Τμήμα Πληροφορικής Α.Π.Θ.

Προαιρετική Εργασία στο μάθημα
Σχεδίαση Γλωσσών Προγραμματισμού
Αφορά την Εξεταστική Ιουνίου – Ιουλίου 2021

Διδάσκων: Γεώργιος Χρ. Μακρής Ακ. Έτος: 2020-2021

Bonus βαθμού ή απαλλαγή από εξέταση

Η προαιρετική εργασία εκπονείται ***αποκλειστικά σε ατομική βάση*** και το όφελος που θα έχουν οι φοιτητές ανάλογα με την περίπτωση είναι:

- ***απαλλαγή από την εξέταση του Ιουνίου-Ιουλίου 2021***

οι φοιτητές αυτοί θα προσέλθουν στην εξέταση μόνο για να δηλώσουν ότι παρέδωσαν ή πρόκειται να παραδώσουν εργασία· οι ολοκληρωμένες εργασίες που εκτελούνται σωστά ***κατά την επίδειξη του αποτελέσματος στο διδάσκοντα*** θα βαθμολογούνται έως και ΑΡΙΣΤΑ (10).

- ***δυνατότητα να παραδοθεί μέρος μόνο της εργασίας***

οι φοιτητές που θα επιδείξουν ολοκληρωμένη **λεξική**, **συντακτική** και **σημασιολογική** ανάλυση με **πίνακα συμβόλων** και των οποίων ο μεταγλωττιστής θα εκτυπώνει το συντακτικό δέντρο του πηγαίου προγράμματος (αντί να παράγει κώδικα στη γλώσσα-στόχο) ***κατά την επίδειξη του αποτελέσματος στο διδάσκοντα*** θα δικαιούνται 30% bonus στον τελικό βαθμό της εξέτασης.

Αξιολόγηση εργασίας

Για να βαθμολογηθεί η εργασία, θα πρέπει ***να παρουσιαστεί στο διδάσκοντα***. Κατά την παρουσίασή σας θα περιγράψετε τη σχεδίαση του μεταγλωττιστή, θα επιδείξετε σε δικό σας υπολογιστή τη λειτουργία του και θα απαντήσετε σε σχετικές διευκρινιστικές ερωτήσεις. Το παραδοτέο για να είναι αποδεκτό πρέπει να περιλαμβάνει:

- σε ηλεκτρονική μορφή
  i. τον κώδικα της γλώσσας σε πηγαία (source) και εκτελέσιμη (executable) μορφή
  ii. αρχεία κειμένου ASCII με παραδείγματα προγραμμάτων έτοιμα για μεταγλώττιση και εκτέλεση
- αναφορά με
  i. συνοπτική περιγραφή της γλώσσας και της υλοποίησης της
  ii. παραδείγματα και αποτελέσματα από τη μεταγλώττιση τους και την εκτέλεση του κώδικα μηχανής

Αν ο διδάσκων κρίνει ότι το πρόγραμμα, που παραδίνετε δεν είναι αποτέλεσμα προσωπικής εργασίας, τότε δε θα λαμβάνεται υπόψη στον τελικό βαθμό.

Υλοποίηση εργασίας

Ο μεταγλωττιστής μπορεί να υλοποιηθεί στη γλώσσα προγραμματισμού της προτίμησής σας αρκεί να έχετε στη διάθεσή σας τα απαραίτητα εργαλεία (π.χ. flex και yacc ή κλώνους για γλώσσες προγραμματισμού εκτός της C). Τα περισσότερα εργαλεία αυτού του τύπου λειτουργούνε όπως περιγράφεται στις σημειώσεις – διαφάνειες και όπως φαίνεται από την υλοποίηση του πρότυπου εκπαιδευτικού μεταγλωττιστή, που ο κώδικάς του (σε γλώσσα C) διατίθεται στα εργαστηριακά μαθήματα, που έχουν αναρτηθεί στην ιστοσελίδα του μαθήματος στο http://elearning.auth.gr.

Τις τελευταίες εκδόσεις των εργαλείων flex και yacc σε εκτελέσιμη μορφή για windows μπορείτε να τις βρείτε στη διεύθυνση http://gnuwin32.sourceforge.net/packages.html. Τεκμηρίωση για τη χρήση τους και τη χρήση των κλώνων τους μπορείτε να βρείτε στη διεύθυνση http://dinosaur.compilertools.net/.

Ένας πλήρης κατάλογος άλλων σχετικών εργαλείων υπάρχει στη διεύθυνση http://catalog.compilertools.net/. Ο μεταγλωττιστής που θα κατασκευάσετε θα κάνει μετάφραση στη **γλώσσα της μηχανής MIPS**.

Η Γλώσσα Προγραμματισμού SimpleFortran

Γενική Περιγραφή

Η γλώσσα **SimpleFortran** μοιάζει με τη γλώσσα υψηλού επιπέδου FORTRAN, και ορίζεται στη συνέχεια. Η **SimpleFortran** είναι δομημένη με σύνθετες εντολές, σε αντίθεση με την κλασική FORTRAN, η οποία δε διαθέτει τέτοιες εντολές. Εκτός από τους συνήθεις τύπους της FORTRAN, η **SimpleFortran** υποστηρίζει και έναν τύπο λίστας που θυμίζει την αντίστοιχη δομή της LISP, καθώς και έναν τύπο ορμαθού χαρακτήρων. Η **SimpleFortran** δεν έχει αυστηρότητα στη μορφή των προγραμμάτων όπως η κλασική FORTRAN, και το κενό παίζει τον ίδιο ρόλο στη μορφή όπως αυτό παίζει στις πιο σύγχρονες γλώσσες. Ακόμα, η **SimpleFortran** επιτρέπει τον ορισμό υποπρογραμμάτων σε ένα μόνο το εξωτερικό – επίπεδο, όπως και η FORTRAN, επιτρέπει κοινές περιοχές, αλλά όχι δηλώσεις ισοδυναμίας μεταβλητών. Τέλος, η **SimpleFortran** χρησιμοποιεί τη στοίβα για κλήση υποπρογραμμάτων, επιτρέποντας έτσι αναδρομή.

Ειδική Περιγραφή

Α. Λεκτικές Μονάδες

Οι λεκτικές μονάδες που αποτελούν και τα τερματικά σύμβολα της γραμματικής της **SimpleFortran** περιγράφονται στη συνέχεια. Σε παρένθεση – όπου χρειάζεται – δίνονται τα αντίστοιχα συμβολικά ονόματα που εμφανίζονται στη γραμματική της **SimpleFortran**.
Μαζί με τις λεκτικές μονάδες δίνεται και η περιγραφή των σχολίων της **SimpleFortran**, τα οποία όμως δεν εμφανίζονται στη γραμματική της γλώσσας.
Σημειώνεται ότι στη γλώσσα **SimpleFortran** δεν υπάρχει διάκριση μεταξύ κεφαλαίων και πεζών αλφαβητικών χαρακτήρων, εκτός αν αυτοί αποτελούν μέρος της λέξης μιας λεκτικής μονάδας `CCONST` ή `SCONST`.

Λέξεις-κλειδιά

Οι παρακάτω λέξεις που αποτελούν ανεξάρτητες λεκτικές μονάδες της **SimpleFortran**:
`FUNCTION SUBROUTINE END COMMON INTEGER REAL COMPLEX LOGICAL CHARACTER STRING LIST DATA CONTINUE GOTO CALL READ WRITE LENGTH NEW IF THEN ELSE ENDIF DO ENDDO STOP RETURN`

Αναγνωριστικά (`ID`)

Συμβολοσειρές που αρχίζουν με αλφαβητικό χαρακτήρα, ακολουθούμενο από μηδέν ή περισσότερους αλφαριθμητικούς χαρακτήρες, και δεν είναι λέξη-κλειδί. Μετά τον υποχρεωτικό αλφαβητικό χαρακτήρα, το αναγνωριστικό μπορεί να περιέχει και χαρακτήρες "_", οι οποίοι όμως θα πρέπει να μην είναι διαδοχικοί, ενώ αν περιέχει έστω κι έναν χαρακτήρα "_", θα πρέπει και να τελειώνει σε "_".
Αποδεκτά παραδείγματα:

`a100version2`	`a100_version2_`

Μη αποδεκτά παραδείγματα
(Σε όλα τα μη αποδεκτά παραδείγματα που δίνονται, η συμβολοσειρά δεν αναγνωρίζεται συνολικά, είναι όμως δυνατό να αναγνωρίζονται μέρη αυτής ως ανεξάρτητες λεκτικές μονάδες):

`100version2`	`a100_version2`	`_a100_version2_`	`a100__version2_`	`a100--version2`

Απλές σταθερές
Μη προσημασμένες ακέραιες (`ICONST`):
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. Ακόμα, η συμβολοσειρά "0O" ακολουθούμενη από έναν ή περισσότερους αριθμητικούς χαρακτήρες εκτός των "8" και "9" που ο πρώτος δεν είναι ο "0". Στην περίπτωση αυτή, η τιμή που παριστάνεται είναι ο αντίστοιχος αριθμός – μετά το πρόθεμα "0O" – σε οκταδική 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 – μετά το πρόθεμα "0B" – σε δυαδική βάση.

Αποδεκτά παραδείγματα:
```
0
180
0X9F0
0o67
0b1001
```
Μη αποδεκτά παραδείγματα:
```
0180
XB7
0X0
00B10
0XG8A
0O59
0B301
```

Μη προσημασμένες πραγματικές (`RCONST`):
Μηδέν ή περισσότεροι αριθμητικοί χαρακτήρες που ακολουθούνται από το χαρακτήρα "." και τουλάχιστον έναν αριθμητικό χαρακτήρα, ή ένας τουλάχιστον αριθμητικός χαρακτήρας που ακολουθείται προαιρετικά από το χαρακτήρα "." και μηδέν ή περισσότερους αριθμητικούς χαρακτήρες. Ακολουθεί προαιρετικά πεδίο εκθέτη, που ξεκινάει με τον χαρακτήρα "E", ακολουθούμενο από προαιρετικό πρόσημο και τουλάχιστον έναν αριθμητικό χαρακτήρα. Αν δεν υπάρχει πεδίο εκθέτη, ο αριθμός μπορεί να είναι σε δεκαεξαδική βάση, με πρόθεμα "0X", σε οκταδική βάση, με πρόθεμα "0O", ή σε δυαδική βάση, με πρόθεμα "0B". Το ακέραιο μέρος της 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". Τέλος, εξαιρούνται σταθερές που έχουν ήδη οριστεί ως ακέραιες.
Αποδεκτά παραδείγματα:
```
180e-2
.5
180.100
7.
0xa.
0B1.0010
0O3.72
0X0.00B9CF
```
Μη αποδεκτά παραδείγματα:
```
180E-2.2
.e-2
.5.
.5G-2
7.0
05.2E-05
0xBE-2
0B01.1
1001
```

Λογικές (`LCONST`):
Οι συμβολοσειρές "`.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'
'\\'
```

Ορμαθοί χαρακτήρων (SCONST):
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"
```

Τελεστές

Λογικό Η (OROP): **.OR.**
Λογικό ΚΑΙ (ANDOP): **.AND.**
Λογικό ΟΧΙ (NOTOP): **.NOT.**
Τελεστές σύγκρισης (RELOP): **.GT. .GE. .LT. .LE. .EQ. .NE.**
Προσθετικοί τελεστές (ADDOP): **+ -**
Πολλαπλασιασμός (MULOP): *****
Διαίρεση (DIVOP): **/**
Δύναμη (POWEROP): ******
Σημειώστε ότι στις τέσσερις πρώτες κατηγορίες το σύμβολο "." στην αρχή και στο τέλος της λέξης αποτελεί μέρος αυτής.

