

BASIC M6809

NAVODILO ZA UPORABO

V S E B I N A

Stran

1.	UVOD	1
2.	NAČINA DELOVANJA	1
3.	VNAŠANJE PROGRAMA TER UREJANJE	1
3.1	Popravljanje pred zaključkom vrste	1
3.2	Popravljanje vnesenih vrstic	2
3.2.1	Enostavno popravljanje v BASIC-u	2
3.2.2	Popravljanje z ukazom EDIT	2
4.	VRSTICE	5
4.1	Takojšnje in programske vrstice	5
4.2	Komentar	5
4.3	Zgradba vrstice	5
5.	KONSTANTE, SPREMENLJIVKE	6
5.1	Konstante	6
5.2	Spremenljivke	7
5.3	Prirejanje vrednosti spremenljivkam	7
5.4	Indeksirane spremenljivke	8
6.	OPERATORJI	8
6.1	Aritmetični operatorji	8
6.2	Logični operatorji	9
6.3	Relacijski operatorji	9
6.4	Operatorji pri znakovnih spremenljivkah	10
6.5	Hierarhija vseh operatorjev	10
7.	TAKOJŠNJI UKAZI	11
8.	PROGRAMSKI UKAZI	13
8.1	Prireditveni ukazi	14
8.2	Ukazi s spremembo poteka izvajanja	15
8.3	IF-THEN-ELSE, pogojno izvajanje	16
8.4	Vhodno-izhodni stavki	17
8.5	Zanka FOR-NEXT	19
8.6	Zaključni stavki	20
8.7	Ostali ukazi	21

prenos trejim osebam ali uporaba v nedogovorjene namene ni dovoljena po znamnu o avtorski pravici

nažin			
bit cow			
ene			
pedpil			



9.	VSEBOVANE FUNKCIJE	23
9.1	Aritmetične funkcije	23
9.2	Trigonometrične funkcije	24
9.3	Znakovne funkcije	24
9.4	Vhodno-izhodne funkcije	26
9.5	Ostale funkcije	27
10.	NAPAKE	28
10.1	Javljanje napak	28
10.2	Seznam napak	28
10.3	Programska kontrola napak ob izvajanju	30
11.	UPORABNIKOVE FUNKCIJE	32
12.	POPRAVKI IN DODATKI	

Vsek prenos tretjim osebam ali uporata v nedogovorljene namene ni dovoljena do zanaru o avtorski pravici

atn	
En avn	
dne	
popis	



1. UVOD

VTŠ BASIC je popoln BASIC, vsebuje številčne in znakovne (numeric, character) spremenljivke ter niz matematičnih in znakovnih funkcij. Poseben način kodiranja ukazov mu omogoča veliko hitrost.

2. NAČINA DELOVANJA

BASIC pozna dva načina delovanja. V takojšnjem načinu vnesemo ukazno vrstico, ki naj se takoj izvrši, brez začetne številke. Tako lahko računamo podobno kot z navadnim kalkulatorjem. V programskem načinu pa vnašamo oštevilčene ukazne vrstice, ki tvorijo skupaj program. Program v BASIC-u lahko pozneje kolikokrat hočemo izvršimo z ukazom RUN, ali pa ga shranimo (SAVE).

Primer: Takojšen način:

PRINT 7 * 9

pri pritisku na CR dobimo na zaslonu odgovor 63

Programski način:

10 PRINT 7 * 9

s pritiskom na CR se ta stavek vnese v pomnilnik računalnika

Z ukazom RUN in CR se program izvede in na zaslonu dobimo rezultat 63.

3. VNAŠANJE PROGRAMA TER UREJANJE

3.1 Popravljanje pred zaključkom vrstice

Vsako vtipkanico vrstico lahko popravimo še preden jo zaključimo s tipko CR. Pri tem uporabljamo naslednje kontrolne znake:

CR - (carriage return, return): zaključek vtipkane vrstice, priprava za sprejem naslednje.

DEL - (delete): brisanje zadnjega vtipkanega znaka. S to tipko lahko izbrišemo nazadnje vtipkane znake in jih nato na novo vtipkamo.

C/X (Control/X, tipki control in X pritisnjeni istočasno): brisanje celotne vrstice.

C/C - (control/C): prekinitve, deluje pri vnašanju vrstic podobno kot C/X.

S tipko ESC (escape) vključimo oziroma izključimo zapis grafičnih znakov na zaslon. Znaki, ki jih vtipkamo in so mišljeni v našem programu kot grafični, se nam tako (za našo kontrolo) pojavljajo na zaslonu v obliki svojih grafičnih ekvivalentov. Seveda pa so tudi ti znaki shranjeni v računalniku kot normalni ASCII znaki, tako da moramo za njihovo uporabo kot grafične simbole s samim programom zagotoviti vklop grafičnega načina ispisa na zaslon. Prav tako se izpišejo vedno kot normalni znaki na printerju, oziroma zaslonu (ukaz LIST, EDIT), če zaslon tačas nima vklopljenega grafičnega načina prikaza.

V takojšnjem delovanju (vtipkana vrstica ukaza brez številke) lahko popravimo le na opisan način. Čim pritisnemo tipko CR, se ukaz izvede, ozziroma javi napaka pri eventualni nepravilnosti v vrsti. Vrstice po izvedbi ne moremo več priklicati na zaslon.

3.2 Popravljanje vnesenih vrstic

V programskem načinu pa vnašamo oštevilčene vrstice drugo za drugo. Do zaključka s CR lahko vsako vrstico popravimo enako kot takojšnje vrstice, po pritisku na tipko CR pa se, v kolikor ni bila že v sami obliki vrstice napaka, vrstica shrani ter poveže v programski niz z ostalimi vrsticami po njihovih zaporednih številkah.

3.2.1 Enostavno popravljanje v BASIC-u

Enostavno popravljanje v programskem načinu je omogočeno s tem, da na novo vtipkana vrstica z že obstoječo številko prekrije prejšnjo vrstico. Že obstoječo vrstico lahko tako tudi izbrišemo, ako vtipkamo le njeno številko.

3.2.2 Popravljanje z ukazom EDIT

Ob dodatku posebnega znakovnega urejevalnika (editorja) za BASIC (EDITB), pa je omogočeno veliko lažje in hitrejše popravljanje programskega vrednosti. Editor pokličemo z ukazom EDIT, ki ga uporabimo lahko tudi samo za prikaz programa, stran za stranjo, namesto ukaza LIST.



Z ukazom EDIT (ozziroma EDIT n₁, EDIT n₁, n₂) nam urejevalnik prikaže prvo stran vtipkanega programa (ozziroma vrstico n₁, vrstice n₁, n₂) na zaslonu, kjer lahko s pomikom utripajočega kurzora po vsem zaslonu popravljamo stran za stranjo programa.

