Ciclul
„Calculatoare personale
și programarea lor"

FAMILIA DE CALCULATOARE PERSONALE ROMANEȘTI PRAE ȘI LIMBAJUL SĂU BASIC

ing. *Patrubany Nicolae*
ing. *Pop Baldi Nicolae*
mat. *Kiss Alexandru*
mat. *Socaciu Nicușor*
fiz. *Szász Delre*
I.T.C., Filiala Cluj-Napoca

0. CALCULATOARE PRAE CARACTERISTICI FUNCȚIONALE

0.1. Introducere

Cuvîntul PRAE (a se citi pre) este un prefix latin și înseamnă un început. Numele sugerează convingerea autorilor că familia de calculatoare care poartă acest nume reprezintă un preludiu, o deschidere către un mod nou, calitativ superior, de utilizare a calculatoarelor în România.

Această familie de calculatoare a fost elaborată în întregime la Filiala din Cluj-Napoca a Institutului de Cercetări pentru Tehnică de Calcul și este produs în serie la Fabrica de Memorii Timișoara.

Prototipul PRAE (vezi figura) a fost prezentat în public la Sesiunea din Bușteni a Cercului Utilizatorilor de Microcalculatoare și Terminale Programabile din noiembrie 1983.

Familia de microcalculatoare personale PRAE este clădită pe o unitate centrală de 8 biți (microprocesorul Z 80), ea acoperind o gamă largă de aplicații, începînd cu PRAE-T care este un bun de larg consum pînă la PRAE-M un calculator profesional.

PRAE-T este destinat în primul rînd tinerilor, pentru care el poate reprezenta un excelent mijloc de aprofundare a cunoștințelor în orice domeniu. El poate fi utilizat și ca mijloc de agrement elevat. PRAE-T folosește la maximum aparatele electronice existente în gospodăriile cetățenilor :

— televizorul servește ca dispozitiv de afișare iar casetofonul audio ca memorie externă pentru programe și date. Prin orientarea sa către bunurile de larg consum, acest calculator reprezintă un prim pas către o societate în care informația devine avuție națională, o societate în care fiecare individ va fi capabil să folosească calculatorul în vederea prestării unei munci calitativ superioare.

PRAE-L este un calculator ce poate fi utilizat cu succes în laboratoare de cercetare și învățămînt precum și în atelierele de proiectare.

PRAE-M (M specifică faptul că este varianta maximă a familiei) este un microcalculator profesional competitiv cu produse ca APPLE II, COMMODORE 64, M 118 etc.

PRAE-M este dotat cu o unitate duală de disc flexibil de 5'' 1/4 și cu un sistem de operare orientat pe disc. Astfel s-a creat o ambianță de calcul în care utilizatorul dispune de majoritatea limbajelor de programare de nivel înalt.

Familia de calculatoare PRAE prezintă o compatibilitate de sus în jos astfel încît aplicațiile elaborate pe membrii inferiori ai familiei să poată fi transpuse fără modificări pe membrii superiori.

Arhitectura hardware permite transformarea membrilor inferiori ai familiei în oricare membru superior prin simpla adăugare de elemente funcționale, servindu-se astfel interesele beneficiarilor.

Ca limbaj de programare de nivel înalt comun tuturor membrilor familiei folosește PRAE-BASIC, un interpretor deosebit de performant, compatibil cu standardurile internaționale (ANSI, CAER). Precizia de calcul este de 11 cifre semnificative.

Timpul de rulare pentru programul de test BM-7 acceptat internațional pentru calificarea interpretoarelor BASIC este de 53 s.

Prezenta lucrare cuprinde manualul de utilizare a componentelor software, monitorul și interpretorul BASIC V3.5 cum și anumite variabile sistem folositoare utilizatorului. Arhitectura hardware va face obiectul unei alte lucrări.

Cea mai mare parte a lucrării este dedicată învățării limbajului BASIC-PRAE.

0.2. Software rezident

În cei 16 k EPROM se află un monitor, interpretorul PRAE-BASIC 3.5, interfață tastatură, mașină de scris și casetofon, generator de caractere.

Monitorul are următoarele comenzi :

F (Fill)	FAdr1 Adr2 Valoare	(Intre Adr1 și Adr2 se umple cu valoarea Valoare)
G (Go)	GAdr1 Adr2 GAdr1 G Adr2	(Se folosește în timpul depanării unui program)
L (List)	L (CR) L (A, B, C, D, E, H, L, M, SP)	Listarea și/sau modificarea registrelor)
M (Move)	MAdr1 Adr2 Adr3	(Se mută conținutul dintre Adr1 și Adr2 începînd de la Adr3)
Q (Quit)	Q (CR)	(Salt în Basic cu distrugere program Basic)
R (Return)	R (CR)	(Return în Program Basic în locul întrerupt)
S (Substitute)	SAdr	(Substituie conținutul memoriei)

Adrese mai semnificative din monitor :

01EA Convertește o valoare binară pe 4 biți în hexa extern Valoarea binară se află în A, caracterul imprimabil se returnează în C. Nici un alt registru afectat.

01F2 Retur de car. Regiștri afectați : A, C.

0241 Convertește o valoare binară între 0—15 în hexa extern și imprimă. Valoarea se află în A. Regiștri afectați A, C.

024C Convertește o valoare binară între 0—1255 în hexa extern și imprimă. Valoarea în A, registru C afectat.

0247 Convertește o valoare binară între 0—65535 în format hexa extern și îl imprimă.

2D5 Intrare de la consolă ; caracter recepționat în registrul A.

1F9 Ieșire la consolă ; caracter de transmis în registrul C.

1E5 Tipărire spațiu ; BLK

25C Decodifică un caracter hexa extern în valoarea lui binară ; intrare în registrul A ; în cazul unui caracter eronat $C_y=1$.

Interpretorul BASIC primește spre interpretare programe de la tastatură sau de la periferic extern (casetă/bandă magnetică). Calculatorul funcționează și în regim de calculator de buzunar. Se pot executa comenzi, calcule imediate, ceea ce se identifică prin faptul că linia respectivă nu conține număr de ordine.

În cazul cînd o linie conține un număr de ordine între 1—65535 ea se structurează într-un program și se lansează în execuție prin comanda RUN.

Calculele în BASIC se fac în dublă lungime. Reprezentarea internă este flotantă. Se reprezintă pe 6 octeți în care 1 octet este exponentul binar (caracteristica) și 5 octeți mantisa.

Este permis lucrul cu masive de orice dimensiuni, se pot folosi funcții definite. Există variabile de tip șiruri etc.

Biblioteca BASIC conține în afara funcțiilor matematice și funcții referitoare la operații cu șiruri de caractere.

PRAE-BASIC are rutine pentru grafică, foarte rapide și ușor utilizabile.

Pentru a satisface nevoile celor mai diverși utilizatori amintim posibilitatea compunerii de melodii pe acest calculator avînd la dispoziție instrucțiunea BEEP.

Toate aceste posibilități oferite de BASIC vor fi detaliate la descrierea limbajului.

0.3. Software pe casetă. Extinderea interpretorului PRAE-BASIC

Pentru a nu se mări necesarul de memorie fixă (16 kocteți) anumite funcții ale interpretorului PRAE-BASIC se livrează pe casetă. Aceste funcții rulează în RAM.
Amintim două din cele mai importante module:

a) IOARRAY — permite salvarea și recitirea de pe casetofon a variabilelor masive de tip numeric și/ sau alfanumeric.

b) MATRIX — permite operații pe matrici cum ar fi: matricea inversă, matricea transpusă, determinantul unei matrici, adunarea, scăderea și înmulțirea matricilor, inițializarea, citirea și imprimarea matricilor.

0.4. Adrese de subprograme utile

3151	Sistem Console Input	(Caracter în A)
3154	Sistem Console Output	(Caracter în C)
3169	Sistem Console Status	(FF sau caracterul în A)
316C	Serial OUT	
317B	Serial INPUT	
31A8	Test EPROM	
3187	Scrierea pe casetă a conținutului de la 4080—40FF H	
318A	Citirea unui bloc (curent) de pe casetă la adresa 4080 H	
318D	Intrare tastatură proprie (reg A)	
3190	Ieșire pe ecran (reg C)	
319F	Console Status Serial	
31A2	Console Status tastatură proprie	
31A5	Desen miră (HL distrus)	
31A8	Test EPROM	
31AE	Test RAM (Revine după terminal afișarea adresei primei celule defecte).	

0.5. Celule de lucru, variabile sistem mai importante

BEG : 4004—4005 H	Adresă început ecran pentru rutinele de tipărire
BEGPLT : 401A—401B H	Adresă început ecran pentru rutinele de grafică
FIN : 4008—4009 H	Adresă sfîrșit ecran, coincide cu adresa sfîrșit RAM.
PNT : 4006—4007 H	Adresă curentă (pontor) de afișare text pe ecran.
FONDLI : 400AH	Starea SHIFT S tastatură
TIPCON : 400BH	Tip consolă (tastatură proprie sau serială)
SWITCH : 400CH	Starea după instrucțiunea SWITCH Basic (Valoare 0, 1 sau 2)
ULTCAR : 4019H	Ultimul caracter tastat în timpul execuției unui program BASIC (simulare INKEY).
STARES : 4020 H	Validarea sunetului la tastare (=0 — tastatură fără sunet =0 — tastatură cu sunet)
UPAD : 4010 H	Interfața serială
+0	Starea curentă a portului de ieșire
+1	Adresă port ieșire (80 H)
+2	Număr biți pe unitate de transfer (5, 6, 7, sau 8)
+3	Viteza (254_{10} pentru 300 baud 127_{10} pentru 600 baud 64_{10} pentru 1200 baud etc.)
+4	Paritatea b_0=0 transfer fără paritate b_0=1 transfer cu paritate b_1=0 paritate pară b_1=1 paritate impară b_2= bitul de paritate rezultat în cazul transferului cu paritate
+5	Număr biți STOP (1 sau 2)
DENSIT : 4018 H	Densitate de înregistrare pe casetp (10_{10} stand. DENSIT ≥ 6)
LONG : 403C—403D	Lungime bloc casetă (LONG ≤ 134_{10})

1. INTRODUCERE ÎN BASIC

BASIC-ul este un limbaj de programare conversațional folosind cuvinte cheie sugestive din limba engleză și notațiile matematice familiare, ceea ce îl recomandă ca unul din limbajele cele mai ușor asimilabile.

În același timp, pentru programatorii experimentați, limbajul pune la dispoziție tehnici avansate pentru rezolvarea eficientă a problemelor de orice natură.

1.1. Convenții de notații

În cuprinsul acestui manual sînt utilizate unele convenții de notații pentru explicarea detaliată a sintaxei instrucțiunilor și a elementelor compuse ale limbajului.

Pentru descrierea sintaxei instrucțiunilor se folosesc următoarele notații:

1. Elementele scrise cu litere mari (ex. LET, PRINT, IF etc.) trebuie să apară în instrucțiuni așa cum sînt notate, ele constituind vocabularul limbajului (cuvinte cheie);

2. Elementele scrise cu litere mici și închise între paranteze unghiulare indică elementele sintactice ale unei instrucțiuni sau comenzi.

