- 1^ parte
  - Gli Algoritmi
  - _____
  - _____

http://www.elet.polimi.it/upload/martucci/index.html

INTRODUZIONE

INFORMATICA

Scienza che studia la **rappresentazione** e **l'elaborazione** delle informazioni mediante **macchine**, dette calcolatori elettronici.

In inglese: ***Computer Science***

Macchine a **funzionalità intrinseca**, o «cablata»:

sono macchine (elettroniche, ma anche meccaniche, elettriche, ecc.) il cui comportamento è determinato dalla loro **struttura** costruttiva.

Approccio **efficiente** ma rigido.

Macchine a **funzionalità programmata**:

sono macchine il cui comportamento è determinato **esecuzione di programmi**, cioè descrizioni di soluzioni di problemi.

Approccio meno efficiente ma **molto flessibile**.

La massima flessibilità di impiego delle macchine programmabili si ha quando in esse è possibile inserire, a scelta, diversi programmi.
Questa flessibilità è la causa principale della notevole diffusione dei calcolatori e dello sviluppo dell'informatica.
In questo corso ci interessiamo quindi di macchine a funzionalità programmata.

Problemi da risolvere

I problemi che siamo interessati a risolvere sono di natura *molto varia:*

1) Trovare il maggiore fra due numeri
2) Dato un elenco di nomi e numeri di telefono, trovare il numero di una data persona
3) Problema del lupo, della capra e del cavolo
4) Dati *a* e *b*, risolvere l'equazione $ax+b=0$
5) Stabilire se una parola precede alfabeticamente un'altra
6) Ordinare una lista di elementi

7) Identificare e prenotare aerei, treni, hotel, ...
8) Creare, modificare e alterare suoni, canzoni, ...
9) Analizzare e riconoscere immagini, ...
10) Salvare e recuperare delle informazioni, …
11) Trasmettere delle informazioni, …

E’ fondamentale capire la differenza tra…

- **Specifica di un problema**
- **Specifica del processo di risoluzione**
- **Codifica del processo di risoluzione**

Risoluzione di un problema

Con questo termine si indica il processo che:

- dato un *problema*
- individuato un opportuno *metodo risolutivo*

trasforma i dati iniziali nei corrispondenti risultati finali

E’ indispensabile capire a quale “macchina” ci si riferisce”

COMPUTER

Computer

- E’ uno strumento in grado di eseguire insiemi di *azioni* (“mosse”) elementari

- Le azioni vengono eseguite su oggetti (***dati***) per produrre altri oggetti (***risultati***)

- L’esecuzione di azioni viene richiesta all’elaboratore attraverso “**frasi scritte in un qualche linguaggio”** (***istruzioni***)

Componenti di un Sistema di Elaborazione

- **Hardware**
- **Software**
  - Software di sistema: Sistema operativo
  - Software applicativo

Domande fondamentali

- **Come viene rappresentata TUTTA l'informazione all'interno del computer?**
- **Quali istruzioni esegue un computer?**
- **Quali problemi può risolvere un computer?**
- **Esistono problemi che un computer non può risolvere?**

Algoritmo

ALGORITMO

- Sequenza finita di mosse che risolve in un tempo finito una classe di problemi

CODIFICA o IMPLEMENTAZIONE

- Fase di scrittura di un algoritmo attraverso un insieme ordinato di **frasi** (**“istruzioni”**), scritte in un **linguaggio di programmazione**, che specificano le **azioni** da compiere in modo formale interpretabile dal computer

Programma

- **Testo** scritto secondo la ***sintassi*** (alfabeto+regole grammaticali) e la ***semantica*** di un linguaggio di programmazione

- **Un programma può <u>non essere</u> un algoritmo!**

PROGRAMMAZIONE

È l'attività con cui si predispone l'elaboratore ad eseguire un *particolare insieme di azioni* su *particolari dati*, allo scopo di *risolvere un problema*.

ALGORITMO e PROGRAMMA (a confronto)

- Ogni elaboratore è una macchina in grado di eseguire *azioni* elementari su *dati*
- L'esecuzione delle azioni elementari è richiesta all'elaboratore tramite comandi chiamati *istruzioni*
- Le istruzioni sono espresse attraverso *frasi* di un opportuno *linguaggio di programmazione*
- **Un *programma* non è altro che la formulazione testuale di un *algoritmo* in un linguaggio di programmazione**

Elementi degli Algoritmi

In generale negli algoritmi si possono evidenziare i seguenti aspetti:

- **oggetti -> i dati**
- **operazioni -> le operazioni che modificano i dati**
- **flusso di controllo -> in che ordine si opera**

Oggetti. Costituiscono le entità su cui opera l'algoritmo.

