

## PASC320

### Γενική Περιγραφή

Η γλώσσα PASC320 μοιάζει με τη γλώσσα υψηλού επιπέδου PASCAL, και ορίζεται στη συνέχεια. Η PASC320 είναι δομημένη με σύνθετες εντολές, οι οποίες είναι υποσύνολο των εντολών της κλασικής PASCAL. Από τους τύπους της κλασικής PASCAL στην PASC320 δεν περιλαμβάνεται τύπος δείκτη, περιλαμβάνεται όμως ένας τύπος λίστας που θυμίζει την αντίστοιχη δομή της LISP. Η δομή των υποπρογραμμάτων διατηρείται όπως στην κλασική PASCAL, με τη δυνατότητα ορισμού φωλιασμένων υποπρογραμμάτων. Τέλος, στην PASC320 δεν υποστηρίζονται πυκνωμένες δομές.

### Ειδική Περιγραφή

#### A. Λεκτικές Μονάδες

Οι λεκτικές μονάδες που αποτελούν και τα τερματικά σύμβολα της γραμματικής της PASC320 περιγράφονται στη συνέχεια. Σε παρένθεση – όπου χρειάζεται – δίνονται τα αντίστοιχα συμβολικά ονόματα που εμφανίζονται στη γραμματική της PASC320.

Μαζί με τις λεκτικές μονάδες δίνεται και η περιγραφή των σχολίων της PASC320, τα οποία όμως δεν εμφανίζονται στη γραμματική της γλώσσας.

Σημειώνεται ότι στη γλώσσα PASC320 δεν υπάρχει διάκριση μεταξύ κεφαλαίων και πεζών αλφαριθμητικών χαρακτήρων, εκτός αν αυτοί αποτελούν μέρος της λέξης μιας λεκτικής μονάδας CCONST ή STRING.

#### Λέξεις-κλειδιά

Οι παρακάτω λέξεις που αποτελούν ανεξάρτητες λεκτικές μονάδες της PASC320:

PROGRAM CONST TYPE ARRAY LIST SET OF RECORD VAR FUNCTION PROCEDURE  
INTEGER REAL BOOLEAN CHAR FORWARD LENGTH NEW BEGIN END IF THEN ELSE  
WHILE DO CASE OTHERWISE FOR TO DOWNTOW WITH READ WRITE

Άλλες λέξεις-κλειδιά δίνονται πιο κάτω ως σταθερές ή τελεστές.

#### Αναγνωριστικά (ID)

Συμβολοσειρές που αρχίζουν με προαιρετικό χαρακτήρα ‘\_’, ακολουθούμενο από αλφαριθμητικό χαρακτήρα, ακολουθούμενο από μηδέν ή περισσότερους αλφαριθμητικούς χαρακτήρες ή χαρακτήρες ‘\_’, και δεν αποτελούν λέξεις-κλειδιά. Ο χαρακτήρας ‘\_’ δεν μπορεί να είναι ο τελευταίος χαρακτήρας του αναγνωριστικού.

Αποδεκτά παραδείγματα:

a100\_\_version\_2  
\_a100\_\_version2

Μη αποδεκτά παραδείγματα<sup>1</sup>:

100\_\_version\_2  
a100\_\_version2\_  
\_100\_\_version\_2  
a100--version-2

#### Απλές σταθερές

Μη προσημασμένες ακέραιες (ICONST):

---

<sup>1</sup> Σε όλα τα μη αποδεκτά παραδείγματα που δίνονται, η συμβολοσειρά δεν αναγνωρίζεται συνολικά, είναι όμως δυνατό να αναγνωρίζονται μέρη αυτής ως ανεξάρτητες λεκτικές μονάδες.

Ο μοναδικός χαρακτήρας '0', που παριστάνει τη σταθερά με τιμή 0. Επίσης, ένας ή περισσότεροι αριθμητικοί χαρακτήρες, που ο πρώτος δεν είναι ο '0', οπότε η τιμή που παριστάνεται είναι ο αντίστοιχος αριθμός σε δεκαδική βάση. Επίσης, η συμβολοσειρά "0X" ακολουθούμενη από έναν ή περισσότερους αριθμητικούς χαρακτήρες που ο πρώτος δεν είναι ο '0', ή από έναν ή περισσότερους από τους αλφαβητικούς χαρακτήρες 'A', 'B', 'C', 'D', 'E' και 'F'. Στην περίπτωση αυτή, η τιμή που παριστάνεται είναι ο αντίστοιχος αριθμός – μετά το πρόθεμα "0X" – σε δεκαεξαδική βάση. Τέλος, η συμβολοσειρά "0B" ακολουθούμενη από έναν ή περισσότερους από τους αριθμητικούς χαρακτήρες '0' και '1' που ο πρώτος δεν είναι ο '0', οπότε η τιμή που παριστάνεται είναι ο αντίστοιχος αριθμός σε δυαδική βάση.

Αποδεκτά παραδείγματα:

```
0
180
0X9F0
0B1001
```

Μη αποδεκτά παραδείγματα:

```
0180
XB7
0X0
0B010
0XG8A
```

Μη προσημασμένες πραγματικές (RCONST):

Μηδέν ή περισσότεροι αριθμητικοί χαρακτήρες που ακολουθούνται από το χαρακτήρα '.' και τουλάχιστον έναν αριθμητικό χαρακτήρα. Ακολουθεί προαιρετικά πεδίο εκθέτη, που ξεκινάει με τον χαρακτήρα 'E', ακολουθούμενο από προαιρετικό πρόσημο και τουλάχιστον έναν αριθμητικό χαρακτήρα. Εναλλακτικά, μη προσημασμένη ακέραια σταθερά που ακολουθείται υποχρεωτικά από πεδίο εκθέτη. Αν δεν υπάρχει πεδίο εκθέτη, ο αριθμός μπορεί να είναι σε δεκαεξαδική βάση, με πρόθεμα "0X", ή σε δυαδική βάση, με πρόθεμα "0B". Το ακέραιο μέρος μιας πραγματικής σταθεράς, όπως και το αριθμητικό μέρος του εκθέτη, δε μπορεί να ξεκινά με '0' αν δεν είναι "0". Σε κάθε περίπτωση που υπάρχει κλασματικό μέρος, αυτό πρέπει να περιλαμβάνει τουλάχιστον ένα χαρακτήρα διαφορετικό από τον '0', εκτός αν είναι "0".

Αποδεκτά παραδείγματα:

```
180E-2
.5
180.100
7.0
0XA.9
0X.00B9CF
0B1.1001
```

Μη αποδεκτά παραδείγματα:

```
180E-2.2
.E-2
180E
1100
5.
7.00
.5G-2
05.2E-05
0XBE-2
```

Λογικές (BCONST):

Οι λέξεις-κλειδιά "TRUE" και "FALSE".

Χαρακτήρες (CCONST):

Οποιοσδήποτε εκτυπώσιμος ASCII χαρακτήρας (κωδικοί 32-126) μεταξύ δύο εμφανίσεων του ειδικού χαρακτήρα '\'. Επιπλέον, ειδικοί χαρακτήρες ASCII παριστάνονται με τη βοήθεια του χαρακτήρα '\'. Πιο συγκεκριμένα, ο χαρακτήρας LF (Line Feed) παριστάνεται ως

'\n', ο χαρακτήρας FF (Form Feed) ως '\f', ο χαρακτήρας HT (Horizontal Tab) ως '\t', ο χαρακτήρας CR (Carriage Return) ως '\r', ο χαρακτήρας BS (BackSpace) ως '\b' και ο χαρακτήρας VT (Vertical Tab) ως '\v'.

Αποδεκτά παραδείγματα:

```
'a'
'$'
','
'''
'\n'
'\'
```

Μη αποδεκτά παραδείγματα:

```
'ac'
'\p'
'\\'
```

### Τελεστές

Τελεστές σύγκρισης (RELOP): > >= < <= <>

Προσθετικοί τελεστές (ADDOP): + -

Λογικό Η (OROP): OR

Πολλαπλασιαστικοί τελεστές (MULDIVANDOP): \* / DIV MOD AND

Λογικό ΟΧΙ (NOTOP): NOT

Έλεγχος μέλους συνόλου (INOP): IN

### Στοιχεία λίστας (LISTFUNC)

Ένα 'A' και οσαδήποτε 'D', ή κανένα 'A' και τουλάχιστον ένα 'D', μεταξύ 'C' και 'R'.

Αποδεκτά παραδείγματα:

```
CAR
CDR
CADDR
```

Μη αποδεκτά παραδείγματα:

```
CARD
CDAR
```

### Ορμαθοί χαρακτήρων (STRING)

Οποιαδήποτε συμβολοσειρά μεταξύ δύο εμφανίσεων του ειδικού χαρακτήρα '"', συμπεριλαμβανομένης της κενής συμβολοσειράς. Ο χαρακτήρας '"' και οι πιο πάνω ειδικοί χαρακτήρες ASCII παριστάνονται σε έναν ορμαθό χαρακτήρων με τη βοήθεια του χαρακτήρα '\'. Με κάθε άλλη χρήση του χαρακτήρα '\' παριστάνεται ο χαρακτήρας που ακολουθεί. Έτσι, ο χαρακτήρας '\' καθαυτός παριστάνεται ως \\. Ειδικά όταν ο χαρακτήρας '\' βρίσκεται στο τέλος της γραμμής, ο ορμαθός συνεχίζεται στην επόμενη γραμμή, χωρίς οι χαρακτήρες '\' και αλλαγής γραμμής να αποτελούν μέρος αυτού.

Αποδεκτά παραδείγματα:

```
"CHARACTER +"
""
"STRINGS START AND END WITH \'"
"CHARACTER \\ AT THE END OF THE LINE \
EXTENDS STRING IN THE NEXT LINE\n"
```

### Άλλες λεκτικές μονάδες

Άλλοι χαρακτήρες ή ακολουθίες χαρακτήρων που αποτελούν ανεξάρτητες λεκτικές μονάδες είναι:

‘(’ (LPAREN), ‘)’ (RPAREN), ‘;’ (SEMI), ‘.’ (DOT), ‘,’ (COMMA), ‘=’ (EQU), ‘:’ (COLON), ‘[’ (LBRACK), ‘]’ (RBRACK), ‘:=’ (ASSIGN), ‘.’ (DOTDOT), (EOF)

Σε ορισμένες περιπτώσεις κάποιες από τις παραπάνω λεκτικές μονάδες ενεργούν ως τελεστές, όπως αυτό θα φανεί στη γραμματική και σημασιολογία της PASC320.

Η λεκτική μονάδα EOF – που ως χαρακτήρας ορίζεται με ειδικό τρόπο από το κάθε σύστημα – δεν εμφανίζεται στη γραμματική της PASC320, αλλά πρέπει να παράγεται από το λεκτικό αναλυτή με τιμή 0 για τον τερματισμό της συντακτικής ανάλυσης.

### Σχόλια

Τα σχόλια στην PASC320 είναι συμβολοσειρές που περικλείονται από το ζεύγος χαρακτήρων ‘{’, ‘}’.

## B. Συντακτικοί Κανόνες

Η γραμματική της PASC320 περιγράφεται από τους πιο κάτω κανόνες:

program → header declarations subprograms comp\_statement DOT

header → PROGRAM ID SEMI

declarations → constdefs typedefs vardefs

constdefs → CONST constant\_defs SEMI  
| ε

constant\_defs → constant\_defs SEMI ID EQU expression  
| ID EQU expression

expression → expression RELOP expression  
| expression EQU expression  
| expression INOP expression  
| expression OROP expression  
| expression ADDOP expression  
| expression MULDIVANDOP expression  
| ADDOP expression  
| NOTOP expression  
| variable  
| ID LPAREN expressions RPAREN  
| LENGTH LPAREN expression RPAREN  
| NEW LPAREN expression RPAREN  
| constant  
| LPAREN expression RPAREN  
| setlistexpression

variable → ID  
| variable DOT ID  
| variable LBRACK expressions RBRACK  
| LISTFUNC LPAREN expression RPAREN

expressions → expressions COMMA expression  
| expression

constant → ICONST  
| RCONST  
| BCONST  
| CCONST

```

setlistexpression → LBRACK expressions RBRACK
                  | LBRACK RBRACK

typedefs → TYPE type_defs SEMI
          | ε

type_defs → type_defs SEMI ID EQU type_def
           | ID EQU type_def

type_def → ARRAY LBRACK dims RBRACK OF typename
          | LIST OF typename
          | SET OF typename
          | RECORD fields END
          | limit DOTDOT limit

dims → dims COMMA limits
      | limits

limits → limit DOTDOT limit
        | ID

limit → sign ICONST
       | CCONST
       | BCONST
       | ADDOP ID
       | ID

sign → ADDOP | ε

typename → standard_type
          | ID

standard_type → INTEGER | REAL | BOOLEAN | CHAR

fields → fields SEMI field
        | field

field → identifiers COLON typename

identifiers → identifiers COMMA ID
             | ID

vardefs → VAR variable_defs SEMI
         | ε

variable_defs → variable_defs SEMI identifiers COLON typename
               | identifiers COLON typename

subprograms → subprograms subprogram SEMI
              | ε

subprogram → sub_header SEMI FORWARD
             | sub_header SEMI declarations subprograms comp_statement

sub_header → FUNCTION ID formal_parameters COLON standard_type
            | FUNCTION ID formal_parameters COLON LIST
            | PROCEDURE ID formal_parameters
            | FUNCTION ID

```

```

formal_parameters → LPAREN parameter_list RPAREN
                  | ε

parameter_list → parameter_list SEMI pass identifiers COLON typename
               | pass identifiers COLON typename

pass → VAR | ε

comp_statement → BEGIN statements END

statements → statements SEMI statement
           | statement

statement → assignment
          | if_statement
          | case_statement
          | while_statement
          | for_statement
          | with_statement
          | subprogram_call
          | io_statement
          | comp_statement
          | ε

assignment → variable ASSIGN expression
           | variable ASSIGN STRING

if_statement → IF expression THEN statement if_tail

if_tail → ELSE statement
        | ε

case_statement → CASE expression OF cases case_tail END

cases → cases SEMI single_case
      | single_case

single_case → label_list COLON statement
            | ε

label_list → label_list COMMA label
           | label

label → sign constant
      | sign ID

case_tail → SEMI OTHERWISE COLON statement
          | ε

while_statement → WHILE expression DO statement

for_statement → FOR ID ASSIGN iter_space DO statement

iter_space → expression TO expression
           | expression DOWNTO expression

with_statement → WITH variable DO statement

subprogram_call → ID

```

```

| ID LPAREN expressions RPAREN

io_statement → READ LPAREN read_list RPAREN
              | WRITE LPAREN write_list RPAREN

read_list → read_list COMMA read_item
           | read_item

read_item → variable

write_list → write_list COMMA write_item
           | write_item

write_item → expression
           | STRING

```

όπου το σύμβολο ‘|’ διαχωρίζει τα εναλλακτικά δεξιά μέλη των κανόνων και ε είναι η κενή συμβολοσειρά.

