećanje zadržava predhodno zadatu vrednost.

Ukoliko AUTO generiše broj linije koja već postoji, iza broja linije se pojavljuje zvezdica kao upozorenje. Ako je želite sačuvati, pritisnite taster [RET] i linija neće biti promenjena, a AUTO će generisati sledeći broj.

Izvršavanje komande AUTO prekida se istovremenim pritiskom na [CTRL] i C. Poslednja linija tada neće biti zapamćena.

primer:

```
AUTO 100,50   generisaće brojeve linija 100, 150, 200....
AUTO          generisaće brojeve linija 10, 20, 30....
```

4.2 CALL

Format: CALL <ime potprograma> [(<lista promenljivih>)]
Namena: Pozivanje mašinskog potprograma.

Objašnjenje / Primedbe

CALL instrukcija je jedan od načina da program u BASICu pozove mašinski potprogram (vidi funkciju USR u poglavlju BASIC FUNKCIJE).

Pre poziva mašinskog potprograma promenljivoj <ime potprograma> mora se dodeliti memorijska lokacija (adresa) početka mašinskog potprograma. <ime potprograma> ne može biti član niza.

<lista promenljivih> sadrži promenljive koje se prosledjuju mašinskom potprogramu. <lista promenljivih> sadrži isključivo numeričke promenljive.

primer:

```
100 ASS = &HD000
110 CALL ASS(A,B,C)
```

4.3 CHAIN

Format: CHAIN [MERGE]<ime programa>
[,[<izraz za br.lin.>][,ALL][,DELETE <opseg>]]

Namena: Pozivanje (mešanje) više programa sa ili bez čuvanja promenljivih

Objašnjenje / Primedbe

CHAIN bez MERGE briše programsku memoriju i učitava novi program. <ime programa> je znakovna konstanta ili promenljiva. Sadrži ime programa koji se poziva.

<Izraz za br. lin.> je aritmetički izraz čija je vrednost broj linije od koje će početi izvršavanje pozvanog programa. Ukoliko se izostavi, program se izvršava od prve linije u programskoj memoriji.

Opcija ALL čuva vrednosti promenljivih. Ukoliko je izostavljena, može se zadati instrukcija COMMON za čuvanje odredjenih promenljivih (vidi COMMON instrukciju).

Opcija MERGE omogućava mešanje programa. Linije iz pozvanog programa zameniće linije postojećeg programa koje imaju iste brojeve. Pozvani program mora biti ASCII datoteka (vidi komandu SAVE).

Opcija DELETE prilikom povezivanja / mešanja programa briše u postojećem programu linije sa brojevima u zadatom <opseg>-u.

<Izraz za br. lin.> i<opseg> se ne menjaju komandom za renumeraciju programa (vidi komandu RENUM).

primer:

```
CHAIN "P1"
```

Poziva program P1.BAS i izvršava ga od prve linije.

```
CHAIN "P1",100
```

Poziva program P1.BAS i izvršava ga od 100-te linije.

```
CHAIN "P1",100,ALL
```

Poziva program P1.BAS, čuva vrednosti svih promenljivih i izvršava ga od 100-te linije.

```
CHAIN MERGE "P1",100
```

Meša postojeći i pozvani program. Izvršavanje počinje od 100-te linije izmešanog programa.

```
CHAIN MERGE "P1",100,DELETE 1000-5000
```

Briše linije od 1000 do 5000 u postojećem programu, meša postojeći i pozvani program, a izvršavanje počinje od 100 -te linije izmešanog programa.

Napomena: CHAIN instrukcija sa MERGE opcijom ostavlja datoteke otvorene. Ukoliko je MERGE izostavljen, CHAIN ne čuva predefinisane tipove promenljivih i korisnički definisane funkcije. Zato se u pozvanom programu sve potrebne definicije tipa DEFINT, DEFSNG, DEFDBL, DEFSTR i DEFFN moraju ponovo zadati.

4.4 CLEAR

Format: CLEAR [,[<izraz1>][,<izraz2>]]

Namena: Postavlja sve numeričke promenljive na nulu, sve znakovne promenljive na prazno, zatvara sve otvorene datoteke i opciono postavlja kraj raspoložive memorije i dubinu BASIC steka.

Objašnjenje / Primedbe

Vrednost <izraz1> je memorijska adresa. Do ove adrese BASIC memoriju smatra svojim radnim područjem i u njoj pamti program i promenljive.

Vrednost <izraz2> je dubina BASIC steka. Ako se <izraz2> izostavi, podrazumeva se 256 bajtova.

primer:

```
CLEAR
CLEAR ,32768
CLEAR ,,2000
CLEAR ,32768,2000
```

4.5 CLOSE

Format: CLOSE [[#]<broj datoteke>[,[#]<broj datoteke>]...]

Namena: Zatvara zadate ili sve otvorene datoteke.

Objašnjenje / Primedbe

<broj datoteke> je broj pod kojim je datoteka otvorena. CLOSE bez broja zatvara sve otvorene datoteke. Veza izmedju imena datoteke i njenog broja završava se ovom naredbom. Moguće je posle CLOSE uzeti drugu datoteku pod već korišćenim brojem ili pod drugim brojem otvoriti ranije korišćenu datoteku. Za sekvencijalne datoteke CLOSE upisuje poslednji sadržaj bafera u datoteku na disku. END i NEW komande automatski zatvaraju sve otvorene datoteke, za razliku od STOP (vidi OPEN).

primer:

```
CLOSE
CLOSE#1   zatvara samo datoteku broj 1
```

4.6 CLS

Format: `CLS`
Namena: Brisanje ekrana (grafičkog i tekstualnog).

Objašnjenje / Primedbe

CLS postavlja:

- Tekstualni kursor u gornji levi ugao ekrana.
- Koordinatni početak za grafiku (tačku 0,0) u donji levi ugao ekrana.
- Registar za pomeranje ekrana (*scroll*) na nulu.

NAPOMENA: Umesto CLS može se koristiti i `PRINT CHR$(27);"E"`.

primer:

```
10 CLS:MOVE 0,0
20 PRINT "OVDE POCINJE TEKST"
30 DRAW 10,10,0,3 'A ODAVDE GRAFIKA
```

4.7 COMMON

Format: `COMMON` <lista promenljivih>
Namena: Čuvanje vrednosti promenljivih iz <liste promenljivih> pri pozivanju novog programa (vidi instrukciju CHAIN).

Objašnjenje / Primedbe

COMMON instrukcija upotrebljava se uz CHAIN. COMMON može biti bilo gde u programu (najbolje na početku). Ukoliko ima više COMMON instrukcija, imena promenljivih navedenih u jednoj ne treba ponavljati u ostalim. Nizovi se obeležavaju imenom i zagradama `()`. Ukoliko treba preneti sve promenljive, upotrebite opciju ALL u CHAIN, a izostavite COMMON.

primer:

```
100 COMMON A,B,C,NIZ(),ZNACI$
110 CHAIN "P2",10
```

4.8 CONT

Format: CONT
Namena: Nastavak izvodjenja programa posle prekida.

Objašnjenje / Primedbe

Program se može prekinuti pritiskanjem na kombinaciju tastera CTRL i C. Program takodje prekida svoj rad nailaskom na instrukcije STOP i END. Komandom CONT nastavlja se izvršavanje programa tačno na onoj instrukciji gde je bio prekinut. Ukoliko je program bio prekinut na instrukciji INPUT, pa nastavljen, znak za unos (?) ili poruka biće ponovo ispisani.

CONT se obično upotrebljava uz instrukciju STOP kod ispravljanja grešaka u programu. Kada se izvršavanje programa prekine mogu se u komandnom načinu rada gledati i menjati vrednosti promenljivih, listati program itd. a zatim nastaviti sa CONT ili sa GOTO iz komandne linije. CONT se može upotrebiti i za nastavak programa posle prijavljene poruke o grešci. CONT ne radi ukoliko je za vreme prekida ispravljan program.

4.9 DATA

Format: DATA <lista konstanti>
Namena: Čuvanje podataka u programu.

Objašnjenje / Primedbe

DATA instrukcija se ne izvršava, te tako može biti bilo gde u programu. <listu konstanti> čine svi tipovi konstanti u BASICu. Konstante su medjusobno razdvojene zarezom. Za znakovne konstante važi da moraju biti pod znacima navoda (") ukoliko sadrže vodeće i prateće blanko znakove, zarez ili dve tačke. U drugim slučajevima navodnici nisu neophodni. DATA instrukcije formiraju jedan neprekidan niz podataka (DATA listu). Izmedju njih se može nalaziti proizvoljan broj izvršnih instrukcija programa.

Tipovi konstanti se moraju poklapati sa tipovima promenljivih u READ instrukciji. Konstante se čitaju redom od početka do kraja DATA liste. Za ponovno čitanje DATA liste vidi RESTORE instrukciju.

primer:

```
10 DATA 5,7,9,-1,12,abcd
20 DATA "A:B=C:A","3,123",11,23,555,1E-9
```

4.10 DEFFN

Format: DEF FN <ime> [(<lista promenljivih>)] = <definicija funkcije>
Namena: Definisanje nove funkcije.

Objašnjenje / Primedbe

<ime> se formira po istim pravilima kao ime promenljive. Definisana funkcija poziva se u programu sa FN<ime>.

<lista promenljivih> sadrži imena promenljivih koja će biti upotrebljena u <definiciji funkcije>. Ova imena samo naznačavaju kakav tip promenljive će biti upotrebljen i biće zamenjena vrednostima promenljivih navedenim u pozivu funkcije. Promenljive u listi su medjusobno razdvojene zarezima.

<definicija funkcije> je aritmetički ili znakovni izraz koji može biti dugačak najviše jedan programski red. Imena promenljivih u definiciji funkcije nemaju uticaj na promenljive iz programa. Ako je u <definiciji funkcije> navedeno ime promenljive koja se ne nalazi u <listi promenljivih>, uzeće se vrednost promenljive istog imena iz programa.

Instrukcija DEF FN mora biti izvršena pre no što se funkcija pozove. Najbolje je na početak programa postaviti sve DEF FN. Ukoliko se dogodi da se funkcija pozove pre definisanja, ispisuje se poruka o grešci, `Undefined user function` (nedefinisana korisnička funkcija). Funkcija se ne može definisati u komandnom načinu rada.

primer:

```
10 DEF FNZBIRKVAD(X,Y)=X^2+Y^2
20 A=5:B=7
30 PRINT FNZBIRKVAD(A,B),FNZBIRKVAD(1,3)
RUN
 74             10
Ok
```

Linija 10 definiše funkciju ZBIRKVAD, a sledeća ispisuje vrednosti zbira kvadrata za promenljive A i B, i konstante 1 i 3.

4.13 DELETE

Format: DELETE [<broj linije1>][-<broj linije2>]
Namena: Brisanje programske linije ili grupe linija.

Objašnjenje / Primedbe

DELETE briše linije u zadatom opsegu. Ukoliko se ne navede nijedan <broj linije>, ispisuje se greška `Illegal function call` (nedozvoljen poziv funkcije). Posle instrukcije DELETE, BASIC se vraća u komandni način rada.

primer:

`DELETE 40`	briše liniju 40
`DELETE 100-500`	briše linije od 100 do 500 uključujući i njih
`DELETE -40`	briše linije od početka do 40 uključujući i nju
`DELETE 40-`	briše linije od 40 do kraja

NE RADI

4.14 DIM

Format: DIM <lista nizova>
Namena: Rezervisanje mesta u memoriji za nizove, postavljanje maksimalnog člana niza i dodeljivanje nulte vrednosti svim članovima niza.

Objašnjenje / Primedbe

<lista nizova> sadrži imena nizova odvojena zarezima. Uz imena se u malim zagradama navode dimenzije nizova odvojene zarezima. Maksimalan broj dimenzija je 255, a broj članova po dimenziji 32768. Ako se operiše sa članom niza čiji je indeks veći od definisanog u DIM instrukciji, ispisuje se poruka `Subscript out of range` (indeks izvan opsega).

Ukoliko nije definisan DIM instrukcijom, niz može imati maksimalno 11 članova.

Indeksi nizova počinju od 0.

Ponovni pokušaj dimenzionisanja niza izazvaće grešku `Redimensioned array` (ponovljeno dimenzionisanje).

primer:

`10 DIM A(20),B(70)`	Rezerviše mesta za niz A sa 21 elementom i niz B sa 71 elementom.
`10 DIM A(10,20,5,1)`	Rezerviše mesto za četvoro-dimenzionalni niz od 11 × 21 × 6 × 2 elemenata.

4.15 DRAW

Format: DRAW <x koord.>,<y koord.>[,<aps/rel>[,<intenzitet>]]

Namena: Crtanje linije od pozicije grafičkog kursora do tačke zadate apsolutnim koordinatama; crtanje vektora zadatog priraštajima u X i Y pravcu računatih od grafičkog kursora; zadavanje intenziteta linije.

Objašnjenje / Primedbe

<x koord.> i <y koord.> su grafičke koordinate ili priraštaji u pravcu apscise i ordinate računati od pozicije grafičkog kursora. Opseg vrednosti aritmetičkog izraza za <x koord.> i <y koord.> je [-4096,4095]. Na ekranu je vidljiv prvi kvadrant po X osi [0,511] i po Y osi [0,255].

<aps/rel> je aritmetički izraz čija vrednost odredjuje način zadavanja koordinata. Ako se izostavi ili ima vrednost nula, linija se povlači do apsolutno zadatih koordinata. Za vrednost različitu od nule <x koord.> i <y koord.> su priraštaji u pravcu apscise i ordinate, računati od pozicije grafičkog kursora.

<intenzitet> može imati vrednosti iz opsega [0,3]. Ako se izostavi, podrazumeva se prethodno zadata vrednost. Intenzitet linije na ekranu proporcionalan je vrednosti izraza <intenzitet>.

DRAW instrukcija postavlja novu poziciju grafičkog kursora na kraj nacrtane linije.

primer:

```
10 CLS:MOVE 0,0
20 DRAW 256,1000,0,2
30 DRAW 512,0
50 DRAW 256,1000,0,0
60 DRAW 0,0,0,0
70 DRAW 100,100,1,3
80 DRAW -10,-20,1,1
```

4.16 EDIT

Format: EDIT <broj linije>
Namena: Izmena sadržaja programske linije.

Objašnjenje / Primedbe

EDIT komanda omogućava izmenu teksta programske linije. Kada se izvrši, ispiše se broj linije čiji se sadržaj menja. Izmena se vrši EDIT potkomandama.

EDIT potkomande služe za:

1. Pomeranje kursora po liniji
2. Ubacivanje novog teksta u liniju
3. Brisanje teksta iz linije
4. Pronalaženje teksta u liniji
5. Izmenu teksta u liniji
6. Početak ili kraj izmene teksta linije

U tekstu koji sledi: <ZN> je bilo koji znak, <TEKST> pretstavlja niz znakova neodredjene dužine, <CK> označava opcionu celobrojnu konstantu (ako se ne navede, podrazumeva se vrednost 1).

1. Pomeranje kursora po programskoj liniji

BLANKO: Upotrebljava se za kretanje unapred za jedno ili [<CK>] BLANKO za <CK> mesta.

[BS] : se ponaša kao i blanko samo u suprotnom smeru - prema početku programske linije [<CK>] [BS] vraća se <CK> mesta unatrag.

2. Ubacivanje novog teksta

I: I<TEKST> [ESC] sekvenca ubaciće na mesto kursora navedeni tekst. Ubačeni tekst ispisuje se na ekranu. Ako se otkuca [RET] taster, ispisaće se cela linija sa umetnutim tekstom i završiti ispravljanje linije. Prilikom umetanja teksta koristiti [BS] za ispravke pogrešno unetih znakova. Ukoliko se desi da dužina umetnutog teksta i dužina linije koja se ispravlja zajedno prelaze dužinu od 255 znakova, greška se signalizira piskom i znak koji prelazi dužinu od 255 se ignoriše.

X: X potkomanda je korisna za dodavanje teksta sa desne strane linije. Ona će postaviti kursor na kraj linije i uključiti umetanje teksta.

3. Brisanje teksta

D: [<CK>] D obrisaće <CK> znakova od kursora na desno. Obrisani znaci ispisuju se izmedju kosih crta (/). Ukoliko je <CK> veće od broja znakova sa desne strane brišu se samo znaci do kraja linije.

H: H potkomanda automatski briše sve znake sa desne strane kursora i uključuje umetanje teksta. Korisna je za izmenu kraja programske linije.

4. Pronalaženje teksta

S: Potkomanda [<CK>]S<ZN> traži znak jednak <ZN> i to <CK> pojavljivanje u liniji od pozicije kursora. Kursor se postavi na poziciju ispred pronadjenog znaka. Ukoliko se <ZN> ne nadje u liniji, kursor se postavlja na kraj linije.

K: [<CK>]K<ZN> isto se ponaša kao i S potkomanda, s tim što se brišu svi znaci preko kojih prelazi kursor. Kursor se postavi na poziciju pre <ZN>, a svi obrisani znaci su prikazani zatvoreni u kose crte (/).

