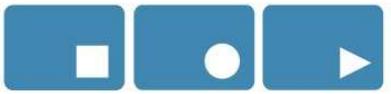
ΥΠΟΥΡΓΕΙΟ ΕΘΝΙΚΗΣ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ ΕΠΕΑΕΚ
ΕΥΡΩΠΑΪΚΗ ΕΝΩΣΗ
ΣΥΓΧΡΗΜΑΤΟΔΟΤΗΣΗ
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
ΕΥΡΩΠΑΪΚΟ ΤΑΜΕΙΟ ΠΕΡΙΦΕΡΕΙΑΚΗΣ ΑΝΑΠΤΥΞΗΣ



ΠΑΙΔΕΙΑ ΜΠΡΟΣΤΑ
2ο Επιχειρησιακό Πρόγραμμα
Εκπαίδευσης και Αρχικής
Επαγγελματικής Κατάρτισης

Πρόγραμμα Αναμόρφωσης Προπτυχιακών Σπουδών

Τμήματος Μαθηματικών

Πανεπιστημιο Πατρων

Μάθημα: Εισαγωγή στην Επιστήμη των Υπολογιστών

4η Εβδομαδα - Τριτη 26/10/2005

Η Γλώσσα Υψηλού Επιπέδου FORTRAN90 (Σύντομη Εισαγωγή)

1. Γενικά

Η **FORTRAN90** είναι η τελευταία τυποποιημένη έκδοση της Γλώσσας Υψηλού Επιπέδου (Γ.Υ.Ε.) FORTRAN (είναι αυτή που χαρακτηρίζεται από το χαρακτηριστικό 90), σχεδιάστηκε έτσι ώστε να μπορέσει να ικανοποιήσει τις ανάγκες της επιστημονικής κοινότητας στην τρίτη χιλιετία για αριθμητικές επεξεργασίες (επιστημονικών, τεχνολογικών και αριθμητικών υπολογισμών).
Βασικά ο στόχος της FORTRAN90 είναι, εκτός των άλλων, η **απλοποίηση** και **επέκταση** των μέχρι σήμερα εκδόσεων της γλώσσας. Έτσι:

α. **Υπάρχει ελευθερία στον τρόπο γραφής** των εντολών που αρχίζουν από τον πρώτο μη κενό χαρακτήρα κάθε γραμμής και **μπορεί να τελειώσουν οπουδήποτε στην γραμμή, ή στον χαρακτήρα (;)**, που σηματοδοτεί το τέλος μιάς εντολής και την ενδεχόμενη απαρχή άλλης, **ή στον χαρακτήρα (!)**, που σηματοδοτεί την απαρχή σχολίου μέχρι το τέλος της γραμμής. **Μια εντολή μπορεί να συνεχίζεται σε επόμενες γραμμές**, αρκεί ο τελευταίος μη κενός χαρακτήρας της γραμμής να είναι ο χαρακτήρας **&** (ampersand).

β. Η πρώτη εντολή κάθε προγράμματος FORTRAN πρέπει να είναι η εντολή (σύστασης):

PROGRAM [miracle]

που αναφέρει το όνομα του προγράμματος που ακολουθεί και που ξεχωρίζει από το PROGRAM (keyword) μ᾽ ένα κενό - και που μπορεί να αποτελείται από 1-31 χαρακτήρες που αρχίζουν από γράμμα και περιέχουν ΜΟΝΟ αλφαριθμητικά στοιχεία και τον χαρακτήρα **underscore (_)**. **Κάθε άλλος χαρακτήρας είναι απαράδεκτος**, **ακόμη και το κενό**· δηλαδή **το όνομα miracle 1 είναι απαράδεκτο**, ενώ **είναι δεκτό το miracle_1**. Επ᾽ ευκαιρίας **το κενό ( )** μπορεί να διαχωρίζει ονόματα (όπως π.χ. το « , ») αριθμούς κλπ., καθ᾽ όσον **«λειτουργεί» ως διαχωριστικό**.

γ. Η τελευταία εντολή κάθε προγράμματος FORTRAN πρέπει να είναι η εντολή (ολοκλήρωσης):

END PROGRAM [miracle]

Σημείωση: Θα ακολουθήσουμε την πρακτική να γράφουμε με κεφαλαία γράμματα τα ονόματα που στο FORTRAN έχουν ειδική σημασία (keywords) και με μικρά γράμματα όλα τα άλλα ονόματα (identifiers), των μεταβλητών.

Τα **μικρά και τα κεφαλαία γράμματα στο FORTRAN είναι ισότιμα** (ένα και το αυτό), απλώς η χρήση τους διευκολύνει εμάς στην γραφή και ανάγνωση προγραμμάτων.

δ. Ένα πρώτο παράδειγμα προγράμματος, γιά την λύση πρωτοβάθμιων εξισώσεων της μορφής $ax+b=0$, μπορεί να είναι το ακόλουθο:

```
PROGRAM solve_first_degree_equations
    READ *, a, b ! Read the coefficients of the equations
    IF (a /= 0) THEN
        x = -b/a ! Υπάρχει ρίζα που είναι η x
        PRINT *, "The root of the given equation is x=", x
    ELSE PRINT *, "No equation, and there is no root!"
    END IF
END PROGRAM solve_first_degree_equations
```

Παρατηρήσεις:

(i) **Οι αγκύλες στις εντολές**, περιέχουν όχι υποχρεωτικό περιεχόμενο· η παρουσία τους είναι **θέμα δικής μας επιλογής**, που διευκολύνει την διαδικασία του προγραμματισμού.

(ii) Ο παραπάνω απλοποιημένος τρόπος εισόδου δεδομένων (Input):
READ `*, a, b, c, d`
και εξόδου αποτελεσμάτων (Output):
PRINT `*, x, y, z`
που ονομάζονται **List-directed input-output**, λειτουργεί με τις ακόλουθες προϋποθέσεις (για την διευκόλυνσή μας):

I. Στην εντολή READ τα a, b, c και d είναι: **MONON ονόματα μεταβλητών**. (Θα επανέλθουμε σύντομα σ᾽ αυτά)

II. Στην εντολή PRINT τα x, y και z μπορεί να είναι: **ονόματα μεταβλητών, σταθερών** ή **εκφράσεων**.

III. Ο χαρακτήρας (*), που ακολουθεί την εντολή, σηματοδοτεί στον compiler ότι αυτή η εντολή I/O είναι στην απλοποιημένη (List-directed) μορφή της (προκατασκευασμένος τρόπος, για διευκόλυνσή μας).

IV. Οι περιφερειακές μονάδες που εμπλέκονται στις απλοποιημένες μονάδες I/O είναι οι τυπικές (Default Units), που είναι συνήθως το **πληκτρολόγιο** και η **οθόνη**.

V. Τα δεδομένα data που δίδονται από το πληκτρολόγιο πρέπει να διαχωρίζονται από (a value separator): $\flat$ **,** / ή το **τέλος της γραμμής**. Ενυπάρχοντα **πολλά κενά μεταξύ τιμών και διαχωριστικών αγνοούνται**, ενώ **δύο διαδοχικά κόμματα θεωρούνται ότι αφήνουν αμετάβλητη την τιμή της αντίστοιχης μεταβλητής**· π.χ. στο παρακάτω πρόγραμμα για τα διαδοχικά δεδομένα της αριστερής πλευράς θα έχουμε τα διαδοχικά αποτελέσματα του άλλου άκρου, που δείχνουν τα βέλη (θα επανέλθουμε όμως σύντομα).

```
PROGRAM test
    i = 0; j = 0; k = 0
    a = 0.0; b = 0.0; c = 0.0
    READ *, i,a,j,b,k,c
    PRINT *, i,a,j,b,k,c
END PROGRAM test
```

		0	0.0	0	0.0	0	0.0	(αρχικές τιμές)
Δεδομένα		**Εκτυπούμενα αποτελέσματα**						
1,2.0,3,4.0,5,6.0	$\rightarrow$	1	2.000	3	4.000	5	6.000	
1 2.0 3 4.0 5 6.0	$\rightarrow$	1	2.000	3	4.000	5	6.000	
1,,, 4.0,, 6.0	$\rightarrow$	1	0.000	0	4.000	0	6.000	
1,,3,, ,6.0	$\rightarrow$	1	0.000	3	0.000	0	6.000	
1,	$\rightarrow$	1	0.000	3	0.000	5	6.000	
3,		$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	
5, 6.0		(i	a	j	b	k	c)	Εκτυπούμενες Μεταβλητές

ε. **τα ειδικά σύμβολα** (Special Characters) της FORTRAN90 έχουν επεκταθεί στα ακόλουθα 22 του πίνακα 1 που ακολουθεί, με την αντίστοιχη αγγλική ονομασία τους (βλέπε σελίδα 143 του βιβλίου σας «Εισαγωγή στην Επιστήμη των Η.Υ.»:

Πίνακας 1

The special characters of the FORTRAN90 language

Character	Name	Character	Name
=	Equal Sign	:	Colon
+	Plus Sign		Blank
−	Minus Sign	!	Exclamation mark
*	Asterisk	"	Quotation mark
/	Slash	%	Percent
(	Left parenthesis	&	Ampersand
)	Right parenthesis	;	Semicolon
,	Comma	<	Less than
.	Decimal point	>	Great than
$	Currency symbol	?	Question mark
'	Apostrophe		

στ. Ένα δεύτερο παράδειγμα προγράμματος είναι το ακόλουθο για την εύρεση τω ριζών των δευτεροβάθμιων εξισώσεων: $Ax^2 + Bx + C = 0$ (N-δεδομένο-το πλήθος των εξισώσεων):

```
 PROGRAM QUADRABEST
READ *, N
DO I = 1, N
        READ *, A, B, C
        IF (A /= 0) THEN
                D = B*B - 4*A*C
                R1 = (-B - D**0.5)/(2*A)
                R2 = (-B + D**0.5)/(2*A)
                PRINT *, "REAL ROOTS", R1, R2, "FOR THE EQUATION:", A, B, C
        ELSE IF (B /= 0) THEN
                R = - C/B
                PRINT *, "LINEAR EQUATION WITH THE ROOT R=", R, "(",A,B,C,")"
        ELSE IF (C /= 0) THEN
                PRINT *, "THERE IS NO EQUATION, AND, OF COURSE, THERE IS NO ROOT", A, B, C
        ELSE
                PRINT *, "THE GIVEN EQUATION IS AN IDENTITY", A, B, C
        ENDIF
ENDDO
END PROGRAM QUADRABEST
```

Εφαρμογή:

$$
\begin{array}{r}
4 \\
1. \quad -5. \quad 6. \\
0. \quad 7. \quad 14. \\
0. \quad 0. \quad 0. \\
0. \quad 0. \quad -6.
\end{array}
$$

Σημείωση: Για την διευκόλυνσή μας, θα μπορούσαμε τα δεδομένα του προγράμματος να είχαμε γράψει σ΄ ένα αρχείο, π.χ. με όνομα datafile.txt, ενώ τα αποτελέσματα θα μπορούσαμε να τα πα•ρουμε σ΄ ένα άλλο αρχείο outfile.txt ενώ εμείς θα έπρεπε για την εκτέλεση του προγράμματος να είχαμε δώσει την εντολή:

```
> QUADRABEST <datafile, >outfile
```

οπότε στο outfile θα είχαμε τα αποτελέσματα.

