ΜΙΑ ΣΥΝΤΟΜΗ ΕΙΣΑΓΩΓΗ ΣΤΗ FORTRAN 77

Μ. Χ. ΔΡΑΚΟΠΟΥΛΟΣ, Δ. Α. ΜΗΤΣΟΥΔΗΣ, Χ. Α. ΣΦΥΡΑΚΗΣ

ΤΜΗΜΑ ΜΑΘΗΜΑΤΙΚΩΝ

ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΑΘΗΝΑ 2002

Περιεχόμενα

Πρόλογος

Οι σημειώσεις αυτές γράφτηκαν για τις ανάγκες του εργαστηρίου που γίνεται στα πλαίσια του μαθήματος Αριθμητική Ανάλυση Ι στο Μαθηματικό Τμήμα του Πανεπιστημίου Αθηνών. Η γλώσσα FORTRAN είναι στενά συνδεδεμένη με εφαρμογές της Αριθμητικής Ανάλυσης από τη δημιουργία της μέχρι και σήμερα. Η συντριπτική πλειοψηφία του λογισμικού που χρησιμοποιείται στα Υπολογιστικά Μαθηματικά, και στις εφαρμογές των Θετικών Επιστημών γενικότερα, έχει γραφτεί σε FORTRAN.

Η σύντομη αυτή εισαγωγή σε καμμία περίπτωση δεν μπορεί να θεωρηθεί πλήρης. Περιγράφονται μόνο κάποια από τα βασικά στοιχεία της γλώσσας, που όμως είναι αρκετά για την υλοποίηση των προγραμματιστικών αναγκών του μαθήματος. Οι σημειώσεις αυτές απευθύνονται σε αναγνώστες που έχουν κάποια προγραμματιστική εμπειρία, όπως αυτή που παρέχεται από το μάθημα Πληροφορική Ι του Τμήματος, και παραπέμπουμε, όπου είναι αυτό δυνατόν, στις αντίστοιχες δομές της γλώσσας C.

Για όσους ενδιαφέρονται υπάρχει αρκετό υλικό (βιβλία, σημειώσεις κ.λ.π.) σχετικά με τη FORTRAN 77 στο διαδίκτυο. Ενδεικτικά αναφέρουμε τα παρακάτω

http://www.star.le.ac.uk/~cgp/prof77.ps
http://www.kcl.ac.uk/kis/support/cit/fortran/f77book.pdf

1. ΕΙΣΑΓΩΓΗ

Η γλώσσα FORTRAN (FORmula TRANslation) δημιουργήθηκε τη δεκαετία του 1950. Η FORTRAN σχεδιάστηκε εξαρχής για μαθηματικούς σκοπούς, για να κάνει δυνατή την υπολογιστική επίλυση μαθηματικών προβλημάτων. Έτσι, εφόσον ένας αλγόριθμος διατυπωθεί σαφώς (π.χ. με διάγραμμα ροής), η υλοποίησή του σε FORTRAN είναι απλή και άμεση. Επιπλέον, προγράμματα γραμμένα σε FORTRAN είναι γενικά σαφή και εύκολα αντιληπτά (εξαρτάται και από τον προγραμματιστή!). Η γλώσσα είναι αυστηρά δομημένη, όσον αφορά τη σύνταξη των εντολών της, και έτσι πολλά από τα συντακτικά λάθη προγραμματισμού μπορούν να ανακαλυφθούν κατά τη μεταγλώττιση του προγράμματος παρέχοντας σημαντική ασφάλεια στον προγραμματιστή. Η δομή της γλώσσας αλλά και η πολύχρονη χρήση της έχουν οδηγήσει σε compilers με βέλτιστη απόδοση.

Η FORTRAN είναι μία γενική γλώσσα προγραμματισμού που μπορεί να χρησιμοποιηθεί αποτελεσματικά για την υλοποίηση σύνθετων προβλημάτων. Υπάρχουν αρκετές υλοποιήσεις - διάλεκτοι της FORTRAN, ανάλογα με τον τύπο υπολογιστή στον οποίο δουλεύουμε και φτιάχνουμε τα προγράμματά μας. Για τους «προσωπικούς» υπολογιστές που οι περισσότεροι γνωρίζουμε και χρησιμοποιούμε, πιο διαδεδομένη είναι η FORTRAN 77.

2. Η ΜΟΡΦΗ ΕΝΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ

Τα προγράμματα FORTRAN γράφονται σε αρχείο ASCII με κάποιο κειμενογράφο σύμφωνα με τις παρακάτω οδηγίες:

- Γράφουμε μία εντολή ανά γραμμή. Οι εντολές πρέπει να περιέχονται μεταξύ των στηλών 7 και 72.
- Αν το μήκος μιας γραμμής ξεπερνά την 72η στήλη μπορούμε να συνεχίσουμε την εντολή στην επόμενη σειρά βάζοντας οποιοδήποτε χαρακτήρα (εκτός από το 0) στην 6η στήλη.
- Τα σχόλια μπαίνουν γράφοντας το γράμμα `C` ή το χαρακτήρα * στην 1η στήλη.
- Κάθε εντολή είναι δυνατόν να συσχετιστεί με κάποιον αυθαίρετο αριθμό που ονομάζεται ετικέτα (label), για αναφορά σε αυτήν από άλλες εντολές. Η ετικέτα μιας εντολής πρέπει να γράφεται μεταξύ των στηλών 1 και 5 (δηλαδή δεν μπορεί να περιλαμβάνει πάνω από 5 διαδοχικά ψηφία.)
- Τα κενά αγνοούνται από τον compiler (εκτός από την περίπτωση σταθερών τύπου `CHARACTER`).
- Η εντολή `PROGRAM` δηλώνει την αρχή του προγράμματος.
- Η εντολή `STOP` τερματίζει την εκτέλεση του προγράμματος. Η εντολή `END` δηλώνει το πέρας του προγράμματος για τον compiler. Σ' ένα πρόγραμμα μπορούν να υπάρχουν περισσότερες από μία `STOP`, αλλά μόνο μία `END`.

