Fortran και Αντικειμενοστραφής προγραμματισμός

www.corelab.ntua.gr/courses/fortran_naval/naval

Δδάσκοντες: Άρης Παγουρτζής (pagour@cs.ntua.gr)
(Επίκουρος Καθηγητής ΣΗΜΜΥ)
Δώρα Σούλιου (dsouliou@mail.ntua.gr)
(ΕΔΙΠ ΣΗΜΜΥ)

5η ενότητα

- ✓ Πίνακες
- ✓ Αναζήτηση
- ✓ Ταξινόμηση
- ✓ Πολυδιάστατοι πίνακες
- ✓ Προηγμένος χειρισμός πινάκων

Πίνακες (i)

- **Δομημένη μεταβλητή**: αποθηκεύει μια συλλογή από τιμές δεδομένων
- **Πίνακας** (array): δομημένη μεταβλητή που αποθηκεύει πολλές τιμές του ίδιου τύπου

```
INTEGER:: pinakas(100)
```

ή

```
INTEGER, DIMENSION(100)::pinakas
```

ορίζει έναν πίνακα εκατό ακεραίων, τα στοιχεία του οποίου είναι:

```
pinakas(1), pinakas(2), pinakas(3), ... pinakas(100)
```

και έχουν τύπο **ακέραιο**

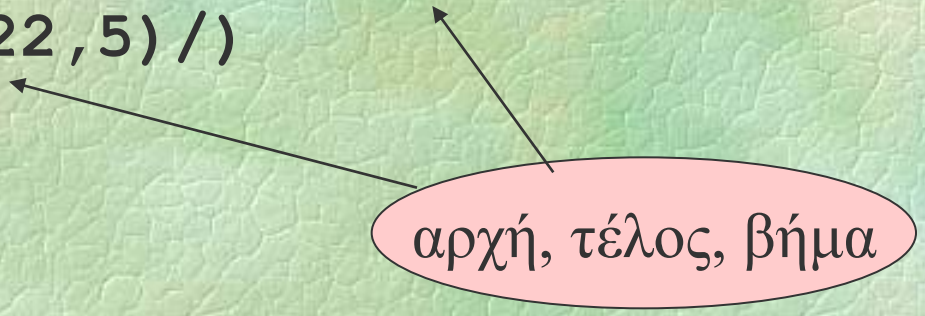
Πίνακες (ii)

Παραδείγματα

```
INTEGER:: a(12), Y(5)
REAL::    b(5)
CHARACTER(LEN=10):: NAMES(12)
```

Εντολές εκχώρησης

```
a(1)=4; a(2)=5; ... ; a(12)=8
b = (/2.3, 4.6, 5.8, 9.1, 10.0/)
READ(*,*) (NAMES(k), k=1,12,1)
Y = (/(i, i=1,22,5)/)
```



Πίνακες (iii)

- Διάβασμα πίνακα

```
REAL:: a(12)
READ(*,*) (a(k), k=1,12,1)    ή, ισοδύναμα:
DO i=1,12
  READ (*,*)  a(i)
END DO
```

- Εκτύπωση πίνακα

```
WRITE (*,*) a(:)    ή, ισοδύναμα:
DO i=1,12
  WRITE (*,*) a(i)
END DO
```

Πράξεις με πίνακες

- Απλές πράξεις, π.χ.

```
a(k) = a(k)+1;
z = a(1)+a(n);
IF (a(k) > a(k+1)) THEN ...
```

- Αρχικοποίηση (με μηδενικά)

```
DO i=1,10; a(i)=0; END DO
```

- Εύρεση ελάχιστου στοιχείου

```
x = a(1)
DO i=2,10
  IF (a(i) < x) THEN
    x = a(i)
  END IF
END DO
```

Γραμμική αναζήτηση (i)

- Πρόβλημα (αναζήτησης): δίνεται ένας πίνακας ακεραίων **a** και ζητείται να βρεθεί αν υπάρχει ο ακέραιος **x** στα στοιχεία του

```
IMPLICIT NONE
INTEGER:: x,a(10)
    άλλες δηλώσεις;
    διάβασμα του πίνακα a;
    διάβασμα του ακεραίου x;
    ψάξιμο στον πίνακα για τον x;
    παρουσίαση αποτελεσμάτων
END
```

Γραμμική αναζήτηση (ii)

- Μια δυνατή συγκεκριμενοποίηση

```
READ(*,*)  (a(i) i=1,10,1)
j=1
DO WHILE ((j<11) .AND. (a(j)/=x))
   j=j+1
END DO
IF (a(j)==x) THEN
  WRITE(*,*)  'Το βρήκα στη θέση ',  j
ELSE
  WRITE(*,*)  'Δεν το βρήκα'
END IF
```

