ΕΚΤΗ ΔΙΑΛΕΞΗ

Συνοπτικά για διάνυσμα

Δημιουργία διανύσματος με

Γνωστό πλήθος στοιχείων

```
DOUBLE PRECISION :: a(100), b(0:20)
```

«Άγνωστο» πλήθος στοιχείων

```
INTEGER :: n, m
DOUBLE PRECISION, ALLOCATABLE :: a(:), b(:)
READ *, n, m
ALLOCATE(a(n))
ALLOCATE(b(-m:m))
```

Μπορούμε να εξαγάγουμε ως νέο διάνυσμα ένα τμήμα διανύσματος a. Το

a(αρχική θέση : τελική θέση : βήμα μεταβολής)

είναι διάνυσμα με τα στοιχεία του a στις θέσεις «αρχική θέση», «αρχική θέση» + «βήμα μεταβολής», «αρχική θέση» + 2*«βήμα μεταβολής», κλπ. έως το στοιχείο που δεν ξεπερνά την «τελική θέση».
Αν παραλείπεται:

- το βήμα μεταβολής, υπονοείται το 1.
- η αρχική ή η τελική θέση υπονοείται το πρώτο και το τελευταίο στοιχείο αντίστοιχα.

Το βήμα είναι θετικό ή αρνητικό.

Αν `INTEGER :: a(-10:20)` τότε

- το `a(-10:10)` είναι νέο διάνυσμα με στοιχεία τα στοιχεία του `a` μεταξύ των θέσεων -10 έως και 10.
- το `a(1:)` είναι νέο διάνυσμα με στοιχεία τα στοιχεία του `a` από τη θέση 1 έως και την τελευταία.
- το `a(:5)` είναι νέο διάνυσμα με στοιχεία τα στοιχεία του `a` από τη θέση -10 έως και τη θέση 5.
- το `a(1:11:2)` είναι νέο διάνυσμα με στοιχεία τα στοιχεία του `a` στις θέσεις 1, 3, 5, 7, 9, 11.
- το `a(6:2:-2)` είναι νέο διάνυσμα με στοιχεία τα στοιχεία του `a` στις θέσεις 6, 4, 2.

Έστω ότι έχουμε δεδομένα που οργανώνονται σε γραμμές και στήλες. Π.χ. έχω τις ακόλουθες πειραματικές μετρήσεις, οργανωμένες σε ημέρες και ώρες:

	00:00	06:00	12:00	18:00
Κυριακή	1.6	2.3	3.6	1.4
Δευτέρα	1.2	2.1	3.2	1.5
⋮	⋮	⋮	⋮	⋮
Σάββατο	1.4	2.5	3.4	2.3

- Πώς θα αποθηκεύσω σε πρόγραμμα τις 28 τιμές;
- Μπορώ να χρησιμοποιήσω διάνυσμα αλλά είναι δύσχρηστο. Υπάρχει κάτι πιο βολικό;

Πίνακες

Δήλωση

Αν θέλουμε να οργανώσουμε ποσότητες ίδιου τύπου σε γραμμές και στήλες, χρησιμοποιούμε πίνακα:

Πίνακας γνωστών διαστάσεων

τύπος :: όνομα(πλήθος_γραμμών, πλήθος_στηλών)

Τα πλήθη πρέπει να είναι σταθερές ακέραιες τιμές, ακέραιες εκφράσεις, κλπ. που να μπορούν να υπολογιστούν όταν γράφουμε το πρόγραμμα.

Πίνακες

Δήλωση

Αν θέλουμε να οργανώσουμε ποσότητες ίδιου τύπου σε γραμμές και στήλες, χρησιμοποιούμε πίνακα:

Πίνακας γνωστών διαστάσεων

τύπος :: όνομα(πλήθος_γραμμών, πλήθος_στηλών)

Τα πλήθη πρέπει να είναι σταθερές ακέραιες τιμές, ακέραιες εκφράσεις, κλπ. που να μπορούν να υπολογιστούν όταν γράφουμε το πρόγραμμα.

Πίνακας «άγνωστων» διαστάσεων

τύπος, ***ALLOCATABLE*** *:: onoma(: , :)*

...

ALLOCATE*(onoma(πλήθος_γραμμών, πλήθος_στηλών))*

Πίνακες

Δήλωση

Αν θέλουμε να οργανώσουμε ποσότητες ίδιου τύπου σε γραμμές και στήλες, χρησιμοποιούμε πίνακα:

Πίνακας γνωστών διαστάσεων

τύπος :: όνομα(πλήθος_γραμμών, πλήθος_στηλών)

Τα πλήθη πρέπει να είναι σταθερές ακέραιες τιμές, ακέραιες εκφράσεις, κλπ. που να μπορούν να υπολογιστούν όταν γράφουμε το πρόγραμμα.