5. Izmena teksta u liniji

C:C<ZN> menja sledeći znak u <ZN>. Potkomanda <CK>C<ZN_1><ZN_2>..<$ZN_{<CK>}$> zameniće <CK> znakova zadatim znacima.

6. Početak ili kraj izmene teksta programske linije

RET : Prikazuje ostatak linije, neispisan na ekranu, i završava ispravljanje teksta linije.

E: E potkomanda se ponaša isto kao i RET s tim što se ostatak ne prikazuje.

Q: Napušta se izmena teksta. Bez obzira na ono što je već izmenjeno, zadržava se prvobitni tekst linije.

L: L potkomanda upotrebljava se za listanje linije koja će se ispravljati ili je već ispravljana. L postavlja kursor na početak linije.

A: Lista originalnu (neispravljanu) liniju.

Greške prilikom izmene teksta linije izazivaju zvučni signal. Kada nastupi sintaksna (pravopisna) greška **Syntax error** BASIC interpretator automatski izvršava EDIT komandu sa brojem linije u kojoj je greška nastala. Ako se pritisne RET ili E u tom trenutku biće zaboravljene vrednosti promenljivih u memoriji (zbog menjanja programa), te ukoliko želite prvo ispitati promenljive, zadajte Q potkomandu.

CTRL A omogućava izmenu teksta komandne linije.

4.17 ELIPSE

Format: ELIPSE <radijus x>[,<radijus y>[,<intenzitet>]]

Namena: Crtanje elipse ili kruga zadatog intenziteta sa centrom na poziciji grafičkog kursora.

Objašnjenje / Primedbe

<radijus x> je dužina poluprečnika po x osi.

Ako je <radijus y> izostavljen, biće izračunat tako da elipsa na ekranu izgleda kao krug. Odnos X i Y ose je približno 2 prema 3 ($y : x = 171 : 256$).

<radijus x> i <radijus y> mogu imati vrednosti iz opsega [1,2047]. Za vrednosti manje od 1 i vrednosti veće od 2048 elipsa se neće nacrtati.

<intenzitet> može imati vrednosti iz opsega [0,3]. Ako se izostavi, podrazumeva se prethodno zadata vrednost. Intenzitet elipse ili kruga na ekranu je proporcionalan vrednosti izraza <intenzitet>.

ELIPSE ne menja poziciju grafičkog kursora.

primer:

```
10 CLS
20 MOVE 256,128,0,3
30 ELIPSE 40,40
40 MOVE 0,0,1,2 'intenzitet kruga se menja ovako
50 ELIPSE 40
60 MOVE 256,-50
70 ELIPSE 100,70,1
RUN
```

4.18 END

Format: END

Namena: Prekid izvršavanja programa, zatvaranje svih datoteka i povratak u komandni način rada.

Objašnjenje / Primedbe

END instrukcija može se nalaziti bilo gde u programu. END ne ispisuje BREAK poruku koja se inače pojavljuje kod STOP-a. END instrukcija ne mora stajati na kraju programa.

primer:

```
940 IF K>1000 THEN END ELSE GOTO 20
```

4.19 ERASE

Format: ERASE <lista nizova>
Namena: Oslobadjanje memorijskog prostora zauzetog nizovima čija su imena navedena u <listi nizova>.

Objašnjenje / Primedbe

Nizovi mogu biti ponovo dimenzionisani po izvršenju ERASE instrukcije. Mesto u memoriji koje je ranije bilo zauzeto može se upotrebiti za druge namene.

primer:

```
10 DIM A(500),B(1,100)
 -
 -
500 ERASE A
510 DIM A(300)
 -
 -
```

4.20 ERR i ERL promenljive

Format: Promenljiva u izrazu.
Namena: Informacije o nastaloj grešci.

Objašnjenje / Primedbe

Kada nastupi greška, u promenljivu ERR upiše se kod greške, a u ERL broj linije u kojoj je greška nastupila. Ako je greška nastupila u komandnom načinu rada, ERL sadrži 65535. UPOZORENJE:

- kada se želi ispitati greška u komandnoj liniji (važi za INPUT instrukciju) pisati:

```
IF 65535 = ERL THEN ....
```

- u ostalim slučajevima pisati:

```
IF ERR = KODGRESKE THEN ....
IF ERL = 1290 THEN ...
```

Ako se kod greške nalazi na desnoj strani znaka jednakosti, RENUM komanda će ga smatrati brojem programske linije i promeniće ga pri renumerisanju programa.

Promenljivama ERL i ERR se ne može dodeljivati vrednost.

primer:

Vidi **dodatak C** – Poruke o greškama.

4.21 ERROR

Format: ERROR <celobrojni izraz>

Namena: 1) Simuliranje pojavljivanja greške.
2) Definisanje kodova za nove greške (koje se mogu javiti pri upotrebi programa).

Objašnjenje / Primedbe

Vrednost <celobrojnog izraza> mora biti u opsegu [0,255]. Ako je vrednost <celobrojni izraz> jednaka kodu greške koja postoji u BASICu, ispisuje se odgovarajuća poruka. Vidi dodatak **Poruke o greškama**.

Za definisanje novog koda greške, koristite vrednosti veće od najvećeg koda greške BASICa.

Ukoliko je naveden kod greške koja ne postoji u BASICu a nije definisano kako će se greška obradjivati, ispisaće se `Unprintable error` (neispisiva greška) i izvršavanje programa će se zaustaviti.

Vidi ON ERROR... i RESUME... instrukcije.

primer:

```
10 S=10
20 T=5
30 ERROR S+T
40 END
RUN
String too long in line 30
```

```
-
110 ON ERROR GOTO 400
120 INPUT "KOLIKI JE NOVCANI ULOG",B
130 IF B>5000 THEN ERROR 210
-
-
400 IF ERROR = 210 THEN PRINT "ULOG NE MOZE BITI VECI OD 5000"
410 IF ERL = 130 THEN RESUME 120
-
-
```

4.22 FIELD

Format: `FIELD [#]`<broj datoteke>`,`<dužina polja>` AS `<znakovna promenljiva>...

Namena: Definisanje mesta u memoriji (u daljem tekstu bafer) za formiranje jednog zapisa datoteke sa slučajnim pristupom.

Objašnjenje / Primedbe

<broj datoteke> je broj pod kojim je datoteka otvorena (sa OPEN).

<dužina polja> je broj znakova koji treba rezervisati za <znakovnu promenljivu>. Na primer:

```
FIELD 1, 20 AS N$, 10 AS ID$, 40 AS ADD$
```

Definiše prvih 20 mesta (bajtova) za znakovnu promenljivu N$ u baferu za datoteku sa slučajnim pristupom, sledećih 10 mesta za ID$, i zatim 40 mesta za ADD$. FIELD ne stavlja nikakve podatke u bafer (vidi LSET, RSET i GET instrukcije).

Ukupan broj bajtova definisanih u FIELD instrukciji ne sme prelaziti ukupan broj bajtova jednog zapisa (vidi OPEN instrukciju). Ukoliko se to desi, ispisuje se greška **`Field overflow`** (prekoračenje dužine polja).

Može se izvršiti bilo koji broj FIELD instrukcija nad istom datotekom i sve FIELD definicije važe u isto vreme. Ovim načinom moguće je definisati grupna polja.

UPOZORENJE:

FIELD preusmerava znakovne promenljive u bafer za datoteku sa slučajnim pristupom. Zato, ako je jedna znakovna promenljiva definisana u FIELD instrukciji, NE KORISTITI JE u INPUT instrukciji ili dodeljivanju.

FIELD instrukcija mora biti izvršena. pre uzimanja podataka sa diska (GET instrukcijom) ili stavljanja podataka na disk (PUT instrukcijom).

primer:

Vidi **Dodatak A**

4.23 FILES

Format: FILES [<ime datoteke>]
Namena: Prikazuje imena datoteka na (zadatom) disku.

Objašnjenje / Primedbe

Ako je <ime datoteke> izostavljeno, prikazaće se imena svih datoteka koje se nalaze na disku.

<ime datoteke> je grupa znakova koja može sadržati i specijalne znakove za zamenu jednog ili više znakova. Znak pitanja (?) se koristi za zamenu jednog znaka, a zvezdica (*) za više.

primer:

`FILES`	će prikazati imena svih datoteka na disku
`FILES "*.BAS"`	će prikazati imena svih BASIC programa
`FILES "M:*.*"`	će prikazati imena svih datoteka na memorijskom disku
`FILES "TEST?.BAS"`	će prikazati imena svih datoteka tipa TEST1.BAS, TEST2.BAS, TESTA.BAS tj. onih koji uz TEST imaju još jedan znak
`FILES "T*.*"`	će prikazati imena datoteka koje počinju slovom T

4.24 FILL

Format: FILL <x koord.>,<y koord.> [,<aps/rel>[,<intenzitet>]]
Namena: Popunjavanje površine grafičkog ekrana do granice omedjene zadatim intenzitetom.

Objašnjenje / Primedbe

<x koord.> i <y koord.> su grafičke koordinate ili priraštaji u pravcu apscise i ordinate, računati od pozicije grafičkog kursora. Opseg vrednosti aritmetičkog izraza za <x koord.> i <y koord.> je [-4096,4095]. Na ekranu je vidljiv prvi kvadrant po X osi [0,511] i po Y osi [0,255]. Površina se popunjava počevši od tačke zadate ovim koordinatama.

<aps/rel> je aritmetički izraz čija vrednost odredjuje način zadavanja koordinata. Ako se izostavi ili ima vrednost nula, linija se povlači do apsolutno zadatih koordinata. Za vrednost različitu od nule <x koord.> i <y koord.> su priraštaji u pravcu apscise i ordinate računati od pozicije grafičkog kursora.

<intenzitet> može imati vrednosti iz opsega [0,3]; a ako se izostavi, podrazumeva se prethodno zadata vrednost. Intenzitet popunjene površine je proporcionalan vrednosti <intenzitet>. Površina se popunjava <intenzitetom> do granice istog intenziteta preko svih različitih intenziteta.

FILL instrukcija ne postavlja novu poziciju grafičkog kursora.
FILL neće ispuniti površinu u sledećim slučajevima:

- Ako je zadata tačka van vidljivog ekrana (ili van limita za koordinate)
- Ako je intenzitet tačke od koje počinje popunjavanje jednak zadatom intenzitetu

primer:

```
10 CLS
20 MOVE 0,0,0,3
30 ELIPSE 128
40 MOVE 100,100,1,1
50 ELIPSE 100,100
60 FILL 0,0,1,1
70 FILL 0,0,1,0
80 GOTO 10
```

4.25 FOR..NEXT

Format: `FOR <np>=<i1> TO <i2> [STEP <i3>]`
-
- (telo petlje)
-
`NEXT [<np>][,<np>]...`
<np> = numerička promenljiva
<i1>, <i2>, <i3> =numerički izrazi

Namena: Omogućiti da se izvestan broj instrukcija (telo petlje) ponavlja tačno odredjen broj puta.

Objašnjenje / Primedbe

<np> je promenljiva u kojoj se čuva broj ponavljanja (brojač).

Prvi numerički izraz <i1> je početna vrednost brojača.

Drugi numerički izraz <i2> je vrednost do koje se uvećava (smanjuje) brojač.

<i3> je vrednost za koju se uvećava (ili smanjuje) brojač.

Instrukcije koje će se ponavljati (telo petlje) slede iza FOR i završavaju se sa NEXT.

Kada BASIC naidje na NEXT on uveća brojač <np> za <i3>, proveri da li je vrednost brojača veća od <i2>; ako nije, onda se vraća na instrukciju iza FOR i ponavlja instrukcije; ako jeste onda izvršava instrukcije iza NEXT-a.

Ukoliko STEP nije upotrebljen podrazumeva se da je uvećanje 1. Za petlje koje smanjuju brojač navedite uz STEP negativnu vrednost, a izraz <i2> treba da bude manji od <i1>. Brojač će se smanjivati u svakom prolazu a testiraće se kraj za <np> manje od <i2>.

Telo petlje će biti preskočeno ako je <i1> puta znak <i3> veće od <i2> puta znak od <i3>, odnosno:

- ako je početna vrednost veća od krajnje, a brojač se uvećava ili,
- ako je početna vrednost manja od krajnje a brojač se umanjuje.

Petlje u petlji:

Kada su petlje unutar petlji svaka od njih mora imati jedinstvenu numeričku promenljivu <np>.

Ako se više petlji završava na istom mestu, samo jedna NEXT instrukcija može biti upotrebljena (NEXT <np>, <np>, <np> ...).

Promenljive iza NEXT mogu biti izostavljene; tada se NEXT odnosi samo na prvu prethodnu FOR instrukciju (po redosledu izvršavanja, a ne po linijskim brojevima).

Ako se NEXT izvrši pre odgovarajuće FOR instrukcije ispisuje se greška `NEXT without FOR` (NEXT bez FOR) a izvršavanje programa se prekida.

Ako se NEXT izostavi ispisaće se greška `FOR without NEXT` (FOR bez NEXT).

primer:

```
10 K=10
20 FOR I=1 TO K STEP 2
30 PRINT I
40 K=K+10
50 PRINT K
60 NEXT
RUN
 1 20
 3 30
 5 40
 7 50
 9 60
Ok
```

```
10 J=0
20 FOR I=1 TO J
30 PRINT I
40 NEXT I
```

U ovom primeru telo petlje se ne izvršava jer početna vrednost prevazilazi krajnju.

```
10 I=5
20 FOR I=1 TO I+5
30 PRINT I;
40 NEXT RUN
 1 2 3 4 5 6 7 8 9 10
Ok
```

Prvo se izračunavaju početna i krajnja vrednost brojača.

```
10 FOR I=1 to 2    ili    10 FOR I=1 to 2
20 FOR J=1 to 2           20 FOR J=1 to 2
30 PRINT j                30 PRINT j
40 NEXT J,I               40 NEXT
RUN                       50 NEXT
 1                        RUN
 2                         1
 1                         2
 2                         1
                           2
```

Kraj više petlji na istom mestu.

4.26 GET,

Format: GET [#]<br. datoteke>[,<broj zapisa>]
Namena: Čitanje zapisa u bafer iz datoteke sa slučajnim pristupom.

Objašnjenje / Primedbe

<br. datoteke> je broj pod kojim je datoteka otvorena (sa OPEN).

<broj zapisa> je redni broj zapisa za čitanje. U slučaju da je izostavljen, uzima se sledeći zapis po redu. Najveći mogući redni <broj zapisa> je 32767.

primer:

Vidi **Dodatak A**

4.27 GOSUB ... RETURN

Format: GOSUB <br. linije>

```
<br linije>početak potprograma
-
RETURN
```

Namena: Skok u potprogram (GOSUB) i povratak iz potprograma (RETURN)

Objašnjenje / Primedbe

<br. linije> je broj programske linije od koje počinje potprogram. Potprogram može biti izvršen više puta u programu. Unutar potprograma može postojati skok u sledeći potprogram. Dubina skakanja iz potprograma u potprogram je ograničena jedino dubinom steka.

RETURN instrukcija vrši povratak iz potprograma tj. vrši skok na instrukciju iza GOSUB koja je pozvala potprogram. Potprogram može sadržati više RETURN instrukcija zavisno od potrebe povratka iz potprograma. Potprogrami mogu biti bilo gde u programu, mada se preporučuje da budu odvojeni od glavnog programa sa STOP, END ili GOTO instrukcijama.

primer:

```
10 GOSUB 100
20 PRINT "POVRATAK IZ POTPROGRAMA"
30 END
100 PRINT "POTPROGRAM ";
110 PRINT "U ";
120 PRINT "TOKU ... ."
130 RETURN
RUN
POTPROGRAM U TOKU ... .
POVRATAK IZ POTPROGRAMA
Ok
```

4.28 GOTO

Format: GOTO <br. linije>
Namena: Bezuslovni skok na zadatu programsku liniju.

Objašnjenje / Primedbe

Ako u programu postoji linija sa <br. linije>, onda će se izvršavanje programa nastaviti od te linije nadalje. Ukoliko linija sa <br. linije> ne postoji ispisuje se greška **Undefined line number in** xxxxx (nedefinisan broj linije u xxxxx).

primer:

```
10 READ R
20 PRINT "R =";R,
30 P=3.14*R^2
40 PRINT "POVRSINA =";P
50 GOTO 10
60 DATA 5,7,12
RUN
R = 5      POVRSINA = 78.5
R = 7      POVRSINA = 153.86
R = 12      POVRSINA = 452.16
?Out of data in 10
Ok
```

4.29 IF ... THEN ... ELSE i IF ... GOTO

Format: 1: `IF` <iz.> `THEN` <ins>/<br.lin.>[`ELSE` <ins>/<br.lin.>]
2: `IF` <iz.> `GOTO` <br.lin.> [`ELSE` <ins>/<br.lin.>]
<A> / <B> = ili A ili B (upotrebljava se jedno od ponudjenih)
<iz> = aritmetički ili logički izraz
<ins> = instrukcija ili instrukcije
<br. lin> = broj programske linije

Namena: Na osnovu izraza <iz> odlučuje se o daljem odvijanju programskog toka (uslovni skok).

