

Linguaggi di Programmazione

- ☛ paradigmi linguistici, costrutti
- ☛ semantica
- ☛ implementazione, strutture a tempo di esecuzione

1

Linguaggi di programmazione e astrazione

- ☛ i linguaggi di programmazione ad alto livello moderni sono il più potente strumento di astrazione messo a disposizione dei programmatori
 - che possono, con un solo costrutto del linguaggio, “rappresentare” un numero (anche infinito) di interminabili sequenze di istruzioni macchina corrispondenti
- ☛ i linguaggi si sono evoluti trasformando in costrutti linguistici (e realizzando una volta per tutte nell’implementazione del linguaggio)
 - tecniche e metodologie sviluppate nell’ambito della programmazione, degli algoritmi, dell’ingegneria del software e dei sistemi operativi
 - in certi casi perfino in settori di applicazioni (basi di dati, intelligenza artificiale, simulazione, etc.)
- ☛ di fondamentale importanza è stata l’introduzione nei linguaggi di vari meccanismi di astrazione, che permettono di
 - estendere il linguaggio (con nuove operazioni, nuovi tipi di dato, etc.) semplicemente scrivendo dei programmi nel linguaggio stesso

2

Un po' di storia

- ☛ i linguaggi di programmazione nascono con la macchina di Von Neumann (macchina a programma memorizzato)
 - i programmi sono un particolare tipo di dato rappresentato nella memoria della macchina
 - la macchina possiede un interprete capace di fare eseguire il programma memorizzato, e quindi di implementare un qualunque algoritmo descrivibile nel “linguaggio macchina”
 - un qualunque linguaggio macchina dotato di semplici operazioni primitive per effettuare la scelta e per iterare (o simili) è Turing-equivalente, cioè può descrivere tutti gli algoritmi
- ☛ i linguaggi hanno tutti lo stesso potere espressivo, ma la caratteristica distintiva importante è il “quanto costa esprimere”
 - direttamente legato al “livello di astrazione” fornito dal linguaggio

3

I linguaggi macchina ad alto livello

- ☛ dai linguaggi macchina ai linguaggi Assembler
 - nomi simbolici per operazioni e dati
- ☛ (anni 50) FORTRAN e COBOL (sempreverdi)
 - notazioni ad alto livello orientate rispettivamente al calcolo scientifico (numerico) ed alla gestione dati (anche su memoria secondaria)
 - astrazione procedurale (sottoprogrammi, ma con caratteristiche molto simili ai costrutti forniti dai linguaggi macchina)
 - nuove operazioni e strutture dati (per esempio, gli arrays in FORTRAN, e i records in COBOL)
 - nulla di significativamente diverso dai linguaggi macchina

4

I favolosi anni '60: LISP e ALGOL'60

- ☛ risultati teorici a monte
 - formalizzazione degli aspetti sintattici
 - primi risultati semantici basati sul λ -calcolo
- ☛ caratteristiche comuni
 - introduzione dell'ambiente
 - vera astrazione procedurale con ricorsione
 - argomenti procedurali e per nome
- ☛ ALGOL'60
 - primo linguaggio imperativo veramente ad alto livello
 - scoping statico
 - gestione dinamica della memoria a stack
- ☛ LISP (sempreverde, ancora oggi il linguaggio dell'A.I.)
 - primo linguaggio funzionale, direttamente ispirato al λ -calcolo
 - scoping dinamico
 - strutture dati dinamiche, gestione dinamica della memoria a heap con garbage collector

5

Estendere la macchina fisica o implementare una logica

- ☛ ALGOL'60, prototipo dei linguaggi imperativi
 - parte dalla struttura della macchina fisica
 - la estende con nuovi potenti meccanismi
 - ☛ LISP, prototipo dei linguaggi logici e funzionali
 - parte da un calcolo logico (λ -calcolo)
 - ne definisce una implementazione sulla macchina fisica
 - ☛ ne nascono concetti simili
 - non a caso basati sulla teoria
 - ☛ gli approcci restano diversi e danno origine a due filoni
 - il filone imperativo
 - il filone logico
- che sono tuttora vitali