Exemplu:

[LET] <variabilă> = <expresie>

3. Parantezele drepte indică posibilitatea prezenței sau absenței elementului cuprins între ele (element opțional);

4. Parantezele acolade indică posibilitatea alegerii uneia din variantele indicate, separate prin semnul "/".

Exemplu:

IF <expresie> THEN {<instr.> / <nr. linie>}
[ELSE <instr.> / <nr. linie>}]

1.2. Programul BASIC-PRAE

Un program constă din una sau mai multe „linii program". Din punct de vedere fizic, linia program este echivalentă cu linia terminal utilizator. „Linia program" este definită ca avînd structura

<nr. linie> <lista de instrucțiuni>

Fiecare linie program începe cu un număr de linie care va identifica linia în contextul întregului program și indică ordinea de execuție a instrucțiunilor. O linie program poate conține una sau mai multe instrucțiuni separate prin simbolul ':'. Lista de instrucțiuni a unei linii poate fi și vidă.

Fizic, linia program reprezintă totalitatea caracterelor cuprinse între promterul emis de sistem și terminatorul unei linii terminal (CR sau "RETUR DE CAR").

Lista de instrucțiuni are forma:

<instr.> : <instr.> : <instr.> ...

Fiecare instrucțiune începe cu un cuvînt cheie exprimat în limba engleză care indică tipul operației implicate de respectiva instrucțiune.

Capitolul 3 conține descrierea tuturor instrucționilor BASIC-PRAE.

1.3. Număr de linie

Fiecare linie program BASIC-PRAE începe cu un număr de linie.

Numărul de linie:

— indică ordinea de execuție a instrucțiunilor programului;
— permite schimbarea ordinii normale de execuție prin instrucțiuni de salt și cicluri;
— permite punerea la punct a unui program dînd posibilitatea schimbării oricărei linii fără afectarea restului programului;
— permite indicarea instrucțiunii eronate în timpul execuției programului.

Numărul de linie este un număr întreg format din 1 pînă la 5 cifre zecimale putînd avea valoarea cuprinsă între 1 și 65 529.

Programul se execută în ordinea crescătoare a numerelor de linie (ordine logică).

Remarcă : la prima scriere a unui program este recomandată numerotarea liniilor folosind ratia 5 sau 10 pentru a permite, în faza de punere la punct a programului, inserarea unor instrucțiuni între instrucțiunile existente ale programului.

Introducerea unei linii avînd un număr de linie deja existent în program, provoacă ștergerea liniei vechi și înlocuirea sa prin linia nou introdusă.

Introducerea unei linii în care numărul de linie este imediat urmat de caracterul CR (RETUR DE CAR) produce ștergerea liniei cu același număr de linie dacă aceasta a existat.

1.4. Linie simplă. Linie multiplă

Instrucțiune multiplă

O linie program poate conține una (linie simplă) sau mai multe (linie multiplă) instrucțiuni separate între ele prin caracterul ':'.

Remarcă : promterul componentei BASIC care se imprimă la începutul unei linii terminal în așteptarea introducerii unei noi linii terminal este simbolul '—'.

Exemplu :

```
10 LET A=0.1
50 FOR I=1 TO 5 : PRINT I ^2 : NEXT I
```

Remarcă : transferul controlului poate fi executat numai la prima instrucțiune a unei linii multiple.

Orice program poate fi salvat, listat, modificat sau executat prin comenzile proprii componentei.

1.5. Utilizarea caracterelor ' ' (spațiu) și TAB (CTRL I)

Caracterul ' ' (spațiu) poate fi utilizat singur sau în mod repetat oriunde în textul unei instrucțiuni pentru a servi estetica textului sursă.
De exemplu linia :

```
100 FORI = 1TO10 : PRINTI 2 : NEXTI
```

poate fi scrisă :

```
100 FOR I = 1 TO 10 : PRINT I ^2 : NEXT I
```

mărind evident posibilitatea remarcării rapide a elementelor componente ale liniei.

Caracterul TAB poate fi deasemeni folosit oriunde în textul unei instrucțiuni pentru a fi ușor citit. Textul liniei în care se folosește caracterul TAB apare identic la listare cu textul introdus.

Exemplu de utilizare :

```
10 FOR X=1 T0 10
20 FOR Y= T0 10
30 A(X, Y)=1/(X+Y-1)+A(X, Y)
40 NEXT Y
50 NEXT X
```

2. ELEMENTELE LIMBAJULUI BASIC-PRAE

2.1. Principii de realizare

Alfabetul limbajului BASIC-PRAE conține :
a) litere : A, B, . . ., Z
b) cifre : 1, 2, . . ., 9, 0
c) caractere speciale

În setul caracterelor speciale intră toate caracterele care posedă un cod imprimabil. Pornind de la caracterele de bază ale limbajului, folosind reguli sintactice precise (detaliat descrise la timpul potrivit) se obțin elementele compuse ale limbajului.

2.2. Elemente aritmetice

2.2.1. *Constante numerice*

În limbajul BASIC-PRAE toate numerele sînt tratate ca numerele zecimale conform notației matematice uzuale.

Pentru numere este folosită denumirea de constantă numerică, pornind de la faptul că acestea pentru programul în sine au tot timpul aceeași valoare.

O constantă numerică este un șir de cifre zecimale a cărui valoare reprezintă o valoare numerică (în sens obișnuit matematic). Valoarea constantelor numerice acceptate de limbaj trebuie să fie cuprinsă în intervalul real (10^—38, 10^+38).

În limbaj se recunosc trei formate de exprimare a constantelor numerice:

— format întreg 123

— format real 123.456

— format exponențial (cu puterile lui 10). 12.3E6

Constantele pot fi reprezentate în oricare din cele trei formate.

Sistemul BASIC consideră constantele ca fiind reale reprezentate în format flotant.

Dacă constantele sînt scrise cu caracterul & în fața, ele sînt constante hexazecimale (&04AC).

2.2.2. *Variabile simple*

Variabila este un simbol algebric reprezentînd un număr.

Numele de variabilă este format dintr-o literă urmată de oricite caractere alfanumerice, dintre care numai primele două sînt luate în considerare.

Valoarea variabilei este reprezentată pe 6 octeți în flotant.

De exemplu:

CORECT	INCORECT
I	2A
A8	2B
X	12

Unei variabile I se poate atribui o valoare prin:

— instrucțiunea LET;

— instrucțiunea INPUT sau READ.

Valoarea unei variabile rămîne constantă pe intervalul cuprins între două atribuiri.

În momentul lansării în execuție a unui program, toate variabilele acestuia sînt puse la 0.

2.2.3. *Variabile indexate*

Pentru prelucrarea listelor, tabelelor sau matricilor limbajul posedă un element special numit „variabilă indexată". BASIC-PRAE permite prelucrarea variabilelor indexate avînd unul, doi sau pînă la 255 indici.

Numele unei variabile indexate este format dintr-un nume admis de variabila simplă urmat de una sau două expresii de indici în paranteze.

Exemplu:

A(I), B(I, J), C (I+K, J+K)

Astfel o listă de 6 elemente căreia i se atribuie numele A poate fi reprezentată prin variabila indexată A (I) unde I poate lua valorile 0, 1, 2, . . ., 5.

A(0), A(1), A(2), A(3), A(4), A(5)

Prin această convenție fiecare element al listei poate fi reprezentat printr-o variabilă indexată avînd indicele egal cu ordinul elementului în listă. O matrice (tablou) bidimensională cu numele B se poate defini și reprezenta grafic astfel :

B (0, 0), B (0, 1), . . ., B (0, N)
B (1, 0), B (1, 1), . . ., B (1, N)
B (2, 0), B (2 ,1), . . ., B (2, N)
B (M, 0), B (M, 1), . . ., B (M, N)

Limitele între care pot varia indicii unei variabile indexate pot fi declarate într-un program :

— implicit, folosind prima reprezentare legală a variabilei indexate ;
— explicit, prin instrucțiunea de declarare DIM.

Într-un program este posibilă definirea prin același nume a unei variabile simple și a unei variabile indexate.

Exemplu :

A — variabilă simplă reală
A(X) — variabilă indexată reală

2.2.4. *Expresii numerice*

Expresia numerică este definită ca un grup de simboluri, elemente numerice ale limbajului, care poate fi evaluat. Expresiile numerice sînt compuse din constante numerice, variabile simple, variabile indexate sau combinații ale acestora separate prin operatori aritmetici, operatori de relație sau operatori logici.

2.2.4.1. *Expresii aritmetice*

Expresia aritmetică este compusă din elemente numerice (constante, variabile simple sau indexate), separate între ele prin operatori aritmetici.

Reprezentarea unei expresii aritmetice este similară notației matematice uzuale.

Exemplu :

A+0.5*B(I, J)/3

Operatorii aritmetici admiși sînt :

Operator	Exemplu	Semnificație
+	A+B	adună A cu B
—	A—B	scade B din A
*	A×B	înmulțește A cu B
/	A/B	împarte A prin B
^	A^B	calculează A la puterea B
+	+A	Operația unară +
—	—A	Operație unară —

Dacă într-o expresie aritmetică apar mai mulți operatori, evaluarea expresiei se va face conform priorității matematice uzuale a acestor operatori.

Într-o expresie oarecare dată, ordinea de execuție (prioritatea) este următoarea :

1) evaluarea expresiilor cuprinse între paranteze
2) în absența parantezelor ordinea este :
   a) operatori unari
   b) ridicarea la putere
   c) înmulțirea și împărțirea
   d) adunarea și scăderea
3) între operatori cu același nivel de prioritate, relația de ordine este indicată prin regula de la stînga la dreapta.

De exemplu în expresia:

A=B*((C ^2+4)/X)

Ordinea de evaluare este :

P1—C ^2
P2 — rezultat P1 + 4
P3 — rezultat P2/X
P4 — rezultat P3•B
P5 — rezultat P4 se atribuie lui A

2.2.4.2. *Expresii de relație*

Expresia de relație definește prin intermediul operatorilor de relație legătura între două entități.

În forma cea mai simplă, expresia de relație este compusă din două expresii aritmetice separate între ele printr-un operator de relație.

Operatorii de relație admiși sînt :

Simbol matematic	Simbol BASIC-PRAE	Exemplu	Semnificație
=	=	A=B	A egal cu B
<	<	A<B	A mai mic decît B
>	>	A>B	A mai mare decît B
<=	<=	A<=B	A mai mic/egal cu B
>=	>=	A>=B	A mai mare/egal cu B
≠	<>	A<>B	A diferit de B

Evaluarea expresiei de relație se execută prin :

— evaluarea expresiilor aritmetice componente,
— evaluarea relației dintre valorile obținute.

Valoarea unei expresii de relație este o valoare logică de adevăr („ADEVĂRAT" sau „FALS").

Expresia de relație redusă la o expresie aritmetică exprimă în fapt relația dintre valoarea expresiei aritmetice și valoarea 0, și are valoarea însăși a expresiei aritmetice.

2.2.4.3. *Expresia logică*

Expresia logică este compusă din una sau mai multe expresii de relație separate între ele prin operatori logici.

Operatorii logici simbolizează operațiile logice din logica matematică obișnuită. Operatorii logici admiși sînt :

Simbol	*Exemplu*	*Semnificație*
NOT	NOT L	operația logică „NOT"
AND	L1 AND L2	operația logică „ȘI"
YOR	L1 OR L2	operație logică „SAU"

Tabelele de adevăr ale operațiilor sînt cele din logica matematică :

L	NOT L
A	F
F	A

L1	L2	L1 AND L2
A	A	A
A	F	F
F	A	F
F	F	F

L1	L2	L1 OR L2
A	A	A
A	F	A
F	A	A
F	F	F

Operația se execută bit cu bit dacă operanzii sînt în intervalul 0, 65535, altfel funcția este refuzată cu mesajul de eroare corespunzător.