UKAZI editorja EDITB:

C/E (control/E)	- izhod iz editorja v basic (exit)
C/N (control/N)	- urejanje naslednje strani (next)
C/X (control/X)	- brisanje vrstice
C/R (control/R)	- obnova tekoče strani
C/T (control/T)	- izpis prve strani
C/P (control/P)	- izpis prejšnje strani
DEL (delete)	- brisanje znaka pred kurzorjem
↓ puščica dol	- pomik za eno vrsto navzdol
← puščica v levo	- pomik za en znak v levo
→ puščica v desno	- pomik za en znak v desno
↑ puščica gor	- pomik za eno vrsto navzgor

PРИМЕРИ:

A. Z ukazom (v basicu):

EDIT 10

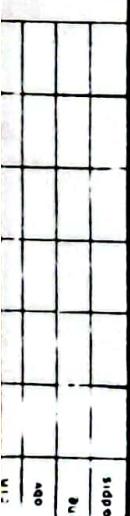
pokličemo urejevalnik, ki nam izpiše samo deseto vrstico na zaslon. Vrstico lahko popravimo. Če vrstice s to številko ni, ostane zaslon prazen. S tipko C/N nato pokličemo na zaslon naslednjo stran programa od vključno vrstice 10 dalje, ali pa se vrnemo v basic s tipko C/E.

B. Z ukazom v basicu:

EDIT

izpiše urejevalnik prvo stran programa na zaslon. S tipko C/N listamo na zaslonu stran za stranjo, dokler ne najdemo vrstice, ki jo želimo popraviti (ob koncu programa se prične zopet prva stran).

Kurzor spravimo s tipko za njegov premik (↓ → ← ↑) do vrstice, ki jo želimo popraviti. Ako je popravkov dosti, jo izbrišemo s tipko C/X. Ostane izpisana le njena številka, vtipkamo njenovo novo vsebino. Ako pa je potrebno popraviti le manjši del vrstice, spravimo kurzor do mesta popravka in zbrisemo napačni





tekst s tipko DEL (brisanje znaka pred kurzorjem), nato pa vrinemo (vtipkamo) pravilni tekst.

S tipko C/R obnovimo tekočo stran in tako lahko kontroliramo, če smo pravilno popravili tekst. Če želimo iz EDITORJA vrvati vrstice z novo številko, uporabimo naslednji trik: vrstici, ki je blizu mesta, kjer želimo vriniti novo vrstico, brišemo številko in vpišemo številko nove vrstice. S tipko C/R obnovimo tekočo stran. Vrstica kateri smo spremenili številko se pojavi dvakrat s svojo prvotno številko in z novo številko. Kurzor nastavimo v vrinjeno vrstico in z CTL/X brišemo njeno vsebino. Nato lahko vtipkamo novo vsebino vrinjene vrstice.

V primeru, da želimo na zaslonu izlistati prejšnjo stran, to dosežemo s tipko CTL/P, vendar pa to lahko izvajamo samo enkrat (ne moremo listati nazaj, kar z CTL/N naprej). Listanje naprej, da lahko potem ponovno izpišemo prejšnjo stran.

S tipko C/E se vrnemo v basic.

C. Z ukazom v basicu:

EDIT

prav tako lažje kot z ukazom LIST pregledamo stran za stranjo programa, obenem pa ugotovimo morebitne napake.

D. Ob ukazu:

RUN

dobimo na zaslonu sporočilo o napaki v basicu

ERRON NO ... IN LINE 30

Z ukazom

EDIT 30

lahko popravimo napačno vrstico 30 z editorjem.

E. Če želimo spremeniti samo številko neke vrstice, lahko storimo to v editorju, vendar dobimo tako podvojeno vrstico z novo številko in moramo vrstico s staro številko nato še (bodisi v basicu ali editorju) zbrisati. Prav tako ne smemo pozabiti na to, da bo vrstica z novo številko izbrisala (prekrila) morebitno že prej obstoječo vrstico z enako številko. Če hočemo na primer spremeniti številke treh vrstic



10 A = 10

20 B = 20

30 C = 30

v vrstice:

20 A = 10

30 B = 20

40 C = 30

moramo to storiti tako, da spremenimo najprej številko 30 v 40, nato 20 v 30 in 10 v 20, ob koncu pa izbrišemo vrstico s številko 10. Če bi spremenjali številke v vrstnem redu 10 - 20 - 30, bi vsaka zbrisala (prekrila) naslednjo z isto številko in bi tako ostalo naslednje:

Ko bi spremenili številko 10 v 20:

10 A = 10

20 A = 10

30 C = 30

- 20 v 30 :

10 A = 10

20 A = 10

30 A = 10

- 30 v 40 :

10 A = 10

20 A = 10

30 A = 10

40 A = 10

F. Z ukazom EDIT 10, 30

se bodo na zaslonu izpisale vse vrstice z številkami: med 10 in 30.

4. VRSTICE

4.1 Komentar

Komentar (poljubni tekst; ponavadi opis, razlaga programe) vnesemo za ukazom REM

Primer:

10 REM: program za izračun

20 REM: površine kroga

naziv	z. obr	dne	počip



30 PRINT: "vpiši velikost polmera"

40 INPUT : P

50 X = PI*3.14

60 PRINT: "površina kroga je"; X

Z ukazom RUN se program izvede in to od 30 -TE vrstice naprej. Vrstico, ki se začne z REM ukazom interpreter enostavno prekoči in išče naslednjo, ki se ne začne z tem ukazom. To se pogosto zgodi če imamo GO TO ... ali GO SUB ukaze.

Nič nebi bilo programske narobe če bi zgornjem primeru dodali vrstico s številko 70:

10 REM - program za izračun

20 REM - površine kroga

30 PRINT : "vpiši velikost polmera"

40 INPUT : P

50 X = PI*3.14

60 PRINT : "površina kroga je"; X

70 GO TO 10

Ukaz REM lahko uporabimo tudi tako:

10 INPUT A, B

20 PRINT A, B : REM izpis spremenljivk A in B.

4.2 Zgradba vrstice:

Vsaka vrstica je lahko dolga največ 72 znakov. Programske vrstice so lahko oštevilčene od 1 do 32767. Običajno jih ob vnašanju oštevilčimo 10, 20, da lahko pozneje po potrebi vrinemo dodatne vrstice.

Ena vrstica lahko vsebuje en ukazni stavek, ali pa več ločenih z dvopičjem ali znakom. Obstajajo ukazi, ki morajo biti prvi v vrstici, ali pa morajo stati sami v vrstici:

30 INPUT "vrsta, stolpec"; Y, X : Y = Y/2 : X = X/2

30 INPUT "vrsta, stolpec"; Y, X

40 Y = Y/2

50 X = X/2

100 SETUSR 5000: dalje se tekst ne izvede

(ukaz SETUSR mora stati sam v vrsti, sledi mu lahko le komentar).





Med posameznimi imeni in ukazi lahko zaradi preglednosti izpuščamo poljubno praznih mest.

