ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΧΗΜΙΚΩΝ ΜΗΧΑΝΙΚΩΝ
ΥΠΟΛΟΓΙΣΤΙΚΟ ΚΕΝΤΡΟ

Προγραμματισμός με FORTRAN – Συνοπτικός Οδηγός

Α. Σπυρόπουλος – Α. Μπουντουβής

Αθήνα, 2015

v1.3_061015

Στον οδηγό αυτό θα χρησιμοποιηθούν οι παρακάτω όροι:

Κώδικας FORTRAN: Είναι το κείμενο που έχει γραφεί χρησιμοποιώντας τις εντολές και τους κανόνες σύνταξης της γλώσσας προγραμματισμού FORTRAN.

Μεταγλωττιστής (Compiler): Μεταφράζει τον κώδικα FORTRAN σε κώδικα που μπορεί να καταλάβει ο υπολογιστής (κώδικας μηχανής).

Εκτελέσιμο Αρχείο: Το αρχείο που δημιουργεί ο μεταγλωττιστής και περιέχει τον κώδικα μηχανής.

Προγραμματιστής: Είναι αυτός που γράφει τον κώδικα FORTRAN

Τρέξιμο (Run): Εφαρμογή (κοινώς! "double click") του εκτελέσιμου αρχείου στον υπολογιστή.

Χρήστης: Αυτός που τρέχει το εκτελέσιμο αρχείο

Ένας κώδικας FORTRAN έχει τη γενική μορφή:

```
program name
IMPLICIT NONE
Δηλώσεις μεταβλητών
Εντολές
end
```

Αρχή ενός κώδικα FORTRAN

```
program name
```

Κάθε κώδικας FORTRAN ξεκινάει με την εντολή **program** ακολουθούμενη από το όνομα *name* του κώδικα. Ο προγραμματιστής μπορεί να επιλέξει όποιο όνομα θέλει. Πρέπει όμως να προσέξει:

- να ξεκινάει με γράμμα κεφαλαίο ή μικρό
- επιτρέπετε η χρήση μόνο γραμμάτων (A – Z, a – z), ψηφίων (0 – 9) και της κάτω παύλας (underscore) _
- το μέγιστο επιτρεπόμενο μέγεθος είναι 31 χαρακτήρες

Τέλος ενός κώδικα FORTRAN

```
end
```

Κάθε κώδικας FORTRAN τερματίζεται με την εντολή **end**

Εκτύπωση μηνυμάτων στην οθόνη του υπολογιστή

```
print *, 'my message'
```

Η παραπάνω εντολή εκτυπώνει στην οθόνη το μήνυμα που βρίσκεται μέσα στα εισαγωγικά ' '

Παράδειγμα 1

```
program My_first_code
IMPLICIT NONE
print *, 'Hello World'
end
```

Όταν τρέξει ο παραπάνω κώδικας FORTRAN τότε στην οθόνη εμφανίζεται:

```
Hello World
```

Σχόλια

Είναι σκόπιμο ο προγραμματιστής να εισάγει στον κώδικά του σχόλια (μη εκτελέσιμες εντολές) για να διευκολύνει την ανάγνωση του κώδικα από τον χρήστη. Για να καταλάβει ο μεταγλωττιστής ότι μια γραμμή είναι σχόλιο και όχι εντολή FORTRAN πρέπει να ξεκινάει με το σύμβολο: **!**

Παράδειγμα 2

```
program My_second_code
IMPLICIT NONE
! This is my second FORTRAN code
print *, 'Hello World'
end
```

Όταν τρέξει ο παραπάνω κώδικας FORTRAN τότε στην οθόνη εμφανίζεται:

```
Hello World
```

Ο μεταγλωττιστής δεν λαμβάνει υπόψη τη δεύτερη γραμμή του παραπάνω κώδικα γιατί ξεκινάει με **!**

Αριθμητικές Μεταβλητές

Για να κάνουμε αριθμητικές πράξεις με μεταβλητές πρέπει να δηλώσουμε τον τύπο τους και το όνομά τους

Για να δηλώσουμε μια μεταβλητή με όνομα *variable_name* στην οποία πρόκειται να αποθηκεύσουμε πραγματικούς (real) αριθμούς, γράφουμε:

```
real variable_name
```

