Università degli Studi Roma Tre – Dipartimento di Matematica e Fisica – Corso di Laurea in Matematica

Appunti del corso di Informatica 1 (IN110)

4 – Linguaggi di programmazione

Prof. Marco Liverani

(liverani@mat.uniroma3.it – http://www.mat.uniroma3.it/users/liverani/IN110)

Sommario

- Linguaggi di programmazione
- Tipi di linguaggio di programmazione
- Compilatori ed interpreti

Linguaggi di programmazione 1

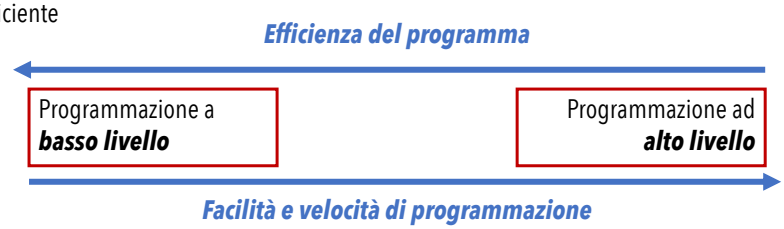
- Un **linguaggio di programmazione** è costituito, come ogni altro tipo di linguaggio, da un **alfabeto** con cui viene costruito un insieme di **parole chiave** (il *vocabolario*) e da un insieme di **regole sintattiche** (la *grammatica*) per l'uso corretto delle parole del linguaggio
- I microprocessori presenti all'interno della macchina sono stati progettati per riconoscere ed eseguire un insieme piuttosto ristretto di istruzioni; tali istruzioni costituiscono il cosiddetto **linguaggio macchina**
- Il linguaggio macchina è basato su una codifica estremamente compatta e poco intuitiva

Linguaggi di programmazione 2

- 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**

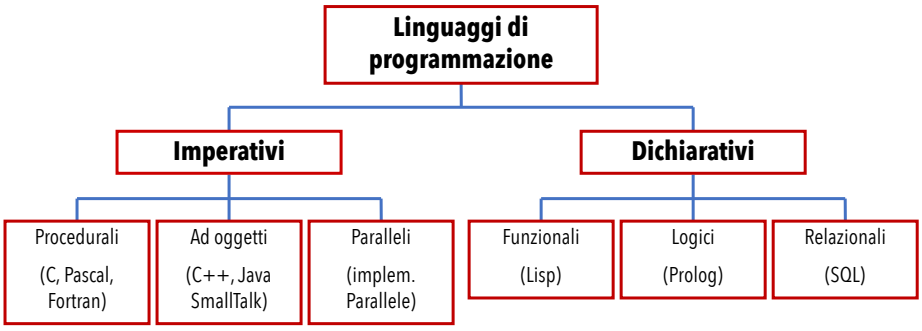
Linguaggi di programmazione 3

- 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*, *Java, Perl*, *Python*, *Go, ...*)
- Il **linguaggio C** si pone ad un livello intermedio
- 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



Tipi di linguaggio di programmazione 1

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



In effetti uno stesso linguaggio di programmazione è spesso basato su più paradigmi, rendendo ben più complessa la classificazione; questo elenco ne è un esempio:

`https://en.wikipedia.org/wiki/List_of_programming_languages_by_type`

Tipi di linguaggio di programmazione 2

- **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*
- **Linguaggi a oggetti**:
  - Il modello computazionale è basato sul concetto di **classe** (insieme) di **oggetti** con cui sono definite le strutture dati per la rappresentazione delle informazioni su cui opera il programma
  - Le funzioni e le procedure sono **metodi** definiti per operare su specifici oggetti
  - Si basano sui concetti di **incapsulamento** (un oggetto presenta una interfaccia con cui può essere usato e non rende visibile la propria implementazione), **ereditarietà** (le classi di oggetti possono essere definite estendendo le caratteristiche di altre classi di oggetti), **polimorfismo** (uno stesso metodo può avere comportamenti diversi su oggetti di classi differenti)

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, Python, Mathematica, Matlab
  - **Intelligenza Artificiale**: Prolog, Lisp, Python
  - **Applicazioni gestionali**: Cobol, SQL, C, C++, C#
  - **Sistemi, *device driver***: Assembler, C
  - **Applicazioni *client* visuali**: Java, JavaScript, Visual Basic, C++, C#, C
  - **Applicazioni su Web**: Perl, PHP, Python, ASP/ASPX, Java, JavaScript, C#, Ruby
  - **Applicazioni distribuite**: Java, C, C++, C#
  - **Applicazioni mobile**: Java, JavaScript, C++, C#, Swift

Compilatori ed interpreti 1

- Un **programma** è la ***codifica*** di un algoritmo eseguita utilizzando un linguaggio di programmazione
- L'unico linguaggio che la macchina è in grado di interpretare è il *linguaggio macchina*
- Per eseguire un programma scritto in un linguaggio di alto livello **è necessario dunque tradurlo in linguaggio macchina**
- Naturalmente è possibile eseguire la **traduzione in modo automatico** utilizzando un programma "traduttore" (un ***compilatore*** o un ***interprete***)
- Ogni traduttore è in grado di interpretare e tradurre *un solo* linguaggio (compilatore C, interprete Python, compilatore Fortran, ecc.)

Compilatori ed interpreti 2

- **Interprete**: itera più volte questo processo
  - *Legge* un'istruzione del programma "sorgente"
  - *Traduce* l'istruzione in linguaggio macchina
  - *Esegue* l'istruzione
  - Passa all'interpretazione dell'*istruzione successiva*
- Al termine di questa operazione, del programma in linguaggio macchina non rimane alcuna traccia (la traduzione *non viene memorizzata*)
- Se il programma torna più volte su una stessa istruzione, questa verrà tradotta (ed eseguita) *ogni* volta
- È necessario disporre dell'interprete per poter eseguire il programma

Compilatori ed interpreti 3

- **Compilatore**: esegue una sola volta il processo
  - *Legge tutte* le istruzioni del *programma "sorgente"*, ne verifica la correttezza sintattica e le traduce in linguaggio macchina
  - *Memorizza* su disco il *programma "eseguibile"* tradotto in linguaggio macchina
- Al termine della compilazione avremo un programma "eseguibile" in linguaggio macchina
- La traduzione di ogni istruzione del programma avviene *una sola volta*, anche se una stessa istruzione viene ripetuta più volte all'interno del programma
- Non ho bisogno di avere il compilatore ed il "sorgente" per *eseguire* il programma: mi basta il programma "eseguibile"

Compilatori ed interpreti 4

- Il linguaggio macchina è diverso a seconda del microprocessore (il tipo di CPU) presente nel calcolatore
- Quindi i programmi eseguibili (in linguaggio macchina) sono strettamente dipendenti dalla **piattaforma hardware** (oltre che dal particolare **sistema operativo** adottato)
- I linguaggi di alto livello sono simili (esistono delle differenze) tra piattaforme differenti
- Un grande problema è la **portabilità** del software da una piattaforma ad un'altra:
  - Se le piattaforme sono uguali o compatibili potrò distribuire il programma in formato eseguibile
  - Se le piattaforme sono incompatibili dovrò distribuire il programma in formato sorgente