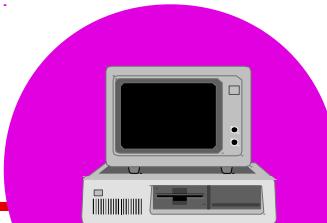
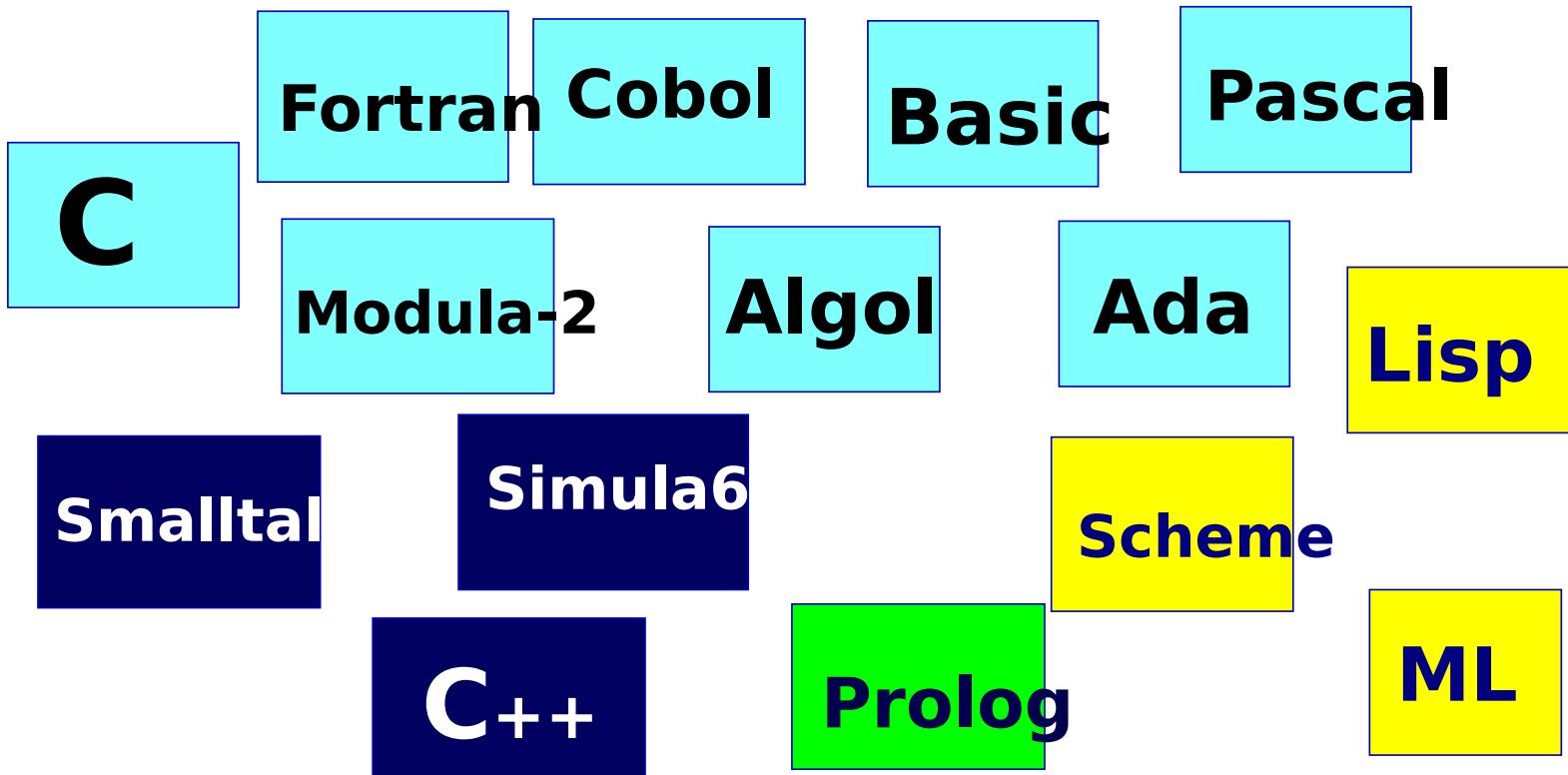