Ομοίως για να δηλώσουμε μια μεταβλητή με όνομα *variable_name* στην οποία πρόκειται να αποθηκεύσουμε ακέραιους (integer) αριθμούς, γράφουμε:

```
integer variable_name
```

Παράδειγμα 3

```
program code_3
IMPLICIT NONE
real i
integer j

i = 5.2
j = 3

print *, i
print *, j

end
```

Όταν τρέξει ο παραπάνω κώδικας FORTRAN τότε στην οθόνη εμφανίζεται:

```
5.2
3
```

Στον κώδικα αυτό δηλώσαμε την πραγματική μεταβλητή **i** και την ακέραια μεταβλητή **j**.
Στη συνέχεια με την εντολή

```
i = 5.2
```

ορίσαμε ότι η μεταβλητή **i** έχει την τιμή 5.2

ενώ με την εντολή

```
j = 3
```

ορίσαμε ότι η μεταβλητή **j** έχει την τιμή 3

Στο παραπάνω παράδειγμα εμφανίζεται και μια παραλλαγή της εντολής **print *,** μετά το κόμμα δε υπάρχει κάποιο μήνυμα μέσα σε εισαγωγικά αλλά το όνομα των μεταβλητών (χωρίς εισαγωγικά). Αυτό έχει ως αποτέλεσμα να τυπωθεί η τιμή της μεταβλητής και όχι το όνομα της.

ΣΗΜΕΙΩΣΗ 1: Σε κάθε κώδικα FORTRAN μετά την εντολή **program** πρέπει ο προγραμματιστής να γράφει την εντολή **IMPLICIT NONE**
Με την εντολή αυτή ο προγραμματιστής υποχρεώνεται να ορίσει κάθε μεταβλητή στον κώδικά του. Αν επιλέξει να μην χρησιμοποιήσει την εντολή αυτή, τότε ο μεταγλωττιστής επιλέγει (έμμεσα) κάθε μεταβλητή που αρχίζει με τα γράμματα i, j, k, l, m, n να είναι integer και όλες τις άλλες real. Ο έμμεσος τρόπος υποδήλωσης των μεταβλητών κρύβει σφάλματα (bugs) που δύσκολα γιατρεύονται!

Παράδειγμα 4

```
program code_4

i = 5.2
j = 3

print *, i
print *, j

end
```

Όταν τρέξει ο παραπάνω κώδικας FORTRAN τότε στην οθόνη εμφανίζεται:

```
5
3
```

Παρόλο που ο προγραμματιστής ορίζει ότι η μεταβλητή i έχει τιμή 5.2 στην οθόνη τυπώνεται η τιμή 5. Αυτό συμβαίνει γιατί ο μεταγλωττιστής ακλουθώντας τον έμμεσο τρόπο υποδήλωσης (δεν υπάρχει η εντολή **IMPLICIT NONE**) υποθέτει ότι η μεταβλητή i είναι integer και κόβει το δεκαδικό μέρος.

Στη Fortran το πρόσημο + μπροστά από έναν ακέραιο ή πραγματικό αριθμό μπορεί να παραληφτεί. Για παράδειγμα είναι το ίδιο αν γράψουμε 5.2 ή +5.2

Εκθετική μορφή

Στη Fortran οι πραγματικοί αριθμοί συμβολίζονται με δυο τρόπους. Είτε ως *πραγματικοί αριθμοί σταθερής υποδιαστολής* π.χ. 5.2 είτε ως *πραγματικοί αριθμοί κινητής υποδιαστολής* (εκθετική μορφή) π.χ. 0.52E+1
Γενικά για την εκθετική μορφή ισχύει:

$$\pm a \cdot 10^{\pm b} = \pm aE\pm b$$

Για παράδειγμα

$0.52E+1 = 0.52E1 = 0.52 \cdot 10^{1} = 5.2$
$0.52E-1 = 0.52 \cdot 10^{-1} = 0.052$

Αριθμητικές παραστάσεις

Για να δώσουμε τιμή σε μια μεταβλητή χρησιμοποιούμε το σύμβολο **=**

variable_name **=** *τιμή*

ή

variable_name **=** *αριθμητική παράσταση*

ΣΗΜΕΙΩΣΗ 2: Το σύμβολο **=** στη FORTRAN είναι **εντολή**

`j = 3` σημαίνει: δώσε στην αριθμητική μεταβλητή `j` την τιμή 3

ενώ

`j = j + 1` σημαίνει: δώσε στην αριθμητική μεταβλητή `j` την τιμή που είχε η μεταβλητή `j` και πρόσθεσε την τιμή 1