- Στη χειρότερη περίπτωση θα ελεγχθούν όλα τα στοιχεία του πίνακα
- Απαιτούνται $a\,n + b$ βήματα $\Rightarrow$ γραμμικό κόστος (a, b σταθερές, n το μέγεθος του πίνακα)

Γραμμική αναζήτηση (iii)

- Εναλλακτική συγκεκριμενοποίηση #1

```
i=1; found=.FALSE.
DO WHILE ((i<11) .AND. (.NOT. found))
  IF a(i)==x THEN
   found=.TRUE.
  ELSE
   found =.FALSE.
  END IF
i=i+1
END DO
IF (found) THEN
   WRITE(*,*) 'Το βρήκα στη θέση ', (i-1)
ELSE
   WRITE (*,*) 'Δεν το βρήκα'
END IF
```

Γραμμική αναζήτηση (iv)

- Εναλλακτική συγκεκριμενοποίηση #2

```
i=1; found=.FALSE.
DO WHILE ((i<11) .AND. (.NOT. found))
  IF a(i)==x THEN
   found=.TRUE.
  END IF
  i=i+1
END DO
IF (found) THEN
   WRITE(*,*)'Το βρήκα στη θέση ', (i-1)
ELSE
   WRITE (*,*) 'Δεν το βρήκα'
END IF
```

Γραμμική αναζήτηση (v)

- Εναλλακτική συγκεκριμενοποίηση #3

```
i=1; found=.FALSE.
DO WHILE ((i<11) .AND. (.NOT. found))
  found = (a(i)==x)
  i=i+1
END DO
IF (found) THEN
   WRITE(*,*) 'Το βρήκα στη θέση ', (i-1)
ELSE
   WRITE(*,*) 'Δεν το βρήκα'
END IF
```

Δυαδική αναζήτηση (i)

- Προϋπόθεση: ο πίνακας να είναι ταξινομημένος, π.χ. σε αύξουσα διάταξη
- Είναι πολύ πιο αποδοτική από τη γραμμική αναζήτηση
  - Στη χειρότερη περίπτωση απαιτούνται
    $a \log_2 n + b$ βήματα
    (a, b σταθερές, n το μέγεθος του πίνακα)

Δυαδική αναζήτηση (ii)

- Το πρόγραμμα

```
PROGRAM binsearch
INTEGER, PARAMETER:: n=100;
INTEGER:: i,howmany,mid,first,last
INTEGER:: a(n), x
LOGICAL:: found
! Μήνυμα επικεφαλίδα και οδηγίες χρήσης
read(howmany)
DO i=1,howmany
  READ(*,*) (a(i))                    !
  υποθέτουμε αύξουσα σειρά
END DO
READ(*,*) x
! Αναζήτηση και εμφάνιση αποτελέσματος
END
```

Δυαδική αναζήτηση (iii)

- Αναζήτηση και εμφάνιση αποτελέσματος

```
first=1; last=howmany
found = false
DO WHILE ((.NOT.found) .AND. (first<=last))
  mid = (first+last) / 2
  found = (x==a(mid))
   IF (x<a(mid)) THEN
   last=mid-1
   ELSE
   first=mid+1
   END IF
END DO
IF (found) THEN
   WRITE(*,*) mid
ELSE
   WRITE(*,*) 'not found'
END IF
```

Ταξινόμηση (i)

- Πρόβλημα: να αναδιαταχθούν τα στοιχεία ενός πίνακα ακεραίων σε αύξουσα σειρά
- Μια από τις σημαντικότερες εφαρμογές των ηλεκτρονικών υπολογιστών
- Βασική διαδικασία: εναλλαγή τιμών (swap)

```
SUBROUTINE SWAP (x,y)
  IMPLICIT NONE
  INTEGER:: x,y,save
  save=x
  x=y
  y=save
END
```

Ταξινόμηση (ii)

- Μέθοδος της φυσαλίδας

```
DO i=1,(n-1)
  DO j=(n-1),i,-1
    IF (a(j)>a(j+1)) THEN
      CALL swap (a(j),a(j+1))
    END IF
  END DO
END DO
```

Πλήθος συγκρίσεων

$$(n-1) + (n-2) + \ldots + 2 + 1 = n\,(n-1) / 2$$

δηλαδή, τάξης n^2 $\Rightarrow O(n^2)$, τετραγωνικό

Ταξινόμηση (iii)