Primer:

$$25X = 3\pi X/4 + 7.25\pi Y - .1$$

$$25 \quad X = 3\pi X/4 + 7.25\pi Y - .1$$

5. KONSTANTE, SPREMENLJIVKE

5.1 Konstante

Konstante so lahko številčne ali znakovne. Številke lahko predstavimo s plavajočo vejico ali eksponentni obliki, kot pozitivne in negativne vrednosti. Točnost je šest desetiških mest, približne meje pa od 10^{-38} do 10^{38} za pozitivna, oziroma od -10^{-38} do -10^{38} za negativna števila.

Primer številčnih konstant:

3.14159

4.5E10

6E-23

Niz znakov predstavljajo znakovne (character) konstante. Konstante so definirane kot poljuben niz ASCII znakov, omejenimi z enojnimi ali dvojnimi narekovaji.

Primeri znakovnih konstant:

"BASIC VTŠ MARIBOR" (dvojni narekovaji)

"A" (samo en znak)

'IME IN PRIIMEK' (enojni narekovaji)

'citirano "ime"' (dve vrsti narekovajev omogočajo tekst z narekovaji vred)

5.2 Spremenljivke

Spremenljivke so enako kot konstante številčne in znakovne. Številčne konstante so samo realne, tudi cela števila so kodirana v basicu v obliki s plavajočo vejico.

Imena spremenljivk so sestavljena iz enega do dveh znakov, pričnejo se s črko. Znakovne spremenljivke so označene z znakom \$ za imenom. Znakovne spremenljivke imajo lahko ista imena kot številčne v istem programu.



Primeri številčnih spremenljivk:

A
X
X1
A2
BB
...

10 REM izračuni faktoriele
20 PRINT "X=";
30 INPUT X
40 LET Z=1
50 FOR I = 1 TO X
60 LET Z = Z * I
70 NEXT I
80 PRINT "X="; X; "X1 = "; Z

Primeri znakovnih spremenljivk:

A\$
XX\$
X1\$
AZ\$

Razen tega, da se znakovne spremenljivke, kot številčne, ne smejo začeti z številko, je še ena omejitev.

Znakovno spremenljivka ne sme vsebovati besede, ki so rezervirane za programske ukaze (AT, TO, ON, IF ...).

5.3 Priredjanje vrednosti spremenljivkam

Spremenljivkam se priredijo (definirajo) vrednosti s prireditvenim stavkom LET, ukazom INPUT ter s kombinacijo stavkov READ in DATA.

Primeri:

LET Y = 4
X = 3.14
INPUT AA
C\$ = 'IME in PRIIMEK'
DATA 1, 2, 3
READ A, B, C
BS = (01011000000001111) (dvojiški zapis)



5.4 Indekuirane spremenljivke

Z obemi tipi spremenljivk lahko tvorimo vektorje ali dvodimenzionalna polja s stavkom DIM. Basic pozna torej spremenljivke z enim ali dvema indeksoma.

Primer:

DIM A (1,2)

sestavek tvori dvodimenzionalno polje realnih spremenljiv (matriko) dimenzijs 2 x 3:

A (0,0) A (0, 1) A (0, 2)
A (1,0) A (1, 1) A (1, 2)

Imena indeksiranih spremenljivk so lahko ista kot imena že obstoječih navadnih spremenljivk v istem programu.

6. OPERATORJI

6.1 Aritmetični operatorji

So znaki za računske operacije:

+ seštevanje
- odštevanje
x množenje
/ deljenje
! potenciranje

Vrstni red (hierarhija) računskih operacij je naslednja:

1. potenciranje
2. minus kot predznak
3. množenje in deljenje
4. seštevanje in odštevanje

Enakovredne računske operacije se vršijo v vrstnem redu od leve proti desni. Vrstni red računskih operacij lahko uredimo drugače s pomočjo okroglih oklepajev na običajni način, kakor smo v matematiki navajeni.





6.2 Logični operatorji

Logični operatorji so: AND, OR in NOT

Kadar so logični operatorji uporabljeni na enem ali dveh številih, takrat izvedejo želeno operacijo na posameznih bitih celih števil oziroma števila. Ker se izvajajo na celih številih, se v basicu najprej prevedejo števila, ki so vedno kodirana kot realna v celo obliko, po izvršeni operaciji pa spet nazaj v redno obliko.

Primeri:

$$A = (0011000011110000)$$

$$B = (1000011110000000)$$

$$\text{NOT } A = (1100111100001111)$$

$$A \text{ AND } B = (0000000010000000)$$

$$A \text{ OR } B = (1011011111110000)$$

Kadar pa so uporabljeni v logičnih izrazih kot test logičnega pogoja v stavku IF-THEN, pa imajo drugačen učinek. V tem primeru se logični izraz logično (ne aritmetično) ovrednoti glede na izpolnjenost pogoja (logično DA, ima številčno vrednost -1), ali njegovo neizpolnjenost (logično NE, s številčno vrednostjo 0).

Primer:

IF $B > 3$ AND $B < 20$ THEN GO TO 100

pogoj bo izpolnjen (skok na stavek 100), kadar bo A med vrednostmi 3 in 20 (izključno).

6.3 Relacijski operatorji

Se uporabljajo v pogojnih testih za primerjavo vrednosti konstant in spremenljivk. Operatorji so:

= enako

<> neenako

< manjše

> večje

<= manjše ali enako

>= večje ali enako



Operatorji se pogosto pojavljajo v testih v kombinaciji z logičnimi operatorji.

Primer:

IF C <> 0 OR (A > 256 AND B = 0) GO TO 200

6.4 Operatorji pri znakovnih spremenljivkah

Operatorje sestavlja spojitveni (concatenation) operator ('+') in relacijski operatorji. Operator '+' spoji dva niza znakov v enega novega. Relacijski operatorji, uporabljeni pri znakovnih spremenljivkah, delujejo po abecednem zaporedju, namesto številčni vrednosti. Če je nek niz znakov "manjši" od drugega, pomeni to, da bi se pojavil pred drugim, če bi ju uredili po abecednem redu. Pri primerjavi dveh nizov se vodeča prazna mesta ne upoštevajo. Če sta dva niza, ki ju primerjamo, različnih dolžin, se krajši niz dopolni z vodečimi praznimi mesti (blanks). Niz z dolžino nič (ničelni niz) se smatra kot popolnoma prazen in je "manjši" kot katerikoli drugi niz, razen niz, izpolnjen s samimi praznimi mesti, ki mu je "enak".

6.5 Hierarhija vseh operatorjev

Pri mešani uporabi vseh možnih vrst operatorjev je vrstni red izvrševanja operacij naslednji:

1. () izrazi v oklepajih
2. ! potenciranje
3. - minus kot predznak
4. */ množenje in deljenje
5. +- seštevanje in odštevanje
6. relacijski operatorji
7. NOT
8. AND logični operatorji
9. OR



7. TAKOJŠNJI UKAZI

Takojšnji ukazi so lahko uporabljeni samo v takojšnjem načinu, ne pa kot ukazi v programske stavkih. Ukazi niso basicove programske kode, temveč navodila basic interpreterju.