Παράδειγμα 5

```
program code_5
IMPLICIT NONE
integer j

j = 3
j = j + 1

print *, j

end
```

Όταν τρέξει ο παραπάνω κώδικας FORTRAN τότε στην οθόνη εμφανίζεται:

```
4
```

Οι **αριθμητικοί τελεστές** που χρησιμοποιούνται στις αριθμητικές παραστάσεις είναι:

```
+  Πρόσθεση
-  Αφαίρεση
*  Πολλαπλασιασμός
/  Διαίρεση
** Ύψωση σε δύναμη
```

Η προτεραιότητα των τελεστών είναι:

```
**       Υψηλή
* και /  Μεσαία
+ και -  Χαμηλή
```

Μεταξύ τελεστών της ίδιας προτεραιότητας οι πράξεις γίνονται από αριστερά προς τα δεξιά.

Για παράδειγμα στην αριθμητική παράσταση:

```
a = f/h**g + i
```
η σειρά που θα γίνουν οι πράξεις είναι:

1. Υπολογισμός του `h**g` και αποθήκευση του αποτελέσματος σε μια προσωρινή μεταβλητή `temp_1`

2. Υπολογισμός του `f/temp_1` και αποθήκευση του αποτελέσματος σε μια προσωρινή μεταβλητή `temp_2`
3. Υπολογισμός του `temp_2 + i` και αποθήκευση του αποτελέσματος στη μεταβλητή `a`

Υπάρχει περίπτωση να αλλάξουμε τη προκαθορισμένη σειρά (προτεραιότητα των τελεστών) που γίνονται οι πράξεις χρησιμοποιώντας παρενθέσεις `( )`

Για παράδειγμα στην αριθμητική παράσταση:

```
a = (f/h)**g + i
```

η σειρά που θα γίνουν οι πράξεις είναι:

1. Υπολογισμός του `f/h` και αποθήκευση του αποτελέσματος σε μια προσωρινή μεταβλητή `temp_1`
2. Υπολογισμός του `temp_1**g` και αποθήκευση του αποτελέσματος σε μια προσωρινή μεταβλητή `temp_2`
3. Υπολογισμός του `temp_2 + i` και αποθήκευση του αποτελέσματος στη μεταβλητή `a`

Οι μεταβλητές και οι αριθμοί που χρησιμοποιούνται στις αριθμητικές παραστάσεις μπορεί να είναι ή μόνο **`real`** ή μόνο **`integer`** ή συνδυασμός και των δυο.
Αυτό που πρέπει να έχουμε υπόψη μας είναι:

1. Πράξεις μεταξύ **`real`** μεταβλητών ή αριθμών έχουν **`real`** αποτέλεσμα
2. Πράξεις μεταξύ **`integer`** μεταβλητών ή αριθμών έχουν **`integer`** αποτέλεσμα
3. Πράξεις μεταξύ **`real`** και **`integer`** μεταβλητών ή αριθμών έχουν **`real`** αποτέλεσμα

Για παράδειγμα, έστω ότι θέλουμε να υπολογίσουμε το αποτέλεσμα της διαίρεσης του 5 με το 2 και να το αποθηκεύσουμε σε μια μεταβλητή `a`
Η τιμή που θα πάρει τελικά η μεταβλητή `a` (βλ. πίνακα) εξαρτάται από το πώς θα την ορίσουμε στην αρχή του κώδικα και από το πώς θα γράψουμε την διαίρεση.

	`real` a	**`integer`** a
	Τιμή	
`a = 5./2.`	2.5	2
`a = 5/2`	2.	2
`a = 5./2`	2.5	2
`a = 5/2.`	2.5	2

Παρατηρούμε ότι όταν θέλουμε να γράψουμε τους αριθμούς 5 ή 2 ως real γράφουμε 5. ή 2. που είναι ισοδύναμο με το να γράφαμε 5.0 ή 2.0 αντίστοιχα.

