Programmare in Fortran

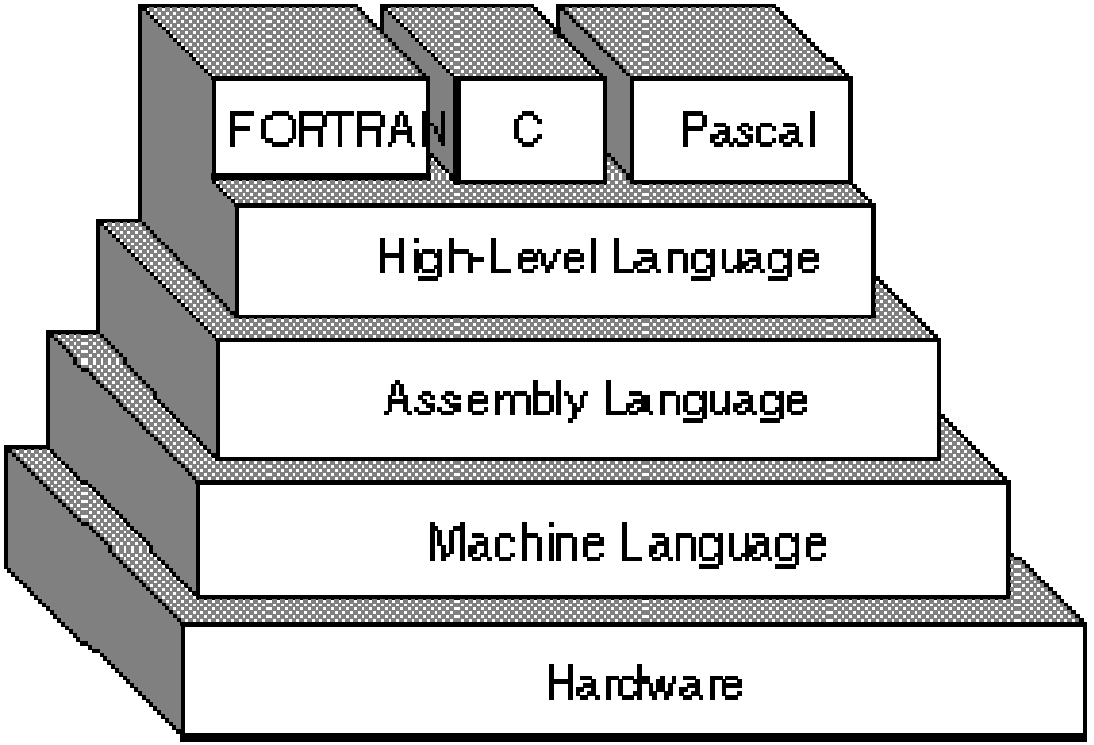
By Luigi Vetrano

Perchè abbiamo bisogno dei linguaggi di programmazione ?

- Prospettiva del programmatore:
  - Modo di pensare
  - Modo di esprimere gli algoritmi
  - Linguaggio comprensibile agli umani

- Astrazione della macchina virtuale
  - Modo di specificare cosa volete che l'HW faccia senza entrare nella complessità della macchina reale
  - E infine, i linguaggi rispecchiano anche il punto di vista dell'implementatore