- CLEAR Vsem spremenljivkam se postavi vrednost na nič. Enak postopek se izvede tudi ob ukazu RUN
- CONT Služi za nadaljevanje programa, prekinjenega bodisi s programskim ukazom STOP ali tipko C/C ob čakanju vhodnega stavka INPUT. V prvem primeru se po ukazu CONT prične izvajati stavek, ki sledi stavku STOP, v drugem primeru pa se ponovno izvede prekinjeni INPUT stavek.
- EDIT EDIT (program od začetka)
 EDIT nl (vrstico nl)
 EDIT nl - n2 (vrstice od vključno nl do n2)
 Editiranje (urejanje) programske vrstic z editorjem EDITB.
- EXIT Izhod iz basica v monitor računalnika.
- LIST LIST (ves program)
 LIST nl (vrstico nl)
 LIST nl - n2 (vrstice od nl do n2)
 Izpis vsega ali dela programa na zaslon. Izpis na zaslonu lahko zaustavimo s tipko C/W ter nato nadaljujemo s pritiskom na katerokoli tipko, ali pa ga prekinemo s tipko C/C.
- LOAD LOAD (prvi program tipa B ali T)
 LOAD ime (program z danim imenom tipa B ali T)
 Nalaganje programa s traku. Programi tipa B so običajni programi v basicu, kakršni se zapišejo z ukazom SAVE, datoteke tipa T pa so sistemski, tovarniški programi, ki se ne dajo popravljati, listati ali ponovno shraniti (LIST, EDIT, SAVE ni možen). Sam postopek je enak monitorjevem LOAD ukazu.

NEW	Izbriše program, ki je trenutno v računalniku. Postopek se izvede tudi ob hladnem resetu basica.
RUN	RUN (program od začetka) RUN nl (program od vrstice nl)
	Prične izvajanje programa v basicu od začetka ali določene vrstice. Ob napaki v vrstici se program prekine in javi številka napačne vrstice in vrsta napake.
SAVE:	SAVE ime
	Shrani na trak program v basicu, ki je trenutno v računalniku. Program dobi podano ime in tip B. Postopek je enak kot pri monitorjevem ukazu SAVE. Pred zaključkom tega ukaza mora biti kasetni magnetofon priključen na računalnik v položaju za snemanje, trak mora teči.
PREND	Prikaže zasedenost spomina s programom in mejo še prostega prostora zanj (Program END) v desetiški obliki.
REND	REND naslov
	Nastavi novo končno mejo razpoložljivega spomina. Naslov je vtipkan kot štirištevilčno desetiško število (Ram END).
	Primer:
	REND 6000
	10 REND HEX ('E000').

8. PROGRAMSKI UKAZI

Programski ukazi so domena programskih, oštevilčenih vrstic. V splošnem pa se lahko skoraj vsi (odvisno od pomena) uporabijo tudi v takojšnjem načinu.

V opisu ukazov bodo uporabljeni določeni simboli in okrajšave, ki pomenijo:

n n1, n2, ni

štavilo

stevio pivo stevio, drago stevio, i vo stevio



str str1, str2, str3
niz znakov prvi niz, drugi niz, i-ti niz znakov
v V1, V2, Vi
 spremenljivka prva, druga, i-ta
expr expr1, expr2
 izraz izraz 1, izraz 2
ln številka vrstice
() v oklepajih datotek ukaza, ki ni nujen.

8.1 Prireditveni ukazi

DATA DATA nl (, n2, n3,...)
 DATA strl (, str2, str3,...)
 DATA -2.5E - 30, 0, 10
 DATA IVO, ANICA, PETER
 DATA "100", "1000", "10.000"

Ukaz določa informacijo (vrednost, podatke), ki se bo prečitala v programu s stavkom READ. Podatki se čitajo od leve proti desni. Vsak ukaz READ v programu ob svojem izvajanju prečita naslednji element iz stavka DATA. Po koncu seznama stavka DATA, prične s seznamom naslednjega stavka DATA. Ukaz DATA, mora stati sam v programske vrstici. Niz, ki vsebuje vejice ali prazna mesta, mora biti v stavku DATA omejen z narekovaji.

LET LET V = expr
 V = expr (nakazani LET)
 LET X = 3.14159
 Y = 2/5
 DAS = "PONEDELJEK"

Ukaz priredi novo vrednost spremenljivkam (glej točko 5.3).



READ READ v 1 (, v2, v3 ...)
 READ X, Y, CC

Stavek prečita podatke iz DATA stavka. Seznam spremenljiv "v" dobi vrednosti podatkov "n" ozziroma "str" iz stavka DATA. Vsak stavek READ prične čitati seznam od mesta, kjer ga je nehal čitati prejšnji stavek READ. Le s stavkom RESTORE se prenese čitanje spet na začetek. Pri poskusu čitanja preko meje seznama zadnjega DATA stavka se javi napaka.

RESTORE RESTORE

Ko se izvrši stavek RESTORE, prične naslednji stavek READ spet čitati podatke z začetka seznama prvega stavka DATA v programu.

Primer:

```
10 READ A, B, C
20 RESTORE
30 READ D, E
40 PRINT "A="; A, "B="; B, "C="; C.
50 PRINT "D="; D, "E="; E
60 DATA 1,2,3,4,5
```

RUN

Dobili bomo

A = 1, B = 2, C = 3

D = 1, E = 2

Ko ne bi bilo ukaza RESTORE, (oz. če izbrišemo vrstico 20 RESTORE).

Na zaslonu, po ukazu RUH, dobimo:

A = 1, B = 2, C = 3, D = 4, E = 5

8.2 Ukazi s spremembo poteka izvajanja

GOSUB GOSUB ln
 GOSUB 370

Stavek povrzoči izvajanje podprograma s številko prve vrstice ln. Po koncu podprograma se s stavkom RETURN vrne izvajanje zopet na stavek, ki sledi stavku GOSUB.

1. lin	2. lin	3. lin	4. lin	5. lin	6. lin	7. lin	8. lin	9. lin	10. lin



GO TO

GO TO 1n

GO TO 400

Brezpogojni skok na stavek s številko 1n. Stavek je lahko le zadnji v vrsti pri več stavkih v eni vrsti.

Primer

10 A = 1

20 B = Ø

30 C = B + A

40 PRINT C

50 A = A + 1

60 IF A = 10 GO TØ 100

70 GO TO 30

100 END

ON GOSUB ON exp GOSUB ln₁, ln₂ (, ln₃, ln₄, ...)
ON A GOSUB 10, 20, 30, 40

Pogojni klic podprogramov s pričetkom stavkov številk ln₁, ln₂ ... glede na celo vrednost + realna števila se ne zaokrožijo, temveč se decimalna vrednost odreže) izraza expr (1, 2, ...). V kolikor pade cela vrednost izraza izven mej seznama številk za GOSUB, se javi napaka.