Οι παραπάνω κανόνες ορίζουν διφορούμενη γραμματική, που με κατάλληλους μετασχηματισμούς ή βοηθητικές περιγραφές προτεραιότητας και προσεταιριστικότητας τελεστών μπορεί να γίνει μη διφορούμενη.

Αρχικό σύμβολο της γραμματικής της PASC320 αποτελεί το “program”.

## **Γ. Σημασιολογία**

Η σημασιολογία της PASC320 καθορίζεται από μια σειρά κανόνων που αφορούν τη δομή ενός *ορθού* προγράμματος. Κάποιοι από τους κανόνες αυτούς είναι πιθανό να καλύπτονται από τη σύνταξη της γλώσσας, ενώ κάποιοι άλλοι μπορούν να χρησιμοποιηθούν για τη μετατροπή της γραμματικής της PASC320 σε μη διφορούμενη. Όλοι οι υπόλοιποι κανόνες κωδικοποιούνται στο μεταγλωττιστή της PASC320 με τη βοήθεια σημασιολογικών ρουτινών που εκτελούνται κατά τη μετατροπή ενός αρχικού προγράμματος σε ενδιάμεσο κώδικα.

### **Τύποι δεδομένων**

Η PASC320 υποστηρίζει τέσσερεις βασικούς τύπους δεδομένων:

- integer : ο αριθμητικός τύπος των ακέραιων αριθμών,
- real : ο αριθμητικός τύπος των πραγματικών αριθμών,
- char : ο τύπος των ASCII χαρακτήρων, και
- boolean : ο τύπος των λογικών τιμών «Αληθής» και «Ψευδής».

Το μέγεθος και η αναπαράσταση των δεδομένων καθενός από τους βασικούς αριθμητικούς τύπους της PASC320 καθορίζονται από την τελική γλώσσα μεταγλώττισης. Αυτή επιτρέπει μεγέθη τύπων και αναπαραστάσεις ανάλογα με την υποστήριξη που δίνει η αρχιτεκτονική για τον καθένα.

Τα δεδομένα του τύπου char καταλαμβάνουν τον ελάχιστο χώρο αποθήκευσης που επιτρέπει η τελική γλώσσα, συνήθως ένα byte. Η αναπαράστασή τους γίνεται με τον κώδικα ASCII.

Όσο αφορά τον τύπο boolean, το μέγεθος των δεδομένων του είναι το ίδιο με το μέγεθος των δεδομένων του τύπου char, ενώ η αναπαράστασή τους γίνεται με τη σταθερά 0 για την τιμή «Ψευδής» και τη σταθερά 1 για την τιμή «Αληθής».

Η κωδικοποίηση των δύο παραπάνω τύπων προσδίδει μια φυσική διάταξη στις τιμές τους, και γι’ αυτό θεωρούνται *διατεταγμένοι* τύποι.

Η ευθυγράμμιση που απαιτείται για την προσπέλαση δεδομένων καθενός από τους πιο πάνω τύπους της PASC320 καθορίζεται από την τελική γλώσσα.

Εκτός από τους πιο πάνω βασικούς, η PASC320 υποστηρίζει και πέντε ακόμα τύπους.

Ο τύπος υποπεριοχής είναι ένας ακόμα βαθμωτός τύπος που ορίζεται ως υποσύνολο ενός από τους υπόλοιπους βαθμωτούς τύπους, εκτός από τύπο `real`.

Παραδείγματα υποπεριοχής είναι τα παρακάτω:

```
-128 .. 127
'A' .. 'F'
```

όπου η πρώτη υποπεριοχή είναι υποσύνολο του τύπου `integer`, ενώ η δεύτερη υποπεριοχή είναι υποσύνολο του τύπου `char`.

Οι τιμές μεταβλητών του τύπου υποπεριοχής λαμβάνονται μέσα από το αντίστοιχο υποσύνολο και κωδικοποιούνται όπως οι τιμές του τύπου από τον οποίο παράγονται.

Η PASC320 υποστηρίζει και τέσσερεις σύνθετους τύπους, τους τύπους πίνακα, συνόλου και λίστας, με στοιχεία δεδομένα του ίδιου τύπου, και τον τύπο εγγραφής, με στοιχεία δεδομένα όχι αναγκαστικά του ίδιου τύπου, που λέγονται και πεδία της εγγραφής. Οι τύποι στοιχείων πίνακα, συνόλου, λίστας ή εγγραφής δεν είναι απαραίτητο να είναι βασικοί. Ειδικότερα, τα στοιχεία μιας λίστας δε μπορούν να είναι τύπου πίνακα ή συνόλου.

Ένα στοιχείο πίνακα αναπαριστάται με το όνομα του πίνακα και μέσα σε αγκύλες μια λίστα από δείκτες, που αντιστοιχούν στις διαστάσεις του πίνακα. Δείκτες διαδοχικών διαστάσεων διαχωρίζονται με το σύμβολο `'.'`.

Παραδείγματα αναφοράς σε στοιχεία πίνακα είναι τα εξής:

```
C[103,5]
C[0,5,'k']
C[-1,false]
```

όπου `false` είναι σταθερά τύπου `boolean`. Ο τύπος του δείκτη πρέπει να είναι βαθμωτός εκτός από `real`, με όρια τιμών που ορίζονται κατά τη δήλωση του πίνακα. Έτσι, η παρακάτω αναφορά σε στοιχείο πίνακα δεν είναι αποδεκτή:

```
C[3.5]
```

Δεικτοδότηση με τη βοήθεια αναγνωριστικού επιτρέπεται μόνο αν ο τύπος δείκτη που προκύπτει είναι βαθμωτός εκτός από `real`. Έτσι, η αναφορά:

```
C[i,j+2]
```

δε θα είναι αποδεκτή, αν ο τύπος του αναγνωριστικού `i` και της έκφρασης `"j+2"` είναι μη βαθμωτός ή `real`.

Σε αντίθεση με τα στοιχεία πίνακα, αναφορά στα οποία γίνεται με δεικτοδότηση, αναφορά στα πεδία του τύπου εγγραφής γίνεται ονομαστικά. Έτσι, ένα πεδίο εγγραφής αναπαριστάται με το όνομα της εγγραφής, ακολουθούμενο από το σύμβολο `'.'` και το όνομα του πεδίου.

Παράδειγμα αναφοράς σε πεδίο εγγραφής είναι το εξής:

```
D.str
```

Ένα πεδίο εγγραφής μπορεί να αναπαρασταθεί και χωρίς το όνομα της εγγραφής, με τη χρήση ειδικής εντολής της PASC320, όπως θα δούμε αργότερα.

Για τον τύπο συνόλου, δεν ορίζεται αναφορά σε μεμονωμένα στοιχεία συνόλων, καθώς δεν υπάρχει ούτε δεικτοδότηση, αλλά ούτε και ονομασία αυτών. Μεταβλητές τύπου συνόλου χρησιμοποιούνται αυτούσιες σε εκφράσεις, από τις οποίες έμμεσα προκύπτουν στοιχεία τους, όπως θα δούμε παρακάτω.

Όσο αφορά τον τύπο λίστας, τέλος, αναφορά στα στοιχεία του γίνεται με τη βοήθεια ειδικών συναρτήσεων στοιχείων λίστας. Οι συναρτήσεις αυτές ορίζονται παρακάτω. Μια ολόκληρη λίστα στοιχείων βασικού τύπου μπορεί επίσης να παρασταθεί και ως έκφραση λίστας με τη βοήθεια αγκυλών. Για παράδειγμα, μια λίστα ακεραίων μπορεί να είναι η:

```
[1,3,-5,0,2]
```

Οι τύποι πίνακα και εγγραφής μπορούν να συνδυάζονται μεταξύ τους. Έτσι, παραδείγματα αναφοράς σε στοιχεία πιο πολύπλοκων σύνθετων τύπων είναι τα πιο κάτω:

```
C[n+1,i].x
A.D.z[0,2][j]
```

Γενικά δεν υπάρχει περιορισμός σε φωλιάσματα πινάκων και εγγραφών. Τα σύνολα από την άλλη μεριά, μπορούν να είναι στοιχεία πινάκων ή πεδία εγγραφών, αλλά οι πίνακες και οι εγγραφές δε μπορούν να είναι στοιχεία συνόλων. Οι λίστες μπορούν να είναι στοιχεία πινά-



κών ή πεδία εγγραφών, αλλά οι πίνακες δε μπορούν να είναι στοιχεία λίστας, ενώ λίστες και σύνολα δεν φωλιάζονται άμεσα μεταξύ τους. Συνολικά, φωλιάσματα τύπων επιτρέπονται μέχρι ένα μέγιστο βάθος 5 τύπων.

Η αποθήκευση των στοιχείων ενός πίνακα της PASC320 στη μνήμη γίνεται σε συνεχόμενες θέσεις, ανάλογα με την ευθυγράμμιση του τύπου αυτών, με αύξουσα σειρά μεταβολής των δεικτών από την τελευταία διάσταση προς την πρώτη. Για παράδειγμα, ένας τρισδιάστατος πίνακας  $A_{3 \times 3 \times 2}$  στοιχείων με τιμές δεικτών για κάθε διάστασή του στις περιοχές ακεραίων από -1 έως 1, από 0 έως 2 και από 0 έως 1, αντίστοιχα, αποθηκεύει τα στοιχεία του με τη σειρά:  $A[-1,0,0]$ ,  $A[-1,0,1]$ ,  $A[-1,1,0]$ ,  $A[-1,1,1]$ ,  $A[-1,2,0]$ ,  $A[-1,2,1]$ ,  $A[0,0,0]$ ,  $A[0,0,1]$ ,  $A[0,1,0]$ ,  $A[0,1,1]$ ,  $A[0,2,0]$ ,  $A[0,2,1]$ ,  $A[1,0,0]$ ,  $A[1,0,1]$ ,  $A[1,1,0]$ ,  $A[1,1,1]$ ,  $A[1,2,0]$  και  $A[1,2,1]$ . Με τον τρόπο αυτό, ένας δισδιάστατος πίνακας αποθηκεύεται γραμμή-γραμμή.

Ένας σύνθετος τύπος πίνακα καταλαμβάνει τόσες θέσεις αποθήκευσης, όσος είναι ο αριθμός των στοιχείων του επί το μέγεθος του ευθυγραμμισμένου στοιχείου του.

Η αποθήκευση μιας μεταβλητής συνόλου της PASC320 στη μνήμη γίνεται με τη μορφή ψηφιοδιανύσματος, όπου κάθε ψηφίο αντιστοιχεί σε ένα από τα δυνατά στοιχεία του συνόλου, και έχει τιμή '1' αν το στοιχείο αυτό είναι παρόν στο σύνολο, διαφορετικά έχει τιμή '0'. Για πρακτικούς λόγους, ο μέγιστος αριθμός στοιχείων ενός συνόλου είναι περιορισμένος. Η PASC320 θέτει το όριο αυτό στον αριθμό στοιχείων του τύπου `char`, που για κωδικοποίηση ASCII, θα είναι 128 στοιχεία. Έτσι, επιτρέπονται σύνολα στοιχείων τύπου `char`, όπως και τύπου `boolean`, αλλά όχι σύνολα στοιχείων τύπου `integer` ή `real`. Επιτρέπονται όμως και άλλα βαθμωτά στοιχεία, όπως θα δούμε παρακάτω, αν ο τύπος τους περιλαμβάνει μέχρι 128 διακριτές τιμές. Συνολικά επομένως, ανάλογα με το μέγιστο αριθμό στοιχείων τους, τα σύνολα απαιτούν από 1 έως 16 bytes για την αποθήκευσή τους στη μνήμη.