6

La fine degli anni '60

- ☛ PL/I: il primo tentativo di linguaggio “totalitario” (targato IBM)
 - tentativo di sintesi fra LISP, ALGOL'60 e COBOL
 - fallito per mancanza di una visione semantica unitaria
- ☛ SIMULA'67: nasce la classe
 - estensione di ALGOL'60 orientato alla simulazione discreta
 - quasi sconosciuto, riscoperto 15 anni dopo

7

Evoluzione del filone imperativo

- ☛ risultati anni '70
 - metodologie di programmazione, tipi di dati astratti, modularità, classi e oggetti
 - programmazione di sistema in linguaggi ad alto livello: eccezioni e concorrenza
- ☛ PASCAL
 - estensione di ALGOL'60 con la definizione di tipi (non astratti), l'uso esplicito di puntatori e la gestione dinamica della memoria a heap (senza garbage collector)
 - semplice implementazione mista (vedi dopo) facilmente portabile

8

Il dopo PASCAL

☛ C

- PASCAL + moduli + tipi astratti + eccezioni + semplice interfaccia per interagire con il sistema operativo

☛ ADA: il secondo tentativo di linguaggio “totalitario” (targato Dipartimento della Difesa U.S.A.)

- come sopra + concorrenza + costrutti per la programmazione in tempo reale
- progetto ambizioso, anche dal punto di vista semantico, con una grande enfasi sulla semantica statica (proprietà verificabili dal compilatore)

☛ C++

- C + classi e oggetti (allocati sulla heap, ancora senza garbage collector)

9

L'evoluzione del filone logico: programmazione logica

☛ PROLOG

- implementazione di un frammento del calcolo dei predicati del primo ordine
- strutture dati molto flessibili (termini) con calcolo effettuato dall'algoritmo di unificazione
- computazioni non-deterministiche
- gestione della memoria a heap con garbage collector

☛ CLP (Constraint Logic Programming)

- PROLOG + calcolo su domini diversi (anche numerici) con opportuni algoritmi di soluzione di vincoli

10

L'evoluzione del filone logico: programmazione funzionale

☞ ML

- implementazione del λ -calcolo tipato
- definizione di nuovi tipi ricorsivi
- i valori dei nuovi tipi sono termini, che possono essere visitati con un meccanismo di pattern matching (versione semplificata dell'unificazione)
- scoping statico (a differenza di LISP)
- semantica statica molto potente (inferenza e controllo dei tipi)
 - un programma “corretto” per la semantica statica quasi sempre va bene
- gestione della memoria a heap con garbage collector

☞ HASKELL

- ML con regola di valutazione “lazy”

11

JAVA

☞ molte caratteristiche dal filone imperativo

- essenzialmente tutte quelle del linguaggio più avanzato del filone, cioè C++

☞ alcune caratteristiche dei linguaggi del filone logico

- gestione della memoria con garbage collector

☞ utilizza il meccanismo delle classi e dell'ereditarietà per ridurre il numero di meccanismi primitivi

- quasi tutto viene realizzato con classi predefinite nelle librerie

☞ ha una implementazione mista (anch'essa tipica del filone logico)

- che ne facilita la portabilità e lo rende particolarmente adatto ad essere integrato nelle applicazioni di rete

12

OCAML

- ☛ Ocaml (Objective CaML), una estensione, orientata ad oggetti (e con un frammento imperativo), di uno dei più importanti linguaggi funzionali (ML)
 - progettato ed implementato all'INRIA (Francia)
- ☛ l'implementazione (per tutte le piattaforme importanti) si può scaricare dal sito
<http://caml.inria.fr/>
- ☛ il manuale on line al sito
<http://caml.inria.fr/ocaml/htmlman/index.html>