- Παράδειγμα εκτέλεσης

input:	12 4 9 8 6 7 5
	12 4 9 8 6 **5 7**
	12 4 9 8 **5 6** 7
	12 4 9 **5 8** 6 7
	12 4 **5 9** 8 6 7
	12 **4 5** 9 8 6 7
$i = 1$:	**4 12** 5 9 8 6 7
	4 12 5 9 8 **6 7**
	4 12 5 9 **6 8** 7
	4 12 5 **6 9** 8 7
	4 12 **5 6** 9 8 7
$i = 2$:	4 **5 12** 6 9 8 7
	4 5 12 6 9 **7 8**
	4 5 12 6 **7 9** 8
	4 5 12 **6 7** 9 8
$i = 3$:	4 5 **6 12** 7 9 8
	4 5 6 12 7 **8 9**
	4 5 6 12 **7 8** 9
$i = 4$:	4 5 6 **7 12** 8 9
	4 5 6 7 12 **8 9**
$i = 5$:	4 5 6 7 **8 12** 9
$i = 6$:	4 5 6 7 8 **9 12**

Πολυδιάστατοι πίνακες (i)

- Παράδειγμα

```
INTEGER:: a(77,99) ή
INTEGER, DIMENSION (77,99):: a
DO i=1,10
  DO j=1,5
    READ(*,*) a(i,j)
  END DO
END DO
```

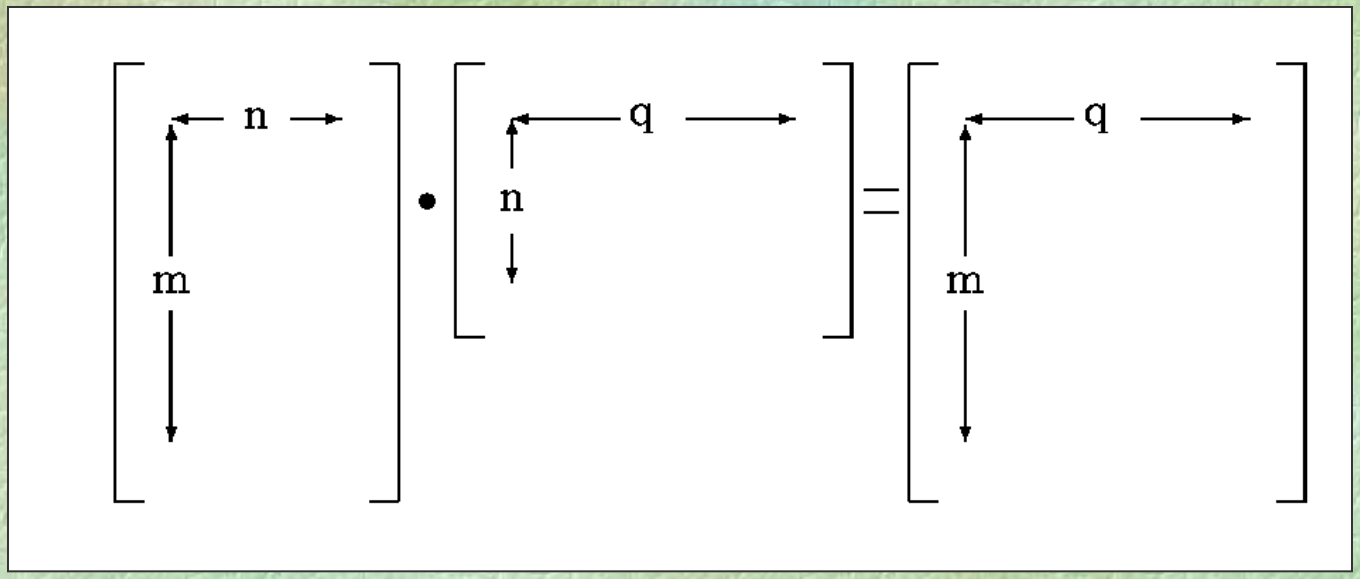
- Πίνακες περισσότερων διαστάσεων

  Η Fortran επιτρέπει μέχρι επτά διαστάσεις με οποιοδήποτε αριθμό στοιχείων ανά διάσταση.