ζ. Η **γενική μορφή ελέγχου** της γλώσσας πραγματοποιείται με την εντολή «BLOCK IF», που έχει την ακόλουθη δομή:

```
IF (συνθήκη) THEN
      (:block εντολών)
ELSE IF (συνθήκη) THEN
      (:block εντολών)
ELSE IF (συνθήκη) THEN
      ⋮
ELSE
      :(block εντολών)
END IF
```

που την βλέπουμε να υλοποιείται στο δεύτερο παράδειγμα του προγράμματος QUADRABEST, με τρεις ελέγχους. Τον πρώτο:

IF (A /= 0) THEN,

να εξασφαλίζει την ύπαρξη του δευτεροβάθμιου όρου (με $A \neq 0$).

Τον δεύτερο έλεγχο

ELSE IF (B /= 0) THEN

να εξασφαλίζει την παρουσία πρωτοβάθμιας εξίσωσης, όταν δεν υπάρχει δευτεροβάθμιος όρος (με $A = 0$). Τέλος, τον τρίτο έλεγχο:

ELSE IF (C /= 0) THEN

να διαχωρίζει την περίπτωση ταυτότητας (όταν και οι τρεις συντελεστές είναι μηδενικοί), από την αδύνατη ισότητα, όταν υπάρχει η περίπτωση:

$$A = 0,\ B = 0, \quad \text{με} \quad C \neq 0.$$

Φυσικά, **απλοποιημένες μορφές** της γενικής εντολής είναι δυνατές, πιο απλή περίπτωση των οποίων είναι η ακόλουθη

```
IF (συνθήκη) THEN
      (:block εντολών)
END IF
```

με την οποία εκτελείται το block των εντολών **όταν και μόνο όταν** ισχύει η προηγούμενη συνθήκη.
Τέλος, μια άλλη απλοποιημένη μορφή είναι αυτή που περιέχεται στο πρώτο παράδειγμα του προγράμματος solve_first_degree_equations, με την δομή:

```
IF (a /= 0) THEN
      (:block εντολών 1)
ELSE
      :(block εντολών2)
END IF
```

που εξασφαλίζει την εκτέλεση του block εντολών 1, όταν ισχύει η συνθήκη ($a \neq 0$), ή την εκτέλεση του block εντολών 2, όταν δεν ισχύει η συνθήκη ($a = 0$).

η. Η **γενική επαναληπτική εντολή** της γλώσσας είναι η εντολή DO, που η πιο απλή δομή της εμπεριέχεται στο δεύτερο παράδειγμα του προγράμματος QUADRABEST και είναι η:

```
DO I = 1, N
      ( block εντολών)
END DO
```

υπονοεί δε ότι το block των εντολών που ακολουθεί την πρώτη εντολή (DO I=1, N) θα εκτελεσθεί N φορές, με διαφορετική τιμή στον δείκτη (μεταβλητή) I, σύμφωνα **με το πεδίο που ορίσθηκε** (I = 1, N), που σημαίνει ότι την πρώτη φορά το I θα έχει τιμή 1 (**αρχική τιμή**), την δεύτερη φορά το I θα έχει τιμή 2 κλπ. και την τελευταία φορά θα έχει τιμή I=N (**την τελική τιμή**).

Σε περίπτωση που θέλαμε το I να λαμβάνει μόνο τις περιττές τιμές μπορούμε να προσθέσουμε και **το επιθυμητό βήμα**, με την μορφή:
DO I = 1, N, 2
που σημαίνει ότι το block των εντολών εκτελείται για την αρχική τιμή του δείκτη I, που στην συνέχεια αυξάνεται συνεχώς κατά το βήμα (2) και εκτελείται το block των εντολών έως ότου η τιμή του I γίνει μεγαλύτερη του N, οπότε τότε σταματά η επαναληπτική εκτέλεση και εκτελείται η επόμενη εντολή που ακολουθεί την εντολή END DO. Άρα ο δείκτης I θα πάρει τιμές: $1, 3, 5, \ldots, k \leq N$.

θ. Είδη σταθερών (Literal constants of intrinsic type)

Η γλώσσα περιέχει 5 είδη δεδομένων (σταθερών), που χωρίζονται σε δύο κατηγορίες. Η πρώτη κατηγορία περιέχει, τους **τρεις αριθμητικούς τύπους δεδεμένων (Ακέραιοι, Πραγματικοί και Μιγαδικοί αριθμοί)**, που αξιοποιούνται στους αριθμητικούς υπολογισμούς, ενώ η δεύτερη περιλαμβάνει 2 τύπους δεδομένων (χαρακτήρες και Λογικές σταθερές), που χρησιμοποιούνται σε μη αριθμητικές διαδικασίες.

Πιο αναλυτικά:

(i) Οι **Ακέραιοι** (Integer constants) αριθμοί της γλώσσας, είναι όπως οι γνωστοί μας ακέραιοι των μαθηματικών, με μόνο ένα περιορισμό στο μέγεθος, που εξαρτάται από το μέγεθος της λέξης ενός Η.Υ.· πάντως ακέραιους με 7 ψηφία δέχονται όλοι οι Η.Υ., ενώ στο εργαστήριο του τμήματος ο compiler - Fujitsu/Lahey του συστήματος δέχεται ακεραίους με 18 ψηφία (διαθέτει λέξεις των 8 bytes).

(ii) Οι **πραγματικοί** (Real constants) αριθμοί της γλώσσας χαρακτηρίζονται από την παρουσία του δεκαδικού σημείου « . », όπως στον πραγματικό αριθμό 1234.56. Το μέγεθος τους και η ακρίβεια της εξαρτάται από το μέγεθος της λέξεως του Η.Υ.. Πάντως μεγέθη της τάξης $(10^{-38}, 10^{38})$ και ακρίβεια 7 δεκαδικών ψηφίων, εξασφαλίζεται σ᾽ όλους τους υπολογιστές. Τέλος, ο compiler του εργαστηρίου δέχεται πραγματικούς με 33, περίπου, σημαντικά ψηφία και τάξης $(10^{-4931}, 10^{4932})$, αφού διαθέτει λέξεις των 16 bytes.

(iii) Οι **μιγαδικοί** (Complex constants) αριθμοί της γλώσσας χαρακτηρίζονται από τα δύο μέρη τους - πραγματικό και φανταστικό - που διαχωρίζονται από ένα κόμμα, είναι πραγματικοί αριθμοί και περικλείονται μεσα σε δύο παρενθέσεις όπως στον: (-2.31, 6.17), αποθηκεύονται δε σε 2 θέσεις μνήμης.

(iv) Οι **σταθερές τύπου χαρακτήρα** (charcter constants) είναι μη αριθμητικές σταθερές και αποτελούν μιά **πεπερασμένη ακολουθία** (string) χαρακτήρων όλων των ειδών που δέχεται η γλώσσα, περικλειομένων είτε μεταξύ αποστρόφων «'» είτε μεταξύ εισαγωγικών (quotation marks), όπως π.χ. οι περιπτώσεις: `'Tomorrow is Sunday'` και `"Anything you say goes"`. Οι σταθερές τύπου χαρακτήρα, αξιοποιούνται σε επικεφαλίδες ή σε συγγραφή γραπτού κειμένου. **Το σημαντικό είναι ότι οι χρησιμοποιούμενοι χαρακτήρες δεν περιορίζονται μόνο στα σύμβολα της γλώσσας αλλά μπορεί να είναι οποιοσδήποτε γραφικός χαρακτήρας που υποστηρίζει ο επεξεργαστής του Η.Υ.**, εκτός των χαρακτήρων ελέγχου, όπως π.χ. το «newline».

Τέλος, το κενό (*b*) αποτελεί και αυτό ένα χαρακτήρα σημαντικό, ενώ και τα περικλείοντα σύμβολα (’’,’) μπορούν να είναι χαρακτήρες της σταθεράς, αρκεί είτε να είναι ανόμοιοι με τα περικλείοντα είτε να γραφούν **διπλά** ως χαρακτήρες, όπως στις περιπτώσεις:

’She said "Hello" ’ (Ανόμοιοι Χαρακτήρες)
"Isn’’t it a cold day" (Διαπλασιασμός της αποστρόφου)

(v) Οι **λογικές σταθερές** (logical constants) είναι το τελυταίο είδος δεδομένων και μπορούν να πάρουν **μία από τις δύο τιμές** .TRUE. ή .FALSE. (Οι τελείες εμπρός και πίσω είναι απαραίτητες), που μπορούν να πάρουν οι λογικές τιμές (βλέπε και βιβλίο στη σελίδα 154-158)

ι. Μεταβλητές - Ονόματα (στη γλώσσα)

Στα 5 είδη των σταθερών, που αναφέραμε, αντιστοιχούν ισάριθμα 5 είδη μεταβλητών, που ακολουθούν τους ίδιους κανόνες και περιορισμούς με τις σταθερές. Οι μεταβλητές φυσικά έχουν τη δυνατότητα να μεταβάλλουν τιμές στο αυτό πρόγραμμα. Έτσι, μία διάταξη γραμμάτων και αριθμών, καθώς και του χαρακτήρα « _ », που αρχίζει **με γράμμα** μπορεί να παριστά μια μεταβλητή FORTRAN. Π.χ. οι διατάξεις `KOSTAS`, `ANNA`, `TASIA`, `JOHNS` παριστούν μεταβλητές. Το αποδεκτό μέγιστο μήκος διάταξης είναι 31 χαρακτήρες. Ο βασικός λόγος που απαιτείται όπως ο πρώτος χαρακτήρας είναι γράμμα, είναι ότι με τον πρώτο χαρακτήρα γίνεται η **φυσική** διάκριση ακεραίων και πραγματικών μεταβλητών (**κανόνας του ονόματος - name rule**). Ο κανόνας του ονόματος έχει καθολική ισχύ, πλην όμως όχι και απόλυτη, καθ' όσον είναι τροποποιήσιμος· π.χ. με μια εντολή ΤΥΠΟΥ (βλέπε επόμενη παράγραφο.). Αναλυτικότερα, εάν **το πρώτο γράμμα** του ονόματος μιας μεταβλητής είναι ένα από τα I, J, K, L, M, N τότε η μεταβλητή είναι **ακέραια**.