Ordinea de evaluare a unei expresii logice este :

— evaluarea expresiilor de relație,

— evaluarea operațiilor logice în ordinea :

NOT
AND
OR

— între operații cu același nivel ierarhic evaluarea se face de la stînga la dreapta.

2.2.5. *Funcții matematice standard*

Multe din funcțiile matematice comune (sinus, funcția putere, funcția logaritmică) pot fi executate în sistemul BASIC prin intermediul unor funcții aritmetice speciale, recunoscute în limbajul BASIC-PRAE.

De exemplu, pentru calcularea valorii sin (75 de grade) se creează o expresie simbolică formată din numele simbolic al funcției respective (SIN) urmat de valoarea parametrului în radiani închis între paranteze :

SIN (75*PI/180)

Este prezentat în continuare tabelul funcțiilor matematice recunoscute ca elemente ale limbajului BASIC-PRAE :

Notație	Simbol	Semnificație matematică
$\|x\|$	ABS (X)	Valoare absolută
sign x	SGN (X)	Generează : 1 dacă x>0 —1 dacă x<0 0 dacă x=0
[x]	INT (X)	Parte întreagă din x
cos (x)	COS (X)	Cosinus de x (radiani)
sin (x)	SIN (X)	Sinus de x (radiani)
tg (x)	TAN (X)	Generează o valoare reală egală cu valoarea funcției tangentă pentru argumentul x dat în radiani
arctg (x)	ATN (X)	Arctangenta lui x
$\sqrt{x}$	SQR (X)	Rădăcină pătrată din x
e^x	EXP (X)	Calculează e ^x
ln x	LOG (X)	Calculează logaritmul natural din x

2.2.6. *Funcții aritmetice definite*

În practica de programare se observă adeseori necesitatea executării unei secvențe de instrucțiuni sau a unei expresii în mai multe puncte ale programului.

Limbajul BASIC-PRAE pune la dispoziția utilizatorilor săi posibilitatea definirii unice a unei astfel de expresii sau secvențe de instrucțiuni și simplul apel al acestora în diferite puncte ale programului.

Aceasta înseamnă că limbajul permite utilizatorului să definească funcțiile sale proprii și să apeleze apoi aceste funcții în același mod în care se apelează funcțiile aritmetice standard.

Numele unei funcții utilizator este format din maximum 4 caractere :

— primele două caractere standard egale cu „FN"

— un nume de variabilă corect (o literă urmată de maximum 1 caracter alfanumeric).

Exemplu :

CORECT	*INCORECT*
FNI	FN2
FNA8	FN1B

Definiția unei funcții utilizator se introduce printr-o instrucțiune DEFFN.

Apelul unei funcții se realizează prin introducerea numelui de funcție urmat de parametrul (parametrii) de apel.

La apelul unei funcții se face o corespondență strictă între numărul și tipul parametrilor formali și numărul și tipul parametrilor de apel (actuali).

Conform structurii definiției, funcțiile definite utilizator se impart în două categorii : de relație sau logică)

— funcție simplu-definită, a cărei definiție constă dintr-o expresie numerică (aritmetică,

— funcție multilinie, a cărei definiție constă dintr-o secvență de instrucțiuni cuprinse între instrucțiunile DEFFN și FNEND.

Pentru înțelegerea mai detaliată, a se vedea paragraful „Instrucțiunea DEFFN".

2.3. Elemente nenumerice

Limbajul BASIC-PRAE permite manipularea unor informații numerice sub forma șirurilor de caractere.

Prin șir de caractere se înțelege o secvență de caractere BASIC-PRAE admise, privită ca o unitate.

Un șir de caractere poate conține litere, cifre sau caractere speciale. Lungimea unui șir de caractere se definește ca fiind numărul de caractere BASIC-PRAE componente.

În cele ce urmează vom numi „șir vid" șirul de caractere avînd lungimea 0.

Pe mulțimea șirurilor de caractere se pot defini operații ca :

— adunarea (concatenarea) a două șiruri
— extragerea unui subșir dintr-un șir dat
— modificarea elementelor componente ale unui șir etc.

Toate elementele numerice BASIC-PRAE au corespondente în setul elementelor nenumerice :

Exemplu :

constante numerice — constante șir
variabile numerice simple — variabile simple șir
masive numerice — masive șir
funcții standard numerice — funcții standard șir.

2.3.1. *Constante șir de caractere*

Prin constanta șir de caractere se înțelege orice șir închis între simbolurile '"' (ghilimele).

O singură excepție de la acestă definiție există și apare în cazul constantelor șir de caractere dispuse într-o instrucțiune DATA (a se vedea paragraful „Instrucțiunea DATA").

Exemplu :

"123-45"
"LET'S BEGIN"

2.3.2. *Variabile simple de tip șir*

Variabila simplă de tip șir este un simbol reprezentînd un șir de caractere.

Numele unei variabile simple șir este format din :
— nume variabilă (litera urmată optional de maximum 5 caractere alfanumerice)
— caracterul „$"

Exemplu :

Variabile simple numerice	*Variabile simple șir*
A	A $
B2	B2 $
M	M $

Într-un program pot exista variabile simple numerice și variabile simple șir cu același nume.

Exemple :

A, A$

Ca și în cazul variabilelor numerice, unei variabile simple de tip șir i se poate atribui o valoare prin instrucțiunile LET, INPUT, READ.

Variabilele de tip șir sînt inițializate la execuția unui program cu valori egale cu „șirul vid".

2.3.3. *Variabile indexate de tip șir*

BASIC-PRAE permite definirea variabilelor indexate de tip șir pentru rezolvarea prelucrării listelor și masivelor de șiruri de caractere.

Variabilă indexată de tip șir este un simbol care constă din :

— un nume corect de variabilă
— caracterul „$"
— unul sau pînă la 255 indici între paranteze

Exemple :

A$(5)
A1$(I, J)

De reținut că expresiile de indici nu pot fi decît expresii aritmetice.

Limitele între care pot varia indicii unei variabile indexate de tip șir pot fi declarate în program :

— implicit prin prima referință
— implicit prin instrucțiunile de declarare DIM

Este posibilă utilizarea simultană a unei variabile simple șir și a unei variabile indexate șir cu același nume.

Exemplu :

A$ și A$(10)

De asemenea este permisă utilizarea simultană a aceluiași nume pentru variabilele numerice și variabilele șir.

Exemplu :

A, A$, A(I), A$(I)

2.3.4. *Expresii nenumerice*

Expresia șir este definită ca un grup de elemente nenumerice care poate fi evaluat.

Expresiile șir sînt compuse din constante șir, variabile simple sau indexate de tip șir sau conbinații ale acestora separate prin operatori :

— operatorul de concatenare (simbol +)
— operatori de relație (expresie condiție șir)

Operatorul de concatenare marchează operația prin care din două șiruri date se poate obține un nou șir prin alipirea șirurilor date.

Operatorul de concatenare se notează prin simbolul matematic „+", El operează numai asupra șirurilor și nu este comutativ.

Exemplu :

"ABC"+"123" produce șirul "ABC123"

Operatorii de relație operează asupra șirurilor cu respectarea ordinii alfabetice. Compararea a două șiruri se face caracter cu caracter de la stînga la dreapta.

Operatorii de relație permiși și interpretarea lor figurează în tabelul următor :

Operator	*Exemplu*	*Semnificație*
=	A$=B$	Șirurile A$ și B$ sînt identice
<	A$<B$	A$ mai mic alfabetic ca B$ sau A$ subșir al lui B$
>	A$>B$	A$ mai mare alfabetic ca B$ sau B$ subșir stîng al lui A$.

Operator	Exemplu	Semnificație
<=	A$<=B$	A$ mai mic sau egal alfabetic cu B$
>=	A$>=B$	A$ mai mare sau egal alfabetic cu B$
<>	A$<>B$	A$ diferit de B$

Expresia de relație redusă la un singur element marchează în fapt relația între elementul și "șir vid".

2.3.5. Funcții standard pe șiruri de caractere

BASIC-PRAE recunoaște ca și în cazul funcțiilor matematice standard, un set de funcții asupra șirurilor de caractere.

Apelul unei astfel de funcții se poate face în orice punct al unui program prin indicarea numelui simbolic al funcției și a parametrului sau parametrilor de apel.

Funcțiile standard pe șirurile de caractere acceptate sînt :

Cod apel	Semnificație
LEFT$(A$,N)	Extrage subșirul sting de lungime N al șirului A$. LEFT$ ("ABCD",2) produce "AB".
RIGHT$(A$,N)	Extrage subșirul drept de lungime N din șirul A$. RIGHT$ ("ABCD",2) produce "CD".
MID$(A$,N1,L)	Extrage din șirul A$ un subșir de lungime L începînd cu al N1-lea caracter. MID$ ("ABCD",2,2) produce "BC".
LEN(A$)	Generează o valoare întreagă egală cu lungimea șirului A$. LEN ("ABCD") produce 4.
CHR$(N)	Produce un șir de lungime 1 care conține codul ASCII de valoare N. CHR(42) produce "*".
ASC(A$)	Generează valoarea zecimală întreagă a codului ASCII a primului caracter al șirului A$. ASC("ABC") generează 65.
INSTR(A$,B$,S,L)	Generează o valoare întreagă ca rezultat al căutării subșirului B$ de lungime L în șirul A$ începînd cu caracterul al S-lea din A$. Dacă B$ nu se află în A$ se produce valoarea întreagă 0. Dacă B$ se află în A$ produce valoarea întreagă egală cu indicele caracterului din șirul A$ începînd cu care cele două șiruri coincid Instr ("ABCDEF", "X",2) produce 0 Instr ("ABCDEF", "EF",2) produce 5. Prin convenție "șirul vid" este subșir sting al oricărui șir.
SPC(L%)	Generează un șir de blancuri de lungime L%.
STR$(N)	Generează un șir de caractere numerice reprezentînd valoarea de editare a numărului N. STR$(1234567) produce "1234567".
VAL(A$)	Generează valoarea flotantă a șirului de caractere numerice A$. VAL(„123") generează valoarea în reprezentare a numărului 123.

2.3.6. Funcțiile definite pe șiruri de caractere

Limbajul BASIC-PRAE pune la dispoziția utilizatorilor posibilitatea definirii unor funcții proprii de tip șir de caractere.

Indicația de tip se marchează prin introducerea caracterului "$" imediat după numele funcției.

Definiția și apelul funcțiilor definite de tip șir respectă aceleași reguli ca cele de tip numeric. Funcția definită de tip șir poate fi simplă sau multilinie. Într-un program este permisă utilizarea simultană a unei funcții numerice și a unei funcții de tip șir cu același nume.

Exemplu 6

FNA, FNA, FNA$

2.3.7. *Funcții grafice*

Interpretorul BASIC-PRAE recunoaște 6 funcții grafice pentru a face posibilă desenarea pe ecran. Ecranul este considerat ca fiind compus din 256 linii și 256 coloane (64*1 024 puncte).

Pentru stabilirea fondului ecranului există o directivă : SWITCH valoare

Dacă valoarea este 0, ecranul are fond alb ; dacă are valoarea 1, fondul este negru ; dacă valoare are valoarea 2 se realizează o intrare-ieșire serială.

Funcțiile grafice sînt :

Cod apel	*Semnificație*
CIRCLE (x, y, r)	desenează un cerc avînd centrul în punctul (x,y) de rază r.
CIRCLEC (x, y, r)	șterge de pe ecran cercul cu centrul (x,) și de rază r
PLOT (x, y)	desenează un punct pe ecran de coordonare (x,y)
PLOTC (x, y)	șterge punctul de coordonate (x,y)
DRAW (x, y)	leagă cu o dreaptă ultimul punct desenat pe ecran cu punctul de coordonate (x, y)
DRAWC (x, y)	șterge dreapta desenată
x, y, r	sînt expresii numerice, cu valori între 0 și 255.

