

# Linguaggi di programmazione e paradigmi

## Lezione 11 di Fondamenti di informatica

Docente: Giuseppe Scollo

Università di Catania  
Facoltà di Scienze Matematiche, Fisiche e Naturali  
Corso di Laurea in Informatica, I livello, AA 2009-10

### Indice

1. Linguaggi di programmazione e paradigmi
2. linguaggi di prima generazione
3. linguaggi di seconda generazione
4. linguaggi di terza generazione
5. paradigmi di programmazione
6. paradigma imperativo/procedurale
7. programmazione ad oggetti
8. programmazione dichiarativa
9. temi per ulteriori approfondimenti

## linguaggi di prima generazione

una storia di progressiva astrazione dalla macchina

linguaggi *assembler* :

rappresentazione **mnemonica** di codici operativi e registri

**nomi** alfanumerici (identificatori, variabili) delle locazioni di memoria dei dati

esempio:  $\text{CostoTotale} = \text{Prezzo} + \text{CostoSpedizione}$

156C LOAD R5 Prezzo

166D LOAD R6 CostoSpedizione

5056 ADDI R0 R5 R6

306E STORE R0 CostoTotale

C000 HALT

corrispondenza 1-1 delle primitive *assembler*  $\Leftrightarrow$  linguaggio macchina

## linguaggi di seconda generazione

programma *assembler* :

traduttore da linguaggio *assembler* a linguaggio macchina

linguaggi *macro-assembler* :

estensione dell'*assembler* con definizioni di **macro-istruzioni**:

nomi (con eventuali parametri) per sequenze frequenti di istruzioni

**espansione** delle macro-istruzioni: sostituzione testuale

programma *macro-assembler* :

*macro pre-processor* ; *assembler*

limite di queste prime due generazioni di linguaggi:

**dipendenza** dall'architettura della macchina

$\Rightarrow$  scarsa portabilità dei programmi

$\Rightarrow$  algoritmi orientati alla macchina piuttosto che al problema

## linguaggi di terza generazione

linguaggi di programmazione di alto livello, obiettivi:

affrancare lo sviluppo di algoritmi dalla dipendenza dalla macchina  
 individuare primitive e costrutti orientati al problema, di uso generale  
 in un ampio dominio applicativo

ampia diffusione trovano i primi linguaggi di terza generazione (anni '50), dotati di costrutti sofisticati:

**FORTRAN: FORMula TRANslator**

applicazioni scientifiche

**COBOL: COMmon Business Oriented Language**

applicazioni commerciali e gestionali

di notevole rilievo storico, sebbene trovi minor diffusione, è il coevo apparire del linguaggio LISP:

per applicazioni di Intelligenza Artificiale

## paradigmi di programmazione

negli anni '50 e '60 la programmazione imperativa, ispirata dall'architettura Von Neumann, rimane il paradigma dominante

a partire dagli anni '70 emergono altri paradigmi di programmazione:

nuclei di concetti fondamentali che danno vita a diversi approcci e modi di pensare all'espressione di algoritmi

eccone uno sviluppo temporale approssimativo, con i linguaggi più significativi:

Linguaggi	1950	1960	1970	1980	1990	2000	Paradigma
macchina							imperativo / procedurale
	FORTRAN	COBOL ALGOL	BASIC PL/I	C Pascal	Ada Modula-2		
		LISP	Scheme			Caml Haskell	funzionale
			FP ML				
				Prolog			logico
			Shunla67	Smalltalk	C++ Eiffel	Java	ad oggetti
					VisualBasic	G#	

## paradigma imperativo/procedurale

le primitive dei linguaggi imperativi esprimono **comandi** all'esecutore  
**costrutti** caratteristici:

istruzione di **assegnamento**:  $v := E$   
composizione **sequenziale** di comandi  
**iterazione** di comandi

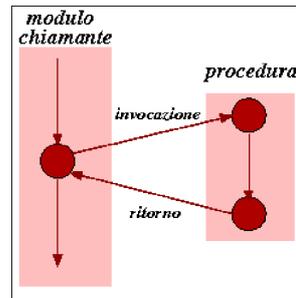
le istruzioni di "salto" vengono progressivamente abbandonate, secondo i canoni stilistici della **programmazione strutturata**

**astrazione procedurale** invece di macroistruzioni:

parametri **formali** nella **definizione** della procedura  
parametri **attuali** nella **invocazione** della procedura

**passaggio dei parametri**:

per **valore**  
per **riferimento**



## programmazione ad oggetti

estende la programmazione procedurale con i concetti di  
**classe**: **incapsulamento** di dati e operazioni per l'accesso ad essi  
**oggetto**: **istanza dinamica** di una classe

ha **stato** (variabili di istanza) e **identità**

nasce nell'ambito dei linguaggi di simulazione: **Simula 67**

ben si presta alla rappresentazione della **concorrenza**

si sviluppa negli anni '80 come **paradigma di progettazione del software**

linguaggi più diffusi: **Smalltalk, C++, Eiffel, Java**

anni '90: si afferma quale **paradigma dominante** nella **produzione industriale di software**, standardizzazione:

**OMG**: **Object Management Group**, <http://www.omg.org>

**CORBA**: **Common Object Request Broker Architecture**

**UML**: **Unified Modeling Language**

## programmazione dichiarativa

idea guida: programma = definizione **costruttiva** del problema

due principali paradigmi dichiarativi:

**programmazione funzionale:** basata sulla **composizione di funzioni**

precursori:  $\lambda$ -calcolo (A. Church, 1941), LISP (J. McCarthy, 1958)

nasce come paradigma con la 1977 *Turing Award Lecture* di J. Backus:  
***Can programming be liberated from the von Neumann style?***

*A functional style and its algebra of programs. CACM, 21(8):613--641 (1978)*

<http://www.stanford.edu/class/cs242/readings/backus.pdf> (~3 MB)

interpretazione di equazioni come **regole di riscrittura di termini**

**programmazione funzionale pura:** priva dell'assegnamento (imperativo)

ben si presta al **calcolo parallelo**

**programmazione logica:** basata sulla **deduzione predicativa**

"istruzioni" : **clausole Horn strettamente positive:**  $\gamma_1, \dots, \gamma_n \Rightarrow \alpha$

"scoperta da R.Kowalski nel 1974, implementata da A.Colmerauer nel 1973"

$\Rightarrow$  **Prolog** (verrà introdotto in una lezione successiva)

## temi per ulteriori approfondimenti

### 1. Programmazione Web

La diffusione di applicazioni e servizi in rete ha promosso lo sviluppo di linguaggi interpretati, o di **scripting**, quali ad es. **JavaScript**, per la programmazione di **siti Web dinamici**. Molte risorse sono reperibili al sito <http://www.javascript.it>

### 2. Programmazione dichiarativa

Ecco un elenco di siti su alcuni linguaggi di programmazione dichiarativa:

**Haskell:** <http://www.haskell.org>

**Scheme:** <http://swiss.csail.mit.edu/projects/scheme>

**SML:** <http://www.smlnj.org>

**Caml:** <http://caml.inria.fr>

**SWI-Prolog:** <http://www.swi-prolog.org>

**GNU Prolog:** <http://www.gprolog.org>

### 3. Ingegneria del software

Nello sviluppo di complessi sistemi software, programmare non è tutto. L'ingegneria del software colloca l'attività di programmazione nel più ampio contesto della pianificazione, progetto, realizzazione, collaudo e manutenzione di prodotti software. Il tema è introdotto nella sez. B.B del testo (Schneider & Gersting, 2007), un buon punto di partenza per approfondimenti.