Πίνακας «άγνωστων» διαστάσεων

τύπος, ***ALLOCATABLE*** *:: onoma(: , :)*

...

ALLOCATE*(onoma(πλήθος_γραμμών, πλήθος_στηλών))*

Παράδειγμα

```
DOUBLE PRECISION :: a(10,20)
INTEGER :: m,n
DOUBLE PRECISION, ALLOCATABLE :: b(:,:)

READ *, m,n
ALLOCATE(b(m,n))
```

Χειρισμός Πινάκων

- Για να έχουμε πρόσβαση σε ένα στοιχείο του πίνακα, πρέπει να προσδιορίσουμε γραμμή και στήλη στις οποίες βρίσκεται, μέσα σε παρενθέσεις. Έτσι, π.χ., αν το a είναι διδιάστατος πίνακας 10×20, το στοιχείο στη γραμμή 3 και στήλη 12 είναι το a(3,12).
- Για να διατρέξουμε κάποια ή όλα τα στοιχεία ενός πίνακα χρειαζόμαστε ΔΥΟ εντολές επανάληψης **DO**, τη μία μέσα στην άλλη:

```
INTEGER :: a(10,20), i, j

DO i=1,10
   DO j = 1,20
     READ *, a(i,j)  ! τα στοιχεία διαβάζονται κατά γραμμές
   END DO
END DO
```

Ενσωματωμένες συναρτήσεις για πίνακες (1/2)

Αν a είναι πίνακας, η συνάρτηση

SIZE(a)	υπολογίζει το πλήθος των στοιχείων του (γραμμές×στήλες).
SIZE(a,1)	υπολογίζει το πλήθος των γραμμών.
SIZE(a,2)	υπολογίζει το πλήθος των στηλών.
LBOUND(a,1)	υπολογίζει το κάτω όριο των γραμμών.
LBOUND(a,2)	υπολογίζει το κάτω όριο των στηλών.
UBOUND(a,1)	υπολογίζει το πάνω όριο των γραμμών.
UBOUND(a,2)	υπολογίζει το πάνω όριο των στηλών.

Ενσωματωμένες συναρτήσεις για πίνακες (2/2)

Αν a είναι πίνακας, η συνάρτηση

SUM(a)	υπολογίζει το άθροισμα των στοιχείων του.
PRODUCT(a)	υπολογίζει το γινόμενο των στοιχείων του.
MAXVAL(a)	υπολογίζει την *τιμή* του μεγαλύτερου στοιχείου του.
MINVAL(a)	υπολογίζει την *τιμή* του μικρότερου στοιχείου του.
MAXLOC(a)	υπολογίζει τη *θέση* του μεγαλύτερου στοιχείου του (σε διάνυσμα δύο θέσεων).
MINLOC(a)	υπολογίζει τη *θέση* του μικρότερου στοιχείου του (σε διάνυσμα δύο θέσεων).

Προσέξτε ότι οι **`MAXLOC`**/**`MINLOC`** επιστρέφουν διάνυσμα δύο θέσεων: στην πρώτη αποθηκεύεται η γραμμή και στη δεύτερη η στήλη.

Μπορούμε να εξαγάγουμε ως νέο διάνυσμα ή πίνακα ένα τμήμα του πίνακα a. Η γενική μορφή είναι

> *a(αρχική γραμμή: τελική γραμμή : βήμα μεταβολής γραμμών, αρχική στήλη: τελική στήλη : βήμα μεταβολής στηλών)*

Παραδείγματα

Αν το a είναι πίνακας 20×30, τότε

- το `a(1:10,1:10)` είναι πίνακας 10×10 με τα στοιχεία του a στην πάνω αριστερή γωνία.
- το `a(3,:)` είναι η τρίτη γραμμή ως διάνυσμα.
- το `a(:,5)` είναι η πέμπτη στήλη ως διάνυσμα.

