EVOLUZIONE DEI LINGUAGGI DI ALTO LIVELLO

- Linguaggi di programmazione classificati in base alle loro caratteristiche fondamentali.
- **Linguaggio macchina**, binario e fortemente legato all'architettura.
- **Linguaggi Assembler**, istruzioni del linguaggio macchina e riferimenti alle celle di memoria espressi mediante simboli.
- **Linguaggi di alto livello**, tentativo di astrazione dell'architettura sottostante.
  - Operazioni di più alto livello rispetto a quelle caratteristiche della macchina.

UNA POSSIBILE CLASSIFICAZIONE

- Linguaggi di programmazione
  - *IMPERATIVI*
  - *FUNZIONALI*
  - *LOGICI*
  - *A OGGETTI*
- A loro volta possono essere *SEQUENZIALI* o *CONCORRENTI*.

PERCHE' TANTI LINGUAGGI ?

- Perché tante sono le differenti **applicazioni**
  - Efficienza
  - Rapida prototipazione
  - Facile comprensione del linguaggio (leggibilità)
  - Modificabilità e riusabilità
  - Programmazione in largo
  - Qualità del programma

LINGUAGGI IMPERATIVI

- Le caratteristiche essenziali sono dettate dalla architettura di Von Neumann (processore e memoria).
  - Tutti i linguaggi tradizionali (ASSEMBLER, FORTRAN, Pascal) sono livelli di astrazione costruiti sopra tale architettura.

- L'architettura consiste di due parti:
  - memoria (passiva)
  - processore (attivo)

- La principale attività del processore è assegnare valori a celle di memoria.

- Variabili come astrazione di celle di memoria e istruzione di assegnamento.

LINGUAGGI IMPERATIVI

- I linguaggi imperativi adottano uno **stile prescrittivo**.
  - Il programma imperativo prescrive le operazioni che il processore deve compiere (es. assegnamento)
  - Esecuzione delle istruzioni nell'ordine in cui appaiono nel programma (eccezione: strutture di controllo)
- Realizzati sia attraverso interpretazione (BASIC,...) che compilazione (Pascal, FORTRAN, COBOL,...)
- Sono forse la classe di linguaggi più matura
- Nati per manipolazione numerica più che simbolica

LINGUAGGI IMPERATIVI

- Sono evoluti verso
  - blocchi (ALGOL)
  - moduli (MODULA)
  - tipi di dato astratto (SIMULA)
  - oggetti (C++)
  - parallelismo (Concurrent Pascal, CSP, etc.)
- in generale verso costrutti adatti per programmazione in largo (programmazione ad oggetti)
- Sono generalmente linguaggi che adottano un controllo statico (stretto) di tipo
- Sono generalmente efficienti

LINGUAGGI IMPERATIVI

PROGRAMMA = ALGORITMO + DATI

- La struttura del programma consiste in
  - Una parte di dichiarazione in cui si dichiarano tutte le variabili del programma e il loro tipo;
  - Una parte istruzioni che descrive l'algoritmo risolutivo utilizzato, mediante istruzioni del linguaggio.

- Le istruzioni si dividono in:
  - Istruzioni di **lettura** e **scrittura**;
  - Istruzione di **assegnamento** (astrazione della cella di memoria);
  - Istruzioni di **controllo**.

SCELTA DI NUOVI PARADIGMI

- Motivazioni
  - Applicazioni di Intelligenza Artificiale
  - Programmazione esplorativa e prototipale
  - Tentativo di sviluppare programmi più concisi, più semplici da scrivere, più vicini alla logica del problema, più semplici da verificare formalmente

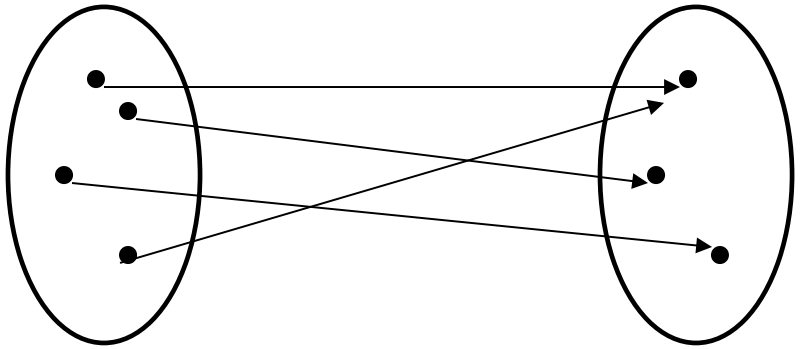
- Le specifiche di un linguaggio ad alto livello possono ispirare una nuova architettura (LISP machine, Progetto Giapponese di V Generazione):
  - ***PROGRAMMAZIONE FUNZIONALE***
  - ***PROGRAMMAZIONE LOGICA***

PROGRAMMAZIONE FUNZIONALE E LOGICA: CARATTERISTICHE COMUNI

- Non manipolazione di numeri, ma di simboli
- Il *programma* è costantemente in evoluzione:
  - modularità spinta
  - programma come struttura dati;
  - l'ambiente tende a confondersi col linguaggio.
- La programmazione è esplorativa ed incrementale: prototipi veloci
- Idea iniziale: il linguaggio deve essere ad "altissimo livello": utilizzabile anche da non-programmatori.