V podanem primeru bi ob vrednosti A = 1, 2, 3 ali 4 programa nadaljeval s podprogramom, ki prične z vrstico s številko 10, 20, 30 ali 40.

Tudi ON GOSUB stavek mora stati na koncu programske vrste.

ON GO TO ON expr GO TO ln₁, ln₂ (, ln₃ ...)

Ukaz deluje enako kot ukaz ON GOSUB, le da povzroči pogojni skok na določene programske vrstice namesto klica podprograma.

Tudi ukaz ON GO TO mora stati na koncu programske vrstice.



Primer

```
10 REM primer uporabe "ON oz. GO TO ..."
20 INPUT A
30 ON A GO TO 100, 40, 80, 60
40 PRINT "A = 2"
50 GO TO 20
60 PRINT "A = 4"
70 GO TO 20
80 PRINT "A = 3"
90 END
100 PRINT "A = 1"
110 END
```

Razlaga: Ko bo numerična vrednost spremenljivke A enaka:

A = 1 se program veji na vrstico 100
A = 2 se program veji na vrstico 40
A = 3 se program veji na vrstico 80
A = 4 se program veji na vrstico 60
A > 4 računalnik sporoči napako

RETURN

RETURN

Mora stati na koncu vsakega podprograma. Stavek vrne potek izvajanja nazaj na stavek za mestom klica podprograma.

Tudi ta ukaz mora stati zadnji v programske vrsti.

8.3 IF - THEN - ELSE, pogojno izvajanje

IF GO TO IF expr GO TO ln

IF C = 0 GO TO 300

IF D < E AND F <> 10 GO TO 500

Izraz se logično izračuna, če je rezultat logični DA, se izvrši stavek za GO TO s številko ln, drugače pa se izvrši prvi stavek za stavkom IF GO TO (torej se ne izvrši nobena akcija).

5	6	7	8	9	10	11	12	13	14	15	16



IF THEN IF expr THEN ln
 IF expr THEN programski stavek
 IF A + B = 7 THEN 400
 IF A = B THEN PRINT "enak B"
 IF A = B THEN IF A = 0 GO TO 100

Ako stoji za THEN številka stavka ln, je učinek enak kot pri stavku IF GO TO. Ako pa stoji namesto številke stavka ln za THEN novi programski stavek, se ta v celoti izvede, kadar je vrednost izraza expr logično DA. Ta novi programski stavek je lahko tudi novi stavek IF THEN ali IF GO TO.

IF THEN IF expr THEN () ELSE ()
ELSE () je lahko ln ali programski stavek

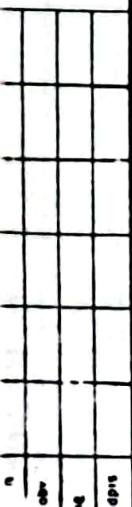
IF C/2 = 10 THEN GO TO 100 ELSE PRINT C

Ako je izraz expr logično DA, je učinek enak kakor pri stavku IF THEN, ako pa je izraz logično NE, se izvrši programski stavek, ki sledi ELSE, oziroma sledi skok na številko stavka ln, ki sledi ELSE.

V podanem primeru bi se v primeru, ko bi bila vrednost C polovico manjša od 10 prenesel potek programa na stavek s številko 100, v nasprotnem primeru pa bi se izpisala vrednost spremenljivke C (s stavkom PRINT C) in bi se nadaljevalo izvajanje s stavkom, ki sledi stavku IF THEN ELSE v programu.

Primer:

```
10 PRINT TAB (1); "A", TAB (17); "B", TAB (33);  
      "C"  
15 PRINT  
20 A = 1  
30 B = A ! 2  
40 C = A ! 3  
50 IF A = 10 THEN 90  
60 PRINT A, B, C  
70 A = A + 1  
80 GO TO 30  
90 END
```





8.4 Vhodno - izhodni stavki

INPUT INPUT ('str'); seznam spremenljivk
 INPUT A, B, C
 INPUT 'vtipkaj da ali ne', S\$
 INPUT A\$, C, D

Ukaz najprej izpiše niz znakov str, v kolikor obstaja, nato izpiše vprašaj in čaka na vtipkanje vrednosti ki jih bodo sprejele spremenljivke iz seznama na koncu stavka.

Več vtipkanih vrednosti ločimo med seboj z vejicami, lahko pa vpišemo tudi vsako v svojo vrsto (vsako vrednost zaključimo s CR).

Namesto novih vrednosti lahko odgovorimo na vprašaj s tipko C/C. Tedaj se program prekine, s stavkom INPUT ga lahko pričnemo nadaljevati zopet po ukazu CONT.

INPUT LINE INPUT LINE ime, znakovne spremenljivke

INPUT LINE A\$

INPUT LINE B\$ (5) (spremenljivka je lahko indeksirana)

Vsaka vtipkana vrsta se prečita in shrani v znakovno spremenljivko s podanim imenom (možna je samo ena). Vrsta je zaključena tudi v znakovni spremenljivki z znakom CR.



PRINT PRINT (v, str; ...),
 (str; v, ...);
 (expr, ...)
PRINT (povzroči samo CR LF)
PRINT "SPOROČILO" (izpiše samo tekst SPOROČILO)
PRINT 'vrednost S =', S
PRINT A, B; X; Y
PRINT "REZULTAT =" ; A/B³ (izraz)

Izpišejo se vrednosti spremenljivk, konstante, vrednosti iz računov izrazov. V kolikor so posamezni argumenti v seznamu za izpis ločeni z vejicami, se bodo izpisali vsak v svojem polju, ki je vsako dolgo 16 znakov (vsaka vrstica je razdeljena na 5 takšnih polj. Če pa so ločeni z podpičji, se bodo izpisali tesno drug ob drugem (znakovni nizi se izpišejo povsem drug ob drugem, števila pa imajo na začetku eno mesto rezervirano za negativni predznak, za koncem števila pa še eno prazno mesto).

PLOT PLOT X₁, Y₁ ; X₂, Y₂; X_n, Y_n

Ukaz postavlja točke na zaslonu na predpisano mesto X, Y. Zaslon predstavlja (semigrafiko) kartezični koordinatni sistem z izhodiščem v levem spodnjem kotu (X=0...79, Y=0-47). Najmanjša točka je četrtina polja karakterja.

UNPLOT UNPLOT X₁, Y₁ (; X₂, Y₂; ... X_n, Y_n)

Ukaz je obraten od PLOT in briše točko na predpisanim mestu.