Εισαγωγή δεδομένων από το πληκτρολόγιο του υπολογιστή

```
read *, variable_name
```

Η χρήση της εντολής **read *,** υποχρεώνει τον χρήστη να εισάγει κάποιο νούμερο ή κείμενο από το πληκτρολόγιο, το οποίο και αποθηκεύεται στην μεταβλητή *variable_name*

Παράδειγμα 6

```
program code_6
IMPLICIT NONE
real x

print *, 'Give a real number'
read *, x

print *, x

end
```

Όταν τρέξει ο παραπάνω κώδικας FORTRAN τότε στην οθόνη εμφανίζεται:

```
Give a real number
```

και η εκτέλεση του σταματά μέχρι ο χρήστης να εισάγει από το πληκτρολόγιο έναν αριθμό και να πατήσει το ENTER.
Στη συνέχεια εκτυπώνεται ο αριθμός που πληκτρολόγησε ο χρήστης.

Έλεγχος της ροής του κώδικα

Δομή 1

```
if (συνθήκη) then
   .
   εντολές
   .
endif
```

Πολλές φορές είναι αναγκαίο να ελέγχεται η τιμή μιας μεταβλητής και ανάλογα με την τιμή της ο κώδικας να εκτελεί διαφορετικές εντολές.

Αν η συνθήκη μέσα στην παρένθεση ικανοποιείται, τότε εκτελούνται οι εντολές που βρίσκονται μετά το **then** και πριν το **endif**, αλλιώς η εκτέλεση του κώδικα συνεχίζεται με τις εντολές που βρίσκονται μετά το **endif**

Παράδειγμα 7

```
program code_7
IMPLICIT NONE
real x

print *, 'Give a real number'
read *, x

if (x<0) then

  print *, 'Negative'

endif

end
```

Όταν τρέξει ο παραπάνω κώδικας FORTRAN τότε στην οθόνη εμφανίζεται:

```
Give a real number
```

και η εκτέλεση του σταματά μέχρι ο χρήστης να εισάγει από το πληκτρολόγιο έναν αριθμό και να πατήσει το ENTER.
Στη συνέχεια αν ο αριθμός που πληκτρολογήσαμε είναι μικρότερος από το μηδέν τότε εμφανίζεται το μήνυμα

```
Negative
```

ενώ αν πληκτρολογήσαμε ένα θετικό αριθμό τότε ο κώδικας δεν εμφανίζει μήνυμα.

Οι τελεστές σύγκρισης που χρησιμοποιούνται στην FORTRAN είναι:

>	μεγαλύτερο
<	μικρότερο
>=	μεγαλύτερο ή ίσο
<=	μικρότερο ή ίσο
/=	διάφορο
==	ίσο (ΠΡΟΣΟΧΗ: ο τελεστής ισότητας συμβολίζεται με 2 ίσον)

Ορισμένοι από τους λογικούς τελεστές που χρησιμοποιούνται στην FORTRAN είναι:
`.AND.`
`.OR.`

Για παράδειγμα στο
if ((*συνθήκη1***).AND.(***συνθήκη2***)) then**
.
.
endif
οι εντολές που βρίσκονται μετά το **then** και πριν το **endif** εκτελούνται μόνο αν η *συνθήκη1* **ΚΑΙ** η *συνθήκη2* είναι αληθής

Ενώ στο

```
if ((συνθήκη1).OR.(συνθήκη2)) then
.
.
endif
```

οι εντολές που βρίσκονται μετά το **then** και πριν το **endif** εκτελούνται μόνο αν η *συνθήκη1* **Ή** η *συνθήκη2* είναι αληθής

Δομή 2

```
if (συνθήκη) then
  .
  εντολές
  .
else
  .
  εντολές
  .
endif
```

Παράδειγμα 8

```
program code_8
IMPLICIT NONE
real x

print *, 'Give a real number'
read *, x

if (x<0) then
  print *, 'Negative'
else
  print *, 'Positive'
endif

end
```

Όταν τρέξει ο παραπάνω κώδικας FORTRAN τότε στην οθόνη εμφανίζεται:

```
Give a real number
```

και η εκτέλεση του σταματά μέχρι ο χρήστης να εισάγει από το πληκτρολόγιο έναν αριθμό και να πατήσει το ENTER.
Στη συνέχεια αν ο αριθμός που πληκτρολογήσαμε είναι μικρότερος από το μηδέν τότε εμφανίζεται το μήνυμα:

```
Negative
```

ενώ αν πληκτρολογήσουμε ένα θετικό αριθμό ή το μηδέν τότε ο κώδικας εμφανίζει το μήνυμα:

```
Positive
```