3. INSTRUCȚIUNI BASIC-PRAE

În acest capitol este dată o descriere detaliată a instrucțiunilor admise în limbajul BASIC-PRAE.

Sînt folosite următoarele convenții și notații :

<notație>	Semnificație
<nr. linie>	Orice număr de linie admis
<instr.>	Orice instrucțiune admisă
<expr.>	Orice expresie admisă indiferent de tip (numerică sau șir)
<expr. num.>	Orice expresie numerică admisă (aritmetica, de relație, logica)
<expr. șir>	Orice expresie șir admisă
<expr. aritm.>	Orice expresie aritmetică admisă
<expr. rel.>	Orice expresie de relație admisă
<expr. log.>	Orice expresie logică admisă
<var.>	Orice variabilă admisă indiferent de tip (simplă, indexată, întregă, șir, reală)
<mesaj>	Lanț de caractere admis ca și comentariu
<cond.>	Orice expresie logică admisă, care se poate reduce la o expresie de relație sau aritmetică
<const.>	Orice constantă numerică sau șir admisă
<șir>	Orice constantă șir admisă
<var. num.>	Orice variabilă numerică admisă indiferent de tip (întreg sau real).

3.1. Instrucțiunea REM

Practica programării arată că un program bine comentat este cu mult mai ușor de testat și pus la zi decît unul necomentat.

În plus, un program bine comentat poate fi ușor înțeles și utilizat de orice alt programator decît autorul.

De remarcat că textul comentariului poate conține orice caractere, acesta fiind în întregime ignorat de tratare.

De reținut că un comentariu pe linie poate fi introdus numai după ultima instrucțiune a liniei ; în caz contrar, textul instrucțiunii este considerat ca făcînd parte din textul comentariului.

Instrucțiunea REM are formatul general :

REM [ARK] <mesaj>

unde prin mesaj se înțelege orice șir de caractere care va fi ignorat la tratare.

Exemplu:

```
10 REM LINIE DE COMENTARIU
```

Pe o linie multiplă, instrucțiunea REM trebuie să fie ultima instrucțiune a liniei.

Exemplu:

```
10 L=2*PI*R : REM LUNGIME CERC
10 L=2*PI*R : 'LUNGIME CERC
```

Linia:

```
100 REM LUNGIME CERC : L=2*PI*R
```

va fi tratată în întregime ca o linie comentariu.

3.2. Instrucțiunea LET

Instrucțiunea LET permite atribuirea la o variabilă simplă sau indexată a unei valori numerice sau șir.

Formatul general al instrucțiunii este:

[LET] <var>=<expr>

Execuția acestei instrucțiuni constă în evaluarea expresiei din dreapta semnului "=" și atribuirea valorii astfel determinate variabilei marcate în stînga semnului "=".

Instrucțiunea LET poate fi plasată oriunde în cadrul unei linii multiple.

Exemplu:

```
100 A=SQR (B^2+C^2) : A1=SQR (B^2-C^2)
```

3.3. Instrucțiunea DIM

Limabjul BASIC-PRAE permite utilizarea variabilelor indexate punînd la dispoziția utilizatorilor mijloace de prelucrare a listelor și matricilor.

În BASIC-PRAE se pot trata variabile indexate cu unul, doi sau pînă la 255 indici.

Într-un program declararea masivelor se poate realiza pe două căi:

— declararea explicită prin instrucțiunea DIM

— declararea implicită prin prima referință la un element de masiv.

La declararea implicită numărul de dimensiuni a masivului declarat este egal cu numărul indicilor variabilei indexate, fiecare dimensiune fiind egală cu 10. Dacă masivul A este declarat implicit printr-o referință de forma:

A(<expr.num.>)

atunci va avea o singură dimensiune egală cu 10. Dacă declarația implicită se realizează prin:

A(<expr.num.1>, <expr.num.2>)

atunci masivul A va fi un masiv bidimensional avînd dimensiunile (10,10).

Instrucțiunea DIM servește la declararea explicită a dimensiunilor unui masiv.

Formatul general:

DIM <var> (<dim1>, <dim2>), <var> (<dim1>, <dim2>)

unde:

var.	reprezintă numele masivului și poate fi orice nume de variabilă indexată admis în limbaj.
dim1 și dim2	reprezintă dimensiunile masivului și pot fi orice expresii numerice admise de limbaj. Numărul dimensiunilor este limitat la 255.

Exemplu:

```
10 N=20 ; M=15
20 DIM A(25), B(N,N). C (10,M)
```

Prin execuţia liniei 10 se declară masivul A avînd dimensiunea (25), masivul bidimensional real B avînd dimensiunile (20, 20) şi masivul bidimensional şir C avînd dimensiunile (10, 15).

Posibilitatea declarării unui masiv cu dimensiuni variabile măreşte mult gama utilizări acestora permiţînd ca la execuţii diferite a unui program spaţiul alocat masivului să aibă lungimi diferite.

Fie programul :

```
10 INPUT N
20 DIM A (N, N), B(N)
100 END
```

Se observă că dimensiunile masivelor A şi B şi deci spaţiul alocat este diferit la execuţii diferite fiind indicate de utilizator prin execuţia instrucţiunii din linia 10.

Declararea explicită a masivelor efectuîndu-se la execuţia instrucţiunii DIM rezultă că aceasta trebuie plasată logic în program înainte de prima referinţă la masivele conţinute, adică înaintea alocării masivului ; în caz contrar, declararea explicită este tratată ca o redimensionare.

De exemplu, fie liniile program :

```
10 A(I+1)=I+1
20 DIM A(100)
```

La execuţia liniei 10 masivul A cu dimensiunea implicită (10) este alocat ; execuţia liniei 20 produce abandonarea execuţiei programului cu indicarea mesajului de redimensionare incorectă.

Se impune deci :

```
10 DIM A(100)
20 A(I+1)=I+1
```

prin care A este declarat cu dimensiunea 100.

Rezultă că este indicată plasarea instrucţiunilor DIM la începutul programelor (ordine logică).

3.4. Instrucţiunea DATA

Instrucţiunea DATA utilizată împreună cu instrucţiunea READ serveşte la introducerea datelor în timpul execuţiei programului.

De remarcat că prin acest cuplu introducerea datelor se face fără intervenţia utilizatorului, fiind dispuse în memorie în blocuri de date odată cu introducerea programului.

Formatul general al instrucţiunii :

DATA <const.>, <const.> ...

Exemplu :

DATA BASIC, 'BASIC'

Instrucţiunile DATA pot fi introduse în orice loc într-un program cu condiţia respectării ordinii crescătoare a numărului de linie în funcţie de care se construieşte şi se exploatează blocul de date al programului.

Practica recomandă gruparea instrucţiunilor DATA la sfîrşitul programului pentru verificarea rapidă la testarea şi execuţia programului.

La comanda RUN se iniţializează un pointer către prima dată din blocul de date asociat primei instrucţiuni DATA.

La fiecare moment acest pointer va indica data posibilă de citit prin READ.

3.5. Instrucțiunea READ

Formatul general al instrucțiunii:

READ <var.>, <var.> ....

unde:

<var.> poate fi o variabilă simplă sau indexată de tip numeric sau șir.

La execuția instrucțiunii READ se face o verificare strictă în privința corespondenței de tip dintre variabila var și data curent disponibilă.

Dacă tipurile coincid, primei variabile din listă i se atribuie data curentă după care pointerul va indica următoarea dată disponibilă din blocul de date. Dacă în blocul de date al unei instrucțiuni nu mai sînt date disponibile, se trece automat la instrucțiunea DATA următoare dacă aceasta există.

Instrucțiunea READ poate fi plasată oriunde în corpul programului.

Datele introduse prin DATA și neexploatate prin READ sînt ignorate.

3.6. Instrucțiunea RESTORE

Instrucțiunea RESTORE permite refolosirea prin READ a datelor introduse prin DATA. Acest lucru este necesar cînd într-un program este necesară folosirea în mod repetat a aceloraşi date.

Formatul general al instrucțiunii:

RESTORE <nr. linie>

<nr. linie> indică numărul de linie al instrucțiunii DATA pe blocul de date al căreia se dorește reinițializarea pointerului.

Dacă nr. linie nu este indicat, reinițializarea pointerului se face pe blocul de date al instrucțiunii DATA cu cel mai mic număr de linie.

Exemple:

```
50 RESTORE 1000
100 RESTORE
1000 DATA 'A—B', 5,7, .21
```

3.7. Instrucțiunea INPUT sau LISTEN

Instrucțiunea INPUT oferă utilizatorului posibilitatea introducerii dinamice a datelor necesare executării unui program. Utilizarea datelor prin DATA READ RESTORE este statică deoarece în momentul executării programului acestea sînt înghețate putînd fi modificate numai prin corectarea unei instrucțiuni DATA.

Folosind instrucțiunea INPUT utilizatorul poate introduce datele necesare execuției programului la cererea expresă a acestuia direct pe terminalul de conversație. În loc de cuvîntul cheie INPUT se poate folosi și LISTEN.

Formatul general al instrucțiunii INPUT:

INPUT <const șir>, <var.>, <var.>

unde:

<const.șir> reprezintă orice conatantă șir admisă în limbaj

<var.> reprezintă orice variabilă simplă sau indexată, numerică sau șir, admisă în limbaj

Exemplu:

```
10 INPUT "RAZA CERCULUI:", R
```

Utilizarea <const.şir> în instrucţiunea INPUT măreşte siguranţa introducerii corecte a datelor necesare în fiecare moment. La execuţia liniei program 10 se editează pe linie nouă mesajul :

RAZA CERCULUI :

după care se lansează efectiv cererea de introducere a valorii necesare care se va atribui variabilei R.

Introducerea datelor se consideră terminată în momentul tastării caracterului retur de car.

Dacă numărul datelor introduse este mai mic decît numărul variabilelor din lista instrucţiunii INPUT se relansează pe linie nouă cererea de introducere a datelor suplimentare necesare prin imprimarea caracterului '?'.

Dacă numărul datelor introduse este mai mare decît numărul variabilelor din lista instrucţiunii INPUT datele introduse în exces sînt neglijate cu afişarea mesajului :

•EXTRA LOST

Remarcă : la utilizarea instrucţiunii INPUT datele vor fi separate între ele la introducerea prin caracterul virgulă.

3.8. Instrucţiunea LINE INPUT

Prin convenţia fixată la instrucţiunea INPUT caracterul ',' (virgulă) dispus pe o linie de date cerută de execuţia unei instrucţiuni INPUT are semnificaţia de separator între date. Dacă la execuţia liniei program :

10 INPUT A$

se introduce următoarea linie de date :

INTERPRETOR BASIC, SISTEM DE CALCUL

Variabilei A$ i se atribuie ca valoare constantă şir 'INTERPRETOR BASIC' restul caracterelor de pe linia de date fiind neglijate.

Instrucţiunea LINE INPUT permite atribuirea întregului text al liniei de date introduse (cuprins între ? şi caracterul retur de car) ca valoare a unei variabile şir. Dacă se înlocuieşte linia 10 anterior descrisă cu linia :

10 LINE INPUT A$

prin introducerea liniei de date :

INTERPRETOR BASIC, SISTEM DE CALCUL

variabilei A$ i se atribuie ca valoare constanta şir

'INTERPRETOR BASIC, SISTEM DE CALCUL'.

