UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Laboratorio di Calcolo Numerico
Laboratorio 2: Primi programmi in Fortran 90

Andrea Franceschini

E-mail: franceschini@dmsa.unipd.it

Dispense:
http://www.math.unipd.it/∼putti/teaching/calcolo_ambientale/index.html

10 Marzo 2015

Ambiente Linux: comandi essenziali

ls	*list*: lista dei file e delle directory presenti in una directory `[studente@pc ~]$ ls`
ls -l	*list*: lista dettagliata del contentuto di una directory `[studente@pc ~]$ ls -l`
ls -a	*list*: lista di tutto il contenuto di una directory (anche nascosto) `[studente@pc ~]$ ls -a`
pwd	*print working directory*: indica la directory in cui ci si trova `[studente@pc ~]$ pwd`
cd	*change directory*: per cambiare directory `[studente@pc ~]$ cd nomedirectory`
cd ..	*change directory*: per tornare alla directory superiore `[studente@pc ~]$ cd ..`

Ambiente Linux: comandi essenziali

mkdir	*make directory*: per creare una nuova directory `[studente@pc ~]$ mkdir nomedirectory`
rmdir	*remove directory*: per cancellare una directory (vuota) `[studente@pc ~]$ rmdir nomedirectory`
cp	*copy*: per copiare un file `[studente@pc ~]$ cp nomefile1 nomefile2`
mv	*move*: per spostare un file `[studente@pc ~]$ mv nomefile nomedirectory`
mv	*move*: per rinominare un file (ATTENZIONE) `[studente@pc ~]$ mv nomefileold nomefilenew`
rm	*remove*: per cancellare un file `[studente@pc ~]$ rm nomefile`

Primo programma Fortran90

- *Obiettivo*: scrivere ed eseguire un programma in Fortran90 per la stampa sul terminale della frase "hello world".
- Aprire l'editor di testo *g*edit con il nome del file *hw.f90*
- Scrivere il seguente testo nel file *hw.f90*:

```
1: program hw
2: write(*,*) 'hello world'
3: end program hw
```

- Salvare il file e compilare scrivendo sul terminale:
  `[studente@pc ~]$ gfortran hw.f90`
- Eseguire il programma 'a.out':
  `[studente@pc ~]$ ./a.out`

Editare un file “fortran90”

Aprire un terminale e digitare i seguenti comandi:

- `[studente@pc ~]$ mkdir laboratorio1`
- `[studente@pc ~]$ cd laboratorio1`
- Creare il file sorgente:
  `[studente@pc laboratorio1]$ gedit hw.f90`
- `[studente@pc ~]$ ls`
  hw.f90

Creare un eseguibile e lanciare un programma

Compilare il file sorgente:

- `[studente@pc laboratorio1]$ gfortran hw.f90`
- `[studente@pc laboratorio1]$ ls`
  `a.out  hw.f90`
- `[studente@pc laboratorio1]$ rm a.out`
- `[studente@pc laboratorio1]$ gfortran hw.f90 -o hw.exe`
- `[studente@pc laboratorio1]$ ls`
  `hw.f90  hw.exe`
- Eseguire il programma:
  `[studente@pc laboratorio1]$ ./hw.exe`

ATTENZIONE: Creare un eseguibile

Per errore, può capitare di scrivere il comando

- `[studente@pc laboratorio1]$ gfortran hw.f90 -o hw.f90`
  In qeusto caso il file sorgente viene perso
  Proviamo con una copia di *hw.f90*
- `[studente@pc laboratorio1]$ cp hw.f90 prova.f90`
- `[studente@pc laboratorio1]$ ls`
  `hw.f90  hw.exe  prova.f90`
- `[studente@pc laboratorio1]$ gfortran prova.f90 -o prova.f90`
- `[studente@pc laboratorio1]$ ls`
  `hw.f90  hw.exe  prova.f90`
- `[studente@pc laboratorio1]$ gedit prova.f90`
  ??????????

