Πίνακες - Arrays

- Απλές μεταβλητές: αποθηκεύουν μόνο μια τιμή κάθε φορά.

  Σήμερα θα δούμε πως κάνουμε κάτι ανάλογο χρησιμοποιώντας πίνακες

Για ποιο λόγο Πίνακες?

- Τι περισσότερες φορές ένα σύνολο δεδομένων έχει ένα κοινό χαρακτηριστικό.

  Συνηθισμένα παραδείγματα: Οι συντεταγμένες ενός σημείου

  Οι συντελεστές ενός συστήματος εξισώσεων

- Χρήσιμο στις περιπτώσεις αυτές να δίνουμε ένα κοινό όνομα στο σύνολο όλων αυτών των δεδομένων

  Για τις συντεταγμένες ενός σημείου λέμε: $r = (x, y, z)$ το σύνολο r έχει 3 στοιχεία

- Μπορούμε να διακρίνουμε κάθε στοιχείο του συνόλου από τις διαφορετικές τιμές ενός ή περισσότερων δεικτών

  Τα στοιχεία του συνόλου των συντεταγμένων r είναι: $r(1) = x \quad r(2) = y \quad r(3) = z$

  Αντί για 3 απλές μεταβλητές x,y,z ορίζουμε 3 άλλες, r(3), που είναι πολύ πιο εύκολο να χρησιμοποιήσουμε $\begin{bmatrix} r_1 & r_2 & r_3 \end{bmatrix}$ Μονοδιάστατος

- ✓ Οι συντελεστές ενός συστήματος γραμμικών εξισώσεων μπορούν να παρασταθούν με α_{ik} όπου i=1,2,…,m και k=1,2,…,n

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

Δυσδιάστατος

Πίνακες

- Μπορούμε να έχουμε πίνακες περισσότερων διαστάσεων

  Στη Fortran μπορούμε μέχρι 7 διαστάσεις

- Πίνακες είναι μια συλλογή σχετιζόμενων μεταβλητών

  Ένα και μόνο όνομα χρησιμοποιείται για την αποθήκευση του πίνακα

  Χρήση δεικτών δίνει πρόσβαση στα επιμέρους στοιχεία του πίνακα

- Το πλήθος των δεικτών που χρειάζεται για να καθορίσουμε ένα στοιχείο του πίνακα ορίζει τη διάσταση του πίνακα

  Γραμμή ενός δυσδιάστατου πίνακα είναι το σύνολο των στοιχείων με κοινό i

  Στήλη ενός δυσδιάστατου πίνακα είναι το σύνολο των στοιχείων με κοινό k

Από τις σπουδαιότερες ιδιότητες ενός υπολογιστή είναι η ευκολία με την οποία επεξεργάζεται πίνακες με μεγάλο αριθμό στοιχείων.

Αυτή η ευκολία οφείλεται κυρίως στο ότι ο Η/Υ ξέρει πως να επεξεργάζεται τους δείκτες

Arrays

Παράδειγμα: Υπολογισμός μέσης ετήσιας θερμοκρασίας

```
      Program Temperature
      INTEGER I, N
      REAL  SUM, AVET
      REAL  T(365)
      OPEN(unit=10,file='temp.dat' ,status='old' )
      N = 0
5     N= N +1
      READ(10,*,end=30)T(N)
      IF (N .LT. 365) GOTO 5
      GOTO 31
30    Print *,' End-of-file while reading'
31    Close(10)
      SUM = 0.0
      I = 0
10    I = I +1
      SUM = SUM + T(I)

      IF (I -N) .LT. 0) GOTO 10

      AVET = SUM/N
      PRINT *,' Mesi thermokrasia =',AVET
      END
```

Δείκτης από I =1 ως 365 (ημέρες του έτους)

Το ίδιο παράδειγμα αλλά με DO loop

```
Program Temperature
INTEGER I
REAL  SUM, AVET
REAL  T(365)

SUM = 0.0
DO  I =1, 365
      SUM = SUM + T(I)
END DO
AVET = SUM/365.0
PRINT *,' Mesi thermokrasia =',AVET
END
```

Fortran Πίνακες - Arrays

- Οι πίνακες στη Fortran έχουν ένα συγκεκριμένο τύπο όπως οι απλές μεταβλητές
  Έχουμε επομένως: REAL INTEGER DOUBLE PRECISION CHARACTER LOGICAL

- Οι πίνακες **πρέπει** να ορισθούν και **πρέπει** να δηλωθεί η διάστασή τους
  Δηλαδή το μέγεθος ή ο αριθμός των στοιχείων ενός πίνακα πρέπει να ορισθεί στην αρχή του προγράμματος

  - Για παράδειγμα:

```
REAL      TEMPERATURE(365), RAIN(12)
INTEGER   FOG(30)
```

Εδώ δηλώνουμε 3 μονοδιάστατους πίνακες με 365, 12 και 30 στοιχεία

- Ένας πίνακας και το μέγεθός του μπορούν να ορισθούν και με την εντολή DIMENSION (πάλι στην αρχή του προγράμματος)

```
DIMENSION TEMPERATURE(365), RAIN(12), IFOG(30)
```