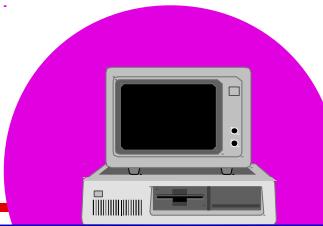
Linguaggi di alto livello



• Barriera di astrazione

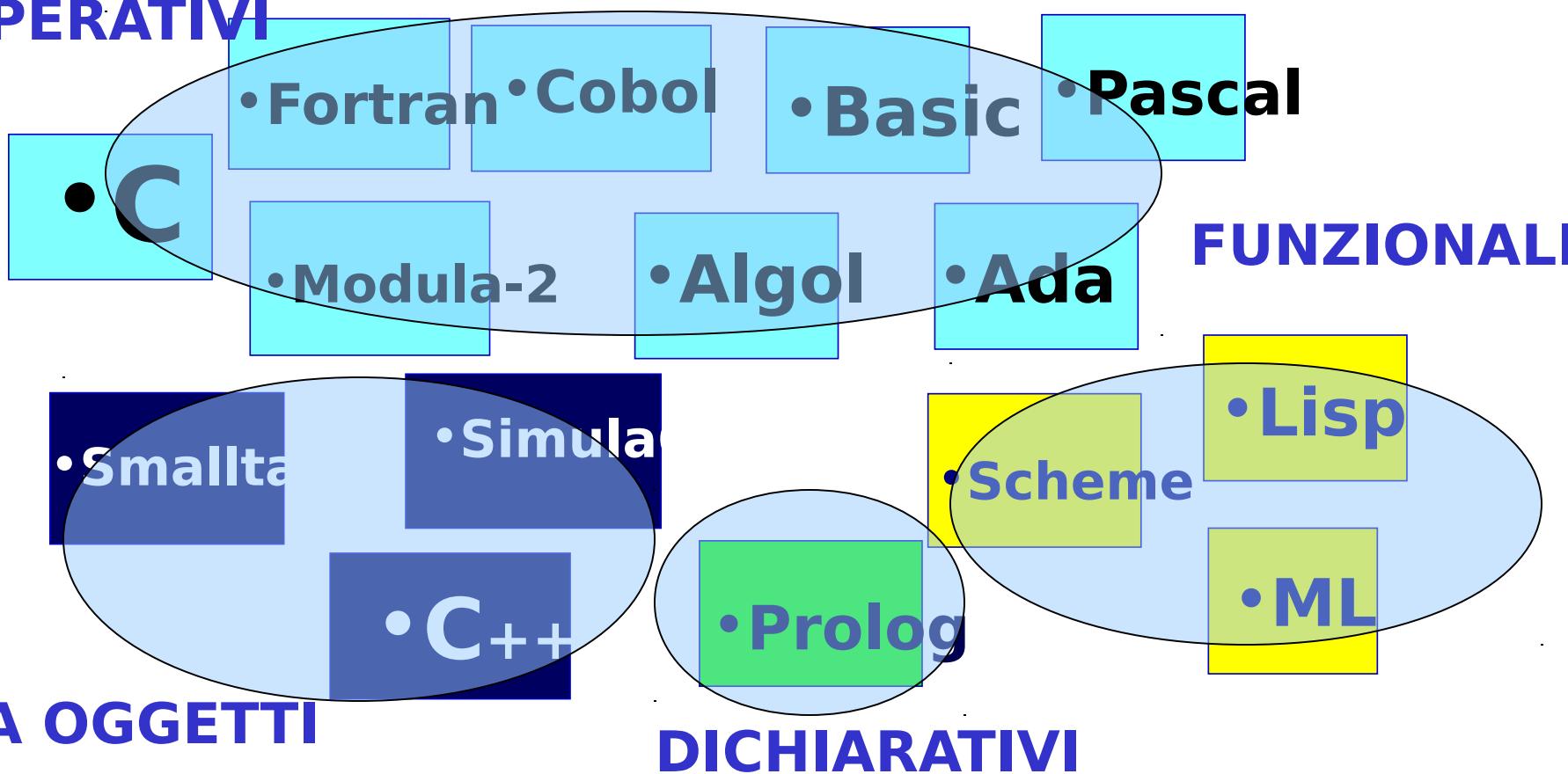


Linguaggi di alto livello



• Barriera di astrazione

IMPERATIVI



A OGGETTI

DICHIARATИVI

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

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 consenta di rispondere a domande come:
 - quali sono le *frasi lecrite*?
 - 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 zero o più ({ }) cifre.

