

linguaggi di programmazione

Matteo

December 18, 2025

ricorda sul libro per modulo 1 capitoli da 1 a 5

1 intro

linguaggi imperativi , linguaggi dichiarativi, funzionali e logici linguaggi imperativi

- basato sulla nozione di stato, le istruzioni sono comandi che cambiano lo stato
- stato= insieme di locazioni di memoria contenente valori
- generalmente di basso livello

linguaggi dichiarativi

- basati sulla nozione di funzioni o relazione
- le istruzioni sono dichiarazioni di nuovi valori

linguaggi funzionali

- basati sulla nozione di funzione: risultato di un programma = valore esplicito di una espressione
- ricorsione
- programmare= costruire una funzione

linguaggi logici

- basati sulla nozione di relazione: risultato di un programma = insieme di valori di variabili determinato da relazioni
- istruzioni = implicazioni logiche fra opportune formule , che possono essere viste come regole di riscrittura
- programmare= definire la relazione che definisce il valore delle variabili di interesse

linguaggi orientati agli oggetti: ora molto usati, sono linguaggi imperativi con alcune metodologie dichiarative. Oggetti (istanze di opportune classi) che contengono i dati (concetti imperativi) e metodi come funzioni per operare su tali oggetti (concetti dichiarativi).

2 Macchine astratte

pag 28 : Molti molto complesso da tradurre in una macchina fisica ad alto livello, quindi vengono costruite macchine fisiche solo per linguaggi di basso livello

altimenti crei un programma che traduce L e i suoi costrutti in L', un linguaggio già esistente implementabile su una macchina già esistente; minore velocità rispetto al caso precedente

altrimenti anziché programmi usi microprogrammi o firmware che traducono L in un linguaggio di basso livello e usano registri di sola lettura, garantendo buone prestazioni

pag 31 impl. inter. pura : non è vera traduzione, I fa corrispondere a una certa parte di L una certa parte di L₀, poco efficiente, I deve decodificare al momento. più flessibile perché permette di interagire direttamente con l'esecuzione del programma e gli interpreti sono più veloci da creare

impl. comp. pura: vera traduzione dal compilatore che avviene prima dell'esecuzione del programma. alcune informazioni del programma sorgente vanno perse che rende più difficile interagire con il programma in tempo reale

vantaggi e svantaggi interpretazione:

- vantaggio: maggiore flessibilità, permette di interagire con l'interazione del programma.
- vantaggio : più veloce da realizzare
- vantaggio : occupa meno memoria, non generando nuovo codice, problema meno sentito oggi.
- svantaggio : la compilazione interpretativa è meno efficiente perché deve effettuare al momento dell'esecuzione un'interpretazione dei costrutti di L. con diverse occorrenze dello stesso costrutto si richiedono ulteriori decodifiche ogni volta

vantaggi e svantaggi compilazione:

- vantaggio: il compilatore deve compilare una sola volta all'inizio poi i costrutti non devono essere decodificati ogni volta
- vantaggio: maggiore efficienza
- svantaggio: perdita del codice sorgente. se ci fosse un problema sarebbe difficile capire da dove si origina l'errore

siamo a pag 35 di fatto nei linguaggi reali sono presenti entrambe le tecniche. ad esempio abbiamo linguaggio L, compilato in linguaggio intermedio L_i, interpretato in L se l'interprete della macchina intermedia è molto diverso dal linguaggio finale L₀ diremo che è un'implementazione interpretativa, se è quasi uguale è un'implementazione compilativa.

la differenza tra completamente interpretativa e mostly interpretativa è che non tutti i costrutti devono essere simulati

per impl. mostly compilativa alcune funzionalità devono essere simulate perché L_i non trova un corrispondente immediato in molto

ci sono tanti spettri intermedi, di solito si predilige flessibilità

di solito ci sono più livelli di macchine astratte con i loro linguaggi e funzionalità. praticamente tutto nell'informatica è creato attraverso una gerarchia di macchine astratte (programmi e linguaggi).

ogni livello accede a quello inferiore e aggiunge nuove funzionalità, in un certo senso creando un nuovo linguaggio.

Ciò permette un certo dominio sulla complessità del sistema e l'indipendenza tra i livelli.

macchina astratta: una formalizzazione astratta di un generico esecutore di algoritmi formalizzato attraverso un linguaggio di programmazione

interprete: un componente essenziale della macchina astratta che ne caratterizza il comportamento, mettendo in relazione "operazionale" il linguaggio della macchina astratta col mondo fisico circostante

3 descrivere i linguaggi di programmazione