Στην περίπτωση αυτή χρησιμοποιείται η σύμβαση των ονομάτων όπως και στις απλές μεταβλητές (ότι αρχίζει από I,J,K,L,M,N είναι ακέραια μεταβλητή π.χ. IFOG)

Η εντολή DIMENSION είναι μή εκτελέσιμη εντολή

Ο πρώτος τρόπος ορισμού πίνακα είναι ο προτιμητέος

Fortran Arrays

- Τα στοιχεία (το περιεχόμενό του δηλαδή) ενός πίνακα **πρέπει** να είναι του ίδιου τύπου με τον ορισμό του πίνακα

  Αν ο πίνακας ορίσθηκε σαν REAL τότε όλα τα στοιχεία πρέπει να'ναι REAL

- Τα στοιχεία ενός πίνακα χρησιμοποιούνται με το όνομα του πίνακα ακολουθούμενο από μια παρένθεση όπου υποδηλώνουμε τη θέση του στοιχείου στο πίνακα είτε με νούμερο είτε με κάποιο δείκτη (ή δείκτες),
  - Πάντοτε INTEGER μεταβλητή

  Για παράδειγμα: FOG(3) δηλώνει το 3ο στοιχείο του πίνακα FOG

  FOG(I) δηλώνει το I-οστό στοιχείο του πίνακα

  με το I να έχει ορισθεί από πριν σαν INTEGER και να πέρνει τιμές από I = 1…N (N η διάσταση του πίνακα)

Για ένα **δυσδιάστατο** πίνακα θα είχαμε:

TEMP(3,2) δηλαδή το στοιχείο της 3ης γραμμής και 2ης στήλης

TEMP(I,J) δηλαδή το στοιχείο της I^{th} γραμμής και J^{th} στήλης

Αυτό σημαίνει ότι αν κατά τη διάρκεια της εκτέλεσης ενός προγράμματος οι μεταβλητές I,J έχουν τιμές 3,2 τότε αν σε μιά πράξη εμφανίζεται η μεταβλητή A(I,J) για τον υπολογιστή αυτό ισοδυναμεί με τη μεταβλητή A(3,2).

Για **τρισδιάστατο** θα είχαμε π.χ. N(I,J,K)

Arrays

- Μέχρι τώρα όλοι οι δείκτες ήταν ακέραιες σταθερές ή ακέραιες μεταβλητές
- Μπορούμε να χρησιμοποιήσουμε και γενικότερες εκφράσεις π.χ. I+J, I-J, I*J, I*J-K

ΠΡΟΣΟΧΗ:

- Σα δείκτης μπορεί να χρησιμοποιηθεί και μια ακέραιη μεταβλητή με δείκτη

  π.χ. μπορούμε να γράψουμε ARS(INDX(I))

- Ο δείκτης πρέπει να έχει πάντοτε τιμές μεταξύ της ελάχιστης και της μέγιστης δυνατής τιμής που έχει καθοριστεί κατά τον ορισμό του πίνακα.

  Ο Η/Υ δεν εξετάζει αν αυτό ισχύει και αφήνεται στον προγραμματιστή

  Οι συνέπειες μπορεί να είναι πολύ άσχημες αν ξεπεράσουμε (προς τα κάτω, <1 ή προς τα πάνω >N) τις θέσεις του πίνακα μιά και αρχίζουμε να διαβάζουμε άλλες περιοχές της μνήμης του υπολογιστή με απρόβλεπτα αποτελέσματα.

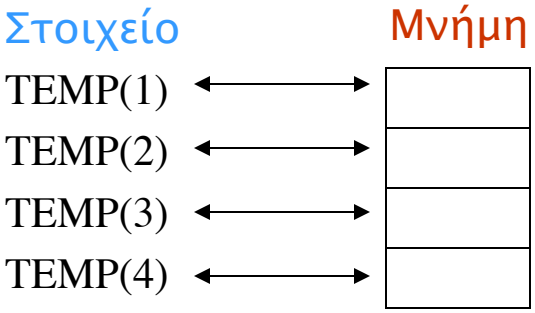
- Διαβάζοντας κάποιο στοιχείο του πίνακα που δεν υπάρχει, δηλαδή έξω από τις διαστάσεις του πίνακα (μη συγχέετε το δεν υπάρχει με το στοιχείο να είναι 0) θα πάρετε σφάλμα εκτέλεσης (execution fault)

  "Bus error Core Dumped"

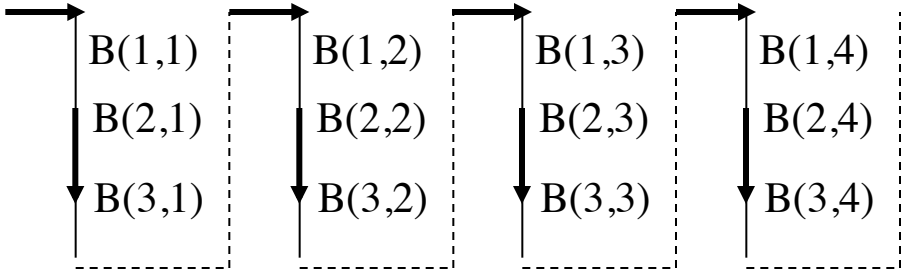
Αποθήκευση στοιχείων ενός Array στη μνήμη