Η αποθήκευση μιας λίστας της PASC320 στη μνήμη γίνεται με δυναμικό τρόπο. Ένα στοιχείο λίστας αποθηκεύεται ως ένα διατεταγμένο ζεύγος (περιεχόμενο, διεύθυνση επόμενου στοιχείου). Μια μεταβλητή τύπου λίστας έχει μέγεθος όσο το μέγεθος μιας διεύθυνσης μνήμης και τιμή τη διεύθυνση του πρώτου στοιχείου της λίστας. Το τελευταίο στοιχείο μιας λίστας έχει τιμή 0 ως διεύθυνση επόμενου στοιχείου. Μια έκφραση λίστας δημιουργεί μια ακολουθία τέτοιων ζευγών. Οι διευθύνσεις στοιχείων παράγονται από κάποιο μηχανισμό δυναμικής εκχώρησης μνήμης, και ως εκ τούτου διαδοχικά στοιχεία δεν αποθηκεύονται κατ' ανάγκη στη σειρά. Κάθε στοιχείο λίστας καταλαμβάνει τόσες θέσεις μνήμης, όσο είναι το άθροισμα των θέσεων του ευθυγραμμισμένου περιεχομένου του και της ευθυγραμμισμένης διεύθυνσης επόμενου στοιχείου.

Η αποθήκευση των πεδίων μιας εγγραφής της PASC320 στη μνήμη γίνεται σε συνεχόμενες θέσεις, ανάλογα με την ευθυγράμμιση αυτών, και με τη σειρά που αυτά απαντώνται στη δήλωση του τύπου της εγγραφής.

Ένας σύνθετος τύπος εγγραφής καταλαμβάνει τόσες θέσεις αποθήκευσης, όσο είναι το άθροισμα των θέσεων που καταλαμβάνουν τα ευθυγραμμισμένα πεδία του.

### Δομή ενός προγράμματος PASC320

Ένα πρόγραμμα σε γλώσσα PASC320 αποτελείται από την κύρια μονάδα και μονάδες υποπρογραμμάτων. Οι μονάδες υποπρογραμμάτων δηλώνονται μέσα στην κύρια μονάδα, ή φωλιασμένες σε άλλες μονάδες υποπρογραμμάτων. Οι μονάδες υποπρογραμμάτων έχουν την ίδια δομή με την κύρια μονάδα.

Κάθε μονάδα περιέχει:

- Επικεφαλίδα: Η επικεφαλίδα της μονάδας καθορίζει ανάμεσα στα άλλα τις παραμέτρους της, δηλαδή τον τρόπο επικοινωνίας αυτής με τις άλλες μονάδες του προγράμματος.
- Δηλώσεις σταθερών, τύπων και μεταβλητών (προαιρετικά): Οι δηλώσεις αυτές έχουν εμβέλεια τη μονάδα.

- Φωλιασμένες δηλώσεις υποπρογραμμάτων (προαιρετικά): Και αυτές οι δηλώσεις έχουν εμβέλεια τη μονάδα, αλλά επιπλέον ορίζουν εσωτερικές εμβέλειες σε αυτήν.
- Μια σύνθετη εντολή: Αυτή είναι μια σειρά απλών, δομημένων ή άλλων σύνθετων εντολών μεταξύ των λέξεων-κλειδιά “begin” και “end”. Η σειρά αυτή μπορεί να είναι κενή, όπως φαίνεται και στους συντακτικούς κανόνες της γλώσσας. Η σύνθετη εντολή της PASC320 δεν περιέχει δηλώσεις.

Η εκτέλεση του κώδικα μιας μονάδας ξεκινάει με την πρώτη και τερματίζεται με την τελευταία εντολή της σύνθετης εντολής της μονάδας.

### Επικεφαλίδα

Η επικεφαλίδα της κύριας μονάδας περιλαμβάνει τη λέξη-κλειδί “program”, για να τη διαχωρίσει από τις επικεφαλίδες υποπρογραμμάτων, και το όνομά της, που είναι και το πρώτο αναγνωριστικό καθολικής εμβέλειας του προγράμματος.

Αν και στην κλασική PASCAL η κύρια μονάδα έχει παραμέτρους που καθορίζουν την επικοινωνία αυτής με το λειτουργικό σύστημα, στην PASC320 η κύρια μονάδα δεν έχει παραμέτρους.

Πληροφορίες για επικεφαλίδες υποπρογραμμάτων δίνονται σε επόμενη παράγραφο.

### Δηλώσεις σταθερών, τύπων και μεταβλητών

Με μια τέτοια δήλωση αποδίδεται σε ένα αναγνωριστικό μια σταθερά, ένας τύπος ή μια μεταβλητή του προγράμματος. Κάθε τέτοια δήλωση εισάγει το αναγνωριστικό στον πίνακα συμβόλων, με εμβέλεια την τρέχουσα μονάδα. Ένα αναγνωριστικό μπορεί να δηλωθεί ξανά, πιθανά με άλλο τρόπο, σε εσωτερική εμβέλεια της μονάδας, οπότε η νέα δήλωση επισκιάζει την παλιά. Ένα αναγνωριστικό δε μπορεί να δηλωθεί για δεύτερη φορά στην ίδια εμβέλεια.

Μέσα στην ίδια μονάδα οι δηλώσεις σταθερών προηγούνται των υπολοίπων, ενώ οι δηλώσεις τύπων προηγούνται των δηλώσεων μεταβλητών. Οι δηλώσεις σταθερών, τύπων και μεταβλητών διαχωρίζονται μεταξύ τους, αλλά και με τη σύνθετη εντολή της μονάδας – αν δεν ακολουθούν δηλώσεις υποπρογραμμάτων, με το χαρακτήρα ‘;’.

#### Δηλώσεις σταθερών

Μια δήλωση σταθεράς στην PASC320 αποδίδει σε ένα αναγνωριστικό μια σταθερή τιμή, μια τιμή δηλαδή που δεν επιτρέπεται να αλλάξει με κάποια ανάθεση μέσα στην εμβέλεια στην οποία υπάρχει η δήλωση. Η σταθερή τιμή μπορεί να δίνεται είτε ως μία λεκτική μονάδα σταθεράς είτε ως έκφραση.

Ο τύπος της σταθεράς πρέπει να είναι βασικός, αλλά δεν ορίζεται ρητά μέσα από τη δήλωση. Έτσι, ο σημασιολογικός αναλυτής πρέπει να βρει και να αποδώσει τον κατάλληλο τύπο στο αναγνωριστικό της σταθεράς.

Παραδείγματα δηλώσεων σταθερών είναι τα παρακάτω:

```
const a = 10; b = -a * 3;
const x = -3.0; y = b * x; z = true; w = '0';
```

Ο σημασιολογικός αναλυτής πρέπει να αποδώσει τον τύπο integer στα αναγνωριστικά a και b, τον τύπο real στα αναγνωριστικά x και y, τον τύπο boolean στο αναγνωριστικό z και τον τύπο char στο αναγνωριστικό w.

Ας σημειωθεί ότι η λέξη-κλειδί “const” εμφανίζεται μόνο στην πρώτη από τις δηλώσεις σταθερών μιας μονάδας.

Σε περίπτωση που η σταθερά ορίζεται με τη βοήθεια έκφρασης, πρέπει να ακολουθούνται οι κανόνες που διέπουν τις εκφράσεις, όπως αυτοί περιγράφονται πιο κάτω. Επιπλέον, η έκφραση πρέπει να δίνει σταθερή τιμή, κάτι που θα συμβαίνει αν σε αυτή συμμετέχουν μόνο στα-

θερές, είτε ως λεκτικές μονάδες σταθερών, είτε ως άλλες δηλωμένες σταθερές, ορατές στο σημείο της παρούσας δήλωσης. Τη σταθερή τιμή της έκφρασης πρέπει να προσδιορίζει ο σημασιολογικός αναλυτής και να αποδίδει στο αναγνωριστικό της νέας σταθεράς.

### Δηλώσεις τύπων

Μια δήλωση τύπου στην PASC320 αποδίδει σε ένα αναγνωριστικό τη δομή που περιγράφει έναν τύπο υποπεριοχής ή ένα σύνθετο τύπο.

Η PASC320 επιβάλλει στον προγραμματιστή να δηλώνει όλους τους μη βασικούς τύπους που χρησιμοποιεί. Μετά από μια δήλωση, ο τύπος θεωρείται γνωστός, κι επομένως το όνομά του μπορεί να χρησιμοποιηθεί σε επόμενη δήλωση. Επειδή ένα όνομα τύπου δε μπορεί να χρησιμοποιηθεί πριν τη δήλωσή του, δεν υλοποιούνται κυκλικές δομές δεδομένων.

Παραδείγματα δηλώσεων τύπων είναι τα παρακάτω:

```
type day = 1..7;
type ar = array[-1..M, day] of boolean;
type arr = record
    num, fno : integer;
    data : ar
end;
type weekdays = set of day;
type arrl = list of arr;
```

όπου η δεύτερη και η τέταρτη δήλωση πρέπει να βρίσκονται μέσα στην ορατότητα της πρώτης, η τρίτη πρέπει να βρίσκεται μέσα στην ορατότητα της δεύτερης, ενώ η πέμπτη δήλωση πρέπει να βρίσκεται στην ορατότητα της τρίτης.

Όπως και με τις δηλώσεις σταθερών, η λέξη-κλειδί “type” εμφανίζεται μόνο στην πρώτη από τις δηλώσεις τύπων μιας μονάδας.

Πιο συγκεκριμένα, για δηλώσεις τύπων υποπεριοχής:

1. Τα όρια μιας υποπεριοχής ορίζονται με σταθερές ή αναγνωριστικά σταθερών που έχουν νωρίτερα δηλωθεί και είναι ορατά στο σημείο της δήλωσης της υποπεριοχής, και διαχωρίζονται μεταξύ τους με τη λεκτική μονάδα “..”. Το κάτω όριο τοποθετείται πριν, και το άνω όριο τοποθετείται μετά τη μονάδα διαχωρισμού.
2. Το κάτω όριο μιας υποπεριοχής πρέπει να έχει τιμή μικρότερη από ή ίση με την τιμή του πάνω ορίου.
3. Ένα όριο υποπεριοχής μπορεί να είναι προσημασμένο, αρκεί ο τύπος της σταθεράς ή του αναγνωριστικού που τον ορίζει να δέχεται πρόσημο, δηλαδή να μην είναι boolean ή char.

Για δηλώσεις τύπων πίνακα:

1. Οι δυνατές τιμές των δεικτών ενός πίνακα για κάθε διάστασή του δίνονται σε αγκύλες, μετά τη λέξη-κλειδί “array”. Οι τιμές δεικτών διαδοχικών διαστάσεων διαχωρίζονται μεταξύ τους με το χαρακτήρα ‘,’.
2. Οι τιμές που μπορεί να πάρει ο δείκτης μιας διάστασης του πίνακα δίνονται με μια περιγραφή ακέραιας υποπεριοχής, όπου ως ακέραια νοείται η υποπεριοχή τιμών τύπου integer, char ή boolean. Εναλλακτικά μπορούν να δοθούν με έναν τύπο ακέραιας υποπεριοχής που έχει δηλωθεί νωρίτερα.
3. Ο τύπος των στοιχείων του πίνακα δίνεται με όνομα τύπου, ο οποίος πρέπει να είναι βασικός ή να έχει οριστεί νωρίτερα, και αναγράφεται μετά τη λέξη-κλειδί “of”.

Για δηλώσεις τύπων συνόλου:

1. Ο τύπος των στοιχείων του συνόλου δίνεται με όνομα τύπου, ο οποίος πρέπει να είναι βασικός – εκτός από integer ή real – ή να είναι τύπος υποπεριοχής που έχει οριστεί νωρίτερα, και αναγράφεται μετά τη λέξη-κλειδί “of”.

2. Αν ο τύπος στοιχείων ενός συνόλου δεν είναι βασικός, τότε θα πρέπει να περιλαμβάνει μέχρι 128 το πολύ διακριτές τιμές. Για παράδειγμα, η υποπεριοχή ακεραίων 1..999 δε μπορεί να ορίζει στοιχεία συνόλου.

Για δηλώσεις τύπων λίστας:

1. Ο τύπος των στοιχείων της λίστας δίνεται με όνομα τύπου, ο οποίος πρέπει να είναι βασικός, ή άλλος τύπος – εκτός από πίνακα ή συνόλου – που έχει οριστεί νωρίτερα, και αναγράφεται μετά τη λέξη-κλειδί “of”.

Για δηλώσεις τύπων εγγραφής:

1. Τα πεδία της εγγραφής περιγράφονται μεταξύ των λέξεων-κλειδιά “record” και “end”, ενώ διαδοχικά πεδία μιας εγγραφής διαχωρίζονται μεταξύ τους με το χαρακτήρα ‘;’. Μια εγγραφή πρέπει να έχει τουλάχιστον ένα πεδίο.
2. Η δήλωση ενός πεδίου γίνεται με το αναγνωριστικό του και το όνομα του τύπου του, που πρέπει να είναι βασικός ή να έχει δηλωθεί νωρίτερα. Το διαχωριστικό ‘:’ παρεμβάλλεται μεταξύ του ονόματος του πεδίου και του ονόματος του τύπου του.
3. Πολλαπλά πεδία του ίδιου τύπου μπορούν να δηλώνονται μαζί, όπως φαίνεται στο παράδειγμα.

