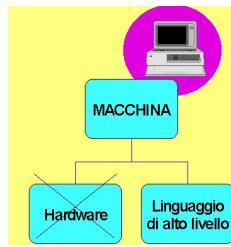


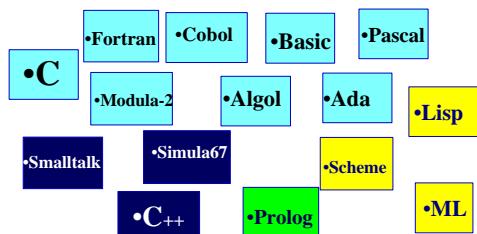
LINGUAGGI DI ALTO LIVELLO

Si basano su una *macchina virtuale* le cui "mosse" non sono quelle della macchina hardware



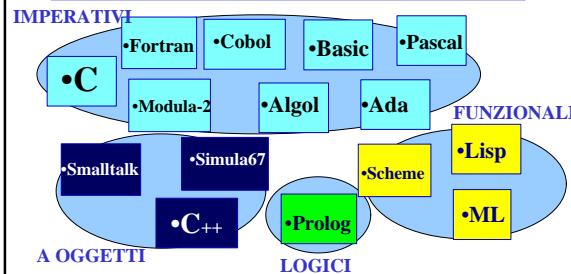
Linguaggi di alto livello

•Barriera di astrazione

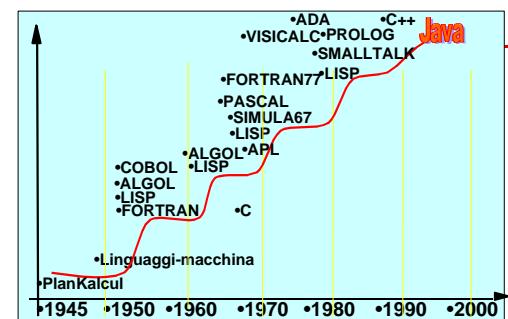


Linguaggi di alto livello

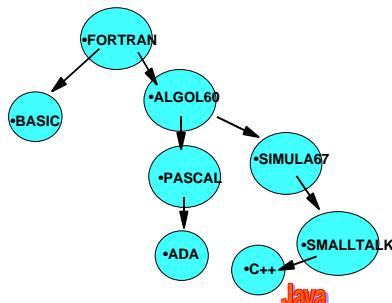
•Barriera di astrazione



Evoluzione dei linguaggi



Evoluzione dei linguaggi



COS'È UN LINGUAGGIO?

"Un linguaggio è un *insieme di parole* e di *metodi di combinazione delle parole* usate e comprese da una comunità di persone."

• È una definizione poco precisa:

- non evita le ambiguità dei linguaggi naturali
- non si presta a descrivere processi computazionali meccanizzabili
- non aiuta a stabilire proprietà

LA NOZIONE DI LINGUAGGIO

- Occorre una **nozione di linguaggio più precisa**
- **Linguaggio come sistema matematico**
che consente di rispondere a domande come:
 - quali sono le **frasi lecite**?
 - si può stabilire se una frase **appartiene al linguaggio**?
 - come si stabilisce il **significato** di una frase?
 - quali **elementi linguistici primitivi** ?

LINGUAGGIO & PROGRAMMA

- Dato un algoritmo, **un programma** è la sua **descrizione in un particolare linguaggio** di programmazione
- **Un linguaggio di programmazione** è una **notazione formale** che può essere usata per descrivere algoritmi.
Due aspetti del linguaggio:
 - **SINTASSI**
 - **SEMANTICA**

SINTASSI & SEMANTICA

- **Sintassi:** l'insieme di regole formali per la scrittura di programmi in un linguaggio, che dettano le **modalità per costruire frasi corrette** nel linguaggio stesso.
- **Semantica:** l'insieme dei **significati** da attribuire alle frasi (sintatticamente corrette) costruite nel linguaggio.

NB: una frase può essere **sintatticamente corretta** e tuttavia **non avere significato!**

SINTASSI

Le regole sintattiche sono espresse attraverso **notazioni formali**:

- ◆ **BNF (Backus-Naur Form)**
- ◆ **EBNF (Extended BNF)**
- ◆ **diagrammi sintattici**

SINTASSI EBNF: ESEMPIO

Sintassi di un numero naturale

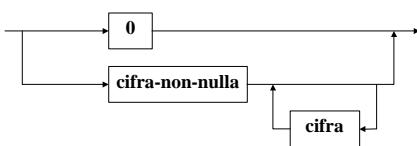
```
<naturale> ::=  
  0 | <cifra-non-nulla>{<cifra>}  
<cifra-non-nulla> ::=  
  1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9  
<cifra> ::=  
  0 | <cifra-non-nulla>
```