- Στη FORTRAN τα στοιχεία ενός πίνακα αποθηκεύονται στη μνήμη του υπολογιστή κατά στήλες σύμφωνα με το παρακάτω τρόπο:

REAL TEMP(4)



REAL B(3,4)



- Τα στοιχεία του πίνακα αποθηκεύονται στην μνήμη σειριακά ξεκινώντας από το πρώτο και καταλήγοντας στο τελευταίο
- Σε δυσδιάστατους πίνακες τα στοιχεία αποθηκεύονται κατά στήλες

  Δηλαδή πρώτα αποθηκεύονται στοιχεία της πρώτης στήλης από την πρώτη ως την τελευταία γραμμή και μετά η επόμενη στήλη κ.ο.κ.

  Η αποθήκευση είναι και πάλι σειριακή

  Αυτό πρέπει να το λάβουμε υπ' όψη ώστε να δίνουμε τα στοιχεία του πίνακα με τη σωστή σειρά

  Το ίδιο ισχύει και για την έξοδο.

  Δηλαδή θα πρέπει να τυπώνουμε τα στοιχεία με την σωστή σειρά

Arrays – Περίληψη Κανόνων και Ιδιοτήτων

- Πρέπει να ορίζονται στην αρχή του προγράμματος μαζί με τις υπόλοιπες μεταβλητές

π.χ.
```
Program bigarray
REAL MYDATA(55,60,3), X(55), Y(60), Z(3)
```

- Το εύρος των δεικτών μπορεί να είναι οποιοδήποτε εύρος σειριακών ακεραίων

```
      REAL WDAT(-100:-50)
      DO 100 I = -100, -50
         WDAT(I) = I
100   CONTINUE
```

- Κάθε στοιχείο του πίνακα μπορεί να ληφθεί απευθείας από τη μνήμη χρησιμοποιώντας μια μεταβλητή με δείκτες

π.χ.
```
C Write out the 1st element of the MYDATA array as a check:
   WRITE(6,*) MYDATA(1,1,1)
```

- Ο δείκτης πρέπει να είναι ακέραιος ή ακέραια μεταβλητή ή έκφραση μέσα στα όρια του ορισμού του πίνακα.

```
Y(16.5) = 2*F              ! Πραγματικοί δείκτες δεν επιτρέπονται
MYDATA(1,1,4) = 10.8       ! Ο 3ος δείκτης (4) είναι εκτός περιοχής (από 1ο παράδειγμα)
P=WDAT(1)*3.14             ! WDAT(1) είναι εκτός περιοχής (από το 2ο παράδειγμα).
                           ! Το εύρος τιμών των δεικτών είναι από -100 ως -50
```

Παραδείγματα

- Έστω το παρακάτω κομμάτι κώδικα:

```
      DO 100 IROW = 1, 3
         DO 200 JCOL = 1, 2
            READ(5,*) MAT2D(IROW,JCOL)
200      CONTINUE
100   CONTINUE
```

1	2
3	4
5	6

Τα στοιχεία που δίνονται είναι: 1, 2, 3, 4, 5, 6

Σε ποιες θέσεις της μνήμης θα γραφούν?

- Σε ποιες θέσεις θα γραφούν αν διαβαστούν με την ακόλουθη εντολή?

```
READ(5,*)MAT2D
```

1	4
2	5
3	6

Ένας γρήγορος τρόπος για να το δείτε:

```
program test
integer mat(3,2)
integer iraw,jcol
do iraw=1, 3
   do jcol = 1, 2
      read(5,*) mat(iraw,jcol)
   enddo
enddo
print *, mat    ! θα τυπώσει : 1,3,5,2,4,6 όπως είναι στη μνήμη
read(5,*) mat
print *, mat    ! θα τυπώσει 1,2,3,4,5,6
end
```

Παραδείγματα

- Έστω ότι έχουμε 1000 πραγματικούς αριθμούς. Να γραφεί ένα πρόγραμμα που να τυπώνει το ελάχιστο και μέγιστο.

Λύση

Το πρόβλημα ουσιαστικά ανάγεται σε 2 μικρότερα προβλήματα

Έστω A_i οι αριθμοί αυτοί, τότε πρέπει να κάνουμε τα ακόλουθα

(α) Να περαστούν οι N αριθμοί στη μνήμη (β) Να βρεθεί ο min και max

Το διάβασμα των τιμών A(I) που παριστάνουν τα νούμερα A_i γίνεται εύκολα με ένα READ και κάποια επανάληψη DO loop

Εύρεση του μικρότερου και μεγαλύτερου των A(I)

Σε μια θέση της μνήμης που αποθηκεύουμε τον AMIN γράφουμε αρχικά τον A(1)

Συγκρίνουμε τον AMIN με τον A(2) και γράφουμε τον A(2) στη θέση του AMIN αν AMIN>A(2). Αλλιώς προχωρούμε στη σύγκριση με τον A(3) κ.ο.κ.