Για κάθε άλλο πρώτο γράμμα η μεταβλητή είναι **πραγματική**. Οι μεταβλητές **διπλής ακρίβειας**, **μιγαδικές** και **λογικές** ορίζονται μόνο με μιά δήλωση ΤΥΠΟΥ. Αυτό συμβαίνει είτε διότι απαιτούνται δύο λέξεις μνήμης (διπλής ακρίβειας, μιγαδικές), είτε ένα μόνο bit μιάς λέξης μνήμης (λογικές).

Τέλος, **οι μεταβλητές μπορούν να έχουν και δείκτες**· αυτοί μπορεί να είναι μόνο ακέραιες σταθερές ή ακέραιες μεταβλητές, ή κάποια γραμμική έκφραση αυτών, που γράφονται παράπλευρα της μεταβλητής και εσωκλείονται μέσα σε παρενθέσεις, χωρίζονται δε (οι δείκτες) μεταξύ τους, όπως πάντα με κόμματα.

Το μέγιστο πλήθος των δεικτών που γίνεται δεκτό εξαρτάται από τον διεκπεραιωτή· οι τρεις δείκτες, πάντως, γίνονται δεκτοί απ' όλους τους διεκπεραιωτές.

Μερικά παραδείγματα μεταβλητών*:

i. ακεραίων: JOHN, MARY, LOSS, JROAD1, ISTREET.

ii. πραγματικών: PROFIT, YEAR74, FIRST6.

iii. με δείκτες: WEEK2(I,JS,20), KAPAI(30), ABC(KLM15).

Τα παρακάτω παραδείγματα μεταβλητών **δεν** είναι αποδεκτά για τους λόγους που αναγράφονται μέσα στις παρενθέσεις:

MATHEMATICS% (υπάρχει μη αποδεκτός χαρακτήρας - το %)

K.I.I (υπάρχουν μη αποδεκτοί χαρακτήρες - οι τελείες)

5SCHOL (αρχίζει από αριθμό)

THE UNIVERSITY (υπάρχει μη αποδεκτός χαρακτήρας - το κενό)

*Στον compiler του εργαστηρίου του τμήματος το μήκος του ονόματος των μεταβλητών μπορεί να είναι μέχρι **240 χαρακτήρες** και να περιέχει σαν πρώτο χαρακτήρα το σύμβολο του «$» για ειδικούς σκοπούς.

ια. Ιστορική αναδρομή και σπουδαιότητα της γλώσσας FORTRAN

Στην ακροτελεύτια αυτή παράγραφο του ενημερωτικού αυτού σημειώματος θα παραθέσουμε λίγα ιστορικά στοιχεία για τη γλώσσα, που θα την εντάξουν μέσα στον «κόσμο» των γλωσσών υψηλού επιπέδου (που απαριθμεί καμιά χιλιάδα μέλη) και θα τονίσουν την σπουδαιότητά της στις σύγχρονες εφαρμογές.

Η FORTRAN οφείλει **την γένεσή της στον J. Backus, πρωτοπόρο της επιστήμης των Ηλεκτρονικών Υπολογιστών**, ο οποίος προς το τέλος του 1953 επρότεινε στην IBM τη σύσταση μιας ερευνητικής ομάδας για την επινόηση ενός πιο αποτελεσματικού τρόπου προγραμματισμού, από τον τότε γνωστό προγραμματισμό σε συμβολική γλώσσα (Assembly-language programming), πράγμα που θα διέδιδε τη χρήση των Η.Υ., αφού πλέον θα ήταν ευκολότερος ο προγραμματισμός τους.

Η IBM δέχθηκε αμέσως την πρόταση και η ομάδα άρχισε χωρίς χρονοτριβή την εργασία της, για τον IBM 704 Η.Υ., με αποτέλεσμα περί τα μέσα του 1954 να έχουν προετοιμασθεί οι προδιαγραφές της γλώσσας, για εσωτερική και μόνο χρήση της IBM, που την διέκριναν η **ευελιξία**, και η **δύναμη**, επρόκειτο δε να ονομασθεί «The IBM Mathematical FORmula TRANslation Sytem, FORTRAN». Η επιτυχία του ήταν σημαντική και το πρώτο εγχειρίδιο αναφοράς της γλώσσας για τους προγραμματιστές εκδόθηκε τον Οκτώβριο του 1956, ενώ οι απαιτήσεις των πελατών της IBM είχαν ως αποτέλεσμα να παραδοθούν διαικπεραιωτές (compilers) της νέας γλώσσας σε πελάτες της IBM τον Απρίλο του 1957. **Αυτό ήταν το FORTRAN I**.

Δώδεκα μήνες αργότερα μιά βελτιωμένη έκδοση της γλώσσας με εμπλουτισμένα διαγνωστικά μηνύματα και έναν αριθμό σημαντικών βελτιώσεων ενεφανίσθηκε. **Αυτή ήταν η FORTRAN II**. Στην επόμενη φάση η ομάδα δημιούργησε FORTRAN compilers για τις άλλες μηχανές της IBM (709, 650, 1620, 7070, κ.λ.π.). Το βήμα που ακολούθησε ήταν η επιτυχία της IBM, να προκαλέσει τις άλλες εταιρείες κατασκευής Η.Υ. να την μιμηθούν και να κατασκευάσουν FORTRAN compilers για τους δικούς τους Η.Υ., με αποτέλεσμα το 1963 να υπάρχουν διαθέσιμοι 50 διαφορετικοί διεκπεραιωτές FORTRAN. **Το πρόβλημα που ανεφύει σ' όλους αυτούς τους διεκπεραιωτές ήταν οι «δυσαρμονίες»** (incompatibilities) που εμφανίστηκαν όχι μόνο μεταξύ των διεκπεραιωτών διαφορετικών εταιρειών αλλά **και μεταξύ των διεκπεραιωτών της αυτής εταιρείας, αλλά για διαφορετικές μηχανές**.

Έτσι, από την πίεση της αγοράς η IBM αναγκάστηκε το 1962 να αναπτύξει και διαθέσει **μια νέα έκδοση της FORTRAN που ξεπέρασε** τις δυσαρμονίες, καταργώντας όλα εκείνα τα χαρακτηριστικά των περσμένων εκδόσεων που σχετίζονταν με την ίδια τη μηχανή (The machine depended features)· αυτό ήταν το FORTRAN IV. Το επόμενο μεγάλο βήμα προήλθε από το **Amerixan National Standards Institute** που το 1962 συνέστησε επιτροπή με στόχο την τυποποίηση της FORTRAN IV. Έτσι, λοιπόν, γεννήθηκε **η πρώτη τυποποιημένη έκδοση της γλώσσας, η ANSI 66**, τον Μάρτιο του 1966.

Με την πάροδο του χρόνου και τις νέες εφαρμογές των Η.Υ. σε όλες σχεδόν τις εκφάνσεις της κοινωνικής ζωής μας αλλά και τον αναγκασμό από τις νέες γλώσσες που εμφανίστηκαν (ALGOL 60 & 68, COBOL, BASIC, PASCAL, PL/I) προκάλεσαν στην FORTRAN την ανάγκη για νέες επεκτάσεις της γλώσσας, με αποτέλεσμα τη σύσταση νέας επιτροπής για την καινούρια τυποποίηση της γλώσσας, της οποίας η έγκριση και αποδοχή έγινε το 1978, **ονομάστηκε δε FORTRAN 77** και αντικατέστησε την πρώτη τυποποίηση της γλώσσας την FORTRAN 66.

Τέλος, η τελευταία τυποπολίηση της γλώσσας, που ξεκίνησε το 1980, είναι η FORTRAN90, που θα έχουμε εμείς ως βάση, για τα εργαστήριά μας. Επίσης, είναι βοηθητικό να τονισθεί ότι κάθε νέα τυποποίημένη επέκταση της γλώσσας **ενσωματώνει καινούριες δυνατότητες** σ΄ αυτήν ή **θεραπεύει ενυπάρχουσες αδυναμίες** (π.χ. δυσχρηστότητα στις διαδικασίες εισόδου δεδομένων ή εξόδου αποτελεσμάτων).

Αναφορικά, τώρα, με τη **σπουδαιότητα** και χρησιμότητα της γλώσσας FORTRAN, αλλά και την ένταξή της στον **κόσμο των γλωσσών**, θα πρέπει να τονισθεί ως **προς τη χρήση σε παγκόσμιο επίπεδο**, μαζί με την COBOL (1960), **είναι σαφώς οι πιο χρησιμοποιούμενες γλώσσες**. Η μεν COBOL για τις εμπορικές εφαρμογές και η **FORTRAN για τις επιστημονικές και τεχνολογικές εφαρμογές**. Μερικά παραδείγματα είναι σαφώς εύγλωττα.

Πρώτον: Το **μεγαλύτερο μέρος της ατράκτου ενός JAMBO JET (Boeing 747)** κατασκευάζεται, από μηχανικά εργαλεία των οποίων οι κινήσεις ελέγχονται με τη βοήθεια προγραμμάτων FORTRAN. Το ίδιο συμβαίνει και με τα σεληνόπλοια της NASA.

Δεύτερον: Τα καλούπια που χρησιμοποιούνται στην μαζική παραγωγή των καροσερί των αυτοκινήτων, κατασκευάζονται με τη βοήθεια μηχανών που ελέγχονται από Η.Υ., με τη βοήθεια προγραμμάτων FORTRAN.

Τρίτον: Τα μοντέλα των πειραμάτων γιά την έρευνα της δομής των ατόμων*, όπως και η ανάλυση των αποτελεσμάτων αυτής γίνονται με τη βοήθεια Η.Υ. διά προγραμμάτων FORTRAN. (Π.χ. στο CERN της Ελβετίας).

Τέταρτον: Η στατική μελέτη των γεφυριών και ουρανοξυστών καθώς και ο σχεδιασμός των γεννητριών ρεύματος (βλέπε ΔΕΗ κ.τ.λ.) συνήθως γίνονται με την βοήθεια προγραμμάτων FORTRAN.