Δομή 3

```
if (συνθήκη) εντολή
```

Αν θέλουμε να εκτελεστεί μια μόνο εντολή υπό συνθήκη τότε μπορούμε να γράψουμε την εντολή **if** όπως φαίνεται πιο πάνω, χωρίς τη χρήση **else** και **endif**

Παράδειγμα 9

```
program code_9
IMPLICIT NONE
real x

print *, 'Give a real number'
read *, x

if (x<0) print *, 'Negative'
if (x>=0) print *, 'Positive'

end
```

Ο κώδικας του παραδείγματος αυτού κάνει ότι και ο κώδικας του παραδείγματος 8

Επανάληψη do

```
do μετρητής = αρχική τιμή, τελική τιμή, βήμα
   .
   εντολές
   .
enddo
```

μετρητής : είναι το όνομα μιας αριθμητικής μεταβλητής που έχουμε ορίσει,
αρχική τιμή, *τελική τιμή*, *βήμα* : ορίζουν το σύνολο των τιμών που θα πάρει ο *μετρητής*.

Σε αρκετές περιπτώσεις κατά τη συγγραφή ενός κώδικα FORTRAN είναι απαραίτητο ορισμένες εντολές να εκτελεστούν πολλές φορές. Για να το πετύχουμε αυτό χρησιμοποιούμε την επανάληψη **do**

Παράδειγμα 10

```
program code_10
IMPLICIT NONE
integer i

do i = 4, 10, 2
  print *, i
enddo

end
```

Όταν τρέξει ο παραπάνω κώδικας FORTRAN τότε, η εντολή `print *, i` εκτελείται 4 φορές και στην οθόνη εμφανίζεται:

```
4
6
8
10
```

Αν το *βήμα* είναι 1 τότε μπορεί να παραληφθεί.

Παράδειγμα 11

```
program code_11
IMPLICIT NONE
integer i

do i = 4, 10
  print *, i
enddo

end
```

Όταν τρέξει ο παραπάνω κώδικας FORTRAN τότε, η εντολή `print *, i` εκτελείται 7 φορές και στην οθόνη εμφανίζεται:

```
4
5
6
7
8
9
10
```

ΣΗΜΕΙΩΣΗ 3: Ο προγραμματιστής μεταξύ **do** και **enddo** δεν πρέπει να δίνει τιμές στον μετρητή.

Για παράδειγμα το παρακάτω κομμάτι κώδικα είναι λάθος και ο μεταγλωττιστής δεν θα φτιάξει εκτελέσιμο αρχείο.

```
do i = 4, 10
  i = 1
enddo
```

Μπορούμε επίσης να χρησιμοποιήσουμε δυο επαναλήψεις **do** τη μία μέσα στην άλλη αρκεί να μην έχουν τους ίδιους μετρητές.

Παράδειγμα 12

```
program code_12
IMPLICIT NONE
integer i,j

do i = 1, 4
  do j = 1, 3
    print *, i,j
  enddo
enddo

end
```

Όταν τρέξει ο παραπάνω κώδικας FORTRAN τότε, η εντολή **print *,** i,j εκτελείται 12 φορές και στην οθόνη εμφανίζεται:

```
1     1
1     2
1     3
2     1
2     2
2     3
3     1
3     2
3     3
4     1
4     2
4     3
```

Αέναη Επανάληψη

```
do
  .
  εντολές
  if (συνθήκη) exit
  .
enddo
```

Υπάρχει περίπτωση να θέλουμε να εκτελεστούν κάποιες εντολές αρκετές φορές χωρίς να γνωρίζουμε πόσες, αλλά να γνωρίζουμε την συνθήκη που πρέπει να ικανοποιηθεί για να τερματιστεί η εκτέλεσή τους. Στην περίπτωση αυτή χρησιμοποιούμε την παραπάνω μορφή **do** (χωρίς μετρητή) μαζί με την εντολή **if** και την εντολή **exit**. Όταν ικανοποιηθεί η

συνθήκη της εντολής **if** τότε η εντολή **exit** μας βγάζει από την αέναη επανάληψη και η εκτέλεση του κώδικα συνεχίζεται με τις εντολές που βρίσκονται μετά το **enddo**

Παράδειγμα 13

```
program code_13
IMPLICIT NONE
integer i

i = 0
do
  i = i + 1
  print *, i
  if (i>5) exit
enddo

end
```

Όταν τρέξει ο παραπάνω κώδικας FORTRAN τότε στην οθόνη εμφανίζεται:

```
1
2
3
4
5
6
```

Διατεταγμένα σύνολα τιμών (Arrays)

Κατά τη συγγραφή ενός κώδικα FORTRAN πολλές φορές θα συναντήσουμε υπολογισμούς που αναφέρονται σε διατεταγμένα σύνολα τιμών (arrays). Για παράδειγμα όταν έχουμε πολλά πειραματικά δεδομένα και πρέπει να επαναλάβουμε τους ίδιους υπολογισμούς σε κάθε ένα αυτά.

