Εισαγωγή στη Fortran

Μάθημα 1ο

Ελευθερία Λιούκα
liouka.eleftheria@gmail.com

Περιεχόμενα

- Ιστορία της Fortran
- Βασικές γνώσεις Fortran
  - Επιτρεπτοί χαρακτήρες
  - Μορφή προγράμματος
  - Τύποι μεταβλητών
  - Πράξεις και λογικές εκφράσεις
  - Ακρίβεια δεκαδικών ψηφίων

Ιστορία της Fortran

- Ονομασία : FORmula TRANslation
- Ανάπτυξη ανάμεσα στο 1954 και 1957 από την IBM
- Σκοπός : Επίλυση μαθηματικών προβλημάτων
- Εκδόσεις :
  - FORTRAN-66
  - FORTRAN-77
  - FORTRAN-90
  - FORTRAN-95
  - FORTRAN 2000
  - FORTRAN 2003
  - FORTRAN 2008

Γιατί FORTRAN?

- Πιο ισχυρή από matlab, C++
- Η ανώτερη γλώσσα ως σήμερα για αριθμητικές, επιστημονικές,τεχνολογικές εφαρμογές

FORTRAN vs C++

- Παρραληλισμός δεδομένων
- Εξαγωγή δεδομένων
- Συναρτησιακός προγραμματισμός
- Αριθμητική αξιοπιστία

C++ : Αντικειμενοστραφής προγραμματισμός

Coding in UNIX

- Οποιοδήποτε πρόγραμμα επεξεργασίας κειμένου
- Αποθήκευση με κατάληξη .f90
  - mycode.f90

Compile program

- Πληκτρολόγηση f90 ακολουθούμενο από το όνομα του προγράμματος
  - f90 mycode.f90

  Δημιουργία εκτελέσιμου αρχείου a.out

Διαφορές FORTRAN 77 – FORTRAN 90

FORTRAN 77

- Fixed source form
- Γραφή σε στήλες
  - 1-5 statement labels
  - 6 continuation indicator
  - 7-72 statements
- Χρήση συγκεκριμένων χαρακτήρων
  - A-Z, 0-9
  - + - * / = ( ) ‘ . , $: blank

FORTRAN 90

- Free source form
- Ελεύθερη γραφή
- Επιπλέον χαρακτήρες
  - a-z
  - ! “ & ; < > ?
- Έκδοση 2003
  - ~ \ [ ] ` ^ { } | # @

Χρήσιμοι χαρακτήρες

- ! : Ονομασία bang
  - Χρήση : Εισαγωγή πριν το σχόλιο (αγνόηση από τον compiler)
- &
  - Χρήση : Συνέχιση εντολής σε επόμενη γραμμή
  - Μέγιστη επέκταση : 39 συνεχόμενες γραμμές

  (Μήκος γραμμής : 132 χαρακτήρες)

Παράδειγμα

```
1 PROGRAM test
2 !This line is a comment
3 !Ignored by the compiler
4 a=3.0+4.0+&
5     13.0
6 !After execution variable a will have the value 20.0
7  PRINT*a
8  END
```

Ιδιότητες

- Case insensitive
  - Α ίδιο με a
  - Write, write, WRITE, Write, …
- Λέξεις κλειδιά με κεφαλαία
  - PROGRAM, END, PRINT, READ, …

Μορφή Προγράμματος

1) Heading
2) Implicit None
3) Specification Section
4) Execution Section
5) Internal Subprograms
6) End Program

HEADING

- Έχει τη μορφή PROGRAM name, όπου το όνομα αποτελείται μόνο από επιτρεπτούς χαρακτήρες χωρίς κενά
- Πρέπει να ξεκινάει με γράμμα
- Μέγιστο μήκος 30 χαρακτήρες

Tip:

Να είναι σχετικό με το πρόγραμμα που φτιάχνω

Implicit None

- Εντολή όχι απαραίτητη για να κάνει compile και να τρέξει το πρόγραμμα
- Αποτρέπει τυχόν errors που σχετίζονται με τους τύπους μεταβλητών
- Ακυρώνει τη σύμβαση ονομασίας

Specification Section

- Το τμήμα στο οποίο δηλώνονται απαραίτητα όλες οι μεταβλητές και οι σταθερές που θα χρησιμοποιηθούν

Execution Section

- Το τμήμα στο οποίο προσδιορίζονται οι ενέργειες του προγράμματος

Internal Subprograms

- Αν χρειαστεί να χρησιμοποιηθούν υπορουτίνες τοποθετούνται σε αυτό το τμήμα (μέσα στο κυρίως πρόγραμμα)

End Program

- Δείχνει στον compiler που τελειώνει το πρόγραμμα και σταματά την εκτέλεση

(μπορούμε να χρησιμοποιήσουμε και ετικέτα)

Παράδειγμα

```
1  PROGRAM introduction
2  IMPLICIT NONE
3  !Σε αυτό το τμήμα ορίζω τις σταθερές και τις
4  !μεταβλητές που χρειάζομαι
5  INTEGER :: age=26
6  CHARACTER(4) :: name="ERIC"
7  !Τμήμα εκτέλεσης ενεργειών
8  PRINT*, "My name is ",name,"and I am ",age,"years old."
9  !Τμήμα που θα βάζαμε τις υπορουτίνες αν είχαμε
10 END PROGRAM introduction
```

Τύποι μεταβλητών

- Πέντε βασικοί τύποι
  - Real
  - Integer
  - Complex
  - Character
  - Logical
- Δήλωση μεταβλητών/σταθερών
  - TYPE :: var