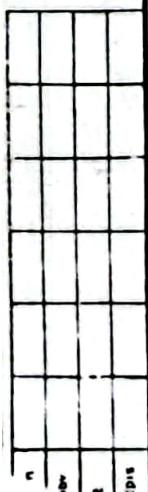
AT AT X, Y

Ukaz se uporablja smo v stavku PRINT, ter postavi kurzor na pozicijo X,Y na zaslonu kjer pomeni X vrstico in Y stolpec

Primer:

PRINT AT 3; 4

Pričetek v 3. vrstici, 4. stolpec





BEEP

BEEP T1, V1 ($;T2, V2; \dots Tn, Vn$)

Ukaz generira pisk ali več zaporednih piskov, kjer določa T trajanje (0-255), V pa višino tona (0-107)

tabela vrednosti ukazov:

trajanje tona

biti	7	6	5	4	3	2	1	0
čas	8s	4s	2s	1s	1/2s	1/4s	1/8s	1/16s

tonska lestvica

oktava	C	C	D	D	E	F	F	G	G	A	A	H
0	0	1	2	3	4	5	6	7	8	9	10	11
1	12	13	14	15	16	17	18	19	20	21	22	23

2

.

.

8 96 97 98 99 100 101 102 103 104 105 106 107

Pri tvorjenju konstante T moramo sešteti vrednosti vseh bitov ($b_0 = 1, b_1 = 2 \dots, b_7 = 128$), oziroma uporabiti šestnajstistički zapis konstante (HEX).

8.5 Zanka FOR - NEXT

FOR

FOR V = expr 1 TO expr 2 (STEP expr 3)

FOR I = 1 TO 500

FOR CC = 1.4x3 (X+7) TO 600-Y STEP 3

Označuje pričetek zanke FOR-NEXT. Podane so začetna, končna vrednost števca zanke in njegovo povečanje pri vsakem prehodu skozi zanko (expr 1, expr 2 in expr 3).

Števec (spremenljivka V) se postavi na začetno vrednost. Zanka (vsi stavki od stavka FOR do pripadajočega stavka NEXT) se izvrši enkrat, nato se pristeje števcu V vrednost expr 3, ki je lahko tudi negativna. V kolikor je vrednost V večja od končne vrednosti expr 2, sledi izhod iz zanke, prične se izvrševati stavek za stavkom NEXT, ki zaključuje zanko. V nasprotnem primeru pa prične nov cikel prehoda skozi

Vrednost števca V lahko





pa je spremenjati.

V kolikor ni podano posebej povečanje števca (STEP), se vzame zanj vrednost +1.

NEXT

NEXT V

Zaključuje zanko FOR-NEXT. Spremenljivka V mora biti ista kot tista v pripadajočem stavku FOR. Ukaz povzroči vrnitev programa na pripadajoči stavek FOR, povečanje števca V in izračun pogoja vrnitve iz zanke.

Primer:

```
10 FOR I = 1 TO 20
```

```
20 PRINT 'prehod'; I (izpiše 20 prehodov skozi zanko)
```

```
30 NEXT I
```

FOR-NEXT zanke so lahko vgnezdenne ene v drugi do poljubne globine, le da mora imeti vsaka svojo spremenljivko.

Primeri:

```
10 FOR I = 1 TO 100: REM zunanja zanka
```

```
20 A = I * 2
```

```
30 FOR J = 1 TO 200 STEP 10 : REM vgnezdena zanka
```

```
40 B = J + I + B
```

```
50 NEXT J : REM zaključek vgnezdena zanke
```

```
60 NEXT I : REM zaključek zunanje zanke
```

```
10 FOR I = 1 TO 1000 : NEXT I (samo zakasnitev v programu).
```

Z uporabo FOR/NEXT zanke se program iz prejšnjega primera zelo poenostavi.

```
10 PRINT TAB(2); "X", TAB(17); "X!2";
```

```
20 PRINT TAB(30); "X!3"
```

```
30 PRINT "... ....."
```

```
40 FOR X = 1 TO 10
```

```
50 PRINT TAB(1); X, TAB(17); X!2, TAB(30); X!3
```

```
60 Next X
```

```
70 END
```



8.6 Zaključni stavki

END END

Povzroči zaključek izvajanja programa.

Stoji za zadnjim programskim stavkom vendar to ni nujno. Ako stoji sredi programa, se program tam konča. Programa ni mogoče nato nadaljevati z ukazom CONT.

STOP STOP

Povzroči zaključek izvajanja programa ter izpis sporočila o prekinitvi:

STOP AT LINE ln

, kjer je ln številka STOP stavka, kjer se je končalo izvajanje. Program se lahko nadaljuje za stavkom STOP s takojšnjim ukazom CONT.

8.7 Ostali ukazi

DEF DEF FN ((V1)) = expr

S stavkom DEF definira uporabnik lastno funkcijo, ki obsega le en stavek. Možno je uporabiti le eno funkcijo naenkrat, vendar pa jo lahko v programu poljubnokrat nadomestimo z novo (na novo definirano s stavkom DEF). Funkcija lahko vsebuje samo en argument, spremenljivka, ki ga predstavlja v funkciji (vl) je lokalna glede na stavek DEF in nima drugje v programu nikakršnega pomena.

Ime funkcije je sestavljeno iz predpone "FN" in imena številčne spremenljivke (FNv). Funkcija se kliče s samim svojim imenom.

Uporaba funkcije deluje na ta način, da se vsakokrat, ko interpreter med izvajanjem programa naleti na njeno ime, izračuna vrednost po izrazu v stavku DEF ter imenu funkcije (spremenljivki FNv) priredi to vrednost. V vsakem primeru torej lahko uporabimo vrednost spremenljivke FNv kot pri vsaki drugi številni spremenljivki.

sl	sl	sl	eng	eng



Primer:

10 DEF FNXY (Z) = Z * Z + 2

•
•
•
•

200 B = 3

210 A = FNXY (B) + 10

220 PRINT A, FNXY (2)

S stavkom 10 se funkcija definira. Stavek 210 izračuna: $A = B * B + 2 = 3 * 3 + 2 = 11$, stavek 220 izpiše vrednosti 11 in 6 ($6 = 2 * 2 + 2$).

DIM DIM vl(i), (,v2(i),...)
 (i,j) (i,j)

Ukaz, ki rezervira v spominu prostor za polja spremenljivk, je natanko opisan v točki 5.4.

POKE POKE naslov, vsebina
 POKE 1000, 33
 POKE A, B
 POKE (HEX ("FEO0")), (10001111) - šestnajstički in dvojiški zapis

Vpis vsebine na nek naslov v spominu.

Naslov je v mejah od 0 do 32767, kadar je podan v desetiški obliki in v mejah od 0 do HEX ("FFFF") - desetiško 65535, kadar je podan v šestnajstiški obliki.

Vsebina je število v mejah od 0 do 255 = HEX ("FF").

REM REM komentar
 Ukaz je opisan v točki 4.2

CLS Ukaz je brez parametrov in izbriše zaslon ter postavi kurzor v zgornji levi kot zaslona.

