

Evoluzione dei linguaggi di programmazione

Fondamenti di architettura e programmazione
Corso di laurea in Comunicazione digitale

La traduzione dei linguaggi

- Evoluzione verso sistemi di codici complessi e potenti, orientati più all'uomo che alla macchina
- La sfida degli anni '50 sulla traduzione dei linguaggi
- Linguaggi ad **alto livello** e a **basso livello** ovvero *i linguaggi macchina*

AA 2008/09
© Alberti

2

Programmazione
4. Evoluzione dei linguaggi di programmazione

I linguaggi anni '50

- I primi linguaggi ad alto livello
 - **FORTRAN**, introduce il concetto di sottoprogrammi che operano su dati comuni
 - **ALGOL**, introduce il concetto di struttura dei programmi e di procedure *ricorsive*, ovvero che richiamano sè stesse
 - **COBOL**, introduce il concetto di FILE e di descrizione dei dati

AA 2008/09
© Alberti

3

Programmazione
4. Evoluzione dei linguaggi di programmazione

I linguaggi anni '60

- Notazioni per la descrizione dei linguaggi
- Meno enfasi sull'efficienza, attenzione al modello di computazione
- I linguaggi orientati al problema
 - **LISP**, uniformità tra dati e programmi e un paradigma di programmazione basato sul concetto di funzione
 - **APL**, linguaggio matematico, ricco di notazioni e operatori per operare su strutture come vettori e matrici
 - **SNOBOL**, offre strumenti utili per la manipolazione di sequenze di caratteri

AA 2008/09
© Alberti

4

Programmazione
4. Evoluzione dei linguaggi di programmazione

Evoluzione dei linguaggi – '70

- Metodologia di programmazione
 - **PASCAL**, ha lo scopo di insegnare la programmazione strutturata, una possibile risposta alla necessità di programmare con un metodo. Segue **Modula-2** che introduce il concetto di modulo.
 - **C**, linguaggio ad alto livello con visibilità e accesso alla macchina
 - **Prolog**, linguaggio basato sulla logica, non convenzionale, diventato di nicchia

AA 2008/09
© Alberti

5

Programmazione
4. Evoluzione dei linguaggi di programmazione

I linguaggi anni '80

- Programmazione in grande, esigenza di modularità e di astrazione
 - La programmazione a oggetti. Una classe definisce un insieme di oggetti e le procedure per manipolarli
 - **SmallTalk**, **Objective-C**, **C++**

AA 2008/09
© Alberti

6

Programmazione
4. Evoluzione dei linguaggi di programmazione

I paradigmi di programmazione

- Forniscono la filosofia e la metodologia con cui si scrivono i programmi
- I linguaggi devono *consentire* ma soprattutto *spingere* all'adozione di un particolare paradigma
 - Procedurale
 - Funzionale
 - Modulare
 - Orientato agli oggetti

AA 2008/09
© Alberti

7

Programmazione
4. Evoluzione dei linguaggi di programmazione

Paradigma procedurale

- Enfasi sulla soluzione algoritmica dei problemi mediante modifica **progressiva** dei dati in memoria
 - Esecuzione sequenziale di istruzioni
 - Cambiamento dello stato di memoria (le variabili) tramite assegnamento
 - Programmazione per effetto collaterale (*side-effect*)
- Aderenti al modello della macchina di von Neumann
- Molto efficienti
- Ha mostrato limiti nello sviluppo e mantenimento di sw complessi
- I linguaggi **imperativi**: **Pascal**, **C**

AA 2008/09
© Alberti

8

Programmazione
4. Evoluzione dei linguaggi di programmazione

Influenza del modello di macchina

- **Concetto di istruzione**
 - l'unità di base del programma, memorizzata in successive celle di memoria
- **Concetto di sequenzialità e iterazione**
 - Il programma assolve il compito eseguendo le istruzioni in sequenza
 - Presente in diversi costrutti dei linguaggi e in tutto il processo di esecuzione
- **Concetto di variabile e di assegnamento**
 - Le celle di memoria hanno un indirizzo e contengono i dati da manipolare
 - Le variabili hanno un nome e un valore
 - L'assegnamento di un valore a una variabile equivale al trasferimento di un dato in una cella

AA 2008/09
© Alberti

9

Programmazione
4. Evoluzione dei linguaggi di programmazione

È sorprendente che il computer di Von Neumann sia rimasto così a lungo il paradigma fondamentale dell'architettura dei calcolatori.

Ma dato il fatto, non è sorprendente che i linguaggi imperativi siano i principali oggetti di studio e sviluppo.

Perché come Backus ha sottolineato i linguaggi imperativi hanno solide radici nell'architettura della macchina di Von Neumann e ne sono l'immagine.

Horowitz *Fundamentals of Programming Languages*, 1983

AA 2008/09
© Alberti

10

Programmazione
4. Evoluzione dei linguaggi di programmazione

La macchina di Von Neumann

- Il calcolatore dello I.A.S. (Princeton, 1952): Von Neumann, Goldstein, Burks ...
- Primo modello con programmazione e memorizzazione del programma
 - Organo aritmetico-logico (oggi CPU o data-path)
 - Memoria
 - Organo di controllo
 - Organo per la gestione input/output

AA 2008/09
© Alberti

11

Programmazione
4. Evoluzione dei linguaggi di programmazione

Paradigma funzionale

- Primo tentativo di non rifarsi al modello di macchina di von Neumann
- La computazione avviene tramite funzioni che applicate ai dati riportano nuovi valori
 - Ogni funzione è un modulo a sé, dipendente unicamente dal valore dei suoi argomenti
 - L'effetto globale è ottenuto concatenando opportunamente funzioni anche richiamando sé stesse (*ricorsione*)
 - Modello che si rifà alla teoria delle funzioni ricorsive
 - Scarso supporto ai costrutti di ripetizione tramite iterazione
- **Lisp**, **ML**

AA 2008/09
© Alberti

12

Programmazione
4. Evoluzione dei linguaggi di programmazione

Componenti dei linguaggi funzionali

- Un insieme di funzioni primitive
- Una legge di **composizione** di funzioni
- Una legge di **applicazione** di una funzione ai suoi argomenti
- Un insieme di oggetti su cui operare

AA 2008/09
© Alberti

13

Programmazione
4. Evoluzione dei linguaggi di programmazione

Paradigma modulare

- Introduce il concetto di modulo che nasconde i dati all'utente
- I dati possono essere letti solo tramite un'opportuna interfaccia
- **Modula-2, Ada**

AA 2008/09
© Alberti

14

Programmazione
4. Evoluzione dei linguaggi di programmazione

Esempi

- Il problema di sommare i numeri dispari in un insieme dato
 - In **Pascal** `sm_dispari_pascal`
 - In **APL** `sm_dispari_apl`
 - In **Lisp** `sm_dispari_lisp`
 - In **Logo** `sm_dispari_logo`
 - In **Forth** `sm_dispari_forth`

I file sono raggiungibili dalla pagina:
<http://homes.dico.unimi.it/~alberti/Fond08/Lucidi/Intro/ling.html>

AA 2008/09
© Alberti

15

Programmazione
4. Evoluzione dei linguaggi di programmazione