Objašnjenje / Primedbe

Ako rezultat <iz.> nije nula (tačan izraz):

- Izvršavaju se instrukcije iza THEN do ELSE ako postoji, a zatim se prelazi na sledeći programski red, ukoliko iza THEN ne postoji skok na neki drugi programski red.
- Ako je iza THEN ili GOTO broj programske linije, vrši se skok na programsku liniju <br. lin.>.

Ako je rezultat <iz.> nula, (netačan izraz):

- Izvršavaju se instrukcije iza ELSE, a zatim nastavlja izvodjenje od sledeće programske linije, ukoliko ne postoji skok na neku drugu liniju.
- Izvršava se programski red čiji je broj naveden iza ELSE.
- Ako ELSE nije naveden, izvršavanje se nastavlja od sledeće programske linije.

Legalna je i konstrukcija `IF` <iz> `THEN GOTO` <br. lin.>.

Odluka u odluci:
Može se praviti odluka u odluci što je ograničeno samo dužinom programske linije. Na primer:

```
IF X>Y THEN PRINT"VECE" ELSE IF Y>X THEN PRINT"MANJE" ELSE PRINT"JEDNAKO"
```

Ako programski red ne sadrži isti broj ELSE i THEN instrukcija, svako ELSE odgovara prvom prethodnom THEN. Na primer:

```
IF A=B THEN IF B=C THEN PRINT "A=C" ELSE PRINT "A<>C"
```

ne ispisuje A<>C za uslov A<>B već se prelazi na izvršavanje sledećeg programskog reda jer ELSE odgovara THEN-u iz drugog IF-a.

UPOZORENJE:

Kada upotrebljavate IF za testiranje jednakosti za vrednosti koje su rezultat operacija u pokretnom zarezu, prisetite se konačne tačnosti računara. Može se dogoditi da rezultat odstupa od očekivanog. Na primer:

```
IF 0.1^2 = 0.01 THEN ...
```

neće biti jednako

Još neki primeri odluka:

```
200 IF I THEN GET#1,I
```

će učitavati sa diska zapise čiji je redni broj različit od nule.

```
100 IF (I<20)*(I>10) THEN DB=I*5:GOTO 300
110 PRINT "Izvan opsega"
```

U ovom primeru test glasi: ako je I veće od 10 i manje od 20. U slučaju da je uslov zadovoljen računa se DB i skače na liniju 300. Ako I nije u zadatom opsegu, izvršavanje se nastavlja na liniji 110. Upotrebljeno je množenje umesto logičkog operatora AND.

```
210 IF IOFLAG THEN PRINT A$ ELSE LPRINT A$
```

U zavisnosti od promenljive IOFLAG vršiće se ispis na ekran (IOFLAG<>0) odnosno štampač (IOFLAG=0).

4.30 INPUT

Format: INPUT [;][<poruka>;] <lista promenljivih>
Namena: Omogućiti unos podataka sa tastature za vreme rada programa.

Objašnjenje / Primedbe

INPUT instrukcija zaustavlja program i ispisuje znak pitanja na ekranu kao poruku da se očekuju podaci.

Ako postoji <poruka>, biće ispisana pre znaka pitanja. Zatim se očekuju podaci sa tastature. Otkucani podaci se završavaju tasterom RET.

Ako se umesto tačka-zareza (;) iza poruke upotrebi zarez (,), znak pitanja se neće ispisati na ekranu. Na primer:

`INPUT "X= ",X` će ispisati poruku bez znaka pitanja.

Ako se iza INPUT prvo navede tačka-zarez, kursor neće preći u sledeću liniju ekrana po pritisku na taster RET.

Podaci se unose po redosledu kojim se dodeljuju promenljivama u <listi promenljivih>. Podaci se odvajaju zarezom. Podataka treba da ima isto koliko i promenljivih u listi.

Tip podatka koji se unosi mora biti u skladu sa tipom odgovarajuće promenljive iz liste. Znakovna konstanta koja se unosi ne mora imati znake navoda (vidi objašnjenje uz DATA instrukciju).

Unos previše ili premalo podataka, kao i unos pogrešnog tipa podatka, prouzrokovaće grešku `?Redo from start` (uradi ponovo od početka).

primer:

```
10 INPUT X
20 PRINT X;" NA KVADRAT JE ";X^2
30 END
RUN
? 5
 5 NA KVADRAT JE 25
Ok
```

```
10 PI=3.14
20 INPUT "KOLIKI JE POLUPRECNIK ";R
30 A=PI*R^2
40 PRINT "POVRSINA KRUGA JE";A
50 PRINT
60 GOTO 20
RUN
KOLIKI JE POLUPRECNIK? 7.4
POVRSINA KRUGA JE 171.946
```

4.31 INPUT#

Format: INPUT# <br. datoteke>, <lista promenljivih>

Namena: Čitanje podataka iz sekvencijalne datoteke ili bafera za datoteku sa slučajnim pristupom i dodeljivanje podataka promenljivama u <listi promenljivih>.

Objašnjenje / Primedbe

<br. datoteke> je broj pod kojim je datoteka otvorena za ulaz (sa OPEN).

<lista promenljivih> sadrži spisak promenljivih kojima će se po redosledu pojavljivanja dodeljivati vrednosti učitane iz datoteke. Tipovi podataka u datoteci moraju odgovarati tipu promenljive kojoj se dodeljuju.

Kada se čitaju numeričke vrednosti iz datoteke, sledeći znaci služe za odvajanje podataka: blanko (kod 32), CR (kod 13) ,LF (kod 10) i zarez (,).

Ako INPUT# očekuje znakovni podatak, ignorišu se vodeća prazna mesta, CR i LF , a znakom podatka se smatra prvi znak različit od navedenih.

Kada je prvi znak, znak navoda ("), podaci za znakovnu promenljivu su svi znaci izmedju ovog i sledećeg znaka navoda. Zato znakovni podaci ne mogu sadržati znak navoda.

Ukoliko pak znakovni podaci nisu ograničeni znacima navoda, znakovi će se učitavati u promenljivu sve dok ne naidje zarez, CR, LF ili dok se ne učita 255 znakova.

EOF (*End Of File* - kraj datoteke ili CTRL Z, čiji je kod 26) je znak kraja podataka u datoteci.

primer:

Vidi u PRINT#, WRITE# i **Dodatak A.**

4.32 KILL

Format: KILL <ime datoteke>
Namena: Brisanje datoteke (programa) sa diska.

Objašnjenje / Primedbe

<ime datoteke> je puno ime datoteke (sa nastavkom).

KILL se upotrebljava za brisanje svih vrsta datoteka : programa snimljenih na disk, datoteka sa slučajnim pristupom i sekvencijalnih datoteka.

Ako se KILL naredba upotrebi da obriše datoteku koja je otvorena radi čitanja (sa OPEN), ispisuje se greška **File already open** (Datoteka je još otvorena).

primer:

```
200 KILL "PODACI.DAT"
KILL "*.*"  (briše sve datoteke)
Ok
KILL "*.BAS"  (briše sve BASIC programe)
Ok
```

4.33 LINE INPUT

Format: LINE INPUT [;][<poruka>;] <znakovna promenljiva>
Namena: Unos velikih grupa znakova (do 254 znaka) u znakovnu promenljivu.

Objašnjenje / Primedbe

Ako postoji <poruka>, ispisaće se pre prihvatanja podataka. Znak pitanja se ne ispisuje, osim u slučaju da je deo poruke.

Svi znaci od početka unosa do [RET] biće dodeljeni <znakovnoj promenljivoj>. Ako se [LF] otkuca neposredno pre [RET] , oba znaka se dodeljuju <znakovnoj promenljivoj>, a unos se nastavlja dok se ponovo ne otkuca [RET].

Tačka-zarez (;) iza LINE INPUT omogućava da se po završetku unosa ne pomera kursor u sledeću liniju na ekranu.

LINE INPUT može biti prekinut sa [CTRL] C i tada se BASIC vraća u komandni način rada. Komandom CONT program nastavlja ponovnim izvršavanjem LINE INPUT.

primer:

Vidi LINE INPUT#.

4.34 LINE INPUT#

Format: LINE INPUT#<br. datoteke>,<znakovna promenljiva>

Namena: Unos velikih grupa znakova (do 254 znaka) iz sekvencijalne datoteke (sa diska) u znakovnu promenljivu.

Objašnjenje / Primedbe

<br. datoteke> je broj pod kojim je datoteka otvorena (sa OPEN).

Promenljivoj <znakovna promenljiva> se dodeljuju podaci učitani iz datoteke.

LINE INPUT# čita podatke do CR. Ako naidje sekvenca LF i CR, sačuvaće se u promenljivoj i čitanje se nastavlja do sledećeg CR.

Instrukcija LINE INPUT# je korisna ako je svaka linija u datoteci rasparčana u polja ili ako se čita program zapisan na disku u ASCII načinu zapisa (za tekst datoteke).

primer:

```
 5 REM .... UPIS U DATOTEKU
 10 OPEN "O",1,"TELEFONI"
 20 LINE INPUT "IME I TELEFON -";C$
 30 PRINT #1,C$
 40 CLOSE 1
 50 REM ....CITANJE DATOTEKE
 60 OPEN "I",1,"TELEFONI"
 70 LINE INPUT #1,C$
 80 PRINT C$
 90 CLOSE 1
 100 END
 RUN
 IME I TELEFON -PETROVIC JOVAN, 224312
 PETROVIC JOVAN, 224312
 Ok
```

4.35 LIST

Format: LIST [<broj linije> [-[<broj linije>]]]

Namena: Prikazivanje na ekranu programa ili dela programa koji se nalazi u memoriji.

Objašnjenje / Primedbe

Posle LIST komande BASIC prelazi u komandni način rada. Prikaz programa se može: prekinuti (pritiskanjem tastera CTRL i C), zaustaviti (pritiskanjem tastera CTRL i S) i nastaviti (pritiskanjem tastera CTRL i Q).

primer:

LIST	prikazuje ceo program
LIST 500	prikazuje liniju sa brojem 500 (ako postoji)
LIST 150-	prikazuje liniju 150 i sve koje sledе do kraja programa
LIST -1000	prikazuje linije od početka programa do linije 1000 uključujući i nju
LIST 150-1000	prikazuje linije od 150 do 1000 uključujući i njih

4.36 LLIST

Format: LLIST [<br.linije>[-[<br.linije>]]]

Namena: Štampanje programa ili dela programa, koji se nalazi u memoriji, na štampaču.

Objašnjenje / Primedbe

LLIST podrazumeva da je štampač širine 132 znaka. Objašnjenje za LLIST je isto kao i za LIST. Za postavljanje širine reda na štampaču vidi instrukciju WIDTH.

primer:

vidi LIST instrukciju.

4.37 LOAD

Format: LOAD <program>[,R]
Namena: Učitavanje programa sa diska u programsku memoriju.

Objašnjenje / Primedbe

<program> je ime programa snimljenog na disk. Ako se nastavak izostavi podrazumeva se .BAS.

LOAD briše promenljive, program koji je trenutno u memoriji, zatvara datoteke i učitava program sa diska.

Ako se navede R opcija, neće se zatvarati datoteke, program će se učitati sa diska i izvršiti (kao sa RUN). R opcija je korisna za programske segmente koji se učitavaju i izvršavaju jedan za drugim i medjusobno komuniciraju preko podataka u datotekama na disku.

primer:

```
LOAD "PROG1"
LOAD "DEO2",R
```

4.38 LPRINT i LPRINT USING

Format: LPRINT [<lista izraza>]
LPRINT USING <znakovni izraz>;<lista izraza>
Namena: Štampanje podataka iz programa na štampaču, sa i bez formatiranja.

Objašnjenje / Primedbe

LPRINT i LPRINT USING se ponašaju isto kao i PRINT i PRINT USING s tim što izlaz ide na štampač umesto na ekran. Za štampač se podrazumeva da je 132-kolonski ako nije drugačije definisano (vidi WIDTH).

primer:

vidi PRINT i PRINT USING

4.39 LSET i RSET

Format: LSET <znakovna promenljiva>=<znakovni izraz>
RSET <znakovna promenljiva>=<znakovni izraz>

Namena: Upisivanje poravnatih podataka u bafer za datoteku sa slučajnim pristupom. Priprema podataka za snimanje na disk (vidi PUT instrukciju).

Objašnjenje / Primedbe

Ako rezultat <znakovnog izraza> zahteva manje mesta nego što je to definisano u FIELD instrukciji:

- LSET upisuje znakove levo poravnate u polje, a
- RSET upisuje znakove desno poravnate u polje.

U preostala mesta se upisuje znak za prazno mesto (blanko).

Ako rezultat <znakovnog izraza> zahteva više mesta no što je rezervisano za <znakovnu promenljivu> u FIELD instrukciji, biće odbačen višak znakova sa desne strane (važi i za RSET i za LSET).

Numeričke vrednosti moraju pre upisa biti prevedene u znakove (vidi MKI$, MKS$ i MKD$ funkcije).

LSET i RSET mogu biti upotrebljeni i za poravnjavanje pri ispisu na ekran ili štampač.

primer:

```
10 OPEN #1,"O","PODACI"
20 FIELD #1, 30 AS IME, 10 AS TELEFON
30 I$="JOVAN"
40 T=323987
50 LSET IME$=I$
60 RSET TELEFON$=MKS$(T)
70 CLOSE 1
-

100 I$="BEOGRAD"
110 A$=SPACE$(20)
120 RSET A$=I$
-
```

RSET upotrebljen za poravnjavanje.

4.40 MERGE

Format: MERGE <ime programa>

Namena: Omogućava mešanje programa sa diska i programa u programskoj memoriji.

Objašnjenje / Primedbe

<ime programa> je znakovni izraz čija je vrednost ime programa zapamćenog na disk. Nastavak je .BAS ako nije naveden neki drugi.

Program koji se poziva mora na disk biti snimljen u ASCII formatu (kao tekst datoteka), ako nije ispisuje se poruka Bad file mode (Pogrešan način rada sa datotekom).

Ukoliko u programu na disku i programu u memoriji postoje linije sa istim brojem, programske linije u memoriji biće zamenjene linijama sa diska.

Basic posle MERGE komande prelazi u komandni način rada.

primer:

```
MERGE "PROG2"
```

4.41. MID$

Format: MID$(<zi1>,<p>[,<k>])=<zi2>
<zi1>.<zi2>= znakovni izrazi
<p>.<k>=celobrojni izrazi

Namena: Zamena dela grupe znakova drugom grupom znakova u znakovnim izrazima.

Objašnjenje / Primedbe

Znakovi u <zi1> počevši od p-tog zamenjuju se redom sa znakovima iz <zi2>.

Opcioni izraz k odredjuje koliko će se znakova menjati. Ako je k izostavljeno, <zi2> će se prepisati do dužine <zi1>.

MID$ može biti funkcija ako je sa desne strane jednakosti (vidi MID$ funkciju, poglavlje BASIC FUNKCIJE).

primer:

```
10 A$="AXXXEFGHIJKLMNOP"
20 MID$(A$,2,3)="BCD"
30 PRINT A$
RUN
ABCDEFGHIJKLMNOP
Ok
```

4.42 MOVE

Format: MOVE <x koord.>,<y koord.>[,<aps/rel>[,<intenzitet>]]

Namena: Postavljanje nove pozicije grafičkog kursora na zadate koordinate. Postavljanje boje.

Objašnjenje / Primedbe

<x koord.> i <y koord.> su grafičke koordinate ili priraštaji u pravcu apscise i ordinate, računati od pozicije grafičkog kursora. Opseg vrednosti aritmetičkog izraza za <x koord.> i <y koord.> je [-4096,4095]. Na ekranu je vidljiv prvi kvadrant po X osi [0,511] i po Y osi [0,255].

<aps/rel> je aritmetički izraz čija vrednost odredjuje način zadavanja koordinata. Ako se izostavi ili ima vrednost nula, koordinate su zadate apsolutno. Za vrednost različitu od nule <x koord.> i <y koord.> su priraštaji u pravcu apscise i ordinate, računati od pozicije grafičkog kursora.

<intenzitet> uzima vrednosti iz opsega [0,3]. Intenzitet je proporcionalan vrednosti <intenziteta>. Ako se izostavi, podrazumeva se prethodno zadata vrednost.

primer:

```
10 CLS
20 MOVE 10,10,0,2
30 DRAW 100,10
50 MOVE 10,100,1
60 DRAW 100,0
```

4.43 NAME

Format: NAME <staro> AS <novo>

Namena: Promena imena datoteke na disku.