2.1. Ένα πρώτο πρόγραμμα

```
C234567
      PROGRAM FIRST
C Εδώ μπαίνει ένα σχόλιο
      INTEGER I
      DO 10 I = 1,3
        WRITE(*,*) ´Hello World´
  10  CONTINUE
* Μόλις γράψαμε Hello World 3 φορές
      STOP
      END
```

3. ΜΕΤΑΒΛΗΤΕΣ

3.1. Ονομασία μεταβλητών - υποπρογραμμάτων

Τα ονόματα των μεταβλητών πρέπει να δίνονται βάσει των ακόλουθων κανόνων:

- Επιτρέπονται τα γράμματα του λατινικού αλφαβήτου A,…,Z και τα ψηφία 0,…,9.
- Ο πρώτος χαρακτήρας πρέπει να είναι πάντα γράμμα.
- Το μήκος τους δεν πρέπει να ξεπερνά (στη standard FORTRAN 77) τους 6 χαρακτήρες. (Στην πράξη οι περισσότεροι compilers δέχονται μέχρι 31 χαρακτήρες, αυτό όμως μπορεί να δημιουργήσει προβλήματα στη δυνατότητα μεταφοράς).
- Δεν γίνεται διάκριση μεταξύ πεζών και κεφαλαίων. Ο compiler μετατρέπει τα πεζά σε κεφαλαία.
- Πρέπει να προσέχουμε να μην χρησιμοποιούμε τυχόν δεσμευμένες λέξεις (π.χ. εντολές FORTRAN).

Παράδειγμα:

☺ Αποδεκτά ονόματα μεταβλητών: `GEORGE, A3, PROD, sum_1, SINUS, IWRITE`
☹ Μη αποδεκτά ονόματα μεταβλητών: `A$3, sum-1, SIN, WRITE, 3a, 1WRITE`

Παρατήρηση: Πρέπει να είστε προσεκτικοί γιατί συνήθως γίνεται σύγχυση μεταξύ του ψηφίου 0 και του γράμματος Ο, καθώς και του ψηφίου 1 με το γράμμα Ι.

3.2. Τύποι δεδομένων

Στον παρακάτω πίνακα δίνεται ο τύπος των δεδομένων που υποστηρίζει η FORTRAN και η αντιστοιχία τους στην C.

ΠΕΡΙΓΡΑΦΗ ΤΥΠΩΝ	C	FORTRAN	ΠΑΡΑΔΕΙΓΜΑ
Ακέραιοι	`int`	`INTEGER`	`123,-1,+1`
Πραγματικοί (απλή ακρίβεια)	`float`	`REAL`	`12.3,-1.,-1.0E6, 1E-6`
Πραγματικοί (διπλή ακρίβεια)	`double`	`DOUBLE PRECISION` `REAL*8`	`3.141592654, 6.23D12`

Μιγαδικοί	-	COMPLEX	(0.,1.)
Λογικοί	int	LOGICAL	.TRUE. , .FALSE.
Χαρακτήρες	char	CHARACTER	΄DONALD KNUTH΄

3.3. Δήλωση μεταβλητών

Συνιστάται όλες οι μεταβλητές να δηλώνονται στην αρχή ενός προγράμματος, ώστε ο compiler να γνωρίζει το μέγεθος και τον τύπο τους για να δεσμεύσει την απαιτούμενη μνήμη και να έχει τη βέλτιστη απόδοση. Σε αντίθεση με τη C, ο τύπος μιας «αδήλωτης» μεταβλητής καθορίζεται έμμεσα από το πρώτο γράμμα της ως εξής:

- Είναι πραγματική αν αρχίζει με A...H, O...Z.
- Είναι ακέραια αν αρχίζει με I,J,K,L,M,N.

Παραδείγματα

```
INTEGER I, SUM
REAL SINUS,X1,X2
LOGICAL FLAG
DOUBLE PRECISION PI
```

4. ΑΡΙΘΜΗΤΙΚΕΣ ΕΚΦΡΑΣΕΙΣ

Οι σημαντικότεροι αριθμητικοί τελεστές είναι η πρόσθεση +, η αφαίρεση -, ο πολλαπλασιασμός *, η διαίρεση /, και η ύψωση σε δύναμη **. Π.χ. `A+B-C, A*(-B), A*B/C, Z**I`.

Η προτεραιότητα είναι:

α) Υπολογισμός παρενθέσεων (από τις εσωτερικές προς τις εξωτερικές).
β) Ύψωση σε δύναμη.
γ) Πολλαπλασιασμός και διαίρεση.
δ) Πρόσθεση και αφαίρεση.

Παρατηρήσεις:

- Όταν δύο τελεστές έχουν την ίδια προτεραιότητα η έκφραση υπολογίζεται από αριστερά προς τα δεξιά. Σε περίπτωση αμφιβολίας ως προς τη σειρά υπολογισμού συνιστάται η χρήση παρενθέσεων.
- Το αποτέλεσμα μιας διαίρεσης μεταξύ ακεραίων είναι πάντα ακέραιος. (`Π.χ. 10/3 → 3`).
- Όταν μια έκφραση περιέχει και ακέραιες και πραγματικές τιμές, αρχικά οι ακέραιες τιμές μετατρέπονται από τον compiler σε πραγματικές και στη συνέχεια εκτελείται η πράξη. (`Π.χ. 3.0*1/3 → 3.0*1.0/3 → 3.0*1.0/3.0 → 1.0`).

Παραδείγματα:

Η έκφραση `A*B-C/D` υπολογίζεται ως: `(A*B)-(C/D)`
Η έκφραση `A/B*C` υπολογίζεται ως: `(A/B)*C`

5. ΕΝΤΟΛΕΣ ΑΝΑΘΕΣΗΣ

Η απλούστερη εντολή στη FORTRAN είναι η ανάθεση και έχει τη μορφή

μτβ = εκφρ

η οποία αναθέτει την τιμή της έκφρασης *εκφρ* στη μεταβλητή *μτβ*.

Π.χ. `RES = B + C*D**3` σημαίνει $RES = B + C\ D^3$.

Ο τύπος της *εκφρ* πρέπει εν γένει να συμφωνεί με τον τύπο της *μτβ*. Σε αντίθετη περίπτωση η τιμή της *εκφρ* μετατρέπεται αυτόματα στον τύπο της *μτβ* προτού γίνει ανάθεση.

Π.χ.

`B = 9` → Στην πραγματική μεταβλητή `B` ανατίθεται η τιμή `9.0`.