  Όπου TYPE : τύπος δεδομένων π.χ. Real, Integer κτλ

  var : το όνομα της μεταβλητής

Παραδείγματα

```
REAL :: temp
INTEGER :: A=4
CHARACTER :: name,answer
REAL :: A,B,C=2.0
```

REAL

- Οι πραγματικοί αριθμοί αναπαριστώνται σε δεκαδική μορφή
  - Το 3 δεν ειναι πραγματικός, το 3.0 ειναι
- Οι δυνάμεις έχουν τη μορφή:
  - 4.56E2

INTEGER

- Θετικοί ή αρνητικοί αριθμοί (και το 0)
- Χωρίς κόμμα, τελεία κτλ

COMPLEX

- Δυνατότητα ορισμού μιγαδικών (φανταστικών) αριθμών
- Το πρώτο όρισμα αναπαριστά το πραγματικό μέρος και το δεύτερο το φανταστικό μέρος

  – Π.χ. Αν θέλω να γράψω τον αριθμό 4.2 + 2.3i :

  COMPLEX :: A

  A = (4.2,2.3)

CHARACTER

- Μια ακολουθία επιτρεπτών χαρακτήρων/ συμβόλων της FORTRAN που περικλείεται από εισαγωγικά ή απόστροφους
- Καλό είναι να ορίζουμε το μήκος του CHARACTER που θα χρησιμοποιήσουμε
- CHARACTER (LEN = n) :: name
  CHARACTER (n) :: name
- word = "can't"
  word = 'can''t'

LOGICAL

- Παίρνουν την τιμή .TRUE ή .FALSE
- Αν ζητηθεί από το χρήστη να εισάγει κάποια από τις δύο τιμές:

```
LOGICAL :: A, B

PRINT *, "What are the values of A and B?"
READ (*,*) A, B
```

  - Εισαγωγή .TRUE, .Turtle κτλ σημαίνει TRUE

PARAMETER

- Συγκεκριμένη σταθερά εμφανίζεται συχνά, την ορίζουμε ως παράμετρο
  - Π.χ. INTEGER, PARAMETER :: N = 10
    REAL, PARAMETER :: pi = 3.141593
- Δεν μπορούμε να αλλάξουμε την τιμή ξανά

Αριθμητικές Πράξεις

- Είναι οι: +,-,*,/,**
- Σειρά πράξεων:
  - Οι δυνάμεις πρώτα από δεξιά προς αριστερά
  - Πολλαπλασιασμοί και διαιρέσεις
  - Προσθέσεις και αφαιρέσεις
- Χρήση παρενθέσεων αν χρειάζεται

Λογικές Εκφράσεις

- Εκφράσεις που δίνουν σαν αποτέλεσμα μια λογική σταθερά (TRUE, FALSE)

Σύμβολο	Ερμηνεία
< ή .LT.	Μικρότερο από
> ή .GT.	Μεγαλύτερο από
== ή .EQ.	Ίσο με
<= ή .LE.	Μικρότερο ή ίσο από
>= ή .GE.	Μεγαλύτερο ή ίσο από
/= ή .NE.	Διάφορο από

- A .NE. 6
  C * A >= 3

Λογικές Εκφράσεις

- Οι χαρακτήρες μπορούν να συγκριθούν ως προς αλφαβητική σειρά
  - “A” > “B”

    “better” > “between”

    “good year” > “goodyear”

  Όλα τα παραπάνω είναι TRUE

Σύνθετες Λογικές Εκφράσεις

- Χρήση των:
  - .NOT. , .AND. , .OR. , .EQV. , .NEQV.
- .NOT. : Επιστρέφει την αντίθετη τιμή
- .AND. :
  - TRUE αν και οι δύο εκφράσεις είναι TRUE
  - FALSE διαφορετικά
- .OR. :
  - FALSE αν και οι δύο εκφράσεις είναι FALSE
  - TRUE διαφορετικά
- .EQV. :
  - TRUE αν οι δυο εκφράσεις έχουν την ίδια τιμή
  - FALSE διαφορετικά
- .NEQV. : Το ανάποδο από την .EQV.

Ακρίβεια Πραγματικών Αριθμών

- Απλή ή διπλή ακρίβεια
- E : Απλή ακρίβεια
  - Π.χ. 3.0E1
- D : Διπλή ακρίβεια
  - Π.χ. 3.0D1
- Δήλωση πραγματικού αριθμού με απλή/διπλή ακρίβεια:

REAL(KIND = *kind_number*) :: *vars*

kind_number = 1 : απλή

kind_number = 2 : διπλή

Ακρίβεια Πραγματικών Αριθμών

- Σε άλλους compiler
  - *kind_number = 4 : απλή*
  - *kind_number = 8 : διπλή*
- Γενική εντολή:
  - SELECTED REAL KIND (*p*, *r*)

  Όπου p: πλήθος ψηφίων ακρίβειας

  r: εύρος ακρίβειας (-10^r to 10^r)
  - Είτε ξέρω το p είτε το r ο compiler καταλαβαίνει τι είδους ακρίβεια χρειάζομαι

Ακρίβεια Πραγματικών Αριθμών

- Χρήση μέσω παραμέτρου:
  - INTEGER, PARAMETER :: double = SELECTED_REAL_KIND (13)
    REAL (KIND = double) :: A, B
- *Παράδειγμα*

```
PROGRAM accuracy
IMPLICIT NONE
INTEGER, PARAMETER :: double = SELECTED_REAL_KIND (13)
REAL (KIND = double) :: A, B, C
A = 2.0_double
B = 0.1 * A
C = 0.1_double * A
PRINT *, B
PRINT *, C
END PROGRAM
```

Τέλος