Παραδείγματα χρήσης πινάκων

```
INTEGER :: a(10,10), i, j

! ανάγνωση κατά στήλες
DO j = 1,SIZE(a,2)
   DO i=1,SIZE(a,1)
     READ *, a(i,j)
   END DO
END DO

! εκτύπωση κατά γραμμές
DO i=1,SIZE(a,1)
   DO j = 1,SIZE(a,2)
     PRINT *, a(i,j)
   END DO
END DO

! εκτύπωση της πέμπτης γραμμής ανάποδα (από το τέλος προς την αρχή)
PRINT *, a(5,10:1:-1)
```

Αρχεία

Εναλλακτικές εντολές για ανάγνωση/εγγραφή

Ανάγνωση από το πληκτρολόγιο και εγγραφή στην οθόνη γίνεται με τις εντολές

```
READ *, ...
PRINT *, ...
```

Εναλλακτικές μορφές είναι οι

```
READ (*,*) ...
WRITE (*,*) ...
```

Οι τελευταίες μπορούν να επεκταθούν για ανάγνωση και εγγραφή σε αρχείο.

Σύνδεση αρχείου σε πρόγραμμα (1/3)

Για να χρησιμοποιήσουμε αρχείο για ανάγνωση/εγγραφή, πρέπει να εκτελέσουμε την εντολή

```
OPEN (unit = ..., file = ..., action = ..., status = ...)
```

όπου

unit = ... ένας μη αρνητικός ακέραιος αριθμός ή ακέραια σταθερή ποσότητα (**PARAMETER**). Προτεινόμενες τιμές 10–99.

file = ... το όνομα του αρχείου σε εισαγωγικά.

action = ... `"read"` ή `"write"` αν θέλουμε να χρησιμοποιήσουμε το αρχείο για ανάγνωση ή εγγραφή.

status = ... `"old"` ή `"new"` ή `"replace"`.

- `"old"` το αρχείο υπάρχει ήδη.
- `"new"` το αρχείο δεν υπάρχει και θα δημιουργηθεί.
- `"replace"` αν υπάρχει το αρχείο θα σβηστεί και θα δημιουργηθεί, αν δεν υπάρχει θα δημιουργηθεί.

Σύνδεση αρχείου σε πρόγραμμα (2/3)

Παρατηρήσεις

- Η εκτέλεση της εντολής **OPEN** αντιστοιχίζει ένα αρχείο με συγκεκριμένο αριθμό unit.

Σύνδεση αρχείου σε πρόγραμμα (2/3)

Παρατηρήσεις

- Η εκτέλεση της εντολής **OPEN** αντιστοιχίζει ένα αρχείο με συγκεκριμένο αριθμό unit.
- Η σειρά των ορισμάτων (**unit**=..., **file**=..., **action**=..., **status**=...) μπορεί να είναι οποιαδήποτε.

Σύνδεση αρχείου σε πρόγραμμα (2/3)

Παρατηρήσεις

- Η εκτέλεση της εντολής **OPEN** αντιστοιχίζει ένα αρχείο με συγκεκριμένο αριθμό unit.
- Η σειρά των ορισμάτων (**unit**=..., **file**=..., **action**=..., **status**=...) μπορεί να είναι οποιαδήποτε.
- Το **unit**=, αν είναι πρώτο, μπορεί να παραληφθεί (όχι όμως ο ακέραιος):

```
OPEN (ακέραιος, file=..., action=..., status=...)
```

Σύνδεση αρχείου σε πρόγραμμα (2/3)

Παρατηρήσεις

- Η εκτέλεση της εντολής **OPEN** αντιστοιχίζει ένα αρχείο με συγκεκριμένο αριθμό unit.
- Η σειρά των ορισμάτων (**unit**=..., **file**=..., **action**=..., **status**=...) μπορεί να είναι οποιαδήποτε.
- Το **unit**=, αν είναι πρώτο, μπορεί να παραληφθεί (όχι όμως ο ακέραιος):

```
OPEN (ακέραιος, file=..., action=..., status=...)
```

- Τα **action** = ..., **status** = ... μπορούν να παραληφθούν (μην το κάνετε!). Τα μόνα απαραίτητα είναι το unit και το όνομα αρχείου.

Σύνδεση αρχείου σε πρόγραμμα (2/3)

Παρατηρήσεις

- Η εκτέλεση της εντολής **OPEN** αντιστοιχίζει ένα αρχείο με συγκεκριμένο αριθμό unit.
- Η σειρά των ορισμάτων (**unit**=..., **file**=..., **action**=..., **status**=...) μπορεί να είναι οποιαδήποτε.
- Το **unit**=, αν είναι πρώτο, μπορεί να παραληφθεί (όχι όμως ο ακέραιος):

```
OPEN (ακέραιος, file=..., action=..., status=...)
```

- Τα **action** = ..., **status** = ... μπορούν να παραληφθούν (μην το κάνετε!). Τα μόνα απαραίτητα είναι το unit και το όνομα αρχείου.
- Σε ένα πρόγραμμα, μπορούμε να έχουμε συνδεδεμένα ταυτόχρονα περισσότερα από ένα αρχεία (με διαφορετικούς αριθμούς).