In genere si tratta di dati iniziali del problema, informazioni ausiliarie e risultati parziali e finali. Si noti che in informatica le informazioni sono generalmente dette ***dati***, anche se costituiscono risultati parziali o finali di elaborazioni. Possono essere **variabili** o **costanti** di vari tipi.

Operazioni. Sono gli interventi da effettuare sugli oggetti, cioè sui dati.

In genere si tratta di calcoli, confronti, ricopiature, acquisizioni emissioni, ecc.

Flusso di controllo. Costituisce l'indicazione delle possibili evoluzioni dell'esecuzione delle operazioni, cioè le possibili successioni dei passi

E' importante notare che la correttezza dei risultati dipende non solo dalla corretta esecuzione delle singole operazioni, ma anche dalla corretta sequenza con cui sono eseguite.

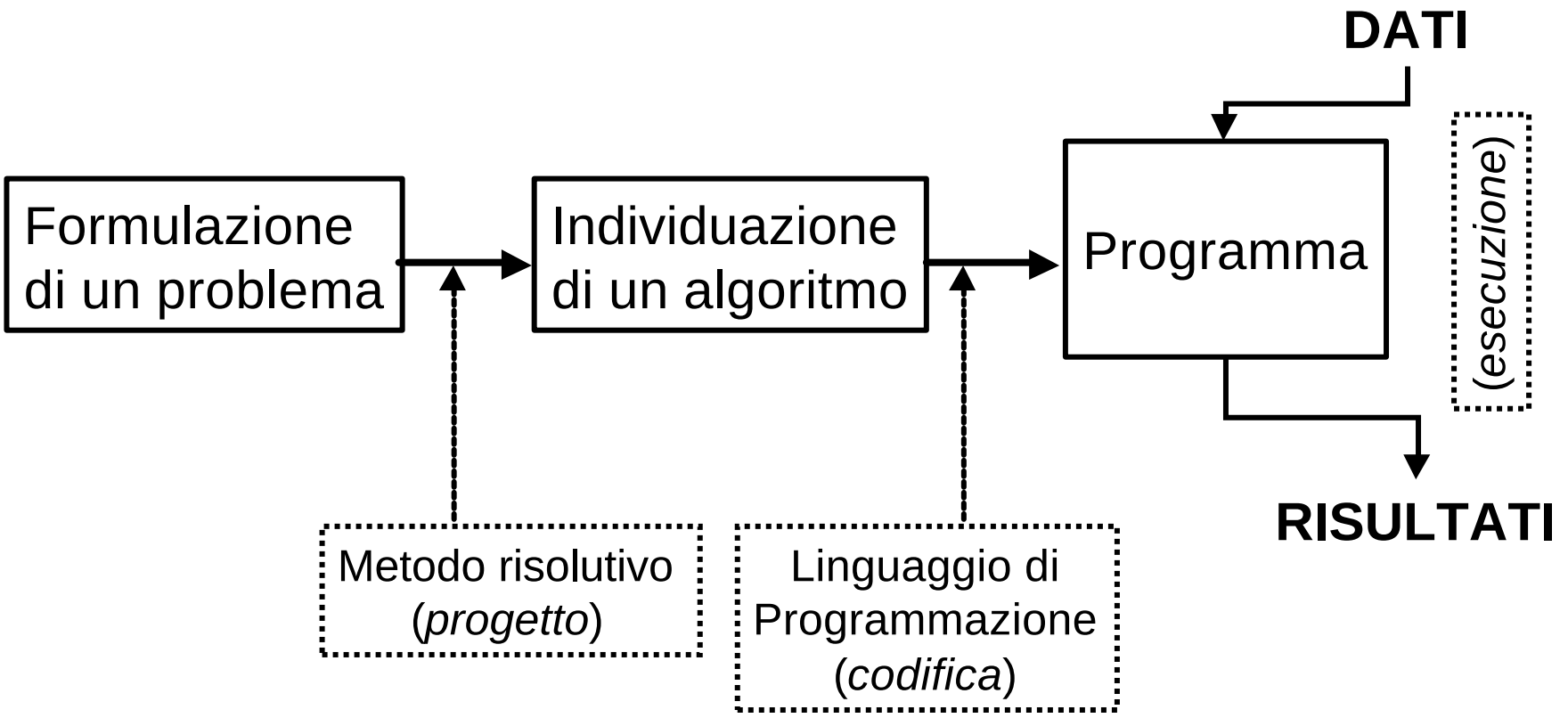
NOTA: è importante chiarire la differenza tra

Flusso di controllo e flusso di esecuzione.