LINGUAGGI FUNZIONALI

- Concetto primitivo: quello di **funzione** e di **applicazione di funzione**.
  - Una **funzione** è una regola di corrispondenza che associa a ogni elemento del suo dominio un **unico** elemento nel codominio.
  - Una definizione di funzione specifica il **dominio**, il **codominio** e la **regola di associazione**. Ad esempio: incr(x)::=x + 1.
  - Dopo la definizione una funzione può essere **applicata** a un elemento del dominio per restituire l'elemento associato del codominio (**valutazione**): incr(3) → 4.



LINGUAGGI FUNZIONALI

- La sola operazione del modello funzionale è l'*applicazione di funzioni a operandi*;

- Il ruolo della *macchina funzionale* (**interprete**) è *valutare* il programma e produrre un valore;
  - Il valore di una funzione è determinato solo dai suoi argomenti (assenza di effetti collaterali, funzionale puro);

- L'essenza della programmazione funzionale è *combinare* funzioni (utilizzo della ricorsione);

- Le *variabili* sono variabili *matematiche* che denotano un valore non mutabile (nessun assegnamento)

LINGUAGGI FUNZIONALI

PROGRAMMA = FUNZIONE

- Esecuzione del programma = valutazione della funzione
- Lavorano su strutture dati come alberi binari, liste
- Anche i programmi sono strutture dati

LINGUAGGIO LISP (LISt Processing)

- Nato nel 1958 ad opera di J. McCarthy
- Originariamente era puramente funzionale
- Il LISP ha un ricchissimo ambiente di programmazione:
  - InterLisp
  - MacLisp
  - CommonLisp
  - ...

LINGUAGGIO LISP (LISt Processing)

```
(defun member (item list)
  (cond      ((null list) nil)
  ((equal item (car list)) T)
  (T (member item (cdr list)))))

(member 'A '(B C D))
```

- Il programma è una struttura dati;
- Non c'è dichiarazione di tipo;
- Non c'è assegnamento.

LINGUAGGIO LISP (LISt Processing)

- Quando usare il LISP e la programmazione funzionale
  - Manipolazione simbolica e non numerica
  - Programmazione esplorativa
  - Modifica dinamica dei programmi
  - Problemi naturalmente ricorsivi
  - Non problemi di controllo dei processi o real-time

- È un linguaggio da cui si può imparare molto sulla costruzione di algoritmi e sulla realizzazione dei linguaggi (legami di variabili; scope rules).

- Un interprete di LISP puro può essere scritto in LISP molto semplicemente (EVAL).

LINGUAGGI LOGICI

PROGRAMMA = CONOSCENZA + CONTROLLO

- *Stile dichiarativo*: la conoscenza sul problema è espressa indipendentemente dal suo utilizzo (COSA non COME)
  - che significa *dichiarativo*?
- Alta modularità ed espressività
- Definire un linguaggio logico significa definire come il programmatore può esprimere la conoscenza e quale tipo di controllo può utilizzare;
  - Problematiche di *rappresentazione* della conoscenza

PROGRAMMAZIONE DICHIARATIVA E PROCEDURALE

- Problema: ordinare una lista
- *procedurale*
  - Controlla prima se la lista è vuota; se sì dai come risultato la lista vuota
  - Altrimenti calcola una permutazione L' di L e controlla se e ordinata; se sì termina dando come risultato L', altrimenti calcola un'altra permutazione di L etc....
- *dichiarativo*
  - Il risultato dell'ordinamento di una lista vuota è la lista vuota
  - Il risultato dell'ordinamento di una lista L è L' se la lista L' è ordinata ed è la permutazione di L

LINGUAGGI A VINCOLI

- Nascono come estensioni della programmazione logica per superarne alcuni limiti
  - inefficienza nell'esplorazione dello spazio delle soluzioni
  - unificazione sintattica
  - rigidità nel controllo
- Nel 1987 nasce la Programmazione Logica a Vincoli come schema generale che può essere specializzato su diversi domini (controllo ~ semantica)
- Oggi i principali linguaggi a vincoli non sono legati al paradigma logico

LINGUAGGI A VINCOLI

- Esistono vari linguaggi a vincoli specializzati sul dominio dei numeri reali, razionali, interi, booleani, intervalli ecc.

- Vedremo in dettaglio i Linguaggi a vincoli su domini finiti di interi particolarmente utili per risolvere particolari classi di problemi detti Problemi di Soddisfacimento di Vincoli e Problemi di Ottimizzazione combinatoria.

PROGRAMMA = CONOSCENZA + CONTROLLO

- Conoscenza: variabili domini e vincoli
- Controllo: propagazione di vincoli e ricerca