Πέμπτον: Η επικρατούσα γλώσσα εφαρμογών του ακαδημαϊκού χώρου είναι η FORTRAN. Γνωστά πακέτα βιβλιοθηκών, όπως το **SPSS** (για θέματα Στατιστικής), απαραίτητο εφόδιο σ' όλους τους επιστήμονες των Κοινωνικών Επιστημών για την ανάλυση των δειγματοληπτικών ερευνών και την εν γένει έρευνα. Άλλα αξιοσημείωτα πακέτα είναι το **NAG Library** (μία τεράστια και εκτεταμένη συλλογή υποπρογραμμάτων για εφαρμογές Αριθμητικής Αναλύσεως και Υπολογιστικών Μαθηματικών), η βιβλιοθήκη της **IMSL** (αμφότερα τα πακέτα υπάρχουν στο εργαστήριο του τμήματος), κτλ.

*Φυσική Υψηλών Ενεργειών (High Energy Physics) και Φυσική σωματιδίων (Particle Physics)

2. Τελεστές

α. Αριθμητικοί τελεστές - Ενσωματωμένες συναρτήσεις Ο διεκπεραιωτής FORTRAN αναγνωρίζει τις εξής πέντε αριθμητικές πράξεις, που αντιστοιχούν στους αντίστοιχους τελεστές (Arithmetic operators):

Πρόσθεση	$+$	(π.χ., $A + 6.0$)
Αφαίρεση	$-$	(π.χ., $K - 20$)
Πολλαπλασιασμός	$*$	(π.χ., $A * A$)
Διαίρεση	$/$	(π.χ., A/B)
Ανύψωση σε δύναμη	$**$	(π.χ., $A * *2$)

(Οι τελεστές της πρόσθεσης και της αφαίρεσης χρησιμοποιούνται και ως πρόσημα στις σταθερές και μεταβλητές με τη συνήθη μαθηματική έννοια).

Εάν στις παραπάνω αριθμητικές πράξεις προσθέσουμε και τη δυνατότητα χρησιμοποίησης των ενσωματωμένων στη γλώσσα FORTRAN συναρτήσεων (που δίνουν ένα απλό και εύχρηστο τρόπο εύρεσης τιμών των πολυχρήστων μαθηματικών συναρτήσεων), τότε έχουμε όλα τα «δομικά υλικά» για το κτίσιμο των μαθηματικών εκφράσεων.

Μερικές από τις ενσωματωμένες συναρτήσεις βιβλιοθήκης είναι:

ABS(X) για την $|X|$ {π.χ., ABS(X+2.0*A)}

SIN(X) για το $\eta\mu X$ {π.χ., SIN(3.0*X**2 - 1.0)}

COS(X) για το $\sigma\upsilon\nu X$ {π.χ., COS(2.0*3.14159 + 1.0)}

ALOG(X) για το $\ln X$ {π.χ., ALOG(4.5*A - 1.0)}

EXP(X) για το e^X {π.χ., EXP(7.0/A)}

SQRT(X) για την $\sqrt{X}$ {π.χ., SQRT(B*B - 4.0*A*C)}

CBRT(X) για την $\sqrt[3]{X}$

ALOG10(X) για τον δεκαδικό λογάριθμο του X

TAN(X) για την $\epsilon\phi X$

ASIN(X) για το $\tau o\xi\sigma\upsilon\nu X$

ACOS(X) για το $\tau o\xi\eta\mu X$, κλπ.

Για ένα πλήρη κατάλογο των ενσωματωμένων συναρτήσεων, βλέπε στο τέλος του κεφαλαίου

Έτσι, κατά την εκτέλεση του προγράμματος η απλή αναφορά του ονόματος της συνάρτησης και η παράθεση της μεταβλητής της μέσα σε παρενθέσεις, ακριβώς όπως είναι γραμμένες παραπάνω, έχει σαν αποτέλεσμα να τίθεται αντί του ονόματος της συνάρτησης, η τιμή της.

Με τον ίδιο τρόπο, οι μαθηματικές εκφράσεις στο FORTRAN είναι ένας συνδυασμός συμβόλων, πράξεων, συναρτήσεων, και παρενθέσεων για τον υπολογισμό μιάς αριθμητικής τιμής· π.χ. εάν θέλαμε να βρούμε την τιμή του γνωστού μας τριωνύμου, θα γράφαμε την έκφραση FORTRAN:

```
A*X**2 + B*X + C
```

ή, εάν θέλαμε η προηγούμενη τιμή να απομνημονευθεί στη θέση Υ, την

```
Y = A*X**2 + B*X + C.
```

Όμοια για τη διακρίνουσα D του ίδιου τριωνύμου θα είχαμε:

```
D = B*B - 4*A*C
```

οπότε στη μεταβλητή D θα λαμβάναμε τη τιμή της διακρίνουσας.

β. Λογικοί τελεστές

Ο διεκπεραιωτής FORTRAN αναγνωρίζει επίσης και τις εξής πέντε λογικές πράξεις, που αντιστοιχούν στους αντίστοιχους λογικούς τελεστές (Logical operators):

«Λογική» πρόσθεση	.OR.	(Disjunction)
«Λογικός» πολλαπλασιασμός	.AND.	(Conjunction)
Άρνηση	.NOT.	(Negation)
Ισοδυναμία	.EQV.	(Equivelence)
Μη ισοδυναμία	.NEQV.	(Non Equivalence)

με τους ακόλουθους πίνακες αληθείας:

Πίνακας αληθείας των **λογικών** τελεστών

A	B	A.OR.B	A.AND.B	A.EQV.B	A.NEQV.B	.NOT.A
T	T	T	T	T	F	F
T	F	T	F	F	T	F
F	T	T	F	F	T	T
F	F	F	F	T	F	T

γ. Τελεστής χαρακτήρων

Για τον χειρισμό δεδομένων τύπου χαρακτήρα, υπάρχει ένας μόνο τελεστής, ο τελεστής της **παράθεσης** (the concatenation operator): //, ο οποίος παραθέτει το ένα δεδομένο δίπλα στο άλλο και δημιουργεί ένα νέο δεδομένο τύπου χαρακτήρα με τη σύνθεση των δύο.

Π.χ., εάν είχαμε τις σταθερές 'ABC' και 'XYZ', τότε η πράξη:

'ABC' // 'XYZ'

θα έδιδε τη σταθερά:

'ABCXYZ'.

Ενδιαφέρον είναι να τονισθεί η δυνατότητα σύνδεσης μέρους των σταθερών. Π.χ., εάν είχαμε τις 2 σταθερές WORDA Και WORDB μήκους 4 χαρακτήρων και τιμές:

WORDA = 'LOOP', WORDB = 'HOLE',

τότε η πράξη:

WORDA(4:4) // WORDB(2:4)

θα έδιδε τη σταθερά **'POLE'**, ενώ η πράξη:

WORDA // WORDB // 'S'

θα έδιδε τη σταθερά: **'LOOPHOLES'**.

δ. Τελεστές σύγκρισης

Εάν στους παραπάνω τελεστές, επισυνάψουμε τους τελεστές σύγκρισης (Relational Operators), τότε λαμβάνουμε το πλήρες σύνολο τελεστών που χρησιμοποιείται από ένα διεκπεραιωτή FORTRAN.

Οι τελεστές σύγκρισης είναι:

Τελεστής «ίσον με»	**.EQ.**	ή	==
Τελεστής «διάφορο του»	**.NE.**	ή	/=
Τελεστής «μικρότερο του»	**.LT.**	ή	<
Τελεστής «μικρότερο του ή ίσο με»	**.LE.**	ή	<=
Τελεστής «μεγαλύτερο του»	**.GT.**	ή	>
Τελεστής «μεγαλύτερο του ή ίσο με»	**.GE.**	ή	>=

ε. Κανόνες σχηματισμού των παραστάσεων

Ο σχηματισμός των παραστάσεων ακολουθεί ορισμένους ευνόητους κανόνες, μερικούς από τους οποίους είναι είναι χρήσιμο ενδεικτικά να τονίσουμε.

(i) Είναι προφανές ότι οι τελεστές των διμελών πράξεων δεν μπορούν να ακολουθούν ο ένας τον άλλον (φυσικά η χρήση των προσήμων + και - δεν είναι απαγορευτική, πάντως θα πρέπει και πάλι να εμπεριέχονται με τις αντίστοιχες ποσότητες μέσα σε παρενθέσεις). Απλούστερα, **δεν μπορούν δύο σύμβολα πράξεων να τοποθετούνται παράπλευρα**· π.χ., η γραφή 3.0*(-4) είναι επιτρεπτή, ενώ η (-6.0)**-2 δεν είναι.

(ii) Οι παρενθέσεις ομαδοποιούν τις πράξεις και βοηθούν στην σωστή έκφραση της αλληλουχίας των πράξεων που πρόκειται να εκτελεστούν.

(iii) Κατά τον υπολογισμό μιάς αριθμητικής παράστασης, η απόλυτη τιμή κάθε όρου δεν πρέπει να υπερβαίνει τη μέγιστη επιτρεπόμενη τιμή τη σχετική με τον τύπο του όρου (π.χ., ακέραιοι, πραγματικοί, διπλής ακρίβειας, κτλ.).

(iv) Κατά τον υπολογισμό μιάς αριθμητικής παράστασης δεν πρέπει αρνητικοί αριθμοί να υψώνονται σε πραγματική δύναμη.