### Δηλώσεις μεταβλητών

Μια δήλωση μεταβλητής μοιάζει με δήλωση πεδίου εγγραφής, και αποδίδει σε ένα αναγνωριστικό μια μεταβλητή της μονάδας, ενός βασικού τύπου ή τύπου που έχει δηλωθεί νωρίτερα.

Παραδείγματα δηλώσεων μεταβλητών είναι τα παρακάτω:

```
var i,j,k : integer;
var l: boolean; dev: real; x: char; w: integer;
var c: complex; a: c_array;
```

όπου complex και c\_array είναι τύποι που έχουν δηλωθεί νωρίτερα, και η ορατότητά τους περιλαμβάνει την παρούσα μονάδα. Όπως και με τις δηλώσεις σταθερών και τύπων, η λέξη-κλειδί “var” εμφανίζεται μόνο στην πρώτη από τις δηλώσεις μεταβλητών μιας μονάδας.

Κάθε δήλωση γίνεται για ένα μόνο τύπο. Περισσότερες δηλώσεις για τον ίδιο τύπο μπορούν να ακολουθούν πιο κάτω.

Όλες οι μεταβλητές της PASC320 – εκτός από τις μεταβλητές της κύριας μονάδας του προγράμματος – είναι μεταβλητές στοίβας. Οι μεταβλητές της κύριας μονάδας είναι στατικές καθολικής εμβέλειας, κι επομένως δεν τοποθετούνται στη στοίβα, αλλά έχουν σταθερές διευθύνσεις έξω από αυτή, έχουν δε διάρκεια ζωής όλη τη διάρκεια εκτέλεσης του προγράμματος.

### Δηλώσεις υποπρογραμμάτων

Υποπρογράμματα στην PASC320 μπορούν να δηλώνονται μέσα σε οποιαδήποτε μονάδα του προγράμματος, αμέσως μετά τις δηλώσεις σταθερών, τύπων και μεταβλητών. Έχουν δε εμβέλεια παρόμοια με αυτή των υπόλοιπων δηλώσεων της μονάδας και ο έλεγχος της δήλωσης του ονόματος ενός υποπρογράμματος είναι παρόμοιος με τον έλεγχο δηλώσεων των σταθερών, τύπων και μεταβλητών της PASC320. Οι δηλώσεις υποπρογραμμάτων διαχωρίζονται μεταξύ τους, αλλά και με τη σύνθετη εντολή της μονάδας, με το χαρακτήρα ‘;’.

Τα υποπρογράμματα της PASC320 διακρίνονται σε διαδικασίες (procedures) και συναρτήσεις (functions). Οι διαδικασίες καλούνται μέσα από ειδική εντολή της PASC320, ενώ οι συναρτήσεις μέσα από εκφράσεις. Οι συναρτήσεις επιστρέφουν αποτελέσματα μέσα από τα ονόματά τους που για το σκοπό αυτό λειτουργούν και σαν τοπικές μεταβλητές τους.

Κάθε υποπρόγραμμα είναι μια μονάδα, και η δήλωσή του περιέχει ολόκληρη τη δομή του, σύμφωνα με τη δομή μονάδας που περιγράφηκε πιο πάνω. Ειδικά όσο αφορά την επικεφαλίδα ενός υποπρογράμματος της PASC320, αυτή περιέχει με τη σειρά:

- (α) το είδος του υποπρογράμματος,
- (β) το όνομά του, που πρέπει να είναι μοναδικό στην εμβέλεια στην οποία βρίσκεται η δήλωση,
- (γ) τις τυπικές παραμέτρους του και τον τρόπο περάσματος αυτών μέσα σε παρενθέσεις, και
- (δ) τον τύπο του αποτελέσματος, εάν το υποπρόγραμμα είναι συνάρτηση.

Παραδείγματα επικεφαλίδων υποπρογραμμάτων είναι τα εξής:

```
function LL (y: real; var w: c_array): real;
function F (x: i_list): list;
procedure A(m,n: integer; x: x_type; var z: char);
procedure S;
```

Ειδικότερα:

1. Η δήλωση μιας τυπικής παραμέτρου περιλαμβάνει το όνομα του τύπου της, ο οποίος μπορεί να είναι βασικός ή να έχει δηλωθεί νωρίτερα.
2. Το πέρασμα μιας παραμέτρου γίνεται κατ' αναφορά, εάν της δήλωσης της παραμέτρου προηγείται η λέξη-κλειδί "var", διαφορετικά γίνεται κατ' αξία.
3. Αν ο τύπος μιας τυπικής παραμέτρου είναι σύνθετος, τότε η παράμετρος *πρέπει* να μεταδίδεται κατ' αναφορά.
4. Κάθε τυπική παράμετρος που μεταδίδεται κατ' αξία αποτελεί τοπική μεταβλητή της μονάδας του υποπρογράμματος και αποθηκεύεται στη στοίβα.
5. Μια διαδικασία μπορεί να μην έχει παραμέτρους, μια συνάρτηση όμως πρέπει να έχει τουλάχιστον μία παράμετρο.
6. Ο τύπος του αποτελέσματος μιας συνάρτησης πρέπει να είναι βασικός ή λίστας. Στην τελευταία περίπτωση, ως τύπος αποτελέσματος αναφέρεται μόνο η λέξη-κλειδί list και δεν ορίζεται ο τύπος στοιχείου της λίστας, ενώ τιμή αποτελέσματος θα είναι η διεύθυνση του πρώτου στοιχείου της λίστας.

Μια επικεφαλίδα μονάδας διαχωρίζεται από τις δηλώσεις ή τη σύνθετη εντολή της – αν δεν υπάρχουν δηλώσεις – με το χαρακτήρα ';'.

Σε αντίθεση με τις δηλώσεις τύπων, όπου ένα όνομα τύπου χρησιμοποιείται μετά τη δήλωσή του, σε μια δήλωση υποπρογράμματος μπορεί να χρησιμοποιηθεί το ίδιο το υποπρόγραμμα. Αυτό, σε συνδυασμό με το γεγονός ότι οι τοπικές μεταβλητές ενός υποπρογράμματος είναι μεταβλητές στοίβας, επιτρέπει στην PASC320 την υποστήριξη αναδρομής.

Για μεγαλύτερη ευελιξία και συγκεκριμένα για την υποστήριξη έμμεσης αναδρομής, η PASC320 – σύμφωνα με την κλασική PASCAL – επιτρέπει προαναγγελία σε μια δήλωση υποπρογράμματος, ως εξής:

1. Ένα υποπρόγραμμα – διαδικασία ή συνάρτηση – μπορεί να δηλωθεί με τη λέξη-κλειδί "forward", η οποία ακολουθεί την επικεφαλίδα και αναβάλλει την υπόλοιπη δήλωση γι' αργότερα.
2. Στο σημείο που επιθυμεί ο προγραμματιστής, το οποίο θα πρέπει να βρίσκεται στην ίδια μονάδα και να ακολουθεί την προηγούμενη δήλωση, συμπληρώνεται η δήλωση του υποπρογράμματος. Για το σκοπό αυτό, ξαναδίνεται μια σύντομη επικεφαλίδα, η οποία περιλαμβάνει μόνο το είδος και το όνομα του υποπρογράμματος.

Ένα παράδειγμα δήλωσης συνάρτησης με προαναγγελία είναι το πιο κάτω απόσπασμα από το χώρο δηλώσεων υποπρογραμμάτων μιας μονάδας:

```
function X (n:integer) : list; forward;
... { ένα ή περισσότερα άλλα υποπρογράμματα που πιθανά καλούν την X }
function X;
{ η υπόλοιπη δήλωση της X }
```

Ο σημασιολογικός αναλυτής πρέπει να εξετάζει αν οι προαναγγελίες δηλώσεων υποπρογραμμάτων γίνονται σύμφωνα με τα παραπάνω, αν δηλαδή (α) μετά από μια προαναγγελία υπάρχει όντως η συνέχεια της δήλωσης, και (β) αν μια δήλωση υποπρογράμματος με σύντομη επικεφαλίδα αντιστοιχεί σε προηγούμενη προαναγγελία.

Σύμφωνα με τη σύνταξη της PASC320 επιτρέπονται φωλιασμένες δηλώσεις υποπρογραμμάτων. Επειδή κάθε υποπρόγραμμα της PASC320 ορίζει μια εμβέλεια, κάθε αναφορά σε ένα αναγνωριστικό που δεν είναι τοπικό στην μονάδα στην οποία βρίσκεται η αναφορά, πρέπει να επιλύεται σε άλλη μονάδα. Καθώς το δέσιμο αναγνωριστικών στην PASC320 είναι στατικό, η επίλυση αναφορών γίνεται ακολουθώντας προς τα έξω τις φωλιασμένες μονάδες του προγράμματος.

Το μέγιστο βάθος φωλιάσματος υποπρογραμμάτων στην PASC320 είναι 10.

### Εκφράσεις

Η PASC320 υποστηρίζει αριθμητικές εκφράσεις, λογικές εκφράσεις, εκφράσεις συνόλων και εκφράσεις λίστας. Οι τιμές των πρώτων μπορούν να είναι τύπου integer ή real, οι τιμές των δεύτερων μπορούν να είναι μόνο τύπου boolean, ενώ οι τιμές των υπολοίπων είναι σύνθετου τύπου συνόλου ή λίστας αντίστοιχα. Οι εκφράσεις αποτιμώνται με βάση συγκεκριμένους κανόνες και με τη βοήθεια των τελεστών της PASC320. Τόσο οι αριθμητικές, όσο και οι εκφράσεις συνόλων, μπορούν να περιέχονται σε λογικές εκφράσεις, αλλά όχι το αντίστροφο.

Εκτός από τις πιο πάνω, ορίζονται και απλές εκφράσεις τύπου char, οι οποίες είτε εμφανίζονται μεμονωμένα σε κλήσεις υποπρογραμμάτων και ορισμένες εντολές της PASC320, είτε περιέχονται σε λογικές εκφράσεις. Τέλος, ορίζονται και απλές εκφράσεις σύνθετου τύπου πίνακα ή εγγραφής, που εμφανίζονται μόνο σε κλήσεις υποπρογραμμάτων και ορισμένες εντολές της PASC320, όπως θα δούμε παρακάτω.

Μια έκφραση της PASC320 περιλαμβάνει:

- Τιμές αριστερής προσπέλασης (L-values), δηλαδή διευθύνσεις του χώρου δεδομένων του προγράμματος που αντιστοιχούν σε μεταβλητές, στοιχεία πινάκων, στοιχεία λίστας ή πεδία εγγραφών.
- Σταθερές, είτε σαν τιμές που υπάρχουν στον αρχικό κώδικα της μονάδας, είτε σαν τιμές ονομάτων σταθερών, ορατών στη μονάδα.
- Τελεστές, που επιτρέπουν πράξεις μεταξύ υποεκφράσεων.
- Κλήσεις συναρτήσεων, που έχουν σαν αποτέλεσμα την αντικατάσταση της συνάρτησης στην έκφραση με την τιμή που αυτή επιστρέφει.

Τα τρία τελευταία αντικείμενα ορίζουν τιμές δεξιάς προσπέλασης (R-values).

Παραδείγματα εκφράσεων της PASC320 είναι τα εξής:

```
i
a+1
a[3,i+1,j-1]*w.k*2 + (tu(b,z[2]) div (i-1))*i
(a+b[j-i].f[-2] > 0) and not (x in [0,1]) or (m = 'm')
z.i+(w.i/2-2.0)/1.2
[3, x+3, f(4*y)]
```

### Τιμές αριστερής προσπέλασης

Κάθε διεύθυνση στο χώρο δεδομένων του προγράμματος ονομάζεται τιμή αριστερής προσπέλασης, επειδή μπορεί να βρεθεί στο αριστερό μέρος μιας ανάθεσης. Τέτοιες τιμές στην PASC320 αποτελούν οι μεταβλητές, τα στοιχεία πινάκων, τα στοιχεία λίστας και τα πεδία εγγραφών.

Όταν μια τιμή αριστερής προσπέλασης βρεθεί σε μια έκφραση, αποτιμάται, και η τιμή που αποδίδεται γι' αυτήν είναι η τιμή που περιέχεται στη διεύθυνση που αυτή παριστάνει, δηλαδή η τιμή της μεταβλητής, του στοιχείου πίνακα ή λίστας, ή του πεδίου εγγραφής. Απαραίτητη προϋπόθεση για αποτίμηση είναι η προηγούμενη δήλωση της αντίστοιχης μεταβλητής, του αντίστοιχου πίνακα, της αντίστοιχης λίστας ή της αντίστοιχης εγγραφής, στην ίδια εμβέλεια ή σε κάποια εξωτερική της. Σε περίπτωση απουσίας τέτοιας δήλωσης υπάρχει σημασιολογικό σφάλμα.