Έτσι η τιμή του AMIN ισούται πάντα με τον μικρότερο από όλους τους A(I) που έχουμε οπότε βρήκαμε τον ελάχιστο

Παρόμοια βρίσκουμε τον μέγιστο των A(I)

Κάθε φορά που βρίσκεται ο AMIN ή AMAX κρατάμε την τιμή του δείκτη, I, της θέσης του στον πίνακα των τιμών σαν τιμή του IMIN ή IMAX

Επομένως μπορούμε να γνωρίζουμε και την θέση I για την οποία A(I) είναι ελάχιστο ή μέγιστο

Κώδικας – Με Βρόγχο – Οχι πολύ αποδοτικός

```
      PROGRAM MINMAX
      REAL A(1000)
      REAL AMIN, AMAX
      INTEGER I, IMIN, IMAX
      DO 10 I=1, 1000
         READ *, A(I)
10    CONTINUE
      AMIN =  A(1)
      AMAX =  A(1)
      IMIN  = 1
      IMAX  = 1
      DO 20 I = 1, 1000
         IF (AMIN.GT.A(I)) THEN
           AMIN = A(I)
           IMIN  = I
         ENDIF
         IF (AMAX .LT. A(I)) THEN
           AMAX = A(I)
           IMAX = I
         ENDIF
20    CONTINUE
      PRINT *, IMIN, AMIN, IMAX, AMAX
      END
```

Ο κώδικας αυτός δεν είναι ιδιαίτερα αποδοτικός μια και υπάρχουν πολλές εντολές και επίσης χρησιμοποιούμε ένα μεγάλο μέρος μνήμης (ο πίνακας) χωρίς ιδιαίτερο λόγο.

Κώδικας – Πιο αποδοτικός

```
Program MINMAXEFF
REAL A, AMIN, AMAX
INTEGER IMIN, IMAX
READ *, A
IMIN = 1
IMAX = 1
AMAX = A
AMIN = A
DO 10 I = 2, 1000
    READ *, A
    IF (A .GT. AMAX) THEN
        AMAX = A
        IMAX  = I
    ENDIF
    IF (A .LT. AMIN) THEN
        AMIN = A
        IMIN  = I
    ENDIF
10  CONTINUE
    PRINT *, IMIN, AMIN, IMAX, AMAX
    END
```

Χρησιμοποιούμε το 1ο αριθμό για να ορίσουμε το μέγιστο και το ελάχιστο

! Δεν χρησιμοποιούμε πίνακα

Παράδειγμα

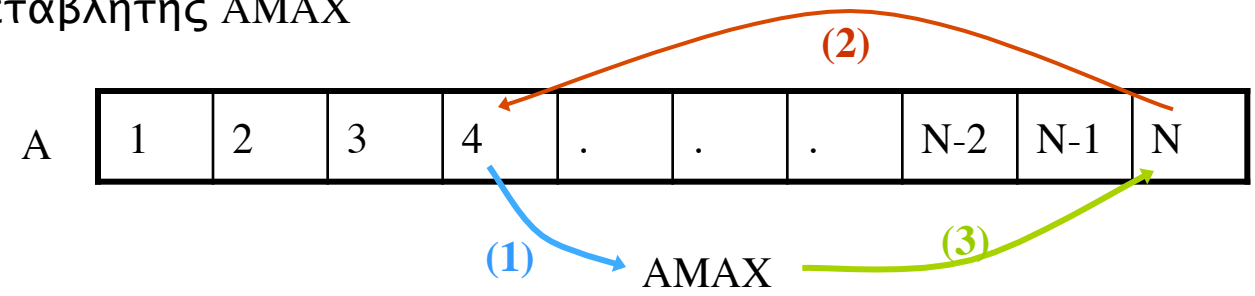
Μας δίνονται 1000 αριθμοί και θέλουμε να ταξινομηθούν κατ' αύξουσα σειρά και να τυπωθούν

Λύση

Υπάρχουν πολλοί τρόποι ταξινομήσεως N αριθμών

Αφού γραφούν στην μνήμη σα στοιχεία μονοδιάστατου πίνακα A(I) (I=1,2,...,1000) βρίσκουμε το μεγαλύτερο

Έστω A(IMAX) ο μέγιστος, τότε δίνουμε την τιμή του στην μεταβλητή AMAX και ανταλλάσουμε τη θέση του με το τελευταίο στοιχείο του πίνακα μέσω της μεταβλητής AMAX