- Il **flusso di controllo** è la descrizione a priori di tutte le possibili sequenze nell'esecuzione dei passi dell'algoritmo, in particolare di operazioni in alternativa e di operazioni da ripetere più volte ciclicamente.

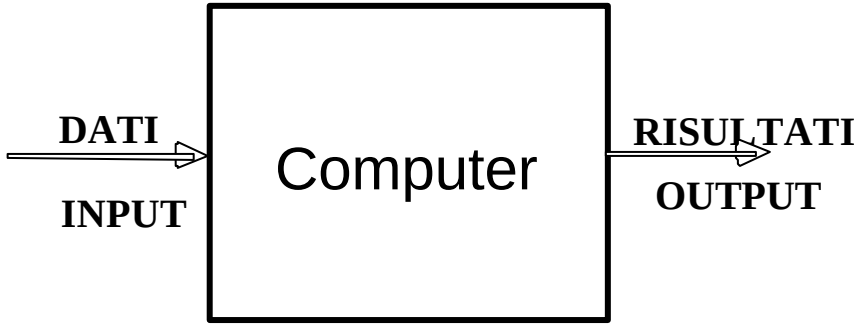
- Il **flusso di esecuzione** è la sequenza di operazioni effettivamente seguita durante una particolare esecuzione dell'algoritmo e che dipende dai particolari valori che i dati assumono in quella esecuzione.

Riassumendo…



Esecuzione di un programma

- L'esecuzione delle azioni *nell'ordine specificato dall'algoritmo* consente di ottenere, a partire dai dati di ingresso, i risultati che risolvono il problema



Esempio 1

Descrizione del problema

Stampare la somma di due numeri dati

Individuazione di un algoritmo

- Leggere il primo numero (p.es., da tastiera)
- Leggere il secondo numero (p.es., da tastiera)
- Effettuare la somma
- Stampare il risultato (p.es., su video)

Esempio 1 (*cont.*)

Codifica in un programma (***in linguaggio C***)

```
main()
 {
  int A, B, ris;
   printf("Immettere due numeri: ");
   scanf("%d", &A);
   scanf("%d", &B);
   ris=A+B;
   printf("Somma: %d\n", ris);
 }
```

Caratteristiche di un algoritmo

- **Eseguibilità**: ogni azione deve essere *eseguibile* da parte dell'esecutore dell'algoritmo in un tempo finito

- **Non-ambiguità**: ogni azione deve essere *univocamente interpretabile* dall'esecutore

- **Terminazione**: il numero totale di azioni da eseguire, per ogni insieme di dati di ingresso, deve essere finito

Algoritmo (*cont.*)

Quindi, l'algoritmo deve:

- essere *applicabile a qualsiasi insieme di dati di ingresso* appartenenti al *dominio di definizione* dell'algoritmo
- essere costituito da operazioni appartenenti ad un determinato insieme di operazioni fondamentali
- essere costituito da regole non ambigue, cioè interpretabili in modo univoco qualunque sia l'esecutore (persona o "macchina") che le legge

ALTRE PROPRIETA'

- **Determinismo**
- **Efficienza**
- **Terminazione**

Algoritmi equivalenti

Due algoritmi si dicono **equivalenti** quando:

- hanno lo stesso dominio di ingresso
- hanno lo stesso dominio di uscita
- in corrispondenza degli stessi valori nel dominio di ingresso *producono gli stessi valori* nel dominio di uscita

Due algoritmi equivalenti:

- forniscono lo **stesso risultato**
- ma possono avere **diversa efficienza**
- e possono essere **profondamente diversi !**