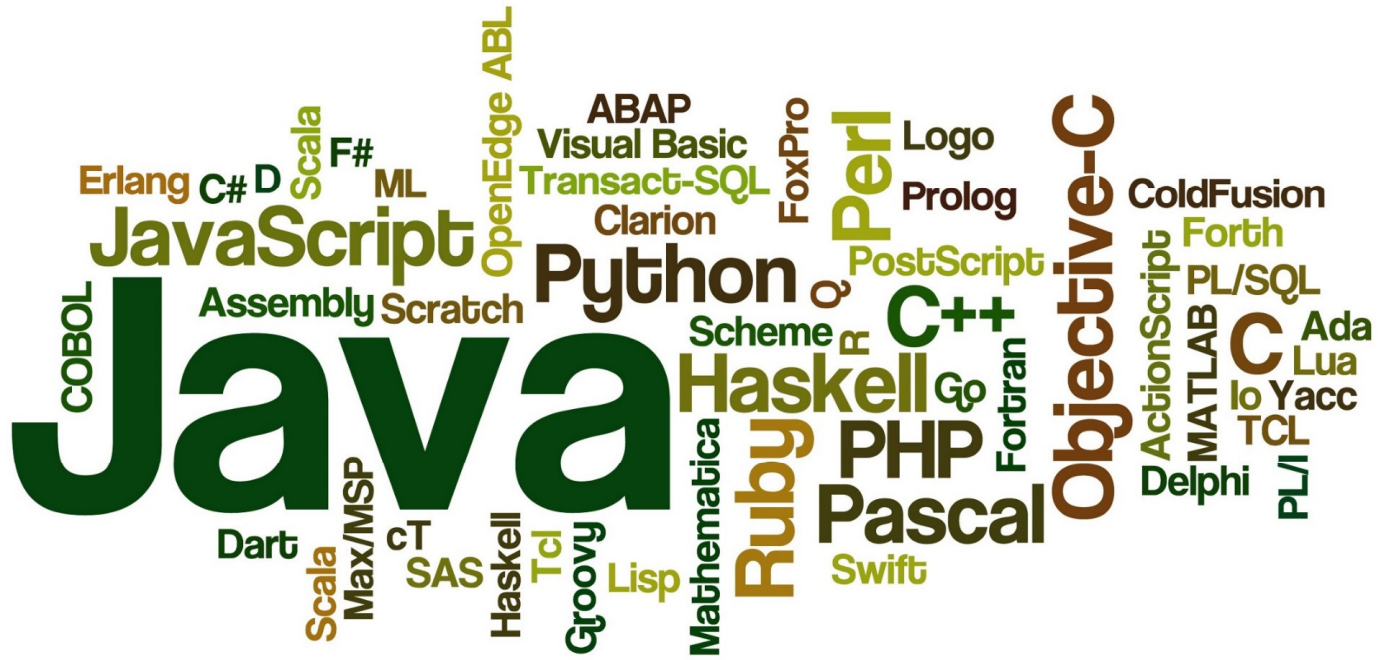
Storia dei linguaggi di programmazione



Perchè ci sono tanti linguaggi di programmazione ?

- Evoluzione – abbiamo imparato a fare le cose sempre meglio col tempo
- Fattori Socio-economici :
  - Interessi Proprietari legati a vantaggi commerciali
- Orientamento verso particolari applicazioni
- Orientamento verso HW specializzato
- Diverse idee su cosa è 'piacevole' da usare

Linguaggi di programmazione



Cosa rende un linguaggio di successo?

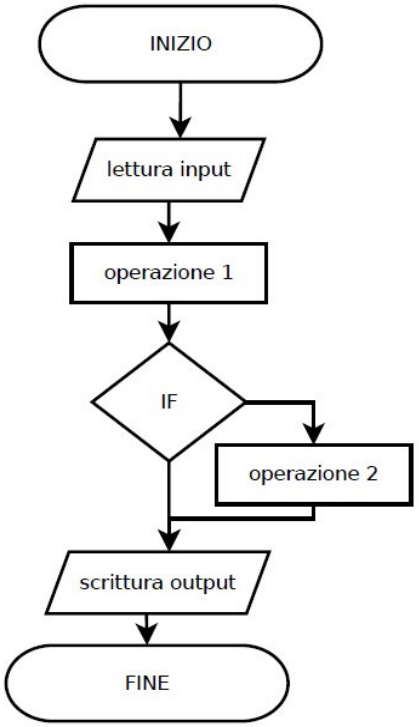
- Facilità di uso:
  - Facile da imparare (BASIC, Pascal, LOGO, Scheme)
  - Facile per esprimere le cose
  - Facile da usare e "potente" (C#, C++, Prolog, Lisp, APL, Algol-68, perl, php, Java, Phyton)
  - Facile da implementare (BASIC, Forth) e ottimizzato per l'esecuzione (veloce/compatto) (C, ForTran)