În contextul instrucţiunii LINE INPUT caracterul ',' (virgulă) îşi pierde semnificaţia de separator, întreaga linie de date (textul cuprins între ? şi retur de car) fiind privită ca constantă şir.

Formatul general al instrucţiunii :

LINE INPUT [<mesaj>], <var.şir>

unde :

<mesaj> reprezintă orice constantă şir încadrată între ghilimele (") ; şi reprezintă mesajul de cerere a liniei.

<var.şir> reprezintă o variabilă simplă sau indexată de tip şir de caractere admisă de limbaj.

La execuţia instrucţiunii LINE INPUT linia de date introdusă are promterul :

? dacă mesaj lipseşte altfel apare promterul <mesaj> ?.

3.9. Instrucțiunea PRINT

Instrucțiunea PRINT este utilizată pentru editarea anumitor date rezultate intermediare sau finale ale unei execuții la terminalul utilizatorului. Formatul general al instrucțiunilor este :

PRINT <lista>

sau

PRINT AT <expr1>, <expr2> ; <lista>

unde :

<lista> reprezintă o listă de elemente admise de limbaj separate între ele prin caracterele ',' (virgula) sau ';' (punct virgulă).

<expr1>,<expr2> reprezintă coordonatele x,y a ecranului începînd de unde se dorește scrierea. În cazul cînd ieșirea nu este pe ecran această parte a instrucțiunii este inefectivă. Valoarea x se trunchează la multiplu de 8.

Elementele unei liste PRINT pot fi :

— expresii numerice sau șir care se pot reduce la constante sau variabile ;
— TAB (expr. num.) unde expr. num. poate fi orice expresie numerică admisă de limbaj ;
— SPC (expr. num.)

O succesiune de două caractere separator semnifică absența elementului de tipărit. Lista de elemente se poate termina prin separator.

Prin execuția operatorului PRINT se construiește un text care se transmite perifericului. Acest text este rezultatul prelucrării succesive a elementelor de tipărit și a separatorilor.

Prin editarea expresiilor șir se construiește o secvență constituită din toate caracterele valorii expresiei șir.

Exemplu :

```
10 PRINT ''BASIC-PRAE'' + ''CLUJ-NAPOCA''
```

va extrage pe terminal

```
BASIC-PRAE CLUJ-NAPOCA
```

Prin editarea unei expresii aritmetice se construiește o succesiune de simboluri care constituie reprezentarea zecimală a valorii expresiei. Convențiile de editare în formatul implicit sînt :

— reprezentarea zecimală a unui număr este precedată de spațiu (nr. pozitiv) sau semnul minus (număr negativ), și urmată de blanc.

— precizia reprezentării numerelor este de 11 cifre semnificative.

Orice număr care poate fi reprezentat ca întreg va fi extras ca număr zecimal întreg (fără punct).

Numerele a căror valoare absolută este cuprinsă între 1/100 și 1 000 000 și se pot reprezenta exact sub forma unui număr zecimal, se reprezintă cu format zecimal. Zerourile nesemnificative nu se reprezintă. Restul numerelor se reprezintă în format zecimal cu exponent sub forma :

semn mantisa E semn exponent spațiu

cu mantisa cuprinsă între 1 și 10 avînd cel mult 11 cifre semnificative, iar exponentul cuprins între 0 și 38.

De exemplu :

— valoarea 625 se va extrage sub forma : 625
— valoarea 0.012345 se va extrage sub forma .012345
— valoarea 0.00123456 se va extrage sub forma :

1.23456E-03

Liniile care vor fi extrase se împart în zone a 14 caractere (5 cîmpuri pentru un terminal cu 80 caractere, respectiv 8 cîmpuri pentru linia cu 132 caractere).

Separatorul ';' produce generarea în textul de tipărit a unui șir vid. Textele generate din două valori de elemente separate prin ';' se succed nemijlocit.

Separatorul ',' produce generarea în textul de ieșire după valoarea elementului care-l precede a unuia sau a mai multor caractere blanc (spațiu) pînă la umplerea zonei respective.

Existența unui separator (',' sau ';') după ultimul element al listei instrucțiunii PRINT suprimă operația RETUR DE CAR și salt la linie nouă. În acest fel instrucțiunea PRINT imediat următoare va edita datele pe aceeași linie terminal.
Exemplu :

```
10 R=2. : PRINT "RAZA=";R;
15 PI=4*ATN(1)
20 PRINT "ARIA CERCULUI="; PI*R**2
RUN
RAZA=2 ARIA CERCULUI=12.566370614
```

Instrucțiunea PRINT fără listă extrage pe terminal tot șirul creat anterior, iar dacă nu există scrie o linie vidă.

Pentru un control mai eficient al formatului de editare, limbajul BASIC PRAE pune la dispoziția utilizatorilor funcția de tabulare TAB.

Funcția TAB acceptă ca parametru orice expresie numerică. La execuție se evaluează valoarea expresiei și dacă este necesar se realizează conversia real-întreg pentru a obține o valoare întreagă care va reprezenta numărul coloanei curente în care se va scrie în continuare. Deoarece numărul coloanei curente are o evoluție strict progresivă rezultă că funcția TAB este ineficientă pentru argumente negative, nule sau mai mici decît numărul coloanei curente. Dacă valoarea transmisă ca parametru este mai mare decît lungimea unei linii terminal, funcția TAB provoacă extragerea liniei curente și poziționarea la începutul liniei terminal imediat următoare. Instrucțiunea PRINT poate provoca extragerea uneia sau mai multor linii terminal.

Funcția SPC acceptă ca parametru orice expresie numerică. Funcția generează atîtea spații cît este valoarea expresiei trunchiată la întreg.

Instrucțiunea PRINT poate fi folosită în mod imediat și poate să apară în orice poziție pe o linie multiplă.

3.10. Instrucțiunea PRINT USING

Dacă într-un program se dorește editarea datelor altfel decît cu formatul implicit, limbajul BASIC-PRAE pune la dispoziția utilizatorului instrucțiunea PRINT USING.

Formatul general al instrucțiunii este :

```
PRINT USING <format>, <listă>
PRINT AT <e1>, <e2> : USING <format>, <listă>
```

unde :

<format>	reprezintă o expresie șir ce cuprinde formatul de editare
<listă>	reprezintă o listă de elemente separate între ele prin virgulă sau punct-virgulă. Din punct de vedere sintactic lista este identică cu lista de la instrucțiunea PRINT.
<e1>, <e2>	sînt coordonatele x,y ale ecranului începînd de unde se dorește tipărirea.

Instrucțiunea PRINT USING asigură editarea datelor, rezultatelor la terminalul utilizatorului sub forma specificată la format.

În timpul execuției instrucțiunii, lista este parcursă în paralel cu șirul format. La terminarea tratării listei caracterele rămase netratate în șir dacă există, vor fi extrase așa cum apar. La terminarea șirului format elementele rămase în listă sînt editate cu formatul dat.

Exemplu :

```
PRINT USING "##.##ESTE SOLUȚIA", 12.34 ; 12.34E-2
12.34.12 ESTE SOLUȚIA
```

Separatorii dintre elementele listei nu își păstrează semnificația la instrucțiunea PRINT; funcția TAB nu mai poate figura în listă.

Șirul format este interpretat caracter cu caracter și poate conține orice caracter admis de limbaj. Pentru definirea formatului anumite caractere au semnificație specială (prin convenție). Caracterele cu semnificație specială sînt : # (diez), , (virgulă), * (asterisc), . (punct),

$ (dolar), ^ (circumflex), — (minus), ' (apostrof), și în anumite forme literele : L, R, C, E. Orice alt caracter diferit de cele înșirate va fi editat așa cum apare în format :

Exemplu :

```
PRINT USING "ALFABETA/DELTA",X
ALFABETA/DELTA
                              O
```

3.10.1. *Editarea elementelor numerice*

PRINT USING editează elementele numerice conform formatului șir specificat. Conveniile pentru format sînt :

— caracterul (diez) plasat în format specifică un cîmp numeric. Numărul caracterelor apărute consecutiv specifică numărul cifrelor ocupate de număr cadrat la dreapta. Cîmpurile neocupate se umplu cu blancuri.

Exemplu :

```
PRINT USING"#### ########",1234 ; 1·E6
1234 1000000
PRINT USING"####",12
12
```

Dacă numărul nu încape în cîmpul specificat numărul va fi extras cu format implicit

Exemplu :

```
PRINT USING"##",135
%135
```

Numerele vor fi rotunjite dacă este cazul.

Exemplu :

```
PRINT USING"## ##",12·4 ; 12·5
12 13
```

Dacă se dorește editarea unui număr cu punct zecimal, se introduce caracterul (punct) în șirul format. Numărul caracterelor dispuse la stînga punctului zecimal în șirul format indică numărul de cifre zecimale a părții întregi a numărului ; cele dispuse la dreapta punctului zecimal dau numărul de cifre a părții zecimale a numărului.

Exemplu :

```
PRINT USING"#·# ·# #·", 1·1 ; ·1 : 1·6
1·1 ·1 2·
```

Pentru editarea numerelor negative trebuie să fie introdus în șirul format un caracter pentru semn :

Exemplu :

```
PRINT USING"##·## ##·##",—1 ; —2
—1·00 —2·00
```

Dacă se dorește editarea semnului la dreapta numărului, atunci șirul caracterelor se termină cu caracterul — (minus) :

Exemplu :

```
PRINT USING"##·#— ##·##—",1·0 ; —1·0
1·0 1·0—
```

Dacă în șirul format două caractere * (asterisc) preced formatul numeric, atunci cîmpul este completat la stînga de asteriscuri :

Exemplu :

```
PRINT USING "SOLUȚIA :X=**##",1
SOLUȚIA X=***1
```

Numerele negative pot fi extrase cu acest format ,sau se folosește — la sfîrșit de format :

Exemplu :

```
PRINT USING "''##",-1
'-1
PRINT USING "'##-'",-1
''1-
```

Dacă formatul numeric este precedat de două caractere $ (dolar), în editare numărul va fi precedat de semnul $ (dolar). Numerele negative pot fi extrase cu acest format :

Exemplu :

```
PRINT USING "PRET=$$##",12
PRET=$12
PRET USING "PRET=$$##", -1
PRET=-$1
PRINT USING "PRET=$$##-",-1
PRET= $ 1-
```

Plasarea caracterului , (virgulă) oriunde în stînga punctului zecimal între caracterele în cadrul formatului produce separarea prin virgulă în grupe de 3 (trei) cifre din stînga punctului zecimal a numărului care trebuie editat.

Exemplu :

```
PRINT USING"##,#########$$#,####",1.E6,12345-6
1,000,000 $ 12,346
```

Formatul exponențial este specificat prin caractere ^ (circumflex) după formatul numeric. Formatul exponențial nu admite caracterele *, $ și —. Dacă șirul format conține mai puține caractere ^ decît 4, ele vor apare la editare așa cum sînt specificate la format. În cazul cînd un element din listă nu începe în cîmpul specificat și este editat în mod implicit se abandonează tratarea șirului format în continuare și toate elementele vor fi extrase cu formatul implicit. Primul cîmp caracter din format este rezervat pentru semn.

Exemplu :

```
PRINT USING "##^^^^", 12.90E20
1E+20
PRINT USING
"; ^^^^; ^^^  ^^^^", 1E7 ; 12.2 ; 1E8
   1.0E+07 12.2  ^^^ 1.0E+08
```

Formatul exponențial nu este compatibil cu descriptorii '$' și '*'.