Οι arrays είναι αριθμητικές μεταβλητές που μπορούν να φιλοξενήσουν περισσότερες από μία τιμές.

Διανύσματα ή Μονοδιάστατες Arrays

Για να ορίσουμε μια μονοδιάστατη array με όνομα *variable_name* στην οποία πρόκειται να αποθηκεύσουμε πραγματικούς (real) αριθμούς, γράφουμε:

```
real, allocatable, dimension(:) :: variable_name
```

Για να δηλώσουμε ότι η μονοδιάστατη array *variable_name* θα φιλοξενήσει *count* αριθμούς, μέσα στο πρόγραμμα γράφουμε:

```
allocate (variable_name(count))
```

Ομοίως, για να <u>ορίσουμε</u> μια μονοδιάστατη array με όνομα *variable_name* στην οποία πρόκειται να αποθηκεύσουμε ακέραιους (integer) αριθμούς, γράφουμε:

```
integer, allocatable, dimension(:) :: variable_name
```

Για να δηλώσουμε ότι η μονοδιάστατη array *variable_name* θα φιλοξενήσει *count* αριθμούς, <u>μέσα στο πρόγραμμα</u> γράφουμε:

```
allocate (variable_name(count))
```

Για παράδειγμα:

```
integer, allocatable, dimension(:) :: x

allocate (x(5))
```

σημαίνει ότι ορίζουμε 5 ακέραιες μεταβλητές με ονόματα:

```
x(1) x(2) x(3) x(4) x(5)
```

<u>Παράδειγμα 14</u>

```
program code_14
IMPLICIT NONE
integer, allocatable, dimension(:) :: x
integer N

N=5
allocate (x(N))

x(1)= 3
x(2)= 6
x(3)= 4
x(4)= 1
x(5)= 7

print *, x(1)
print *, x(2)
print *, x(3)
print *, x(4)
print *, x(5)

end
```

Όταν τρέξει ο παραπάνω κώδικας FORTRAN τότε στην οθόνη εμφανίζεται:

```
3
6
4
1
7
```

Για να μη γράφουμε πολλές φορές την εντολή **print *,** μπορούμε να τροποποιήσουμε τον παραπάνω κώδικα κάνοντας χρήση της επαναληπτικής διαδικασίας **do**

ΣΗΜΕΙΩΣΗ 4: Είναι πάγια προγραμματιστική τεχνική η χρήση των arrays μαζί την επαναληπτική διαδικασία **do** για να εκτελέσουμε τις ίδιες πράξεις/εντολές πάνω στις τιμές διατεταγμένων συνόλων.

Παράδειγμα 15

```
program code_15
IMPLICIT NONE
integer i, N
integer, allocatable, dimension(:) :: x

N=5

allocate (x(N))

x(1)= 3
x(2)= 6
x(3)= 4
x(4)= 1
x(5)= 7

do i = 1, N
  print *, x(i)
enddo

end
```

Όταν τρέξει ο παραπάνω κώδικας FORTRAN τότε στην οθόνη εμφανίζεται:

```
3
6
4
1
7
```

Το ίδιο ακριβώς αποτέλεσμα με το Παράδειγμα 14.