- Fattori di successo:
  - Un potente sponsor (COBOL, PL/1, Ada, Visual Basic)
  - Una diffusione enorme e costo basso (Pascal, Java, PHP, Perl, C, C++)

Paradigmi di Programmazione

Imperativo	
Procedurale (von Neumann)	ForTran, C, Basic, Pascal, Algol60
Object-Oriented	Simula67, C++, Smalltalk, Java, C#
Dichiarativo	
Funzionale	LISP, Scheme, ML, Haskell
Logica	Prolog, VisiCalc, RPG, spreadsheets

Programmazione e algoritmi



- Programmare significa scrivere lo svolgimento di un algoritmo in un linguaggio interpretabile poi dalla macchina.
- Un algoritmo è una sequenza ordinata e finita di passi (operazioni o istruzioni) elementari e interpretabili che conduce ad un ben determinato risultato in un tempo finito.

Concetti fondamentali di programmazione

- definizione di programmi
- dichiarazioni di variabili e tipi predefiniti
- istruzioni
  - assegnamento, istruzione condizionale, istruzione iterativa
  - I/O con Standard Input, Standard Output, file
- tipi definiti dall'utente
- array e record
- sottoprogrammi
  - funzioni e procedure
  - passaggio parametri (per valore/per indirizzo)
  - la visibilità degli elementi
  - il concetto di modulo

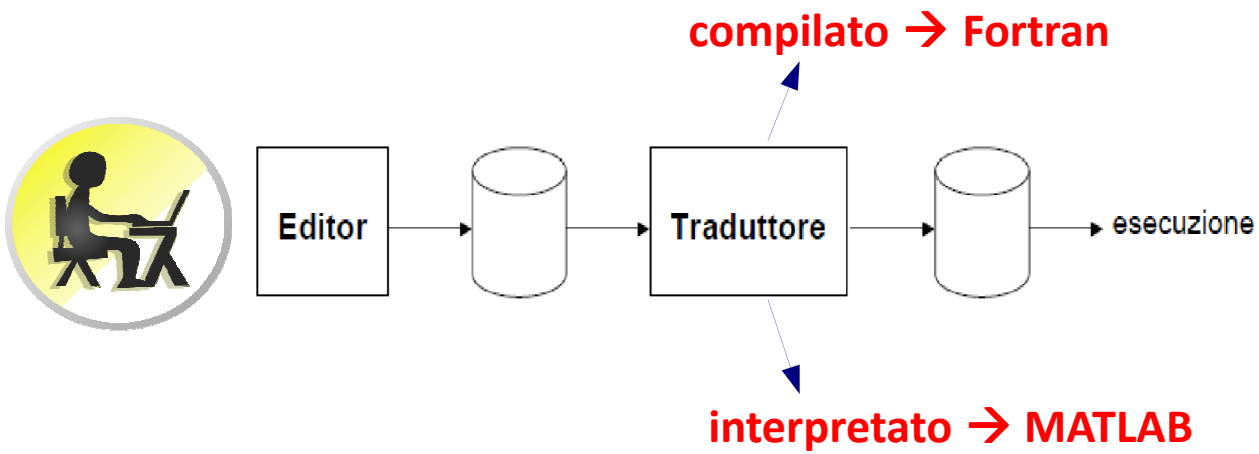
Fortran

- Il Fortran è un linguaggio adatto a svolgere calcoli scientifici e a sviluppare modelli numerici efficienti.
- È utilizzato da più di 50 anni per implementare modelli di centrali nucleari, nella progettazione di aerei, nell'elaborazione dei segnali sismici, nel campo dell'Idraulica.
- Il nome sta per IBM Mathematical Formula Translation System o più brevemente: FORmula TRANslation
- È il più vecchio linguaggio di programmazione tra tutti i linguaggi evoluti

Fortran vs MATLAB

- + circa 10 volte più veloce in esecuzione
- + possibilità di creare eseguibili portabili
- + compilatori gratuiti
- + linguaggio standard
- linguaggio formale di base (utile per imparare!!)
- quasi assenza di pacchetti grafici

Compilatore vs Interprete



Perché usare il Fortran?