(ν) Εάν σε κάποια έκφραση υπάρχουν διαφορετικές λειτουργίες - πράξεις, τότε η σειρά εκτέλεσης των υπολογισμών **ελλείψη παρενθέσεων**, γίνεται με την παρακάτω φθίνουσα ιεραρχία (αφού πρώτα γίνει ο υπολογισμόε των εμπεριεχομένων συναρτήσεων) του πίνακα:

Η ιεραρχία των πράξεων σε φθίνουσα δυναμικότητα

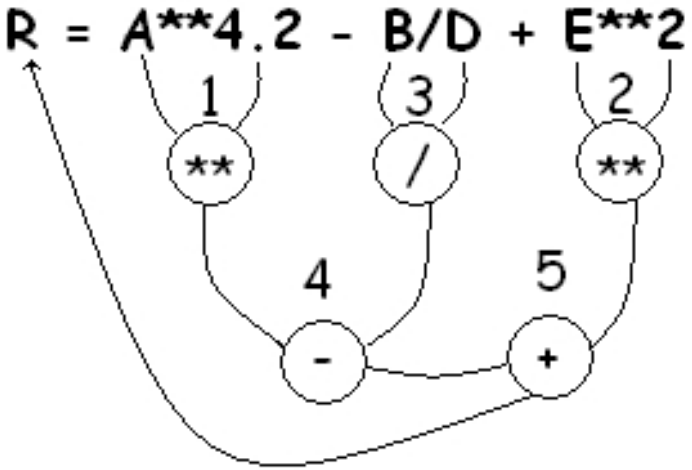
Τελεστής	Πράξη	Ενεργούμενα
**	exponentiation	two numeric
* and /	multiplication and division	two numeric
+ and −	unary adition and subtraction	one numeric
+ and −	binary adition and subtraction	two numeric
//	concatenation	two CHARACTER
.EQ. and == .NE. and /= .LT. and < .LE. and <= .GT. and > .GE. and >=	equal to not equal to less than less than or equal to greater than greater than or equal to	two numeric or two CHARACTER --- two non-COMPLEX numeric or two CHARACTER
.NOT.	logical negation	one LOGICAL
.AND.	logical conjuction	two LOGICAL
.OR.	logical inclusive conjuction	two LOGICAL
.EQV. and .NEQV.	logical equivalence and non-equivalence	two LOGICAL

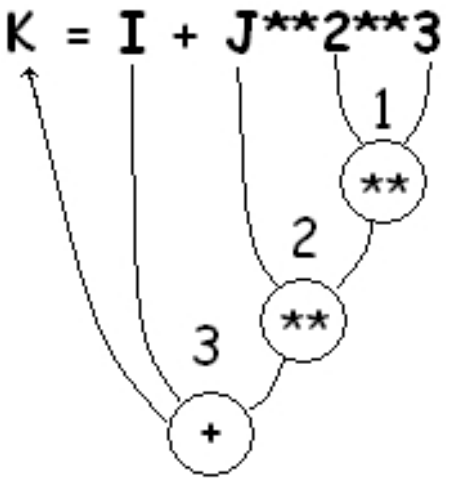
όπου οι ομαδοποιημένες πράξεις σε πλαίσιο είναι **της ιδίας προτεραιότητας** και εκτελεούνται **εξ αριστερών προς τα δεξιά**. Τέλος, **επανειλημένες υψώσεις** σε δύναμη εκτελούνται **εξ δεξιών προς τα αριστερά**.

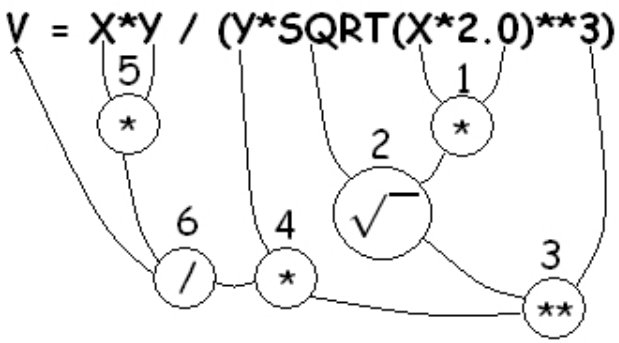
(vi) Εάν μια πράξη γίνεται σε **δεδομένα του αυτού του τύπου** τότε **το αποτέλεσμα** είναι του ίδιου τύπου, ενώ εάν είναι διαφορετικού, τότε το αποτέλεσμα είναι του **«υψηλοτέρου» τύπου**, όπου οι μιγαδικοί είναι «υψηλότεροι» των πραγματικών, που είναι υψηλότερη των ακεραίων. Τέλος, **στη διαίρεση δύο ακεραίων**, το αποτέλεσμα είναι το ακέραιο μέρος του αληθηνού πηλίκου. Π.χ. 7/5 δίδει 1.

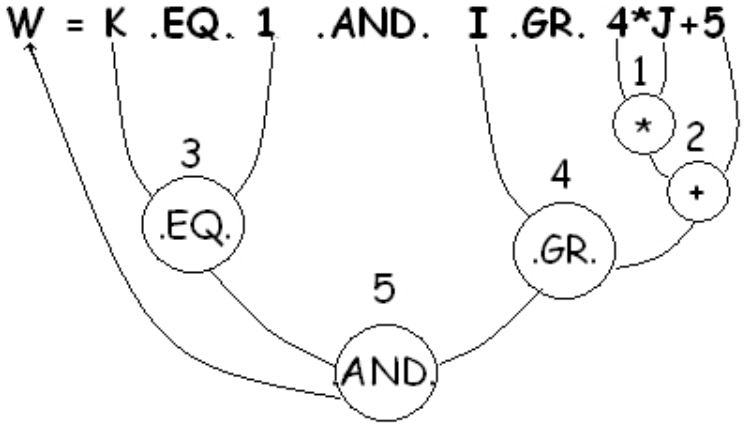
στ. Παραδείγματα

1. Στις παρακάτω μαθηματικές εκφράσεις έχουμε αριθμήσει τη σειρά των πράξεων που θα εκτελεστούν, **με αυξανόμενη αρίθμηση για τις πράξεις που ακολουθούν**. Τα διαγράμματα είναι εύγλωττα και δεν έχουν ανάγκη από επεξηγήσεις.









2. Στο ακόλουθο δεύτερο παράδειγμα τονίζεται η σπουδαιότητα της παρατήρησης 6, προηγούμενα που αφορά τον τύπο του αποτελέσματος μιάς μεικτής αριθμητικής. Αναλυτικότερα, ζητείται να βρεθούν οι αριθμητικές τιμές των παραστάσεων:

i) $L = \frac{1}{3} + \frac{1}{3} + \frac{1}{3}$ $\rightarrow L = 1/3 + 1/3 + 1/3$

ii) $G = \frac{1.0}{4} + \frac{1}{4.0} + \frac{1.0}{4.0} + \frac{1.}{4.}$ $\rightarrow G = 1.0/4 + 1/4.0 + 1.0/4.0 + 1./4.$

iii) $F = 4 * \left(\frac{1}{4}\right) - 5.0 * \left(\frac{1}{5}\right) + 6 * \left(\frac{1.0}{6}\right)$ $\rightarrow F = 4 * (1/4) - 5.0 * (1/5) + 6 * (1.0/6).$

Στην περίπτωση i, τα τρία πηλίκα ακεραίων θα δώσουν ως αποτελέσματα, με οδηγό και πάλι το (6), τρεις ακεραίους που θα είναι τα ακέραια μέρη των πραγματικών τους πηλίκων: 0.333333...3· δηλαδή το 0! Άρα, **η τιμή του L θα ισούται με το μηδέν**.

Στην περίπτωση ii, τα δύο πρώτα πηλίκα, επειδή περιέχουν ένα πραγματικό όρο θα είναι πραγματικοί αριθμοί και ίσοι με το 0.25. Όμοια τα δύο επόμενα πηλίκα είναι πραγματικοί αριθμοί, αφού αμφότεροι οι όροι τους είναι πραγματικοί. **Άρα το G θα έχει τιμή ίση με την μονάδα (0.25 + 0.25 + 0.25 + 0.25)**.

Στην περίπτωση iii, θα εκτελεστούν πρώτα οι πράξεις μέσα στις παρενθέσεις, που θα δώσουν αντίστοιχα 0, 0 και 0.16666...6· τα δε γινόμενα θα είναι, κατά συνέπεια αντίστοιχα ίσα με: 0, 0 και 1.0· πράγμα που θα κάνει **την τιμή της F = 0 - 0 + 1.0 = 1**.

3. Προτάσεις υποπρογραμμάτων

α. Γενικά

Είναι κοινή πείρα των εφαρμογών, ότι πολλές φορές, ορισμένοι υπολογισμοί επαναλαμβάνονται οι ίδιοι στα διάφορα προβλήματα (π.χ., πολλές φορές επιζητείται η εύρεση των ριζών μιας δευτεροβάθμιας εξίσωσης), ή ακόμα, και στα διάφορα μέρη του αυτού προγράμματος (π.χ., η εύρεση της τετραγωνικής ρίζας διαφόρων παραστάσεων). Προφανώς δε, θα ήταν πολύ βοηθητικό, εάν μπορούσε κανείς να δημιουργήσει μια διαδικασία, η οποία να ενεργοποιεί αυτούς ακριβώς τους πολλαπλά απαιτούμενους υπολογισμούς, ενώ, συγχρόνως, θα έχει ένα απλό τρόπο κατάλληλης επίκλησής της, όπου αυτό χρειάζεται, μεταβάλλοντας φυσικά εκάστοτε τις περιεχόμενες μεταβλητές (παραμέτρους).

Άλλωστε, η έννοια των ενσωματωμένων στην γλώσσα συναρτήσεων, αυτή ακριβώς την ανάγκη εξυπηρετούν (με απλή αναγραφή του ονόματος της σχετικής συνάρτησης, π.χ., του EXP(X), ενεργοποιείται η σειρά εκείνη των υπολογισμών με τους οποίους επιτυγχάνεται η εύρεση της τιμής της συνάρτησης e^x για την τρέχουσα τιμή της μεταβλητής - παραμέτρου - X). Το σύστημα FORTRAN δίνει την δυνατότητα γενίκευσης της ιδέας αυτής με επέκταση της σε οιαδήποτε περίπτωση με την βοήθεια της έννοιας των υποπρογραμμάτων.

β. Διάκριση υποπρογραμμάτων

Οι προτάσεις υποπρογραμμάτων διακρίνονται σε δύο κατηγορίες:

- Στις συναρτήσεις (Functions),
- Στις υπορουτίνες (Subroutines).

Αμφότερες ορίζονται με ένα όνομα, μήκους 1 έως 6 αλφαριθμητικών χαρακτήρων που αρχίζουν με ένα γράμμα, που ακολουθείται από ένα σύνολο παραμέτρων (Arguments), που χωρίζονται με κόμματα και περικλείονται μέσα σε παρενθέσεις· π,χ., EQUAT2 (A,B,C).

Η βασική διαφορά μεταξύ συναρτήσεων και υπορουτινών έγκειται στα εξής 2 σημεία. Πρώτον, **το όνομα της συνάρτησης συνδέεται με μία αριθμητική τιμή**, δίκη μεταβλητής (την τιμή της συνάρτησης), που ακολουθεί τον κανόνα του ονόματος, ενώ **το όνομα της υπορουτίνας δεν συνδέεται με καμία τιμή**. Δεύτερον, η κλήση της συνάρτησης γίνεται με απλή αναγραφή, του ονόματος της και με παράθεση των παραμέτρων της*, ενώ οι υπορουτίνες απαιτούν την ειδική κλήση CALL· π.χ., εάν στο προηγούμενο παράδειγμα θεωρήσουμε την EQUAT2 (A,B,C) σαν συνάρτηση και θέλουμε την τιμή της για A=1.2, B=100.0, C=0.1 στην θέση ROOT, τότε θα πρέπει να γράφουμε:

*Όπως ακριβώς οι γνωστές μας συναρτήσεις SIN(X), EXP(X), SQRT(X) κλπ. Βλέπε §2α.

$$ROOT = EQUAT2(1, 2, 100, 0, 0, 1)$$

Εάν τώρα, η EQUAT3 (A,B,C) έχει ορισθεί ως υπορουτίνα, τότε γενικά θα έπρεπε μία από τις παραμέτρους να δίνει το εξαγόμενο αποτέλεσμα, έστω αυτή η A, και συνεπώς εάν θέλουμε την τιμή της A στην θέση ROOT για τις τιμές του B=50.1 και C=0.1, θα πρέπει να γράφουμε:

$$CALL\ EOUAT3(A, 50.1, 0.1)$$
$$ROOT = A$$

γ. Διάκριση συναρτήσεων

Οι συναρτήσεις γενικά διακρίνονται στις εξής κατηγορίες:

i. Εσωτερικές συναρτήσεις (Built in and library functions).

ii. Συναρτήσεις - προτάσεις (Statement functions).

iii. Συναρτήσεις υποπρογράμματα (Function subprograms).