Objašnjenje / Primedbe

<novo> ime ne sme postojati na disku, a <staro> mora, inače se prijavljuje greška. Posle promene imena datoteka ne menja mesto na disku.

primer:

```
NAME "VERZIJA1" AS "VERZIJA2"
Ok
```

datoteka menja ime iz VERZIJA1 u VERZIJA2

4.44 NEW

Format: NEW
Namena: Brisanje programa i promenljivih iz memorije

Objašnjenje / Primedbe

NEW je komanda koja se upisuje u komandnom načinu rada radi brisanja programa i promenljivih iz memorije. Basic se posle NEW uvek vraća u komandni način rada.

4.45 ON ERROR GOTO

Format: ON ERROR GOTO <br linije>
Namena: Omogućuje da se pri pojavi greške umesto prekida programa izvrši potprogram koji obradjuje grešku.

Objašnjenje / Primedbe

Kada je definisano da se greške obradjuju u programu, one se usmeravaju u deo programa koji je definisan u ON ERROR instrukciji.

<br linije> pokazuje programsku liniju od koje se obradjuju greške. Ako linija sa <br linije> ne postoji u programu, ispisuje se greška `Undefined line` (linija nije definisana).

Isključivanje programske obrade grešaka vrši se instrukcijom ON ERROR GOTO 0. Posle ove instrukcije greške se prijavljuju na uobičajeni način (porukom i prekidanjem rada programa).

Ukoliko se ON ERROR GOTO 0 upotrebi u delu programa gde se greška obradjuje, greška će biti ispisana i izvršavanje će se prekinuti. Ovo je korisno kada je greška fatalna te program ne može da je obradi.

Ako greška nastane u delu programa koji obradjuje greške, poruka greške se ispisuje, a izvršavanje programa prekida.

Za rad sa greškama vidi instrukcije ERROR i RESUME, i poglavlje PORUKE O GREŠKAMA.

primer:

```
10 ON ERROR GOTO 1000
-
-
1000 REM ... . odavde pocinje obrada gresaka
```

4.46 ON ... GOSUB i ON ... GOTO

Format: `ON` <izraz> `GOTO` <br linije>,<br linije>,<br linije> ...
`ON` <izraz> `GOSUB` <br linije>,<br linije>,<br linije> ...

Namena: Skok na <br linije> u zavisnosti od rezultata <izraza>. Višestruko grananje ili razgranati poziv potprograma.

Objašnjenje / Primedbe

Za ON...GOTO celobrojna vrednost <izraza> je redni broj celobrojne konstante <br linije> čija vrednost je broj programske linije na koju se skače.

Kod ON...GOSUB brojevi linija označavaju početak potprograma.

U slučaju da je vrednost <izraz> 0 ili veća od broja navedenih <br.linija>, BASIC će preći na izvršavanje sledeće instrukcije (ovo je korisno kada lista ne može stati u jednu programsku liniju).

Ako je vrednost <izraza> manja od 0 ili veća od 255, ispisuje se greška `Illegal function call` (nedozvoljen poziv funkcije).

primer:

```
100 ON A-1 GOTO 150,300,470,250
```

Za vrednosti izraza A-1 1, 2, 3 ili 4 skače se na linije 150, 300, 470 ili 250.

4.47 OPEN

Format: OPEN <način>,[#]<broj datoteke>,<ime datoteke>[,<duž.zapisa>]
Namena: Otvaranje datoteke za rad.

Objašnjenje / Primedbe

Datoteka, pre bilo kakvog rada sa njom, mora biti otvorena. Otvaranjem datoteke rezerviše se bafer za rad sa datotekom.

<način> je znakovni izraz koji odredjuje način rada sa datotekom. Postoje tri vrste rada sa datotekama u BASIC-u:

"O" - upisivanje u sekvencijalnu datoteku (*Output*)

"I" - čitanje iz sekvencijalne datoteke (*Input*)

"R" - čitanje / pisanje u datoteku sa slučajnim pristupom (*Random*)

<broj datoteke> je broj pod kojim se datoteka otvara. Svaki kasniji pristup datotekama zasniva se na ovom broju (vidi GET, PUT, INPUT#, LINE INPUT#, PRINT#, PRINT# USING, CLOSE ...). Broj datoteke mora biti u opsegu 1 do 15 i ne menja se sve dok se datoteka ne zatvori.

<ime datoteke> je znakovni izraz koji označava ime datoteke. Ime datoteke mora biti u skladu sa pravilima davanja imena datoteka u operativnom sistemu: 8 znakova, tačka i nastavak od 3 znaka.

<duž. zapisa> je celobrojni izraz koji, ako je naveden, postavlja dužinu zapisa na disku za datoteke sa slučajnim pristupom, a ako nije, dužina zapisa je 128 bajtova.

NAPOMENA: Datoteka može biti otvorena na više mesta (pod raznim brojevima) samo za sekvencijalno čitanje ili slučajni pristup. Datoteka može biti otvorena za sekvencijalni upis samo pod jednim brojem (u isto vreme).

primer:

```
10 OPEN "I",2,"OCENE"
20 OPEN "R",5,"PREDMETI"
-
-
```

4.48 OUT

Format: OUT <i>,<j>
gde su <i> i <j> celobrojni izrazi

Namena: Slanje jednog bajta na U/I (ulazno/izlazni) port mikroprocesora.

Objašnjenje / Primedbe

Celobrojni izraz <i> može imati vrednost od 0 do 65535, a <j> može biti u opsegu [0,255]. <i> je broj U/I porta na koji se šalje, a <j> podatak koji se šalje na port.

Grafička memorija se nalazi na U/I adresama od 32768 do 65535. U U/I prostoru (prvih 256 adresa, od 0 do 255) postoje još i registri za kontrolu U/I uredjaja računara.

Ovu instrukciju koristite samo ako ste sigurni šta radite, inače može dovesti do poremećaja rada periferih uredjaja i/ili pada sistema.

primer:

```
  5 'popunjavanje ekrana maksimalnim intenzitetom
 10 FOR I=32768 to 65535
 20 OUT I,&HFF
 30 NEXT
 RUN
```

4.49 PAINT

Format: PAINT <x koord.>,<y koord.>[,<aps/rel>[,<intenzitet>]]
Namena: Promena boje površine.

Objašnjenje / Primedbe

<x koord.> i <y koord.> su grafičke koordinate ili priraštaji u pravcu apscise i ordinate, računati od grafičkog kursora. Opseg vrednosti aritmetičkog izraza za <x koord.> i <y koord.> je [-4096,4095]. Na ekranu je vidljiv prvi kvadrant po X osi [0,511] i po Y osi [0,255]. Površina se popunjava počevši od tačke zadate ovim koordinatama.

<aps/rel> je aritmetički izraz čija vrednost odredjuje način zadavanja koordinata. Ako se izostavi ili ima vrednost nula, koordinate su zadate apsolutno. Za vrednost različitu od nule <x koord.> i <y koord.> su priraštaji u pravcu apscise i ordinate, računati od pozicije grafičkog kursora.

<intenzitet> može imati vrednosti iz opsega [0,3], a ako se izostavi, podrazumeva se prethodno zadata vrednost. Intenzitet popunjene površine je proporcionalan vrednosti <intenzitet>. Površina se popunjava <intenzitetom> do granica čiji je intenzitet različit od intenziteta površine koja se popunjava.

PAINT instrukcija ne postavlja novu poziciju grafičkog kursora.

PAINT neće ispuniti površinu u sledećim slučajevima:

- Ako je zadata tačka van vidljivog ekrana (ili van limita za koordinate)
- Ako je intenzitet tačke od koje počinje popunjavanje jednak zadatom intenzitetu

primer:

```
10 CLS : MOVE 0,0,0,3
20 ELIPSE 128 : MOVE 100,100,1,1
30 ELIPSE 100,100 : PAINT 0,0,1,2
40 PAINT 0,0,1,0 : GOTO 10
```

4.50 PLOT

Format: PLOT <x koord.>,<y koord.>[,<aps/rel>[,<intenzitet>]]
Namena: Crtanje tačke i postavljanje pozicije grafičkog kursora.

Objašnjenje / Primedbe

<x koord.> i <y koord.> su grafičke koordinate ili priraštaji u pravcu apscise i ordinate, računati od pozicije grafičkog kursora. Opseg vrednosti aritmetičkog izraza za <x koord.>

i <y koord.> je [-4096,4095]. Na ekranu je vidljiv prvi kvadrant po X osi [0,511] i po Y osi [0,255].

<aps/rel> je aritmetički izraz čija vrednost odredjuje način zadavanja koordinata. Ako se izostavi ili ima vrednost nula, koordinate su zadate apsolutno. Za vrednost različitu od nule <x koord.> i <y koord.> su priraštaji u pravcu apscise i ordinate, računati od pozicije grafičkog kursora.

<intenzitet> može imati vrednosti iz opsega [0.3], a ako se izostavi, podrazumeva se prethodno zadata vrednost. Intenzitet tačke je proporcionalan vrednosti <intenziteta>.

PLOT instrukcija postavlja novu poziciju grafičkog kursora na koordinate nacrtane tačke.

primer:

```
10 CLS
20 FOR I=1 TO 1000
30 PLOT RND*512,RND*256,0,RND*4
50 NEXT
```

4.51 POKE

Format: POKE <i>,<j>
gde su <i> i <j> celobrojni izrazi

Namena: Upis bajta na zadatu memorijsku lokaciju.

Objašnjenje / Primedbe

<i> je adresa na koju će podatak <j> biti upisan. Opseg vrednosti za <i> je [0.65535], a za <j> [0,255]. POKE je namenjen za upis mašinskog potprograma, prenos vrednosti mašinskim potprogramima i dr. (za čitanje vidi funkciju PEEK).

POKE koristite samo ako ste sigurni u to šta radite. POKE može dovesti do nepravilnog funkcionisanja sistema (pa i pada sistema).

primer:

```
10 POKE &HE906,&HC9  'iskljuci kursor
20 PRINT "KURSOR ISKLJUCEN"
30 POKE &HE906,&HC3  'ukljuci kursor
```

4.52 PRINT

Format: PRINT [<lista izraza>]
Namena: Ispisivanje podataka na ekran.

Objašnjenje / Primedbe

Ako <lista izraza> nije navedena, ispisuje se prazna linija. Izrazi mogu biti aritmetički i/ili znakovni (znakovne konstante moraju biti zatvorene u znake navoda).

Pozicije pri štampanju:

Pozicija svake vrednosti izraza iz liste odredjena je znakom razdvajanja izmedju izraza.

Ako je za razdvajanje izraza upotrebljen zarez (,), vrednosti izraza se ispisuju u zonama dužine 14 znakovnih mesta.

Kada su izrazi razdvojeni znakom tačka-zarez (;), vrednosti se ispisuju bez razmaka. Istu funkciju ima jedan ili više blanko znakova.

Ako je zarez (,) ili tačka-zarez (;) na kraju liste izraza, vrednosti u sledećoj PRINT instrukciji se ispisuju u istom redu; u protivnom ispis vrednosti se nastavlja u sledećoj liniji ekrana.

Umesto znaka plus (+), za pozitivne brojeve ispisuje se jedan blanko znak; znak minus (-) za negativne se uvek ispisuje. Iza brojeva dodaje se jedan blanko znak.

Vrednosti jednostruke preciznosti sa 7 ili manje cifara ispisuju se u normalnom zapisu (1E-6 se ispisuje kao .000001), a sa više od 7 u pokretnom zarezu (1E-8 se ispisuje kao 1E-08). Za vrednosti duple preciznosti važi analogija (1D-15 biće .000000000000001 a 1D-16 biće 1D-16).

PRINT se može zameniti znakom pitanja (?). Znak pitanja u programskoj liniji na mestu instrukcije automatski se zamenjuje instrukcijom PRINT.

primer:

```
10 X=5
20 PRINT X+5, X-5, X*(-5), X^5
30 END
RUN
 10       0       -25       3125
Ok
```

Primer za izraze odvojene zarezima.

```
10 INPUT X
20 PRINT X;"NA KVADRAT JE";X^2 "I";
30 PRINT X "NA TRECI STEPEN JE" X^3
40 PRINT
50 GOTO 10
RUN
? 9
 9 NA KVADRAT JE 81 I 9 NA TRECI STEPEN JE 729

? 21
 21 NA KVADRAT JE 441 I 21 NA TRECI STEPEN JE 9261
```

Primeri
za izraze odvojene tačka-zarezima (lin. 20),
za tačka–zarez na kraju reda (lin. 20),
za izraze odvojene blanko znacima (lin. 30)
i ispis prazne linije (lin. 40).

```
10 FOR X=1 TO 5
20 J=J+5
30 K=K+10
40 ?J;K;
50 NEXT X
RUN
 5 10 10 20 15 30 20 40 25 50
```

Pozitivni brojevi imaju ispred i iza jedan blanko znak.

```
10?A
LIST
10 PRINT A
Ok
```

4.53 PRINT USING

Format: PRINT USING <format>;<lista izraza>
Namena: Ispisivanje numeričkih i znakovnih vrednosti koristeći (*using*) zadati format ispisa.

Objašnjenje / Primedbe

<lista izraza> je sastavljena od znakovnih i/ili numeričkih izraza odvojenih tačka-zarezom (;).

<format> je znakovna konstanta ili promenljiva koja sadrži specijalne znakove za formatiranje ispisa. Ovi specijalni znaci za formatiranje ispisa odredjuju polja i format znakova i brojeva koji se ispisuju.

Znakovna polja:

Kada se PRINT USING upotrebljava za ispis vrednosti znakovnih izraza iz <liste izraza> jedan od sledeća tri specijalna formatirajuća znaka se upotrebljava u <formatu>:

"!" (uzvičnik) - Nalaže ispisivanje samo prvog znaka znakovnog izraza. "\n blankova\" - Ograničava dužinu polja na 2+n znakova.

Ako su obrnute kose crte otkucane bez blanko znaka (\\), biće ispisana 2 znaka; sa jednim blanko znakom izmedju kosih crta (\ \) biće ispisana 3 znaka znakovnog izraza itd. Višak znakova se ignoriše. Ukoliko je polje duže, znakovi će se ispisati uz levu ivicu, a ostatak će biti popunjen blanko znacima. "&" - Odredjuje promenljivu dužinu polja. Kada je format polja odredjen sa "&", cela grupa znakova (celokupna dužina) biće ispisana.

primer:

```
10 A$="ABCD":B$="XYZ"
20 PRINT USING "!";A$;B$
30 PRINT USING "\ \";A$;B$
40 PRINT USING "\   \";A$;B$;"!!"
50 PRINT USING "!";A$;
60 PRINT USING "&";B$
RUN
AX
ABCDXYZ
ABCD XYZ !!
AXYZ
```

Numerička polja:

Kada se sa PRINT USING ispisuju brojevi, sledeći specijalni znaci se koriste za formatiranje:

"#" - se upotrebljava da odredi svako pojedinačnò cifarsko mesto u broju. Ako je broj kraći od broja definisanih mesta, biće desno poravnat a višak mesta se popunjava blanko znacima.

"." - decimalna tačka može biti umetnuta bilo gde izmedju # znaka da odredi mesto decimalne tačke. Ukoliko nema cifara levo od decimalne tačke (nema celog dela) biće ispisana 0. Decimalni deo broja zaokružuje se na zadati broj decimala (ako ih ima više od zadatog formata).

Na primer:

```
PRINT USING "##.##";.56
 0.56

PRINT USING "###.##";987.654
987.65

PRINT USING "##.##  ";10.2;5.3;66.789;.234
10.20  5.30  66.79  0.23
```

(u ovom primeru tri blanko znaka upotrebljena su za razdvajanje brojeva)

"+" - plus znak na početku ili kraju formata izričito naredjuje da se ispiše znak (+ ili -) ispred ili iza broja.

"-" - minus napisan iza formata naredjuje da se minus piše iza broja (ako je broj negativan).

Na primer:

```
PRINT USING "+##.##  ";-68.95,2.4,55.6,-.9
-68.95  +2.40  +55.60  -0.90

PRINT USING "##.##-  ";-68.95,22.449,-7.01
68.95-   22.45    7.01-
```

"**" - dve zvezdice odredjuju da se, ako je broj manji od definisanog polja, višak mesta na početku popunjava zvezdicama. Takodje, dve zvezdice definišu 2 cifarska mesta. Zvezdice se navode na početku formata.

Na primer:

```
PRINT USING "*****.#  ";12.39,-0.9,99765.07
***12.4  ***-0.9  99765.1
```

"$$" - dva znaka dolara odredjuju da se znak $ piše uz levu ivicu broja. Znaci $$ definišu 2 cifarska mesta. Eksponencijalni format (vidi niže) ne može se koristiti uz $$. Negativne vrednosti se takodje ne mogu koristiti uz $$ osim kada je znak minus (-) zadat sa desne strane broja. Ovaj format namenjen je za ispis dolarskih iznosa.

Na primer:

```
PRINT USING "$$###.###";456.78
 $456.78
```

"**$" - na početku formata znači isto što i prethodna dva zajedno. Vodeći blanko znaci biće zamenjeni zvezdicama i dolarom ispred broja. **$ definiše 3 cifarska mesta od kojih će jedno biti popunjeno znakom za dolar.