3.10.2. Editarea șirurilor prin PRINT USING

Formatul de editare șir furnizează informații privind : numărul de caractere alfanumerice de extras, cadrarea șirurilor (stînga, dreapta).

Descrierea formatului șir începe totdeauna cu caracterul '(apostrof) și se poate continua cu un șir de caractere identice (caracterul repetat poate fi 'C', 'E', 'L', 'R'). Lungimea cîmpului șir indicat de un format șir este dat de numărul de repetiții din format mărit cu unu.

În cazul în care lungimea șirului de extras este mai mare decît lungimea cîmpului șir descris de format, la editare se vor extrage începînd de la stînga atîtea caractere cîte indică lungimea cîmpului. Dacă lungimea șirului este mai mică decît lungimea șirului format atunci șirul va fi extras în întregime cadrat conform descriptorului și completat cu blancuri. În situația în care formatul conține numai caracterul apostrof ('), se va edita primul caracter din șir :

Exemplu :

```
PRINT USING "'", "ABCD"
A
```

Formatul șir cu indicația cadraj la stînga este constituit din caracterul aspostrof (') urmat de o serie de caractere "L". În acest caz șirul se editează începînd cu primul caracter din stînga, de lungime egală cu numărul caracterelor "L" plus 1.

Exemplu :

```
PRINT USING " 'LLLL", "ABC"
ABC
PRINT USING " 'LLLL", "LMNOPRS"
LMNOP
```

Formatul șir cu indicația cadraj la dreapta este format din caracterul apostrof ('), urmat de o serie de caractere "R". În acest caz șirul se editează cadrat la dreapta completat eventual la stînga cu blancuri.

Dacă lungimea șirului depășește lungimea cîmpului șirul va fi extras cadrat la stînga !

Exemplu :

```
PRINT USING " 'RRRR","ABC"
ABC
PRINT USING " 'RRRR","ABCDE"
ABCDE
PRINT USING " 'RRRR","BASIC-PROE CLUJ-NAPOCA"
BASIC
```

Formatul centrat constă dintr-un caracter apostrof (') urmat de o serie de caractere "C". Cu acest format se editează un șir pe cîmpul format cadrat central și completat cu blancuri în stînga și dreapta.

Dacă lungimea șirului este mai mare decît lungimea cîmpului format șirul va fi cadrat la stînga.

Exemplu :

```
PRINT USING " 'CCCCCCC", "ABC"
 ABC
PRINT USING " 'CCCCCCC","ABCD"
ABCD
PRINT USING " 'CCCCCCC","ABCDEFGHIJKLMNO"
ABCDEFGH
```

Formatul extins constă dintr-un caracter apostrof (') urmat de o serie de caractere "E". Cu acest format se editează un șir pe cîmpul format, cadrat la stînga.

Dacă lungimea șirului depășește cîmpul specificat cîmpul va fi extins și tot șirul va fi extras :

Exemplu :

```
PRINT USING " 'EEE****","A"
A ****
PRINT USING ",EEE****","ABCD"
ABCD****
PRINT USING " 'EEE****","ABCDEFGH"
ABCDEFGH****
```

3.11. Instrucțiunea GOTO

Instrucțiunea GOTO servește la modificarea ordinii secvențiale de execuție a instrucțiunilor program, (salt necondiționat).

Formatul general al instrucțiunii :

GOTO <nr. linie>

unde nr. linie este orice număr de linie admis (23767).

La execuţia acestei instrucţiuni controlul este transferat primei instrucţiuni a liniei indicate. Din formatul specificat este evident că nu se poate da controlul altei instrucţiuni decît la prima de pe o linie program :

Exemplu :

```
50 GOTO 200
200 X=X+1 : PRINT X
```

În exemplul dat controlul nu poate fi transferat instrucţiunii PRINT.

Dacă prima instrucţiune a liniei specificate este neexecutabilă (DATA, REM) controlul va fi transferat primei instrucţiuni care urmează.

Exemplu :

```
10 GOTO 50
50 DATA 1,2,3
60 X=A 2+B 2
```

În acest caz, controlul va fi transferat instrucţiunii X=A2+B2. Încercarea de transfer a controlului la o linie inexistentă se soldează cu oprirea execuţiei şi afişarea unei erori.

Instrucţiunea GOTO poate avea orice poziţie pe o linie multiplă.

3.12. Instrucţiunea ON . . . GOTO

Instrucţiunea ON...GOTO serveşte la realizarea transferului selectiv conform valorii unei expresii. Formatul general al instrucţiunii este :

ON <expr. num.> GO TO <nr. linie>, <nr. linie>, ...

La execuţia instrucţiunii se evaluează expresia numerică iar partea întreagă a valorii calculate se foloseşte ca index în lista numerelor de linie pentru a determina linia căreia i se predă transferul. La execuţia instrucţiunii :

ON X GOTO 100,200,300

controlul este transferat liniei 100 dacă X are valoarea 1, liniei 200 dacă valoarea liniei X este 2 liniei 300 dacă X are valoarea 3. În cazul în care partea întreagă este negativă, nulă, sai mai mare decît numărul elementelor din listă, controlul trece la instrucţiunea ce urmează în secvenţă.

Instrucţiunea ON . . . GOTO poate avea orice poziţie pe linia multiplă.

3.13. Instrucţiunea IF

Instrucţiunea IF serveşte pentru realizarea unui transfer al controlului sau execuţia unei instrucţiuni în funcţie de valoarea logică a unei condiţii. Formatul general al instrucţiunii :

IF <cond.> THEN <nr. linie>, <instr.> [ELSE <nr.linie>, <instr.>]

unde :

<cond.>	poate fi orice expresie logică de relaţie sau aritmetică.
<nr. linie>	poate fi orice număr de linie admis
<instr.>	poate fi orice instrucţiune alta decît : FOR, NEXT, FNEND, END, DIM, DEF, IF, REM, DATA.

Dacă valoarea logică a condiţiei este adevărată atunci se execută partea THEN (sau GOTO) a instrucţiunii, altfel se execută partea ELSE a instrucţiunii :

IF A>B THEN PRINT "ADEVĂRAT" ELSE PRINT "FALS"

3.14. Cicluri

Prin ciclu se înțelege un set de instrucțiuni a căror execuție se repetă pînă la îndeplinirea unei condiții. Fiecare ciclu posedă elemente caracteristice :

— variabilă de ciclu ;
— valoarea inițială ;
— test final.

Exemplu :

```
100 I=1
110 IF I 10 THEN 140
120 PRINT I, I 2, I 3
130 I=I+1 : GO TO 110
140 PRINT "SFIRSIT CICLU"
```

Fiecare ciclu are patru părți :

— inițializarea ciclului prin care se inițializează variabila ciclului. (linia 100) ;
— condiția de terminare a execuției ciclului (linia 110) ;
— corpul ciclului (linia 120).
— modificarea valorii variabilei cu pasul ciclului.

Limbajul BASIC-PRAE prin intermediul instrucțiunilor FOR și NEXT permite declararea și constituirea comodă și simplă a ciclurilor.

3.15. Instrucțiunea FOR ... TO

Instrucțiunea FOR definește începutul ciclului, formatul general al instrucțiunii FOR :

FOR <var.num> = <expr.num1> TO <expr.num2> STEP <expr.num3>

unde :

<var. num.> poate fi orice variabilă simplă și reprezintă variabila de ciclu.
<expr. num1> poate fi orice expresie numerică, valoarea ei reprezentînd valoarea inițială a variabilei ciclului.
<expr. num2> poate fi orice expresie numerică, valoarea ei reprezentînd valoarea finală a variabilei ciclului.
<expr. num3> poate fi orice expresie numerică, valoarea ei reprezentînd valoarea pasul cu care se modifică variabila ciclului.

În cazul în care indicarea pasului lipsește, pasul de ciclare se consideră implicit 1.

Variabila de control a ciclului poate fi modificată în interiorul ciclului.

Expresiile care apar pe instrucțiunea FOR sînt evaluate o singură dată la execuția instrucțiunii FOR.

3.16. Instrucțiunea NEXT

Instrucțiunea NEXT definește sfîrșitul ciclului. Formatul general al instrucțiunii este :

NEXT <var.num.>

unde <var.num.> este variabila cu același nume ca și pe instrucțiunea FOR corespunzătoare. La execuția instrucțiunii NEXT se incrementează valoarea variabilei ciclului cu valoarea pasului și se face testul final al ciclului. Ciclurile FOR ... NEXT pot fi imbricate adică corpul unui ciclu poate să conțină alt ciclu.

3.17. Subrutine

Limbajul BASIC-PRAE permite declararea unui set de instrucțiuni ca o subrutină prin intermediul a două instrucțiuni :

— Instrucțiunea GOSUB de apel a subrutinei.

— instrucţiunea RETURN, de retur din subrutina la instrucţiunea imediat următoare apelului.

Într-un program pot fi definite mai multe subrutine. Acestea pot fi plasate oriunde în corpul programului.

3.17.1. *Instrucţiunea GOSUB, instrucţiunea RETURN*

Instrucţiunea GOSUB provoacă lansarea execuţiei unei secvenţe de instrucţiuni declarate ca subrutină a programului, cu reţinerea instrucţiunii care urmează în secvenţa instrucţiuni de apel.

Formatul general :

GOSUB <nr. linie>

În BASIC-PRAE nu există posibilitatea declarării explicite a unei subrutine, mai precis a începutului subrutinei. Controlul poate fi transferat oricărei instrucţiuni. Odată controlul transferat unei subrutine, acesta se execută pînă la întîlnirea instrucţiunii RETURN. Instrucţiunea RETURN asigură marcarea sfîrşitului subrutinei şi revenirea în program la instrucţiunea care urmează după instrucţiunea GOSUB de apel.

Formatul general :

RETURN

Transferul controlului dintr-o subrutină spre exteriorul acesteia este permis, dar este mai indicată ieşirea prin instrucţiunea RETURN.

Ca şi ciclurile program, subrutinele pot fi imbricate :

Exemplu :

```
10 REM IMBRICARI SUBRUTINE
20 GOSUB 100
30 STOP
100 PRINT "SINT IN SUBRUTINA 1"
110 GOSUB 200
120 RETURN
200 PRINT "SINT IN SUBRUTINA 2"
210 RETURN
```

O subrutină poate apela orice subrutină chiar şi pe ea însăşi. Profunzimea imbricării depinde numai de spaţiul de memorie pus la dispoziţia utilizatorului.

O subrutină poate conţine mai multe instrucţiuni de revenire (RETURN).

3.17.2. *Instrucţiunea ON ···GOSUB*

Instrucţiunea ON ... GOSUB permite transferul calculat al controlului la una din subrutinele indicate.

Formatul general al instrucţiunii :

ON <expr. num.> GOSUB <nr. linie> , <nr. linie> ...

La execuţia instrucţiunii se evaluează expresia numerică, partea întreagă a valorii calculate este folosită ca index în lista numerelor de linie. Controlul va fi transferat subrutinei indicată de index. Execuţia instrucţiunii RETURN provoacă returul în program la instrucţiunea care urmează după ON ... GOSUB.

Instrucţiunea ON ... GOSUB poate servi la apelarea unei subrutine prin unul din punctele sale de intrare.

Exemplu :

```
50 ON X GOSUB 100,200,300
100 REM INCEPUTUL SUBRUTINEI
200 REM AL DOILEA PUNCT DE INTRARE
300 REM AL TREILEA PUNCT DE INTRARE
500 RETURN
```

Controlul este transferat liniei 100 dacă X este egal cu 1, liniei 200 dacă X are valoarea 2, liniei 300 dacă X=3, și instrucțiunii care urmează după ON ... GOSUB dacă X=0 sau X=4.

