Linguaggi
e
paradigmi di programmazione

Linguaggi di programmazione

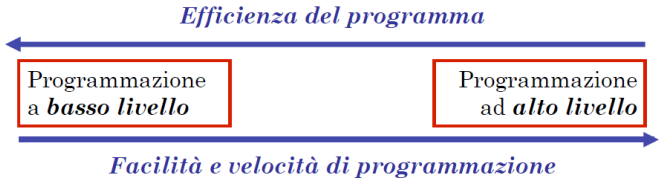
- Codificare un programma utilizzando il *linguaggio macchina* è assai arduo e richiede una conoscenza approfondita del funzionamento di un particolare calcolatore (o meglio: del microprocessore che costituisce la CPU della macchina).
- Per ovviare a questo problema, che ha costituito per molti anni un grosso limite alla diffusione della programmazione e quindi anche dell'uso dei calcolatori, sono stati sviluppati dei **linguaggi di programmazione più evoluti**, che si pongono a metà strada fra il nostro linguaggio naturale ed il linguaggio macchina.
- Sono **semplici** e **poveri** (poche parole chiave, poche regole), ma **privi di qualsiasi ambiguità**.

Linguaggi di programmazione

- In informatica si parla di **programmazione a basso livello** quando si utilizza un linguaggio molto vicino alla macchina, al suo funzionamento interno.
- Si parla invece di **programmazione ad alto livello** quando si utilizzano linguaggi più sofisticati ed astratti, slegati dal funzionamento fisico della macchina.
- Si viene così a creare una **gerarchia di linguaggi**, dai meno evoluti (il *linguaggio macchina* e l'*assembler*) a quelli più evoluti (*Pascal*, *Fortran*, *Cobol*, *Perl*, *Java*);
- Il **linguaggio C** si pone ad un livello intermedio.

Linguaggi di programmazione

- La programmazione a *basso livello* è più ardua e meno intuitiva, ma consente di sviluppare programmi molto efficienti su uno specifico sistema hardware/software, sfruttandone a fondo le caratteristiche.
- Ad *alto livello* la programmazione è più "naturale" e rapida, ma è possibile che non consenta di produrre software particolarmente efficiente.



Linguaggi a basso livello

- **I problemi dei linguaggi macchina**
  - **Sono specifici della macchina** perchè ogni CPU ha il proprio linguaggio macchina.
  - Occorre conoscere l'architettura della macchina per scrivere programmi.
  - I programmi **non sono portabili**.
  - I codici sono **illeggibili** all'uomo.
- **I linguaggi Assembly**

  Rispetto al linguaggio macchina:
  - **E' un vero e proprio linguaggio di programmazione**
  - **Più semplice la programmazione** perché permette di usare nomi simbolici sia per i codici operativi delle istruzioni sia per le locazioni di memoria al posto di lunghe sequenze di bit.
  - **Maggiore leggibilità**
  - Richiede tuttavia di un **processo di traduzione** da Assembly a linguaggio macchina, ma a differenza dei linguaggi ad alto livello la corrispondenza è 1:1 (ad ogni istruzione in linguaggio Assembly corrisponde una sola istruzione in linguaggio macchina).
  - Permane la **non portabilità dei programmi**, perché le istruzioni restano strettamente dipendenti dal processore.

Vantaggi e svantaggi dell'assembler

- Vantaggio: la dipendenza dall'architettura del calcolatore permette
  - Ottimizzazione delle prestazioni (maggiore *efficienza*)
  - Programmi (potenzialmente) più compatti
  - Massimo sfruttamento delle potenzialità dell'hardware sottostante
  - Importante per
    - ✓ Programmare controller di processi e macchinari (anche real-time)
    - ✓ Programmazione di apparati limitati (embedded computer, dispositivi portatili, telefonini cellulari, …)
- Principali svantaggi della programmazione
  - Strutture di controllo in forme limitate (minore espressività)
  - Necessario conoscere i dettagli dell'architettura
  - Mancanza di portabilità su architetture diverse
  - Difficoltà di comprensione, possibile lunghezza maggiore, facilità di errore rispetto a programmi scritti in linguaggio ad alto livello (minore produttività del programmatore)

Benefici dei linguaggi ad alto livello

- Notazione vicina al linguaggio corrente ed alla notazione algebrica (maggiore *espressività* e *leggibilità*)
- Incremento di *produttività*
  - Facilitano la programmazione, svincolandola dalla conoscenza dei dettagli architetturali della macchina utilizzata
- Indipendenza dalle caratteristiche peculiari dell'architettura (processore) su cui il programma va eseguito (*portabilità*)
  - Ideati non per essere compresi direttamente da macchine reali, ma da macchine astratte, in grado di effettuare operazioni più di alto livello rispetto alle operazioni elementari dei processori reali
- Permettono l'uso di librerie di funzionalità già scritte (riusabilità del codice)

Paradigmi di programmazione

Un **paradigma di programmazione** è l'insieme degli strumenti concettuali forniti da un determinato linguaggio per la codifica di un programma e definisce il modo con cui il programmatore concepisce il programma stesso.