(i) Οι **εσωτερικές συναρτήσεις** περιλαμβάνουν τις ενσωματωμένες στην γλώσσα συναρτήσεις (ανοικτές) καθώς και τις συναρτήσεις της βιβλιοθήκης (κλειστές). Η χρήση τους είναι απλούστατη· αρκεί η αναγραφή του ονόματος τους και η παράθεση, των απαιτούμενων παραμέτρων μέσα σε παρενθέσεις.

Ο διεκπεραιωτής είτε αντικαθιστά την συνάρτηση με ένα πλήθος εντολών σε γλώσσα μηχανής (ανοικτές συναρτήσεις), είτε καλεί το σχετικό υποπρόγ ράμμα από την βιβλιοθήκη (κλειστές συναρτήσεις). Το τελικό αποτέλεσμα για τον προγραμματιστή είναι να λαμβάνει, στη θέση της συνάρτησης, την τιμή της. Π.χ., εάν είχαμε τις εντολές 5, 6, τότε η τιμή της μεταβλητής A μετά την εκτέλεση της εντολής 5 θα ήταν ίση με την μονάδα, ενώ η τιμή της

```
5         A = COS(X)**2 + SIN(X)**2
6         B =         SIN(0.0)
```

μεταβλητής B, μετά την εκτέλεση της εντολής 6, θα ήταν ίση με το μηδέν.

(ii) Οι **συνάρτησεις - προτάσεις**, αποτελούνται από μία μόνο εντολή, η οποία τίθεται στην αρχή του προγράμματος μπροστά από κάθε άλλη εκτελέσιμη εντολή· κατά τα λοιπά, η χρήση τους είναι όπως ακριβώς και των ενσωματωμένων συναρτήσεων.

Η γενική μορφή ορισμού των είναι:

Όνομα συνάρτησης (τυπικές παράμετροι) = Αριθμητική έκφραση

Π.χ., EQUAT4 (A, B, C, X) = A*X**2 + B*X + C

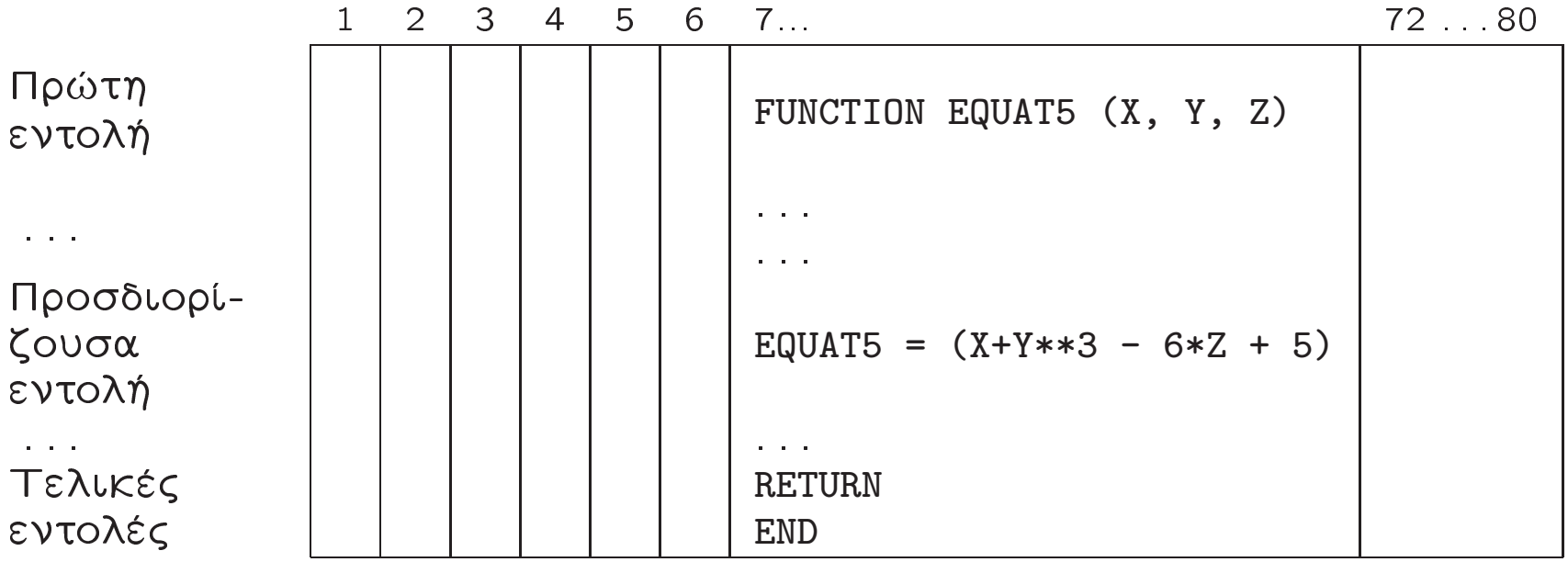
για τον υπολογισμό του γνωστού μας τριωνύμου, όπου στην θέση των τυπικών παραμέτρων είναι μία οιαδήποτε μεταβλητή χωρίς δείκτες, ενώ στο δεξιό μέλος της αριθμητικής έκφρασης είναι δυνατό να περιέχονται και μεταβλητές με δείκτες.

(iii) Οι συναρτήσεις υποπρογράμματα είναι συναρτήσεις, που δεν μπορούν να ορισθούν με μία μόνο εντολή, αλλά απαιτούν περισσότερες, ενώ συγχρόνως η χρήση τους δεν είναι τόσο εκτενής για να συμπεριληφθούν στις συναρτήσεις της βιβλιοθήκης της γλώσσας. Η διαφορά τους από τις προηγούμενες συνάρτησεις - προτάσεις, είναι, ότι **διεκπεραιώνονται ανεξάρτητα** από το πρόγραμμα που τις καλεί και στη συνέχεια ενσωματώνονται σ' αυτό. Π.χ., εάν είχαμε το πρόβλημα της εύρεσης μιας πραγματικής ρίζας της τριτοβάθμιας εξίσωσης, τότε προφανώς μία μόνο εντολή είναι ανεπαρκής για την κάλυψη όλων των περιπτώσεων· παράλληλα, η ανάγκη επίλυσης τριτοβάθμιων εξισώσεων δεν είναι τόσο συχνή για να προσαρτηθεί η συνάρτηση στον πίνακα της βιβλιοθήκης της γλώσσας.

Η οργάνωση των εντολών της συνάρτησης - υποπρόγράμμα γίνεται ως εξής. Κατ' αρχή, έχει ως πρώτη εντολή, την εντολή ορισμού της συνάρτησης, που αποτελείται από τη λέξη FUNCTION, το όνομα της συνάρτησης και τις μεταβλητές της μέσα σε δύο παρενθέσεις, όπως φαίνεται στο σχήμα 7.

Ακολουθεί μετά το σώμα των εντολών της συνάρτησης, που συμπληρώνεται με τις εξής δύο τελικές εντολές, RETURN, END. Δηλαδή, για την συνάρτηση EQUAT5 (Q,U,2) θα έχουμε την διάταξη του σχήματος 7.

Σχήμα 7. Δομή συνάρτησης υποπρόγραμμα



Σημειώστε, πρώτο την διαφορά στην εντολή ορισμού, όπου είναι απαραίτητη η προσάρτηση της λέξης FUNCTION, πράγμα το οποίο δεν γίνεται στις συνάρτησεις - προτάσεις.

Δεύτερο την ύπαρξη μιας τουλάχιστον εντολής RETURN, με την οποία γίνεται η επαναφορά της ροής του προγράμματος στο σημείο κλήσης της συνάρτησης (από το κύριο πρόγραμμα). Τέλος, πρέπει να σημειωθεί, ότι μέσα στο σώμα της συνάρτησης πρέπει απαραίτητα να υπάρχει τουλάχιστον μία εντολή που να προσδιορίζει την συνάρτηση. Π.χ., στην περίπτωση της EOUAT5, πρέπει στο σώμα της συνάρτησης να υπάρχει μία αριθμητική εντολή της μορφής:

$$E0UAT5 = (X + Y) * *3 - 6 * B + 5$$

με την οποία το όνομα της συνάρτησης λαμβάνει την τιμή της.

δ. Υπορουτίνες (Subroutines)

Ο ορισμός υπορουτινών και η οργάνωση των εντολών, γίνεται όπως ακριβώς και των συναρτήσεων υποπρογραμμάτων, Η διαφορά τους έγκειται, όπως αναφέρθηκε παραπάνω, πρώτα στον τρόπο κλήσης τους, και μετά στην μη σύνδεση αριθμητικής τιμής με το όνομα της υπορουτίνας (άρα δεν απαιτείται εμφάνιση της προσδιορίζουσας εντολής με το όνομα της υπορουτίνας, μέσα στο σώμα της).

ε. Η εντολή EXTERNAL

Όπως είπαμε στην §3, το όνομα της συνάρτησης, δίκη μεταβλητής, συνδέεται με μια τιμή· συνεπώς, θα μπορούσε να χρησιμοποιηθεί ως παράμετρος ενός άλλου υποπρογράμματος, μόνο που σε μια τέτοια περίπτωση θα πρέπει να ενημερωθεί ο διεκπεραιωτής με μια εντολή EXTERNAL, η γενική μορφή της οποίας είναι:

$$EXTERNAL\ Q,\ U,\ Z,\ EOUAT4,\ \ldots$$

Με την παραπάνω εντολή, ο διεκπεραιωτής πληροφορείται ότι τα ονόματα των συναρτήσεων X, Y, Z, EOUAT4, ... πιθανόν να παρουσιαστούν ως παράμετροι υποπρογραμμάτων.