Πολλαπλασιασμός πινάκων (i)

- Δίνονται οι πίνακες: $a\ (m \times n)$, $b\ (n \times q)$
- Ζητείται ο πίνακας: $c = a\ b\ \ (m \times q)$ όπου:

$$c_{i,j} = \sum_{k=1}^{n} a_{i,k} b_{k,j}$$



Πολλαπλασιασμός πινάκων (ii)

- Το πρόγραμμα

```
INTEGER:: a(m,n), b(n,q), c(m,q)
DO i=1,m
  DO j=1,q
    c(i,j)=0
    DO k=1,n
      c(i,j)=c(i,j)+a(i,k)*b(k,j)
    END DO
  END DO
END DO
```

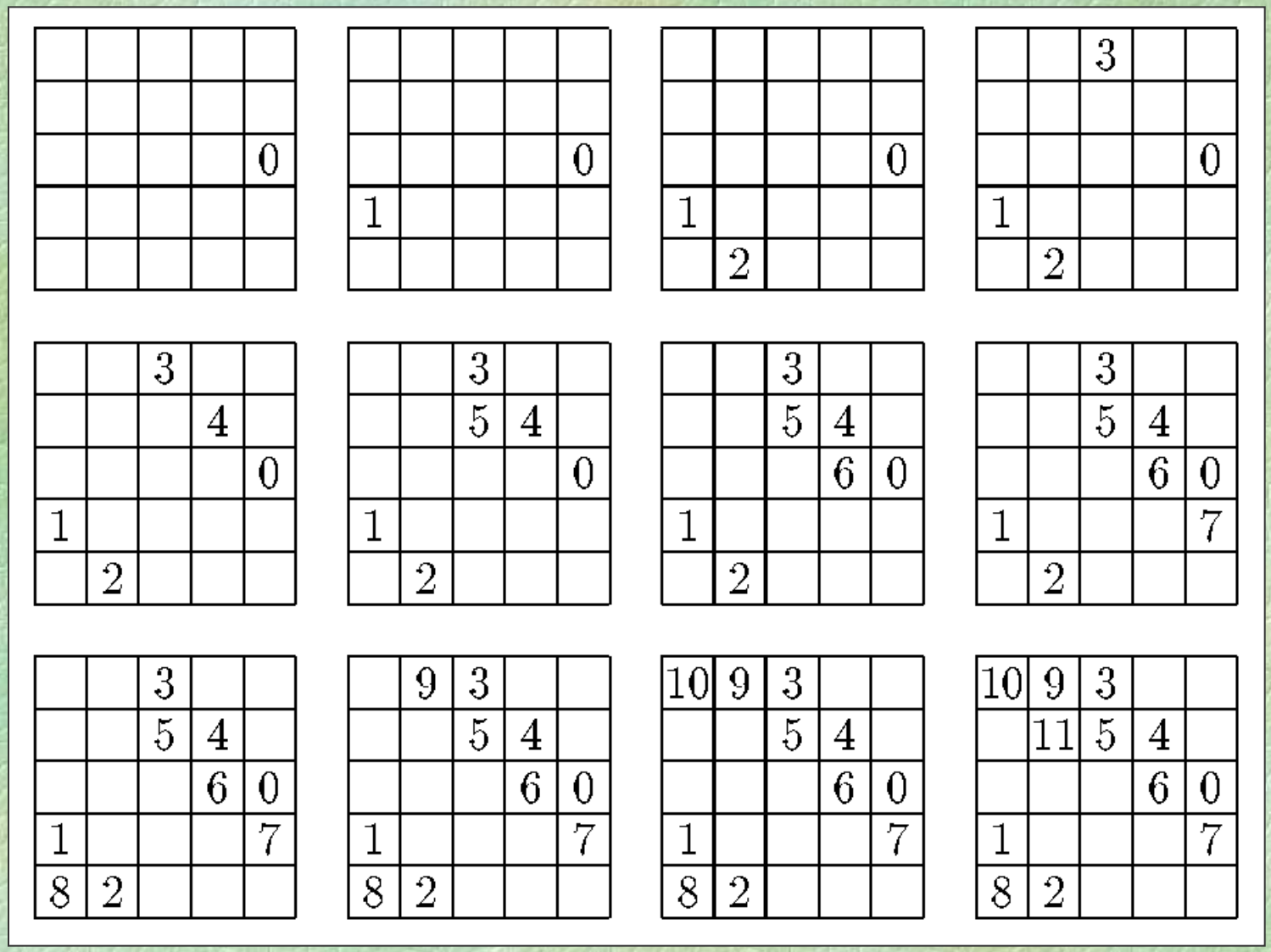
Μαγικά τετράγωνα (i)

- Διδιάστατοι πίνακες ($n \times n$) που περιέχουν όλους τους φυσικούς μεταξύ 0 και n^2-1
  - το άθροισμα των στοιχείων κάθε στήλης, γραμμής και διαγωνίου είναι σταθερό

10	9	3	22	16
17	11	5	4	23
24	18	12	6	0
1	20	19	13	7
8	2	21	15	14

- Πρόβλημα: κατασκευή μαγικού τετραγώνου ($n \times n$) για περιττό n

Μαγικά τετράγωνα (ii)



. . .

