Programmazione in Fortran:

Lezione 1

A.A. 2009/2010

Ing. A . Siviglia

nunzio.siviglia@ing.unitn.it

stanza:

Laboratorio didattico di modellistica idrodinamica (2° piano)

Tel 2440

Obiettivi

- Imparare ad utilizzare il calcolatore come strumento di calcolo

Strumenti

- Linguaggio di programmazione Fortran

Modalità

- Lezioni in aula ----> costruire un codice partendo (solitamente) da un codice non completo assegnato dall'esercitatore.

- Verifica a casa e confronto con codice messo a disposizione dell'esercitatore.

Libri consultabili in biblioteca

- Fortran 90/95 guida alla programmazione

S.J. Chapman

- Fortran 90/95 for Scientists and Engineers

S.J. Chapman

- Principi di Programmazione in Fortran per l'Ingegneria Ambientale

G. Antonacci, G. Vignoli

PERCHE' USARE IL FORTRAN?

I programmatori utilizzano vari linguaggi ad alto livello, ciascuno dei quali ha delle caratteristiche speciali. Alcuni linguaggi sono stati appositamente ideati per risolvere problemi scientifici, altri per creare applicazioni di tipo gestionale e altri ancora per programmare i sistemi operativi.

E' importante scegliere il linguaggio con caratteristiche appropriate al tipo di problema da risolvere.

Fra i linguaggi ad alto livello ricordiamo, Basic, C, COBOL, Fortran, Pascal.

Il Fortran e' il linguaggio piu' appropriato a svolgere calcoli scientifici. E' utilizzato da piu' di 50 anni per implementare modelli di centrali nucleari, nella progettazione di aerei, nei sistemi di elaborazione dei segnali sismici e nel campo dell'Idraulica

FORTRAN

Il nome sta per IBM Mathematical Formula Translation System o più brevemente:

FORmula TRANslation

E’ il linguaggio di programmazione più vecchio tra tutti i linguaggi evoluti

Cenni Storici

Primo compilatore 1957	**Fortran 2**
Primo standard ufficiale 1972	**Fortran 66**
Rivisto nel 1980	**Fortran 77**
Rivisto nel 1991	**Fortran 90**
Upgrade nel 1996	**Fortran 95**
Ultimo upgrade nel 2004	**Fortran 2003**

Struttura di un programma Fortran

Ogni programma Fortran e' formato da un insieme di istruzioni eseguibili e non eseguibili, che devono essere disposte in un determinato ordine

Sezione dichiarativa: La sezione delle dichiarazioni, posta all'inizio del programma, contiene le istruzioni non eseguibili che definiscono il nome del programma, il numero e i tipi di variabili che saranno utilizzate nel programma

Sezione esecutiva: La sezione esecutiva contiene una o più istruzioni eseguibili che descrivono le azioni che dovranno essere svolte dal programma

Sezione conclusiva: La sezione conclusiva contiene le istruzioni STOP e END PROGRAM. STOP indica al computer di interrompere l'esecuzione del programma.

END PROGRAM e' un'istruzione che indica al compilatore che non ci sono altre istruzioni da compilare all'interno del programma.

Struttura di un programma Fortran

```
PROGRAM primo_programma
!Scopo:
!Illustrare alcune funzioni di base di un programma Fortran
!

!Dichiarazioni di variabili
IMPLICIT NONE
INTEGER:: i,j  !dichiarazioni variabili intere input
INTEGER:: k    !dichiarazione variabili intere output


!Acquisizione variabili da moltiplicare
WRITE(*,*) 'introduci due interi separati da uno spazio '
READ(*,*)  i,j

! Moltiplicazione dei numeri
k=i*j

!Visualizza il risultato
WRITE(*,*) 'Risultato =', k


!fine del programma
STOP
END PROGRAM primo_programma
```

Sezione dichiarativa

Sezione esecutiva

Sezione conclusiva

Compilare ed eseguire un programma Fortran

Prima di eseguire il programma campione (`primo_programma`), bisogna compilarlo con un compilatore Fortran e poi collegarlo (link) con le librerie di sistema del computer per produrre un file eseguibile. Questi due passaggi di solito vengono svolti simultaneamente in risposta ad un unico comando del programmatore.