Na primer:

```
PRINT USING "**$##.##";2.45
***$2.45
```

"," - zarez sa leve strane decimalne tačke odvaja svaku treću cifru celog dela broja zarezom (odvaja za red veličine hiljadu). Zarez na kraju formata biće ispisan. Zarez definiše jedno cifarsko mesto. Zarez nema efekta u eksponencijalnom formatu.

Na primer:

```
PRINT USING "####,.##";1234.5
1,234.50
PRINT USING "####.##,";1234.5
1234.50,
```

"^^^^" - četiri kapice, eksponencijalni format. Ovim znacima na kraju formata odredjuju se četiri mesta (za E+xx). Cifarska mesta mogu (a ne moraju) biti definisana u formatu. Takodje, može se uz eksponencijalni format dodati + ili - na početku ili na kraju za zapis znaka broja.

Na primer:

```
PRINT USING "##.##^^^^";234.56
 2.35E+02

PRINT USING ".####^^^^-";666666
 .66667E+06

PRINT USING "+.##^^^^";123
+.12E+03
```

"_" - podvlaka označava ispis jednog znakovnog simbola. Sama podvlaka može se ispisati kombinacijom dveju podvlaka (__).

Na primer:

```
PRINT USING "_!##.##__";12.34
!12.34_
```

% - postotak prijavljuje grešku prekoračenja dužine formata. On se javlja kada je definisani broj cifarskih mesta manji od potrebnog.

Primeri:

```
PRINT USING "##.##";111.22
%111.22

PRINT USING ".##";.999
%1.00
```

Ukoliko se u formatu definišu više od 24 cifarska mesta biće prijavljena greška `Illegal function call` (Nedozvoljen funkcijski poziv).

4.54 PRINT# i PRINT# USING

Format: PRINT#<broj datoteke>,[USING<format>;]<lista izraza>
Namena: Ispis vrednosti (podataka) u sekvencijalnu datoteku.

Objašnjenje / Primedbe

<broj datoteke> je broj pod kojim je datoteka otvorena za upis (sa OPEN).

<format> je grupa specijalnih znakova za formatiranje opisana u PRINT USING.

<lista izraza> sastoji se iz znakovnih i/ili numeričkih izraza (vidi PRINT) čije će vrednosti biti upisane u datoteku.

PRINT# ne sabija podatke kada ih upisuje u datoteku, već se ponaša identično instrukciji PRINT.

Kada se upisuju znakovi (grupe znakova) na disk radi pouzdanog čitanja, potrebno je definisati znak za razdvajanje (na primer zarez).

Neka je
A$="BASIC" i B$="UPUTSTVO".

```
PRINT#1,A$;B$
```

će prouzrokovati zapis BASICUPUTSTVO. Pošto nema znaka za razdvajanje, po izvršenju instrukcije

```
INPUT#1 A$,B$
```

A$ će biti BASICUPUTSTVO. Korišćenjem

```
PRINT#1,A$;",";B$
```

zapisuje se BASIC,UPUTSTVO a instrukcija

```
INPUT#1 A$,B$
```

će ispravno pročitati obe reči.

Ako grupe znakova već sadrže zarez ili blanko znake kao sastavni deo, upišite znake navoda ispred i iza grupe znakova pomoću funkcije CHR$(34) (34 je ASCII kod za znake navoda).

```
10 A$="BASIC, INTERPRETATOR" : B$="  UPUTSTVO"
20 PRINT#1,A$;B$
```

će na disk upisati BASIC, INTERPRETATOR UPUTSTVO, a instrukcija

```
30 INPUT#1,A$,B$
```

će učitati u A$ "BASIC", a u B$ "INTERPRETATOR UPUTSTVO". Ispravno je

```
20 PRINT#1,CHR$(34);A$;CHR$(34);CHR$(34);B$;CHR$(34)
```

što će na disk upisati "BASIC, INTERPRETATOR"" UPUTSTVO", a

```
INPUT#1,A$,B$
```

će učitati u A$ "BASIC, INTERPRETATOR", a u B$ " UPUTSTVO".

Uz PRINT# se može koristiti USING za formatizovan upis u datoteku.

primer:

```
PRINT#1,USING"$$###.##,";L;K;J.
```

Za više detalja o formatiranju vidi PRINT USING i WRITE#.

4.55 PUT

Format: PUT [#]<broj datoteke>[,<broj zapisa>]

Namena: Upis zapisa iz bafera u datoteku sa slučajnim pristupom.

Objašnjenje / Primedbe

<broj datoteke> je broj pod kojim je datoteka otvorena (sa OPEN).

Ako je <broj zapisa>, izostavljen sadržaj bafera biće upisan sekvencijalno u sledeći zapis na disku, a ako je naveden, biće upisan na mesto rednog <broja zapisa>. <broj zapisa> kreće se u opsegu od 1 do 32767.

Svaki pokušaj pisanja preko kraja bafera prijaviće grešku `Field overflow` (polje je prekoračeno).

primer:

Vidi **Dodatak A**

4.56 RANDOMIZE

Format: RANDOMIZE [<celobrojni izraz>]
Namena: Inicijalizacija generatora slučajnih brojeva.

Objašnjenje / Primedbe

BASIC poseduje generator pseudo slučajnih brojeva. On generiše slučajne brojeve iz unapred odredjenog niza sa 65536 članova. Inicijalizacijom se bira početni član niza.

Ako je <celobrojni izraz>, izostavljen BASIC će upitati za indeks početnog člana niza postavljanjem sledećeg pitanja:

```
Random Number Seed (-32768 to 32767)?
```

pre izvršavanja RANDOMIZE instrukcije.

Ukoliko se ne navede RANDOMIZE, svaki put kada počne izvršavanje programa RND funkcija vraća istu sekvencu brojeva. Stavljanjem RANDOMIZE na početak moguće je menjati sekvencu slučajnih brojeva.

primer:

```
 10 RANDOMIZE
 20 FOR I=1 TO 5
 30 PRINT RND;
 40 NEXT I
 RUN
 Random number seed (-32768 to 32767)? 3
  .88598 .484668 .586328 .119426 .709225
 Ok
 RUN
 Random number seed (-32768 to 32767)? 4
  .803506 .162462 .929364 .292443 .322921
 Ok
 RUN
 Random number seed (-32768 to 32767)? 3
  .88598 .484668 .586328 .119426 .709225
 Ok
```

4.57 READ

Format: READ <lista promenljivih>

Namena: Čitanje podataka iz DATA liste i dodeljivanje tih vrednosti promenljivama.

Objašnjenje / Primedbe

READ se uvek upotrebljava zajedno sa DATA instrukcijama. Vrednosti iz DATA liste se dodeljuju promenljivama u <listi promenljivih> redom. Promenljive u <listi promenljivih> mogu biti numeričke ili znakovne i moraju redom odgovarati tipu podataka u DATA listi. Ukoliko to nije slučaj, ispisaće se greška **Syntax error** (Pravopisna greška).

Jedna READ instrukcija može čitati jednu ili više vrednosti iz DATA liste, zavisno od toga koliko je promenljivih navedeno u <listi promenljivih>. Ako je broj čitanja veći od broja podataka u DATA listi, ispisuje se greška **Out of data** (nema više podataka).

Čitanje DATA liste od početka ili od zadate linije postiže se instrukcijom RESTORE.

primer:

```
10 FOR I=1 TO 10
20 READ A(I)
30 NEXT I
40 DATA 17,21,3.02,24,99
50 DATA 5,12,19,11.1,.2
-
-
```

ovo će dodeliti promenljivama A(1) 17, A(2) 21, A(3) 3.02 itd.

```
10 PRINT "IME","TELEFON"
15 PRINT
20 FOR I=1 TO 3
30 READ I$,T$
40 PRINT I$,T$
50 NEXT I
60 DATA "ZORAN,",217395
70 DATA "JOVAN,","764-222"
RUN
IME     TELEFON

ZORAN,     217395
JOVAN,     764-222
Out of data in line 30
Ok
```

4.58 REM

Format: REM <tekst primedbe>

Namena: Komentarisanje programa ili odvajanje funkcionalnih celina programa objašnjenjima.

Objašnjenje / Primedbe

REM je instrukcija koja se ne izvršava. Sve što se nalazi iza REM instrukcije (do kraja programske linije) čuva se kao što je otkucano i smatra se komentarom. Ako program pri izvršavanju naidje na REM, biće automatski upućen na sledeću programsku liniju.

REM se može zameniti jednostrukim znakom navoda (').

UPOZORENJE: Ne koristiti REM u DATA linijama jer se može desiti da se pročita kao podatak.

primer:

```
10 REM --- OVO JE KOMENTAR ---
20 ' OVO JE TAKODJE KOMENTAR
30 PRINT A 'OVDE NE MORAJU STAJATI :
40 PRINT B :REM KAO U OVOM SLUCAJU
50 PRINT C ':PRINT AA PAZI! PRINT AA JE U KOMENTARU!!!
```

4.59 RENUM

Format: RENUM [[<novi broj>][,[<stari broj>][,<uvećanje>]]]
Namena: Renumeracija programskih linija.

Objašnjenje / Primedbe

<novi broj> je prvi broj programske linije koji se koristi u novoj sekvenci označavanja programskih linija. Ukoliko nije naveden, podrazumeva se broj 10.

<stari broj> je broj linije od koje počinje renumeracija. Ako je izostavljen, podrazumeva se prva linija programa.

<uvećanje> je broj za koji će se uvećavati brojevi programskih linija u novoj sekvenci. U slučaju da je izostavljen, podrazumeva se 10.

RENUM pored linijskih brojeva menja brojeve koji slede iza instrukcija: GOTO, GOSUB, THEN, ON...GOTO, ON...GOSUB i ERL tako da referišu na novonumerisane linije. Ukoliko se pri renumeraciji pojavi neka od gore navedenih instrukcija praćenih brojem linije koja ne postoji ispisaće se greška `Undefined line xxx in yyy` (nedefinisana linija xxx u liniji yyy).

RENUM ne može da promeni redosled linija (na primer: `RENUM 15,30` kada postoje linije 10, 20 i 30) ili da renumeriše brojeve linija veće od 65529. U oba slučaja ispisuje se greška `Illegal function call` (Nedozvoljen poziv funkcije).

primer:

```
RENUM
```

renumerisaće ceo program počevši od prve linije kojoj će dati broj 10 i nastaviće sa uvećanjem 10 (10, 20, 30, 40....).

```
RENUM 300,,50
```

renumerisaće ceo program počevši od prve linije kojoj će dodeliti broj 300 i nastaviti sa uvećanjem 50 (300, 350, 400, 450...).

```
RENUM 1000,900,20
```

renumerisaće linije od 900 naviše (do kraja programa) tako da im nove vrednosti počinju od 1000, i razlikuju se za 20.

4.60 RESET

Format: RESET
Namena: Zatvaranje svih otvorenih datoteka i snimanje direktorijuma na disk pre no što se izvadi disketa.

Objašnjenje / Primedbe

Uvek izvršite RESET pre no što promenite disketu u disk jedinici. U suprotnom datoteke koje su otvorene mogu izgubiti sve ili deo podataka. Može se desiti da se podaci upisu na novu disketu u datoteku sa istim imenom (što može biti frustrirajuće) ili će podaci biti pravilno upisani na disk, a to neće biti snimljeno u direktorijum.

primer:

```
Ok
RESET
Ok
```

sada možete zameniti disketu u disketnoj jedinici !

4.61 RESTORE

Format: RESTORE [<br.linije>]
Namena: Inicijalizacija DATA liste za čitanje.

Objašnjenje / Primedbe

Posle izvršavanja instrukcije RESTORE bez parametra, prvi READ koji se bude izvršio pročitaće podatak iz prve DATA instrukcije u programu.

U slučaju da je <br. linije> naveden, READ će čitati podatke iz te linije ukoliko u njoj postoji DATA instrukcija, odnosno iz prve naredne linije koja sadrži DATA instrukciju.

primer:

```
10 READ A,B,C
20 PRINT A;B;C
30 RESTORE
40 READ A,B,C
50 PRINT A;B;C
60 DATA 1,2,3
RUN
 1 2 3
 1 2 3
Ok
```

4.62 RESUME

Format: `RESUME`
`RESUME 0`
`RESUME NEXT`
`RESUME <br.linije>`

Namena: Nastavak rada programa po obradi greške.

Objašnjenje / Primedbe

RESUME ili RESUME 0 ponavlja instrukciju koja je prouzrokovala grešku.

RESUME NEXT nastavlja izvršavanje prve sledeće instrukcije iza one koja je izazvala grešku.

RESUME <br. linije> nastavlja izvodjenje programa od programske linije zadate <brojem linije>.

Ako program u svom radu naidje na RESUME, a nije nastala greška (u programu), prijaviće se `RESUME without error` (RESUME bez pojave greške).

primer:

```
10 ON ERROR GOTO 900
 -
 -
900 IF (ERR=230) AND (ERL=90) THEN PRINT "Pokusaj ponovo...":RESUME 90
```

4.63 RUN

Format: `RUN [<broj linije>]`
`RUN <ime programa>[,R]`

Namena: Izvršavanje programa u memoriji uz eventualno prethodno učitavanje sa diska.

Objašnjenje / Primedbe

Izvršavanje programa počinje od <broja linije>. U slučaju da <broj linije> nije naveden, program se izvršava od početka.

Ukoliko je navedeno <ime programa> učitava se program sa diska i izvršava.

RUN zatvara sve otvorene datoteke. Opcija R ostavlja datoteke otvorenim.

primer:

`RUN`	izvršava program od početka
`RUN 5`	izvršava program od linije 5
`RUN "prog1"`	učitava PROG1.BAS sa diska i izvršava ga od početka
`RUN "prog1",R`	isto kao i prethodni primer, ali ostavlja sve datoteke otvorene

4.64 SAVE

Format: `SAVE` <ime programa>`[,A/,P]`
Namena: Upisivanje programa iz memorije u datoteku.

Objašnjenje / Primedbe

<ime programa> je znakovni izraz. Ako datoteka sa istim imenom već postoji na disku, biće izbrisana, a na njeno mesto biće upisan program iz memorije.

"A" opcija omogućava da se program upiše u ASCII formatu (kao tekst datoteka) što je uslov za neke operacije (vidi MERGE). Takodje, ovaj format se može koristiti da se programi pisani u BASICu proslede nekom tekst editoru ili nekom drugom programu radi dalje obrade.

"P" opcija (protected) snima program u zaštićenom kodovanom binarnom obliku. Kada se kasnije pozove program (sa LOAD ili RUN) moći će samo da se izvršava ali ne i da se lista i/ili ispravlja. Čuvajte originalne programe (u tekst obliku) ako koristite ovu opciju.

primer:

```
SAVE "PROG1"
SAVE "PROG2",A
SAVE "PROG3",P
```

4.65 SOUND

Format: SOUND <visina>,<trajanje>
Namena: Generisanje zvuka.

Objašnjenje / Primedbe

<visina> je bilo koji aritmetički izraz čiji je rezultat u opsegu od 0 do 63. (veći brojevi se automatski svode na ovaj opseg po modulu 64).

Vrednosti imaju sledeće značenje:

0 - zvono 1
1 - pauza
2 - najniži ton (boja tona 1)
-
- tonovi
-
30 - najviši ton (boja tona 1)
31 - sirena
32 - pauza
33 - pauza
34 - najniži ton (boja tona 2)
-
- tonovi
-
62 - najviši ton (boja tona 2)
63 - zvono 2

<trajanje> može imati vrednost u opsegu od 0 do 65535 i zadato je u milisekundama.

4.66 STOP

Format: STOP
Namena: Prekid izvršavanja programa i povratak u komandni način rada.

Objašnjenje / Primedbe

STOP instrukcija može se nalaziti bilo gde u programu i ima funkciju da prekida njegovo izvršavanje. Na ekranu ispisuje **Break in line** nnnnn (prekid na liniji nnnnn) a BASIC prelazi na komandni način rada. STOP ne zatvara datoteke. Izvršavanje programa se može nastaviti komandom CONT. STOP i CONT se najčešće koriste pri testiranju rada programa.

primer:

```
10 INPUT A,B,C
20 K=A^2*5.3:L=B^3 / .26
30 STOP
40 M=C*K+100:PRINT M
RUN
? 1,2,3
Break in 30
Ok
PRINT L
 30.7692
Ok
CONT
 115.9
Ok
```

4.67 SWAP

Format: SWAP <promenljiva>, <promenljiva>
Namena: Medjusobna zamena vrednosti dve promenljive.

Objašnjenje / Primedbe

SWAP zamenjuje vrednosti dve promenljive tj. prvoj promenljivoj dodeljuje vrednost druge, a drugoj prve.