- I programmatori utilizzano diversi 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.
- È importante scegliere il linguaggio con caratteristiche appropriate al tipo di problema da risolvere.
- Fra i linguaggi ad alto livello ricordiamo C++, Fortran, Pascal, Java, Python, Matlab.

Introduzione al FORTRAN

- FORTRAN = FORmula TRANslation
  - Nasce in IBM per programmare equazioni scientifiche
- Prima versione (IBM): anni 50
  - fu il primo linguaggio ad introdurre il concetto di compilatore
- Versioni successive:
  - FORTRAN II (1958)
  - FORTRAN IV (1962)
  - FORTRAN 77 (1977)
- Ulteriore versioni: Fortran 90-95-2003
- Fortran è in continua evoluzione: le ultime versioni sono studiate per facilitare la programmazione di algoritmi di calcolo paralleli in cui pezzi diversi dell'algoritmo sono eseguiti da macchine/processori diversi in parallelo

Struttura di un programma Fortran

- Ogni programma Fortran è 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.
- 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 è un'istruzione che indica al compilatore che non ci sono altre istruzioni da compilare all'interno del programma.

Il nostro primo programma

```
PROGRAM Mult
IMPLICIT NONE
INTEGER:: i, j , k
WRITE(*,*) ‘Introdurre due interi'
READ(*,*)  i, j
k=i*j
WRITE(*,*) 'Risultato = ', k
STOP
END PROGRAM Mult
```

! ← SEZIONE DICHIARATIVA

!Dichiarazioni di variabili

!Dichiarazione input & output

! ← SEZIONE ESECUTIVA

! Legge variabili da moltiplicare

! Moltiplicazione dei numeri

!Visualizza il risultato

! ← SEZIONE CONCLUSIVA

!fine del programma

Da notare

- Nella stesura di un programma è bene utilizzare commenti in abbondanza.
- Ovunque dopo il ! (punto esclamativo) si ha un commento. Si può pertanto inserire un commento dopo un'istruzione eseguibile.
- Il ! (punto esclamativo) in una stringa non è interpretato come commento.
  - PRINT*, "Ciao !! Tutto bene ?"
  - WRITE(*,*) " ..... " ! Abbiamo finito ?

Compilazione

- La compilazione trasforma le istruzioni Fortran (adatte agli umani) in linguaggio macchina (adatto ai computer)
- Prima di eseguire il codice, 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.
- Compilatori
  - Compaq Visual Fortran (sulle macchine dell'Università)
  - Compilatore Salford FTN95 per Windows (scaricabile dal sito http://www.silverfrost.com/)
  - Compilatore Intel per Linux
  - Compilatore gfortran per ambienti UNIX (Linux, MacOS e Windows)

L'insieme dei caratteri in Fortran

- Alfanumerici: a-z, A-Z, 0-9, e _ (Underscore)
- Simboli : space + * ( ) , ' : ! % > < ? = / . ; & $
- Il carattere 'space' (spazio bianco) è significativo nei seguenti contesti:
  - Tra parole chiave:
    - INTEGER :: uno ! corretto
    - INT EGER :: uno ! errato
  - Tra nomi di variabili:
    - REAL :: nome_uno ! corretto
    - REAL :: nome uno ! errato
- In particolare gli spazi bianchi debbono assolutamente 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).