Πίνακες ή Διδιάστατες Arrays

Για να ορίσουμε μια διδιάστατη array με όνομα *variable_name* στην οποία πρόκειται να αποθηκεύσουμε πραγματικούς (real) αριθμούς, γράφουμε:

```
real, allocatable, dimension(:,:) :: variable_name
```

Για να δηλώσουμε ότι η διδιάστατη array *variable_name* θα φιλοξενήσει (*count1* x *count2*) αριθμούς, μέσα στο πρόγραμμα γράφουμε:

```
allocate (variable_name(count1,count2))
```

Ομοίως, για να <u>ορίσουμε</u> μια διδιάστατη array με όνομα *variable_name* στην οποία πρόκειται να αποθηκεύσουμε ακέραιους (integer) αριθμούς, γράφουμε:

```
integer, allocatable, dimension(:,:) :: variable_name
```

Για να δηλώσουμε ότι η διδιάστατη array *variable_name* θα φιλοξενήσει (*count1* x *count2*) αριθμούς, <u>μέσα στο πρόγραμμα</u> γράφουμε:

```
allocate (variable_name(count1,count2))
```

Για παράδειγμα:

```
integer, allocatable, dimension(:,:) :: x

allocate (x(4,3))
```

σημαίνει ότι ορίζουμε 4 x 3 = 12 ακέραιες μεταβλητές με ονόματα:

```
x(1,1) x(1,2) x(1,3)
x(2,1) x(2,2) x(2,3)
x(3,1) x(3,2) x(3,3)
x(4,1) x(4,2) x(4,3)
```

Παράδειγμα 16

```
program code_16
IMPLICIT NONE
integer i,j, N, M
integer, allocatable, dimension(:,:) :: x

N=4
M=3
allocate (x(N,M))

x(1,1)= 3
x(1,2)= 6
x(1,3)= 4
x(2,1)= 1
x(2,2)= 9
x(2,3)= 7
x(3,1)= 4
x(3,2)= 3
x(3,3)= 8
x(4,1)= 10
x(4,2)= 12
x(4,3)= 11


do i = 1, N
  do j = 1, M
    print *, x(i,j)
  enddo
enddo

end
```

Όταν τρέξει ο παραπάνω κώδικας FORTRAN τότε στην οθόνη εμφανίζεται:

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

Παρατηρούμε ότι για να τυπωθεί μια διδιάστατη array (πίνακας) πρέπει να γίνει χρήση δυο επαναλήψεων **do** το ένα μέσα στο άλλο με διαφορετικούς μετρητές, i και j (βλ. ΣΗΜΕΙΩΣΗ 3 και Παράδειγμα 12)

Στο Παράδειγμα 12 είχαμε γράψει **print *,** i,j για να τυπωθούν τα 9 ζευγάρια τιμών i, j
Στο Παράδειγμα 16 έχουμε γράψει **print *,** x(i,j) για να τυπωθούν οι 9 τιμές της array x

Υποπρογράμματα

Τα υποπρογράμματα είναι ανεξάρτητα προγράμματα τα οποία μπορούν να χρησιμοποιηθούν από το κυρίως πρόγραμμα.

Οι κατηγορίες των υποπρογραμμάτων είναι:
Διαδικασίες (Subroutines)
Συναρτήσεις (Functions)

Κάθε κατηγορία μπορεί να είναι είτε:
Εσωτερική (Internal)
Εξωτερική (External)
Βιβλιοθήκη (Intrinsic)

Στις σημειώσεις αυτές αλλά και στο μάθημα θα ασχοληθούμε με:

1. Εξωτερικές Διαδικασίες
2. Εξωτερικές Συναρτήσεις
3. Συναρτήσεις βιβλιοθήκης

Εξωτερικές Διαδικασίες (External Subroutines)

Μια εξωτερική διαδικασία γράφετε μετά την εντολή **end** του κυρίως προγράμματος ή σε ξεχωριστό αρχείο και έχει τη γενική μορφή:

```
subroutine name(μεταβλητή1, μεταβλητή2, …)
IMPLICIT NONE
Δηλώσεις μεταβλητών
Εντολές
end subroutine
```

Η εξωτερική διαδικασία καλείται μέσα από το κυρίως πρόγραμμα με την εντολή **call**

```
call name(μεταβλητή1, μεταβλητή2, …)
```

Εξωτερικές Συναρτήσεις (External Functions)

Μια εξωτερική συνάρτηση γράφετε μετά την εντολή **end** του κυρίως προγράμματος ή σε ξεχωριστό αρχείο και έχει τη γενική μορφή:

```
τύπος function name(μεταβλητή1, μεταβλητή2, …)
IMPLICIT NONE
Δηλώσεις μεταβλητών
Εντολές
end function
```

Όπου *τύπος* δηλώνεται ο τύπος (π.χ. real, integer) της μεταβλητής *name*. Η δήλωση αυτή είναι απαραίτητη γιατί οι συναρτήσεις επιστρέφουν μια τιμή μέσω του ονόματός τους *name*.