Esercizio 1

Scrivere un programma che valuti la funzione

$$y = 4x^3 - 3x^2 - 3$$

in un punto x. Il punto scelto è $x = 3$. La soluzione sarà $y = 78$.
Come visto prima, si utilizza la sintassi del comando *write* per stampare a schermo il risultato di un'operazione. Il comando diventa:

```
1: write(*,*) 'y =',y
```

Si devono evitare i caratteri accentati. Se vogliamo scrivere qualcosa come *Il risultato è*, si evita la *è* e la stringa diventa:

```
1: write(*,*) 'risultato =',y
```

NOTA: Tutte le variabili sono reali e il programma si chiama *eval1.f90*.

Programma 1

```
1: program eval1
2: implicit none      ! Dichiarazione variabili
3: real*8 :: x,y
4: x = 3.d0
5: ! La funzione da valutare e' y = 4x^3-3x^2-3
6: y = 4.d0 * x**3 - 3.d0 * x**2 - 3.d0
7: write(*,*) 'risultato =',y
8: end program eval1
```

Esercizio 2

Scrivere un programma che valuti la stessa funzione $y = 4x^3 - 3x^2 - 3$ in un punto letto in input. Si utilizza la sintassi del comando *read*:

```
1: read(*,*) x
```

Esercizio 3

Scrivere un programma che valuti la solita funzione in $n + 1$ punti equispaziati nell'intervallo $I = [x_{min}, x_{max}]$:
$x_0 = x_{min}, x_1 = x_0 + h, \ldots, x_n = x_{max}$, con $h = (x_{max} - x_{min})/n$.
Considerare, per esempio, $x_{min} = -4$, $x_{max} = 6$ e $n = 11$.
Serve utilizzare un ciclo *do*. La sintassi è:

```
1: do i = 1,n + 1
2:       ...
3: end do
```

Programma 2

```
1: program eval2
2: implicit none      ! Dichiarazione variabili
3: real*8 :: x,y
4: write(*,*) 'Inserire il valore x'
5: read(*,*) x
6: ! La funzione da valutare e' y = 4x^3-3x^2-3
7: y = 4.d0 * x**3 - 3.d0 * x**2 - 3.d0
8: write(*,*) 'risultato =',y
9: end program eval2
```

Programma 3

```
 1: program eval3
 2: implicit none      ! Dichiarazione variabili
 3: integer :: n,i
 4: real*8 :: x,y,xmin,xmax,h
 5: n = 11
 6: xmin = -4.d0
 7: xmax = 6.d0
 8: ! "Omogeneizzazione" dell'operazione (n reale)
 9: h = (xmax - xmin) / float(n)
10: do i = 1,n+1
11:       x = xmin + (i-1) * h
12:       ! La funzione da valutare e' y = 4x^3-3x^2-3
13:       y = 4.d0 * x**3 - 3.d0 * x**2 - 3.d0
14:       write(*,*) 'La funzione vale y = ',y,'per x = ',x
15: end do
16: end program eval3
```

Esercizio 4

Scrivere un programma che calcoli le radici reali x_1, x_2 di una equazione di secondo grado $ax^2 + bx + c = 0$. I coefficienti a, b, c sono letti dal terminale.
La sintassi per invocare il comando radice quadrata è: y = **sqrt**(x).
Serve controllare che il discriminante sia positivo: si usa il costrutto *if*.

```
1: if (istruzione1) then
2:       ! Eseguito se istruzione1 è vera
3:       ...
4: else
5:       ! Eseguito se istruzione1 è falsa
6:       ...
7: end if
```

Programma 4

Sostituire le parti in rosso e scrivere le istruzioni per i vari casi *if*.

```
 1: program radici
 2: implicit none
 3: real*8 :: a,b,c,delta,x1,x2
 4: Input dati
 5: delta = b**2 - 4.d0 * a * c
 6: if (delta .gt. 0.d0) then ! Esistono due soluzioni
 7:       ...
 8: else
 9:       if (delta .eq. 0.d0) then ! Soluzioni coincidenti
10:             ...
11:       else ! Non esistono soluzioni
12:             ...
13:       end if
14: end if
15: end program radici
```