4. Παραδείγματα

α. Χρήση συνάρτησης και υπορουτίνας

(i) Παρακάτω δίνουμε μια συνάρτηση SUMQUA για τον υπολογισμό των αθροισμάτων των τετραγώνων των πρώτων K (≤ 999) συντεταγμένων ενός διανύσματος A(999), και μια υπορουτίνα SORT για την διάταξη ενός πλήθους N αριθμών: $X_1, X_2, \ldots, X_\nu$ κατά τάξη αυξανόμενου μεγέθους κλπ.

```
 1 2 3 4 5 6 7 ...                                                  73 ...80
-------------------------------------------------------------------------
          FUNCTION SUMQUA (A, K)
          DIMENSION A(999)
          SUM = 0.0
          DO 20 1 = 1, K
     20   SUM=SUM+A(I)*A(I)
 ⟶       SUMQUA=SUM
          RETURN
          END
          . . . . . .
```

```
      SUBROUTINE SORT(X,N)
      DIMENSION X(1)
      N1=N-1
      DO 20 I = 1, N1
      J = I + 1
      DO 5 K = J, N
      IF(X(I) .LE. X(K))
      GO TO 5
      HELP = X(I)
      X(I) = X(K)
      X(K) = HELP
    5 CONTINUE
   20 CONTINUE
      RETURN
      END
```

Το κύριο πρόγραμμα διαβάζει τις K(≤ 999) συντεταγμένες ενός διανύσματος A, και εκτυπώνει αφ' ενός μεν το άθροισμα των τετραγώνων τους, που βρίσκει με την χρήση της συνάρτησης SUMQUA, αφ' ετέρου δε τις συντεταγμένες κατά αύξουσα σειρά μεγέθους:

```
1 2 3 4 5 6 7 ...                                                    73 ...80
-------------------------------------------------------------------------------
      PROGRAM SORTING
      DIMENSION A(999)
      READ 1, K, (A(I), I = 1, K)
    1 FORMAT (I5/(10F8.0))
  ⟶   RESULT = SUMQUA(A, K)
 //⟶   CALL SORT(A, K)
      PRINT 2, RESULT
    2 FORMAT (29H SUM OF SQUARES OF COMPONENTS, F20.2///)
      PRINT 4, (I, A(I), I = 1, K)
    4 FORMAT (I10, F10.2/)
      STOP
      END SORTING
```

Σημειώστε στο σώμα της συνάρτησης SUMQUA την προσδιορίζουσα εντολή (φέρει βέλος), την παράλειψή της στην SORT, όπως επίσης τον τρόπο επίκλησης της SUMQUA με απλή αναφορά του ονόματος της στο κύριο πρόγραμμα (φέρει βέλος), ενώ η SORT απαιτεί το CALL (φέρει διαγραμμισμένο βέλος).

β. Ο αλγόριθμος του Gauss για την ημερομηνία του Πάσχα

Στην συνέχεια δίνουμε μια θαυμάσια εφαρμογή χρήσης συνάρτησης μιας εντολής, στο πρόγραμμα του υπολογισμού της ημερομηνίας του Πάσχα, για τα έτη 1985-2000 με την βοήθεια του τύπου του Gauss. Εάν Ε παριστά το έτος, το δε σύμβολο $[I]_J$ παριστά το υπόλοιπο της διαίρεσης του ακέραιου I δια του ακέραιου J, τότε η ημερομηνία του Πάσχα είναι η 3η Απριλίου αυξημένη κατά το πλήθος των ημερών που δίνονται από την ποσότητα:

$$
\begin{array}{lcrcll}
 & : & \Pi & = & D+F & (1)\\
\text{όπου} & : & D & = & [19A+16]_{30}, & (2)\\
 & : & F & = & [2B+4C+6D]_7, & (3)\\
\text{με} & : & A & = & [E]_{19},\ B=[E]_4,\ C=[E]_7. & (4)
\end{array}
$$

Για την κατασκευή του προγράμματος θα πρέπει να παρατηρηθεί ότι το πλήθος των ετών για τα οποία θα ζητήσουμε την ημερομηνία του Πάσχα δεν παρουσιάζουν δυσκολία, καθόσον η χρήση της εντολής DO μπορεί να ικανοποιήσει οποιαδήποτε ανάγκη μας. Ένα δεύτερο σημείο που είναι βοηθητικό, είναι η ιδιότητα των τύπων (2), (3) και (4) να εκφράζονται με την βοήθεια της διαδικασίας του υπόλοιπου της διαίρεσης. Συνεπώς, ενδείκνυται η χρήση μιας συνάρτησης μιας εντολής που θα παρέχει αυτό το υπόλοιπο. Έτσι, εάν καλέσουμε με:

$$MOD\ (I, J)$$

την συνάρτηση $[I]_J$ το ζητούμενο πρόγραμμα θα είναι το ακόλουθο (μέσα στο ένθετο παρουσιάζονται τα αποτελέσματα):

```
      PROGRAM EASTER
      INTEGER E, A, B, C, D, F, P
      MOD(I,J) = I - (I/J)*J
      WRITE (6,1)
      FORMAT (1H1, 53X, 'UNIVERSITY OF PATRAS'/ &
                  53X, 'DEPARTMENT OF MATHEMATICS'/ &
                  53X, 13(2H*-)/ &
                  58X, 'DATES OF EASTER'/58X,15(1H-)/ &
                  58X, 'YEARS', 5X, 'EASTER'/ &
                  58X, 5(1H-), 5X, 6(1H-)/ )
      DO 50 E = 1985, 2000
      A = MOD (E, 19)
      B = MOD (E, 4)
      C = MOD (E, 7)
      D = MOD (19*A + 16, 30)
```

```
      F = MOD (2*B + 4*C + 6*D, 7)
      P = 3 + D + F
      IF (P.GT.30) GO TO 10
      WRITE (6, 5) E, P
 5    FORMAT (58X, I4, 5X, I2,' APRIL')
      GO TO 50
10    P = P - 30
      WRITE (6, 15) E, P
15    FORMAT (58X, I4, 5X, I2, ' MAY')
50    CONTINUE
      END EASTER
```

UNIVERSITY OF PATRAS
DEPARTMENT OF MATHEMATICS

DATES OF EASTER

YEARS	EASTER	
1985	14	APRIL
1986	4	MAY
1987	19	APRIL
1988	10	APRIL
1989	30	APRIL
1990	15	APRIL
1991	7	APRIL
1992	26	APRIL
1993	18	APRIL
1994	1	MAY
1995	23	APRIL
1996	14	APRIL
1997	27	APRIL
1998	19	APRIL
1999	11	APRIL
2000	30	APRIL

γ, Η λύση της πρωτοβάθμιας εξίσωσης με υπορουτίνα

Το παράδειγμα αυτό δίνει μια υπορουτίνα - που υλοποιεί την επίλυση μιας πρωτοβάθμιας εξίσωσης - με την εξής περιγραφή:

$$SUBROUTINE\ EOUAT1\ (A, B, X, K)$$

όπου:

Α και Β	οι συντελεστές της εξίσωσης $Ax + B = 0$
Χ	η λύση της εξίσωσης, εφ' όσον υπάρχει
Κ	ένας δείκτης πληροφοριακός με τιμές:
	1 όταν υπάρχει λύση
	2 όταν η εξίσωση είναι αδύνατη
	3 όταν η εξίσωση καθίσταται ταυτότητα

Η υπορουτίνα είναι η ακόλουθη:

```
      SUBROUTINE EQUAT1 (A, B, X, K)
      IF (A == 0) GO TO 15
      K = 1
      Q = -B/A
      RETURN
15    IF (B == 0) GO TO 30
      K = 2
      RETURN
30    K = 3
      RETURN
      END
```

Το ακόλουθο κύριο πρόγραμμα καλεί την EQUAT1 και εκτυπώνει την ρίζα με την μορφή που δίνει η παρακάτω επικεφαλίδα:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
			U	N	I	V	E	R	S	I	T	Y		O	F		P	A	T	R	A	S								
	S	C	H	O	O	L		F	O	R		P	H	Y	S	I	C	A	L		S	C	I	E	N	C	E	S		
		S	O	L	U	T	I	O	N		O	F		F	I	R	S	T		D	E	G	R	E	E					
					E	Q	U	A	T	I	O	N	S		A	X	+	B	=	0										
	-	*	-	*	-	*	-	*	-	*	-	*	-	*	-	*	-	*	-	*	-	*	-	*	-	*	-	*	-	*	-	*
		A	=												B	=												X	=	

Φυσικά σε περίπτωση ταυτότητας ή αδυναμίας το πρόγραμμα εκτυπώνει κατάλληλο μήνυμα.

```
      PROGRAM FDEquations
     READ 5, A, B
  5  FORMAT (2F10.2)
     CALL EQUAT1 (A, B, X, K)
     GO TO (5, 20, 30), K
  5  WRITE (6,10) A, B, X
 10  FORMAT (3X, 2OHUNIVERSITY OF PATRAS/ &
```

```
                1X, 28HSCHOOL FOR PHYSICAL SCIENCES, &
               2X, 24HSOLUTION OF FIRST DEGREE, &
               5X, 16HEQUATIONS AX+B=0/ &
               1X, 15(2H-*)/2X, 2HA=, E10.3, 1X, &
     62HB=, E10.3, 1Q, 2HQ=, E10.3)
     STOP 1
20   WRITE (6,25) A, B
25   FORMAT (6H ERROR, 3x, 2F10.2)
     STOP 2
30   WRITE (6,35) A, B
35   FORMAT (9H IDENTITY, 2F10.2)
     STOP 3
     END FDEquations
```

δ. Η λύση της δευτεροβάθμιας εξίσωσης με υπορουτίνα

Με βάση το πρόγραμμα της §1στ δίνουμε την υπορουτίνα EQUAT2 για την επίλυση της δευτεροβάθμιας εξίσωσης $Ax^2 + Bx + C = 0$ με την εξής περιγραφή:

$$SUBROUTINE\ EQUAT2\ (A, B, C, R1, R2, Y1, Y2, K)$$

	A, B, C	οι συντελεστές της εξίσωσης
	R1, R2	οι πραγματικές ρίζες, ή τα πραγματικά μέρη των μιγαδικών ριζών (X1 θα παριστά την ρίζα σε περίπτωση εκφυλισμού σε πρωτοβάθμια εξίσωση)
	Y1, Y2	τα φανταστικά μέρη των μιγαδικών ριζών
όπου:	K	δείκτης πληροφοριακός με τιμές: 1 στην περίπτωση εκφυλισμού σε πρωτοβάθμια 2 στην περίπτωση 2 πραγματικών ριζών 3 στην περίπτωση 2 μιγαδικών ριζών 4 στην περίπτωση 1 διπλής ρίζας 5 στην περίπτωση αδυναμίας 6 στην περίπτωση ταυτότητας

```
      SUBROUTINE EQUAT2(A, B, C, R1, R2, Y1, Y2, K)
      IF(A .EQ. 0) GO TO 30
      D = B*B - 4*A*C
      IF(D) 10, 15, 20
10    R1=-B/(A+A)
      R2 = R1
      Y1 = SQRT(-D)/(A+A)
      Y2 = -Y1
      K = 3
      RETURN
15    R1 = -B/(A+A)
      Y1 = 0
      Y2 = 0
      R2 = R1
      K = 4
      RETURN
```

```
 20   D1 = SQRT(D)
      R1 = (-B + D1) /(A + A)
      R2 = (-B - D1)/( A + A)
      Y1 = 0
      Y2 = 0
      K = 2
      RETURN
30    IF(B .EQ. 0) GO TO 40
      R1 = -C/B
      K = 1
      RETURN
40    IF (C .EQ. 0) GO TO 50
      K = 5
      RETURN
50    K = 6
      RETURN
      END
```