Αν η τιμή αριστερής προσπέλασης αντιστοιχεί σε σύνθετο τύπο, η αποτίμηση αυτής δίνει ολόκληρο το περιεχόμενο αυτής, δηλαδή του αντίστοιχου πίνακα, συνόλου, λίστας ή εγγραφής. Τιμές αριστερής προσπέλασης σύνθετου τύπου πίνακα ή εγγραφής μπορούν να αποτελούν μόνο ανεξάρτητες απλές εκφράσεις, επειδή κανένας τελεστής δε μπορεί να εφαρμοστεί σε τέτοιον τύπο.

Η αποτίμηση μεταβλητής που έχει δηλωθεί κανονικά σε μια μονάδα γίνεται με ανάγνωση της τιμής της μεταβλητής από το χώρο δεδομένων της μονάδας.

Η αποτίμηση στοιχείου πίνακα που επίσης έχει δηλωθεί κανονικά μπορεί να γίνει, εφ' όσον οι τιμές των εκφράσεων που αποδίδονται στους δείκτες του στοιχείου αυτού είναι μέσα στα δηλωμένα όρια για τις αντίστοιχες διαστάσεις του πίνακα. Μ' άλλα λόγια, στην αποτίμηση στοιχείου πίνακα προηγείται η αποτίμηση των δεικτών του. Εάν οι τιμές των δεικτών είναι αποδεκτές, υπολογίζεται η διεύθυνση του ζητούμενου στοιχείου, λαμβάνοντας υπ' όψη την αρχική διεύθυνση του πίνακα στο χώρο δεδομένων της μονάδας, τον τρόπο αποθήκευσης των στοιχείων του πίνακα, τις διαστάσεις, τα όρια υποπεριοχής σε κάθε διάσταση, καθώς και το μέγεθος του στοιχείου του. Στη συνέχεια μπορεί να αποτιμηθεί το στοιχείο αυτό, με ανάγνωσή του από το χώρο δεδομένων.

Η PASC320 έχει ισχυρό σύστημα τύπων, κι επομένως ο πιο πάνω έλεγχος τιμών των δεικτών πρέπει να γίνεται από τον κώδικα του προγράμματος. Ο μεταγλωττιστής, δηλαδή, πρέπει για κάθε προσπέλαση πίνακα να παράγει κώδικα που να κάνει αυτόν τον έλεγχο.

Για μια εγγραφή που έχει δηλωθεί κανονικά, η αποτίμηση ενός πεδίου της μπορεί να γίνει, εφ' όσον αυτό είναι δηλωμένο στην εγγραφή. Η διεύθυνσή του υπολογίζεται λαμβάνοντας υπ' όψη την αρχική διεύθυνση της εγγραφής στο χώρο δεδομένων της μονάδας, καθώς και το μέγεθος των πεδίων που προηγούνται. Στη συνέχεια το στοιχείο αυτό αποτιμάται με ανάγνωσή του από το χώρο δεδομένων.

Για μια λίστα, τέλος, που έχει δηλωθεί κανονικά, η αποτίμηση ενός στοιχείου της γίνεται μόνο μέσω των ειδικών συναρτήσεων στοιχείων λίστας, οι οποίες με παράμετρο μια λίστα αναφέρονται στο περιεχόμενο και στη διεύθυνση επόμενου στοιχείου ως εξής:

1. Η συνάρτηση CAR() αναφέρεται στο περιεχόμενο του πρώτου στοιχείου της λίστας.
2. Η συνάρτηση CDR() αναφέρεται στη διεύθυνση του επόμενου στοιχείου της λίστας.
3. Η συνάρτηση CADR() αναφέρεται στο περιεχόμενο του δεύτερου στοιχείου της λίστας.
4. Η συνάρτηση CDDR() αναφέρεται στη διεύθυνση του μεθεπόμενου στοιχείου της λίστας.
5. Όμοια ορίζονται και άλλες συναρτήσεις που ακολουθούν την περιγραφή της λεκτικής μονάδας LISTFUNC που δόθηκε νωρίτερα.

Οι συναρτήσεις αυτές είναι τιμές αριστερής προσπέλασης που μέσα σε εκφράσεις αποτιμώνται και επιστρέφουν τις παραπάνω τιμές.

Η αποτίμηση μεταβλητής που δεν είναι δηλωμένη στην παρούσα μονάδα, αλλά σε κάποια εξωτερική της, πρέπει να γίνεται στο χώρο δεδομένων εκείνης της μονάδας. Επειδή η PASC320 υποστηρίζει στατικό δέσιμο, η διεύθυνση της μεταβλητής στο χώρο δεδομένων της εξωτερικής μονάδας βρίσκεται με τη βοήθεια συνδέσμων προσπέλασης ή πινάκων δεικτών. Ο μεταγλωττιστής πρέπει να παράγει κώδικα, ο οποίος να υπολογίζει τη διεύθυνση αυτή, και στη συνέχεια να προχωράει στην αποτίμηση της μεταβλητής.

Οι τυπικές παράμετροι ενός υποπρογράμματος είναι τιμές αριστερής προσπέλασης στο δυναμικό χώρο δεδομένων του υποπρογράμματος στη στοίβα, αν μεταδίδονται κατ' αξία, και στον

υπόλοιπο χώρο δεδομένων, αν μεταδίδονται κατ' αναφορά. Στη δεύτερη περίπτωση, και σε κάθε κλήση του υποπρογράμματος, οι αντίστοιχες πραγματικές παράμετροι πρέπει να είναι τιμές αριστερής προσπέλασης, που δεν αποτιμώνται, αλλά περνάνε στο υποπρόγραμμα με τη διεύθυνσή τους. Η αποτίμηση μιας τυπικής παραμέτρου που μεταδίδεται κατ' αναφορά γίνεται σε εκφράσεις που αυτή εμφανίζεται, με ανάγνωση από τη διεύθυνσή της. Ειδικά αν η τυπική παράμετρος είναι σύνθετου τύπου πίνακα, λίστας ή εγγραφής, οπότε και μεταδίδεται κατ' αναφορά, η προσπέλαση κάποιου στοιχείου της γίνεται όπως αναφέρθηκε παραπάνω.

### Σταθερές

Οι σταθερές της PASC320 είναι αυτές που αναγνωρίζονται άμεσα από το λεκτικό αναλυτή και περιγράφηκαν στην αντίστοιχη ενότητα, καθώς και σταθερές που έχουν δηλωθεί με αναγνωριστικά σε δήλωση σταθερών, και είναι ορατές στην παρούσα μονάδα.

Ειδικά όσο αφορά τις πρώτες:

Οι τιμές των αριθμητικών σταθερών προκύπτουν άμεσα με μετατροπή των αντίστοιχων λέξεων σε αριθμητικές τιμές. Οι τιμές των λογικών σταθερών, που συμβολίζονται με τις λέξεις "true" και "false" για «Αληθής» και «Ψευδής» αντίστοιχα, αποδίδονται με την κωδικοποίηση που αναφέρθηκε παραπάνω. Οι τιμές των σταθερών χαρακτήρων αποδίδονται με την κωδικοποίηση ASCII.

Οι αριθμητικές σταθερές δεν έχουν πρόσημο, και προσημασμένοι αριθμοί προκύπτουν από εκφράσεις με τη βοήθεια του τελεστή προσήμου ADDOP.

### Τελεστές

Τελεστές είναι ειδικά σύμβολα, που εφαρμοζόμενα σε έναν αριθμό εκφράσεων – που ονομάζονται *τελούμενα εισόδου* ή *τελεστέοι*, παράγουν μια νέα έκφραση.

Οι τελεστές της PASC320 δίνονται στο σχετικό πίνακα και διακρίνονται σε τελεστές με ένα τελούμενο και τελεστές με δύο τελούμενα εισόδου. Στην πρώτη κατηγορία ανήκουν οι τελεστές του προσήμου και της λογικής άρνησης. Οι τελεστές αυτής της κατηγορίας αναγράφονται πριν το τελούμενό τους. Στη δεύτερη κατηγορία ανήκουν οι υπόλοιποι τελεστές αριθμητικών και λογικών πράξεων. Οι τελεστές αυτής της κατηγορίας αναγράφονται ανάμεσα στα τελούμενά τους.

Τελεστές με ένα ή δύο τελούμενα εισόδου μπορούν να θεωρηθούν και τα διαχωριστικά σύμβολα '(', ')', '[', ']' και '.', σε περιπτώσεις που θα αναλυθούν στη συνέχεια.

Πιο συγκεκριμένα:

- Στην απλούστερη περίπτωση, ο τελεστής αναφοράς σε πεδίο εγγραφής '.' έχει σύνταξη:  
 <αναγνωριστικό> '.' <αναγνωριστικό>  
 Ο τελεστής αυτός έχει δύο τελούμενα εισόδου, από τα οποία το πρώτο είναι το όνομα της εγγραφής και το δεύτερο το όνομα ενός πεδίου αυτής. Στην περίπτωση αυτή, η αναφορά σε πεδίο εγγραφής αντιμετωπίζεται όπως έχει περιγραφεί παραπάνω.  
 Στην πιο γενική περίπτωση, είναι δυνατό το πρώτο όνομα να είναι όνομα πεδίου άλλης εγγραφής – πιθανά δεικτοδοτημένο, όταν το πεδίο αυτό είναι τύπου εγγραφής, ή ακόμα, το δεύτερο να είναι όνομα εγγραφής, όταν αυτή είναι πεδίο της πρώτης. Τότε, για την αναφορά στο τελευταίο αναγραφόμενο πεδίο εγγραφής, και για τον υπολογισμό της διεύθυνσής του, πρέπει να προηγηθούν οι επιλύσεις αναφορών για όλους τους προηγούμενους τύπους, που μπορεί να είναι άλλες εγγραφές ή πίνακες. Ένα παράδειγμα αναφοράς σε πεδίο εγγραφής τέτοιας μορφής δίνεται πιο κάτω.
- Στην απλούστερη περίπτωση, ο τελεστής αναφοράς σε στοιχείο πίνακα '[' έχει σύνταξη:  
 <αναγνωριστικό> '[' <λίστα εκφράσεων> ']'  
 Ο τελεστής αυτός έχει δύο τελούμενα εισόδου, από τα οποία το πρώτο είναι το όνομα του πίνακα, και το δεύτερο μια λίστα εκφράσεων, οι τιμές των οποίων αποδίδονται στους δείκτες του στοιχείου πίνακα. Η αποτίμηση των εκφράσεων στη λίστα γίνεται από τα α-



Τελεστής	Περιγραφή	Αριθμός τελούμενων	Προσεταιριστικότητα
<code>\. ' , \ [ ] ' , \ ( ) '</code>	Αναφορά σε πεδίο εγγραφής, αναφορά σε στοιχείο πίνακα, κλήση συνάρτησης	2, 2, 2	αριστερή
NOTOP	Λογική άρνηση	1	-
MULDIVANDOP	Πολλαπλασιασμός, ακέραια / πραγματική διαίρεση, τομή συνόλων, λογικό γινόμενο	2	αριστερή
ADDOP, OROP	Πρόσθημη, πρόσθεση, αφαίρεση, ένωση και διαφορά συνόλων, ένωση λιστών, λογικό άθροισμα	1, 2	αριστερή
INOP, RELOP, \ = '	Σχεσιακοί τελεστές	2	καμία
<code>\ [ ] '</code>	Σύνθεση συνόλου ή λίστας	1	-

#### Τελεστές της PASC320 σε φθίνουσα σειρά προτεραιότητας

ριστερά προς τα δεξιά, και η αναφορά στο στοιχείο πίνακα γίνεται όπως περιγράφεται παραπάνω.

Στην πιο γενική περίπτωση, ο πίνακας μπορεί να είναι στοιχείο άλλου πίνακα ή πεδίο εγγραφής, αλλά και το στοιχείο μπορεί να είναι επίσης τύπου πίνακα. Τότε, για την αναφορά στο τελευταίο αναγραφόμενο στοιχείο πίνακα και τον υπολογισμό της διεύθυνσής του, πρέπει να προηγηθούν οι επιλύσεις αναφορών για τους προηγούμενους τύπους, που μπορεί να είναι άλλοι πίνακες ή εγγραφές. Ένα παράδειγμα αναφοράς σε στοιχείο πίνακα τέτοιας μορφής δίνεται πιο κάτω.

- Ο τελεστής κλήσης συνάρτησης `(' )` έχει σύνταξη:

<αναγνωριστικό> `(' <λίστα εκφράσεων> ')`

Ο τελεστής αυτός έχει δύο τελούμενα εισόδου, από τα οποία το πρώτο είναι το όνομα της συνάρτησης, και το δεύτερο μια λίστα εκφράσεων, οι τιμές των οποίων αποδίδονται στις πραγματικές παραμέτρους της κλήσης. Η αποτίμηση των εκφράσεων στη λίστα γίνεται από τα αριστερά προς τα δεξιά. Η κλήση συνάρτησης θα επεξηγηθεί παρακάτω.