Obe promenljive moraju biti istog tipa; u protivom se ispisuje greška **Type mismatch** (Neslaganje tipova).

primer:

vidi WHILE...WEND primer

4.68 SYSTEM

Format: SYSTEM
Namena: Kraj rada u BASIC-u i povratak u operativni sistem

4.69 TEXT

Format: TEXT <zi>,<x koord.>,<y koord.>[,<aps/rel>[,<intenzitet>]]
gde je <zi> znakovni izraz
Namena: Ispisivanje teksta na proizvoljnoj koordinati na ekranu.

Objašnjenje / Primedbe

<zi> je znakovni izraz čija će vrednost biti ispisana (znakovna konstanata, promenljiva, elemenat niza itd.)

<x koord.> i <y koord.> su grafičke koordinate ili priraštaji u pravcu apscise i ordinate, računati od pozicije grafičkog kursora. Opseg vrednosti aritmetičkog izraza za <x koord.> i <y koord.> je [-4096,4095]. Na ekranu je vidljiv prvi kvadrant po X osi [0,511] i po Y osi [0,255].

<aps/rel> je aritmetički izraz čija vrednost odredjuje način zadavanja koordinata. Ako se izostavi ili ima vrednost nula, koordinate su zadate apsolutno. Za vrednost različitu od nule <x koord.> i <y koord.> su priraštaji u pravcu apscise i ordinate, računati od pozicije grafičkog kursora.

<intenzitet> može imati vrednosti iz opsega [0,3], a ako se izostavi, podrazumeva se prethodno zadata vrednost. Intenzitet teksta je proporcionalan vrednosti <intenziteta>.

TEXT instrukcija ne menja poziciju grafičkog kursora.

TEXT instrukcija ignoriše ispisivanje teksta koji ide van vidljivog opsega.

Ispis instrukcijom TEXT ne menja intenzitet pozadine, što je pogodno za označavanje crteža.

Veličina znakova je 6 x 10 tačaka.

primer:

```
10 CLS: MOVE 0,0,0,3
20 TEXT "Osenceni tekst.",100,100,0,1
30 TEXT "Osenceni tekst.",99,101,0,3
40 DRAW 511,0,1 : DRAW 0,255,1
50 DRAW -511,0,1 : DRAW 0,-255,1
```

4.70 TRON/TROFF

Format: TRON
TROFF

Namena: Uključuje / isključuje ispis brojeva programskih linija koje se izvršavaju.

Objašnjenje / Primedbe

TRON i TROFF mogu biti zadati kako u komandnom načinu rada, tako i u programu kao instrukcije.

Brojevi linija se prikazuju zatvoreni u srednje zagrade ([ ]). Komanda NEW isključuje ispis.

primer:

```
TRON
Ok
10 K=10
20 FOR J=1 TO 2
30 L=K+10
40 PRINT J;K;L
50 K=K+10
60 NEXT
70 END
RUN
[10][20][30][40] 1 10 20
[50][60][30][40] 2 20 30
[50][60][70]
```

4.71 WHILE...WEND

Format: `WHILE <izraz>`
`-`
`- [ <telo petlje> ]`
`-`
`WEND`

Namena: Izvršavanje serije instrukcija dok je <izraz> tačan (tj. različit od nule).

Objašnjenje / Primedbe

Po nailasku na WHILE, izračunava se <izraz>: ako je tačan (tj. različit od 0), izvršava se <telo petlje> i prelazi se (ponovo) na WHILE; ako nije tačan, izvršavanje se nastavlja od prve instrukcije iza WEND.

WHILE...WEND petlja (instrukcije u petlji) ne mora se izvršiti ni jednom (u zavisnosti od uslova). WHILE...WEND je takozvana petlja sa izlazom na vrhu.

WHILE...WEND petlje mogu biti jedna u drugoj. Broj petlji u petlji je ograničen samo raspoloživom memorijom. Svaki WEND mora imati svoj WHILE i obrnuto. Ako program naidje na WEND, a nije prošao kroz WHILE, javiće se greška `WEND without WHILE` (WEND bez WHILE), ako pak program prodje dva puta kroz isti WHILE, javiće se greška `WHILE without WEND` (WHILE bez WENDa).

primer:

```
90 'BUBBLE sort rutina za A$(1..J)
100 USLOV=1  ' 1 - niz je nesortiran, 0 - niz sortiran
110 WHILE USLOV
120 ::USLOV=0 ' ne vrsi nikakvu funkciju tu je zbog citljivosti
130 ::FOR I=1 TO J-1
140 ::::IF A$(I) > A$(I+1) THEN SWAP A$(I),A$(I+1):USLOV=1
150 ::NEXT I
160 WEND
```

4.72 WIDTH

Format: WIDTH [LPRINT]<celobrojni izraz>

Namena: Postavljanje širine ispisa na ekranu ili štampaču.

Objašnjenje / Primedbe

<celobrojni izraz> je nova vrednost širine i izražava se u znakovnim mestima (ekran ima 80 znakovnih mesta, štampači imaju širinu od 40 naviše znakovnih mesta u liniji).

<celobrojni izraz> u WIDTH može imati vrednosti od 15 do 255. Ako je <celobrojni izraz> 255, BASIC će smatrati liniju beskonačno dugačkom (neće automatski ispisivati CR LF sekvencu na kraju linije).

Ukoliko je LPRINT opcija izostavljena, širina se odnosi na ekran a ukoliko je uključena - na štampač.

primer:

```
10 PRINT "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
RUN
ABCDEFGHIJKLMNOPQRSTUVWXYZ
Ok
WIDTH 18
Ok
RUN
ABCDEFGHIJKLMNOPQR
STUVWXYZ
Ok
WIDTH LPRINT 80
Ok
```

4.73 WRITE

Format: WRITE [<lista izraza>]
Namena: Ispis podataka na ekran.

Objašnjenje / Primedbe

Ako je <lista izraza> izostavljena, biće ispisana prazna linija. Izrazi u listi mogu biti numerički i/ili znakovni i moraju biti razdvojeni zarezom (,).

Vrednosti se ispisuju jedna za drugom i iza svake se ispisuje zarez. Znakovi (grupe znakova) se ispisuju zatvoreni u znake navoda (").

WRITE ispisuje numeričke vrednosti po svim pravilima iz PRINT instrukcije (vidi PRINT).

primer:

```
10 A=99:B$="DELTA"
20 WRITE B$,A
RUN
"DELTA",99
Ok
```

4.74 WRITE#

Format: WRITE#<broj datoteke>,<lista izraza>
Namena: Ispisivanje podataka u sekvencijalnu datoteku.

Objašnjenje / Primedbe

<broj datoteke> je broj pod kojim je datoteka otvorena (sa OPEN). Datoteka mora biti otvorena OPEN opcijom "O".

Za <listu izraza> važe ista pravila kao i za WRITE.

Razlika izmedju WRITE# i PRINT# je u tome da se ne moraju eksplicitno navoditi znaci za razdvajanje kada su u pitanju mešoviti podaci (vidi PRINT#).

WRITE# posle ispisa vrednosti u listi (na kraju linije) upisuje CR - LF sekvencu.

primer:

Neka je I$="DRAGAN". a T$="323480"; onda

```
WRITE#1,I$,T$
```

upisuje na disk sledeće

```
"DRAGAN","323480"
```

```
INPUT\#1,I$,T$
```

će pročitati DRAGAN u I$. a 323480 u T$.

5 BASIC funkcije

Funkcije BASICa su opisane u ovom poglavlju. Ove funkcije se mogu pozivati u bilo kom delu programa bez dodatnog definisanja. Funkcije se nalaze u izrazima sa desne strane znaka jednakosti ili u naredbama u kojima se koriste aritmetički ili znakovni izrazi.

Argumenti funkcija su uvek zatvoreni u zagrade. U formatima datim u ovom poglavlju upotrebljavaju se, pored opisa datih u poglavlju 1.1, sledeće skraćenice:

X ili Y predstavlja bilo koji numerički izraz ili funkciju;

I ili J predstavlja bilo koji celobrojni izraz ili funkciju;

X$ ili Y$ predstavlja bilo koji znakovni izraz ili funkciju;

Ako je isporučena vrednost sa decimalama tamo gde se očekuje celobrojna vrednost, razlomljeni deo biće zaokružen na bliži ceo broj i to će se upotrebiti kao ulazna vrednost za funkciju.

BASIC interpretator za rezultat aritmetičkih funkcija može vratiti samo vrednosti jednostruke preciznosti (dvostruka nije realizovana).

Sve BASIC funkcije opisane su na sledeći način:

Format: (prikazuje način zadavanja funkcije)

Funkcija: (objašnjava funkciju)

Primeri: (navode se jednostavni primeri upotrebe funkcije)

5.1 ABS

Format: `ABS(X)`
Funkcija: Vraća apsolutnu vrednost izraza X.

primer:

```
PRINT ABS(2*(-2))
 4
Ok
```

5.2 ASC

Format: `ASC(X$)`
Funkcija: Vraća numeričku vrednost, ASCII kod, prvog znaka iz rezultata znakovnog izraza X$. Ako je rezultat izraza X$ NULL znak (znak sa ASCII kodom 0), ispisuje se **`Illegal function call`** (Nedozvoljen poziv funkcije).

primer:

```
10 X$ = "Torba"
20 PRINT ASC(X$)
RUN
 84
Ok
```

vidi CHR$ funkciju.

5.3 ATN

Format: `ATN(X)`
Funkcija: Vraća vrednost funkcije ARKUS TANGENS izraza X zadatog u radijanima. Rezultat funkcije ATN je uvek u opsegu `-PI/2` do `PI/2`. Rezultat izraza X može biti bilo koji numerički tip, a rezultat ATN funkcije je uvek u jednostrukoj preciznosti.

primer:

```
10 INPUT X
20 PRINT ATN(X)
RUN
? 3
 1.24905
Ok
```

5.4 CDBL

Format: `CDBL(X)`
Funkcija: Pretvara vrednost izraza X u vrednost dvostruke preciznosti.

primer:

```
10 A=454.67
20 PRINT A;CDBL(A)
RUN
 454.67 454.6700134277344
Ok
```

5.5 CHR$

Format: `CHR$(I)`
Funkcija: Vraća znak čiji je ASCII kod vrednost izraza I. CHR$ se često upotrebljava za slanje nekog znaka koji se ne prikazuje na ekranu (štampaču), ali vrši neku funkciju.

primer:

```
LPRINT CHR$(27);"W1"    kod EPSON štampača uključuje široka slova
PRINT CHR$(27);"E"      briše ekran
```

Vidi ASC funkciju.

5.6 CINT

Format: `CINT(X)`
Funkcija: Vraća vrednost X zaokruženu na bliži ceo broj. Ako X nije u opsegu -32768 do 32767, ispisuje se greška `Overflow` (Prekoračenje).

primer:

```
PRINT CINT(45.67)
 46
Ok
```

Vidi INT i FIX funkcije.

5.7 COS

Format: `COS(X)`
Funkcija: Vraća kosinus ugla X zadatog u radijanima. Rezultat COS funkcije je u jednostrukoj preciznosti.

primer:

```
10 X=2*COS(.4)
20 PRINT X
RUN
 1.84212
Ok
```

5.8 CSNG

Format: `CSNG(X)`
Funkcija: Pretvara vrednost X u vrednost jednostruke preciznosti.

primer:

```
10 A#=975.4321#
20 PRINT A#;CSNG(A#)
RUN
 975.3421 975.342
Ok
```

Vidi CINT i CDBL funkcije.

5.9 CVI, CVS, CVD

Format: CVI(<grupa znakova dužine 2 bajta>)
CVS(<grupa znakova dužine 4 bajta>)
CVD(<grupa znakova dužine 8 bajta>)

Funkcija: Pretvaranje grupe znakova pročitanih iz datoteke sa slučajnim pristupom u numeričke vrednosti. CVI pretvara 2 bajta u celobrojnu vrednost, CVS 4 bajta u vrednost jednostruke i CVD 8 bajtova u vrednost dvostruke preciznosti.

primer:

```
-
-
100 FIELD #1,4 AS N$, 12 AS B$
110 GET #1
120 Y=CVS(N$)
-
-
```

Vidi MKI$, MKS$ i MKD$ funkcije.

5.10 EOF

Format: EOF(<broj datoteke>)

Funkcija: Vraća -1 (tačno) kada je dostignut kraj datoteke. Upotrebite EOF za testiranje kraja pri čitanju datoteke da izbegnete grešku **Input past end** (Čitanje posle kraja). EOF radi i sa datotekama sa slučajnim pristupom. GET posle kraja datoteke sa slučajnim pristupom prouzrokuje da EOF bude -1. EOF se može upotrebljavati za nalaženje dužine datoteke sa slučajnim pristupom ili za binarno pretraživanje istih.

primer:

```
10 OPEN "I",1,"PODACI"
20 C=0
30 IF EOF(1) THEN 100
40 INPUT #1,P(C)
50 C=C+1:GOTO 30
-
-
```

5.11 EXP

Format: EXP(X)
Funkcija: Vraća vrednost stepena osnove prirodnog logaritma (osnova e na stepen X). X mora biti manje ili jednako 87.3365. U suprotnom ispisuje se Overflow greška (prekoračenje), rezultat je mašinska beskonačnost, a izvršavanje programa se nastavlja.

primer:

```
10 X=5
20 PRINT EXP(X-1)
RUN
 54.5982
Ok
```

5.12 FIX

Format: FIX(X)
Funkcija: Vraća celobrojni deo bliži nuli. Funkcija FIX je ekvivalentna izrazu SGN(X)*INT(ABS(X)).

primer:

```
PRINT FIX(58.75)
 58
Ok

PRINT FIX(-58.75)
 -58
Ok
```

Vidi INT i CINT funkcije.

5.13 FRE

Format: FRE(0)
FRE("")

Funkcija: Rezultat FRE(0) funkcije je količina raspoložive memorije u bajtovima. FRE("") čisti memoriju pre davanja vrednosti slobodne memorije. Čišćenje (*garbage collection*) može potrajati i do 1 minut. BASIC inače ne zahteva čišćenje memorije sve dok se sva slobodna memorija ne utroši.

primer:

```
PRINT FRE(0)
 13222
Ok
```

5.14 HEX$

Format: HEX$(X)

Funkcija: Vraća znakovnu vrednost koja je heksadecimalni zapis vrednosti X. Ukoliko X nije celobrojna vrednost, vrši se zaokruživanje.

primer:

```
10 INPUT X
20 PRINT X " DECIMALNO JE " HEX$(X) " HEKSADECIMALNO."
RUN
? 64
 64 DECIMALNO JE 40 HEKSADECIMALNO.
Ok
```

Vidi OCT$ funkciju.

5.15 INKEY$

Format: INKEY$

Funkcija: Vraća znak koji je pritisnut na tastaturi. Ukoliko ništa nije otkucano, vraća znak NULL (znak sa ASCII kodom nula). Primljeni znak se ne prikazuje na ekranu. Svi znaci sa tastature mogu biti rezultat funkcije, osim CTRL C koji prekida rad programa.

primer:

```
-
-
1000 PRINT "PRITISNITE BILO KOJI TASTER ZA NASTAVAK"
1010 PRINT "ILI CTRL/C ZA PREKID PROGRAMA ....."
1020 A$=INKEY$: IF A$="" THEN 1020
1030 'NASTAVAK PROGRAMA
```

5.16 INP

Format: INP(I)

Funkcija: Vraća celobrojnu vrednost (bajt) pročitanu sa U/I porta. I može imati vrednost iz opsega [0,65535].

primer:

```
10 PRINT INP(32768)
```

čita prvi bajt grafičke memorije

Vidi OUT instrukciju.

5.17 INPUT$

Format: INPUT$(X[,[#]Y])

Funkcija: Vraća grupu znakova dužine X pročitanu sa tastature ili iz datoteke pod brojem Y. Ako je za ulaz iskorišćena tastatura, znaci koji se kucaju ne prikazuju se na ekranu. Svi znaci mogu biti rezultat funkcije INPUT$, osim CTRL C koji prekida rad programa.

primer:

```
 5 'HEX DUMP SEKVENCIJALNE DATOTEKE
 10 OPEN "I",1,"PODACI.DTA"
 20 IF EOF(1) THEN 50
 30 PRINT HEX$(ASC(INPUT$(1,#1)));" ";
 40 GOTO 20
 50 PRINT
 60 END


 -
 -
 100 PRINT "OTKUCAJTE LOZINKU"
 110 L$=INPUT$(9)
 120 IF L$<>"MIKI MAUS" THEN END
 -
 -
```

5.18 INSTR

Format: INSTR([I,]X$,Y$)

Funkcija: Pretražuje Y$ da nadje pojavljivanje X$ u Y$ i vraća poziciju X$ u Y$. Opciono I je pozicija od koje će se pretraživati (ako nije navedeno, I je jednako 1). I mora biti u opsegu 1 do 255, a ukoliko je 0 ispisuje se greška **Illegal argument in line** XXXXX (Nedozvoljena vrednost argumenta u liniji XXXXX).
Ako je I>LEN(X$), X$ prazan ili Y$ ne postoji, INSTR vraća vrednost 0.
Ako je Y$ prazan, rezultat INSTR je I ili 1.

primer:

```
 10 X$ = "TELEFONI"
 20 Y$ = "E"
 30 PRINT INSTR(X$,Y$) INSTR(3,X$,Y$)
 40 END
 RUN
  2 4
 Ok
```

5.19 INT

Format: INT(X)
Funkcija: Zaokružuje X na manji ceo broj.

primer:

```
PRINT INT(99.89)
 99
Ok

PRINT INT(-99.89)
-100
Ok
```

Vidi CINT i FIX funkcije.