INKEY**INKEY \$**

Je znakovna (karakter) funkcija, ki vrne vtipkan znak na tastaturi. V kolikor tastatura od zadnjega klicanja INKEY \$ funkcije ni bila pritisnjena, vrne funkcija znak SP (presledek).

9. VSEBOVANE FUNKCIJE

Basic vsebuje množico že pripravljenih funkcij, ki jih lahko uporabljamo v programu. Kličejo se z imenom ter argumentom v oklepaju. Argument pri nekaterih funkcijah nima pomena, pisati ga moramo (karkoli) le zaradi enotne oblike vseh funkcij. Znakovne funkcije, ki imajo imena znakovnih spremenljivk, vrnejo kot vrednost znakovni niz, vse ostale pa številčno vrednost. Mehanizem klica funkcij deluje tako: kjerkoli v programu se pojavi ime ter argument funkcije, se izračuna vrednost funkcije in se priredi imenu.

Primer:

$X=SQR(4)$, izračuna se vrednost kvadratnega korena iz $4=2$ X dobi vrednost 2.

$PRINT EXP(2)$, izračuna in izpiše se vrednost e na drugo potenco.

9.1 Aritmetične funkcije

EXP(X) Izvede se matematična operacija e na potenco X.
 $e=2.718281$, največji dovoljeni X = 87.3365, pri prekoračitvi vrednosti X se javi napaka.

LOG(X) Naravni logaritem od X.

Ostale logaritme dobimo lahko po enačbi:
 $\text{LOG od X base B} = \text{LOG}(X)/\text{LOG}(B)$

SQR(X) Kvadratni koren iz X

Pri negativnem X se javi napaka.

Primer:

```
10 REM izračun logaritmov vseh pozitivnih števil X,  
     in pozitivnih faz Y  
20 PRINT "BAZA=";  
30 INPUT Y  
40 PRINT "ARGUMENT,X=";  
50 INPUT X  
60 LET J=LOG(X)/LOG(Y)  
70 PRINT  
80 PRINT "LOG"; X; "BAZE"; Y; = ";J
```

9.2 Trigonometrične funkcije

ATN(X) Arcustanges od X, v radianih. Vrednost je med
-PI/2 in PI/2, kjer je PI/2=1.5708.

COS(X) Cosinus X, X je v radianih.

SIN(X) Sinus X, X je podan v radianih.

TAN(X) Tangens X, X je podan v radianih.

9.3 Znakovne funkcije

ASC(X\$) X\$ je izraz znakovnih nizov. Vrnjena vrednost funkcije
je ASCII številčna vrednost prvega znaka v nizu, ali
ničla, če je X\$ ničelni niz.

CHR\$(I) Vrnjena vrednost funkcije je en ASCII znak (znakovni
niz dolžine ena) z ekvivalentno številčno vrednostjo
I. I mora biti med vključno 0 in 255.

HEX(X\$) Funkcija pretvori šestnajstiški znakovni niz v njegov
desetiški ekvivalent.

Primer:

PRINT HEX("FF") Stavek bo izpisal desetiško število
255.



LEN(X\$)	Vrne število znakov v nizu X\$. V številu so vključeni vsi znaki, tudi presledki in kontrolni znaki.
MID\$(X\$, I)	Vrne delni znakovni niz MID\$ niza X\$ od njegovega I-tega znaka do konca
MID\$(X\$, I, J)	MID\$ je delni znakovni niz dolg J znakov niza X\$ od I-tega znaka dalje.
RIGHT\$(X\$, I)	RIGHT\$ je delni znakovni niz zadnjih (desnih) I znakov niza X\$. Če je I večji ali enak celotni dolžini X\$, je RIGHT\$ enak X\$.
STR\$(X)	Pretvori število X v znakovni ASCII niz, primeren za izpis (vodilno mesto za morebitni negativni predznak, ustrezno število znakov desetiške vrednosti, presledek).
VAL(X\$)	Pretvori znakovni niz X\$ z ASCII zapisom števila v vrednost števila VAL. V nizu X\$ je lahko poljubno vodilnih presledkov. Ako ni prvo mesto za presledki predznak, decimalna pika ali številka, je vrednost VAL = 0.
Primer: 10 A\$ = "A B C D E F G H I J K" 20 REM izpis prvih trez znakov 30 PRINT LEFT\$(A\$, 3) 40 PRINT 50 REM izpis zadnjih trez znakov 60 PRINT RIGHT \$ (A\$, 3) 70 REM izpis srednjih treh znakov 80 PRINT MID \$ (A\$, 5, 3) 90 PRINT 100 REM izpis števila znakov v spremenljivki A\$ 110 PRINT LEN (A\$) 120 PRINT 130 REM izpiši desetiški ekvivalent spremenljivke A\$ 	



140 PRINT HEX (A\$)
150 PRINT
160 REM izpis ASCII vrednosti prvega
170 REM znaka v spremenljivki A\$
180 REM v desetiški obliki
190 PRINT ASC (A\$)

9.4 Vhodno-izhodne funkcije

- PEEK (I) Vrne vsebino spomina na naslovu I
I mora biti v mejah od vključno 0 do 32767, če
je v desetiški obliki, oziroma do 65534, če je
podan šestnajstističko s funkcijo HEX
- POS(I) Vrne položaj konca trenutnega izpisa tekoče vrste
(število poslanih znakov na zaslon od zadnjega
CRLF). Začetna vrednost POS = 0.
- SPC(I) Povzroči izpis I presledkov v stavku PRINT, drugje
se ne more uporabiti.
- TAB(I) Povzroči tabulatorski pomik naprej do I-tega
stolpca. Uporablja se samo v PRINT stavku. Če je
izpisanih v vrstici že enako ali več kot I znakov,
ni učinka.

9.5 Ostale funkcije

- ABS(X) Vrne absolutno vrednost X.
- ADV(X) Vrne naslov spremenljivke X v BASIC RAM.
- INT(X) Vrne največje celo število, ki ni večje kot X
Primeri:
INT (-.02) = -1
INT (40) = 40
INT (0) = 0
INT (-4.2) = -5
INT (-12) = -12

član	u	dej	eng	podpis



PI Vrne vrednost PI

RND(X) Generira slučajnostne vrednosti med 0 in 1.

X = 0: Vsakokrat se generira novo slučajnostno število (običajna uporaba)

X < 0: Začetna vrednost (seme) za pričetek cikla generacije slučajnostnih vrednosti (kadar želimo rezultate)

X > 0: Vrne zadnjo generirano število.

ROOM(X) Vrne število prostih zlogov do konca razpoložljivega spomina za program (BASIC RAM). X nima pomena, je lahko karkoli.