Μαγικά τετράγωνα (iii)

```
PROGRAM MAGIC_SQUARES1
IMPLICIT NONE
INTEGER, PARAMETER:: n=5
INTEGER:: h,i,j,k,m,a(n,n)
i=(n+1)/2
j=n+1
k=0
```

Δηλώσεις, Αρχικοποιήσεις

```
DO h=1,n
   j=j-1
   a(i,j)=k
   k=k+1
   DO m=2,n
      i=MOD(i,n)+1
      j=MOD(j,n)+1
      a(i,j)=k
      k=k+1
   END DO
END DO
```

Υπολογισμός των στοιχείων

```
DO i=1,n
   DO j=1,n
      WRITE(*,10) a(i,j)
      10 FORMAT (I4,1X\)
   END DO
   WRITE(*,*)
END DO
END PROGRAM
```

Εκτύπωση των στοιχείων

Μαγικά τετράγωνα (iv)

```
PROGRAM MAGIC_SQUARES2
IMPLICIT NONE
INTEGER:: h,i,j,k,m,n
INTEGER, ALLOCATABLE:: a(:,:)
WRITE(*,*) "Δώσε διάσταση πίνακα"
READ(*,*) n
ALLOCATE(a(n,n))
i=(n+1)/2
j=n+1
k=0
```

Δυναμική δέσμευση πίνακα

```
DO h=1,n
   j=j-1
   a(i,j)=k
   k=k+1
   DO m=2,n
      i=MOD(i,n)+1
      j=MOD(j,n)+1
      a(i,j)=k
      k=k+1
   END DO
END DO
```

```
DO i=1,n
   DO j=1,n
      WRITE(*,10) a(i,j)
      10 FORMAT (I4,1X\)
   END DO
   WRITE(*,*)
END DO
END PROGRAM
```

Αναστροφή πίνακα

```
PROGRAM YPOLOGOISMOS_ANASTROFOU
IMPLICIT NONE
INTEGER, PARAMETER:: N=4, M=3
INTEGER:: i, j, A(N,M), B(M,N)
WRITE(*,*) "Δώσε τιμές για τα στοιχεία του πίνακα"
READ(*,*) ((A(i,j), j=1,M), i=1,N)
DO i=1,N
   DO j=1,M
      B(j,i)=A(i,j)
   END DO
END DO

WRITE(*,10) ((A(i,j), j=1,M), i=1,N)
WRITE(*,20) ((B(i,j), j=1,N), i=1,M)
10 FORMAT (' A='/4(' |',3(I3),' |'/))
20 FORMAT (' B='/3(' |',4(I3),' |'/))
END
```

Πίνακες: τρόποι δήλωσης διαστάσεων (i)

- I. Οι διαστάσεις και το πλήθος του πίνακα είναι γνωστές από τη δήλωση του πίνακα

```
INTEGER:: p1(3,5,8)
INTEGER:: p2(4:7,5:9,-3:3)
```

  - Ο p1 έχει τις θέσεις 1,2,3 στην πρώτη διάσταση, τις 1,2,3,4,5 στη δεύτερη και τα τις 1,2,3,4,5,6,7,8 στην τρίτη (μέγεθος πίνακα 3*5*8=120).
  - Ο p2 έχει τις θέσεις 4,5,6,7 στην πρώτη διάσταση, τις 5,6,7,8,9 στη δεύτερη και τις -3 έως 3 στην τρίτη (μέγεθος πίνακα 140), π.χ. `p2(5,7,-1)=6`
  - Οι δείκτες ενός πίνακα μπορεί να είναι οποιαδήποτε ορθή έκφραση ακεραίων π.χ.

```
X(10,2), X(2*i), X(INT(ABS(a)))
```

Πίνακες: τρόποι δήλωσης διαστάσεων (ii)

- II. Πλήθος διαστάσεων γνωστό αλλά όχι και το μέγεθος. Το πλήθος στοιχείων ανά διάσταση καθορίζεται τη στιγμή της χρήσης του και εξαρτάται από το αντίστοιχο μέγεθος άλλου πίνακα.

```
INTEGER:: p1(5,8)
...
CALL R(p1)
...
SUBROUTINE R(p2)
INTEGER p2(:,:)
```

Τη στιγμή που καλείται η υπορουτίνα καθορίζεται το πλήθος στοιχείων ανά διάσταση.