3.18. Funcții definite utilizator

Practica programării arată că deseori într-un program apare necesitatea repetării, în diverse puncte, a unei expresii matematice sau șir, sau a unui set de instrucțiuni, ceea ce duce la mărirea spațiului de memorie ocupat de program.

Pentru rezolvarea acestei dificultăți BASIC-PRAE pune la dispoziție posibilitatea definirii funcțiilor care se pot apela ca și funcțiile standard.

Funcțiile utilizator se definesc prin instrucțiunea, DEFFN. Funcția utilizator poate fi:
— funcție simplă definită a cărei definiție este data printr-o instrucțiune DEF FN.
— funcție multilinie a cărei definiție este reprezentată de un set de instrucțiuni cuprins între instrucțiunile DEFFN și FNEND.

3.18.1. *Funcția simplu-definită*

Formatul general al instrucțiunii DEF FN pentru definirea unei funcții simplu definite este:

DEF FN <fct> (<par. form.>, ...) = <expr.>

unde:

<fct> poate fi orice nume de funcție admis inclusiv tipul (S).
<par. formal> poate fi orice nume de variabilă simplă sau șir.
<expr.> poate fi orice expresie de același tip cu tipul funcției.

Exemplu:

```
DEF FNA (X) = X 2+X-1
```

Expresia unei funcții poate conține un apel la orice funcție standard sau utilizator definită în prealabil:

Exemplu:

```
DEF FN B (X, Y)=Y  2-INT (FNA (X))
```

De reținut că tipul expresiei de definiție (numerică sau șir) trebuie să coincidă cu tipul declarat al funcției (numerică sau șir). De exemplu instrucțiunea:

```
DEF FNC $(X $, Y)=LEN (X$)+Y
```

nu este admisă.

Orice variabilă care apare în expresia funcției și nu apare în lista variabilelor formale va fi considerată ca fiind o variabilă a programului în care este definită funcția. La apelul unei funcții definite se face o verificare strictă în privința coincidenței între numărul și tipul parametrilor actuali și numărul și tipul parametrilor formali.

3.18.2. *Funcția multilinie*

Formatul definiției unei funcții multilinie este următorul:

```
DEFFN <fct.> (<par. form> ...)
```

Instrucțiuni care definesc corpul definiției

```
FNEND <val. funcției>
```

Instrucțiunea DEFFN pentru funcția multilinie se deosebește de cea pentru funcția simplă definită prin absența semnului '='.

Valoarea produsă la apelul unei funcții multilinie este valoarea expresiei de pe instrucțiunea FNEND.

În corpul definiției unei funcții nu au sens instrucțiunile : DATA, DEF, DIM, END.

Definiția unei funcții multilinie poate să apară oriunde în program, cu condiția ca execuția ei trebuie să preceadă primul apel al funcției.

Din corpul unei funcții multilinie se poate ieși și cu instrucțiunea FNRETURN, care are sintaxa :

FNRETURN <expresie rezultat>

Variabilele ce apar în corpul definiției și nu sînt parametri formali ai funcției sînt considerate ca fiind variabile ale programului apelant.

Exemplu : (definiția funcției factorial)

```
10 DEF FNF (N)
20 IF N=1 THEN FNRETURN 1
30 FNEND FNF (N-1)*N
40 INPUT N : PRINT N;" " :"=";FNF(N)
```

Funcția multilinie poate fi de tip numeric sau de tip șir.

Parametrii formali pot avea orice tip. La apelul unei funcții definite se face o verificare strictă privind numărul și tipul parametrilor de apel (actuali) și numărul și tipul parametrilor formali.

Instrucțiunile din corpul definiției unei funcții multilinie pot fi introduse în orice ordine dar la introducere trebuiesc încadrate între DEFFN și FNEND care stabilesc cîmpul de acțiune al parametrilor formali.

3.19. Instrucțiunea RANDOMIZE

Formatul instrucțiunii :

RANDOMIZE

Instrucțiunea RANDOMIZE poate fi plasată oriunde în program dar ea trebuie să fie totuși la începutul programului, practica recomandă utilizarea instrucțiunii RANDOMIZE după punerea la punct a programului.

3.20. Instrucțiunile STOP și END

Instrucțiunile STOP și END sînt utilizate la oprirea execuției programului. Formatul general al instrucțiunii END :

END

Instrucțiunea END poate fi dispusă și pe linie multiplă.

Instrucțiunea STOP suspendă temporar execuția unui program care poate fi oricînd reluată începînd cu instrucțiunea următoare printr-o comandă CONTINUE sau începînd cu o instrucțiune oarecare specificată prin GOTO <nr. linie> în mod imediat.

Formatul general :

STOP

Instrucțiunea STOP poate să apară în orice punct al programului și pe orice poziție a unei instrucțiuni multiple. La execuția instrucțiunii programul este oprit cu mesajul :

*BREAK LINE N

N fiind numărul de linie pe care se află STOP. Execuția programului putînd fi reluată în orice moment.

Instrucțiunea STOP oferă un mijloc foarte eficient de punere la punct a programelor cînd este utilizată în combinație cu modul imediat de lucru prin care se pot testa elementele programului.

3.21. Funcția INP

BASIC-FEAP pune la dispoziția utilizatorului funcția:

INP (<arg>)

pentru citirea unui port. Argumentul funcției este numărul portului ce urmează să fie citit, iar atribuit valorii A în exemplu:

A=INP (0)

3.22. Instrucțiunea OUT

BASIC-FEAP pune la dispoziția utilizatorului instrucțiunea OUT pentru a scrie un octet pe un port. Formatul general al instrucțiunii este:

OUT <nr. canal>, <expr>,

Instrucțiunea dă valoarea <expr.> canalului specificat de <nr. canal>.

3.23. Funcția PEEK

BASIC-FEAP permite utilizatorului accesul la memoria fizică. Sintaxa funcției este:

PEEK (<adresa>)

unde:

<adresa> este o valoare care reprezintă adresa zecimală a locației de memorie care urmează să fie citită.

Exemplu:

B=PEEK (A)

conținutul adresei A este atribuit variabilei B.

3.24. Instrucțiunea POKE

BASIC-FEAP permite utilizatorului scrierea în memoria fizică. Sintaxa instrucțiunii este:

POKE <adresa>, <valoare>

Prin instrucțiunea POKE valoarea specificată de către <valoare> este atribuită locației de adresă dată de <adresă>.

3.25. Instrucțiunea BEEP

Instrucțiunea BEEP servește la producerea unui sunet cu ajutorul generatorului sonor. Sintaxa instrucțiunii este:

BEEP <durata>, <frecvența>

unde <durata> este o valoare cuprinsă între 0 și 255 și stabilește durata sunetului;
<frecvența> este o valoare cuprinsă între 0 și 32767 și stabilește frecvența sunetului.

3.26. Instrucțiunea CALL

BASIC-PRAE permite utilizatorului apelarea unor subprograme scrise în limbaj de asamblare (cod mașină).

Formatul instrucțiunii este :

CALL <adresă>, <arg. 1>, <arg. 2>, ...

Instrucțiunea predă controlul adresei specificate. Fiecare argument se reprezintă pe doi octeți și schema de transmitere a lor este următoarea :

Regiștrii : HL adresa primului agrument de pe STACK.
BC conține numărul argumentelor aflate pe STACK
SP conține adresa vîrfului STACK-ului unde se află argumentele astfel :

ARG.N ← SP
...
ARG.2
ARG.1 ← HL
ADRESA DE RETUR

3.27. Funcția FRE

Funcția FRE are rolul de a furniza utilizatorului memoria liberă. Dacă parametrul este o variabilă, se dă memoria disponibilă pentru variabile iar dacă se dă o variabilă șir se furnizează zona liberă șir.

Exemplu :

FRE(X) furnizează zona liberă variabile.
FRE(X$) furnizează zona liberă șir.

4. MODUL IMEDIAT

Pentru rezolvarea unor probleme simple reduse la calcularea unor expresii folosind BASIC-PRAE nu se impune scrierea unui program complet și apoi lansarea execuției acestuia.

Majoritarea instrucțiunilor limbajului BASIC-PRAE pot fi utilizate ca niște comenzi "ON LINE" și executate imediat după introducere.

Apar astfel două moduri de lucru care se pot folosi separat sau combinate dacă se utilizează sub sistemul BASIC-PRAE :

— modul imediat (direct) ;
— modul program (indirect).

Instrucțiunea folosită în mod imediat, executîndu-se după introducere, rezultă că nu poate fi folosită pe o linie multiplă ; de aici apare echivalența dintre instrucțiunea imediată și linia imediată.

Deosebirea dintre linia program și linia imediată constă în faptul că linia imediată nu posedă număr de linie.

Exemplu : linia :

10 PRINT "BASIC-PRAE"

este o linie program, în timp ce linia :

PRINT "BASIC-PRAE"

este o linie imediată.

Liniile care încep cu un număr de linie sînt memorate într-o forma internă specială și executate ulterior la o comandă RUN.

Modul de lucru imediat este deosebit de util în două situații tipice :

— punerea la punct a programelor;

— efectuarea unor calcule care nu ar justifica scrierea unui program.

Folosirea anumitor instrucțiuni în mod imediat nu are sens. Din această categorie fac parte instrucțiunile DATA, DEF, FNEND.

Posibilitatea marcării într-un program a punctelor de întrerupere, prin utilizarea instrucțiunii STOP, pune la dispoziția utilizatorilor un mijloc foarte eficace de punere la punct a programelor prin combinarea judicioasă a celor două moduri de lucru. În plus execuția unui program întrerupt prin STOP poate fi oricînd reluată printr-o comandă CONTINUE sau printr-o instrucțiune GO TO în mod imediat.

5. COMENZI BASIC-PRAE

Introducerea unui program în memorie nu este suficientă pentru rezolvarea problemei pentru care acesta a fost proiectat.

Practica programării arată că sînt foarte rare situațiile în care un program scris este absolut corect. Apare astfel problema punerii la punct a unui program introdus. Există situații în care punerea la punct a unui program necesită listarea acestuia sau ștergerea și corectarea unor linii. Odată pus la punct pentru o utilizare ulterioară apare problema salvării programului pe un suport de unde să poată fi regăsit în orice moment. Salvările succesive ale programelor pot duce la lipsa spațiului pe suport extern pentru salvări. Apare astfel problema eliminării unor programe care nu mai sînt utilizate.

Pentru rezolvarea dificultăților mai sus enumerate BASIC-PRAE pune la dispoziția utilizatorilor un set de instrucțiuni speciale numite comenzi. Din punct de vedere al tratării o comandă poate fi privită ca o instrucțiune imediată, ele pot să apară și în program.

Linia de comandă este identică cu linia imediată fiind lipsită de număr de linie ; deoarece se execută imediat după introducerea ei poate fi și linie multiplă. Între comenzi și instrucțiuni există însă o deosebire esențială constînd în aceea că în timp ce o instrucțiune operează asupra entităților dintr-un program, comanda operează asupra programului privit ca o entitate de sine stătătoare.