Il compilatore trasforma le istruzioni Fortran in linguaggio macchina

Compilatori

- Compaq Visual Fortran (sulle macchine dell'Universita')
- Compilatore Salford FTN95 per Windows

  (scaricabile dal sito http://www.silverfrost.com/)
- Compilatore Intel per Linux

Forma del sorgente

Free Form	*Fixed Form*
132 caratteri per linea	1 caratteri per linea (solo fino a **72** utilizzabili)
! Inizio commento	**C** a colonna 1
& carattere di continuazione linea	Un qualunque carattere a colonna 6
; Separatore di istr. (più istruzioni sulla stessa riga)	Una sola istruzione per riga
Bianchi significativi	Bianchi non significativi
	Etichette da colonna 1 a colonna 5

L'insieme dei caratteri in F90/F77

- **Alfanumerici:** **a-z, A-Z, 0-9, e _ (Underscore)**
- **Simboli :** **space + * ( ) , ‘ : ! % > < ? = -**

  / . ; & $ “

Bianchi significativi

In un codice sorgente in formato libero i bianchi non debbono apparire:

- **Tra parole chiave:** INTEGER :: uno ! corretto

  INT EGER :: uno ! errato
- **Tra nomi di variabili:** REAL :: nome_uno ! corretto

  REAL :: nome uno ! errato

Bianchi significativi (segue)

Gli spazi bianchi debbono apparire:

- **Tra due parole chiave**
- **Tra parole chiave e nomi che non siano separate da caratteri speciali**

```
INTEGER  FUNCTION  prova(i)  ! Corretto
INTEGERFUNCTION  prova(i)    ! Errato
INTEGER  FUNCTIONprova(i)    ! Errato
```

COSTANTI E VARIABILI

Costante -----> dato assegnato prima della compilazione e il cui valore non cambia durante l'esecuzione;

Variabile -----> dato il cui valore cambia durante l'esecuzione, (può essere inizializzato) ;

COSTANTI E VARIABILI TIPO INTEGER

I dati di tipo INTEGER sono costanti e variabili intere

Es:

0

-999

123456789

+17

COSTANTI E VARIABILI TIPO

REAL

I dati di tipo REAL sono costanti e variabili in formato reale.

N.B. Posso quindi rappresentare numeri frazionali

Es:

10.

-999.9

123.456789E20

0.12E-3

COSTANTI E VARIABILI TIPO CHARACTER

I dati di tipo CHARACTER sono costituite da stringhe di caratteri alfanumerici

Es:

'THIS IS A TEST'

' '

“3.141593”

Nomi di Variabili e Procedure

(Il FORTRAN non è case_sensitive)

In Fortran 90 nomi di variabili e di procedure:

- **Debbono iniziare con una lettera**

```
REAL :: a1    !   Corretto

REAL :: 1a    !   Errato
```

- **Si posssono utilizzare solo lettere, cifre da 0 a 9 ed il carattere underscore**

```
CHARACTER :: a_z  !   Corretto

CHARACTER :: a-z  !   Errato
```

- **Un nome non può essere più lungo di 31 caratteri (6 in F77)**

Suggerimenti e commenti

Nella stesura di un sorgente è bene utilizzare un sufficiente numero di commenti.

- Ovunque dopo il **!** si ha un commento. Si può pertanto inserire un commento dopo un istruzione eseguibile. In F77 ciò non è possibile in quanto si può commentare solo l'intera riga.

- Il **!** in un contesto di tipo carattere non è interpretato come commento.

```
PRINT*,   "Avete afferrato il concetto !! "

WRITE(*,*)  "  ……….. "   !   In F90
```

Tipi predefiniti

Il Fortran 90 (e Fortran 77) ha tre classi di oggetti predefiniti:

- **Tipo carattere**
- **Tipo logico**
- **Tipo numerico**

```
CHARACTER                 ::  sei     !  Lettera
CHARACTER(LEN=15)         ::  nome    !  Stringa
LOGICAL                   ::  vero    !  Vero/Falso
REAL                      ::  pi      !  3.141592
INTEGER                   ::  codice  !  Intero
COMPLEX                   ::  val     !  X + i Y
DOUBLE PRECISION          ::  pi  !  3.141592.. (No in F90)
```

Tipi implicitamente predefiniti

Variabili non dichiarate hanno un tipo implicito:

- Se la prima lettera è: I, J ,K ,L ,M, N allora il tipo è intero
- Ogni altra lettera fornisce un tipo reale

L'utilizzo di tipi implicitamente predefiniti è spesso pericoloso, si preferisce pertanto eliminare questa possibilità con il comando:

! Fortemente consigliato

Dichiarazioni di tipi numerici e logici

Con l'utilizzo di IMPLICIT NONE tutte le variabili debbono essere dichiarate. La sintassi è del seguente tipo:

<tipo>[,< lista degli attributi >] :: < lista delle variabili >

[= < valore >]

Esempi di dichiarazioni valide:

```
REAL                          :: x
INTEGER                       ::  i, j =4
LOGICAL, POINTER              :: p
REAL, DIMENSION(4,4)          :: a,b  !  Matrice (4 x 4 )
```

Dichiarazioni di tipi carattere

Hanno una sintassi simile alle variabili numeriche.