Η εξωτερική συνάρτηση καλείται μέσα από το κυρίως πρόγραμμα με την εντολή **=**

μεταβλητή **=** *name*(*μεταβλητή1, μεταβλητή2, …*)

Η τιμή αυτή αποθηκεύεται στην μεταβλητή *μεταβλητή* του κυρίως προγράμματος.

Ορίσματα

Οι μεταβλητές *μεταβλητή1, μεταβλητή2, …* που βρίσκονται μέσα στις παρενθέσεις των συναρτήσεων και των διαδικασιών ονομάζονται ορίσματα. Τα ορίσματα αυτά πρέπει να είναι σε αντιστοιχία (τύπος, πλήθος) με τα ορίσματα των εντολών κλίσης τους.

Παράδειγμα χρήσης εξωτερικής διαδικασίας

Το παρακάτω κυρίως πρόγραμμα καλεί τη διαδικασία `add` με την εντολή **`call`** `add(a,b,c)`. Η διαδικασία `add` μεταφέρει τις τιμές των μεταβλητών `a` και `b` στις μεταβλητές `x` και `y` αντίστοιχα που ορίζονται μέσα στη διαδικασία `add`. Στη συνέχεια με την εντολή `z = x + y` προστίθενται οι τιμές των `x` και `y` και το αποτέλεσμα αποθηκεύεται στην μεταβλητή `z`. Η διαδικασία `add` τερματίζεται με την εντολή **`end subroutine`** και στη συνέχεια το κυρίως πρόγραμμα συνεχίζει την εκτέλεσή του μετά την εντολή **`call`** `add(a,b,c)` με τη μεταβλητή `c` να έχει αποκτήσει την τιμή της μεταβλητής `z`.

Παράδειγμα 17

```
program main
IMPLICIT NONE
integer a,b,c

a = 5
b = 6

call add(a,b,c)

print *, 'The sum of',a,b
print *, 'is',c
end

subroutine add(x,y,z)
IMPLICIT NONE
integer, intent(IN)  :: x,y
integer, intent(OUT) :: z

z = x + y

end subroutine
```

Μέσα στη διαδικασία `add` η δήλωση των μεταβλητών περιέχει μια επιπλέον οδηγία που περιγράφει τον σκοπό της μεταβλητής (**`intent(IN)`** ή **`intent(OUT)`**). Η **`intent(IN)`** υποδηλώνει ότι οι μεταβλητές `x` και `y` θα πάρουν τιμές από το κυρίως πρόγραμμα, ενώ η **`intent(OUT)`** υποδηλώνει ότι η μεταβλητή `z` θα δώσει τιμή στο κυρίως πρόγραμμα.

Παράδειγμα χρήσης εξωτερικής συνάρτησης

Ο κώδικας του Παραδείγματος 17 μπορεί να γραφεί και με χρήση εξωτερικής συνάρτησης ως:

Παράδειγμα 18

```
program main
IMPLICIT NONE
integer a,b,c
integer add

a = 5
b = 6

c = add(a,b)

print *, 'The sum of',a,b
print *, 'is',c
end

integer function add(x,y)
IMPLICIT NONE
integer, intent(IN)  :: x,y

add = x + y

end function
```

Συναρτήσεις βιβλιοθήκης(intrinsic functions)

Η FORTRAN παρέχει στον προγραμματιστή μια βιβλιοθήκη από συναρτήσεις οι οποίες χρησιμοποιούνται ευρέως. Μερικές από αυτές είναι:

Συνάρτηση	Επεξήγηση	Δήλωση του x	Αποτέλεσμα
REAL(x)	Μετατροπή του x σε REAL	INTEGER	REAL
INT(x)	Μετατροπή του x σε INTEGER	REAL	INTEGER
ABS(x)	Απόλυτη τιμή του x	INTEGER	INTEGER
		REAL	REAL
SQRT(x)	Ρίζα του x	REAL	REAL
SIN(x)	Ημίτονο του x	REAL	REAL
COS(x)	Συνημίτονο του x	REAL	REAL
TAN(x)	Εφαπτομένη του x	REAL	REAL
EXP(x)	e^x	REAL	REAL
LOG(x)	ln(x)	REAL	REAL
LOG10(x)	log(x)	REAL	REAL