Το κύριο πρόγραμμα που ακολουθεί εκτυπώνει τα αποτελέσματα όλων των έξη δυνατών περιπτώσεων με την μορφή του παρακάτω πίνακα 10:

```
      PROGRAM SDEquations
 1    WRITE (6.1)
      FORMAT (1H1, 45X, 'UNIVERSITY OF PATRAS'/ &
      43X, 'DEPARTMENT OF MATHEMATICS'/42X, 29(1H*)/ &
      42X, 'SOLVING QUADRATIC EQUATIONS'/42X, &
      29(1H-)/26X, 'DATA', 24X, 'ROOTS', 26X, 'COMMENTS'/ &
      26X, 4(1H-), 24X, 5(1H-), 26X, 8(1H-)/14X, 'S/N', 3X, 'A', &
      4X, 'B', 4X, 'C', 4X, 'X1', 6X, 'Y1', 6X, 'X2', 6X, 'Y2')
      DO 102 I = 1, 6
      READ (5, 19) A, B, C
19    FORMAT (3F5.2)
      CALL EQUAT2 (A, B, C, R1, R2, Y1, Y2, K)
      GO TO (11, 21, 31, 61, 41, 51), K
```

```
 11   WRITE (6, 12) I, A, B, C, R1
12   FORMAT (14X, I2, F5.2, F5.2, F5.2, F7.3, 4X, '-', &
     7X, '-', 7X, '-', 3X, 'ONE REAL ROOT')
     GO TO 102
21   WRITE (6, 22) I, A, B, C, R1, R2
22   FORMAT (14X, I2, F5.2, F5.2, F5.2, F7.3, 4X, '-', &
     2X, F7.3, 4X, '-', 3X, 'TWO REAL ROOTS')
     GO TO 102
31   WRITE (6, 32) I, A, B, C, R1, Y1, R2, Y2
32   FORMAT (14X, I2, F5.2, F5.2, F5.2, F7.3, F7.3, &
     F7.3, F7.3, 1X, 'TWO COMPLEX ROOTS')
     GO TO 102
41   WRITE (6, 42) I, A, B, C
42   FORMAT (14X, I2, F5.2, F5.2, F5.2, 11X, '-', 7X, '-', &
     7X, '-', 7X, '-', 3X, 'IMPOSSIBLE')
     GO TO 102
```

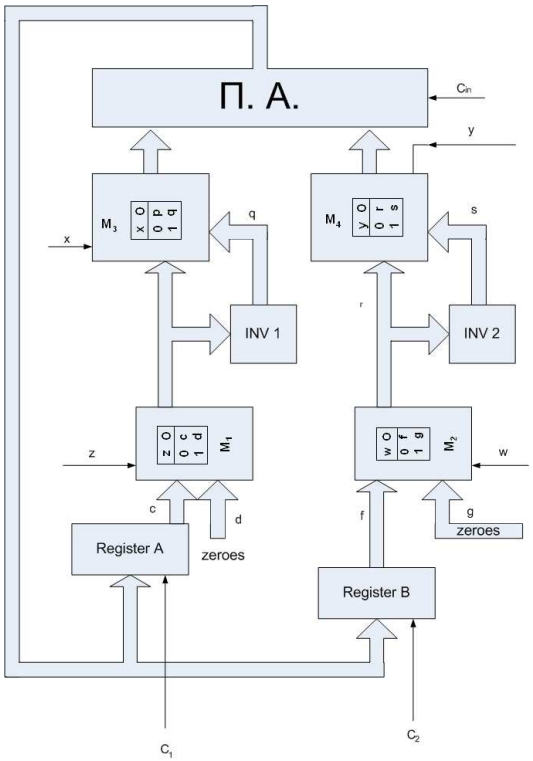
```
 51   WRITE (6, 52) I, A, B, C
 52   FORMAT (14X, I2, F5.2, F5.2, F5.2, 11X, '-', 7X, '-', &
      7X, '-', 7X, '-', 3X, 'IDENTITY')
      GO TO 102
 61   WRITE (6, 62) I, A, B, C, R1
 62   FORMAT (14X, I2, F5.2, F5.2, F5.2, F7.3, 4X, '-', &
      7X, '-', 7X, '-', 3X, 'ONE DOUBLE ROOT')
      GO TO 102
102   CONTINUE
      END SDEquations
```

Πίνακας 10

UNIVERSITY OF PATRAS
DEPARTMENT OF MATHEMATICS
****
SOLVING QUADRATIC EQUATIONS

	DATA			ROOTS				COMMENTS
S/N	A	B	C	R1	Y1	R2	Y2	
1	5.00	8.00	3.00	-.600	-	-1.000	-	TWO REAL ROOTS
2	5.00	7.00	9.00	-.700	1.145	-.700	-1.145	TWO COMPLEX ROOTS
3	1.00	.00	.00	.000	-	-	-	ONE DOUBLE ROOT
4	0.00	9.00	9.00	4.500	-	-	-	ONE REAL ROOT
5	0.00	5.00	5.00	-	-	-	-	IMPOSSIBLE
6	0.00	0.00	0.00	-	-	-	-	IDENTITY

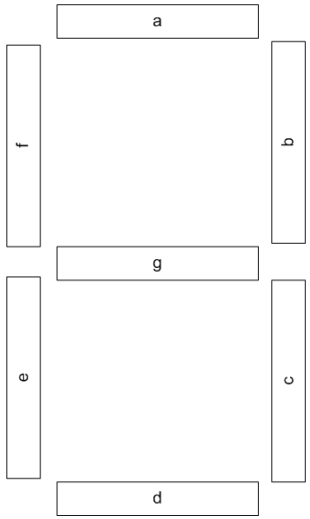
Εργαστήριο 7^{ο}: Να σχεδιαστεί κατάλληλη **Αριθμητική Μονάδα**, με 2 **καταχωρητές A και B** και με ότι άλλα στοιχεία σας χρειάζονται, έτσι ώστε αυτή να είναι ικανή να εκτελεί τις ακόλουθες λειτουργίες:



1. $B \Rightarrow A$
2. $-B \Rightarrow A$
3. $A + B \Rightarrow A$
4. $A - B \Rightarrow A$
5. $B - A \Rightarrow A$
6. $A \Rightarrow B$
7. $-A \Rightarrow B$
8. $A + B \Rightarrow B$
9. $A - B \Rightarrow B$
10. $B - A \Rightarrow B$
11. A Τέλος, δώσατε σε κάθε περίπτω-
12. $B + 1 \Rightarrow B$
13. $-A - 1 \Rightarrow B$
14. $-B - 1 \Rightarrow A$

ση τα σήματα ελέγχου που πρέπει να δημιουργηθούν για την υλοποίηση των παραπάνω διεργασιών.
(**Υπόδειξη:** Εξετάσατε εάν το παρατιθέμενο κύκλωμα μπορεί να ικανοποιήσει τις ανάγκες σας.)

Εργαστήριο 8^{ο}: Δημιουργία Μονοψηφίων αριθμών Ηλεκτρονικά



Δυαδική Μορφή				Αριθμοί	a	b	c	d	e	f	g	Ψηφιακή Παράσταση
A	B	C	D	Δεκαδικός	1: Φωτεινό τμήμα, 0: Σκοτεινό							
0	0	0	0	΄0΄	1	1	1	1	1	1	1	0
0	0	0	1	΄1΄	0	1	1	0	0	0	0	1
0	0	1	0	΄2΄	1	1	0	1	1	0	1	2
0	0	1	1	΄3΄	1							3
0	1	0	0	΄4΄	0							4
0	1	0	1	΄5΄	1							5
0	1	1	0	΄6΄	0							6
0	1	1	1	΄7΄	1							7
1	0	0	0	΄8΄	1							8
1	0	0	1	΄9΄	1	1	1	1	0	1	1	9

Το συνδυαστικό κύκλωμα (σ.κ.) της απεικόνισης καθορίζει την συμπεριφορά των τμημάτων (φωτεινό ή μή) a, b, c, d, e, f και g ανάλογαμε τις τιμές **των δυαδικών μεταβλητών** A, B, C **και** D **(δυαδικά ψηφία των δεκαδικών αριθμών της ψηφιακής παράστασης)**. Έτσι, λοιπόν, για το σ.κ. της συμπεριφοράς του τμήματος $\underline{a}$, βάση του γνωστού κανόνα, λαμβάνουμε όλες τις γραμμές του πίνακα αληθείας, που το $\underline{a}$ έχει τιμή 1 και δημιουργούμε την έκφραση του γινομένου των μεταβλητών (όλων), ή των συμπληρωμάτων τους, ανάλογα της τιμής πού έχουν στη γραμμή αυτή. Έτσι, λοιπόν, για το $\underline{a}$, το σ.κ. **πρέπει να ικανοποιεί την έκφραση:**

$$\bar{A}\cdot\bar{B}\cdot\bar{C}\cdot\bar{D}+\bar{A}\cdot\bar{B}\cdot C\cdot\bar{D}+\bar{A}\cdot\bar{B}\cdot C\cdot D+\bar{A}\cdot B\cdot\bar{C}\cdot D+\bar{A}\cdot B\cdot C\cdot D+A\cdot\bar{B}\cdot\bar{C}\cdot\bar{D}+A\cdot\bar{B}\cdot\bar{C}\cdot D = a(A,B,C,D),$$

πού εύκολα υλοποιείται με τη βοήθεια των πυλών πολλαπλών εισόδων. Να δημιουργηθούν τα σ.κ. όλων των τμημάτων και **να απλοποιηθούν με χάρτη Karnaugh**.