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...

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

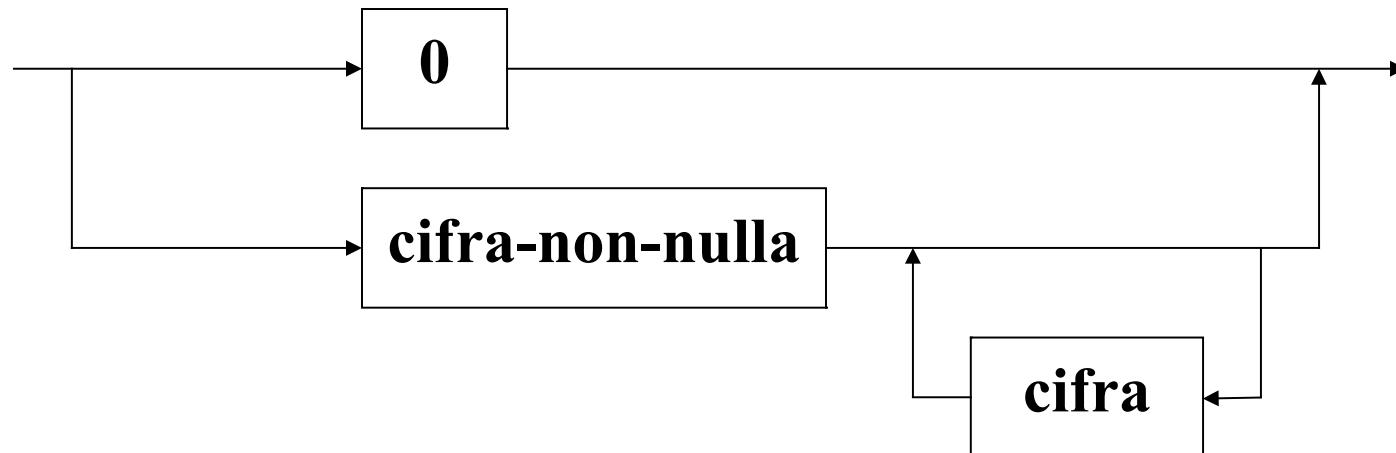
una cifra non nulla si può riscrivere come 1 oppure 2 oppure 3...

<cifra> ::= 0 | <cifra-non-nulla>

una cifra si può riscrivere come 0 oppure come una cifra non nulla (definita prece

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**

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.

AMBIENTI DI PROGRAMMAZIONE

È l'insieme dei programmi che consentono la scrittura, la verifica e l'esecuzione di nuovi programmi (*fasi di sviluppo*).

Sviluppo di un programma

- Affinché un programma scritto in un qualsiasi linguaggio di programmazione sia comprensibile (e quindi eseguibile) da un calcolatore, *occorre tradurlo* dal linguaggio originario al linguaggio della macchina.
- Questa operazione viene normalmente svolta da speciali programmi, detti *traduttori*.