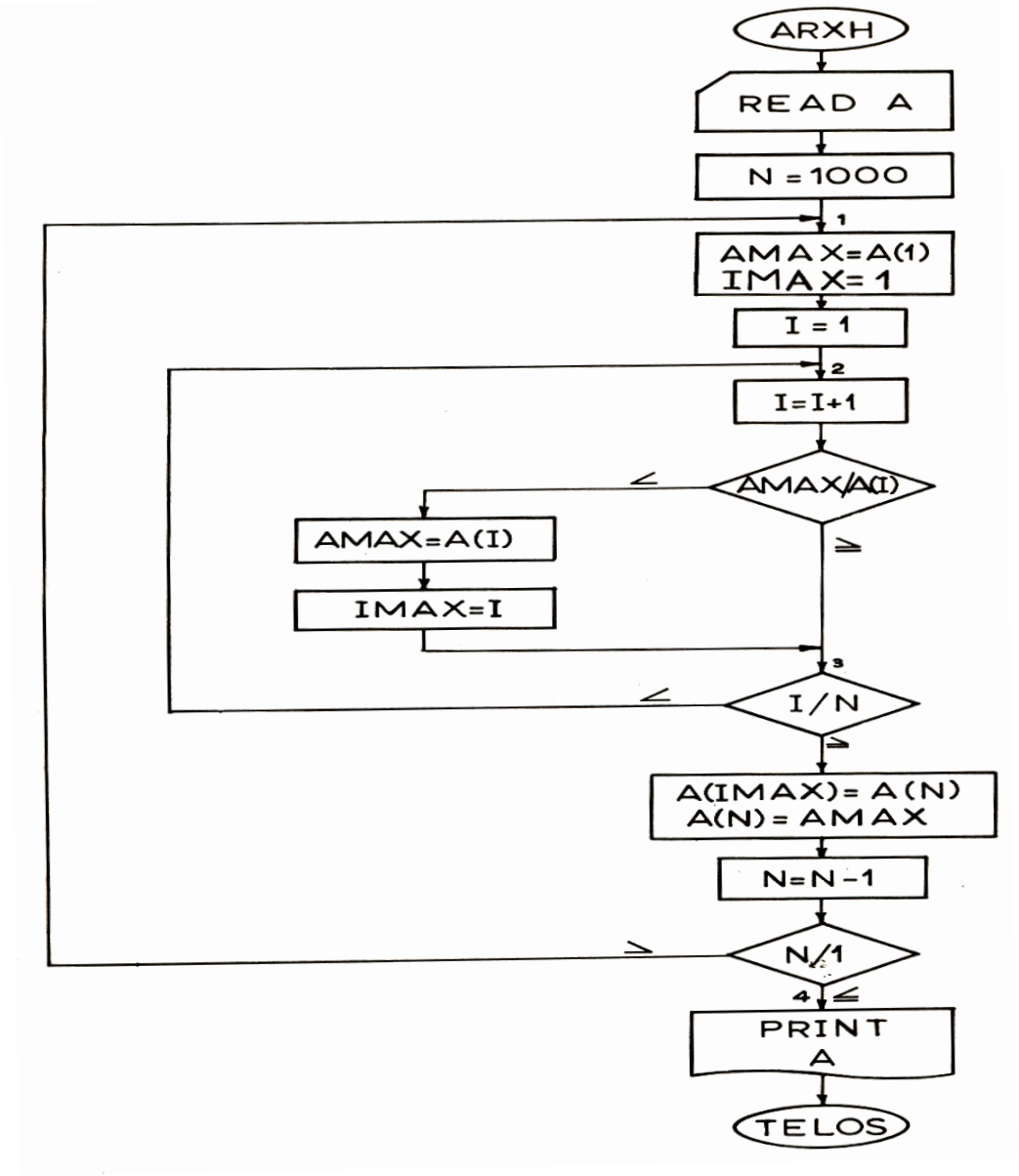


Πετυχαίνουμε έτσι να έχουμε το max από τους 1000 αριθμούς στη θέση A(1000).

Επαναλαμβάνουμε τη διαδικασία αυτή για τους υπόλοιπους N-1 αριθμούς, και το max που βρίσκουμε το βάζουμε στη θέση N-1.

Συνεχίζουμε την διαδικασία για τους N-2 κ.λ.π. μέχρι να μείνει ένας μόνο αριθμός στη θέση 1 που είναι και ο ελάχιστος όλων

Λογικό διάγραμμα της προηγούμενης ταξινόμησης



Κώδικας ταξινόμησης

```
      PROGRAM SORT
      REAL A(1000), AMAX
      INTEGER IMAX, I, N
      N = 1000
5     AMAX = A(1)
      IMAX  = 1
      DO 10  I = 2,  N                  ! Προσέξτε ότι ξεκινώ από το 2ο στοιχείο
         IF (AMAX.GE.A(I)) GOTO 10      ! Δεν είναι μεγαλύτερος από AMAX
         IMAX = I                       ! Βρίσκω το μέγιστο
         AMAX = A(I)
10    CONTINUE
      A(IMAX) = A(N)                    ! Βάζω στη θέση του A(IMAX) το τελευταίο στοιχείο
      A(N) = AMAX                       ! Στη θέση του Nth στοιχείου βάζω τον AMAX
      N = N - 1                         ! Ελαττώνω το N κατά 1 για να βρω τον μέγιστο
                                        ! από τα υπόλοιπα στοιχεία του πίνακα
      IF (N.GT.1) GOTO 5                ! Επανέλαβε τη διαδικασία μέχρι το N να γίνει 1
      PRINT *, A                        ! Θα τυπώσει όλο το πίνακα
      END
```