Σύνδεση αρχείου σε πρόγραμμα (3/3)

Παραδείγματα

- Σύνδεση του αρχείου με όνομα `"data.txt"` για ανάγνωση, που (προφανώς) προϋπάρχει:

```
OPEN(unit=25, file="data.txt", action="read", status="old")
```

Σύνδεση αρχείου σε πρόγραμμα (3/3)

Παραδείγματα

- Σύνδεση του αρχείου με όνομα `"data.txt"` για ανάγνωση, που (προφανώς) προϋπάρχει:

```
OPEN(unit=25, file="data.txt", action="read", status="old")
```

- Σύνδεση του αρχείου με όνομα `"out.txt"` για εγγραφή:

```
OPEN(unit=59, file="out.txt", action="write", status="replace")
```

Ανάγνωση/Εγγραφή αρχείου (1/2)

Αφού εκτελεστεί η εντολή **OPEN**:

- Ανάγνωση μίας γραμμής αρχείου που αντιστοιχήθηκε στον αριθμό 25 και ανοίχθηκε για ανάγνωση:

  ```
  READ (25, *) ...
  ```

Ανάγνωση/Εγγραφή αρχείου (1/2)

Αφού εκτελεστεί η εντολή **OPEN**:

- Ανάγνωση μίας γραμμής αρχείου που αντιστοιχήθηκε στον αριθμό 25 και ανοίχθηκε για ανάγνωση:

  ```
  READ (25, *) ...
  ```

- Εγγραφή μίας γραμμής σε αρχείο που αντιστοιχήθηκε στον αριθμό 59 και ανοίχθηκε για εγγραφή:

  ```
  WRITE (59, *) ...
  ```

Κάθε **READ** ή **WRITE** προχωρά αυτόματα στην επόμενη γραμμή.

Ανάγνωση/Εγγραφή αρχείου (2/2)

Αν υπάρχουν περισσότερες μεταβλητές στην εντολή **READ** από τα δεδομένα στη γραμμή του αρχείου, η ανάγνωση συνεχίζεται στην επόμενη γραμμή (η οποία διαβάζεται ολόκληρη).

Παράδειγμα

Αρχείο με **unit** 25:

```
 1  2  3
 4  5  6
 7  8  9
10 11 12
```

Εντολές:

```
INTEGER :: a,b,c,d,e,f,g
READ (25,*) a,b,c,d
READ (25,*) e
READ (25,*) f,g
```

Τιμές: το a 1, το b 2, το c 3, το d 4, το e 7, το f 10, το g 11.

Αποσύνδεση αρχείου

Αφού ολοκληρώσουμε τη χρήση του αρχείου:

```
CLOSE (unit = ακέραιος)
```

ή πιο απλά, αφού το **unit=** μπορεί να παραληφθεί:

```
CLOSE (ακέραιος)
```

Αποσύνδεση αρχείου

Αφού ολοκληρώσουμε τη χρήση του αρχείου:

```
CLOSE (unit = ακέραιος)
```

ή πιο απλά, αφού το **unit=** μπορεί να παραληφθεί:

```
CLOSE (ακέραιος)
```

Παρατηρήσεις

Μετά την εκτέλεση της εντολής **CLOSE**

- δεν μπορούμε να διαβάσουμε/γράψουμε στο αρχείο,
- μπορούμε να χρησιμοποιήσουμε τον αριθμό unit για άλλο αρχείο.

Παράδειγμα ανάγνωσης αρχείου

Το αρχείο "data.txt" έχει 10 γραμμές με δύο ακέραιους αριθμούς σε κάθε γραμμή. Η αποθήκευση σε διανύσματα γίνεται με τον κώδικα

```
INTEGER :: a(10), b(10), i

OPEN (16, file="data.txt", action="read", status="old")

DO i = 1,SIZE(a)
   READ (16, *) a(i), b(i)
END DO

CLOSE (16)
```

Παράδειγμα εγγραφής σε αρχείο

Το διάνυσμα a περιέχει αριθμούς. Θέλουμε να γραφούν στο αρχείο "data.txt", με κάθε αριθμό σε ξεχωριστή γραμμή:

```
INTEGER :: i

OPEN (51, file = "data.txt", action = "write", status = "replace")

DO i = 1,SIZE(a)
   WRITE (51, *) a(i)
END DO

CLOSE (51)
```