ΠΕΜΠΤΗ ΔΙΑΛΕΞΗ

Άθροισμα τριών ποσοτήτων (1/2)

```
INTEGER :: s, a1, a2, a3
READ *, a1,a2,a3
! ... εντολές
s = a1 + a2 + a3
```

Πρόβλημα

Πώς γενικεύεται για π.χ. 300 ποσότητες;

Άθροισμα τριών ποσοτήτων (2/2)

Να το τροποποιήσω ώστε να χρησιμοποιήσω εντολή **DO**;

```
INTEGER :: s
INTEGER :: a1
INTEGER :: a2
INTEGER :: a3

READ *, a1
READ *, a2
READ *, a3
! ... εντολές
s = 0
s = s + a1
s = s + a2
s = s + a3
```

Άθροισμα τριών ποσοτήτων (2/2)

Να το τροποποιήσω ώστε να χρησιμοποιήσω εντολή **DO**;

```
INTEGER :: s
INTEGER :: a1
INTEGER :: a2
INTEGER :: a3

READ *, a1
READ *, a2
READ *, a3
! ... εντολές
s = 0
s = s + a1
s = s + a2
s = s + a3
```

ΛΑΘΟΣ →

```
INTEGER :: s, i
DO i = 1,3
   INTEGER :: ai
END DO

DO i = 1,3
   READ *, ai
END DO
! ... εντολές
s = 0
DO i = 1,3
   s = s + ai
END DO
```

Άθροισμα τριών ποσοτήτων (2/2)

Να το τροποποιήσω ώστε να χρησιμοποιήσω εντολή **DO**;

```
INTEGER :: s
INTEGER :: a1
INTEGER :: a2
INTEGER :: a3

READ *, a1
READ *, a2
READ *, a3
! ... εντολές
s = 0
s = s + a1
s = s + a2
s = s + a3
```

ΛΑΘΟΣ →

```
INTEGER :: s, i
DO i = 1,3
   INTEGER :: ai
END DO

DO i = 1,3
   READ *, ai
END DO
! ... εντολές
s = 0
DO i = 1,3
   s = s + ai
END DO
```

Πώς θα δηλώσω και θα χειριστώ πολλές μεταβλητές;

Διάνυσμα γνωστής διάστασης

Δήλωση

Δήλωση μεταβλητών ίδιου τύπου, που σχετίζονται μεταξύ τους, με γνωστό πλήθος:

τύπος :: όνομα(πλήθος)

Το πλήθος πρέπει να είναι

- σταθερός ακέραιος αριθμός, ή
- σταθερή ακέραια ποσότητα (**PARAMETER**), ή
- έκφραση με ακέραια τιμή, γνωστή όταν γράφουμε το πρόγραμμα.

Διάνυσμα γνωστής διάστασης

Δήλωση

Δήλωση μεταβλητών ίδιου τύπου, που σχετίζονται μεταξύ τους, με γνωστό πλήθος:

τύπος :: όνομα(πλήθος)

Το πλήθος πρέπει να είναι

- σταθερός ακέραιος αριθμός, ή
- σταθερή ακέραια ποσότητα (**PARAMETER**), ή
- έκφραση με ακέραια τιμή, γνωστή όταν γράφουμε το πρόγραμμα.

Παράδειγμα

```
INTEGER, PARAMETER :: n = 15
DOUBLE PRECISION :: a(10), b(n), c(2*n+1)
```

Στη δήλωση που είδαμε, η αρίθμηση των θέσεων ξεκινά από το 1. Έτσι, a(1) είναι η πρώτη, a(2) είναι η δεύτερη, ..., a(10) είναι η δέκατη μεταβλητή.

Παρατηρήσεις

- Ο δείκτης είναι ακέραιος, από το 1 έως το πλήθος των θέσεων.
- Άλλο είναι το `a(2)` και άλλο το `a2`.
- Άλλος είναι ο αριθμός της θέσης και άλλο η τιμή που αποθηκεύεται στη θέση:

1	2	3	4	5	6	7	8	9	10
42	71	78	12	93	36	25	97	31	44

Διάνυσμα γνωστής διάστασης

Παράδειγμα

Για να διατρέξουμε τα στοιχεία (όλα ή κάποια) ενός διανύσματος χρησιμοποιούμε μία εντολή **DO**:

```
PROGRAM example
   IMPLICIT NONE
   INTEGER :: i, a(100)

   DO i = 1, 100
      READ *, a(i)
      a(i) = a(i) + 5
   END DO

   DO i = 100,1,-1
      PRINT *, a(i)
   END DO
END PROGRAM example
```