10. NAPAKE

10.1 Javljanje napak

Napake se javljajo sproti, ko se odkrijejo. Prvič se lahko pojavi napaka že v sami obliki vrstice, ko vrstice sploh ni mogo kodirati. Te vrste napak se javijo takoj, ko vrstico vtipkamo ter jo zaključimo s CR. Drugič pa ugotavlja interpreter napake ob izvajanju programa, ob dešifriranju kodiranih ukazov v programske vrstici. Po ukazu RUN se torej program izvede do napake vrstice, tam pa se prekine in javi napaka. V takojšnjem načinu seveda sledi ob vtipkanju vrstice takoj njeno izvajanje in se torej javlja prav tako oboje vrst napak.

Napake v programske vrsticah sejavljajo s sporočilom:
ERROR NO. i IN LINE m,

kjer je i številka napake, m pa številka vrstice, kjer se je napaka pojavila.

S stavkom ON ERROR lahko preprečimo javljanje napak v času izvajanja, nemesto tega prevzame kontrolu nad njimi sam program. Pri tem uporabljamo še funkciji ERR, ERL in stavek RESUME (glej točko 10.3).

vi	odv

10.2 Seznam napak

- 1 slovnična napaka
- 2 nedovoljen izraz
- 3 izraz preveč zapleten
- 4 nepravilni oklepaji
- 5 nepravilen zaključek vrste
- 6 napaka v mešanem izrazu
- 7 .v znakovnem nizu manjka narekovaj
- 8 tip podatkov ne ustreza
- 9 nepravilna številka vrste
- 10 dosežena meja razpoložljivega spomina
- 11 argument izven mej 0-255
- 12 nepravilen tip spremenljivke
- 13 stavek ni prepoznan
- 14 prepovedan znak v vrsti
- 15 deljenje z nič
- 16 realno število izven možnih mej zapisa
- 17 število preveliko za pretvorbo v celo vrednost
- 18 argument prevelik
- 19 imaginaren kvadratni koren
- 20 napaka pri čitanju datoteke (LOAD)
- 21 nepravilen argument v stavku ON
- 22 argument izven mej 0-32767
- 23 dosežena meja polja
- 24 klic polja spremenljivk izven obsega
- 25 nepravilna uporaba indeksirane spremenljivke
- 26 RESUME brez ON ERROR
- 27 RETURN brez GOSUB
- 28 nepravilna številka stavka v GO TO, GOSUB
- 29 argument v LOG je negativen ali nič
- 30 FOR-NEXT zanka napačno vgnezdena
- 31 ni možnega nadaljevanja
- 32 nepravilna referenca funkcije
- 33 število preveliko pri pretvorbi
- 34 napaka pri pretvorbi celega števila pri INPUT
- 35 prepovedan simbol
- 36 manjka THEN v stavku IF
- 37 presežen seznam spremenljivk v stavku READ



10.3 Programska kontrola napak ob izvajanjju

Ukazi:

ON ERROR

ON ERROR ln

ON ERROR programski stavek

Po izvedbi stavka ON ERROR se ob napaki izvajanja programa ne bo več avtomatsko pojavilo sporočilo o njej, program se ne bo prekinil, temveč se bo nadaljeval na številki stavka "ln", oziroma se bo izvedel "programski stavek", kakor je zapisan za besedama ON ERROR.

RESUME

V kolikor se izvede po napaki ob že poprej izvedenem ukazu ON ERROR stavek RESUME, se bo program nadaljeval na mestu, kjer se je zgodila napaka - stavek z najdeno napako se bo še enkrat ponovil.

Običajna uporaba je pri vhodnih stavkih, kjer je včitan podatek povzročil napako (npr. deljenje z nič), po eventualnem lokalnem sporočilu pa želimo včitati drug, pravilen podatek.

Stavek RESUME se lahko uporabi le v zvezi s stawkom ON ERROR.

Funkcije:

ERR Vrne številko zadnje napake

ERL Vrne številko vrstice ob zadnji napaki

S podanimi ukazoma in funkcijama lahko sami v programu krojimo potek programa ob napakah.

Primeri:

A. 10 ON ERROR THEN 100

.

.

.

100 PRINT 'NAPAKA V VRSTI'; ERL, 'ŠTEVILKA'; ERR
110 END

B. 10 ON ERROR THEN PRINT 'NAPAKA'
C. 10 ON ERROR THEN 200
•
•
30 INPUT X
•
•
200 IF ERL = 30 THEN RESUME ELSE STOP

V primeru A se izpiše s programom ustvarjeno sporočilo o napaki, program se zaustavi.

V primeru B je kratko sporočilo kar v stavku ON ERROR.

V primeru C pa se ob napaki pri včitanju števila X to število ponovno včita (če je napaka v vrstici 30, se izvede stavek RESUME, ki povzroči povratek na stavek 30, kjer se je zgodila napaka), drugače pa se program prekine s stavkom STOP.

11. UPORABNIKOVE FUNKCIJE

Ukaz USR(X) omogoča uporabniku klicati iz basic programa podprogram v strojni kodi (assemblerju), ki ga vsebuje sistem (monitor), ali pa ga je izdelal sam.

Lastne podprograme lahko uporabnik ustvari na koncu razpoložljivega BASIC RAM spomina, zgornjo mejo programa v basicu pa lahko s stavkom REND postavi primerno navzdol, da se ne more pripetiti da bi programski stavki prekrili področje podprograma v strojni kodi. Klicanih podprogramov je lahko več, vendar je potrebno pred vsakim klicom s stavkom SETUSR določiti ustrezeni začetni naslov podprograma.



Podprogram ima lahko en vhodni in en izhodni parameter, ki se v dvojiški 16-bitni obliki prenese vanj, oziroma iz njega preko D registra. Podprogram se mora kot običajno zaključiti s strojno kodo RTS ter ne sme spremeniti sistemskega kazalca S. Vhodni in izhodni parameter basic ustrezno pretvori kot 16-bitno celo število v obliki dvojiškega komplementa v - oziroma iz - internega kodiranja realnih števil s plavajočo vejico.

Ukaz: USR(X) X je vhodni parameter podprograma in se prekodira v 16-bitno dvojiško število v register D ob klicu uporabnikovega podprograma.

USR vrnjena vrednost je prekodirana realna vrednost iz 16-bitnega dvojiškega števila v registru D ob vrnitvi iz uporabnikovega podprograma.

SETUSR naslov Naslov Naslov je štirimestno desetiško število naslova začetka uporabnikovega podprograma.

Vsek klic USR funkcije bo klical uporabnikov podprogram na tem naslovu vse do naslednje spremembe naslova z ukazom SETUSR.

Za ukazom SETUSR v vrsti ne sme biti nobenega drugega ukaza.

Primer:

10 A = HEX ('DC20')

20 SETUSR A: PRINT 'NADALJEVANJE VRSTE', USR (Ø)

Primer:

10 SETUSR 2A00 (naslov podprograma)

•

30 INPUT X (včitanje X)

•

40 PRINT USR(X) (X kot vhodni parameter, izpis izhodnega parametra - vrednost funkcije)
•

podprogram: ORG \$2A00

STD PARVH vhodni parameter iz D

•

•

LDD PARIZ izhodni parameter v D
RTS