Tipi predefiniti

Il Fortran ha tre classi di tipi predefiniti:

- Tipo carattere
  - CHARACTER :: sei ! Singola Lettera
  - CHARACTER(LEN=15):: nome ! Stringa
- Tipo logico
  - LOGICAL :: vero ! Vero/Falso
- Tipo numerico
  - REAL :: pi ! 3.141592
  - INTEGER :: codice ! Intero
  - COMPLEX :: val ! X + i Y

Dati REAL

- I dati di tipo REAL sono costanti e variabili in formato reale. Posso quindi rappresentare numeri razionali
- Es:
  - 10.0
  - -999.999
  - 123.456789E20
  - 0.123E-3

Dati INTEGER

- I dati di tipo INTEGER sono costanti e variabili intere
- Esempio:
  - 0
  - -999
  - 123456789
  - +17

Dati CHARACTER

- I dati di tipo CHARACTER sono costituiti da stringhe di caratteri alfanumerici
- Esempio:
  - 'THIS IS A TEST'
  - ' '
  - “3.141593”

Nomi di VARIABILI e PROCEDURE

- In Fortran 90 nomi di variabili e di procedure devono iniziare con una lettera
  - REAL :: pippo123 ! Corretto
  - REAL :: 123pippo ! Errato
- Si possono utilizzare solo lettere, cifre da 0 a 9 ed il carattere _ (underscore)
  - CHARACTER :: a_zeta ! Corretto
  - CHARACTER :: a-zeta ! Errato
- Un nome non può essere più lungo di 31 caratteri

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 sconsigliato, si preferisce pertanto eliminare questa possibilità con il comando:
  - IMPLICIT NONE
- 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 >]
    - REAL :: x
    - INTEGER :: i, j =4
    - LOGICAL, POINTER :: p
    - REAL, DIMENSION(4,4) :: a,b ! Due Matrici REALI di dimensione (4 x 4)

Dichiarazioni di tipi carattere

- Hanno una sintassi simile alle variabili numeriche: ci si può riferire ad un solo carattere; ci si può 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

  Nell'ultimo esempio A è una matrice(3x3) e ogni elemento della matrice è una stringa di lunghezza 5

Costanti Parametriche

- Costanti simboliche, note come parametri, in FORTRAN possono essere definite o tramite un attributo o tramite l'istruzione PARAMETER.
  - REAL , PARAMETER :: PIG = 3.141592
  - CHARACTER(LEN=*) , PARAMETER :: nome ="Luigi"

  Nell'ultimo 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

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.
- Esempio, supponiamo che : 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 ha come risultato sempre un valore logico.

Operatori predefiniti (segue)

- Operatore di concatenazione di stringhe: Stringa1 // Stringa2
- Quando si usano le stringhe si può far riferimento ad una sottostringa con la sintassi stringa(inf:sup)
  - stringa1 è "TOPO"
  - stringa1(1:1) è "T" ! stringa1(1) sarebbe errato
  - stringa1(2:4) è "OPO"
  - CHARACTER (LEN=*) , PARAMETER :: str1="Pepe"
  - CHARACTER (LEN=*) , PARAMETER :: str2="rino"
  - WRITE (*,*) str1//str2 ! Output → "Peperino"
  - WRITE (*,*) str1(2:3)//str2(2:2) ! Output → "epi"

Precedenza degli operatori (dal più alto al più basso)

Operazione	Operatore	Esempi
Elevemento a potenza	****	2**10 → 1024
Moltiplicazione e divisione	* /	A*B A/B
Segno (unario)	+ -	+3 -4
Somma e sottrazione (binario)	+ -	A+B A-B
Concatenazione stringhe	//	Str1//Str2
Confronto	= =/ > >= < <=	A>=B A<B A=/B
Negazione logica	.NOT.	.NOT.A
AND Logico	.AND.	A.AND.B
OR Logico	.OR.	A.OR.B
Dis/Uguaglianza Logica	.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.
  - L'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 da fare al computer

- Scrivere un codice che divida due numeri interi, che moltiplichi due numeri reali e che restituisca due variabili carattere unite.
- Utilizzando il codice di cui al punto 1, eseguire le seguenti operazioni: 3/4, 5/4, 3./4., 3/4., 3./4.
- Scrivere un codice che calcoli circonferenza e area del cerchio e il volume della sfera di assegnato raggio R.