Πίνακες: τρόποι δήλωσης διαστάσεων (ii)

```
PROGRAM pinakes
  IMPLICIT NONE
  INTEGER I,J,K,X(3,2)
  K=10
  DO I=1,3
     DO J=1,2
       X(I,J)=K; K=K+10
     END DO
  END DO
  WRITE(*,*), SHAPE(X)
  WRITE(*,*), X(1,:)
  CALL SUB1(X)
  CALL SUB2(X)
END
```

```
SUBROUTINE SUB1(V)
  INTEGER V(2,*)
  WRITE(*,*) SIZE(V,1)
  WRITE(*,*) V(:,1)
  WRITE(*,*) V(:,2)
END SUBROUTINE
```

```
SUBROUTINE SUB2(V)
  INTEGER V(*)
  WRITE(*,*) V(5)
END SUBROUTINE
```

έξοδος

```
 3       2
10      20
 2
10      30
50      20
40
```

Πίνακες: τρόποι δήλωσης διαστάσεων (iii)

- III. Οι διαστάσεις είναι γνωστές αλλά όχι και το μέγεθος. Το μέγεθος καθορίζεται κατά τη διάρκεια εκτέλεσης του προγράμματος. Δηλώνονται με τη χρήση της εντολής **ALLOCATE** :

```
INTEGER,ALLOCATABLE,DIMENSION(:,:,:):: pin1
...
ALLOCATE (pin1(10,10,10))
...
DEALLOCATE(pin1)
ALLOCATE (pin1(15,15,15))
```

Πίνακες: τρόποι δήλωσης διαστάσεων (iii)

```
PROGRAM allocatable_array
IMPLICIT NONE
INTEGER:: i,j
INTEGER, ALLOCATABLE, DIMENSION(:,:,:):: pin1
ALLOCATE (pin1(5,5,5))
pin1=5
WRITE(*,*) pin1; WRITE(*,*)
   !τυπώνει τον αριθμό 5, 125 φορές
DEALLOCATE(pin1)
ALLOCATE (pin1(10,10,10))
pin1=8
WRITE(*,*) pin1; WRITE(*,*)
   !τυπώνει τον αριθμό 8,  1000 φορές
DEALLOCATE(pin1)
END
```

Στοιχεία πίνακα ως δείκτες πίνακα

- Τα στοιχεία ενός πίνακα μπορούν να χρησιμοποιηθούν ως δείκτες στοιχείων άλλου αρκεί οι δείκτες να είναι ακέραιοι

```
INTEGER:: pin(5)
INTEGER:: pin1(8)
INTEGER:: pin2(5)
pin=(/2,3,4,5,8/)
pin1=(/2,3,4,5,8,3,4,5/)
pin2=pin1(pin)
```

Οι τιμές των στοιχείων του πίνακα pin2 είναι 3,4,5,8,5

Υποπίνακες με μη συνεχή στοιχεία (i)

- Η χρήση βήματος δίνει τη δυνατότητα πρόσβασης σε μη συνεχή στοιχεία πίνακα.
  - `B = A(10:20:2)` ! ο υποπίνακας A(10:20:2) περιλαμβάνει τα στοιχεία A(10), A(12), A(14), A(16), A(18) και A(20).
  - Τα στοιχεία του πίνακα B παίρνουν τις τιμές του υποπίνακα A όπως αυτός ορίστηκε παραπάνω. Δηλ. τα B(1), B(2), ... , B(6) θα πάρουν τις τιμές των στοιχείων A(10), A(12), ... ,A(20) αντίστοιχα

Υποπίνακες με μη συνεχή στοιχεία (ii)

- Η χρήση βήματος δίνει τη δυνατότητα πρόσβασης σε μη συνεχή στοιχεία πίνακα.
  - Στο βήμα μπορεί να έχουμε και αρνητικές τιμές. Για παράδειγμα B(10:1:-1). Και αν ορίσω μονοδιάστατο πίνακα C(10) και δώσω **`C=B(10:1:-1)`** παίρνω τον αντίστροφο πίνακα του B.
  - Τα άνω και κάτω όρια ενός πίνακα μπορεί να παραλειφθούν αν έχουν δηλωθεί προηγούμενα. Π.χ. η εντολή **`C(:10)=5`** θα εκχωρήσει την τιμή 5 στα στοιχεία του πίνακα C από το C(1) έως το C(10). Το ίδιο ισχύει και αν γράψω **`C(1:)=5`** ή **`C(:)=5`**

Τμήματα πινάκων (υποπίνακες) (i)

- Κάποιο τμήμα ενός πίνακα (ένας υποπίνακας) μπορεί να πάρει τιμές από κάποιο άλλο τμήμα του πίνακα υπό ορισμένες προϋποθέσεις.