5.20 LEFT$

Format: LEFT$(X$,I)
Funkcija: Vraća prvih I znakova iz X$. Ako je I>LEN(X$), ceo X$ je rezultat.

primer:

```
10 A$ = "1000 KNJIGA"
20 B$ = LEFT$(A$,5)
30 PRINT B$
RUN
1000
Ok
```

Vidi MID$ i RIGHT$ funkcije.

5.21 LEN

Format: LEN(X$)
Funkcija: Vraća broj znakova sadržanih u X$. Broje se svi znaci uključujući i one koji se ne mogu ispisati.

primer

```
PRINT LEN("BASIC UPUTSTVO")
 14
Ok
```

5.22 LOC

Format: LOC(<broj datoteke>)

Funkcija: Za datoteke sa slučajnim pristupom:
LOC vraća broj zapisa koji je upravo pročitan ili upisan (sa GET ili PUT instrukcijom). Ako je datoteka otvorena, a nije joj pristupano radi čitanja ili pisanja LOC, vraća 0.
Za sekvencijalne datoteke:
LOC vraća broj sektora (1 sektor = 128 bajtova) pročitanih ili upisanih od otvaranja datoteke (zavisno da li je ulazna ili izlazna datoteka, vidi OPEN).

primer:

```
-
-
200 IF LOC(1)>100 THEN CLOSE 1
-
-
```

5.23 LOF

Format: LOF(<broj datoteke>)

Funkcija: Vraća broj sektora zapisanog u poslednjoj ekstenziji opisa smeštanja datoteke. Ovaj opis se nalazi u direktorijumu. Ako datoteka nije veća od 1 ekstenzije (1 ekstenzija = 128 sektora = 16 Kb), tada LOF vraća pravu dužinu datoteke (u sektorima).

primer:

```
-
-
100 IF LOF(1)>4 THEN PRINT "DATOTEKA VECA OD 1Kb"
-
-
```

5.24 LOG

Format: LOG(X)
Funkcija: Vraća logaritam X za osnovu e (prirodni logaritam). X mora biti veće od 0.

primer:

```
PRINT LOG(45/7)
 1.86075
Ok
```

5.25 LPOS

Format: LPOS(X)
Funkcija: Vraća poziciju glave na štampaču. X nema nikakvu funkciju.

primer:

```
-
-
100 IF LPOS(X)>60 THEN LPRINT
-
-
```

5.26 MID$

Format: MID$(X$,I[,J])
Funkcija: Vraća J znakova iz X$ počevši od I-te pozicije. I i J moraju biti u opsegu od 1 do 255. Ako je I=0 ispisuje se `Illegal argument in line` XXXXX (Nedozvoljena vrednost u liniji XXXXX). Ukoliko od I-te pozicije ima manje od J znakova, J se ignoriše i uzimaju se svi znakovi od I-te pozicije do kraja grupe znakova. Ako je I>LEN(X$), MID$ vraća praznu grupu znakova (dužine 0).

primer:

```
10 A$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
20 B$=""
30 PRINT A$
40 FOR I=LEN(A$) TO 1 STEP -1
50 B$=B$+MID$(A$,I,1)
60 NEXT
70 PRINT B$
RUN
ABCDEFGHIJKLMNOPQRSTUVWXYZ
ZYXWVUTSRQPONMLKJIHGFEDCBA
Ok
```

Vidi LEFT$ i RIGHT$ funkcije.

5.27 MKI$, MKS$, MKD$

Format: MKI$(<celobrojni izraz>)
MKS$(<izraz jednostruke preciznosti>)
MKD$(<izraz dvostruke preciznosti>)

Funkcija: Pretvara numeričke vrednosti u grupe znakova. Numeričke vrednosti pre stavljanja u bafer za datoteku sa slučajnim pristupom moraju biti pretvorene u znakove (zbog LSET i RSET instrukcija). MKI$ pretvara celobrojnu vrednost u grupu od 2 znaka (2 bajta), MKS$ jednostruku preciznost u 4 znaka, a MKD dvostruku u 8 znakova.

primer:

```
-
-
100 FIELD #1, 4 AS C$, 20 AS I$
110 LSET C$ = MKS$(CENA)
120 LSET I$ = IME$
130 PUT #1
-
```

Vidi CVI, CVS i CVD funkcije.

5.28 OCT$

Format: OCT$(X)
Funkcija: Vraća grupu znakova koja predstavlja vrednost X u oktalnom brojnom sistemu. Ukoliko X nije ceo broj, biće zaokružen.

primer:

```
PRINT OCT$(32)
 40
Ok
```

Vidi HEX$ funkciju.

5.29 PEEK

Format: PEEK(I)
Funkcija: Vraća vrednost I-te memorijske lokacije. Rezultat je u opsegu od 0 do 255. a I može imati vrednost od 0 do 65535.

primer:

```
10 FOR I:=0 TO 65535
20 PRINT CHR$(PEEK(I));
30 NEXT
RUN
-
-
-
```

Vidi POKE instrukciju.

5.30 POS

Format: POS(I)
Funkcija: Vraća poziciju kursora u okviru linije. Krajnja leva pozicija je 1. I se ignoriše.

primer:

```
IF POS(0)>75 THEN PRINT
```

vidi LPOS funkciju.

5.31 RIGHT$

Format: RIGHT$(X$,I)

Funkcija: Vraća poslednjih I znakova iz X$. Ako je I>=LEN(X$), vraća X$, a ako je I=0, vraća se prazna grupa znakova (dužine 0).

primer:

```
10 A$ = "1000 KNJIGA"
20 B$ = RIGHT$(A$,6)
30 PRINT B$
RUN
KNJIGA
Ok
```

Vidi LEFT$ i MID$ funkcije.

5.32 RND

Format: RND[(X)]

Funkcija: Vraća slučajni broj izmedju 0 i 1. Sekvenca slučajnih brojeva se ponavlja istim redosledom ako nije upotrebljena instrukcija RANDOMIZE. Ako je X<0, ponavlja se sekvenca od početka, za X=0 ponavlja se prethodni slučajni broj, a ako je X>0 ili izostavljeno, rezultat je sledeći slučajni broj sekvence.

primer:

```
10 FOR I=1 TO 5
20 PRINT INT(RND*100)
30 NEXT
RUN
 24 30 31 51 5
Ok
```

5.33 SGN

Format: SGN(X)
Funkcija: ako je X>0 rezultat je 1.
ako je X=0 rezultat je 0.
ako je X<0 rezultat je -1.

primer:

```
10 INPUT X
20 ON SNG(X)+2 GOTO 30,40,50
30 PRINT "NEGATIVAN BROJ" :END
40 PRINT "NULA" :END
50 PRINT "POZITIVAN BROJ"
60 END
RUN
? 7
POZITIVAN BROJ
```

5.34 SIN

Format: SIN(X)
Funkcija: Vraća sinus ugla X zadatog u radijanima. Rezultat SIN funkcije je jednostruke preciznosti.

primer:

```
PRINT SIN(1.5)
 .997495
Ok
```

5.35 SPACE$

Format: SPACE$(X)
Funkcija: Vraća X blanko znakova. Ako X nije ceo broj, biće zaokružen. X može imati vrednost od 0 do 255.

primer:

```
10 FOR I=1 TO 5
20 X$=SPACE$(I)
30 PRINT X$;I
40 NEXT I
RUN
  1
   2
    3
     4
      5
Ok
```

Vidi SPC i TAB funkcije.

5.36 SPC

Format: SPC(I)

Funkcija: Štampa I blanko znakova. SPC može biti upotrebljen samo uz PRINT i LPRINT instrukciju. I može biti od 0 do 255. Uz SPC se podrazumeva znak tačka-zarez (;) na kraju.

primer:

```
PRINT "OVDE I" SPC(20) "ONDE"
OVDE I                    ONDE
Ok
```

Vidi SPACE$ i TAB funkcije.

5.37 SQR

Format: SQR(X)

Funkcija: Vraća kvadratni koren vrednosti X. X mora biti ≥ 0.

primer:

```
PRINT SQR(10)
 3.16228
Ok
```

5.38 STR$

Format: STR$(X)
Funkcija: Vraća znakovnu reprezentaciju vrednosti X (pretvara brojeve u grupu znakova).

primer:

```
? LEFT$(STR$(552.12),3)
 55          (znak broja zauzima jedno mesto)
```

Vidi VAL funkciju.

5.39 STRING$

Format: STRING$(I,J)
STRING$(I,X$)
Funkcija: Prvi format vraća I znakova sa ASCII kodom J.
Drugi format vraća prvi znak iz X$.

primer:

```
10 X$=STRING$(10,45)
20 PRINT X$;" IZVESTAJ ";X$
RUN
---------- IZVESTAJ ----------
Ok
```

5.40 TAB

Format: TAB(I)
Funkcija: Pozicionira kursor u I-tu kolonu. Ako je I manje od trenutnog položaja, kursor se postavlja u sledeću liniju na I-tu poziciju. I može imati vrednost od 1 do 255. TAB se upotrebljava samo u PRINT i LPRINT instrukcijama.

primer:

```
10 PRINT "IME" TAB(40) "TELEFON"
20 PRINT "----------------------" TAB(40) "-------"
RUN
IME                                     TELEFON
--------------------                    -------
Ok
```

Vidi SPC i SPACE$ funkcije.

5.41 TAN

Format: `TAN(X)`

Funkcija: Vraća tangens ugla X zadatog u radijanima. Rezultat TAN funkcije je jednostruke preciznosti. Ako dodje do prekoračenja, ispisuje se poruka `Overflow` (prekoračenje), rezultat je mašinska beskonačnost sa odgovarajućim predznakom, a izvršavanje programa se nastavlja.

primer:

```
10 A=TAN(ALFA)
```

5.42 USR

Format: `USR[<broj>](X)`

Funkcija: Poziva mašinski potprogram i prosledjuje mu vrednost X. <broj> je cifra izmedju 0 i 9. Podrazumeva se 0, ako drugačije nije navedeno. Adresa mašinskog potprograma definisana je u DEF USR instrukciji.

primer:

vidi DEF USR

5.43 VAL

Format: `VAL(X$)`

Funkcija: Vraća numeričku vrednost grupe znakova koji predstavljaju numeričku konstantu. VAL ignoriše blanko, CR, LF i TAB znakove.

primer:

```
10 A$="  12.4ABCD"
20 PRINT VAL(A$)+5
RUN
 17.4
Ok
```

Vidi STR$ funkciju.

5.44 VARPTR

Format: 1) VARPTR(<ime promenljive>)
2) VARPTR(#<broj datoteke>)

Funkcija: 1) Vraća adresu prvog bajta podatka promenljive. Promenljiva mora postojati tj. mora joj se pre poziva ove funkcije dodeliti neka vrednost; u suprotnom ispisuje se greška `Illegal function call` (Nedozvoljen poziv funkcije). VARPTR vraća vrednost u opsegu -32768 do 32767. Za znakovne promenljive VARPTR vraća adresu na kojoj se nalazi dužina sadržaja promenljive. Adresa prvog znaka sadržaja promenljive nalazi se na memorijskim lokacijama koje slede.

primer:

```
10 A%=10000
20 A$="ABCDEFG"
30 A1=VARPTR(A%)
40 PRINT PEEK(A1)+256*PEEK(A1+1)
50 D=VARPTR(A$)
60 AP=PEEK(D+1)+256*PEEK(D+2)
70 FOR I=0 TO PEEK(D)-1
80 PRINT CHR$(PEEK(AP+I);
90 NEXT
RUN
 10000
ABCDEFG
Ok
```

2) Vraća početnu adresu bafera za navedenu datoteku.

Dodatak A — Rad sa datotekama

U GBASIC-u postoje dva tipa datoteka koje se čuvaju na disku: sekvencijalne i datoteke sa slučajnim pristupom. One se mogu kreirati BASIC programom. Pristup i manipulacija podacima iz datoteke takodje se vrše programom.

A.1 Sekvencijalne datoteke

Podaci se u sekvencijalnu datoteku upisuju jedan za drugim u poretku kojim su poslati i istim redosledom se čitaju iz nje. Instrukcije i funkcije koje se koriste za rad sa sekvencijalnim datotekama su: OPEN, PRINT#, PRINT# USING, INPUT#, LINE INPUT#, WRITE#, CLOSE, EOF i LOC. Sledeći programski koraci su neophodni da se sekvencijalna datoteka kreira i izvrši pristup njenim podacima:

1. `OPEN "O",#1,"SEKDAT"`

Otvori sekvencijalnu datoteku #1 (broj 1) za upis imena SEKDAT.

2. `PRINT#1,A$,B$,C$`

Upiši znakovne promenljive A$, B$ i C$ u datoteku (umesto PRINT# može se upotrebiti i WRITE#).

3. `CLOSE#1`

Zatvori datoteku (kraj upisa).

4. `OPEN "I",#1,"SEKDAT"`

Otvori za čitanje datoteku #1 imena SEKDAT.

5. `INPUT#1,X$,Y$,Z$`

Dodeli znakovnim promenljivama X$, Y$ i Z$ vrednosti koje se nalaze u datoteci #1.

primer:

```
10 OPEN "O",#1,"SEKDAT"
20 INPUT "IME";I$
30 IF I$="KRAJ" THEN CLOSE#1:END
40 INPUT "GODINA RODJENJA";G$
50 PRINT#1, I$; ";"; D$; ","   'zarez razdvaja podatke
60 PRINT: GOTO 20
RUN
IME? SLOBODAN BADZEVIC
GODINA RODJENJA? 1949

IME? DANKO JEVTOVIC
GODINA RODJENJA? 1963

IME? MILAN KOVINIC
GODINA RODJENJA? 1949

IME? KRAJ
Ok
```

```
100 OPEN "I",#1,"SEKDAT"
110 INPUT#1,I$,G$
120 IF G$ ="1949" THEN PRINT I$
130 GOTO 110
RUN
SLOBODAN BADZEVIC
MILAN KOVINIC
Input past end in 110
Ok
```

Ovaj program čita redom svaki podatak u datoteci. Kada se svi podaci pročitaju ispisuje se greška Input past end in 110 (čitanje iza kraja). Ako želimo da je izbegnemo, potrebno je dodati programski red u kome se vrši testiranje kraja datoteke:

```
105 IF EOF(1) THEN END
```

i izmeniti liniju 130 :

```
130 GOTO 105
```

Program koji kreira sekvencijalnu datoteku može u nju upisati formatirane podatke upotrebom PRINT# USING instrukcije.

primer:

```
PRINT#1, USING "####.###, "A, B, C, D
```

Upisuje numeričke podatke na disk, a zarez na kraju formata služi da odvoji podatke u datoteci.

Funkcija LOC, upotrebljena u radu sa sekvencijalnom datotekom, vraća broj sektora u koje su podaci upisani ili iz kojih su podaci pročitani od trenutka njenog otvaranja. Sektori su blokovi podataka dužine 128 bajtova.

A.1.1 Dodavanje podataka sekvencijalnoj datoteci

Ako sekvencijalna datoteka već postoji na disku i želimo da na njen kraj dodamo još podataka, to ne možemo uraditi jednostavnim otvaranjem datoteke i upisivanjem novih podataka. Stari podaci bivaju uništeni u trenutku otvaranja datoteke za upis. Sledeći programski koraci ilustruju postupak za dodavanje novih podataka na kraj već postojeće sekvencijalne datoteke:

1. `OPEN "I",#1,"SEKDAT"`

Otvori staru datoteku za čitanje.

2. `OPEN "O",#2,"NOVDAT"`

Otvori novu datoteku za upis.

3. Pročitaj podatke iz stare i upiši ih u novu datoteku.
4. Upiši nove podatke u novu datoteku.
5. Zatvori staru datoteku i uništi je.
6. Promeni ime nove datoteke (NOVDAT) u SEKDAT.

primer:

```
10 ON ERROR GOTO 2000
20 OPEN "I",#1,"SEKDAT"
30 OPEN "O",#2,"NOVDAT"
40 IF EOF(1) THEN 80
50 LINE INPUT#1,A$
60 PRINT#2,A$
70 GOTO 40
80 CLOSE#1
90 KILL "SEKDAT"
100 INPUT "IME";I$
110 IF I$ ="KRAJ" THEN 170
120 INPUT "GODINA RODJENJA";G$
130 PRINT#2,I$
140 PRINT#2,G$
150 PRINT:GOTO 100
160 NAME "NOVDAT" AS "SEKDAT"
170 CLOSE:END
2000 IF ERR=53 AND ERL=20 THEN OPEN "O",#2,"NOVDAT":RESUME 100
2010 ON ERROR GOTO 0
```

Ovaj program koristi instrukciju LINE INPUT# za čitanje znakovnih nizova koji sadrže zareze iz datoteke na disku. LINE INPUT# čita znakove dok se ne pojavi znak CR (kraj reda) ili ne pročita 255 znakova. U liniji 2000 se nalazi zamka za hvatanje grešku **File not found** (datoteka nije nadjena) koja može da se pojavi u liniji 20 ako datoteka sa imenom SEKDAT ne postoji.