- Ο τελεστής σύνθεσης συνόλου ή λίστας `[ ]` έχει σύνταξη:

`[ ' <λίστα εκφράσεων> ]`

Ο τελεστής αυτός έχει σα μοναδικό τελούμενο εισόδου μια λίστα εκφράσεων, οι τιμές των οποίων αποδίδονται στα στοιχεία ενός συνόλου ή στα διαδοχικά στοιχεία μιας λίστας. Η αποτίμηση των εκφράσεων στη λίστα γίνεται από τα αριστερά προς τα δεξιά. Η λίστα εκφράσεων μπορεί να είναι κενή, οπότε κατασκευάζεται το κενό σύνολο ή μια κενή λίστα, όπου κενή λίστα ισοδυναμεί με μηδενική διεύθυνση πρώτου στοιχείου.

Αναλυτικότερα, αν ο τελεστής αναφέρεται σε σύνθεση συνόλου, οι επιμέρους εκφράσεις της λίστας πρέπει να είναι του ίδιου τύπου, βασικού εκτός από real. Επίσης, μπορούν να είναι απλές εκφράσεις τύπου υποπεριοχής, οπότε θεωρούνται ως του τύπου από τον οποίο παράγονται. Το αποτέλεσμα είναι τύπου συνόλου, που ο μεταγλωττιστής χρησιμοποιεί είτε σε κάποια πράξη συνόλων, είτε σε κάποια ανάθεση σε τιμή αριστερής προσπέλασης τύπου συνόλου. Στη δεύτερη περίπτωση, η έκφραση πρέπει να μετατρέπεται σε ψηφιοδιάνυσμα.

Αν ο τελεστής αναφέρεται σε σύνθεση λίστας, οι επιμέρους εκφράσεις πρέπει να είναι του ίδιου τύπου, βασικού ή άλλου βαθμωτού τύπου. Το αποτέλεσμα της σύνθεσης είναι τύπου λίστας, που ο μεταγλωττιστής χρησιμοποιεί είτε σε κάποια πράξη, είτε σε κάποια ανάθεση σε τιμή αριστερής προσπέλασης τύπου λίστας. Σε κάθε περίπτωση, ο τύπος των στοιχείων της λίστας πρέπει να είναι συμβατός με τον άλλο τελεστέο αν πρόκειται για πράξη, ή με την τιμή αριστερής προσπέλασης αν πρόκειται για ανάθεση. Για τη σύνθεση της λίστας, ο μεταγλωττιστής παράγει κώδικα, ο οποίος να καλεί κάποια συνάρτηση δυναμικής εκχώρησης μνήμης για καθένα από τα στοιχεία της ακολουθίας, ώστε να κατα-

σκευαστούν τα ζεύγη (περιεχόμενο, διεύθυνση επόμενου στοιχείου) που αποτελούν τη λίστα. Το τελευταίο στοιχείο λαμβάνει το 0 ως διεύθυνση επόμενου στοιχείου. Η τιμή του αποτελέσματος της σύνθεσης θα είναι έτσι η διεύθυνση του πρώτου στοιχείου της λίστας.

Ανάλογα με τα συμφραζόμενα, ο σημασιολογικός αναλυτής κρίνει αν ο τελεστής συνθέτει σύνολο ή λίστα.

- Ο τελεστής λογικής άρνησης NOTOP. Ο μοναδικός τελεστής του τελεστή αυτού πρέπει να είναι τύπου boolean και το αποτέλεσμα είναι του ίδιου τύπου.
- Οι τελεστές προσήμου ADDOP. Ο μοναδικός τελεστής των τελεστών αυτών πρέπει να είναι αριθμητικού τύπου και το αποτέλεσμα της εφαρμογής τους είναι του ίδιου τύπου.
- Οι αριθμητικοί τελεστές MULDIVANDOP '\*', '/', 'div' (ακέραιο πηλίκο) και 'mod' (ακέραιο υπόλοιπο), και οι τελεστές ADDOP. Όταν οι τελεστές αυτοί εφαρμόζονται σε αριθμητικούς τύπους, η συμβατότητα τύπων των τελεστών αυτών και ο τύπος του αποτελέσματος της εφαρμογής τους καθορίζονται στον επόμενο πίνακα:

A \ B		
	integer	real
integer	integer, real με '/'	real, όχι 'div', 'mod'
real	real, όχι 'div', 'mod'	real, όχι 'div', 'mod'

Τύπος αποτελέσματος αριθμητικής έκφρασης A op B

Όπως δείχνει ο πίνακας, η μόνη περίπτωση μη συμβατότητας τύπων αφορά τους τελεστές MULDIVANDOP 'div' και 'mod', οι οποίοι μπορούν να έχουν τελούμενα εισόδου μόνο τύπου integer. Ακόμα, ο τελεστής MULDIVANDOP '/' δίνει πάντα αποτέλεσμα τύπου real. Σε οποιαδήποτε περίπτωση τα τελούμενα εισόδου είναι διαφορετικού τύπου, είναι απαραίτητο ο κώδικας να περιέχει μετατροπή του ενός τύπου στον τύπο του αποτελέσματος, η δε πράξη πρέπει να γίνεται για τον τύπο του αποτελέσματος. Ειδικά για τον τελεστή '/', ακόμα κι όταν και τα δύο τελούμενά του είναι τύπου integer, πρέπει αυτά να μετατρέπονται σε τύπο real.

Οι τελεστές MULDIVANDOP '\*' και ADDOP '+' και '-' εφαρμόζονται και σε τύπους συνόλου, οπότε υλοποιούν τις πράξεις της τομής, της ένωσης και της διαφοράς συνόλων, αντίστοιχα. Πράξεις συνόλων εφαρμόζονται σε σύνολα με στοιχεία του ίδιου τύπου, όπου τα στοιχεία τύπου υποπεριοχής θεωρούνται ως του τύπου από τον οποίο παράγονται. Το αποτέλεσμα της εφαρμογής των τελεστών αυτών είναι του ίδιου τύπου συνόλου. Αν ένα τουλάχιστον από τα δύο τελούμενα εισόδου είναι τιμή αριστερής προσπέλασης, τότε το αποτέλεσμα θα πρέπει να μπορεί να κωδικοποιηθεί σε ψηφιοδιάνυσμα. Ο μεταγλωττιστής θα πρέπει να φροντίσει, ώστε ο κώδικας που θα παράγει, να εκτελεί την πράξη, ανεξάρτητα από το αν τα τελούμενα εισόδου είναι σε μορφή ψηφιοδιανυσμάτων ή όχι.

Τέλος, ο τελεστής ADDOP '+' εφαρμόζεται και σε τύπο λίστας, οπότε υλοποιεί την ένωση δύο λιστών. Η ένωση είναι η μόνη πράξη που επιτρέπεται στον τύπο αυτό, και επιτρέπεται σε λίστες με στοιχεία του ίδιου τύπου. Το αποτέλεσμα της ένωσης είναι του ίδιου τύπου λίστας, όπου η διεύθυνση επόμενου στοιχείου του τελευταίου στοιχείου της πρώτης λίστας έχει λάβει τιμή τη διεύθυνση πρώτου στοιχείου της δεύτερης λίστας.

- Οι τελεστές λογικών πράξεων MULDIVANDOP 'and' και OROP. Τα τελούμενα εισόδου αυτών πρέπει να είναι τύπου boolean. Το αποτέλεσμα της εφαρμογής τους είναι επίσης τύπου boolean.
- Οι σχεσιακοί τελεστές σύγκρισης RELOP και ισότητας '='. Τα τελούμενα εισόδου των τελεστών αυτών μπορούν να είναι του ίδιου ή διαφορετικού αριθμητικού τύπου. Στην τελευταία περίπτωση, ο κώδικας πρέπει να μετατρέπει το τελούμενο τύπου integer σε real. Μετά από πιθανή μετατροπή, η σύγκριση πρέπει να γίνεται για τον τύπο των τελούμενων. Οι τελεστές αυτοί δέχονται και τελούμενα μη αριθμητικών διατεταγμένων τύπων, δηλαδή των βασικών τύπων char και boolean, όπως και υποπεριοχής. Επίσης, δέχονται και τε-

λούμενα τύπων συνόλου. Στην περίπτωση αυτή, οι τελεστές RELOP ' $\leq$ ', ' $<$ ', ' $\geq$ ' και ' $>$ ' υλοποιούν τις σχέσεις υποσύνολο, γνήσιο υποσύνολο, υπερσύνολο και γνήσιο υπερσύνολο, αντίστοιχα, και, όπως και προηγουμένως, ο μεταγλωττιστής θα πρέπει να φροντίσει, ώστε ο κώδικας που θα παράγει, να εκτελεί τη σύγκριση, ανεξάρτητα από το αν τα τελούμενα εισόδου είναι σε μορφή ψηφιοδιανυσμάτων ή όχι. Το αποτέλεσμα της εφαρμογής των σχεσιακών τελεστών είναι τύπου boolean.

- Ο σχεσιακός τελεστής ελέγχου μέλους συνόλου INOP. Ο τελεστής αυτός δέχεται δύο τελούμενα εισόδου, από τα οποία το δεξί είναι ενός τύπου συνόλου και το αριστερό είναι του τύπου των στοιχείων του συνόλου. Το αποτέλεσμα της εφαρμογής του τελεστή αυτού είναι τύπου boolean, με τιμή που δηλώνει αν το αριστερό τελούμενο ανήκει στο δεξί. Ο κώδικας που θα παραχθεί πρέπει να εκτελεί τον έλεγχο, ανεξάρτητα από το αν το δεξί τελούμενο εισόδου είναι σε μορφή ψηφιοδιανύσματος ή όχι.

Παρά την αναπαράσταση των τύπων boolean και char με ακέραιες τιμές, στη PASC320 δεν υπάρχει συμβατότητα μεταξύ αριθμητικών και μη αριθμητικών τιμών. Έτσι, η συμμετοχή αριθμητικών εκφράσεων σε λογικές που αναφέρθηκε νωρίτερα μπορεί να γίνει μόνο μέσω των πιο πάνω σχεσιακών τελεστών. Αυτοί είναι οι μόνοι τελεστές που κατασκευάζουν λογικές εκφράσεις από αριθμητικές, καθώς δέχονται αριθμητικά τελούμενα και παράγουν αποτέλεσμα τύπου boolean. Το αντίστροφο δεν είναι εφικτό, γι' αυτό και δε μπορεί μια λογική έκφραση να συμμετέχει σε μια αριθμητική. Παρόμοια και για τον τύπο char – όπως και υποπεριοχών αυτού, μια έκφραση αυτού του τύπου μπορεί να συμμετάσχει σε κάποια λογική έκφραση μόνο μέσω των σχεσιακών τελεστών που αναφέρθηκαν, ενώ δε μπορεί να συμβεί το αντίστροφο, αφού δεν υπάρχει τελεστής που να δίνει αποτέλεσμα τύπου char. Επίσης δεν υπάρχει καμία δυνατότητα συμμετοχής τύπου char ή υποπεριοχών αυτού σε αριθμητική έκφραση.

Εξαίρεση στα παραπάνω αποτελούν οι τελεστές '[' και '(' που από τη μία προσφέρουν τη δυνατότητα *έμμεσης* συμμετοχής μιας έκφρασης σε μια μεγαλύτερη, αλλά και από την άλλη μπορούν να δώσουν αποτέλεσμα τύπου που δε μπορούν να δώσουν άλλοι τελεστές.

Σε κάθε εφαρμογή τελεστή, και όταν συμμετέχουν περισσότερα από ένα τελούμενα εισόδου, η αποτίμηση αυτών γίνεται από αριστερά προς τα δεξιά. Σε κάθε περίπτωση απαιτείται αποτίμηση όλων των τελούμενων, μια που η PASC320 δεν υποστηρίζει βραχυκύκλωση.

Μέσα σε μια έκφραση μπορούν να υπάρχουν πολλοί τελεστές. Η σειρά εφαρμογής αυτών για την αποτίμηση της έκφρασης καθορίζεται από παρενθέσεις ή από κανόνες προτεραιότητας και προσεταιριστικότητας. Η σειρά προτεραιότητας των τελεστών καθορίζεται από τη σειρά που αυτοί αναγράφονται στον πρώτο από τους δύο παραπάνω πίνακες, από μεγαλύτερη προς μικρότερη προτεραιότητα. Η προσεταιριστικότητα των τελεστών, δίνεται στον ίδιο πίνακα. Αν δεν έχει νόημα, η προσεταιριστικότητα δεν είναι ορισμένη και αναγράφεται ως '-'. Αξίζει να σημειωθεί ότι οι τελεστές RELOP δε μπορούν να έχουν καμία προσεταιριστικότητα, εφ' όσον η εφαρμογή τους αλλάζει τον τύπο της έκφρασης. Έτσι, η έκφραση:

$$x+2 > y > 0$$

είναι λανθασμένη. Εφαρμογή οποιουδήποτε από τους δύο τελεστές RELOP '>' δίνει αποτέλεσμα τύπου boolean, που δεν επιτρέπει την εφαρμογή του άλλου.

Σε μια σύνθετη έκφραση δεν επιτρέπεται διαδοχική εφαρμογή τελεστών προσήμου ADDOP, όπως για παράδειγμα στην έκφραση:

$$x > -+3$$

που δεν είναι αποδεκτή.