  Έστω για παράδειγμα ο πίνακας

```
INTEGER, DIMENSION(1:6,1:8) :: P
```

Επιτρεπτές εκχωρήσεις τιμών

```
P(1:3,1:4)=P(1:6:2,1:8:2)   ! υποπίνακας παίρνει τιμές από άλλον ίδιων διαστάσεων
P(1:3,1:4)=2                ! υποπίνακας παίρνει σταθερή τιμή 2
P(2:6:2,1:7:3)=P(1:3,1:4)   ! υποπίνακας διαστάσεων 3x3 παίρνει τιμές υποπίνακα διαστάσεων 3x4 (δεν είναι επιτρεπτή σε όλους τους compilers)
```

Μη επιτρεπτές εκχωρήσεις τιμών

```
P(2:6:2,1:7:3)=P(2:5,7)     ! υποπίνακας παίρνει τιμές από υποπίνακα μικρότερων διαστάσεων
```

Τμήματα πινάκων (υποπίνακες) (ii)

```
INTEGER, DIMENSION(1:6,1:8):: P
INTEGER:: I,J
DO I=1,6
   DO J=1,8
      P(I,J)=I*J
   END DO
END DO
DO I=1,6
   WRITE(*,*) P(I,:); WRITE(*,*)
END DO
WRITE(*,*)
P(1:3,1:4)=P(1:6:2,1:8:2)
DO I=1,6
   WRITE(*,*) P(I,:); WRITE(*,*)
END DO
WRITE(*,*)
P(1:3,1:4)=2
DO I=1,6
   WRITE(*,*) P(I,:); WRITE(*,*)
END DO
END
```

```
P(2:6:2,1:7:3)=P(1:3,1:3)
DO I=1,6
   WRITE(*,*) P(I,:)
   WRITE(*,*)
END DO

P(2:6:2,1:7:3)=P(2:5,7)
DO I=1,6
   WRITE(*,*) P(I,:)
   WRITE(*,*)
END DO
```

Χειρισμός πινάκων χωρίς την εντολή DO

- Εκχώρηση τιμών

```
DO i=1,5
      A(i)=A(i+1)
END DO
```

Απλουστευμένη γραφή

```
A(1:5)=A(2:6)
```

- Τα παρακάτω όμως δεν είναι ισοδύναμα! Γιατί;

```
DO i=1,5
      A(i+1)=A(i)
END DO
```

Απλουστευμένη γραφή

```
A(2:6)=A(1:5)
```

Χειρισμός πινάκων χωρίς την εντολή DO

```
DO i=1,10
 A(i)=i
END DO
```

```
DO i=1,5
 A(i)=A(i+1)
END DO
WRITE(*,*) A(1:6)
```

Τυπώνει: `2 3 4 5 6 6`

```
A(1:5)=A(2:6)
WRITE(*,*) A(1:6)
```

Τυπώνει: `2 3 4 5 6 6`

- Προσοχή! Τα παρακάτω διαφέρουν:

```
DO i=1,5
      A(i+1)=A(i)
END DO
WRITE(*,*) A(1:6)
```

Τυπώνει: `1 1 1 1 1 1`

```
A(2:6)=A(1:5)
WRITE(*,*) A(1:6)
```

Τυπώνει: `1 1 2 3 4 5`

Εντολή WHERE (i)

- Εκτέλεση εντολών υπό συνθήκη σε πίνακα

```
WHERE (συνθήκη)
```

Εντολές ανάθεσης για στοιχεία πίνακα που ικανοποιούν συνθήκη (και αντίστοιχα στοιχεία άλλων πινάκων)

```
ELSEWHERE
```

Εντολές ανάθεσης για τα υπόλοιπα στοιχεία

```
ENDWHERE
```

Παράδειγμα:

```
WHERE (A>5)
    B= -A
ELSEWHERE
    B= A
ENDWHERE
```

Εντολή WHERE (ii)

```
PROGRAM WHERE_ARRAY
IMPLICIT NONE
INTEGER :: A(10), B(10), G(10)
WRITE(*,*) "Δώσε 10 τιμές για τα στοιχεία του A"
READ(*,*) A
B=0;  G=10
WHERE (A >= 5)
       B=B+1; G=G-1
ELSEWHERE
       A=-1; G=A
ENDWHERE
WRITE(*,*) "PINAKAS A"
WRITE(*,*) A
WRITE(*,*) "PINAKAS B"
WRITE(*,*) B
WRITE(*,*) "PINAKAS G"
WRITE(*,*) G
END
```