Όταν χρειάζεται να εκτελεστεί το πρόγραμμα για να βρούμε (π.χ. με **READ** ή κάποιο υπολογισμό) πόσες θέσεις έχει ένα διάνυσμα, πώς το δηλώνουμε;

Όταν χρειάζεται να εκτελεστεί το πρόγραμμα για να βρούμε (π.χ. με **READ** ή κάποιο υπολογισμό) πόσες θέσεις έχει ένα διάνυσμα, πώς το δηλώνουμε;

Α΄ απόπειρα (ΛΑΘΟΣ)

```
INTEGER :: n
DOUBLE PRECISION :: b(n)
READ *, n
```

Διάνυσμα «άγνωστης» διάστασης

Εισαγωγή

Όταν χρειάζεται να εκτελεστεί το πρόγραμμα για να βρούμε (π.χ. με **READ** ή κάποιο υπολογισμό) πόσες θέσεις έχει ένα διάνυσμα, πώς το δηλώνουμε;

Α΄ απόπειρα (ΛΑΘΟΣ)

```
INTEGER :: n
DOUBLE PRECISION :: b(n)
READ *, n
```

Β΄ απόπειρα (ΛΑΘΟΣ)

```
INTEGER :: n
READ *, n
DOUBLE PRECISION :: b(n)
```

Όταν χρειάζεται να εκτελεστεί το πρόγραμμα για να βρούμε (π.χ. με **READ** ή κάποιο υπολογισμό) πόσες θέσεις έχει ένα διάνυσμα, πώς το δηλώνουμε;

Α΄ απόπειρα (ΛΑΘΟΣ)

```
INTEGER :: n
DOUBLE PRECISION :: b(n)
READ *, n
```

Β΄ απόπειρα (ΛΑΘΟΣ)

```
INTEGER :: n
READ *, n
DOUBLE PRECISION :: b(n)
```

Γ΄ απόπειρα (ΣΩΣΤΗ)

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

Διάνυσμα «άγνωστης» διάστασης

Δήλωση, δημιουργία, χρήση, καταστροφή

Δήλωση και δημιουργία

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

...

`ALLOCATE`(*onoma(πλήθος_στοιχείων)*)

Το πλήθος στοιχείων πρέπει να είναι ακέραιο.

Χρήση

Αφού δημιουργηθεί ένα διάνυσμα «άγνωστης» διάστασης, χρησιμοποιείται ακριβώς όπως ένα διάνυσμα γνωστής διάστασης.

Καταστροφή

Αφού ολοκληρώσουμε τη χρήση του διανύσματος, μπορούμε, αν θέλουμε, να ελευθερώσουμε ρητά τη μνήμη που δεσμεύτηκε κατά τη δημιουργία του, με την εντολή **`DEALLOCATE`**:

`DEALLOCATE`(*onoma*)

Εναλλακτικά, η εντολή αυτή εκτελείται αυτόματα στο τέλος του προγράμματος. Η απελευθέρωση της μνήμης είναι απαραίτητη αν θέλουμε να χρησιμοποιήσουμε το όνομα του διανύσματος για άλλο διάνυσμα.

Δημιουργία διανύσματος με συγκεκριμένα όρια

Κάποιες φορές θέλουμε να μην αρχίζει η αρίθμηση των θέσεων από το 1. Μπορούμε να δημιουργήσουμε τέτοιο διάνυσμα ως εξής:

τύπος :: όνομα(κάτω όριο : άνω όριο)

ή για διάνυσμα που έχει δηλωθεί ως **ALLOCATABLE**,

ALLOCATE*(όνομα(κάτω όριο : άνω όριο))*

Δημιουργία διανύσματος με συγκεκριμένα όρια

Κάποιες φορές θέλουμε να μην αρχίζει η αρίθμηση των θέσεων από το 1. Μπορούμε να δημιουργήσουμε τέτοιο διάνυσμα ως εξής:

τύπος :: όνομα(κάτω όριο : άνω όριο)

ή για διάνυσμα που έχει δηλωθεί ως **ALLOCATABLE**,

ALLOCATE*(όνομα(κάτω όριο : άνω όριο))*

Παράδειγμα

```
INTEGER :: a(-10:20), b(1:10), n
INTEGER, ALLOCATABLE :: c(:)
READ *, n
ALLOCATE(c(0:n-1))
```

Πρώτο στοιχείο του `a` το `a(-10)`.
Πρώτο στοιχείο του `c` το `c(0)`.