Ας θεωρήσουμε για παράδειγμα την έκφραση:

$$(i*x-y-j*z < -a \text{ div } k \text{ div } 2) \text{ and not } (b.k[i+2,m][0].a <> 'a')$$

στην οποία κατ' αρχήν υποθέτουμε ότι οι τύποι των τιμών που συμμετέχουν είναι οι προβλεπόμενοι.

Στην έκφραση αυτή ο τελεστής MULDIVANDOP 'and' έχει τη μικρότερη προτεραιότητα και δε μπορεί να εφαρμοστεί πριν την αποτίμηση και των δύο τελεστών του. Αυτό συμβαίνει

λόγω της χρήσης παρενθέσεων, μια που χωρίς αυτές, ο τελεστής MULDIVANDOP θα εφαρμόζοταν πριν τους τελεστές RELOP και ADDOP. Επίσης ο σχεσιακός τελεστής RELOP '<' στο αριστερό τελούμενο του προηγούμενου – το οποίο και αποτιμάται πριν από το δεξί – εφαρμόζεται μόνο αφού έχουν αποτιμηθεί και τα δύο τελούμενά του. Στο αριστερό τελούμενο του τελευταίου, οι δύο διαδοχικές αφαιρέσεις που ορίζονται από τον τελεστή ADDOP '-' εφαρμόζονται από αριστερά προς τα δεξιά, λόγω αριστερής προσεταιριστικότητας του τελεστή αυτού.

Ο πρώτος τελεστής που θα εφαρμοστεί στην πιο πάνω έκφραση θα είναι ο πρώτος από αριστερά τελεστής MULDIVANDOP '\*' που έχει μεγαλύτερη προτεραιότητα από τον ADDOP. Στη συνέχεια θα εφαρμοστεί ο πρώτος από αριστερά τελεστής ADDOP '-', ενώ πριν εφαρμοστεί ο δεύτερος από τους δύο διαδοχικούς ADDOP, θα πρέπει να εφαρμοστεί ο δεύτερος τελεστής MULDIVANDOP '\*' που υπάρχει στο δεξί τελούμενο του προηγούμενου. Η εφαρμογή του πρώτου από αριστερά τελεστή MULDIVANDOP 'div' προηγείται του δεύτερου, ενώ ο τελεστής προσήμου ADDOP '-' εφαρμόζεται μετά τους δύο διαδοχικούς MULDIVANDOP 'div', επιτρέποντας στη συνέχεια την εφαρμογή του τελεστή RELOP '<'. Έτσι ολοκληρώνεται η αποτίμηση του αριστερού τελεστέου του τελεστή MULDIVANDOP 'and' και μπορούμε να συνεχίσουμε με την αποτίμηση του δεξιού.

Στο τελούμενο του τελεστή NOTOP 'not' – που λόγω των παρενθέσεων εφαρμόζεται μετά την εφαρμογή του RELOP '<' – αποτιμάται πρώτα το αριστερό τελούμενο του τελευταίου, το οποίο είναι πεδίο εγγραφής, η οποία είναι στοιχείο πίνακα, ο οποίος είναι επίσης στοιχείο πίνακα, ο οποίος είναι πεδίο εγγραφής.

Έτσι, προηγείται ο υπολογισμός της αρχικής διεύθυνσης του πίνακα "b.k" που είναι πεδίο της εγγραφής "b", επειδή η εφαρμογή των τελεστών '.' και '[' έχει ίδια προτεραιότητα και αριστερή προσεταιριστικότητα. Στη συνέχεια, και εφ' όσον η εφαρμογή του τελεστή '[' γίνεται με αριστερή προσεταιριστικότητα, ακολουθεί η αποτίμηση των εκφράσεων των δεικτών που αναγράφονται πρώτοι από αριστερά ("i+2" και "m"), και από αριστερά προς τα δεξιά. Αφού γίνει ο έλεγχος ορθότητας τιμών των δεικτών, υπολογίζεται η αρχική διεύθυνση του στοιχείου "b.k[i+2,m]". Ακολουθεί η αποτίμηση της έκφρασης του δείκτη στη δεύτερη εφαρμογή του '[' , και ο έλεγχος ορθότητας της τιμής του. Στη συνέχεια υπολογίζεται η διεύθυνση του στοιχείου "b.k[i+2,m][0]", που είναι τύπου εγγραφής. Τέλος, υπολογίζεται η διεύθυνση του πεδίου a της εγγραφής αυτής, το οποίο αποτιμάται για να μας δώσει τη ζητούμενη τιμή, που πρέπει να είναι τύπου char.

Η αποτίμηση του δεύτερου τελούμενου του RELOP '<' δίνει την τιμή της σταθεράς 'a' τύπου char.

Τελειώνοντας, εφαρμόζεται ο παραπάνω τελεστής RELOP '<', ακολουθούμενος από τον NOTOP. Η εφαρμογή του MULDIVANDOP 'and' μπορεί τώρα να πραγματοποιηθεί.

Όπως φάνηκε στο παραπάνω παράδειγμα, πρέπει να χρησιμοποιούμε παρενθέσεις σε κάθε περίπτωση που επιθυμούμε παρέμβαση στους πιο πάνω κανόνες, όπως προβλέπει η σύνταξη των εκφράσεων της PASC320. Πιο γενικά, παρενθέσεις μπορούν να χρησιμοποιηθούν και για βελτίωση της εμφάνισης μιας έκφρασης, χωρίς αναγκαστικά να επηρεάζουν τους κανόνες εφαρμογής των τελεστών που συμμετέχουν σε αυτήν.

### Κλήση συναρτήσεων

Αν 'func' είναι το όνομα μιας συνάρτησης με αποτέλεσμα τύπου 'type', τότε η έκφραση

func(e<sub>1</sub>, e<sub>2</sub>, ..., e<sub>n</sub>)

είναι μια τιμή δεξιάς προσπέλασης τύπου 'type'. Ειδικά αν το αποτέλεσμα της συνάρτησης είναι τύπου λίστας, τότε η πιο πάνω έκφραση είναι αριστερής προσπέλασης αυτού του τύπου. Η αποτίμηση της έκφρασης αυτής γίνεται με την εκτέλεση του κώδικα της μονάδας της συνάρτησης, με τις εξής προϋποθέσεις:

- Ο αριθμός n των εκφράσεων στις παρενθέσεις – οι πραγματικές παράμετροι – πρέπει να είναι ίσος με τον αριθμό των τυπικών παραμέτρων.

- Ο τύπος κάθε πραγματικής παραμέτρου με πέρασμα κατ' αξία πρέπει να είναι συμβατός με τον τύπο της αντίστοιχης τυπικής παραμέτρου, σύμφωνα με τους κανόνες συμβατότητας για ανάθεση που περιγράφονται πιο κάτω.
- Ο τύπος κάθε πραγματικής παραμέτρου με πέρασμα κατ' αναφορά πρέπει να ταυτίζεται με τον τύπο της αντίστοιχης τυπικής παραμέτρου.

Κατά την κλήση μιας συνάρτησης οι πραγματικές παράμετροι αποτιμώνται από αριστερά προς τα δεξιά και τοποθετούνται στη στοίβα, στο χώρο που δεσμεύεται σε χώρος δεδομένων της συνάρτησης, απ' όπου θα μπορεί να τους χρησιμοποιήσει η μονάδα αυτής. Με την έξοδο από τη συνάρτηση, θα πρέπει η τιμή του αποτελέσματος να βρίσκεται σε προκαθορισμένο σημείο αποθήκευσης, επίσης στη στοίβα. Με κάθε κλήση μιας συνάρτησης, η στοίβα μεταβάλλεται, κι έτσι ο χώρος δεδομένων της συνάρτησης, δηλαδή το *εγγράφημα δραστηριοποίησης* της συνάρτησης, είναι δυναμικός.

Εξαίρεση στα παραπάνω αποτελούν οι προκαθορισμένες συναρτήσεις στοιχείων λίστας, οι οποίες δε λειτουργούν ως συναρτήσεις, αλλά απλά απομονώνουν και επιστρέφουν ένα μέρος της λίστας που δίνεται ως παράμετρος. Το μέρος αυτό μάλιστα επιστρέφεται ως τιμή αριστερής προσπέλασης, όπως ακριβώς ένα στοιχείο πίνακα ή ένα πεδίο εγγραφής.

Εκτός από τις προκαθορισμένες συναρτήσεις στοιχείων λίστας, ορίζεται και η προκαθορισμένη συνάρτηση LENGTH(), η οποία δέχεται ως παράμετρο μια λίστα και επιστρέφει το πλήθος των στοιχείων της λίστας, καθώς και η προκαθορισμένη συνάρτηση NEW(), η οποία δέχεται ως παράμετρο μια τιμή και επιστρέφει μια λίστα ενός στοιχείου, με στοιχείο την τιμή-παράμετρο, και με διεύθυνση επόμενου στοιχείου ίση με 0.

### Εντολές

Η PASC320 υποστηρίζει απλές και δομημένες εντολές, καθώς και τη σύνθετη εντολή. Μια δομημένη εντολή της PASC320 περιέχει μία ή περισσότερες απλές, δομημένες ή σύνθετες εντολές. Υπενθυμίζεται ότι μια σύνθετη εντολή μπορεί να είναι κενή, αλλά περιέχει οπωσδήποτε τις λέξεις-κλειδιά "begin" και "end", ενώ δε μπορεί να περιέχει δηλώσεις.

Οι απλές εντολές της PASC320 είναι:

- Η εντολή ανάθεσης
- Η εντολή κλήσης διαδικασίας
- Οι εντολές εισόδου/εξόδου

Οι δομημένες εντολές της PASC320 είναι:

- Η εντολή διακλάδωσης
- Η εντολή επιλογής
- Η εντολή βρόχου
- Η εντολή επανάληψης
- Η εντολή with

Μεταξύ οποιωνδήποτε διαδοχικών εντολών της PASC320 πρέπει να υπάρχει ο χαρακτήρας ';',.

### Η εντολή ανάθεσης

Η εντολή αυτή αποδίδει την τιμή μιας έκφρασης σε μια τιμή αριστερής προσπέλασης, δηλαδή σε μια μεταβλητή, ένα στοιχείο πίνακα, ένα στοιχείο λίστας ή ένα πεδίο εγγραφής του χώρου δεδομένων του προγράμματος. Στη δεύτερη περίπτωση, της ανάθεσης πρέπει να προηγηθεί η αποτίμηση των εκφράσεων των δεικτών και ο υπολογισμός της τιμής αριστερής προσπέλασης του στοιχείου, ενώ στην τελευταία περίπτωση, της ανάθεσης πρέπει να προηγηθεί ο υπολογισμός της τιμής αριστερής προσπέλασης του πεδίου.

Η ανάθεση γίνεται με αποθήκευση της τιμής της έκφρασης στη διεύθυνση που παριστάνει η τιμή αριστερής προσπέλασης.

Παραδείγματα αναθέσεων είναι τα εξής:

```
a := w > 0.0
ch := 'c'
c[i-1, mm[j]].x := x*x+1
s := []
l := NEW(1)
CDR(l) := [2] + CDR([5,3,-4])
```

όπου mm είναι πίνακας ακεραίων, c είναι πίνακας εγγραφών, s είναι σύνολο ή λίστα, ενώ l είναι λίστα.

Για να μπορεί να γίνει ανάθεση, πρέπει η τιμή της έκφρασης να είναι συμβατού τύπου με τον τύπο της αριστερής προσπέλασης.

Δύο βασικοί τύποι είναι συμβατοί για ανάθεση, εάν:

(α) ταυτίζονται, ή αλλιώς

(β) είναι αριθμητικοί, οπότε συμβαίνει μετατροπή σε πραγματικό για ανάθεση από τύπο integer σε τύπο real, ή αποκοπή του κλασματικού μέρους για ανάθεση από τύπο real σε τύπο integer.

Ο τύπος υποπεριοχής θεωρείται ως του τύπου από τον οποίο παράγεται, όσο αφορά συμβατότητα για ανάθεση. Όμως, και αν ο τύπος αριστερής προσπέλασης είναι τύπος υποπεριοχής, ο κώδικας θα πρέπει πριν την ανάθεση να ελέγχει αν η τιμή της έκφρασης είναι μέσα στα όρια της υποπεριοχής.

Ανάθεση μεταξύ σύνθετων τύπων επιτρέπεται, όταν αυτοί έχουν την ίδια δομή και οι βασικοί τύποι των στοιχείων ή πεδίων τους είναι συμβατοί. Τότε, ο μεταγλωττιστής πρέπει να κατασκευάσει τον κώδικα που θα αντιγράψει πιθανά πολλαπλές τιμές για την εκτέλεση της ανάθεσης. Αν για παράδειγμα x και y είναι πίνακες εγγραφών της ίδιας δομής με πεδία συμβατών βασικών τύπων, η ανάθεση

```
x := y
```

γίνεται με αντιγραφή όλων των πεδίων κάθε στοιχείου του y στο αντίστοιχο στοιχείο του x. Ειδικότερα, αν ο τύπος αριστερής προσπέλασης είναι τύπος συνόλου, αλλά η έκφραση συνόλου του δεξιού μέλους δεν είναι κωδικοποιημένη σε ψηφιοδιάγραμμα, ο μεταγλωττιστής, πριν τη μετατρέψει σε ψηφιοδιάγραμμα, θα πρέπει να ελέγξει αν τα στοιχεία της βρίσκονται στο εύρος στοιχείων του τύπου αυτού.