1. Ci si può riferire ad un solo carattere
2. Riferire ad una stringa di caratteri

Le seguenti dichiarazioni sono tutte valide:

```
CHARACTER(LEN=10)          :: nome
CHARACTER                  :: sesso
CHARACTER(LEN=5) , DIMENSION(3,3) :: a
```

Costanti (Parametriche)

Costanti simboliche, note come parametri, in FORTRAN possono essere definite o tramite un attributo o tramite l'istruzione parameter.

REAL , PARAMETER :: pi = 3.141592

CHARACTER(LEN=*) , PARAMETER :: a ="MARIO"

In tal caso la lunghezza della stringa è pari al valore associato.

Grandezze parametriche possono essere usate:

- Se è noto che tale variabile può assumere un solo valore
- Per leggibilità, quando in un programma compaiono costanti particolari, come ad esempio π

1 Quando c'è necessità di dover cambiare tale valore

Operatori predefiniti

Numerici : **, *, /, +, -

Possono essere applicati a variabili, costanti sia di tipo scalare che vettoriale. L'unica restrizione è che l'operando a destra di ** non può essere un vettore.

Di confronto : =, /=, >, >=, <, <=

.EQ. .NE. .GT. .GE. .LT. .LE.

Restituiscono un valore logico quando combinati con operandi numerici. (boolean= I .GT. J)

Attenzione nell'utilizzo di tali operatori quando i due operandi sono espressioni con risultato reale.

Operatori predefiniti (segue)

Operatori Logici : **.NOT.** **.AND.** **.OR.** **.EQV.** **.NEQV.**

T = .TRUE. ; **F = .FALSE.**

.NOT. T → Falso ; **.NOT. F → Vero**

T .AND. F → Falso ; **T .AND. T → Vero**

T .OR. F → Vero ; **F .OR. F → Falso**

T. EQV. F → Falso ; **F .EQV. F → Vero**

T .NEQV. F → Vero ; **F .NEQV. F → Falso**

Un’espressione logica da sempre come risultato un valore logico.

Operatori predefiniti (segue)

Operatore predefinito che agisce su stringhe: //

Quando si usano le stringhe si può far riferimento ad una sottostringa con la sintassi str1(inf:sup)

1. str1 è "CAME"
2. str1(1:1) è "C" (str1(1) è errato)
3. Str1(2:4) è "AME"

CHARACTER (LEN=*) , PARAMETER :: str1="CAME"

CHARACTER (LEN=*) , PARAMETER :: str2="RINO"

WRITE (*,*) str1//str2 ! Output→ "CAMERINO"

WRITE (*,*) str1(2:3)//str2(2:2) ! Output → "AMI"

Precedenze degli operatori

Dalla più alta alla più bassa

OPERATORE	ESEMPIO
**	10**4
* /	89*55
+ -	-4
+ -	5+8
//	str1//str2
= =/ > >= < <=	A<=B
.NOT.	.NOT. A
.AND.	A .AND. B
.OR.	A .OR. B
.EQV. .NEQV.	A .EQV. B

Istruzione di assegnazione

< var > = < expr >

La variabile (l'oggetto) a sinistra deve essere dello stesso tipo del risultato dell'espressione a destra.

Una espressione è la combinazione tramite gli operatori, sia predefiniti che dell'utente, di variabili, costanti, funzioni, costanti parametriche …, eventualmente con l'utilizzo delle parentesi tonde per poter definire le precedenze tra le varie operazioni

Esempi:

a = b

c = SIN(0.6) * 12.5

boolean = (a .EQ. b .OR. c .NE. d)

Esercizi al calcolatore

1. Scrivere un codice che divida due numeri interi, che moltiplichi due numeri reali e che restituisca sullo schermo due variabili carattere.
2. Utilizzando il codice di cui al punto 1, Eseguire le ss operazioni: 3/4, 5/4, 3./4., 3/4., 3./4.
3. Scrivere un codice che calcoli l'area di un cerchio e il volume di una sfera assegnato il raggio R.