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))
```

Άθροισμα τριών ποσοτήτων: γενίκευση

Ας επανέλθουμε στο αρχικό μας πρόβλημα:

```
INTEGER :: s
INTEGER :: a1
INTEGER :: a2
INTEGER :: a3

READ *, a1
READ *, a2
READ *, a3

s = 0
s = s + a1
s = s + a2
s = s + a3
```

Άθροισμα τριών ποσοτήτων: γενίκευση

Ας επανέλθουμε στο αρχικό μας πρόβλημα:

```
INTEGER :: s
INTEGER :: a1
INTEGER :: a2
INTEGER :: a3

READ *, a1
READ *, a2
READ *, a3

s = 0
s = s + a1
s = s + a2
s = s + a3
```

ΣΩΣΤΟ $\longrightarrow$

```
INTEGER :: s, i
INTEGER :: a(3)

DO i = 1, 3
   READ *, a(i)
END DO

s = 0
DO i = 1, 3
   s = s + a(i)
END DO
```

Ενσωματωμένες συναρτήσεις για διανύσματα

Θέσεις (1/2)

Αν a είναι ένα διάνυσμα, η συνάρτηση

SIZE(a) υπολογίζει το πλήθος των στοιχείων του.

LBOUND(a,1) υπολογίζει το κάτω όριο των θέσεών του.

UBOUND(a,1) υπολογίζει το πάνω όριο των θέσεών του.

Ενσωματωμένες συναρτήσεις για διανύσματα

Θέσεις (1/2)

Αν a είναι ένα διάνυσμα, η συνάρτηση

SIZE(a) υπολογίζει το πλήθος των στοιχείων του.

LBOUND(a,1) υπολογίζει το κάτω όριο των θέσεών του.

UBOUND(a,1) υπολογίζει το πάνω όριο των θέσεών του.

Παράδειγμα

Αν δηλώθηκαν τα a,b με την εντολή

```
INTEGER :: a(-5:10), b(20)
```

τότε

- το **LBOUND**(a,1) είναι -5, το **UBOUND**(a,1) είναι 10, το **SIZE**(a) έχει τιμή 16 (γιατί;).
- το **LBOUND**(b,1) είναι 1, το **UBOUND**(b,1) είναι 20 και το **SIZE**(b) έχει τιμή 20.

Παράδειγμα

```
DOUBLE PRECISION :: b(-10:10), c(4)
INTEGER :: i

DO i = LBOUND(b,1), UBOUND(b,1)
   READ *, b(i)
END DO

c(1) = 2.1d0
c(2) = -5.4d0
c(3) = -3.1d0
c(4) = -7.1d0

DO i=1,SIZE(a)
   PRINT *, c(i)
END DO
```

Ενσωματωμένες συναρτήσεις για διανύσματα

άθροισμα/γινόμενο

Αν a είναι ένα διάνυσμα αριθμών, η συνάρτηση

SUM(a) υπολογίζει το άθροισμα των στοιχείων του.

PRODUCT(a) υπολογίζει το γινόμενο των στοιχείων του.

Ενσωματωμένες συναρτήσεις για διανύσματα

άθροισμα/γινόμενο

Αν a είναι ένα διάνυσμα αριθμών, η συνάρτηση

SUM(a) υπολογίζει το άθροισμα των στοιχείων του.

PRODUCT(a) υπολογίζει το γινόμενο των στοιχείων του.

Αν το a έχει κάτω όριο θέσεων το 1, οι παρακάτω κώδικες είναι ισοδύναμοι:

```
s = 0
DO i = 1, SIZE(a)
   s = s + a(i)
END DO
```

```
s = SUM(a)
```

```
p = 1
DO i = 1, SIZE(a)
   p = p * a(i)
END DO
```

```
p = PRODUCT(a)
```

Ενσωματωμένες συναρτήσεις για διανύσματα

εσωτερικό γινόμενο

Αν `a`, `b` είναι διανύσματα αριθμών με ίδιο πλήθος στοιχείων, η συνάρτηση **`DOT_PRODUCT`**`(a,b)` υπολογίζει το εσωτερικό γινόμενο των διανυσμάτων, δηλαδή το άθροισμα των γινομένων των αντίστοιχων στοιχείων:

$$\sum a_i * b_i \ .$$

Ενσωματωμένες συναρτήσεις για διανύσματα

εσωτερικό γινόμενο

Αν a, b είναι διανύσματα αριθμών με ίδιο πλήθος στοιχείων, η συνάρτηση **DOT_PRODUCT**(a,b) υπολογίζει το εσωτερικό γινόμενο των διανυσμάτων, δηλαδή το άθροισμα των γινομένων των αντίστοιχων στοιχείων:

$$\sum a_i * b_i \ .$$

Έστω ότι τα a, b έχουν κάτω όριο θέσεων το 1. Οι παρακάτω κώδικες είναι ισοδύναμοι:

```
s = 0
DO i = 1, SIZE(a)
   s = s + a(i) * b(i)
END DO
```

```
s = DOT_PRODUCT(a,b)
```

Ενσωματωμένες συναρτήσεις για διανύσματα

μέγιστο/ελάχιστο

Αν a είναι ένα διάνυσμα, η συνάρτηση

MAXVAL(a) υπολογίζει την *τιμή* του μεγαλύτερου στοιχείου του.

MINVAL(a) υπολογίζει την *τιμή* του μικρότερου στοιχείου του.

MAXLOC(a,1) υπολογίζει τη *θέση* του μεγαλύτερου στοιχείου του.

MINLOC(a,1) υπολογίζει τη *θέση* του μικρότερου στοιχείου του.

Ενσωματωμένες συναρτήσεις για διανύσματα

μέγιστο/ελάχιστο: Παράδειγμα 1

Έστω ότι δηλώθηκε το ακέραιο διάνυσμα a, με κάτω όριο θέσεων το 1, και του έχουμε δώσει τιμές. Οι παρακάτω κώδικες είναι ισοδύναμοι:

```
INTEGER :: m

INTEGER :: i
m = a(1)
DO i = 2,SIZE(a)
   IF (m > a(i)) m = a(i)
END DO

PRINT *, "Ελάχιστο ", m
```

```
INTEGER :: m

m = MINVAL(a)

PRINT *, "Ελάχιστο ", m
```

Ενσωματωμένες συναρτήσεις για διανύσματα

μέγιστο/ελάχιστο: Παράδειγμα 2

Έστω ότι δηλώθηκε το διάνυσμα a με κάτω όριο θέσεων το 1, στο οποίο έχουμε δώσει τιμές. Οι παρακάτω κώδικες είναι ισοδύναμοι:

```
INTEGER :: j

INTEGER :: i
j = 1
DO i = 2,SIZE(a)
   IF (a(j) > a(i)) j = i
END DO

PRINT *, "Θέση ", j
PRINT *, "Ελάχιστο ", a(j)
```

```
INTEGER :: j

j = MINLOC(a,1)

PRINT *, "Θέση ", j
PRINT *, "Ελάχιστο ", a(j)
```

Απλή ή σύνθετη λογική έκφραση στην οποία συμμετέχει (τουλάχιστον ένα) διάνυσμα έχει αποτέλεσμα ένα διάνυσμα λογικού τύπου (με τιμές `.TRUE.` */* `.FALSE.` *).*

Παράδειγμα

Αν `a` είναι διάνυσμα ακέραιων, το `a > 2` είναι διάνυσμα λογικού τύπου, με ίδιο πλήθος στοιχείων με το `a` και τιμές `.TRUE.` στις θέσεις που το `a` έχει στοιχεία μεγαλύτερα από το 2 και `.FALSE.` στις υπόλοιπες.

Ενσωματωμένες συναρτήσεις για διανύσματα

μετρήσεις (2/4)

Οι συναρτήσεις **ALL**(), **ANY**(), **COUNT**() δέχονται ως όρισμα ένα διάνυσμα λογικού τύπου και

- Η **ALL**() επιστρέφει .**TRUE**. αν όλα τα στοιχεία του ορίσματος είναι .**TRUE**. αλλιώς επιστρέφει .**FALSE**.
- Η **ANY**() επιστρέφει .**TRUE**. αν κάποιο στοιχείο του ορίσματος είναι .**TRUE**. αλλιώς επιστρέφει .**FALSE**.
- Η **COUNT**() μετρά το πλήθος των στοιχείων του ορίσματος που είναι .**TRUE**.

Π.χ. **ALL**(a>10), **ANY**(a>5 .**AND**. a<20), **COUNT**(a>0).

Ενσωματωμένες συναρτήσεις για διανύσματα

μετρήσεις (3/4)

Ισοδύναμος κώδικας για το `z = ANY(a>5 .AND. a<20)`, αν το διάνυσμα `a` έχει κάτω όριο θέσεων το 1:

```
LOGICAL :: z
INTEGER :: i
z = .FALSE.
DO i = 1, SIZE(a)
   IF (a(i) > 5 .AND. a(i) < 20) z = .TRUE.
END DO
```

Ισοδύναμος κώδικας για το `k = COUNT(a>0)`, αν το διάνυσμα `a` έχει κάτω όριο θέσεων το 1:

```
INTEGER :: k, i
k = 0
DO i = 1, SIZE(a)
   IF (a(i) > 0) k = k + 1
END DO
```

Εκχώρηση τιμής σε διάνυσμα ως σύνολο (1/3)

Το διάνυσμα a αποκτά τιμή συνολικά με

- εκχώρηση μίας τιμής:

  ```
  a = 5
  ```

  αντί για

  ```
  DO i = 1,SIZE(a)
     a(i) = 5
  END DO
  ```

  (αν το κάτω όριο θέσεων του a είναι 1).

Εκχώρηση τιμής σε διάνυσμα ως σύνολο (2/3)

Το διάνυσμα a αποκτά τιμή συνολικά με

- εντολή **READ**:

  ```
  READ *, a
  ```

  αντί για

  ```
  READ *, (a(i), i=1,SIZE(a))
  ```

  (αν το κάτω όριο θέσεων του a είναι 1).

Εκχώρηση τιμής σε διάνυσμα ως σύνολο (3/3)

Το διάνυσμα a αποκτά τιμή συνολικά με

- εκχώρηση άλλου διανύσματος με ίδια διάσταση:

  ```
  a = b
  ```

  αντί για

  ```
  DO i = 1,SIZE(a)
     a(i) = b(i)
  END DO
  ```

  (αν το κάτω όριο θέσεων των διανυσμάτων είναι 1).

- εκχώρηση τιμών μέσα σε (/ , /) :

  ```
  INTEGER :: a(3)
  a = (/ 3,7,8 /)
  ```

  αντί για

  ```
  a(1) = 3; a(2) = 7; a(3) = 8
  ```

Πράξεις διανυσμάτων κατά στοιχείο

Μπορούμε να κάνουμε πράξη (πρόσθεση, αφαίρεση, πολλαπλασιασμό, κλπ.) ανάμεσα σε δύο διανύσματα `a`, `b`, ίδιου πλήθους στοιχείων, *που έχουν ήδη τιμές*. Το αποτέλεσμα είναι διάνυσμα ίδιου πλήθους στοιχείων:

```
c = a + b
```

αντί για

```
DO i = 1,SIZE(a)
   c(i) = a(i) + b(i)
END DO
```

(αν το κάτω όριο θέσεων των `a`, `b`, `c` είναι 1).

Πράξεις διανυσμάτων κατά στοιχείο

Μπορούμε να κάνουμε πράξη (πρόσθεση, αφαίρεση, πολλαπλασιασμό, κλπ.) ανάμεσα σε δύο διανύσματα `a`, `b`, ίδιου πλήθους στοιχείων, *που έχουν ήδη τιμές*. Το αποτέλεσμα είναι διάνυσμα ίδιου πλήθους στοιχείων:

```
c = a + b
```

αντί για

```
DO i = 1,SIZE(a)
   c(i) = a(i) + b(i)
END DO
```

(αν το κάτω όριο θέσεων των `a`, `b`, `c` είναι 1).
Πράξη μπορεί να γίνει και μεταξύ αριθμού και διανύσματος: η πράξη γίνεται με κάθε στοιχείο (ή ισοδύναμα, ο αριθμός μετατρέπεται σε διάνυσμα κατάλληλης διάστασης και γίνεται πράξη διανυσμάτων).

Μαθηματικές συναρτήσεις με όρισμα διάνυσμα

Μια μαθηματική συνάρτηση (**ABS**( ), **SQRT**( ), κλπ.) με όρισμα ένα διάνυσμα εκτελεί τον αντίστοιχο υπολογισμό σε κάθε στοιχείο του διανύσματος και επιστρέφει διάνυσμα ίδιου πλήθους στοιχείων.

Παράδειγμα

Αν a, b πραγματικά διανύσματα ίδιας διάστασης, και το a έχει τιμή, μπορούμε να έχουμε

```
b = SIN(a)
```

αντί για

```
DO i = 1,SIZE(a)
   b(i) = SIN(a(i))
END DO
```

(αν το κάτω όριο θέσεων των a, b είναι 1).