Εντολές I/O

- Για να γράψουμε ή να διαβάσουμε κάτι έχουμε χρησιμοποιήσει μέχρι τώρα:

READ *, A	READ(30,*) A	READ(5,*)A
PRINT *, A	WRITE(30,*)A	WRITE(6,*)A

Συνήθως θέλουμε να γράψουμε δεδομένα που είναι στοιχισμένα κατά στήλες ή θα θέλαμε να αποφύγουμε ορισμένους χαρακτήρες ή τέλος θα θέλαμε να γράψουμε/διαβάσουμε σε/από κάποιο αρχείο και όχι στην οθόνη ή το πληκτρολόγιο.

- Εκτός από την περίπτωση της πρώτης στήλης οι υπόλοιπες εντολές έχουν την ακόλουθη μορφή (παρόλο που δεν το εξηγήσαμε διεξοδικά)

```
READ (unit=, format=, options)  A,B,….
WRITE (unit=, format=, options) A,B,…
```

Δηλαδή READ/WRITE από/σε κάποια μονάδα (UNIT)

με κάποιο συγκεκριμένο τρόπο (FORMAT)

Χρησιμοποιώντας * απλά αποδεχόμαστε τον προκαθορισμένο τρόπο γραφής για το συγκεκριμένο τύπο μεταβλητής (θυμηθείτε τα πολλά ψηφία στους REAL)

- Στην Fortran η **UNIT=5 αντιστοιχεί στο πληκτρολόγιο** – READ(5,*) ενώ η **UNIT=6 αντιστοιχεί στην οθόνη** – WRITE(6,*).

Εντολές I/O

- Όταν οι διεργασίες εισόδου/εξόδου σχετίζονται με κάποιο αρχείο, πρέπει να συσχετίζεται το όνομα του αρχείο (FILENAME) με κάποια UNIT.
  - Μπορούμε να χρησιμοποιήσουμε οποιαδήποτε μονάδα εκτός από την μονάδα 5 και 6 που χρησιμοποιούνται από την FORTRAN.

- Ο συσχετισμός του FILENAME με το συγκεκριμένο UNIT πραγματοποιείται με την εντολή OPEN (FILE = '/home/fotis/test.txt')

  OPEN (UNIT=n, FILE='filename' , επιλογές..)

  Η εντολή αυτή συσχετίζει την UNIT=n με το FILE το οποίο αναφέρουμε

  Οποιαδήποτε READ ή WRITE εντολές που αναφέρονται στη UNIT=n θα γίνονται απ' ευθείας στο FILE= 'filename'

- Οι επιλογές που μπορεί ή όχι να ακολουθούν το όνομα του FILE, είναι:

  STATUS: (STATUS ='NEW' , 'OLD' , 'UKNOWN') (νέο, υπάρχει, δεν ξέρουμε)

  FORM: (FORM='FORMATTED') (data στο file είναι ή όχι κάποιας μορφής)

  ACCESS: (ACCESS='SEQUENTIAL' , 'READONLY')

- Όταν τελειώσουμε με την γραφή ή ανάγνωση δεδομένων από το αρχείο πρέπει να το κλείσουμε και να απελευθερώσουμε τη UNIT.

  Αυτό γίνεται με την εντολή CLOSE

  CLOSE(UNIT = n) Δεν είναι υποχρεωτική

Εντολή READ – επιλογή

- Υπάρχουν 3 επιλογές που είναι χρήσιμες όταν διαβάζουμε (READ) δεδομένα από ένα αρχείο

END = νούμερο εντολής

Καθορίζει μια εντολή στην οποία θα πάει το πρόγραμμα όταν φτάσει στο τέλος του αρχείου

ERR = νούμερο εντολής

Καθορίζει μια εντολή στην οποία θα πάει το πρόγραμμα όταν προκληθεί κάποιο λάθος σε I/O

(π.χ. λάθος είδος μεταβλητής, περιμένει INTEGER και διαβάζει REAL)

IOSTAT=n

Μια ακέραια μεταβλητή δίνεται στη μεταβλητή IOSTAT που δείχνει την επιτυχία ή μή της I/O εντολής

- Οι ακέραιες τιμές είναι:
  - 0 δεν υπήρξε πρόβλημα
  - < 0 φθάσαμε στο END-OF-FILE σημείο του αρχείου
  - > 0 υπήρξε ERROR κατά την ανάγνωση

Εντολή READ – παραδείγματα

Παράδειγμα 1

```
      OPEN(unit=10,file='test.dat',status='old')
      I = 1
100   I = I + 1
      Read(10, 30,end=20) A
      B = A**2
      GOTO 100
30    FORMAT(1x,F5.2)
20    CLOSE(10)
```

Παράδειγμα 2

```
      OPEN(unit=10,file='test.dat',status='old')
      I = 1
100   I = I + 1
      Read(10, 30, err=50, end=20) A
      B = A**2
      GOTO 100
30    FORMAT(1x,F5.2)
50    PRINT *, 'Lathos input'
20    CLOSE(10)
```