Dal momento che un linguaggio realizza un determinato paradigma di programmazione se consente di scrivere programmi in accordo con esso, i paradigmi costituiscono un metodo di classificazione dei linguaggi.

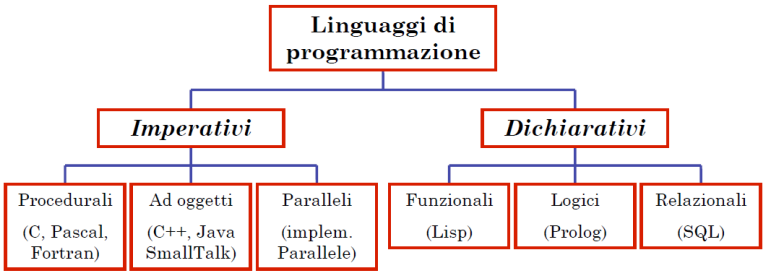
Ad oggi i paradigmi più utilizzati sono i seguenti:

- **Programmazione imperativa**
- **Programmazione dichiarativa**

In ciascuno di questi paradigmi si possono individuare linguaggi che, a seconda dell'approccio che offrono per la risoluzione di un problema, vengono classificati secondo ulteriori sotto-paradigmi.

Tipi di linguaggio di programmazione

- Possiamo aggregare i numerosi linguaggi di programmazione esistenti sulla base del **modello astratto di programmazione** che sottintendono e che è necessario adottare per utilizzarli.



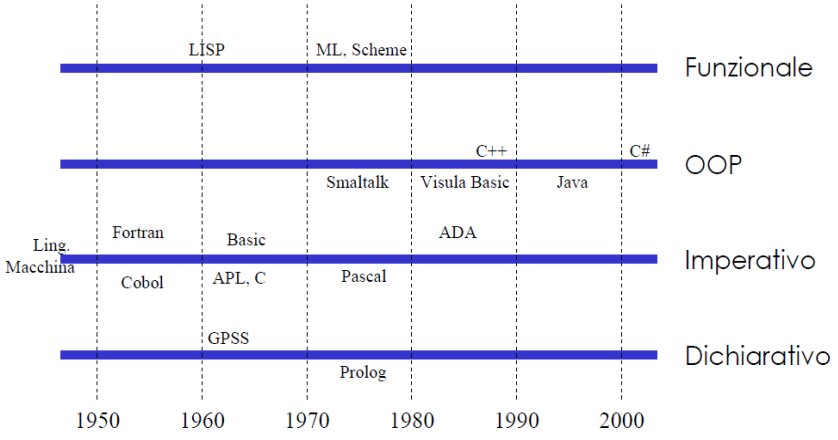
Tipi di linguaggio di programmazione

- **Linguaggi imperativi**:
  - Il modello computazionale è basato sui *cambiamenti di stato della memoria* della macchina.
  - È centrale il concetto di *assegnazione di un valore* ad una locazione di memoria (*variabile*).
  - Il compito del programmatore è costruire una sequenza di assegnazioni (spesso reiterandole più volte) che producano lo stato finale (in modo tale che questo rappresenti la soluzione del problema).
- **Linguaggi dichiarativi**:
  - Il modello computazionale è basato sui concetti di *funzione* e *relazione*.
  - Il programmatore non ragiona in termini di assegnazioni di valori, ma di *relazioni fra entità* e di *valori di una funzione*.

Tipi di linguaggio di programmazione 3

- Sulla base dell'*ambito* in cui è necessario risolvere il problema, è opportuno adottare un linguaggio piuttosto che un altro:
  - **Calcolo scientifico**: Fortran, C
  - **Intelligenza Artificiale**: Prolog, Lisp
  - **Applicazioni gestionali**: Cobol, SQL, C, C++, C#
  - **Sistemi, *device driver***: Assembler, C
  - **Applicazioni *client* visuali**: Java, Visual Basic, C++, C#, C
  - **Applicazioni su Web**: Perl, PHP, ASP, Java, JavaScript, C#
  - **Applicazioni distribuite**: Java, C, C++, C#
  - **Applicazioni mobile**: Java, JavaScript, C++, C#

Evoluzione dei paradigmi di programmazione



Evoluzione delle tecniche di programmazione