**Quarta lezione di programmazione 1  (Traduttori)**

**Linguaggi di programmazione**

Si utilizzano per codificare gli algoritmi. Un linguaggio di programmazione è  un insieme di istruzioni che vengono utilizzate per codificare programmi. Distinguiamo due tipologia di linguaggi di programmazione:

-   **Linguaggi a basso livello:**

o  **Linguaggio macchina:**

o  **Assembly:** è molto simile al linguaggi macchina con la differenza che si utilizzano forme mnemoniche per le corrispondenti istruzioni macchina.

-   **Linguaggi a alto livello(si avvicinano al nostro modo di parlare):** ha i seguenti vantaggi:

·  **astrazione**

·  **semplificazione**

·  **similarità con il ragionamento umano**

**Paradigmi di programmazione**

**I paradigmi sono la filosofia con cui si scrivono i programmi.**

Un linguaggio di programmazione è basato su un paradigma di programmazione che è caratterizzato da:

-   **metodologia**  con cui si scrivono i programmi

-   **computazione**  istruzioni o funzioni matematiche

Esistono svariati paradigmi di programmazione come:

-   **programmazione funzionale:**  è costituita da valutazione funzione matematiche. Si ha un assenza di side-effect perché non si fa uso di istruzioni imperative (non si posso fare modifiche alla memoria)

-   **programmazione logica:** descrive la struttura logica del programma (logica del primo ordine)

-   **programmazione imperativa:** sequenza di istruzioni

-   **programmazione strutturata :** è basata sul teorema di Bohm Jacobini (**tre costrutti sequenza, selezione e iterazione)**.L'istruzione  Go to permette di fare dei salti in alcune parti del programma.

-   **programmazione procedurale:** blocchi di codice sorgente identificati da un nome ed eventuali argomenti.

-   **programmazione modulare**:  strutturare un programma in moduli ben definiti. mettendo insieme queste funzioni si può pensare di aggregarle logicamente in un modulo. Il modulo è fatto da un certo set di funzioni, alcune di queste saranno dedicate al funzionamento interno del  modulo quindi alle computazioni che avvengono all'interno che non sono visibili all'utente finale. Altre di queste funzioni costituiranno la cosiddetta interfaccia del modulo sono funzioni che si potranno invocare esplicitamente attraverso le quali potremmo attivare certe computazioni.

-   **programmazione orientata agli oggetti(OOP):**Introduce una serie di costrutti. la classe è una estrazione della realtà invece l'oggetto è un'istanza della classe. I metodi sono le funzionalità dell'oggetto. con la programmazione ad oggetti sia un legame tra i  dati e i metodi che vengono incapsulate nelle classi. Si ha anche la possibilità di avere dei metodi che non sono visibili all'utente cosiddetti metodi privati che servono solo per l'implementazione dell'oggetto.

Immagine che contiene testo, schermata, Carattere, design

Descrizione generata automaticamente

**I linguaggi di programmazione sono definiti da un insieme di regole formali cioè la sintassi che include tutti i simboli da utilizzare durante la scrittura del programma. Le regole di sintassi definiscono come comporre i simboli e le parole per comporre istruzioni corrette. La semantica di un'istruzione è il significato di questa .  attenzione!!  se in un programma non ci sono errori sintattici non vuol dire che la semantica del programma.**

**Traduzione**Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamente

Grazie al concetto di traduzione si ha:

* **evoluzione del linguaggio di programmazione**( Maggiormente espressivi)
* **facilità nella scrittura del programma** perché sono vicini al linguaggio parlato
* **Indipendenza dalla piattaforma:** Il programmatore che sta programmando in c non si deve preoccupare della piattaforma(CPU).  Questo problema se la deve porre chi scrive e progetta i compilatori  per le varie CPU.

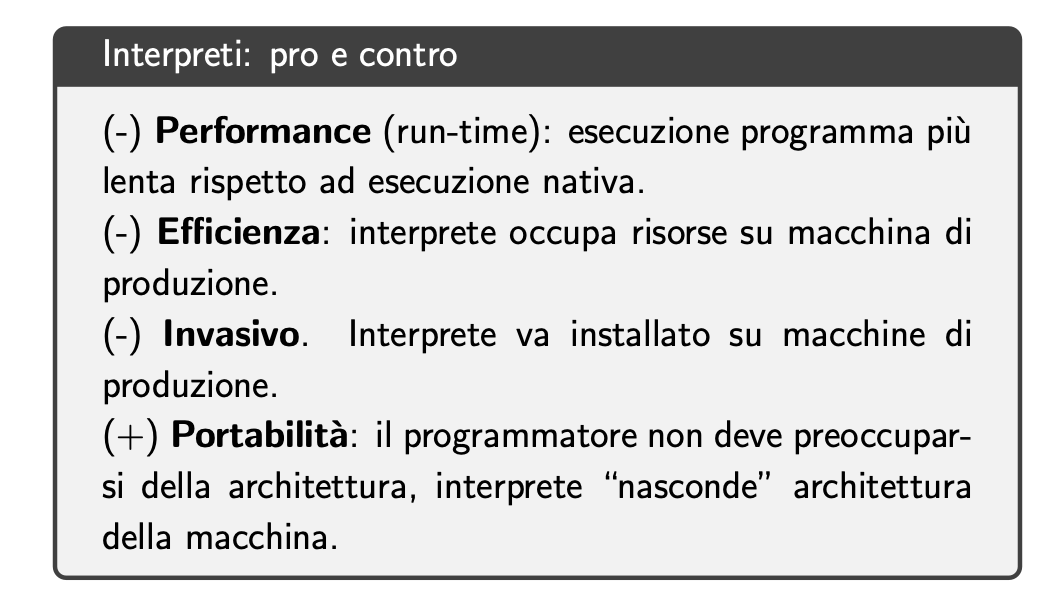
**si hanno due tipi di traduttori:**

1. **L'interpretazione** traduce ed esegue ogni singola istruzione del programma. Legge ed esegue il codice sorgente del programma senza creare un file oggetto eseguibile. E' più lenta rispetto alla compilazione**.**
2. **compilatore:**  il compilatore prende in pasto un programma scritto con un linguaggio alto livello e lo traduce in linguaggio macchina. Il compilatore opera su una architettura che si dice architettura " target ” se su una macchina si installa il compilatore e si va a compilare un programma si ottiene un programma eseguibile per qualunque altra macchina che abbia la stessa architettura.  Se invece il programmatore Dovesse compilare questo programma per architetture diverse dall'architettura della macchina di sviluppo, allora Dovrebbe effettuare una procedura  chiamata cross compilazione.

Nel processo di traduzione del compilatore si hanno almeno due parti:

1. **back end**:  è la parte che cambia da architettura ad architettura e si occupa di generare  istruzioni macchina
2. **front end**: ha la funzione di eseguire l'analisi sintattica del programma di fare il parsing.

I linguaggi interpretati e compilati hanno dei vantaggi degli svantaggi:



Nei linguaggi interpretati il programmatore **non si deve preoccupare nell'architettura su cui verrà eseguito il programma.** infatti il programmatore scrive il programma in Java la compilazione produce un **programma in byte code**  che è un linguaggio simile al linguaggio macchina ma è indipendente dall'architettura e **il  byte code  viene eseguito mediante un interprete Java installato su una macchina.**

**JVM  è la macchina virtuale che interpreta il codice Byte code**