Παράδειγμα 3

```
OPEN(unit=10,file='test.dat',status='old')
Read(10,30,iostat) A

IF (iostat.eq.0) then
  B = A**2
ELSEIF (iostat .GT. 0)
  GOTO 50
ELSE
  GOTO 20
ENDIF
30    FORMAT(1x,F5.2)
50    PRINT *, 'Lathos input'
20    CLOSE(10)
```

Η εντολή FORMAT

- Η εντολή αυτή ενημερώνει τον compiler με ποιο τρόπο θέλουμε να διαβάσουμε ή να γραψουμε κάποια στοιχεία από/σε file ή από/στην οθόνη

- Αυτό γίνεται ώς εξής:

  READ (unit=5, form=10) A (ακόμα διαβάζουμε από πληκτρολόγιο – unit=5)

  WRITE (unit=6, form=10) A (ακόμα γράφουμε στην οθόνη – unit=6)

  Τώρα όμως πληροφορούμε τον compiler ότι αυτό θα γίνει βάσει της εντολής 10

- Τί είναι όμως η εντολή την οποία βάζουμε στην θέση της form?

  - Η εντολή αυτή λέγεται FORMAT

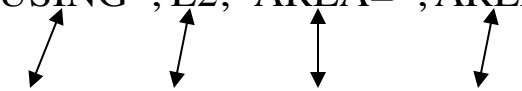
    Σχετίζεται πάντα με ένα αριθμό εντολής (10 στο παράδειγμα)
    ακολουθούμενη από την λέξη FORMAT και μετά μια περιγραφή για το ποια μέθοδος θα ακολουθεί για το διάβασμα/γράψιμο της μεταβλητής/μεταβλητών

  - Σύνταξη <αριθμός εντολής> FORMAT(c, ed1, ed2, 'κάποιο μύνημα', ed3,…)

    όπου:
    - αριθμός εντολής — αριθμός αναγνώρισης για τις εντολές READ/PRINT
    - ed1, ed2 — είναι κώδικες προδιαγραφής ξεχωριζόμενοι με κόμμα
    - text — μηνύματα που περιέχονται σε μονά '...'
    - c — χρησιμοποιείται μόνο για OUTPUT και δίνει <return>

FORMAT

```
WRITE(6,10) 'USING' , L2, 'AREA=' , AREA
```



```
10  FORMAT(1X, A5, 2X, I3, 4X, A6, 2X, F6.2)
```

1X, 2X, κλπ αντιπροσωπεύουν
πόσες κενές θέσεις παρεμβάλλονται

Υπάρχει μια προς μια αντιστοιχία μεταξύ των στοιχείων της λίστας μιας εντολής I/O και των περιγραφικών στοιχείων μιας εντολής FORMAT

Εξαίρεση ορισμένα στοιχεία περιγραφής όπως τα

- nX (αριθμός n κενών)
- / (αλλαγή σειράς)
- T (tabbing)

Η εντολή FORMAT ορίζεται μόνο μία φορά στο πρόγραμμα για κάθε αριθμό εντολής και μπορούμε να την έχουμε οπουδήποτε σε ένα πρόγραμμα ή υποπρόγραμμα

Μπορεί όμως να χρησιμοποιηθεί από πολλές I/O εντολές

αρκεί να έχουν το ίδιο πλήθος και τύπο στοιχείων

Οι κώδικες προδιαγραφής της εντολής FORMAT

- Οι κώδικες προδιαγραφής καθορίζουν τη μορφή των δεδομένων ή αποτελεσμάτων, πως δηλαδή θα γραφεί ή θα διαβαστεί μια μεταβλητή

  - Αποτελείται βασικά από ένα γράμμα και μια ή δύο ακέραιες θετικές σταθερές:

Iw: Καθορίζει το πεδίο ενός ακέραιου M. Ο w δηλώνει τα διαθέσιμα ψηφία

Αν ο M έχει λιγότερα από w ψηφία τοποθετείται στις w δεξιά θέσεις αφήνοντας κενές τις υπόλοιπες. Αν M<0 προτάσσεται το –

π.χ. M = 343 FORMAT(I3)

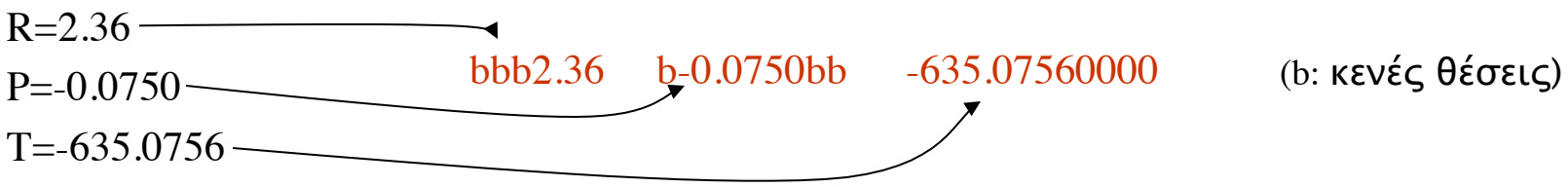
M = -2701 FORMAT(I5)

Fw.d: Καθορίζει το πεδίο ενός πραγματικού R σε δεκαδική μορφή

Διατίθενται συνολικά w θέσεις στις οποίες περιλαμβάνονται η τελεία και το ±

Το d, όταν λαμβάνεται υπ' όψη καθορίζει το πλήθος των δεκαδικών ψηφίων που γράφονται στις d δεξιότερες θέσεις του πεδίου.

