Sviluppare un programma in FORTRAN

- Scrivere il file sorgente.
  $\rightarrow$ Editor di testo (**emacs**, vi, notepad, etc...)
- Compilare.
  $\rightarrow$ Compilatore (ifort, gfortran, etc...)
- Eseguire il programma.
  $\rightarrow$ Riga di comando, doppio click, etc...

Vantaggi di emacs

Se il file che si sta creando termina per .f90 emacs attiva la modalità ambiente di sviluppo che attiva

- syntax highlight (colora le parole chiave, gli identificatori, le stringhe in modo diverso)
- indent region (sposta e allinea a destra le istruzioni all'interno dei blocchi)
- tab completion (premendo tab dopo aver scritto end aggiunge le parole chiave opportune)
- comment region, change keyword case, etc...

Sequenza di comandi da digitare in ambiente bash shell

```
-> emacs mioprogramma.f90 &
-> ifort mioprogramma.f90
-> ./a.out
```

oppure

```
-> ifort mioprogramma.f90 -o eseguibile
-> ./eseguibile
```

Se & segue un comando, il processo viene mandato in background ed è possibile digitare nuovi comandi.

Compilazione

Digitando

```
-> ifort mioprogramma.f90 -o eseguibile
```

viene creato il file `eseguibile` che è un programma. Per eseguirlo in `bash shell` occorre premettere `./`

In questo modo comunicate al sistema che il file `eseguibile` si trova in `./` che è la directory corrente.

Se non viene usata l'opzione `-o` il file creato è `a.out` che si esegue con `./a.out`

Struttura di un programma FORTRAN

```
PROGRAM nomeprogramma
   IMPLICIT NONE
   ⋮
   dichiarazioni di variabili
   istruzioni
   ⋮
END PROGRAM nomeprogramma
```

Tipi semplici e dichiarazioni

- `real`: numeri di macchina in virgola mobile (su 4 bytes)
- `integer`: interi di macchina (su 2 o 4 bytes)
- `character`: carattere ASCII o stringa
- `logical`: variabile booleana (vero o falso)
- `complex`: variabile complessa

Il comando

```
real ::  numero
```

dichiara la variabile `numero` di tipo reale.

Tipi semplici e dichiarazioni

È possibile dichiarare più variabili dello stesso tipo sulla stessa linea e inizializzarle in fase dichiartiva
Esempio

```
integer ::  k,h,s=1
```

Gli identificatori sono stringhe alfanumeriche che cominciano per un carattere alfabetico.

`ciao`, `a1243`, `decisamentetroppolunga` sono nomi validi
`1casa` non è valido.

Tipi semplici e dichiarazioni

È possibile aggiungere uno o più parametri che definiscono meglio il tipo di una variabile

```
real(8) ::  a
```

dichiara una variabile reale su 8 bytes
(doppia precisione: valori da 10^{-323} a 10^{308} circa, 16 cifre significative).

```
integer(4) ::  a
```

dichiara un intero su 4 bytes
(valori compresi tra -2^{31} a $2^{31}-1$)

Tipi semplici e dichiarazioni

```
character(7) ::  mia
```

dichiara una stringa di 7 caratteri.

La stringa si assegna con il comando

```
mia=''casetta''
```

È possibile dichiarare una costante con l'opzione `parameter`

```
real(8),parameter::pi=3.141592653589793
```

Operazioni

- Operazioni aritmetiche `+ * - / **`
- Operazioni logiche `== /= < > <= >= .and. .or.`

Conversioni di tipi

In un'operazione tra due variabili di tipo diverso viene effettuata una conversione di tipo, il risultato è del tipo più forte.

integer * real(4) $\rightarrow$ real(4)

real(4) + real(8) $\rightarrow$ real(8)

ATTENZIONE alle costanti letterali!
1/3 è un intero e il risultato è 0

Per avere un risultato reale occorre scrivere 1.0/3.0

Per avere un risultato in doppia precisione occorre scrivere 1.0d0/3.0d0 (basta 1.d0/3)

Il comando IF

Ci sono due forme per il comando IF

- Con una sola istruzione
  ```
  IF (condizione) istruzione
  ```
- Con un blocco di istruzioni
  ```
  IF (condizione) THEN
     istruzioni
     ⋮
  ELSE
     istruzioni
     ⋮
  END IF
  ```

Il comando DO

```
label:  DO i=iniziale,finale,incremento
   istruzioni
   ⋮
END DO label
```

I valori iniziali, finali e l'incremento possono essere delle espressioni, incremento è opzionale se vale 1.

La label (etichetta) è opzionale e si usa per aumentare la leggibilità in caso di cicli annidati.

Il comando DO *incondizionato*

Ciclo incondizionato

```
DO
    ⋮
   IF (condizione) EXIT
    ⋮
END DO
```

Il comando SELECT CASE

Selezione per casi

```
SELECT CASE (variabile)
CASE(valore1)
   istruzioni
   ⋮
CASE(valore2)
   istruzioni
   ⋮
CASE DEFAULT
   istruzioni
   ⋮
END SELECT
```