`I = 1.9` → Στην ακέραια μεταβλητή `I` ανατίθεται η τιμή `1`

6. ΕΝΤΟΛΕΣ ΕΙΣΟΔΟΥ – ΕΞΟΔΟΥ

Στα περισσότερα προγράμματα απαιτείται η εισαγωγή δεδομένων από το χρήστη, συνήθως από το πληκτρολόγιο. Σχεδόν δε πάντα ένα χρήσιμο πρόγραμμα εκτυπώνει κάποια αποτελέσματα, π.χ. στην οθόνη. Για τους σκοπούς αυτούς χρησιμοποιούμε τις εντολές εισόδου – εξόδου.

6.1. Η εντολή READ

Σύνταξη: `READ(a,b) μτβ1, μτβ2, ...`

Το *a* δηλώνει από ποια μονάδα θα διαβάσουμε τα δεδομένα. Για είσοδο από το πληκτρολόγιο *a*=5 ή *a*=*.

Αν *b*=* η μορφή της ανάγνωσης καθορίζεται από τον τύπο του αντίστοιχου δεδομένου.

Αν το *b* είναι ένας φυσικός αριθμός, τότε παραπέμπει σε εντολή `FORMAT` που καθορίζει την ακριβή μορφή ανάγνωσης.

6.2. Η εντολή WRITE

Σύνταξη: `WRITE(a,b) εκφρ1,εκφρ2, ...`

Το *a* δηλώνει σε ποια μονάδα θα εκτυπώσουμε τα δεδομένα. Για έξοδο στην οθόνη *a*=6 ή *a*=*.

Αν *b*=* η μορφή της εκτύπωσης καθορίζεται από τον τύπο του αντίστοιχου δεδομένου.

Αν το *b* είναι ένας φυσικός αριθμός, τότε παραπέμπει σε εντολή `FORMAT` που καθορίζει την ακριβή μορφή εκτύπωσης.

6.3. Η εντολή περιγραφής FORMAT

Αν θέλουμε να καθορίσουμε ακριβώς τη μορφή της εισόδου/εξόδου των δεδομένων χρησιμοποιούμε την εντολή FORMAT που συντάσσεται ως εξής:

`FORMAT (`*ακολουθία περιγραφής*`)`

Η περιγραφή των συνηθέστερων τύπων δεδομένων δίνεται στον παρακάτω πίνακα

Τύπος δεδομένων	Περιγραφή
Ακέραιος	I*w*
Πραγματικός	F*w*.*d*
Χαρακτήρας	A

όπου *w* είναι το ολικό μήκος του πεδίου και *d* είναι το πλήθος των ψηφίων μετά την υποδιαστολή (ακρίβεια). (Πρέπει *w* ≥ *d* + 3). Επίσης με *n*X μετακινούμαστε *n* θέσεις προς τα δεξιά (αφήνοντας *n* κενά). Η εντολή FORMAT είναι μη εκτελέσιμη και χρησιμοποιείται σε συνδυασμό με κάποια READ ή WRITE, οπότε μπροστά από τη FORMAT (μεταξύ των στηλών 1 και 5) πρέπει να υπάρχει μία ετικέτα που να τη συνδέει με την αντίστοιχη READ ή WRITE.

Παράδειγμα: Με τις εντολές

```
      PI = 3.141593
      WRITE(*,10) ´The number pi is:’,PI
10    FORMAT(2X,A,F5.2)
```

θα τυπώσουμε στην οθόνη τη γραμμή:

```
␣␣The number pi is:␣3.14
```

7. ΛΟΓΙΚΕΣ ΕΚΦΡΑΣΕΙΣ

Είναι εκφράσεις που παίρνουν τις τιμές: .TRUE. ή .FALSE.
Σχηματίζονται συνήθως από συγκρίσεις αριθμητικών δεδομένων (τελεστές συσχέτισης) και συνδυασμούς τους (λογικοί τελεστές).

Στον πίνακα που ακολουθεί δίνουμε την μορφή των *τελεστών συσχέτισης* στη FORTRAN και την αντιστοιχία τους στη C.

FORTRAN	.EQ.	.NE.	.GT.	.LT.	.GE.	.LE.
C	==	!=	>	<	>=	<=

Στον ακόλουθο δίνουμε την μορφή των *λογικών τελεστών* στη FORTRAN και την αντιστοιχία τους στη C.

FORTRAN	.NOT.	.AND.	.OR.
C	!	&&	||

Π.χ. η έκφραση (A .GE. B) είναι αληθής αν η τιμή της μεταβλητής A είναι μεγαλύτερη ή ίση από την τιμή της μεταβλητής B.

Προτεραιότητα τελεστών:
- ( )
- Αριθμητικοί τελεστές

- Τελεστές συσχέτισης
- Λογικοί τελεστές

Παράδειγμα: Οι ακόλουθες δύο γραμμές είναι ισοδύναμες

```
 A .LE. B .AND. B .LE. C
(A .LE. B) .AND. (B .LE. C)
```

8. Η ΕΝΤΟΛΗ ΕΛΕΓΧΟΥ IF

Σύνταξη:

- ```
  IF (λογική παράσταση) THEN
    εντολές (εκτελούνται αν λογική παράσταση αληθής)
  ENDIF
  ```

- ```
  IF (λογική παράσταση) THEN
    εντολές (εκτελούνται αν λογική παράσταση αληθής)
  ELSE
    εντολές (εκτελούνται αν λογική παράσταση ψευδής)
  ENDIF
  ```

- ```
  IF (λογ. παράστ. 1) THEN
    εντολές (εκτελούνται αν η 1 είναι αληθής)
  ELSE IF (λογ. παράστ. 2) THEN
    εντολές (εκτελούνται αν η 2 είναι αληθής)
       ∂
  ELSE IF (λογ. παράστ. n) THEN
    εντολές (εκτελούνται αν η n είναι αληθής)
  ELSE
    εντολές (εκτελούνται αν οι 1,2,¥,n  είναι ψευδείς)
  ENDIF
  ```

Παράδειγμα:
Για να βρούμε το πρόσημο SIGNUM ενός αριθμού NUMBER (SIGNUM=1 για NUMBER>0, SIGNUM=-1 για NUMBER<0, SIGNUM=0 για NUMBER=0) μπορούμε να χρησιμοποιήσουμε την ακόλουθη δομή:

```
IF (NUMBER .LT. 0) THEN
  SIGNUM = -1
ELSE IF (NUMBER .GT. 0) THEN
  SIGNUM = 1
ELSE
  SIGNUM = 0
ENDIF
```
```

Το αντίστοιχο πρόγραμμα C είναι:

```
if (number < 0){
    signum = -1;
} else if (number > 0) {
    signum = 1;
} else {
    signum = 0;
}
```

9. Η ΕΝΤΟΛΗ ΕΠΑΝΑΛΗΨΗΣ DO

Σύνταξη:

```
      DO s μτβ = εκφρ1, εκφρ2, εκφρ3
        εντολές
   s  CONTINUE
```

Παρατηρήσεις:

- Συνήθως η *εκφρ1* αντιστοιχεί σε κάποια αρχική τιμή, η *εκφρ2* σε κάποια τελική τιμή, ενώ η *εκφρ3* είναι το βήμα. Η *εκφρ3* είναι προαιρετική. Όταν παραλείπεται το βήμα είναι 1.
- Ο αριθμός των επαναλήψεων *κ* που θα εκτελεσθούν τελικά, δίνεται από τον τύπο:
  κ = MAX (INT((*εκφρ2* − *εκφρ1* + *εκφρ3*)/*εκφρ3*) , 0)
- Η τιμή της *μτβ* ελέγχεται αποκλειστικά από την εντολή DO και δεν πρέπει να μεταβληθεί μέσα στην επανάληψη.
- Προτού αρχίσουν οι επαναλήψεις:
  - Υπολογίζονται οι τιμές των *εκφρ1, εκφρ2* και *κ* .
  - Η *μτβ* αρχικοποιείται από *εκφρ1* .
  - Ελέγχεται αν *κ* > 0 οπότε και εκτελούνται οι εντολές του DO.
- Όταν κάθε επανάληψη φτάνει στην εντολή CONTINUE:
  - Η *μτβ* αυξάνει κατά το βήμα (*εκφρ3*).
  - Η τιμή του *κ* ελαττώνεται κατά 1.
  - Ελέγχεται αν *κ* > 0 οπότε και συνεχίζονται οι επαναλήψεις, διαφορετικά η επαναληπτική διαδικασία ολοκληρώνεται και εκτελείται η επόμενη εντολή της CONTINUE.

Παραδείγματα:

```
      DO 10 I=1,3
        WRITE(*,*)I     T Τυπώνει στην οθόνη (σε μία στήλη) τους αριθμούς 1,2,3
  10  CONTINUE

      DO 200 K=10,2,-1
        Y = 2.0*K       T Τυπώνει στην οθόνη (σε μία στήλη) τους αριθμούς 20,18,...,4
        WRITE(6,*) Y
200   CONTINUE
```

```
      PI = 3.141593
      DO 19 KAPPA=3,9,3
        RES = COS(KAPPA*PI)  Τ Τυπώνει στην οθόνη τους αριθμούς -1,1,-1
        WRITE(*,*) RES
   19 CONTINUE
```

Το αντίστοιχο πρόγραμμα στη C είναι:

```
pi = 3.141593;
for (kappa = 3; kappa <= 9; kappa + = 3) {
   res = cos(kappa*pi);
   printf("%f\n",res);
}
```

9.1. Το υπαινισσόμενο (implied) DO

Σύνταξη:

εντολή εισόδου/εξόδου, (*μτβ = εκφρ1,εκφρ2,εκφρ3*)

Όπως και στην εντολή DO, συνήθως η *εκφρ1* αντιστοιχεί σε κάποια αρχική τιμή, η *εκφρ2* σε κάποια τελική τιμή, ενώ η *εκφρ3* είναι το βήμα. Η *εκφρ3* είναι προαιρετική. Όταν παραλείπεται το βήμα είναι 1.

Παράδειγμα:

```
WRITE(*,*) (I*I, I=1,3)   Τ Τυπώνει σε μία σειρά στην οθόνη
                            τους αριθμούς 1  4  9
```

10. Η ΕΝΤΟΛΗ ΕΠΑΝΑΛΗΨΗΣ " WHILE"

Στη C χρησιμοποιείται και η εντολή while:

```
   while (λογική έκφραση) {
      εντολές (αληθής)
   }
```

Στη FORTRAN 77 όμως δεν υπάρχει αντίστοιχη εντολή και η δομή που αντιπροσωπεύεται από τη while υλοποιείται ως εξής:

```
s  IF (λογική έκφραση) THEN
      εντολές (αληθής)
      GOTO s
   ENDIF
```

Παράδειγμα 1: Να γραφεί πρόγραμμα FORTRAN που να αθροίζει όλους τους ακέραιους αριθμούς μεταξύ δύο δεδομένων ακεραίων a και b

```
C234567
      PROGRAM SUMAB
C  Αθροίζει τους ακεραίους μεταξύ a και b.
      INTEGER A, B, SUM, I
      WRITE(*,*) 'Δώστε δύο ακεραίους: '
      READ(*,*) A, B
      SUM = 0
      IF (A .LT. B) THEN
        DO 10 I= A,B
          SUM = SUM + I
 10     CONTINUE
      ELSE
        DO 20 I = A, B,-1
          SUM = SUM + I
 20     CONTINUE
      ENDIF
      WRITE(*,*) 'Το άθροισμα είναι: ',SUM
      STOP
      END
```

Παράδειγμα 2: Να γραφεί πρόγραμμα FORTRAN που να υπολογίζει το άθροισμα και το πλήθος των όρων της σειράς 1 + 1/2 + 1/3 + 1/4 + ..., μέχρι τον όρο που είναι μικρότερος από ε = 0.005

```
*234567
      PROGRAM SEIRA
      REAL  SUM,OROS,EPSLON
      INTEGER  COUNT
      EPSLON = 5.E-3
      OROS = 1.
      COUNT = 1
      SUM = 0. 10
 10   IF (OROS .GE. EPSLON) THEN
        SUM = SUM + OROS
        COUNT = COUNT + 1
        OROS = 1./FLOAT(COUNT)
        GOTO 10
      ENDIF
      WRITE(*,*) 'Άθροισμα = ', SUM,' Πλήθος όρων = ',COUNT-1
      STOP
      END
```

Παράδειγμα 3: Να γραφεί πρόγραμμα που να τυπώνει τον παρακάτω πίνακα πολλαπλασιασμού:

```
   0   1   2   3   4   5   6   7   8   9
   1   1
   2   2   4
   3   3   6   9
   4   4   8  12  16
   5   5  10  15  20  25
   6   6  12  18  24  30  36
   7   7  14  21  28  35  42  49
   8   8  16  24  32  40  48  56  64
   9   9  18  27  36  45  54  63  72  81
```

```
*234567
      PROGRAM MMATR
c  Υπολογισμός πίνακα πολλαπλασιασμού.
      INTEGER I, J
      WRITE(*,*) (I, I=0,9)
      DO 10 I = 1,9
        WRITE(*,*) I, (I*J,J=1,I)
 10   CONTINUE
      STOP
      END
```

11. ΠΙΝΑΚΕΣ – ΔΙΑΝΥΣΜΑΤΑ

Ένας πίνακας (array) είναι μία δομημένη μεταβλητή που χρησιμεύει για την καταχώριση δεδομένων του ίδιου τύπου. Οι πίνακες δηλώνονται όπως και οι συνήθεις μεταβλητές, μόνο που πρέπει να ορίσουμε την αρχική και την τελική τιμή των δεικτών. Δηλαδή γράφουμε

$$\textit{τύπος}\ \ \textit{πιν}(l_1{:}u_1, l_2{:}u_2, \Omega, l_n{:}u_n)$$

όπου $\textit{πιν}$ είναι το όνομα του πίνακα, l_i είναι ακέραια σταθερά που δηλώνει το κάτω φράγμα του δείκτη του i διανύσματος $(i=1,2,\ \Omega,n)$ και u_i είναι ακέραια σταθερά που δηλώνει το άνω φράγμα του δείκτη του i διανύσματος $(i=1,2,\ \Omega,n)$. Όταν παραλείπεται το l_i θεωρούμε ότι είναι 1.

Παράδειγμα 1:

`REAL A(4)` → καθορίζει το διάνυσμα Α με στοιχεία A_1, A_2, A_3, A_4.

`DIMENSION A(4)` → καθορίζει το διάνυσμα Α με στοιχεία A_1, A_2, A_3, A_4.

`REAL X(3,0:3)` → καθορίζει τον πίνακα Χ με 12 στοιχεία X_{10}, X_{20}, X_{30}, X_{11}, X_{21}, X_{31}, X_{12}, X_{22}, X_{32}, X_{13}, X_{23}, X_{33} που είναι αποθηκευμένα σε *συνεχόμενες* θέσεις στη μνήμη του υπολογιστή. Παρατηρήστε ότι η αποθήκευση στη μνήμη των στοιχείων του πίνακα γίνεται κατά στήλες.

Παράδειγμα 2:
Να γραφεί ένα πρόγραμμα FORTRAN που να διαβάζει δύο διανύσματα του $\mathbf{R}^3$ και να υπολογίζει το εσωτερικό τους γινόμενο.

```
C234567
      PROGRAM INPROD
      REAL A(3), B(3)
      INTEGER I
      REAL ES_GIN
      WRITE(*,*) 'Δώστε το πρώτο διάνυσμα:'
      DO 10 I= 1,3
        READ(*,*) A(I)
 10   CONTINUE
      WRITE(*,*) 'Δώστε το δεύτερο διάνυσμα:'
      DO 20 I = 1,3
        READ(*,*) B(I)
 20   CONTINUE
      ES_GIN = 0.
      DO 30 I = 1,3
        ES_GIN = ES_GIN + A(I)*B(I)
 30   CONTINUE
      WRITE(*,*) ' A = (',(A(I), I=1,3), ')'
      WRITE(*,*) ' B = (',(B(I), I=1,3), ')'
      WRITE(*,*) ' Εσωτ. γινόμενο: ',ES_GIN
      STOP
      END
```

Παράδειγμα 3: Να γραφεί πρόγραμμα που να διαβάζει ένα πραγματικό πίνακα 4×4 και να τυπώνει τα αθροίσματα στηλών, γραμμών και όλων των στοιχείων.

```
C234567
      PROGRAM  MATR
      REAL  A(4,4)
      INTEGER  I, J
      REAL SUM
C Τα διανύσματα STILI και GRAMMI αποθηκεύουν τα μερικά αθροίσματα
C στηλών και γραμμών.
      REAL STILI(4),GRAMMI(4)
      DO 10  I=1,4
        DO 20 J=1,4
          WRITE(*,*) ΄Δώσε το στοιχειο A(',I,',',J,')=΄
          READ(*,*) A(I,J)
 20     CONTINUE
 10   CONTINUE
      DO 30 I=1,4
        GRAMMI(I) = 0.
```

```
        STILI(I) = 0.
30    CONTINUE
      SUM = 0.
      DO 40  I=1,4
        DO 50 J=1,4
          GRAMMI(I) = GRAMMI(I) + A(I,J)
          STILI(I) = STILI(I) + A(J,I)
          SUM = SUM + A(I,J)
50      CONTINUE
40    CONTINUE
      DO 60 I=1,4
        WRITE(*,*) (A(I,J), J=1,4)
60    CONTINUE
      WRITE(*,*)
      DO 70  I=1,4
        WRITE(*,*)'Το άθροισμα της',I,' γραμμής είναι:', GRAMMI(I)
70    CONTINUE
      DO 80  I=1,4
        WRITE(*,*)'Το άθροισμα της',I,' στήλης είναι:', STILI(I)
80    CONTINUE
      WRITE(*,*)'Το συνολικό άθροισμα',SUM
      STOP
      END
```

12. ΕΙΣΟΔΟΣ - ΕΞΟΔΟΣ ΑΠΟ ΑΡΧΕΙΟ

Πολλές φορές ο όγκος των δεδομένων που πρέπει να διαβαστούν από ένα πρόγραμμα είναι μεγάλος, επομένως η πληκτρολόγηση τους κάθε φορά που εκτελείται το πρόγραμμα είναι κουραστική ή και πρακτικά αδύνατη, και φυσικά υπάρχει μεγάλη πιθανότητα να γίνει κάποιο λάθος. Επίσης συχνά θέλουμε να διατηρήσουμε τα αποτελέσματα ενός προγράμματος είτε για μελέτη, είτε για πιθανή χρήση τους ως δεδομένα εισόδου από κάποιο άλλο πρόγραμμα Στις περιπτώσεις αυτές χρησιμοποιούμε αρχεία για την είσοδο ή έξοδο των δεδομένων. Τα αρχεία εισόδου/εξόδου που θα χρησιμοποιήσουμε θα πρέπει να τα συνδέσουμε μέσα στο πρόγραμμα με κάποια μονάδα εισόδου/εξόδου *a* (βλ. `READ`, `WRITE`). Η σύνδεση γίνεται με χρήση της εντολής `OPEN`.

Σύνταξη:

```
OPEN(a, FILE='arxeio')
```

Ένα αρχείο που συνδέθηκε στο πρόγραμμα με την εντολή `OPEN` θα πρέπει να αποσυνδεθεί από την μονάδα *a* με την εντολή `CLOSE`.

Σύνταξη:

```
CLOSE(a)
```

Επισημαίνουμε ότι το a είναι ένας φυσικός διαφορετικός του 5 και του 6, αφού οι αριθμοί αυτοί έχουν συνδεθεί με το πληκτρολόγιο και την οθόνη, αντίστοιχα (βλέπε READ,WRITE.)

Παρατήρηση: Δεν μπορείτε να συνδέσετε το ίδιο αρχείο σε δύο μονάδες, αν δεν το έχετε πρώτα αποσυνδέσει με την εντολή CLOSE. Επίσης, δεν μπορείτε να συνδέσετε δύο αρχεία στην ίδια μονάδα.

Παράδειγμα: Να γραφεί ένα πρόγραμμα FORTRAN που να διαβάζει από το αρχείο input.dat τη διάσταση n, ένα (n×n) πίνακα A και ένα (n×1) διάνυσμα X, να υπολογίζει το διάνυσμα AX και να το αποθηκεύει σ' ένα αρχείο με όνομα output.dat

```
C234567
      PROGRAM MATVEC
      REAL A(30,30), X(30), S
      OPEN(1,FILE='input.dat')
      READ(1,*) N
      DO 10 I=1,N
        READ(1,*) (A(I,J), J=1,N)
  10  CONTINUE
      READ(1,*) (X(I), I=1,N)
      CLOSE(1)
      OPEN(2,FILE='output.dat')
      WRITE(2,*) 'Το διάνυσμα είναι:'
      DO 20 I=1,N
        S=0.0
        DO 30 J=1,N
          S = S + A(I,J)*X(J)
  30    CONTINUE
        WRITE(2,*) S
  20  CONTINUE
      STOP
      END
```

Το αρχείο input.dat πρέπει να είναι π.χ. της μορφής:

```
 3
 1    0    1
 0    2   -1
-2    1    0
-0.5
 1
 0.5
```

Στην περίπτωση αυτή n=3, $A = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 2 & -1 \\ -2 & 1 & 0 \end{pmatrix}$ και $B = \begin{pmatrix} -0.5 \\ 1 \\ 0.5 \end{pmatrix}$,

ενώ το αρχείο `output.dat` που θα προκύψει θα είναι το

```
 0
1.5
 2
```

13. ΣΥΝΑΡΤΗΣΕΙΣ - ΥΠΟΡΟΥΤΙΝΕΣ

13.1. Εσωτερικές συναρτήσεις (Intrinsic functions)

Η FORTRAN έχει μια βιβλιοθήκη από «εσωτερικές» συναρτήσεις που είναι δυνατόν να χρησιμοποιηθούν σε κάθε πρόγραμμα. Μερικές από τις πιο σημαντικές αυτές συναρτήσεις είναι οι `ABS(X)`, `TAN(X)`, `SIN(X)`, `COS(X)`, `EXP(X)`, `LOG(X)`(βάση το e), `MIN(X1,X2,...)`, `MAX(X1,X2,...)`, `INT(X)`, `FLOAT(I)`, `SQRT(X)`, `MOD(X,Y)`. Οι συναρτήσεις αυτές επιστρέφουν μια τιμή ανάλογα με την τιμή που παίρνουν τα ορίσματά τους. Κάποιες από τις συναρτήσεις αυτές είναι γενικές με την έννοια ότι μπορούν να δεχθούν διαφόρων τύπων ορίσματα, ενώ άλλες δέχονται μόνο ορίσματα συγκεκριμένου τύπου. Π.χ. είναι λάθος να γράψουμε `SQRT(9)`. Το σωστό είναι `SQRT(9.0)`, αφού το όρισμα της `SQRT` δεν μπορεί να είναι ακέραιος. Στο τέλος αυτής της παρουσίασης υπάρχει ένας ενδεικτικός πίνακας με περισσότερες πληροφορίες για κάποιες εσωτερικές συναρτήσεις.

13.2. Συναρτήσεις εντολής (Statement functions)

Οι συναρτήσεις εντολής (statement functions) χρησιμοποιούνται για απλές συνήθως συναρτήσεις (μιας ή περισσοτέρων μεταβλητών) και ορίζονται στην αρχή του προγράμματος. Οι συναρτήσεις αυτές παίρνουν το όνομα τους από την ιδιότητα τους να γράφονται σε μια εντολή.

Παρατηρήστε ότι θα πρέπει να αποφεύγουμε να δίνουμε στις συναρτήσεις που ορίζουμε εμείς ονόματα ίδια με των «εσωτερικών» συναρτήσεων. Επίσης η FORTRAN δεν μας επιτρέπει να δηλώνουμε συναρτήσεις αναδρομικά, αφού δεν επιτρέπεται μια συνάρτηση να καλεί τον εαυτό της.

Παράδειγμα:

```
F(X) = X*X + 3.0*X - 2.0
D(X,Y) = ABS(X-Y)
```

13.3. Υποπρογράμματα

Με τον όρο υποπρογράμματα στη FORTRAN εννοούμε «μικρά» ανεξάρτητα προγράμματα που λειτουργούν βοηθητικά στο κυρίως πρόγραμμα. Υπάρχουν δύο κατηγορίες υποπρογραμμάτων: Οι συναρτήσεις (functions) και τα υποπρογράμματα (subroutines).

13.3.1. Συναρτήσεις ως υποπρογράμματα

Συχνά θέλουμε να ορίσουμε πιο πολύπλοκες συναρτήσεις, τις οποίες ορίζουμε μέσω υποπρογραμμάτων.

Γενική μορφή Συνάρτησης:

```
[τύπος] FUNCTION όνομα(μτβ1, μτβ2,…, μτβν)
Δήλωση μεταβλητών μτβ1, μτβ2,…, μτβν
εντολές
όνομα = παράσταση
RETURN (→ η RETURN επιστρέφει την τιμή στο κυρίως πρόγραμμα)
END
```

13.3.2. Το υποπρόγραμμα SUBROUTINE

Ένα υποπρόγραμμα SUBROUTINE είναι μια διαδικασία που διαφέρει από μια συνάρτηση στα ακόλουθα σημεία:

α) Συνήθως η συνάρτηση επιστρέφει μία μόνο τιμή στο κυρίως πρόγραμμα, ενώ η υπορουτίνα επιστρέφει πάνω από μία τιμή. Σε κάποιες περιπτώσεις δεν επιστρέφει καμμία τιμή, αλλά εκτελεί κάποια συγκεκριμένη εργασία, π.χ. παραθέτει κάποιες οδηγίες στο χρήστη.

β) Η συνάρτηση επιστρέφει την τιμή μέσω του ονόματος της, ενώ η υπορουτίνα μέσω των ορισμάτων της.

γ) Η συνάρτηση καλείται μέσω του ονόματος της, ενώ η υπορουτίνα μέσω της εντολής CALL.

Σύνταξη:

```
SUBROUTINE όνομα(μτβ1, μτβ1,…, μτβν)
Δήλωση μεταβλητών μτβ1, μτβ1,…, μτβν
   ∂
 Εντολές
   ∂
RETURN
END
```

Τρόπος κλήσης:

```
CALL όνομα(μτβ1, μτβ1,…, μτβν)
```

Παράδειγμα 1 (συναρτήσεις εντολής): Να γραφεί πρόγραμμα που να υπολογίζει την τιμή του γινομένου δυο δεδομένων συναρτήσεων F(X), G(X), στα σημεία X_i, i=0,1,…,100, ενός ομοιομόρφου διαμερισμού του διαστήματος [0,2]. Να τυπώνει σε ένα αρχείο με το όνομα result.dat τα αποτελέσματα.

```
C234567
      PROGRAM SYNART
      F(X) = COS(X) + SIN(X)
      G(X) = X**2 - 3.0*x + 2.
      REAL  X,H,TMP
      INTEGER  I
      H = 2./100.
```

```
      OPEN(1,FILE=´result.dat´)
      DO 10 I=0,100
        X = H*I
        TMP = F(X)*G(X)
        WRITE(1,12) X, ´  ´, TMP
  10  CONTINUE
      CLOSE(1)
  12  FORMAT(F10.5,A,F10.5)
      STOP
      END
```

Παράδειγμα 2 (συνάρτηση ως υποπρόγραμμα): Δεδομένων δύο φυσικών αριθμών *n* και *k (n ≥ k)* να γραφεί πρόγραμμα που να υπολογίζει το πλήθος των συνδυασμών *n* ανά *k* που δίνεται από τον τύπο: $\binom{n}{k} = \frac{n!}{k!(n-k)!}$.

```
C234567
      PROGRAM COMB
      INTEGER K,N,RES,FACTOR
      WRITE(*,*) ´Δώσε τους αριθμούς K και N με K<= N´
      IF (K .GT. N) THEN
        WRITE(*,*)´Δώσατε λάθος αριθμούς´
        STOP
      ENDIF
      RES = FACTOR(N)/(FACTOR(K)*FACTOR(N-K))
      WRITE(*,*)´Οι n ανά k συνδυασμοί είναι:´,RES
      STOP
      END

      FUNCTION FACTOR(L)
* Συνάρτηση που υπολογίζει το L!
      INTEGER FACTOR,L,I
      FACTOR = 1
      DO 10 I=2,L
        FACTOR = FACTOR * I
  10  CONTINUE
      RETURN
      END
```

Παράδειγμα 3 (υπορουτίνα): Το παρακάτω πρόγραμμα διαβάζει τις πολικές συντεταγμένες ενός σημείου τις μετατρέπει σε καρτεσιανές και τις εκτυπώνει στην οθόνη. Για το σκοπό αυτό χρησιμοποιείται η subroutine `CONVER`.

```
C234567
      PROGRAM POLCAR
      REAL RCOORD, TCOORD, XCOORD, YCOORD
```

```
      INTEGER RESPON
      RESPON = 1
 10   IF (RESPON .EQ. 1) THEN
        WRITE(*,*) ΄Δώσε τις πολικές συντ/νες (σε rad)΄
        READ(*,*) RCOORD,TCOORD
        CALL CONVER(RCOORD,TCOORD,XCOORD,YCOORD)
        WRITE(*,*)΄Καρτεσιανές συντ/νες:΄
        WRITE(*,*) XCOORD,YCOORD
        WRITE(*,*)
        WRITE(*,*)΄Μετατροπή και άλλου σημείου;'
        WRITE(*,*)΄Αν ναι πατήστε 1, αν όχι 0΄
        READ(*,*) RESPON
        GOTO 10
      ENDIF
      STOP
      END


      SUBROUTINE CONVER(R,THETA,X,Y)
      REAL R,THETA,X,Y
      X = R*COS(THETA)
      Y = R*SIN(THETA)
      RETURN
      END
```

13.4. Η εντολή EXTERNAL

Όταν ένα υποπρόγραμμα (συνάρτηση-FUNCTION ή μια διαδικασία-SUBROUTINE) είναι όρισμα κάποιου άλλου υποπρογράμματος, τότε το όνομα του πρέπει να δηλώνεται με την εντολή EXTERNAL. Η δήλωση της γίνεται ως εξής:

```
EXTERNAL onoma1, onoma2,..., onoman
```

Σημειώνουμε ότι η EXTERNAL χρησιμοποιείται στην αρχή του κυρίως προγράμματος.

Παράδειγμα: Ένα ανεξάρτητο πρόγραμμα γραφικών για να κάνει τη γραφική παράσταση μιας συνάρτησης f παίρνει τα δεδομένα από ένα αρχείο GRAPH.DAT, που περιέχει σε δύο στήλες τα x και f(x), αντίστοιχα, υπολογισμένα σε n+1 διακριτά ισαπέχοντα σημεία. Θέλουμε να γράψουμε ένα πρόγραμμα που να διαβάζει τα a, b, n και στη συνέχεια να καλεί την υπορουτίνα PLOTW που δημιουργεί το αρχείο GRAPH.DAT για οποιαδήποτε συνάρτηση επιθυμούμε. Για να το πετύχουμε αυτό περνάμε ως επιπλέον όρισμα στην PLOTW το όνομα της συνάρτησης.