A.2 Datoteke sa slučajnim pristupom

Velika prednost datoteka sa slučajnim pristupom u odnosu na sekvencijalne je u tome što se podacima može pristupati slučajno tj. preko reda, bilo gde u datoteci. Ovo je moguće zato što je svaki podatak zasebna celina koja se zove zapis (*record*) kojoj je pridružen redni broj. Sledeća prednost leži u manjem prostoru koji datoteka zauzima pošto se podaci pamte u binarnom obliku. (Podaci sekvencijalne datoteke zabeleženi su u ASCII kodu).

Instrukcije i funkcije koje se koriste za rad sa datotekama sa slučajnim pristupom su: OPEN, FIELD, LSET, RSET, GET, PUT, CLOSE, LOC, MKI$, MKS$, MKD$, CVI, CVS i CVP.

Sledeći programski koraci neophodni su za kreiranje datoteke sa slučajnim pristupom:

1. `OPEN "R",#1,"IMEDAT",32`

Otvori datoteku za slučajan pristup. U ovom primeru dužina zapisa je 32 bajta. Ako se podatak za dužinu zapisa ne navede, podrazumeva se dužina od 128 bajtova.

2. `FIELD #1 20 AS A$, 6 AS B$, 6 AS C$`

Instrukcijom FIELD se rezerviše mesto u baferu za promenljive koje će biti upisane u datoteku. U primeru se za A$ rezerviše 20 bajtova/znakova za A$, 6 za B$ i 6 za C$.

3.
```
LSET A$ =I$
LSET B$ =TEL$
LSET C$ =MKI$(BRIMP%)
```

Instrukcija LSET prenosi podatke u bafer za datoteku sa slučajnim pristupom. Numeričke vrednosti se moraju pretvoriti u znakovne pre prenošenja u bafer. Za to se koriste funkcije MKI$, MKS$ i MKD$ koje pretvaraju celobrojne vrednosti, vrednosti jednostruke preciznosti i vrednosti dvostruke preciznosti u znakovne vrednosti.

4. `PUT #1,REDBR%`

Upiši podatke iz bafera na disk u zapis sa rednim brojem REDBR% koristeći instrukciju PUT.

primer:

```
10 OPEN "R",#1,"IMEDAT",32
20 FIELD #1,20 AS A$, 6 AS B$, 6 AS C$
30 INPUT "REDNI BROJ ZAPISA"; REDBR%
40 INPUT "IME";I$
50 INPUT "TELEFON";TEL$
60 INPUT "BROJ IMPULSA"; BRIMP%
70 LSET A$=I$
80 LSET B$=TEL$
90 LSET C$=BRIMP$%
100 PUT #1,REDBR%
110 GOTO 30
```

Programski koraci potrebni za čitanje datoteke sa slučajnim pristupom su:

1. `OPEN "R",#1,"IMEDAT",32`

Otvori datoteku za slučajni pristup.

2. `FIELD #1,20 AS A$, 6 AS B$, 6 AS C$`

Instrukcijom FIELD rezerviši mesto u baferu za promenljive koje će biti pročitane iz datoteke.

3. `GET #1,REDBR%`

Upotrebi GET instrukciju za čitanje zapisa pod rednim brojem REDBR% i njegovo smešetanje u bafer.

4. Podacima u baferu se sada može pristupiti. Numeričke vrednosti se moraju pretvoriti u brojeve funkcijama CVI, CVS i CVD.

primer:

```
10 OPEN "R",#1,"IMEDAT",32
20 FIELD #1,20 AS A$,6 AS B$,6 AS C$
30 INPUT "REDNI BROJ ZAPISA";REDBR%
40 GET#1,REDBR%
50 PRINT "IME:";A$
60 PRINT "TELEFON:";B$
70 PRINT "BROJ IMPULSA";CVI(C$)
80 PRINT
90 GOTO 30
```

Dodatak B — Izvedene funkcije

Matematičke funkcije koje se mogu izvesti (nisu ugradjene u BASIC):

FUNKCIJA	BASIC	REALIZACIJA
sekans	SEC(X)	= 1/COS(X)
kosekans	CSC(X)	= 1/SIN(X)
kotangens	COT(X)	= 1/TAN(X)
inv.sinus	ARCSIN(X)	= ATN(X/SQR(-X*X+1))
inv.kosinus	ARCCOS(X)	= -ATN(X/SQR(-X*X+1))+1.5708
inv.sekans	ARCSEC(X)	= ATN(X/SQR(-X*X+1))+SGN(SGN(X)-1)*1.5708
inv.kosekans	ARCCSC(X)	= ATN(X/SQR(-X*X+1))+(SGN(X)-1)*1.5708
inv.kotangens	ARCCOT(X)	= ATN(X)+1.5708
hip.sinus	SINH(X)	= (EXP(X)-EXP(-X))/2
hip.kosinus	COSH(X)	= (EXP(X)+EXP(-X))/2
hip.tangens	TANH(X)	= EXP(-X)/(EXP(X)+EXP(-X))*2+1
hip.sekans	SECH(X)	= 2/(EXP(X)+EXP(-X))
hip.kosekans	CSCH(X)	= 2/(EXP(X)-EXP(-X))
hip.kotangens	TANH(X)	= EXP(-X)/(EXP(X)-EXP(-X))*2+1
inv.hip.sinus	ARCSINH(X)	= LOG(X+SQR(X*X+1))
inv.hip.kosinus	ARCCOSH(X)	= LOG(X+SQR(X*X-1))
inv.hip.tangens	ARCTANH(X)	= LOG((1+X)/(1-X))/2
inv.hip.sekans	ARCSECH(X)	= LOG((SQR(-X*X+1)+1)/X)
inv.hip.kosekans	ARCCSCH(X)	= LOG((SGN(X)*SQR(X*X+1)+1)/X)
inv.hip.kotangens	ARCTANH(X)	= LOG((X+1)/(X-1))/2

Navedene formule pretstavljaju analitičke definicije funkcija. Njihova numerička tačnost zavisi kako od tačnosti osnovnih funkcija, tako i od konkretnih vrednosti argumenata.

Dodatak C — Poruke o greškama

Poruke o greškama uredjene su po kodu greške.

1 **NEXT WITHOUT FOR (NEXT bez FORa)**

Promenljive uz NEXT nema ni u jednoj predhodno izvršenoj FOR instrukciji ili uopšte nema FOR instrukcije.

2 **SYNTAX ERROR (sintaksna-pravopisna greška)**

Programska linija ili komanda ima pravopisnu grešku. Neki od mogućih uzroka su: neodgovarajući broj otvorenih ili zatvorenih zagrada, pogrešno otkucana komanda, pogrešna interpunkcija itd.

3 **RETURN WITHOUT GOSUB (RETURN bez GOSUBa)**

Izvršena je instrukcija RETURN, a nije bilo poziva potprograma (GOSUB instrukcije).

4 **OUT OF DATA (nema podataka)**

READ instrukcija nema više šta da pročita iz DATA liste, jer su svi podaci pročitani ili ih nije ni bilo.

5 **ILLEGAL FUNCTION CALL (nedozvoljen poziv funkcije)**

Vrednost prosledjena aritmetičkoj ili znakovnoj funkciji je izvan dozvoljenog opsega. Ova greška se može pojaviti i u sledećim slučajevima:

- element niza ima preveliki ili negativan indeks,
- negativna osnova stepenovana necelobrojnim eksponentom,
- poziv USR funkcije kojoj nije zadata početna adresa,
- nepravilne ulazne vrednosti za:
  OUT, POKE, ON...GOTO, MID$, LEFT$, RIGHT$, INP, PEEK, TAB, SPC, STRING$, SPACE$, INSTR, SQR, LOG...

6 **OVERFLOW (prekoračenje)**

Rezultat je veći od maksimalne vrednosti broja u BASICu.

7 **OUT OF MEMORY (nema (više) memorijskog prostora)**

Mogući uzroci: program je prevelik, ima mnogo FOR petlji, previše potprograma, mnogo promenljivih, preveliki nizovi, previše komplikovan izraz.

8 **UNDEFINED LINE (nepostojeća linija)**

Linijski broj u GOTO, GOSUB, IF...THEN...ELSE ili DELETE ne postoji u programu.

9 SUBSCRIPT OUT OF RANGE (indeks izvan opsega)

Traženi element je sa pogrešnim brojem dimenzija ili izvan definisanog opsega niza.

10 REDIMENSIONED ARRAY (redimenzionisan niz)

Dve DIM instrukcije definišu isti niz ili se DIM upotrebljava posle dodeljivanja vrednosti elementu niza.

11 DIVISION BY ZERO (deljenje nulom)

U izrazu se deli nulom ili se vrednost nula diže na negativni stepen. U oba slučaja rezultat je mašinska beskonačnost sa odgovarajućim znakom, a izvršavanje programa se nastavlja.

12 ILLEGAL DIRECT (nedozvoljeno u komandnom načinu rada)

Instrukcija koja se ne izvršava u komandnom načinu rada je otkucana kao komanda.

13 TYPE MISMATCH (nesaglasnost tipova (vrednosti))

Umesto numeričke vrednosti isporučena je znakovna ili obrnuto.

14 OUT OF STRING SPACE (nema prostora za znakovne promenljive)

Znakovna promenljiva ne može da stane u memorijski prostor koji se koristi za pamćenje znakovnih vrednosti.

15 STRING TOO LONG (grupa znakova suviše dugačka)

Formirana je grupa znakova duža od 255.

16 STRING FORMULA TOO COMPLEX (previše složen znakovni izraz)

Znakovni izraz je previše komplikovan ili predugačak. Treba ga razbiti na više kraćih izraza.

17 CAN'T CONTINUE (nastavak nije moguć)

Pokušaj izvršavanja programa koji je zaustavljen na fatalnoj grešci ili je posle prekida ispravljan.

18 UNDEFINED USER FUNCTION (nedefinisana korisnička funkcija)

USR funkcija je pozvana pre no što je definisana (sa DEF USR).

19 NO RESUME (nema RESUME)

U potprogramu za obradu greške nema RESUME instrukcije (nije navedeno gde se treba vratiti).

20 RESUME WITHOUT ERROR (povratak bez pojave greške)

Program je pri izvršavanju naišao na RESUME instrukciju a predhodno se nije desila greška.

21 UNPRINTABLE ERROR (neispisiva greška)

Ne postoji poruka o grešci koja je prekinula program. Ova greška nastaje obično korišćenjem ERROR instrukcije sa brojem greške koja nema poruku.

22 MISSING OPERAND (nedostaje operand)

Izraz sadrži operator bez pratećih operanada ili je zadata instrukcija sa nedovoljnim brojem argumenata.

23 LINE BUFFER OVERFLOW (prekoračenje linijskog bafera)

Pokušaj unosa linije sa previše znakova.

26 FOR WITHOUT NEXT (FOR bez NEXTa)

Ponovo je izvršen isti FOR a da predhodno nije izvršen odgovarajući NEXT. Petlje se preklapaju.

29 WHILE WITHOUT WEND (WHILE bez WENDa)

WHILE nema odgovarajući WEND.

30 WEND WITHOUT WHILE (WEND bez WHILEa)
WEND instrukcija je izvršena pre WHILE instrukcije.

DISK GREŠKE

50 FIELD OVERFLOW (prekoračenje dužine polja)

FIELD instrukcijom je zadato da se rezerviše više mesta no što je moguće.

51 INTERNAL ERROR (unutrašnja greška)

Interna greška BASIC interpretatora; javite ovlašćenom servisu pod kojim uslovima ste izazvali ovu grešku.

52 BAD FILE NUMBER (pogrešan broj datoteke)

Pokušan pristup datoteci sa brojem koji nije definisan (ni u jednoj OPEN instrukciji) ili je broj izvan opsega.

53 FILE NOT FOUND (datoteka nije nadjena)

U LOAD, KILL ili OPEN navedeno je ime datoteke koja ne postoji na disku.

54 BAD FILE MODE (pogrešan način rada sa datotekom)

Mogući uzroci: pokušaj upotrebe PUT, GET ili LOF nad sekvencijalnom datotekom, pokušaj učitavanja datotcke sa slučajnim pristupom LOAD komandom, nepoznat način rada u OPEN instrukciji (različit od I, O ili R), MERGE sa datotekom koja nije u ASCII formatu.

55 **FILE ALREADY OPEN (datoteka je već otvorena)**

Pokušaj otvaranja sekvencijalne datoteke koja je već otvorena ili pokušaj brisanja (sa KILL) datoteke koja je još uvek otvorena.

57 **DISK I/O ERROR (U/I disk greška)**

Fatalna disk greška, operativni sistem ne može otkloniti grešku. Greška se javlja prilikom pristupa podacima na disku. Promenite disketu.

58 **FILE ALREADY EXISTS (datoteka postoji na disku)**

Novo ime datoteke u NAME instrukciji već postoji na disku.

61 **DISK FULL (disk je popunjen)**

Nema više slobodnog prostora na disku (upotrebite RESET i promenite disketu).

63 **INPUT PAST END (čitanje iza kraja)**

Pokušaj čitanja prazne datoteke ili čitanje iza poslednjeg podatka datoteke. Upotrebite EOF funkciju.

64 **BAD FILE NAME (pogrešno ime datoteke)**

Ime datoteke navedeno u LOAD, SAVE, KILL ili OPEN nije u skladu sa pravilima zadavanja imena u operativnom sistemu (najviše 8 znakova, opciona tačka i 3 znaka).

66 **DIRECT STATEMENT IN FILE (komanda u datoteci)**

Prilikom LOAD komande ASCII datoteka koja se učitava ima pored programskih linija komande (bez broja linije).

67 **TOO MANY FILES (previše datoteka)**

Pokušaj kreiranja nove datoteke (SAVE, OPEN) kada je svih 255 mesta u direktorijumu ispunjeno.

Dodatak D — ASCII kod

ASCII kod (American Standard Code for Information Interchange)

Kontrolni znaci				Znaci koji se mogu prikazati								
DEC	HEX	CHR	IME	DEC	HEX	CHR	DEC	HEX	CHR	DEC	HEX	CHR
0	00	^@	NUL	32	20		64	40	@	96	60	`
1	01	^A	SOH	33	21	!	65	41	A	97	61	a
2	02	^B	STX	34	22	"	66	42	B	98	62	b
3	03	^C	ETX	35	23	#	67	43	C	99	63	c
4	04	^D	EOT	36	24	$	68	44	D	100	64	d
5	05	^E	ENQ	37	25	%	69	45	E	101	65	e
6	06	^F	ACK	38	26	&	70	46	F	102	66	f
7	07	^G	BEL	39	27	'	71	47	G	103	67	g
8	08	^H	BS	40	28	(	72	48	H	104	68	h
9	09	^I	HT	41	29	)	73	49	I	105	69	i
10	0A	^J	LF	42	2A	*	74	4A	J	106	6A	j
11	0B	^K	VT	43	2B	+	75	4B	K	107	6B	k
12	0C	^L	FF	44	2C	,	76	4C	L	108	6C	l
13	0D	^M	CR	45	2D	-	77	4D	M	109	6D	m
14	0E	^N	SO	46	2E	.	78	4E	N	110	6E	n
15	0F	^O	SI	47	2F	/	79	4F	O	111	6F	o
16	10	^P	DLE	48	30	0	80	50	P	112	70	p
17	11	^Q	DC1	49	31	1	81	51	Q	113	71	q
18	12	^R	DC2	50	32	2	82	52	R	114	72	r
19	13	^S	DC3	51	33	3	83	53	S	115	73	s
20	14	^T	DC4	52	34	4	84	54	T	116	74	t
21	15	^U	NAK	53	35	5	85	55	U	117	75	u
22	16	^V	SYN	54	36	6	86	56	V	118	76	v
23	17	^W	ETB	55	37	7	87	57	W	119	77	w
24	18	^X	CAN	56	38	8	88	58	X	120	78	x
25	19	^Y	EM	57	39	9	89	59	Y	121	79	y
26	1A	^Z	SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B	^[	ESC	59	3B	;	91	5B	[	123	7B	{
28	1C	^\	GS	60	3C	<	92	5C	\	124	7C	|
29	1D	^]	FS	61	3D	=	93	5D	]	125	7D	}
30	1E	^^	RS	62	3E	>	94	5E	^	126	7E	~
31	1F	^_	US	63	3F	?	95	5F	_	127	7F	DEL