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
```

Άλλες λεκτικές μονάδες

Άλλοι χαρακτήρες που αποτελούν ανεξάρτητες λεκτικές μονάδες είναι:
"(" (LPAREN), ")" (RPAREN), "," (COMMA), "=" (ASSIGN), ":" (COLON), "[" (LBRACK), "]" (RBRACK), (EOF)
Σε ορισμένες περιπτώσεις κάποιες από τις παραπάνω λεκτικές μονάδες ενεργούν ως τελεστές, όπως αυτό θα φανεί στη γραμματική και σημασιολογία της **SimpleFortran**.
Η λεκτική μονάδα **EOF** – που ως χαρακτήρας ορίζεται με ειδικό τρόπο από το κάθε σύστημα δεν εμφανίζεται στη γραμματική της **SimpleFortran**, αλλά πρέπει να παράγεται από το λεκτικό αναλυτή με τιμή 0 για τον τερματισμό της συντακτικής ανάλυσης.

Σχόλια

Τα σχόλια στην **SimpleFortran** είναι συμβολοσειρές που ξεκινάνε από το χαρακτήρα '$' με εύρος μέχρι το τέλος της τρέχουσας γραμμής.

Β. Συντακτικοί Κανόνες

Η γραμματική της **SimpleFortran** περιγράφεται από τους πιο κάτω κανόνες:

```
program          → body END subprograms

body             → declarations statements

declarations     → declarations type vars
                 | declarations COMMON cblock_list
                 | declarations DATA vals
                 | ε

type             → INTEGER | REAL | COMPLEX | LOGICAL | CHARACTER | STRING

vars             → vars COMMA undef_variable
                 | undef_variable

undef_variable   → listspec ID LPAREN dims RPAREN
                 | listspec ID

listspec         → LIST | ε

dims             → dims COMMA dim
                 | dim

dim              → ICONST | ID

cblock_list      → cblock_list cblock
                 | cblock

cblock           → DIVOP ID DIVOP id_list

id_list          → id_list COMMA ID
                 | ID

vals             → vals COMMA ID value_list
                 | ID value_list

value_list       → DIVOP values DIVOP

values           → values COMMA value
                 | value

value            → repeat sign constant
                 | ADDOP constant
                 | constant

repeat           → ICONST MULOP
                 | MULOP

sign             → ADDOP | ε

constant         → simple_constant
                 | complex_constant

simple_constant → ICONST | RCONST | LCONST | CCONST | SCONST

complex_constant → LPAREN RCONST COLON sign RCONST RPAREN
```

```
statements       → statements labeled_statement
                 | labeled_statement

labeled_statement → label statement
                   | statement
label            → ICONST

statement        → simple_statement
                 | compound_statement

simple_statement → assignment
                 | goto_statement
                 | if_statement
                 | subroutine_call
                 | io_statement
                 | CONTINUE
                 | RETURN
                 | STOP

assignment       → variable ASSIGN expression

variable         → ID LPAREN expressions RPAREN
                 | LISTFUNC LPAREN expression RPAREN
                 | ID

expressions      → expressions COMMA expression
                 | expression

expression       → expression OROP expression
                 | expression ANDOP expression
                 | expression RELOP expression
                 | expression ADDOP expression
                 | expression MULOP expression
                 | expression DIVOP expression
                 | expression POWEROP expression
                 | NOTOP expression
                 | ADDOP expression
                 | variable
                 | simple_constant
                 | LENGTH LPAREN expression RPAREN
                 | NEW LPAREN expression RPAREN
                 | LPAREN expression RPAREN
                 | LPAREN expression COLON expression RPAREN
                 | listexpression

listexpression   → LBRACK expressions RBRACK
                 | LBRACK RBRACK

goto_statement   → GOTO label
                 | GOTO ID COMMA LPAREN labels RPAREN

labels           → labels COMMA label
                 | label

if_statement     → IF LPAREN expression RPAREN label COMMA label COMMA label
                 | IF LPAREN expression RPAREN simple_statement

subroutine_call → CALL variable

io_statement     → READ read_list
                 | WRITE write_list
```

```
read_list        → read_list COMMA read_item
                 | read_item

read_item        → variable
                 | LPAREN read_list COMMA ID ASSIGN iter_space RPAREN

iter_space       → expression COMMA expression step

step             → COMMA expression
                 | ε

write_list       → write_list COMMA write_item
                 | write_item

write_item       → expression
                 | LPAREN write_list COMMA ID ASSIGN iter_space RPAREN

compound_statement → branch_statement
                   | loop_statement

branch_statement → IF LPAREN expression RPAREN THEN body tail

tail             → ELSE body ENDIF
                 | ENDIF

loop_statement   → DO ID ASSIGN iter_space body ENDDO

subprograms      → subprograms subprogram
                 | ε

subprogram       → header body END

header           → type listspec FUNCTION ID LPAREN formal_parameters RPAREN
                 | SUBROUTINE ID LPAREN formal_parameters RPAREN
                 | SUBROUTINE ID

formal_parameters → type vars COMMA formal_parameters
                  | type vars
```

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 **SimpleFortran** αποτελεί το **"program"**.

Γ. Σημασιολογία

Η σημασιολογία της **SimpleFortran** 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 **SimpleFortran** σε μη διφορούμενη. Όλοι οι υπόλοιποι κανόνες κωδικοποιούνται στο μεταγλωττιστή της **SimpleFortran** με τη βοήθεια σημασιολογικών ρουτινών που εκτελούνται κατά τη μετατροπή ενός αρχικού προγράμματος σε ενδιάμεσο κώδικα.

Τύποι δεδομένων

Η **SimpleFortran** υποστηρίζει τέσσερις βασικούς τύπους δεδομένων:

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

Το μέγεθος και η αναπαράσταση των δεδομένων καθενός από τους βασικούς αριθμητικούς τύπους της **SimpleFortran** καθορίζονται από την τελική γλώσσα μεταγλώττισης. Αυτή επιτρέπει μεγέθη τύπων και αναπαραστάσεις ανάλογα με την υποστήριξη που δίνει η αρχιτεκτονική για τον καθένα. Η ευθυγράμμιση που απαιτείται για την προσπέλαση δεδομένων καθενός από τους τύπους αυτούς καθορίζεται επίσης από την τελική γλώσσα.
Όσο αφορά τον τύπο logical, το μέγεθος των δεδομένων του είναι το ελάχιστο που επιτρέπει η τελική γλώσσα, συνήθως ένα byte, ενώ η αναπαράστασή τους γίνεται με τη σταθερά 0 για την τιμή «Ψευδής» και τη σταθερά 1 για την τιμή «Αληθής».
Παρόμοια, το μέγεθος των δεδομένων του τύπου character είναι επίσης το ελάχιστο που επιτρέπει η τελική γλώσσα. Η αναπαράστασή τους γίνεται με τον κώδικα ASCII.

Εκτός από τους πιο πάνω βασικούς τύπους, η **SimpleFortran** υποστηρίζει και τέσσερις ακόμα τύπους.

Ο τύπος complex είναι ο αριθμητικός τύπος των μιγαδικών αριθμών. Συμμετέχει στη γραμματική της **SimpleFortran** ως βασικός, γι' αυτό και στη συνέχεια θα συμπεριλαμβάνεται σε αυτούς. Σε αντίθεση όμως με τους πιο πάνω τύπους, ο τύπος complex παράγεται από τον τύπο real, καθώς οι τιμές του προκύπτουν από δύο τιμές αριθμών τύπου real, που αντιστοιχούν στο πραγματικό και στο φανταστικό μέρος ενός μιγαδικού αριθμού. Μια τιμή τύπου complex μπορεί να παρασταθεί με παράθεση των δύο μερών του, διαχωρισμένων με το σύμβολο ':', και τον παραγόμενο μιγαδικό αριθμό να τοποθετείται σε παρενθέσεις. Τα δεδομένα τύπου complex έχουν μέγεθος δύο δεδομένων τύπου real, και αποθηκεύονται στη μνήμη συνεχόμενα, ανάλογα με την ευθυγράμμιση του τύπου real.

Ο τύπος string είναι ο τύπος των ορμαθών χαρακτήρων. Όπως και ο προηγούμενος, συμμετέχει στη γραμματική της **SimpleFortran** ως βασικός, γι' αυτό και στη συνέχεια θα συμπεριλαμβάνεται σε αυτούς. Ο τύπος string παράγεται από τον τύπο character, καθώς οι τιμές του προκύπτουν σαν ένα διάνυσμα τιμών τύπου character, με την τελευταία τιμή να είναι πάντα 0. Ο τύπος string είναι διατεταγμένος με την λεξικογραφική σειρά των ορμαθών χαρακτήρων που παριστάνονται, με βάση τον κώδικα ASCII των επιμέρους χαρακτήρων. Η αποθήκευση των χαρακτήρων ενός ορμαθού γίνεται σε συνεχόμενες θέσεις μνήμης, μέσα σε χώρο μεγέθους 256 φορές το μέγεθος του τύπου character. Αυτό σημαίνει ότι το μήκος ενός ορμαθού – συμπεριλαμβανομένου του τερματικού 0 – δε μπορεί να ξεπερνάει το 256.

Η **SimpleFortran** υποστηρίζει δύο σύνθετους τύπους, τους τύπους πίνακα και λίστας, με στοιχεία δεδομένα του ίδιου βασικού τύπου.
Ένα στοιχείο πίνακα αναπαριστάται με το όνομα του πίνακα και μέσα σε παρενθέσεις μια λίστα από θετικούς ακέραιους δείκτες, που αντιστοιχούν στις διαστάσεις του πίνακα. Δείκτες διαδοχικών διαστάσεων διαχωρίζονται με το σύμβολο ",".
Παράδειγμα αναφοράς σε στοιχείο πίνακα είναι το εξής:

```
C(103,5)
```

Η δεικτοδότηση ενός πίνακα N στοιχείων σε οποιαδήποτε διάσταση γίνεται με ακέραιες τιμές δείκτη από 1 έως N. Έτσι, οι παρακάτω αναφορές σε στοιχεία πίνακα δεν είναι αποδεκτές:

```
C(0,5,8)
C(3.5)
C(-3,1,10)
```

Δεικτοδότηση με τη βοήθεια μεταβλητής επιτρέπεται μόνο αν ο τύπος του δείκτη είναι ακέραιος. Έτσι, η αναφορά:

```
C(i,k+2)
```

είναι αποδεκτή, μόνο αν τα αναγνωριστικά i και k είναι ακέραιου τύπου.
Όσο αφορά τον τύπο λίστας, αναφορά στα στοιχεία του γίνεται με τη βοήθεια ειδικών συναρτήσεων στοιχείων λίστας.

Οι συναρτήσεις αυτές ορίζονται παρακάτω. Μια ολόκληρη λίστα 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]

Η αποθήκευση των στοιχείων ενός πίνακα της **SimpleFortran** στη μνήμη γίνεται σε συνεχόμενες θέσεις, ανάλογα με την ευθυγράμμιση του τύπου αυτών, με αύξουσα σειρά μεταβολής των δεικτών από την πρώτη διάσταση προς την τελευταία. Για παράδειγμα, ένας τρισδιάστατος πίνακας A 3×3×2 στοιχείων αποθηκεύει τα στοιχεία του με τη σειρά: A(1,1,1), A(2,1,1), A(3,1,1), A(1,2,1), A(2,2,1), A(3,2,1), A(1,3,1), A(2,3,1), A(3,3,1), A(1,1,2), A(2,1,2), A(3,1,2), A(1,2,2), A(2,2,2), A(3,2,2), A(1,3,2), A(2,3,2) και A(3,3,2). Με τον τρόπο αυτό, ένας δισδιάστατος πίνακας αποθηκεύεται στήλη-στήλη.
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 λίστας της **SimpleFortran** 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.

Δομή ενός προγράμματος SimpleFortran

Ένα πρόγραμμα σε γλώσσα **SimpleFortran** αποτελείται από την κύρια μονάδα και μονάδες υποπρογραμμάτων. Κάθε υποπρόγραμμα έχει την ίδια δομή με την κύρια μονάδα, και μια επικεφαλίδα που καθορίζει τις παραμέτρους του.
Η **SimpleFortran** – όπως η κλασική FORTRAN – δε διαθέτει καθολικές μεταβλητές, σε αντίθεση με τις περισσότερες σύγχρονες γλώσσες προγραμματισμού. Έτσι η επικοινωνία μεταξύ των μονάδων του προγράμματος γίνεται είτε μέσω των παραμέτρων των υποπρογραμμάτων, είτε μέσω κοινών περιοχών (common blocks). Οι τελευταίες είναι περιοχές μνήμης, όπου έχουν δηλωθεί τοπικές μεταβλητές δύο ή περισσότερων μονάδων του προγράμματος κάτω από το ίδιο όνομα περιοχής.

Κάθε μονάδα περιέχει:

- Δηλώσεις μεταβλητών (προαιρετικά): Οι μεταβλητές αυτές έχουν εμβέλεια τη μονάδα.
- Εσωτερικές εμβέλειες ορίζονται με κάθε δομημένη εντολή της **SimpleFortran**, κι επομένως μια μονάδα μπορεί να έχει φωλιασμένες εμβέλειες.
- Δηλώσεις κοινών περιοχών (προαιρετικά): Οι μεταβλητές που δηλώθηκαν νωρίτερα μπορούν να υπαχθούν σε κάποια κοινή περιοχή του προγράμματος.
- Αποδόσεις αρχικών τιμών (προαιρετικά): Οι μεταβλητές που δηλώθηκαν νωρίτερα μπορούν να αρχικοποιούνται στην αρχή της εκτέλεσης του προγράμματος.
- Εντολές: Τουλάχιστον μια εντολή πρέπει να υπάρχει σε κάθε μονάδα, όπως προκύπτει κι από τους συντακτικούς κανόνες της γλώσσας. Οι εντολές χωρίζονται σε απλές και δομημένες, με τις τελευταίες να μπορούν να περιέχουν δηλώσεις μεταβλητών και τουλάχιστον μια άλλη απλή ή δομημένη εντολή.
- Ετικέτες: Μιας εντολής μπορεί να προηγείται μια ετικέτα, που είναι μια ακέραια σταθερά με ορατότητα την τρέχουσα εμβέλεια. Η ετικέτα αναπαριστά τη διεύθυνση της εντολής στον κώδικα του προγράμματος και μπορεί να χρησιμοποιηθεί σε συνδυασμό με κάποιες εντολές της **SimpleFortran**.
- Δήλωση τερματισμού μονάδας end.

Οι δηλώσεις των αναγνωριστικών της **SimpleFortran** πρέπει να είναι μοναδικές σε κάθε εμβέλεια, είτε αυτά αντιστοιχούν σε υποπρογράμματα, είτε σε μεταβλητές, είτε σε κοινές περιοχές. Στην εξωτερική εμβέλεια του προγράμματος ανήκουν τα ονόματα των υποπρογραμμάτων και των κοινών περιοχών, ενώ στις εσωτερικές εμβέλειες ανήκουν τα ονόματα των μεταβλητών. Η επίλυση αναφοράς σε μη τοπικά ονόματα γίνεται με στατικό δέσιμο.

Η εκτέλεση του κώδικα μιας μονάδας ξεκινάει με την πρώτη εντολή της μονάδας, ενώ τερματίζεται με ειδικές εντολές της **SimpleFortran**.

Δηλώσεις μεταβλητών

Οι μεταβλητές δηλώνονται στην αρχή των εμβελειών μιας μονάδας. Κάθε μεταβλητή έχει ορατότητα την εμβέλεια στην οποία δηλώνεται, εκτός από όσες φωλιασμένες εμβέλειες την επισκιάζουν.

Μια δήλωση μεταβλητών αποτελείται από:
(α) το όνομα ενός βασικού τύπου,
(β) ένα ή περισσότερα αναγνωριστικά που αποδίδονται σε μεταβλητές αυτού του τύπου,
(γ) πριν από ένα αναγνωριστικό την προαιρετική λέξη-κλειδί list που μετατρέπει τη δήλωση γι' αυτό το αναγνωριστικό σε δήλωση λίστας, και
(δ) μετά από ένα αναγνωριστικό δηλώσεις διαστάσεων για μετατροπή της δήλωσης αυτού του αναγνωριστικού σε δήλωση πίνακα.

Παραδείγματα δηλώσεων είναι τα παρακάτω:
```
integer i, j, k logical l real dev
complex c(100,10), list d, e(5) string str(2) real a, list b, z(5), p(100), q
```

Ειδικότερα:
1. Κάθε δήλωση γίνεται για ένα μόνο βασικό τύπο. Περισσότερες δηλώσεις για τον ίδιο τύπο μπορούν να ακολουθούν παρακάτω.
2. Για έναν τύπο πίνακα, η δήλωσή του γίνεται με βάση τον τύπο των στοιχείων του. Ο αριθμός και το μέγεθος των διαστάσεων του πίνακα καθορίζονται σε παρένθεση δίπλα στο όνομα του πίνακα. Έτσι, με: `c(100,10)` δηλώνεται ένας δισδιάστατος πίνακας 100 γραμμών και 10 στηλών.
3. Για έναν τύπο λίστας, η δήλωσή του γίνεται επίσης με βάση τον τύπο των στοιχείων του. Ο τύπος των στοιχείων μιας λίστας δεν επιτρέπεται να είναι string.
4. Η ίδια μεταβλητή δε μπορεί να δηλώνεται ταυτόχρονα ως πίνακας και ως λίστα, παρόλο που αυτό δεν αποκλείεται από τη σύνταξη της **SimpleFortran**. Τέτοια περίπτωση δήλωσης αποτελεί σημασιολογικό σφάλμα.
5. Σε κάθε δήλωση μπορούν να συνυπάρχουν αναγνωριστικά βαθμωτών μεταβλητών, μεταβλητών τύπου πίνακα και μεταβλητών τύπου λίστας.

Όλες οι μεταβλητές της **SimpleFortran** – εκτός από τις τυπικές παραμέτρους ενός υποπρογράμματος – είναι στατικές. Δεν τοποθετούνται στη στοίβα, αλλά έχουν σταθερές διευθύνσεις έξω από αυτή, και έχουν διάρκεια ζωής όλη τη διάρκεια εκτέλεσης του προγράμματος.

Δηλώσεις κοινών περιοχών

Η δήλωση μιας κοινής περιοχής αντιστοιχεί μια σειρά από μεταβλητές σε κάποιο συγκεκριμένο χώρο στατικών δεδομένων του προγράμματος, το οποίο και ονομάζει. Οι μεταβλητές της κοινής περιοχής τοποθετούνται σε διαδοχικές θέσεις στο χώρο αυτό, με τη σειρά που αναφέρονται στη δήλωση της κοινής περιοχής, και πάντα σύμφωνα με την ευθυγράμμισή τους. Τα ονόματα των κοινών περιοχών έχουν καθολική εμβέλεια στο πρόγραμμα και μπορούν να χρησιμοποιηθούν από περισσότερες μονάδες αυτού. Έτσι, καθορίζεται ένας έμμεσος τρόπος επικοινωνίας μεταξύ των μονάδων ενός προγράμματος. Δηλαδή, μεταβλητές από διαφορετικές μονάδες με το ίδιο ή διαφορετικό όνομα, που έχουν όμως δηλωθεί στην ίδια κοινή περιοχή, μπορούν να έχουν ίδιες τιμές, επειδή μοιράζονται τις ίδιες θέσεις μνήμης.

Μια δήλωση κοινών περιοχών αποτελείται από:
(α) τη λέξη-κλειδί **common**,
(β) το όνομα που αποδίδεται σε μια περιοχή,
(γ) ένα ή περισσότερα αναγνωριστικά μεταβλητών που έχουν νωρίτερα δηλωθεί, και οι οποίες τοποθετούνται στην περιοχή, και
(δ) αν θέλουμε, άλλες κοινές περιοχές.

Παραδείγματα δηλώσεων κοινών περιοχών είναι τα πιο κάτω:
```
common /bl1/x,y /bl2/a common /bl1/i,j,k
```

Αν οι παραπάνω δηλώσεις γίνονται σε διαφορετικές μονάδες του προγράμματος, και οι μεταβλητές x και y, είναι δηλωμένες με τον ίδιο τύπο με τις i και j, αντίστοιχα, τότε όποια αλλαγή συμβεί στη μεταβλητή x στη μια μονάδα επηρεάζει και τη μεταβλητή i της άλλης μονάδας, και όποια αλλαγή συμβεί στη μεταβλητή i στη δεύτερη επηρεάζει και τη μεταβλητή x στην πρώτη. Παρόμοια και για τις μεταβλητές y και j.

Ειδικότερα:
1. Οι δηλώσεις κοινών περιοχών γίνονται με τη βοήθεια του διαχωριστικού συμβόλου '/' – ταυτόσημου με τον τελεστή της διαίρεσης divop, που τοποθετείται πριν και μετά από τα ονόματα των κοινών περιοχών.
2. Όλες οι μεταβλητές που εμφανίζονται σε δήλωση κοινών περιοχών μιας μονάδας πρέπει να έχουν δηλωθεί νωρίτερα στη μονάδα, και μπορούν να είναι οποιουδήποτε τύπου. Ειδικά για μια μεταβλητή τύπου λίστας, η κοινή περιοχή αναφέρεται μόνο στην καθαυτή μεταβλητή και όχι σε ολόκληρη τη λίστα.

3. Οι μεταβλητές μιας κοινής περιοχής που δηλώνεται σε δύο διαφορετικές μονάδες του προγράμματος δεν πρέπει να είναι υποχρεωτικά δηλωμένες με τον ίδιο τύπο. Αν στο πιο πάνω παράδειγμα, με τις δύο δηλώσεις σε διαφορετικές μονάδες, η μεταβλητή x έχει δηλωθεί ως πίνακας δύο στοιχείων τύπου integer, ενώ οι μεταβλητές y, i, j και k έχουν δηλωθεί ως βαθμωτές τύπου integer, τα στοιχεία x(1) και x(2) θα βρίσκονται στις ίδιες θέσεις του χώρου δεδομένων με τις μεταβλητές i και j, ενώ η μεταβλητή y θα βρίσκεται στην ίδια θέση με τη μεταβλητή k. Η ορθότητα στην επικοινωνία των μονάδων με τον τρόπο αυτό είναι στην ευθύνη του προγραμματιστή, και όχι του μεταγλωττιστή.
4. Μια μεταβλητή μιας μονάδας μπορεί να εμφανίζεται σε μία μόνο κοινή περιοχή του προγράμματος, ενώ θα πρέπει να εμφανίζεται μία μόνο φορά στην περιοχή αυτή.
5. Μια κοινή περιοχή μπορεί να δηλώνεται ξανά στην ίδια μονάδα. Η νέα δήλωση θεωρείται ότι συνεχίζει την προηγούμενη, και οι μεταβλητές που εμφανίζονται σε αυτή τοποθετούνται αμέσως μετά από τις μεταβλητές της προηγούμενης δήλωσης. Αν στο πιο πάνω παράδειγμα οι δύο δηλώσεις ήταν στην ίδια μονάδα, οι μεταβλητές i, j και k θα τοποθετούνταν στο χώρο δεδομένων αμέσως μετά τη μεταβλητή y.
6. Μια κοινή περιοχή μπορεί να περιέχει μία μόνο μεταβλητή.
7. Δηλώσεις κοινών περιοχών επιτρέπονται μόνο στην πιο εξωτερική εμβέλεια των μονάδων.

Αξίζει να σημειωθεί ότι δε θα μπορούσαμε να έχουμε κοινές περιοχές στη **SimpleFortran**, αν οι μεταβλητές των μονάδων δεν ήταν στατικές, ώστε να έχουν σταθερές διευθύνσεις.

Αποδόσεις αρχικών τιμών

Οι μεταβλητές μιας μονάδας μπορούν να αρχικοποιηθούν με ειδική δήλωση στην αρχή της μονάδας.

Μια δήλωση απόδοσης αρχικών τιμών αποτελείται από:
(α) τη λέξη-κλειδί data,
(β) το όνομα μιας μεταβλητής που θέλουμε να αρχικοποιήσουμε,
(γ) μια λίστα αρχικών τιμών που αποδίδονται στη μεταβλητή, και
(δ) αν θέλουμε, αποδόσεις αρχικών τιμών για άλλες μεταβλητές.

Παραδείγματα δηλώσεων απόδοσης αρχικών τιμών είναι τα πιο κάτω:
```
data i/1/, x/.true./, c/(3.2:1.8),4*(3.1:0.3),(0.:-3.4)/
data a/8.e-2/, z/2*1., 3.2, 2*2.5/, p/*-1./
data s/2*"string", "another string"/, v/'g',10*'o'/
```

Ειδικότερα:
1. Οι δηλώσεις απόδοσης αρχικών τιμών γίνονται με τη βοήθεια του διαχωριστικού συμβόλου ‘/’, που τοποθετείται πριν και μετά από μια λίστα αρχικών τιμών.
2. Όλες οι μεταβλητές που εμφανίζονται σε δήλωση απόδοσης αρχικών τιμών μιας μονάδας πρέπει να έχουν δηλωθεί νωρίτερα στη μονάδα. Ο μεταγλωττιστής πρέπει να ελέγχει, αν οι αρχικές τιμές που αποδίδονται είναι τύπου συμβατού για ανάθεση με τον τύπο της αντίστοιχης μεταβλητής, ή, για τύπο πίνακα, με τον τύπο των στοιχείων του πίνακα. Η συμβατότητα για ανάθεση ορίζεται πιο κάτω.
3. Μια απόδοση αρχικών τιμών σε μεταβλητή τύπου πίνακα γίνεται με απόδοση των τιμών της λίστας σε διαδοχικά στοιχεία του πίνακα, με τη σειρά αποθήκευσης που αναφέρθηκε νωρίτερα. Αν ο αριθμός των αρχικών τιμών που αποδίδονται σε έναν πίνακα δεν είναι ο ίδιος με τον αριθμό των στοιχείων του, τότε: αν είναι μικρότερος από το δεύτερο, τα υπόλοιπα στοιχεία του πίνακα θα έχουν μηδενική αρχική τιμή – ανάλογα με τον τύπο τους, διαφορετικά οι υπόλοιπες αρχικές τιμές αγνοούνται. Παρόμοια, αν σε μια βαθμωτή μεταβλητή αποδοθούν περισσότερες από μία τιμή, μόνο η πρώτη αρχικοποιεί τη μεταβλητή, ενώ οι υπόλοιπες αγνοούνται.
4. Σε αρχικοποίηση μεταβλητής τύπου string, οι διαδοχικοί χαρακτήρες του ορμαθού αποθηκεύονται σε διαδοχικές θέσεις του χώρου 256 θέσεων που έχουν δεσμευτεί για τη μεταβλητή, ξεκινώντας από την πρώτη θέση. Μετά τους χαρακτήρες του ορμαθού, οι υπόλοιπες θέσεις του χώρου μέχρι τις 256 λαμβάνουν τιμή 0. Ο ορμαθός που παρέχεται δεν πρέπει να έχει μήκος μεγαλύτερο από 255 χαρακτήρες. Απόδοση αρχικών τιμών για πίνακες στοιχείων τύπου string γίνεται με διαδοχικούς ορμαθούς χαρακτήρων.
5. Μιας τιμής μπορεί να προηγείται μια θετική ακέραια σταθερά σαν παράγοντας επανάληψης, με τη βοήθεια του συμβόλου ‘*’ – ταυτόσημου με τον τελεστή του πολλαπλασιασμού MULOP. Η παρουσία του συμβόλου ‘*’ χωρίς τον παράγοντα επανάληψης αποδίδει την αρχική τιμή που το ακολουθεί σε όλα τα υπόλοιπα στοιχεία του πίνακα, εκτός εάν ακολουθούν άλλες αρχικές τιμές για τον ίδιο πίνακα, οπότε αυτές θα αποδοθούν στο τέλος του. Για παράδειγμα, με:
   ```
   w/3*1, *2, 0/
   ```
   η τιμή 1 αποδίδεται στα 3 πρώτα στοιχεία του πίνακα w, η τιμή 0 στο τελευταίο στοιχείο του w, ενώ τα υπόλοιπα 6 στοιχεία του πίνακα παίρνουν τιμή 2. Σε κάθε αρχικοποίηση πίνακα, μπορούμε να έχουμε μόνο μια φορά το σύμβολο ‘*’ χωρίς παράγοντα επανάληψης. Αν ο αριθμός των αρχικών τιμών που προηγούνται και ακολουθούν

την τιμή που αποδίδεται με το σύμβολο αυτό δεν είναι μικρότερος από τον αριθμό των στοιχείων του πίνακα, η τιμή αυτή θα αγνοηθεί. Αν μάλιστα είναι μεγαλύτερος, θα αγνοηθούν και οι τιμές που περισσεύουν.

6. Οι αρχικές τιμές επιτρέπεται να είναι προσημασμένες, αν είναι αριθμητικού τύπου. Η χρήση προσήμου για τιμές τύπου logical, character ή string αποτελεί σημασιολογικό σφάλμα.
7. Μεταβλητές που βρίσκονται σε κάποια κοινή περιοχή του προγράμματος επιδέχονται αρχικοποίηση με δήλωση απόδοσης αρχικών τιμών. Αν όμως μεταβλητές της ίδιας κοινής περιοχής λαμβάνουν αρχικές τιμές σε παραπάνω από μια μονάδα του προγράμματος, το αποτέλεσμα της απόδοσης αρχικών τιμών στην περιοχή αυτή θα είναι απροσδιόριστο.
8. Δηλώσεις απόδοσης αρχικών τιμών επιτρέπονται μόνο στην πιο εξωτερική εμβέλεια των μονάδων.
9. Απόδοση αρχικής τιμής σε τύπο λίστας επιτρέπεται μόνο για τιμή 0. Απόδοση μη μηδενικής αρχικής τιμής αποτελεί σημασιολογικό σφάλμα.

Επειδή οι μεταβλητές της **SimpleFortran** είναι στατικές, οι αρχικές τιμές που τους αποδίδονται με τον παραπάνω τρόπο αποθηκεύονται κατ' ευθείαν σε κατάλληλο χώρο στον κώδικα που παράγει ο μεταγλωττιστής και δεν αποδίδονται κατά την εκτέλεση του προγράμματος.

Δηλώσεις υποπρογραμμάτων

Τα υποπρογράμματα της **SimpleFortran** έχουν καθολική εμβέλεια και δηλώνονται στο εξωτερικό περιβάλλον, αμέσως μετά την κύρια μονάδα του προγράμματος.
Ένα υποπρόγραμμα δε χρειάζεται να έχει δηλωθεί πριν την κλήση του, γεγονός που έχει σαν αποτέλεσμα ο έλεγχος ορθότητας της κλήσης να πρέπει να γίνεται εκ των υστέρων – όταν δηλαδή συναντηθεί η δήλωσή του – με την τεχνική του μπαλώματος (backpatching), δηλαδή με επιστροφή της ανάλυσης στην εντολή κλήσης.

Τα υποπρογράμματα της **SimpleFortran** διακρίνονται σε υπορουτίνες (subroutines) και συναρτήσεις (functions). Οι υπορουτίνες καλούνται μέσα από ειδική εντολή της **SimpleFortran**, ενώ οι συναρτήσεις μέσα από εκφράσεις. Οι συναρτήσεις επιστρέφουν αποτελέσματα μέσα από τα ονόματά τους που για το σκοπό αυτό λειτουργούν και σαν καθολικές μεταβλητές του προγράμματος.

Κάθε υποπρόγραμμα είναι μια μονάδα, που εκτός από τη δομή της μονάδας, διαθέτει και επικεφαλίδα. Αυτή περιέχει με τη σειρά:
(α) τον τύπο του αποτελέσματος, εάν το υποπρόγραμμα είναι συνάρτηση,
(β) το είδος του υποπρογράμματος,
(γ) το όνομά του, και
(δ) τις τυπικές παραμέτρους του.

Παραδείγματα επικεφαλίδων υποπρογραμμάτων είναι τα εξής:

```
real list function LL (integer y, real list w)
subroutine A(integer n, x(n), logical z(100))
subroutine S
```

Ειδικότερα:

1. Η δήλωση των τυπικών παραμέτρων είναι πλήρης, περιλαμβάνει δηλαδή τον τύπο και τις διαστάσεις τους. Έτσι, πριν από μια λίστα παραμέτρων πρέπει να δηλώνεται ο τύπος τους, και εάν μια παράμετρος είναι τύπου πίνακα, η δήλωση των διαστάσεών της είναι υποχρεωτική. Μια παράμετρος μπορεί να είναι τύπου λίστας – αλλά όχι ταυτόχρονα και λίστας και πίνακα.
2. Μία παράμετρος τύπου πίνακα μπορεί να δηλωθεί με εικονικό μέγεθος των διαστάσεών της, με τη βοήθεια ενός ή περισσότερων αναγνωριστικών. Τα αναγνωριστικά αυτά πρέπει να συγκαταλέγονται στις παραμέτρους του υποπρογράμματος, να προηγούνται του ονόματος του πίνακα, και βέβαια να δηλώνονται ως τύπου integer.
3. Το πέρασμα μιας παραμέτρου γίνεται κατ' αναφορά, εάν υπάρχει ανάθεση σε αυτήν μέσα στο υποπρόγραμμα, διαφορετικά γίνεται κατ' αξία. Οι παράμετροι τύπου string, πίνακα και λίστας περνούν κατ' αναφορά.
4. Μια υπορουτίνα μπορεί να μην έχει παραμέτρους, μια συνάρτηση όμως πρέπει να έχει τουλάχιστον μία παράμετρο.
5. Ο τύπος του αποτελέσματος μιας συνάρτησης πρέπει να είναι βασικός εκτός από string, ή λίστας. Στην τελευταία περίπτωση τιμή αποτελέσματος θα είναι η διεύθυνση του πρώτου στοιχείου της λίστας.

Η κλασική FORTRAN δεν υποστηρίζει αναδρομή. Έτσι, κάθε υποπρόγραμμα έχει ακριβώς ένα ενεργό αντίγραφο του χώρου δεδομένων του, το οποίο καθορίζεται στατικά από το μεταγλωττιστή. Η **SimpleFortran** υποστηρίζει αναδρομή μειωμένων δυνατοτήτων, χρησιμοποιώντας τη στοίβα για τη μετάδοση των παραμέτρων του υποπρογράμματος και για την επιστροφή του αποτελέσματος μιας συνάρτησης. Κατά τα άλλα, ακολουθεί την κλασική FORTRAN για τον υπόλοιπο χώρο δεδομένων, κι επομένως οι μεταβλητές μιας αναδρομικής συνάρτησης είναι κοινές για όλες τις

αναδρομικές κλήσεις της συνάρτησης..

Εκφράσεις

Η **SimpleFortran** υποστηρίζει αριθμητικές εκφράσεις, λογικές εκφράσεις και εκφράσεις λίστας. Οι τιμές των πρώτων μπορούν να είναι τύπου integer, real ή complex, οι τιμές των δεύτερων μπορούν να είναι μόνο τύπου logical, ενώ οι τιμές των τρίτων είναι σύνθετου τύπου λίστας. Οι εκφράσεις αποτιμώνται με βάση συγκεκριμένους κανόνες και με τη βοήθεια των τελεστών της **SimpleFortran**. Οι αριθμητικές εκφράσεις μπορούν να περιέχονται σε λογικές εκφράσεις, αλλά όχι το αντίστροφο.
Εκτός από τις πιο πάνω, ορίζονται και απλές εκφράσεις τύπου character και string, οι οποίες είτε εμφανίζονται μεμονωμένα σε κλήσεις υποπρογραμμάτων ή εντολές της **SimpleFortran**, είτε περιέχονται σε λογικές εκφράσεις. Ειδικότερα, εκφράσεις τύπου string, μπορούν σε κάποιες περιπτώσεις να συμμετέχουν και σε σύνθετες εκφράσεις. Τέλος, ορίζονται και απλές εκφράσεις σύνθετου τύπου πίνακα που εμφανίζονται μόνο σε κλήσεις υποπρογραμμάτων.

Μια έκφραση της **SimpleFortran** περιλαμβάνει:

- Τιμές αριστερής προσπέλασης (L-values), δηλαδή τιμές διευθύνσεων του χώρου δεδομένων του προγράμματος που αντιστοιχούν σε μεταβλητές, στοιχεία πινάκων ή στοιχεία λίστας.
- Σταθερές, δηλαδή τιμές που υπάρχουν στον αρχικό κώδικα της μονάδας.
- 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 **SimpleFortran** είναι τα εξής:

```
i a+1
a(3,i+1,j-1)**k**2 + (tu(b,z(2))/(i-1))*i
a+b .gt. 0. .and. .not. x .eq. 'g' .or. s .lt. "nikos"
z+(w/2:-2.)/(1.2:3.1)
[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 **SimpleFortran** αποτελούν οι μεταβλητές, τα στοιχεία πινάκων και τα στοιχεία λίστας.

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 ή του στοιχείου πίνακα ή λίστας. Απαραίτητη προϋπόθεση για αποτίμηση είναι η προηγούμενη δήλωση της αντίστοιχης μεταβλητής, του αντίστοιχου πίνακα ή της αντίστοιχης λίστας, στην ίδια εμβέλεια ή σε κάποια εξωτερική της. Σε περίπτωση απουσίας τέτοιας δήλωσης υπάρχει σημασιολογικό σφάλμα. Μεταβλητή τύπου πίνακα μπορεί να εμφανιστεί χωρίς δείκτες σε μια έκφραση, μόνο εάν αποτελεί πραγματική παράμετρο στην κλήση ενός υποπρογράμματος.

Η αποτίμηση μεταβλητής που έχει δηλωθεί κανονικά σε μια μονάδα γίνεται με ανάγνωση της τιμής της μεταβλητής από το χώρο δεδομένων.
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 επιτρεπόμενα 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 και το μέγεθός του 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 **SimpleFortran** 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 λίστα 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.

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, οπότε και μεταδίδεται κατ' αναφορά, η προσπέλαση κάποιου στοιχείου της γίνεται όπως αναφέρθηκε παραπάνω.

Σταθερές

Οι σταθερές της **SimpleFortran** είναι αυτές που αναγνωρίζονται άμεσα από το λεκτικό αναλυτή και περιγράφηκαν στην αντίστοιχη ενότητα, καθώς και σταθερές τύπου complex. Οι τελευταίες προκύπτουν από δύο σταθερές τύπου real, με βάση αντίστοιχο συντακτικό κανόνα της γλώσσας, σε συμφωνία με την προηγούμενη περιγραφή τιμών τύπου complex.
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, καθώς και των ορμαθών 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. Εξαίρεση αποτελεί το φανταστικό μέρος των μιγαδικών σταθερών, όπως φαίνεται και από τη σύνταξη αυτών.

Τελεστές

Τελεστές είναι ειδικά σύμβολα, που εφαρμοζόμενα σε έναν αριθμό εκφράσεων – που ονομάζονται *τελούμενα εισόδου* ή *τελεστέοι*, παράγουν μια νέα έκφραση.
Οι τελεστές της **SimpleFortran** δίνονται στο σχετικό πίνακα και διακρίνονται σε τελεστές με να 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:

Τελεστής	Περιγραφή	Αριθμός τελούμενων	Προσεταιριστικότητα
`'()'`	Αναφορά σε στοιχείο πίνακα, κλήση συνάρτησης	2, 2	-
POWEROP	Ύψωση σε δύναμη	2	δεξιά
MULOP, DIVOP	Πολλαπλασιασμός, διαίρεση	2, 2	αριστερή
ADDOP	Πρόσημο, πρόσθεση, αφαίρεση, ένωση λιστών και ορμαθών	1, 2, 2, 2	αριστερή
RELOP	Σχεσιακοί τελεστές	2	καμία
NOTOP	Λογική άρνηση	1	-
ANDOP	Λογικό γινόμενο	2	αριστερή
OROP	Λογικό άθροισμα	2	αριστερή
`'(:)'`	Δημιουργία μιγαδικού αριθμού	2	-
`'[]'`	Σύνθεση λίστας	1	-

Τελεστές της **SimpleFortran** σε φθίνουσα σειρά προτεραιότητας

- Ο τελεστής αναφοράς σε στοιχείο πίνακα / κλήσης συνάρτησης '( )' έχει σύνταξη:
  <αναγνωριστικό> '(' <λίστα εκφράσεων> ')'

Ο τελεστής αυτός έχει δύο τελούμενα εισόδου, από τα οποία το πρώτο είναι το όνομα του πίνακα ή της συνάρτησης, και το δεύτερο μια λίστα εκφράσεων, οι τιμές των οποίων αποδίδονται στους δείκτες του στοιχείου πίνακα ή στις

πραγματικές παραμέτρους, αντίστοιχα. Η αποτίμηση των εκφράσεων στη λίστα γίνεται από τα αριστερά προς τα δεξιά. Η αναφορά σε στοιχείο πίνακα έχει περιγραφεί νωρίτερα, ενώ η κλήση συνάρτησης θα επεξηγηθεί παρακάτω. Στη **SimpleFortran** η σύνταξη της κλήσης συναρτήσεων ταυτίζεται με την σύνταξη της αναφοράς σε στοιχείο πίνακα. Η σημασιολογική ανάλυση διαχωρίζει τη μια λειτουργία από την άλλη.
Ακόμα, ως πίνακας μπορεί να είναι και κάποια έκφραση αριστερής προσπέλασης τύπου complex, οπότε επιτρέπεται μόνο μία έκφραση δείκτη που αναφέρεται στο ένα από τα δύο μέρη του αριθμού. Έτσι, ο δείκτης 1 αναφέρεται στο πραγματικό και ο δείκτης 2 αναφέρεται στο φανταστικό μέρος του μιγαδικού.
Τέλος, ως πίνακας μπορεί να είναι και κάποια έκφραση αριστερής προσπέλασης τύπου string, οπότε επιτρέπεται μόνο μία έκφραση δείκτη που αναφέρεται σε ένα συγκεκριμένο χαρακτήρα του ορμαθού. Επειδή το μήκος του ορμαθού δεν είναι προκαθορισμένο, ο έλεγχος ορίων δείκτη πρέπει να γίνεται με βάση το τρέχον μήκος του ορμαθού. Η αρίθμηση των χαρακτήρων ενός ορμαθού γίνεται ξεκινώντας με 1 από την αρχή του ορμαθού.

- Ο τελεστής σύνθεσης λίστας ‘[ ]’ έχει σύνταξη:

‘[’ <ακολουθία εκφράσεων> ‘]’

Ο τελεστής αυτός έχει σα μοναδικό τελούμενο εισόδου μια ακολουθία εκφράσεων, οι τιμές των οποίων αποδίδονται στα διαδοχικά στοιχεία μιας λίστας. Η αποτίμηση των εκφράσεων στην ακολουθία γίνεται από τα αριστερά προς τα δεξιά. Η ακολουθία εκφράσεων μπορεί να είναι κενή, οπότε δημιουργείται μια κενή λίστα, κάτι που ισοδυναμεί με μηδενική διεύθυνση πρώτου στοιχείου.
Οι επιμέρους εκφράσεις της λίστας πρέπει να είναι του ίδιου βασικού 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 κατασκευαστούν 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.

- Οι τελεστές προσήμου ADDOP. Ο μοναδικός τελεστέος των τελεστών αυτών πρέπει να είναι αριθμητικού τύπου και το αποτέλεσμα της εφαρμογής τους είναι του ίδιου τύπου.
- Οι αριθμητικοί τελεστές POWEROP, MULOP, DIVOP και ADDOP. Η συμβατότητα τύπων των τελεστέων αυτών και ο τύπος του αποτελέσματος της εφαρμογής τους καθορίζονται στον επόμενο πίνακα:

A \ B	integer	real	complex
integer	integer	real	complex όχι POWEROP
real	real	real	complex όχι POWEROP
complex	complex	complex όχι POWEROP	complex όχι POWEROP

Τύπος αποτελέσματος αριθμητικής έκφρασης **A op B**

Όπως δείχνει ο πίνακας, η μόνη περίπτωση μη συμβατότητας τύπων αφορά τον τελεστή POWEROP, για τον οποίο το δεξί τελούμενο εισόδου δε μπορεί να είναι τύπου complex και, εάν το αριστερό τελούμενο είναι τύπου complex, το δεξί μπορεί να είναι μόνο τύπου integer. 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.
Ο τελεστής ADDOP ‘+’ εφαρμόζεται και σε τύπους λίστας και ορμαθών χαρακτήρων, οπότε υλοποιεί την ένωση δύο λιστών ή δύο ορμαθών χαρακτήρων. Η ένωση είναι η μόνο πράξη που επιτρέπεται στους δύο αυτούς τύπους. Ειδικότερα για λίστες, η ένωση 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. Για ορμαθούς χαρακτήρων, το αποτέλεσμα είναι ένας νέος ορμαθός που περιέχει τους χαρακτήρες του πρώτου χωρίς το τερματικό 0, ακολουθούμενους από τους χαρακτήρες του δεύτερου ορμαθού.