```
C234567
      PROGRAM MYPLOT
C Αυτό είναι το κυρίως πρόγραμμα. Ο χρήστης δίνει τα άκρα του
C διαστήματος και το πλήθος των υποδιαστημάτων.
      EXTERNAL  F
      REAL A, B
```

```
      INTEGER N
      WRITE(*,*)'Δώσε τα άκρα του διαστήματος a,b'
      READ(*,*) A, B
      WRITE(*,*)'Δώσε το πλήθος των υποδιαστημάτων n'
      READ(*,*) N
C Εδώ καλείται η υπορουτίνα PLOTW
      CALL PLOTW(F, A, B, N)
      STOP
      END

      SUBROUTINE PLOTW(FX,AX,BX,NINT)
C Η υπορουτίνα PLOTW υπολογίζει τις τιμές της συνάρτησης FX στα
C NINT+1 σημεία ενός ομοιόμορφου διαμερισμού του διαστήματος [AX,BX]
C και αποθηκεύει τις τιμές X και FX(X) σε δύο στήλες στο αρχείο
C GRAPH.DAT
      EXTERNAL FX
      REAL AX,BX,H,X
      INTEGER NINT,I
      OPEN(1, FILE='GRAPH.DAT')
      H = ABS(BX-AX)/FLOAT(NINT)
      X = AX
      DO 10 I = 0, NINT
         WRITE(1, *) X, FX(X)
         X = X + H
   10 CONTINUE
      CLOSE(1)
      RETURN
      END

      REAL FUNCTION F(X)
C Εδώ δίνεται η συνάρτηση F(X). Για να υπολογίσουμε σημεία για
C διάφορες συναρτήσεις, αρκεί να αλλάζουμε αυτή τη συνάρτηση.
      REAL X
      F = SIN(X*X) + COS(X)**2
      RETURN
      END
```

Παράρτημα: Εσωτερικές Συναρτήσεις

Γενική ονομασία συνάρτησης	Περιγραφή	Ειδική ονομασία συνάρτησης	Αριθμός ορισμάτων	Τύπος*	
				Ορίσματος	Συνάρτησης
ABS(X)	Απόλυτη τιμή του X	IABS	1	I	I
		ABS	1	R	R
		DABS	1	DP	DP
ACOS(X)	Τόξο συνημιτόνου X, όπου -1≤X≤1 και 0 ≤ αποτέλεσμα ≤ π.	ACOS	1	R	R
		DACOS	1	DP	DP
AINT(X)	Μετατρέπει το X σε ακέραιο με αποκοπή και στη συνέχεια τον μετατρέπει στον ίδιο τύπο με του ορίσματος	AINT	1	R	R
		DINT	1	DP	DP
ANINT(X)	Μετατρέπει το X σε ακέραιο με στρογγύλευση και στη συνέχεια τον μετατρέπει στον ίδιο τύπο με του ορίσματος	ANINT	1	R	R
		DNINT	1	DP	DP
ASIN(X)	Τόξο ημιτόνου X, όπου -1≤X≤1 και $-\pi/2 \le$ αποτέλεσμα $\le \pi/2$.	ASIN	1	R	R
		DASIN	1	DP	DP
ATAN(X)	Τόξο εφαπτομένης X. $-\pi/2 \le$ αποτέλεσμα $\le \pi/2$.	ATAN	1	R	R
		DATAN	1	DP	DP
COS(X)	Συνημίτονο του X (σε rad).	COS	1	R	R
		DCOS	1	DP	DP
COSH(X)	Υπερβολικό συνημίτονο του X.	COSH	1	R	R
		DCOSH	1	DP	DP
DBLE(X)	Μετατρέπει το X σε διπλή ακρίβεια.	-	1	I	DP
		-	1	R	DP
		-	1	DP	DP
EXP(X)	Η εκθετική συνάρτηση e^x	EXP	1	R	R
		DEXP	1	DP	DP
FLOAT(X)	Μετατρέπει τον X σε πραγματικό	-	1	I	R
		-	1	R	R

* I = INTEGER, R = REAL, DP = DOUBLE PRECISION

INT(X)	Μετατρέπει τον X σε ακέραιο με αποκοπή του δεκαδικού μέρους.	INT INT IDINT	1 1 1	I R DP	I I I
LOG(X)	Λογάριθμος του X με βάση e.	ALOG DLOG	1 1	R DP	R DP
LOG10(X)	Λογάριθμος του X με βάση 10.	ALOG10 DLOG10	1 1	R DP	R DP
MAX(X1,Ω,Xn)	Το μέγιστο των X1,Ω,Xn.	MAX0 AMAX1 DMAX1	≥2 ≥2 ≥2	I R DP	I R DP
MIN(X1,Ω,Xn)	Το ελάχιστο των X1,Ω,Xn.	MIN0 AMIN1 DMIN1	≥2 ≥2 ≥2	I R DP	I R DP
MOD(X,Y)	X - INT(X/Y)*Y	MOD AMOD DMOD	2 2 2	I R DP	I R DP
NINT(X)	Μετατρέπει το X σε ακέραιο με στρογγύλευση	NINT IDNINT	1 1	R DP	I I
REAL(X)	Μετατρέπει το X σε πραγματικό.	- - SNGL	1 1 1	I R DP	R R R
SIGN(X,Y)	ABS(X) αν Y ≥ 0 -ABS(X) αν Y ≤ 0.	ISIGN SIGN DSIGN	2 2 2	I R DP	I R DP
SIN(X)	Ημίτονο του X (σε rad).	SIN DSIN	1 1	R DP	R DP
SQRT(X)	Τετραγωνική ρίζα του X	SQRT DSQRT	1 1	R DP	R DP
SINH(X)	Υπερβολικό ημίτονο του X.	SINH DSINH	1 1	R DP	R DP

TAN(X)	Εφαπτομένη του X (σε rad).	TAN DTAN	1 1	R DP	R DP
TANH(X)	Υπερβολική εφαπτομένη του X.	TANH DTANH	1 1	R DP	R DP