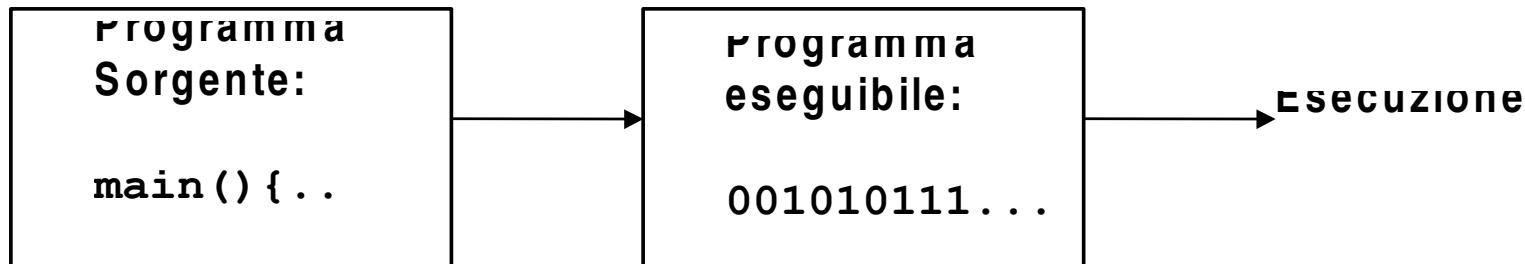
TRADUZIONE DI UN PROGRAMMA

PROGRAMMA	TRADUZIONE
<code>main()</code>	
{ int A;	00100101
...	
<code>A=A+1;</code>	11001..
<code>if....</code>	1011100..

Il traduttore converte

- *il testo* di un programma scritto in un particolare linguaggio di programmazione (*sorgenti*)
- nella corrispondente *rappresentazione in linguaggio macchina* (programma **esegibile**).

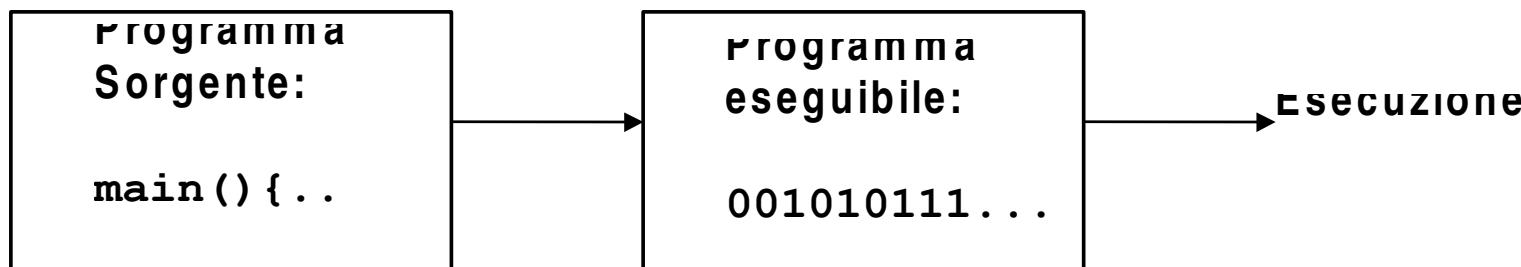
SVILUPPO DI PROGRAMMI



Due categorie di traduttori:

- i **Compilatori** traducono l'intero programma (senza eseguirlo!) e producono in uscita il programma convertito in linguaggio macchina
- gli **Interpreti** traducono ed eseguono immediatamente ogni singola istruzione del *programma sorgente*.

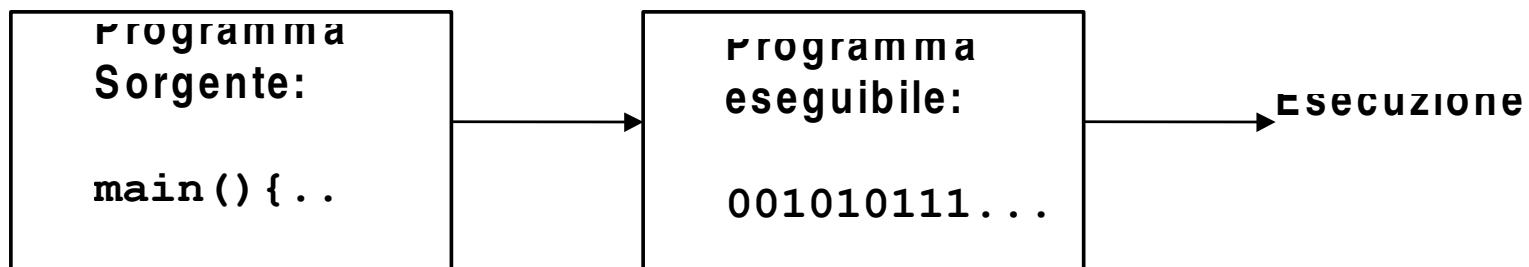
SVILUPPO DI PROGRAMMI (segue)



Quindi:

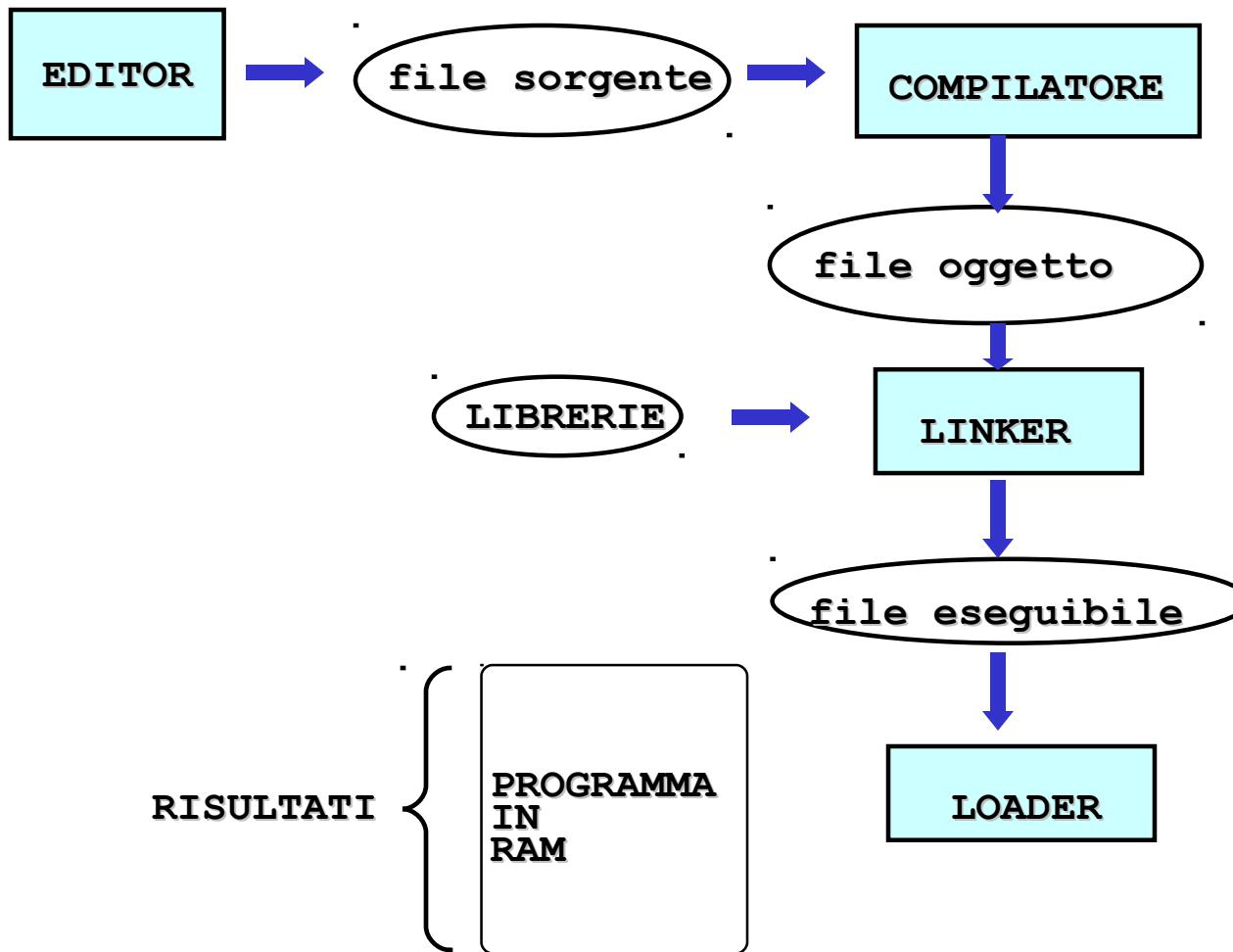
- nel caso del **compilatore**, lo schema precedente viene percorso *una volta sola* prima dell'esecuzione
- nel caso dell'**interprete**, lo schema viene invece attraversato *tante volte quante sono le istruzioni* che compongono il programma.

SVILUPPO DI PROGRAMMI (segue)



L'esecuzione di un programma *compilato* è più veloce
dell'esecuzione di un programma *interpretato*

APPROCCIO COMPILATORE: SCHEMA



APPROCCIO INTERPRETATO: SCHEMA