În funcție de tipul de operații executate asupra programului, comenzile se împart în trei categorii :

a) comenzi de prelucrare a programului curent în memoria pusă la dispoziția utilizatorului (comenzi de editare) :

1. DELETE
2. LIST
3. NEW
4. RENUMBER
5. EDIT
6. AUTO

b) comenzi de execuție și asistare a execuției unui program :

8. CONTINUE
9. TRACE
10. RUN

c) comenzi de salvare (încărcare a programelor în) din bibliotecile de programe :

12. AMERGE
13. LENGTH
14. LOAD, ALOAD
15. PRECISION
16. SAVE, ASAVE
17. KILL

Formatul general al unei linii de comandă este :

<cuvînt cheie> <argumente>

Fiecare dintre paragrafele care urmează conțin informații complete privind comanda pe care o tratează.

5.1. Comenzi de editare

In categoria comenzilor de editare intră comenzile prin care se realizează modificarea unui set de instrucțiuni, fiind parțială sau totală.

Aceste comenzi nu influențează execuția unui program și pregătesc spațiul de memorie al utilizatorului și servesc la corectarea programului.

5.1.1. Comanda DELETE

Comanda DELETE servește la eliminarea dintr-un program a unui set de instrucțiuni consecutive din punctul de vedere al numărului de linie. În fapt, comanda realizează eliminarea unui set de linii program

Formatul general al comenzii

DELETE N1—N2

unde N1, N2 poate fi orice număr de linie admis de limbaj. Se impune însă respectarea relației de condiție:

N1—N2

Comanda realizează eliminarea din program a liniilor având numărul de linie cuprins în intervalul închis N1, N2.

Dacă N1=N2 comanda realizează eliminarea liniei cu numărul N1 (=N2) dacă aceasta există.

5.1.2. Comanda LIST

Comanda LIST servește la extragerea pe terminalul utilizatorului a programului sau a unei părți din programul curent depus în memoria internă.

Formatul general al comenzii:

LIST [N-/-N/N1-N2,

unde N, N1, N2 poate fi orice număr de linie admis de limbaj.
În formatul:

LIST N-

comanda realizează extragerea la terminalul utilizatorului a tuturor liniilor programului având numărul de linie mai mare sau egal cu N-. În formatul:

LIST -N

comanda realizează extragerea la terminalul utilizatorului a tuturor liniilor programului având numărul de linie mai mic sau egal cu N. În formatul:

LIST N1-N2 [E]

comanda realizează extragerea la terminalul utilizatorului a liniilor programului având numărul de linie cuprins în intervalul închis N1, N2.

De reținut că în acest format se impune condiția N1←N2. Dacă N1=N2 se editează linia cu numărul N1(=N2) dacă aceasta există.

Listarea programului la un alt terminal nu poate fi realizată prin comanda LIST ci numai prin comanda LLIST.

Pentru a lista toate variabilele programului cu valorile corespunzătoare, utilizatorul are la dispoziție comanda LVAR

Sintaxa comenzii este:

LVAR

Dacă utilizatorul dorește să scrie (afișeze) la imprimantă, sînt BASIC-FELIX își pune la dispoziție următoarele instrucțiuni și comenzi echivalente cu comenzile din paranteze:

LLIST (LIST); LLVAR (LVAR); LNULL (NULL);
LPRINT (PRINT); LPRINT USING (PRINT USING);
LTRACE (TRACE); LWIDTH (WIDTH); LPOS (POS)

Pentru a fixa lungimea rîndului de ecran, BASIC-FELIX pune la dispoziția utilizatorului comanda WIDTH avînd formatul:

WIDTH <arg.>

unde <arg.> este lungimea liniei.

5.1 3. Comanda TRACE

Comanda TRACE explorează modul de lucru a programului pentru a se putea urmări urma programului. Sintaxa comenzii este:

TRACE <expresie>

Dacă valoarea <expresie> -i este diferită de 0 sau ii lipsește (urmărește) ; atunci se vor afișa numerele de linie ale liniilor executate. Dacă <expresie> are valoarea 0 programul va fi executat fără facilitatea TRACE.

5.1A. Comanda EDIT

Comanda EDIT servește la corectarea unei linii BASIC. Comanda are formatul general:

EDIT <nr. linie>

Unde:

<nr. linie> este numărul de linie al liniei ce urmează să fie corectată.

La această comandă mai există un set de comenzi speciale ce constă din cîteva litere, care nu sînt afișate la tastare. Numerele pot fi în intervalul 1,255.

Lista comenzilor este următoarea

A	începe din nou bufferul EDIT din memorie.
ND	șterge N caractere consecutive.
E	termină editarea liniei și o înlocuiește.
NEX	caută al N-lea caracter X din linie și păstrează pointerul înaintea caracterului.
H	șterge tot ce este la dreapta pointerului și intră în modul de inserare.
I	intră în modul de inserare și inserează toate caracterele ce se introduc pînă cînd se tastează CR (return) sau ESC(SHIFT-E).
NKX	șterge caracterele de la pointer pînă la al N-lea caracter X (acesta nu va fi șters).
L	listează linia
Q	abandonează editarea fără a înlocui linia.
NR	înlocuirea a N caractere următoare cu N caractere introduse.
X	duce pointerul la sfîrșitul liniei și se intră în modul de inserare.
BLANK	duce pointerul la dreapta.
RUBOUT	duce pointerul la stînga.
CR	sfîrșitul editării liniei
ESC	sfîrșit mod de inserare.

5.1.5. Comanda NEW

Comanda NEW realizează eliminarea din memorie a programului curent pentru a face posibilă introducerea unui nou program.

Formatul general al comenzii :

NEW

Exemplu :

NEW

Prin această comandă se elimină din memorie orice informaţie legată de programul curent dacă acesta a existat, pregătindu-se astfel introducerea unui nou program.

5.1.6. *Comanda RENUMBER*

Comanda RENUMBER serveşte la renumerotarea liniilor BASIC. Formatul general al comenzii :

RENUMBER [<număr i>, <număr p>, <număr b>]

Unde :

<număr i> reprezintă numărul de linie de la care se începe renumerotarea.

<număr p> reprezintă mărimea pasului dintre numerele de linie atribuite la două linii consecutive.

<număr b> reprezintă numărul de linie minim ce se renumerotează.

5.1.7. *Comanda NULL*

Sintaxa comenzii este :

NULL <expr.>, <cod>

Unde <expr> este numărul caracterelor ce vor fi adăugate caracterelor CR şi LF la orice cerere de I/E de pe terminal ; <cod> este codul ASCII al caracterului. Pentru terminale lente este indicată execuţia comenzii la începerea sesiunii de lucru.

NULL 3,255

Comanda NULL 0 restabileşte configuraţia de bază.

5.1.8. *Comanda AUTO*

Cu comanda AUTO numerele de linie ale întregului program BASIC sînt generate automat Formatul general al comenzii este :

AUTO [<nr. lin.> [, <pas>]]

Unde :

<nr. lin.> reprezintă numărul de linie atribuit primei linii din program. Valoarea implicită este 10.

<pas> reprezintă incrementul dintre două numere de linie consecutive. Valoarea implicită este 10.

5.1.9. *Comanda CLEAR*

Pentru a putea şterge toate variabilele, BASIC-PRAE dispune de comanda CLEAR. Formatul general al comenzii este :

CLEAR <număr>

Dacă argumentul număr este prezent, alocă pentru zona de şiruri lungimea număr.

5.2. Comenzi de execuție

În categoria comenzilor de execuție intră comenzile de lansare a programului, de pregătirea relansării și de continuare a unui program întrerupt. Aceste comenzi afectează direct execuția unui program.

5.2.1. *Comanda CONTINUE*

Comanda CONTINUE permite reluarea execuției programului din punctul de întrerupere realizat prin instrucțiunea STOP.

Formatul general al comenzii :

CONTINUE

Împreună cu instrucțiunea STOP și cu modul de lucru imediat, constituie mijloace foarte eficiente de punere la punct a programului.

Comanda CONTINUE nu poate înlocui comanda RUN și nu are sens decît în punctele de întrerupere ale unui program lansat prin RUN.

5.2.2. *Comanda RUN*

Comanda RUN realizează lansarea în execuție a programului curent în memorie.

Formatul general al comenzii :

RUN

În cazul unei comenzi RUN se lansează în execuție programul curent dispus în memoria internă a utilizatorului de la instrucțiunea cu cel mai mic număr de linie.

Comanda RUN poate fi lansată și dintr-un program dacă ea are număr de linie :

<nr. linie> RUN

5.3. Comenzi de bibliotecă

În categoria comenzilor de bibliotecă intră comenzile de salvare a programelor din memoria internă pe un suport extern, cele de încărcare a programelor dispuse pe un suport extern în memoria internă și cele care asigură gestiunea programelor pe suportul extern.

5.3.1. *Comanda LOAD, ALOAD*

Comanda LOAD servește la încărcarea unui program de pe un suport extern indicat, în memoria internă.

Formatul general al comenzilor :

LOAD <nume fișier> <adresă început>

respectiv

ALOAD <nume fișier>

unde nume fișier este un șir de caractere cuprinse între apostroafe iar adresa început este adresa la care se începe încărcarea.

Programele scrise în cod ASCII se pot încărca prin comanda ALOAD. Fiecare linie începe cu un număr de linie și se termină cu caracterul CR (retur de car). Sfîrșitul se detectează fie prin codul CTRL/Z din text fie prin marcajul EOF.

Înainte de a încărca programul specificat, comanda ALOAD șterge programul vechi din memorie. În timpul căutării programului se listează pe consolă, numele tuturor programelor întîlnite. Un program salvat cu SAVE nu poate fi încărcat cu comanda ALOAD. Programul salvat cu ASAVE poate fi încărcat cu ALOAD.

5.3.2. *Comanda PRECISION*

Precizia de calcul a componentei BASIC-PRAE este stabilită la 11 cifre semnificative. Totuşi, dacă utilizatorul doreşte ca rezultatele să fie tipărite cu o altă precizie mai mică decît 11, poate folosi comanda PRECISION.

Formatul general al comenzii este:

PRECISION <număr>

Unde:

<număr> este un număr mai mic sau egal cu 11 şi reprezintă precizia de tipărire a tuturor numerelor. (Interior se lucrează în continuare cu 11 cifre semnificative).

5.3.3. *Comanda SAVE, ASAVE*

Comanda SAVE produce salvarea blocului de memorie curentă dispus în memoria internă pe un suport extern.

Formatul general al comenzii:

SAVE <nume fişier> <expr. 1>,[<expr.[2>

Unde:

<nume fişier> este un şir de caractere cuprins între apostroafe
<expr. 1> este adresa de la care se salvează
<expr. 2> este adresa pînă la care se salvează

La execuţia comenzii SAVE se creează pe suportul indicat un fişier cu numele conţinut în specificaţie.

Exemplu:

SAVE „D", 3000,[6000

Această comandă SAVE salvează programul din memorie într-un fişier cu numele D.

Comanda

ASAVE[<nume fişier>

converteşte programul curent aflat în memorie în cod ASCII pentru a permite salvarea programului şi îl salvează apoi pe suportul extern.

5.3.4. *Comanda KILL*

Prin comanda KILL se recîştigă zona alocată pentru liste şi matrici. Formatul general:

KILL <nume matrice>, <nume matrice> ...

Unde:

<nume matrice> este numele unui masiv sau liste

5.3.5. *Comanda AMERGE*

Comanda AMERGE serveşte pentru încărcarea şi interclasarea unui program de pe caseta cu programul aflat în memorie.

Sintaxa comenzii este:

AMERGE <nume fişier>

unde nume fişier este un şir de caractere cuprins între apostroafe.

În timpul căutării programului specificat se listează numele tuturor fişierelor întîlnite pe suport, astfel obţinîndu-se repertoriul programelor existente pe casetă.