- Οι σχεσιακοί τελεστές RELOP. Τα τελούμενα εισόδου μπορεί να είναι: (α) αριθμητικού τύπου εκτός από complex, οπότε, εάν είναι διαφορετικού τύπου, ο κώδικας πρέπει να μετατρέπει το τελούμενο τύπου integer σε real, και μετά την πιθανή μετατροπή, να κάνει τησύγκριση για τον τύπο που προέκυψε, ή (β) τύπου character ή string, οπότε η σύγκριση γίνεται με βάση την κωδικοποίηση αυτών. Ειδικά για τους τελεστές ‘.eq.’ και ‘.ne.’ τα τελούμενα εισόδου μπορεί να είναι και τύπου complex. Σε κάθε περίπτωση το αποτέλεσμα είναι τύπου logical.
- Ο τελεστής λογικής άρνησης NOTOP. Ο μοναδικός τελεστέος του τελεστή αυτού πρέπει να είναι τύπου logical και το αποτέλεσμα είναι του ίδιου τύπου.
- Οι τελεστές λογικών πράξεων ANDOP και OROP. Τα τελούμενα εισόδου αυτών πρέπει να είναι τύπου logical. Το αποτέλεσμα της εφαρμογής τους είναι επίσης τύπου logical.

- Ο τελεστής δημιουργίας μιγαδικού αριθμού ‘(:)’, με σύνταξη:

‘(’ <έκφραση> ‘:’ <έκφραση> ‘)’

Τα δύο τελούμενα εισόδου του τελεστή αυτού πρέπει να είναι αριθμητικού τύπου εκτός από complex, και αν είναι τύπου integer, πρέπει να μετατραπούν σε real. Το αποτέλεσμα είναι ένας μιγαδικός αριθμός, στο πραγματικό μέρος του οποίου αποδίδεται η τιμή της αριστερής, και στο φανταστικό η τιμή της δεξιάς έκφρασης.

Παρά την αναπαράσταση των τύπων logical και character με ακέραιες τιμές, στη **SimpleFortran** δεν υπάρχει συμβατότητα μεταξύ αριθμητικών τιμών, λογικών τιμών και τιμών χαρακτήρων. Έτσι, η συμμετοχή αριθμητικών εκφράσεων σε λογικές μπορεί να γίνει μόνο μέσω των τελεστών RELOP. Αυτοί είναι οι μόνοι τελεστές που κατασκευάζουν λογικές εκφράσεις από αριθμητικές, καθώς μπορούν να δέχονται αριθμητικά τελούμενα και παράγουν αποτέλεσμα τύπου logical. Το αντίστροφο δεν είναι εφικτό, γι’ αυτό και δε μπορεί μια λογική έκφραση να συμμετέχει σε μια αριθμητική. Παρόμοια, τιμές τύπου character μπορούν να συμμετάσχουν σε λογικές εκφράσεις μόνο μέσω των τελεστών RELOP. Τιμές τύπου character δε μπορούν να συμμετάσχουν σε αριθμητικές εκφράσεις.
Εξαίρεση στα παραπάνω αποτελεί ο τελεστής ‘( )’ που από τη μία προσφέρει τη δυνατότητα *έμμεσης* συμμετοχής μιας έκφρασης σε μια μεγαλύτερη, αλλά και από την άλλη μπορεί να δώσει αποτέλεσμα τύπου που δε μπορούν να δώσουν άλλοι τελεστές.
Σε κάθε εφαρμογή τελεστή, και όταν συμμετέχουν περισσότερα από ένα τελούμενο εισόδου, η αποτίμηση αυτών γίνεται από αριστερά προς τα δεξιά. Σε κάθε περίπτωση απαιτείται αποτίμηση όλων των τελούμενων, μια που η **SimpleFortran** 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 .gt. y .gt. 0
```

είναι λανθασμένη. Εφαρμογή οποιουδήποτε από τους δύο τελεστές RELOP ‘.gt.’ δίνει αποτέλεσμα τύπου logical, που δεν επιτρέπει την εφαρμογή του άλλου.
Σε περίπτωση τελεστών με ένα τελούμενο εισόδου, δεν επιτρέπεται διαδοχική εφαρμογή αυτών, εάν έχουν την ίδια προτεραιότητα. Για παράδειγμα, η έκφραση:

```
x.gt.-+3
```

δεν είναι αποδεκτή, λόγω διαδοχικής εφαρμογής δύο τελεστών προσήμου ADDOP. Αντίθετα η έκφραση:

```
.not.-i.lt.-1
```

είναι αποδεκτή.

Ας θεωρήσουμε για παράδειγμα την έκφραση:

```
i*x-y-j*z .lt. -a/k/2 .and. .not. x(i+2,m)**r**2 .ge. 0.
```

στην οποία κατ’ αρχήν υποθέτουμε ότι οι τύποι των τιμών που συμμετέχουν είναι οι προβλεπόμενοι, καθώς και ότι x είναι δισδιάστατος πίνακας.
Στην έκφραση αυτή ο τελεστής ANDOP ‘.and.’ έχει τη μικρότερη προτεραιότητα και δε μπορεί να εφαρμοστεί πριν την αποτίμηση και των δύο τελεστέων του. Επίσης ο σχεσιακός τελεστής RELOP ‘.lt.’ 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 MULOP ‘*’ που έχει μεγαλύτερη προτεραιότητα από τον ADDOP. Στη συ-
νέχεια 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 MULOP ‘*’ που υπάρχει στο δεξί τελούμενο του προηγούμενου. Η εφαρμογή του πρώτου από αριστερά τελεστή DIVOP ‘/’ προηγείται του δεύτερου, ενώ ο τελεστής προσήμου ADDOP ‘-’ εφαρμόζεται μετά τους δύο διαδοχικούς DIVOP, επιτρέποντας στη συνέχεια την εφαρμογή του τελεστή RELOP ‘.lt.’.
Έτσι ολοκληρώνεται η αποτίμηση του αριστερού τελεστέου του τελεστή ANDOP ‘.and.’ και μπορούμε να συνεχίσουμε με την αποτίμηση του δεξιού.
Στο τελούμενο του τελεστή NOTOP ‘.not.’ αποτιμάται πρώτα το αριστερό τελούμενο του τελεστή RELOP ‘.ge.’. Αυτό έχει δύο διαδοχικές εφαρμογές του τελεστή POWEROP ‘**’, όπου η δεξιά προσεταιριστικότητα επιβάλλει πρώτα την εφαρμογή του πρώτου από τα *δεξιά*.
Για την εφαρμογή του δεύτερου τελεστή POWEROP απαιτείται η αποτίμηση ενός στοιχείου πίνακα. Γι’ αυτό προηγείται η αποτίμηση των εκφράσεων των δεικτών του, από αριστερά προς τα δεξιά, και ο έλεγχος ορθότητας των τιμών τους. Στη συνέχεια αποτιμάται το στοιχείο, και εφαρμόζεται ο δεύτερος τελεστής POWEROP.
Τελειώνοντας, εφαρμόζεται ο τελεστής RELOP ‘.ge.’, ακολουθούμενος από τον NOTOP. Η εφαρμογή του ANDOP μπορεί τώρα να πραγματοποιηθεί.

Σε περίπτωση που επιθυμούμε παρέμβαση στους πιο πάνω κανόνες πρέπει να χρησιμοποιήσουμε παρενθέσεις, όπως προβλέπει η σύνταξη των εκφράσεων της γλώσσας. Για παράδειγμα, η έκφραση:
`-(-i+1)*(a-(b-c))`
(α) επιτρέπει διαδοχική εφαρμογή του ίδιου τελεστή προσήμου, (β) επιβάλλει την αποτίμηση του δεξιού τελούμενου του ADDOP '-' πριν από το αριστερό, και (γ) επιβάλλει την εφαρμογή των τελεστών ADDOP '+' και '-' πριν από την εφαρμογή του MULOP '*'.
Ας σημειωθεί ότι 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(e1, e2, …, en)` 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, απ' όπου θα μπορεί να τους χρησιμοποιήσει η μονάδα της συνάρτησης. Για μια παράμετρο κατ' αναφορά, στη στοίβα τοποθετείται η αντίστοιχη διεύθυνση. Ειδικά για παράμετρο τύπου string, αν η πραγματική παράμετρος είναι ένας σταθερός ορμαθός χαρακτήρων και όχι τιμή αριστερής προσπέλασης, ο μεταγλωττιστής δεσμεύει χώρο για μια προσωρινή τοπική μεταβλητή τύπου string στο καλούν περιβάλλον, παράγει κώδικα που να αντιγράφει στο χώρο αυτό το περιεχόμενο του ορμαθού μαζί με το τερματικό 0, και περνάει πια ως πραγματική παράμετρο τη διεύθυνση αυτής της μεταβλητής. Όπως και στις αρχικοποιήσεις, ο ορμαθός που παρέχεται δεν πρέπει να έχει μήκος μεγαλύτερο από 255 χαρακτήρες.
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 που η συνάρτηση διατηρεί τις παραμέτρους και το αποτέλεσμά της, 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.

κτός 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 τιμήπαράμετρο, και με διεύθυνση επόμενου στοιχείου ίση με 0.
Η πρώτη από τις δύο παραπάνω συναρτήσεις μπορεί να εφαρμοστεί και σε έναν ορμαθό χαρακτήρων, οπότε επιστρέφει το πλήθος των χαρακτήρων του ορμαθού – χωρίς το τερματικό 0.
Σε περίπτωση που η συνάρτηση που καλείται δεν έχει ακόμα οριστεί, η θέση της εντολής κλήσης πρέπει να σημειωθεί για μετέπειτα επάνοδο της ανάλυσης σε αυτή, σύμφωνα με την τεχνική του μπαλώματος.

Εντολές
Η **SimpleFortran** υποστηρίζει απλές και δομημένες εντολές. Μια δομημένη εντολή της **SimpleFortran** περιέχει ένα ή δύο σύνολα άλλων εντολών που το καθένα έχει τη δική του εμβέλεια, στην οποία μπορούν να οριστούν μεταβλητές. Μέσα σε κάθε ένα από αυτά τα σύνολα πρέπει να υπάρχει τουλάχιστον μία άλλη εντολή, απλή ή δομημένη.
Οι απλές εντολές της **SimpleFortran** είναι:

- Η εντολή ανάθεσης
- Οι εντολές άμεσου άλματος
- Οι απλές εντολές ελέγχου ροής
- Η εντολή κλήσης υπορουτίνας
- Οι εντολές εισόδου/εξόδου

- Η εντολή continue
- Η εντολή return
- Η εντολή stop

Οι δομημένες εντολές της **SimpleFortran** είναι:

- Η εντολή διακλάδωσης
- Η εντολή βρόχου
- Η εντολή ανάθεσης

Η εντολή αυτή αποδίδει την τιμή μιας έκφρασης σε μια τιμή αριστερής προσπέλασης, δηλαδή σε μια μεταβλητή, ένα στοιχείο πίνακα ή ένα στοιχείο λίστας του χώρου δεδομένων του προγράμματος. Στην τελευταία περίπτωση, της ανάθεσης πρέπει να προηγηθεί η αποτίμηση των εκφράσεων των δεικτών και ο υπολογισμός της τιμής αριστερής προσπέλασης του στοιχείου. Η ανάθεση γίνεται με αποθήκευση της τιμής της έκφρασης στη διεύθυνση που παριστάνει η τιμή αριστερής προσπέλασης.
Παραδείγματα αναθέσεων είναι τα εξής:

```
a = 0.01
c(i-1, mm(j)) = x**2+1
s = NEW(b)
CDR(s) = [2] + CDR([5,3,-4])
```

όπου c και mm είναι πίνακες, ενώ s είναι λίστα.

Για να μπορεί να γίνει ανάθεση, πρέπει η τιμή της έκφρασης να είναι συμβατού τύπου με τον τύπο της αριστερής προσπέλασης.
Δύο τύποι είναι συμβατοί για ανάθεση, εάν: (α) ταυτίζονται, ή αλλιώς (β) είναι αριθμητικοί, εκτός από τύπο complex, 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.

Σε μια ανάθεση ενός ορμαθού χαρακτήρων σε μια μεταβλητή τύπου string, οι διαδοχικοί χαρακτήρες του ορμαθού αντιγράφονται σε διαδοχικές θέσεις του χώρου 256 θέσεων που έχουν δεσμευτεί για τη μεταβλητή, ξεκινώντας από την πρώτη θέση. Μετά τους χαρακτήρες του ορμαθού, αποθηκεύεται η τιμή 0, ενώ οι υπόλοιπες θέσεις του χώρου μέχρι τις 256 διατηρούν την όποια προηγούμενη τιμή τους. Ο ορμαθός που παρέχεται ως τιμή δεξιάς προσπέλασης στο δεξί μέλος της ανάθεσης δεν πρέπει να έχει μήκος μεγαλύτερο από 255 χαρακτήρες.

Ανάθεση μεταξύ σύνθετων τύπων επιτρέπεται μόνο για λίστες στοιχείων συμβατού τύπου. Τότε, 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 ξεκρέμαστων στοιχείων.

Μια ειδική μορφή ανάθεσης της **SimpleFortran** είναι η ανάθεση στη μεταβλητή που παριστάνεται από το όνομα μιας συνάρτησης. Η τιμή αριστερής προσπέλασης αυτής είναι διεύθυνση στη στοίβα. Κάθε συνάρτηση πρέπει να έχει τουλάχιστον μία τέτοια ανάθεση. Η ανάθεση αυτή κατά τα άλλα ακολουθεί τους πιο πάνω κανόνες.

Οι εντολές άμεσου άλματος

Αυτές είναι οι εντολές goto, που είναι δύο:

- Το αδέσμευτο goto, με παράμετρο μια ετικέτα. Η εντολή αυτή εκτελεί άλμα και μεταφέρει τη ροή του προγράμματος στην εντολή που ακολουθεί την ετικέτα. Παράδειγμα αδέσμευτου goto είναι το εξής:
  ```
  goto 100
  ```
- Το υπολογιζόμενο goto, με παραμέτρους μια βαθμωτή μεταβλητή τύπου integer και μια λίστα ετικετών σε παρενθέσεις. Η εντολή αυτή εκτελεί άλμα και μεταφέρει τη ροή του προγράμματος στην εντολή που ακολουθεί μια από τις ετικέτες της λίστας, με βάση την τιμή της μεταβλητής, ή συνεχίζει τη ροή στην επόμενη εντολή. Αν η λίστα περιέχει n ετικέτες και η τιμή της μεταβλητής είναι μεταξύ 1 και n, τότε με την τιμή αυτή να χρησιμοποιείται σα δείκτης στη λίστα, εκτελείται άλμα στην αντίστοιχη εντολή. Σε κάθε περίπτωση που η μεταβλητή έχει τιμή εκτός της παραπάνω περιοχής δεν εκτελείται άλμα. Παράδειγμα υπολογιζόμενου goto είναι το εξής:
  ```
  goto i,(100,200,400,100)
  ```
  όπου η μεταβλητή i πρέπει να έχει τιμή 1, 2, 3 ή 4, για να εκτελέσει το άλμα. Παρατηρήστε ότι οι ετικέτες της λίστας δεν είναι αναγκαστικά διαφορετικές μεταξύ τους.

Μια ετικέτα που συμμετέχει σε μία από τις παραπάνω εντολές πρέπει να είναι μοναδικά ορισμένη σε μια εμβέλεια και η ορατότητά της να καλύπτει την εντολή. Σαν αποτέλεσμα αυτού του περιορισμού, δε μπορεί να βρίσκεται σε εμβέλεια

εσωτερική αυτής στην οποία υπάρχει η εντολή.

Οιαπλέςεντολέςελέγχουροής

Αυτές είναι οι εντολές if, που είναι δύο:

- Το αριθμητικό if, με παραμέτρους μια αριθμητική έκφραση σε παρενθέσεις και μια λίστα τριών ετικετών. Η εντολή αυτή αποτιμά την έκφραση και εκτελεί άλμα στην εντολή που δείχνει η πρώτη ετικέτα, στην εντολή που δείχνει η δεύτερη ετικέτα, ή στην εντολή που δείχνει η τρίτη ετικέτα, αν η τιμή της έκφρασης είναι μικρότερη από, ίση με ή μεγαλύτερη από μηδέν, αντίστοιχα. Παράδειγμα αριθμητικού if είναι:

```
if (a(i)-x) 100,200,300
```

Οι ετικέτες της λίστας – που δεν είναι αναγκαστικά διαφορετικές μεταξύ τους – ακολουθούν τον ίδιο περιορισμό με τις ετικέτες στις εντολές άμεσου άλματος.

- Το λογικό if, με παραμέτρους μια λογική έκφραση σε παρενθέσεις και μια απλή εντολή χωρίς ετικέτα. Η εντολή αυτή αποτιμά τη λογική έκφραση και εκτελεί την απλή εντολή,μόνο αν η τιμή της έκφρασης είναι «Αληθής». Παράδειγμα λογικού if είναι:

```
if (x.gt.0..and.x.lt.a(i)) y(i) = i**x
```

Παρατηρήστε ότι και στις δύο παραπάνω εντολές ο τύπος της έκφρασης σε παρενθέσεις δεν καθορίζεται από τη σύνταξη, αλλά πρέπει να ελέγχεται κατά τη σημασιολογική ανάλυση.

Η εντολή κλήσης υπορουτίνας

Η εντολή αυτή μεταφέρει τη ροή του κώδικα σε κάποια υπορουτίνα. Παράδειγμα κλήσης υπορουτίνας είναι το παρακάτω:

```
call subA(x,n+1,matrix,s(i)**2)
```

Στην εντολή κλήσης υπορουτίνας 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.
Όπως και με τις συναρτήσεις, σε περίπτωση που η υπορουτίνα που καλείται δεν έχει ακόμα οριστεί, η θέση της εντολής κλήσης πρέπει να σημειωθεί για μετέπειτα επάνοδο της ανάλυσης σε αυτή.

Οι εντολές εισόδου / εξόδου

Αυτές είναι οι εντολές ανάγνωσης (read) και εγγραφής (write) δεδομένων. Συντάσσονται με το όνομα της εντολής και μια λίστα στοιχείων, που στην περίπτωση ανάγνωσης είναι τιμές αριστερής προσπέλασης, ενώ στην περίπτωση εγγραφής είναι εκφράσεις. Και στις δύο περιπτώσεις, η λίστα μπορεί να περιλαμβάνει τη μορφή του *υπονοούμενου βρόχου* που περιγράφεται πιο κάτω. Παραδείγματα εντολών εισόδου/εξόδου είναι:

```
read n,(y(i),i=1,n),x
write "Temperature: ", f,"F, or ", 5/9*(f-32), "C."
write "Squares: ", (x(i)**2, ",", i=1,99), x(100)**2
```

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.

Ο υπονοούμενος βρόχος είναι μια μορφή στοιχείου που *υπονοεί* την επανάληψη της εντολής για τη λίστα στοιχείων που είναι μέσα στις παρενθέσεις.
Μετά τη λίστα στοιχείων και πριν τη δεξιά παρένθεση πρέπει να υπάρχει το πεδίο επανάληψης, το οποίο συντάσσεται με τον τρόπο που συντάσσεται το πεδίο επανάληψης στην εντολή βρόχου που θα δούμε σε λίγο. Η λίστα στοιχείων του υπονοούμενου βρόχου μπορεί να χρησιμοποιεί τη μεταβλητή ελέγχου του βρόχου, η οποία ορίζεται στο πεδίο επανάληψης.
Ο υπονοούμενος βρόχος επιτρέπει φωλιάσματα.

Η εντολή continue

Αυτή είναι μια εικονική εντολή, η εκτέλεση της οποίας δεν έχει κανένα αποτέλεσμα. Μ' άλλα λόγια ο μεταγλωττιστής δεν παράγει κώδικα γι' αυτήν.
Η εντολή continue χρησιμοποιείται για να συνοδεύει μια ετικέτα, όταν στο σημείο που θέλουμε να τοποθετήσουμε την ετικέτα δεν υπάρχει άλλη εντολή. Έτσι για παράδειγμα μπορούμε να υλοποιήσουμε άμεσο άλμα στο τέλος μιας δομημένης εντολής.

Η εντολή return

Αυτή η εντολή επιτρέπεται να εμφανίζεται μόνο μέσα σε υποπρογράμματα, και είναι η μόνη εντολή που επιστρέφει τη ροή του προγράμματος από ένα υποπρόγραμμα στη μονάδα που το κάλεσε, και στο σημείο της κλήσης αυτού.

Η εντολή stop

Αυτή είναι η εντολή τερματισμού της εκτέλεσης ενός προγράμματος. Με την εκτέλεση αυτής σηματοδοτείται το τέλος του προγράμματος, και καμία άλλη εντολή δεν εκτελείται στη συνέχεια.

Η εντολή διακλάδωσης

Η εντολή διακλάδωσης είναι μια δομημένη παραλλαγή εντολής if με παραμέτρους μια λογική έκφραση σε παρενθέσεις και ένα ή δύο σύνολα εντολών με μία τουλάχιστον εντολή το κάθε σύνολο. Κάθε σύνολο ορίζει μια εσωτερική εμβέλεια και μπορεί να έχει δηλώσεις μεταβλητών.

Η εκτέλεση της εντολής διακλάδωσης ξεκινά με την αποτίμηση της λογικής έκφρασης. Αν αυτή έχει τιμή «Αληθής», εκτελούνται οι εντολές του συνόλου που ακολουθεί τη λέξη-κλειδί “then”, και τελειώνει με την αντίστοιχη λέξη-κλειδί “else” ή, αν αυτή δεν υπάρχει στη συγκεκριμένη εντολή, με την αντίστοιχη λέξη-κλειδί “endif”. Εάν η λογική έκφραση έχει τιμή «Ψευδής» και υπάρχει η αντίστοιχη λέξη-κλειδί “else”, τότε εκτελούνται οι εντολές του συνόλου που ακολουθεί αυτή τη λέξη, και τελειώνει με την αντίστοιχη λέξη-κλειδί “endif”, ενώ
αν δε υπάρχει η λέξη-κλειδί “else”, δεν εκτελείται καμία εντολή.

Ένα παράδειγμα εντολής διακλάδωσης είναι το παρακάτω:

```
      if (a(i+1).gt.13.5.and.z(j)) then
            a(i-1) = f(z(j+1),i-1)*2
            if (x.eq.0) return
      else
            integer k,n
            read k,n
            if (n-i) 100,101,102
      100   a(i-1) = a(n+i)
            return
      102   a(i-1) = f(z(n-i),i-1)*k+1
      continue
      endif
```

όπου φαίνεται και ένα παράδειγμα χρήσης της εντολής continue.
Οι λέξεις-κλειδιά “then”, “else” και “endif” αποτελούν τα διαχωριστικά μεταξύ των εσωτερικών εμβελειών της εντολής διακλάδωσης. Η λογική έκφραση αποτιμάται στην εμβέλεια στην οποία βρίσκεται η εντολή.
Η σύνταξη της γλώσσας επιτρέπει φωλιάσματα στις εντολές διακλάδωσης, κι επομένως ο σημασιολογικός αναλυτής πρέπει να ακολουθεί τις φωλιασμένες εμβέλειες από τους συντακτικούς κανόνες της **SimpleFortran**.

Η εντολή βρόχου

Η εντολή βρόχου ή εντολή do είναι μια δομημένη εντολή με παραμέτρους ένα πεδίο επανάληψης και ένα σύνολο εντολών με τουλάχιστον μία εντολή. Το σύνολο αυτό ορίζει μια εσωτερική εμβέλεια και μπορεί να έχει δηλώσεις μεταβλητών.
Το πεδίο επανάληψης περιέχει το όνομα μιας βαθμωτής μεταβλητής, που ονομάζεται μεταβλητή ελέγχου του βρόχου, και δύο ή τρεις αριθμητικές εκφράσεις, που αποτιμώμενες δίνουν:
η πρώτη την αρχική τιμή της μεταβλητής ελέγχου, η δεύτερη την τελική τιμή αυτής, και η τρίτη το βήμα αύξησης της μεταβλητής ελέγχου μεταξύ διαδοχικών επαναλήψεων του βρόχου. Εάν η τρίτη έκφραση απουσιάζει, το βήμα θεωρείται ότι είναι 1.
Η εκτέλεση της εντολής do ξεκινά με αποτίμηση των εκφράσεων του πεδίου επανάληψης, και ανάθεση της αρχικής τιμής στη μεταβλητή ελέγχου. Στη συνέχεια εκτελούνται οι εντολές του συνόλου που ακολουθεί το πεδίο επανάληψης και τελειώνει με τη λέξη-κλειδί “enddo”. Αν δεν υπάρχει παρέμβαση στη ροή του προγράμματος, αυξάνεται κατάλληλα η τιμή της μεταβλητής ελέγχου του βρόχου, και η νέα τιμή συγκρίνεται με την τελική. Εάν είναι μικρότερη από ή ίση με αυτή, η διαδικασία επαναλαμβάνεται από την εκτέλεση των εντολών του συνόλου και κάτω. Διαφορετικά, η εκτέλεση του βρόχου τερματίζεται.
Ας σημειωθεί ότι οι εκφράσεις του πεδίου επανάληψης αποτιμώνται *μόνο* μία φορά, και ακόμα ότι οι εντολές του συνόλου εκτελούνται *τουλάχιστον* μία φορά. Κάθε παρέμβαση στη ροή του προγράμματος που τη μεταφέρει έξω από το εσωτερικό σύνολο εντολών του βρόχου τερματίζει την εκτέλεση αυτού.

Ένα παράδειγμα εντολής βρόχου δίνεται παρακάτω:

```
      do i=1,n
            integer j,k
            a(i*m) = y/2
            k = f(n*m,a,x)
            do j=1,m
                  b(j,i+k) = (m-k)*1.
                  if (j .eq. k) goto 100
            enddo
      100 continue
      enddo
```

όπου για άλλη μια φορά δίνεται κι ένα παράδειγμα χρήσης της εντολής continue.

Η σημασιολογία της εντολής βρόχου 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 τρίτη έκφραση του πεδίου επανάληψης, αν υπάρχει, πρέπει να έχει τιμή μεγαλύτερη από 0.
4. Η μεταβλητή ελέγχου δεν επιτρέπεται να λαμβάνει τιμή μέσα στο εσωτερικό σύνολο εντολών του βρόχου – άρα ούτε να μεταδίδεται κατ' αναφορά σε κάποιο υποπρόγραμμα που καλείται μέσα από το σύνολο αυτό.

Η λέξη-κλειδί "enddo" αποτελεί το κάτω όριο της εσωτερικής εμβέλειας του βρόχου. Οι αριθμητικές εκφράσεις του πεδίου επανάληψης αποτιμώνται στην εμβέλεια στην οποία βρίσκεται η εντολή, ενώ η μεταβλητή ελέγχου πρέπει να είναι ορισμένη σε αυτήν ή εξωτερική της εμβέλεια.
Η σύνταξη της γλώσσας επιτρέπει φωλιάσματα στις εντολές βρόχου, κι επομένως ο σημασιολογικός αναλυτής πρέπει να ακολουθεί τις φωλιασμένες εμβέλειες από τους συντακτικούς
κανόνες της **SimpleFortran**.

Ετικέτες

Μια ετικέτα δείχνει τη θέση μιας εντολής στον κώδικα του προγράμματος. Δεν αντιπροσωπεύει κάτι το εκτελέσιμο, δηλαδή ο μεταγλωττιστής δεν παράγει κώδικα για αυτήν, αλλά χρησιμοποιείται σε συνδυασμό με άλλες εντολές για μεταφορά της ροής του προγράμματος σε αυτήν.

Μια ετικέτα δε δηλώνεται – όπως μία μεταβλητή – στην αρχή μιας εμβέλειας, αλλά ορίζεται τη στιγμή που θα συναντηθεί, δίπλα δηλαδή στην αντίστοιχη εντολή. Είναι μάλιστα πολύ πιθανό να ορίζεται, αφού έχει ήδη χρησιμοποιηθεί σε κάποια προηγούμενη εντολή του προγράμματος. Η ορατότητα μιας ετικέτας είναι ολόκληρη η εμβέλεια στην οποία ορίζεται.
Για τον παραπάνω λόγο, η ολοκλήρωση της ανάλυσης των εντολών που χρησιμοποιούν μια ετικέτα, όπως οι goto και if, σε περίπτωση που αυτή δεν είναι ακόμα ορισμένη, γίνεται μόλις αυτή οριστεί, με την τεχνική του μπαλώματος. Επομένως, οι θέσεις των εντολών αυτών πρέπει να σημειώνονται κατά τη σημασιολογική ανάλυσή τους, για μετέπειτα επάνοδο σε αυτές.

Δήλωση τερματισμού μονάδας

Η λεκτική μονάδα end δεν παριστάνει εκτελέσιμη εντολή. Αν ο μεταγλωττιστής συναντήσει μια δήλωση τερματισμού μονάδας, ολοκληρώνει την ανάλυση αυτής. Ύπαρξη άλλης εντολής της μονάδας μετά τη δήλωση τερματισμού αποτελεί συντακτικό σφάλμα.

Η εκτέλεση μιας μονάδας πρέπει να τερματίζεται με κάποια εντολή return ή εντολή stop. Σε περίπτωση που συναντηθεί δήλωση end, αλλά η εντολή που προηγείται αυτής δεν είναι μια από τις δύο αυτές εντολές, ο μεταγλωττιστής θα προσθέσει μια τέτοια εντολή ως εξής:
Αν η μονάδα είναι η κύρια μονάδα του προγράμματος, προστίθεται η εντολή stop, ενώ αν είναι μονάδα υποπρογράμματος, προστίθεται η εντολή return.

Παραδείγματα κώδικα της γλώσσας

test1.f

```
  integer x(g),z,i,j
  string s,s_str_(10)
  data x/(5.:20.),(4.:-2.),(0.3:0.)/,z/.true./
  common /com1/i,j
  data s_str_/"string1", *"string2", "string3"/
  $ no comment
  y(z) = .not. re(x) .and. .not. x(0x10F,i,g(j)).gt.y .or. z .eq. k**3
  100 call a
  s = "sh"
  im(x(1))=-2.1
  if (b .and. (x .gt. a(i**k))) stop
  return
  do x=1,x(10,g(y+im(x(2)))),3
  real list a
  real i
  data i/.314159e-31/,b/5E2/
  1100 read (x,i=1,x(i)),z
  goto 1000
  cdr(a) = [0.1, 6., -3.2, 0.]
  if (car(a).gt.0B.01.and.b.lt.x(i+2.)) then
    integer z
    read z,a(z)
    goto z,(100,101,102,103,104,105)
  else
    logical l(x)
    data l/.false.,.true.,.true./
    do i=1,N,2
      l(i) = x(i+g(i)).le.a
    enddo
  endif
  call try_me_(x(u),y,l)
  if (y(g(y(z-2)))+a(1,i)) 100,1000,10
  y = .not. x(i)
  1000 continue
  enddo
  end
  subroutine try_me_(integer n,a(n))
  do i=1,n
    if (a(i) .gt. 0) a(i) = a(i) - i
  enddo
  return
end
```

test2.f

```
  common /C1/x,z,i,w
  integer x(10,100,2),z(2),i,j,N
  real a(5), list al logical ll,yy(10) complex k,w(20)
  string s,s_str_(10)
  data x/2000/,z/2,-9/,N/0x3FC70/
  data a/.314159e-31,.2e-30,0O.272/,ll/.false./,yy/.false.,.true.,.true./
  data w/7*(5.:20.),6*(4.:-2.),7*(0.3:0.)/
  data s_str_/"string1", *"string2", "string3"/
  $ no comment
  100 yy(z) = .not. ll .and. .not. x(10,i,g(j)).gt.y(z) .or. z .eq. 0B11
  call aa
  1000 if (b(i,j+k) .and. (x(i,j,g(i+j)) .gt. re(a(k**i)))) stop
  10 continue
  al = new(9.)
  cdddr(al) = [.6e-10,2.1e-17]
```

```
s = "ssh"
do i=1,x(10,g(y(i*j)+j),1),3
integer i1,j1,k1,m(100) complex u
200 read (m(i1),((x(i1,j1,k1),i1=1,x(i,j,2)),j1=1,100),k1=1,N,2*m(i1)),z
if (yy(i)) goto 1000
u = -(1.:-3.2)
if (a.gt.0B.01.and.b(0,a*k**i).and.g(m(i)).lt.x(i,i+2,g(j))) then
  integer z,a(10)
  101 read z,a(z)
  105 if (z .gt. 0 .and. z .le. 6) goto z,(100,101,102,103,104,105)
else
  integer i logical l(1000)
  do i=1,N,2
    1000 l(i) = x(i,g(i),g(j)).le.im(u)
  enddo
endif
102 call try_me_(x(i,int(ll),z),m,car(al))
103 if (y(g(y(z-2)))+a) 100,1000,10
104 yy(m(i*j)) = .not. x(i,j,z) .ge. i
enddo $ 1000 $
a = cdr(a)
re(u) = -2.1
end
subroutine try_me_(integer n,a(n), logical l)
integer i
do i=1,n
  if (l .and. a(i) .gt. 0) a(i) = a(i) - i
enddo
return
end
real function y(integer n)
integer i,j
if (n .gt. 0) then y = n
else
  j = 0
  do i=1,n+1,n/2
    j = j+i
  enddo
  y = j
endif
return
end
integer function g(integer i)
g = i**5
return
end
integer function int(logical l)
if (l) then int = 1 else int = 0 endif
stop
end
subroutine www
write "Hello!"
return
end
subroutine aa
call www
end
integer function a(complex z)
common /C1/x,y,k
integer x(2000),y(3)
complex k(20)
integer i
```

```
  read i
  a = 0
  if (b(i,z)) if (i) 100,200,100
  100 return
  200 write z
  a = 1
  end
  logical function b(integer n, complex c)
  b = .false.
  if (n.eq.0x100) b = .true.
end
```