π.χ. Έστω η εντολή FORMAT(F7.2, F10.6, F13.8) και έχω τους αριθμούς



  - Ανάλογα με τη τιμή του d μπορούμε να αποκόψουμε κάποιο αριθμό δεκαδικών

π.χ. R=2.36 FORMAT(F3.1) -> R=2.4

Οι κώδικες προδιαγραφής της εντολής FORMAT – συν.

Ew.d: Καθορίζει το πεδίο ενός πραγματικού R σε εκθετική μορφή *rEi*

Στις w θέσεις που διατίθενται συνολικά, περιλαμβάνονται η τελεία, το πρόσημο, το γράμμα E και ο εκθέτης (αν υπάρχουν)

Όταν το d πέρνεται υπ' όψη καθορίζει το πλήθος των δεκαδικών ψηφίων του r

π.χ. M = 2370.653 FORMAT(E16.8) bbbb.23706530+04

M = -0.00734619 FORMAT(E14.7) bb-.7346100-02

Dw.d: Καθορίζει το πεδίο μιας μεταβλητής διπλής ακρίβειας

Ίδιο αποτέλεσμα με αυτό της Ew.d

Aw: Καθορίζει το πεδίο μιας μεταβλητής χαρακτήρων

Μια αλφαριθμητική μεταβλητή έχει σα τιμή ένα σύνολο χαρακτήρων

Το αποτέλεσμα του κωδικού αυτού εξαρτάται από το πλήθος g των χαρακτήρων που καταγράφονται στη μνήμη στη θέση καταγραφής

Αν w>g θα τυπωθούν w - g κενά και αμέσως μετά οι g χαρακτήρες

Αν g>w τότε θα τυπωθούν οι w αριστερότεροι χαρακτήρες από τους g στη μνήμη

nX: Καθορίζει τον αριθμό των οριζόντιων κενών

/ : Καθορίζει τον αριθμό των κατακόρυφων κενών

Tc: Καθορίζει τα tabs (όπου c θετικός ακέραιος που παραστάνει τον αριθμό στήλης

- Ο πρώτος χαρακτήρας κάθε γραμμής χρησιμοποιείται για control του κατακόρυφου διαχωρισμού γι' αυτό χρησιμοποιείται πάντα 1X

Παραδείγματα

- root1=123.523
  root2=789.158
  WRITE(6,100)root1, root2
  100 FORMAT(1x, ‘ Roots are:’ ,F5.1, 1x, F5.1)

1 κενό

output Roots are: 123.5 789.1

Επανάληψη

- WRITE(6,101)root1,root2
  101 FORMAT(1x, ‘Roots are:’ ,2F5.1)

output Roots are: 123.5 789.1

- Έστω NUM1 = 123, NUM2 = 456, NUM3 = 9
  WRITE(6,102) NUM1, NUM2, NUM3
  102 FORMAT(1x,I3,1x,I3,1x,I3)

output: b123b456bbbb9
123 456 9

- WRITE(6,103) NUM1,NUM2,NUM3
  103 FORMAT(1x,3(I2,1x))

output n*** *** ***

Οι αριθμοί είναι 3-ψηφιοι ενώ το Format τους έχει 2ψήφιους. ΣΦΑΛΜΑ

- Για REAL: NUMBER1=-123.5678 NUMBER2 = -23456.89

Format Fw.d d=# πλήθος δεκαδικών ψηφίων

w-d >= 2 (υπολογίζοντας την υποδιαστολή και +/- πρόσημο)

Η μεταβλητή NUMBER1 χρειάζεται FORMAT F9.4

ενώ η μεταβλητή NUMBER2 χρειάζεται FORMAT F9.2

Gnuplot – The shortest of all summaries

gnuplot>plot sin(x)	σχεδιασμός ημίτονου
gnuplot>help plot	πληροφορίες πώς να χρησιμοποιήσετε την εντολή plot
gnuplot>set term postscript	για να γράψετε τη γραφική παράσταση σε γλώσσα postscript που μπορεί να εκτυπωθεί
gnuplot>set out 'myplot.ps'	για να σώσετε τη γραφική παράσταση στο αρχείο myplot.ps
gnuplot>plot cos(x)	η γραφική του συνημίτονου θα σωθεί στο αρχείο myplot.ps
gnuplot>set term X11	επιστροφή στην γραφική οθόνη
gnuplot>plot "*file*.dat" u 1:2	γραφική των δεδομένων του αρχείου *file*.dat
gnuplot>exit	έξοδος από το γραφικό περιβάλλον