SINTASSI DI UN NUMERO NATURALE

```
<naturale> ::=  
  0 | <cifra-non-nulla>{<cifra>}  
Intuitivamente significa che un numero naturale si può riscrivere come 0 oppure ( ) come una cifra non nulla seguita da una o più ( ) cifre.  
<cifra-non-nulla> ::=  
  1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9  
una cifra non nulla si può riscrivere come 1 oppure 2 oppure 3...  
<cifra> ::= 0 | <cifra-non-nulla>  
una cifra si può riscrivere come 0 oppure come una cifra non nulla (definita precedentemente)
```

DIAGRAMMI SINTATTICI: ESEMPIO

Sintassi di un *numero naturale*



SEMANTICA

La semantica è esprimibile:

- ◆ **a parole** (poco precisa e ambigua)
- ◆ mediante **azioni**
→ **semantica operazionale**
- ◆ mediante **funzioni matematiche**
→ **semantica denotazionale**
- ◆ mediante **formule logiche**
→ **semantica assiomatica**

REALIZZAZIONE

- Per la realizzazione di un linguaggio di programmazione è necessaria la presenza di una macchina (fisica o astratta) che sia in grado di eseguire i programmi del linguaggio.
- Realizzazione di un "traduttore" che renda i programmi eseguibili su un dato elaboratore (**compilatore** o **interprete**).

ASTRAZIONE

- Esistono linguaggi a vari livelli di astrazione

- **Linguaggio Macchina:**
 - implica la conoscenza dei metodi di rappresentazione delle informazioni utilizzati.
- **Linguaggio Macchina e Assembler:**
 - implica la conoscenza dettagliata delle caratteristiche della macchina (registri, dimensioni dati, set di istruzioni)
 - semplici algoritmi implicano la specifica di molte istruzioni
- **Linguaggi di Alto Livello:**
 - Il programmatore può astrarre dai dettagli legati all'architettura ed esprimere i propri algoritmi in modo simbolico.

Sono indipendenti dalla macchina hardware sottostante
ASTRAZIONE

ASTRAZIONE

Linguaggio Macchina:

```

0100 0000 0000 1000
0100 0000 0000 1001
0000 0000 0000 1000
  
```

Difficile leggere e capire un programma scritto in forma binaria

Linguaggio Assembler:

```

... LOADA H
LOADB Z
ADD
...
  
```

Le istruzioni corrispondono univocamente a quelle macchina, ma vengono espresse tramite nomi simbolici (parole chiave).

Linguaggi di Alto Livello:

```

main()
{
    int A;
    scanf("%d", &A);
    if (A==0) {...}
    ...
  
```

Sono indipendenti dalla macchina

ESECUZIONE

- Per eseguire sulla macchina hardware un programma scritto in un linguaggio di alto livello è necessario tradurre il programma in sequenze di istruzioni di basso livello, direttamente eseguite dal processore, attraverso:

- **interpretazione** (ad es. BASIC)
- **compilazione** (ad es. C, FORTRAN, Pascal)

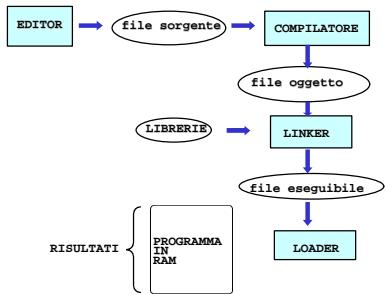
COME SVILUPPARE UN PROGRAMMA

- Qualunque sia il linguaggio di programmazione scelto occorre:
 - Scrivere il **testo del programma** e memorizzarlo su supporti di memoria permanenti (**fase di editing**);
- Se il linguaggio è compilato:
 - Compilare il programma, ossia utilizzare il compilatore che effettua una traduzione automatica del programma scritto in un linguaggio qualunque in un programma equivalente scritto in **linguaggio macchina**;
 - Eseguire il programma tradotto.
- Se il linguaggio è interpretato:
 - Usare l'interprete per eseguire il programma.

SI RICORDA CHE...

- I **compilatori** traducono automaticamente un programma dal linguaggio L a quello macchina (per un determinato elaboratore).
- Gli **interpreti** sono programmi capaci di eseguire direttamente un programma in linguaggio L istruzione per istruzione.
- I programmi compilati sono in generale **più efficienti** di quelli interpretati.

APPROCCIO COMPILATO: SCHEMA



APPROCCIO INTERPRETATO: SCHEMA