Αν ο χρήστης δώσει τιμές
`1,2,3,4,5,6,7,8,9,10` στον πίνακα A

Το πρόγραμμα τυπώνει :

```
-1  -1  -1  -1  5  6  7  8  9 10
```
(τιμές του A)
```
 0   0   0   0  1  1  1  1  1  1
```
(τιμές του B)
```
-1  -1  -1  -1  9  9  9  9  9  9
```
(τιμές του G)

Σημείωση: οι αριθμοί τυπώνονται πιο αραιά απ'ότι φαίνεται παραπάνω

Εντολή FORALL (i)

- Εκτέλεση εντολών υπό συνθήκη σε πίνακα

```
FORALL (συνθήκη)
    Εντολές ανάθεσης
ENDFORALL
```

Παραδείγματα:

```
FORALL (i=1:10) A(i,i) =20
FORALL (i=1:3, (A(i,i)/=0)) A(i,i) =20
FORALL (i=1:N, j=1:N) A(i,j)= 1/REAL(i+j)
FORALL (i=2:3)
    WHERE (C(i,:)<5)
          C(:,i)=-1
    ELSEWHERE
          C(:,i)=1
    ENDWHERE
END FORALL
```

Προσοχή: συχνά απρόβλεπτα αποτελέσματα!

Εντολή FORALL (ii)

```
PROGRAM forall_array
IMPLICIT NONE
INTEGER :: i, j, A(3,3)
READ(*,*) A
FORALL (i=1:3, (A(i,i)/=0)) A(i,i)=20
WRITE(*,*) "ΠΙΝΑΚΑΣ Α"
WRITE(*,*) A
FORALL (i=1:3) A(i,2)=48
WRITE(*,*) "ΠΙΝΑΚΑΣ Α"
WRITE(*,*) A
END
```

Αν ο χρήστης δώσει τον πίνακα Α με τιμές
1 2 3
4 5 6
7 8 9
Το πρόγραμμα θα τυπώσει:
ΠΙΝΑΚΑΣ Α
20 2 3 4 20 6 7 8 20
ΠΙΝΑΚΑΣ Α
20 2 3 48 48 48 7 8 20

```
FORALL (i=1:5)
    WHERE (C(i,:)==0)
        C(:,i)=i
    ELSEWHERE (C(i,:)>2)
        C(i,:)=9
    ENDWHERE
ENDFORALL
```

Για τιμές του C:
1 0 0 0 0
2 1 1 1 0
1 2 2 0 2
2 1 0 2 3
1 0 0 0 0
Τι θα τυπώσει το πρόγ
μετά τη WHERE?

Το πρόγραμμα μετά την FORALL τυπώνει:
1 0 0 0 0
1 1 1 1 5
1 2 2 4 9
1 1 3 2 9
1 2 0 0 5

Πίνακες: παραδείγματα (i)

```
PROGRAM PINAKES
IMPLICIT NONE
REAL, DIMENSION(4):: HEIGHTS=(/ 5.1 , 5.6 , 4.0 , 3.6 /)
CHARACTER (LEN=5),DIMENSION(3):: &
   COLORS=(/'BLUE',' RED ','GREEN'/)
INTEGER::i
INTEGER, DIMENSION(10):: INTS=(/ 100, (i, i=1,8), 100 /)

WRITE(*,*) HEIGHTS
WRITE(*,*) COLORS
WRITE(*,*) INTS
END
```

Πίνακες: παραδείγματα (ii)

```
PROGRAM  EXCHANGE_COLUMNS
IMPLICIT NONE
INTEGER, PARAMETER:: N=2, M=4
INTEGER::i,j,A(N,M), j1, j2, temp
WRITE(*,*) "GIVE ARRAY ELEMENTS"
READ(*,*) A
WRITE(*,*) "old array"
DO i=1,N
       WRITE(*,*) A(i,:)
END DO
WRITE(*,*) "GIVE COLUMNS J1,J2"
READ(*,*) J1,J2
```

```
DO i=1,N
       temp=A(i,j1)
       A(i,j1)=A(i,j2)
       A(i,j2)=temp
END DO
WRITE(*,*) "new array"
DO i=1,N
       WRITE(*,*) A(i,:)
END DO
END
```