Ακόμα, αν ο τύπος αριστερής προσπέλασης είναι τύπος λίστας, δε χρειάζεται ειδικός κώδικας αντιγραφής, αλλά απλά η διεύθυνση πρώτου στοιχείου της λίστας που παριστάνεται με το δεξί μέλος αποδίδεται στην έκφραση λίστας του αριστερού μέλους.

Μια ανάθεση λίστας ελλοχεύει τον κίνδυνο κάποια στοιχεία λίστας να μείνουν “ξεκρέμαστα”, δηλαδή να μην είναι πλέον προσπελάσιμα, κάτι που επιβαρύνει το σύστημα δυναμικής εκχώρησης μνήμης με κίνδυνο ασφυξίας. Για το σκοπό αυτό ο μεταγλωττιστής θα πρέπει να παράγει εδώ επιπλέον κώδικα ανίχνευσης και συλλογής ξεκρεμαστών στοιχείων.

Δύο ειδικές μορφές ανάθεσης της PASC320 είναι οι ακόλουθες:

(α) Η ανάθεση στη μεταβλητή που παριστάνεται από το όνομα μιας συνάρτησης. Η τιμή αριστερής προσπέλασης αυτής είναι διεύθυνση στη στοίβα. Η ανάθεση αυτή κατά τα άλλα ακολουθεί τους πιο πάνω κανόνες. Σε μια συνάρτηση πρέπει να υπάρχει τουλάχιστον μία ανάθεση στο όνομά της.

(β) Η ανάθεση ενός ορμαθού χαρακτήρων σε μια μεταβλητή πίνακα στοιχείων τύπου char. Σε μια τέτοια ανάθεση, αντιγράφονται στον πίνακα διαδοχικοί χαρακτήρες από τον ορμαθό, μέχρι να τελειώσουν οι χαρακτήρες του ορμαθού, ή μέχρι να γεμίσει ο πίνακας. Αν υπάρχει χώρος στον πίνακα, μετά τους χαρακτήρες του ορμαθού αποθηκεύεται ο χαρακτήρας με τιμή 0.

### Η εντολή κλήσης διαδικασίας

Η εντολή αυτή μεταφέρει τη ροή του κώδικα σε κάποια διαδικασία. Παραδείγματα κλήσης διαδικασίας είναι τα παρακάτω:

```
proc_A(x.a, n+2, matrix, s[i] and t)
proc_B
```

Στην εντολή κλήσης διαδικασίας οι περιορισμοί στις πραγματικές παραμέτρους είναι ταυτόσημοι με τους αντίστοιχους περιορισμούς στην κλήση συνάρτησης που περιγράφηκαν νωρίτερα. Ο μηχανισμός κλήσης είναι επίσης ο ίδιος με το μηχανισμό κλήσης μιας συνάρτησης.

#### Οι εντολές εισόδου/εξόδου

Αυτές είναι οι εντολές ανάγνωσης (read) και εγγραφής (write) δεδομένων. Συντάσσονται σαν κλήσεις διαδικασιών – όπως και είναι στην πραγματικότητα – με το όνομα της εντολής και μια λίστα στοιχείων σε παρενθέσεις, που στην περίπτωση ανάγνωσης είναι τιμές αριστερής προσπέλασης, ενώ στην περίπτωση εγγραφής είναι εκφράσεις ή ορμαθοί χαρακτήρων.

Παραδείγματα εντολών εισόδου/εξόδου είναι:

```
read (n,x)
write ("Temperature: ",f,"F, or ",5/9*(f-32),"C.")
```

Η είσοδος και η έξοδος γίνονται στα συνήθη αρχεία εισόδου/εξόδου χωρίς προδιαγραφές, δηλαδή προηγούμενο καθορισμό του τύπου και της μορφής των στοιχείων που διαβάζονται ή γράφονται. Με κάθε εντολή εξόδου γράφεται μια γραμμή κειμένου στην έξοδο του προγράμματος.

#### Η εντολή διακλάδωσης

Η εντολή διακλάδωσης if είναι μια δομημένη εντολή με παραμέτρους μια λογική έκφραση και μία ή δύο άλλες εντολές.

Η εκτέλεση της εντολής διακλάδωσης ξεκινά με την αποτίμηση της λογικής έκφρασης. Αν αυτή έχει τιμή «Αληθής», εκτελείται η εντολή που ακολουθεί τη λέξη-κλειδί “then”, ενώ αν υπάρχει η λέξη-κλειδί “else”, τότε η εντολή που ακολουθεί αυτή τη λέξη δε θα εκτελεστεί. Εάν η λογική έκφραση έχει τιμή «Ψευδής» και υπάρχει η αντίστοιχη λέξη-κλειδί “else”, τότε εκτελείται μόνο η εντολή που ακολουθεί αυτή τη λέξη, ενώ αν δε υπάρχει η λέξη-κλειδί “else”, δεν εκτελείται καμία εντολή.

Ένα παράδειγμα εντολής διακλάδωσης είναι το παρακάτω:

```
if ((a[i+1] > 13.5) and z[j]) then
  begin
    a[i-1] := f(z[j+1],i-1)*2;
    if x.play = 0 then b.over := a[i]
  end
else
  a[i-1] := f(z[n-i],i-1)*k.z[i]+1
```

#### Η εντολή επιλογής

Η εντολή επιλογής case είναι μια δομημένη εντολή με παραμέτρους μια έκφραση και μία λίστα από εντολές, προηγούμενες από διαφορετικές ενδεχόμενες τιμές της έκφρασης ή τη λέξη-κλειδί “otherwise”.

Η εκτέλεση της εντολής επιλογής ξεκινά με την αποτίμηση της έκφρασης. Στη συνέχεια εκτελείται εκείνη η εντολή από τη λίστα, της οποίας κάποια αναγραφόμενη ενδεχόμενη τιμή ισούται με την τιμή της έκφρασης. Αν δεν υπάρχει τέτοια εντολή, και υπάρχει εντολή που να ακολουθεί τη λέξη-κλειδί “otherwise”, τότε εκτελείται εκείνη η εντολή. Διαφορετικά δεν εκτελείται καμία εντολή.

Ένα παράδειγμα εντολής επιλογής δίνεται παρακάτω:

```
case x.info mod 5 of
  0,1: a[i-2][j].t := 0;
  4: begin
    a[i-2][j+1].t := 1;
    a[i-2][j-1].u := -1;
  end;
otherwise: try_other(x,a)
end
```

Σε μια εντολή επιλογής, η έκφραση επιλογής πρέπει να είναι βασικού τύπου εκτός από `real`, ή αντίστοιχου τύπου υποπεριοχής, ενώ οι αναγραφόμενες τιμές πρέπει να έχουν τον ίδιο τύπο με τον τύπο της έκφρασης, και, στην τελευταία περίπτωση, να βρίσκονται μέσα στα όρια της υποπεριοχής.

Η σύνταξη της εντολής επιτρέπει προσημασμένες σταθερές στις αναγραφόμενες τιμές. Ο σημασιολογικός αναλυτής πρέπει να ανιχνεύει σφάλμα, εάν ο τύπος των τιμών δεν δέχεται πρόσημο.

Μετά την εντολή που αντιστοιχεί στη λέξη-κλειδί “otherwise” δεν επιτρέπεται να υπάρχει άλλη εντολή στη λίστα.

### Η εντολή βρόχου

Η εντολή βρόχου `while` είναι μια δομημένη εντολή με παραμέτρους μια λογική έκφραση και μια εντολή.

Η εκτέλεση της εντολής `while` ξεκινά με αποτίμηση της έκφρασης. Αν η έκφραση έχει τιμή «Αληθής», εκτελείται η εντολή που ακολουθεί τη λέξη-κλειδί “do”. Στη συνέχεια η διαδικασία επαναλαμβάνεται με νέα αποτίμηση της έκφρασης, και η εντολή ξαναεκτελείται όσο η τιμή της έκφρασης που υπολογίζεται είναι «Αληθής».

Μόλις η αποτίμηση της έκφρασης δώσει τιμή «Ψευδής», η εκτέλεση του βρόχου τερματίζεται.

Ένα παράδειγμα εντολής βρόχου δίνεται παρακάτω:

```
while i > 0 do
  begin
    a[i*m] := y/2;
    k := f(n*m,a,x);
    while j < 10 do
      b[j,i+k] := g(j,m-k)*1.0;
    end
    i := i-1
  end
```

### Η εντολή επανάληψης

Η εντολή επανάληψης `for` είναι μια δομημένη εντολή της PASC320 με παραμέτρους ένα πεδίο επανάληψης και μια εντολή.

Το πεδίο επανάληψης περιέχει το όνομα μιας μεταβλητής, που ονομάζεται μεταβλητή ελέγχου του βρόχου επανάληψης, και δύο αριθμητικές εκφράσεις, που αποτιμώμενες δίνουν: η πρώτη την αρχική τιμή και η δεύτερη την τελική τιμή της μεταβλητής ελέγχου. Το βήμα αύξησης της μεταβλητής ελέγχου μεταξύ διαδοχικών επαναλήψεων του βρόχου είναι 1, αν μεταξύ των δύο εκφράσεων υπάρχει η λέξη-κλειδί “to”, και -1, αν μεταξύ των δύο εκφράσεων υπάρχει η λέξη-κλειδί “downto”.



Η εκτέλεση της εντολής `for` ξεκινά με αποτίμηση των εκφράσεων του πεδίου επανάληψης, και ανάθεση της αρχικής τιμής στη μεταβλητή ελέγχου. Στη συνέχεια η τιμή αυτή συγκρίνεται με την τελική, και εάν είναι μέσα στα όρια μεταβολής της, εκτελείται η εντολή που ακολουθεί τη λέξη-κλειδί “do”. Έπειτα αυξάνεται κατάλληλα η τιμή της μεταβλητής ελέγχου του βρόχου, και η διαδικασία επαναλαμβάνεται με τη σύγκριση της τιμής αυτής με την τελική. Όταν η τιμή της μεταβλητής ελέγχου βρεθεί εκτός ορίων μεταβολής της, η εκτέλεση του βρόχου τερματίζεται.

Ας σημειωθεί ότι οι εκφράσεις του πεδίου επανάληψης αποτιμώνται *μόνο* μία φορά, στην αρχή της εκτέλεσης της εντολής `for`.

Ένα παράδειγμα εντολής επανάληψης δίνεται παρακάτω:

```
for i := 0 to n-1 do
  begin
    a[i*m] := y/2;
    k := s*f(n*m, a, x.i);
    for j := m downto m-i do
      b[j, i+k] := (m-k)*1.0
    end
```

Η σημασιολογία της εντολής επανάληψης συμπληρώνεται από τους ακόλουθους κανόνες:

1. Η μεταβλητή ελέγχου του βρόχου πρέπει να είναι τύπου `integer`.
2. Ο τύπος των εκφράσεων του πεδίου επανάληψης πρέπει να είναι `integer`.
3. Η μεταβλητή ελέγχου δεν επιτρέπεται να λαμβάνει τιμή μέσα στην εσωτερική εντολή του βρόχου – άρα ούτε να μεταδίδεται κατ’ αναφορά σε κάποιο υποπρόγραμμα που καλείται μέσα από το σύνολο αυτό.

### Η εντολή with

Η εντολή `with` είναι μια δομημένη εντολή της PASC320 με παραμέτρους μια τιμή αριστερής προσπέλασης τύπου εγγραφής και μια άλλη εντολή.

Με την εντολή `with` η εσωτερική εντολή μπορεί να αναφέρεται στα πεδία της εγγραφής χωρίς να αναγράφει το όνομα της εγγραφής.

Η εκτέλεση της εντολής `with` συνίσταται στον υπολογισμό της τιμής αριστερής προσπέλασης της εγγραφής και στην εκτέλεση της εντολής που ακολουθεί τη λέξη-κλειδί “do”. Οι αναφορές σε ονόματα πεδίων της εγγραφής επιλύονται με βάση την τιμή που υπολογίστηκε.

Για παράδειγμα, στην εντολή:

```
with a[i*2+1].field[j][z-1].x do
  begin
    road := 0;
    for n:=0 to i-1 do corner[n] := x mod start;
    if square then address := corner[0] else address := start
  end;
```

όπου τα ονόματα `road`, `corner`, `square` και `address` είναι πεδία της εγγραφής

```
a[i*2+1].field[j][z-1].x
```

η αναφορά σε αυτά γίνεται άμεσα, χωρίς δηλαδή την εγγραφή και τον τελεστή ‘.’.

Για την εκτέλεση μιας εντολής `with`, πρέπει η έκφραση που ακολουθεί τη λέξη-κλειδί “with” να ελέγχεται αν έχει όντως τιμή αριστερής προσπέλασης τύπου εγγραφής.

Η εντολή `with` ορίζει μια ειδικής μορφής εμβέλεια, όπου δεν υπάρχουν δηλώσεις, αλλά κάθε όνομα πεδίου της εγγραφής επισκιάζει κάθε άλλο αναγνωριστικό με το ίδιο όνομα και ορατότητα που καλύπτει την εντολή `with`.