Esempio 2

Calcolo del Massimo Comun Divisore (MCD) fra due interi M ed N

Algoritmo n° 1

- Calcola l'insieme A dei divisori di M
- Calcola l'insieme B dei divisori di N
- Calcola l'insieme C dei divisori comuni = A ∩ B
- Il risultato è il massimo dell'insieme C

Algoritmo n° 2 (metodo di Euclide)

$$\text{MCD}(M,N) = \begin{cases} M \text{ (oppure N)} & \text{se } M=N \\ \text{MCD}(M-N,\ N) & \text{se } M>N \\ \text{MCD}(M,\ N-M) & \text{se } M<N \end{cases}$$

Algoritmo:

- Finché $M \neq N$:
- se $M>N$, sostituisci a M il valore $M' = M-N$
- altrimenti sostituisci a N il valore $N' = N-M$
- Il Massimo Comun Divisore è il valore finale ottenuto quando M e N diventano uguali

Esempio 2 (*cont*.)

Calcolare il MCD dei numeri 12 e 21

Algoritmo 1: Divisori di 12 = 1, 2, 3, 4, 6, 12
Divisori di 21 = 1, 3, 7, 21
Intersezione insieme dei divisori: 1, 3
Massimo dell'insieme dei divisori: **MCD = 3**

Algoritmo 2: M ← 12, N ← 21
N>M, quindi: N ← (N-M) = 9 ➔ M=12, N=9
M>N, quindi: M ← (M-N) = 3 ➔ M=3, N=9
N>M, quindi: N ← (N-M) = 6 ➔ M=3, N=6
N>M, quindi: N ← (N-M) = 3 ➔ M=3, N=3
N=M, quindi: **MCD = 3**

Rappresentazione di un algoritmo destinato all'esecuzione automatica

La rappresentazione di un algoritmo è la descrizione, univoca per l'esecutore, di tutte le possibili sequenze di operazioni da eseguire per risolvere il problema dato.
E' quindi la ***descrizione*** del ***flusso di controllo,*** delle ***operazioni eseguibili***, e degli ***oggetti*** su cui agiscono le singole operazioni

E' necessario un supporto ***formale*** alla descrizione di un algoritmo, cioè un formalismo (linguaggio) costituito da:

1. un insieme di **elementi** per la descrizione di oggetti, operazioni e flusso di controllo: ***vocabolario***

2. un insieme di **regole** per la **composizione** degli elementi in frasi eseguibili e costrutti di controllo (istruzioni): ***sintassi***

Il rispetto delle *regole sintattiche* deve poter essere verificato in modo automatico.

3. un insieme di **regole** per **l'interpretazione** degli elementi e delle istruzioni sintatticamente corrette: ***semantica***

Linguaggi di descrizione di algoritmi

La rappresentazione di un algoritmo destinato all'esecuzione automatica deve essere basata su
formalismi descrittivi costituiti dai
LINGUAGGI (artificiali) DI PROGRAMMAZIONE

I linguaggi di descrizione degli algoritmi rappresentano gli **oggetti** tramite dei **nomi simbolici** (identificatori).
In particolare si introduce il concetto di **variabile**: una variabile costituisce la **rappresentazione simbolica di dati** cui si possono attribuire diversi valori di un certo tipo.

I linguaggi rappresentano le **operazioni** mediante **operatori** (es. +) o **funzioni** (es. cos(x)).

I linguaggi rappresentano la sequenza di operazioni (**flusso di controllo**) con appositi **costrutti** di controllo.

Linguaggi di descrizione
Corrispondenza tra concetti e loro rappresentazione

Concetto	*Rappresentazione*
oggetto	identificatore
operazione	operatore, funzione
direttiva	istruzione
flusso di controllo	costrutti di controllo
algoritmo	programma

PER LA CODIFICA DI UN ALGORITMO…

- Bisogna conoscere la **sintassi** di un linguaggio di programmazione
- Bisogna conoscere la **semantica** di un linguaggio di programmazione
- Bisogna conoscere un **ambiente di programmazione**

Quale **linguaggio**? Quale ambiente di programmazione?

Linguaggi di programmazione

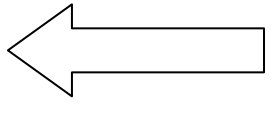
- **Un linguaggio di programmazione è una notazione formale per descrivere algoritmi**

- **Un programma è la "codifica" o "implementazione" di un algoritmo in un linguaggio di programmazione**

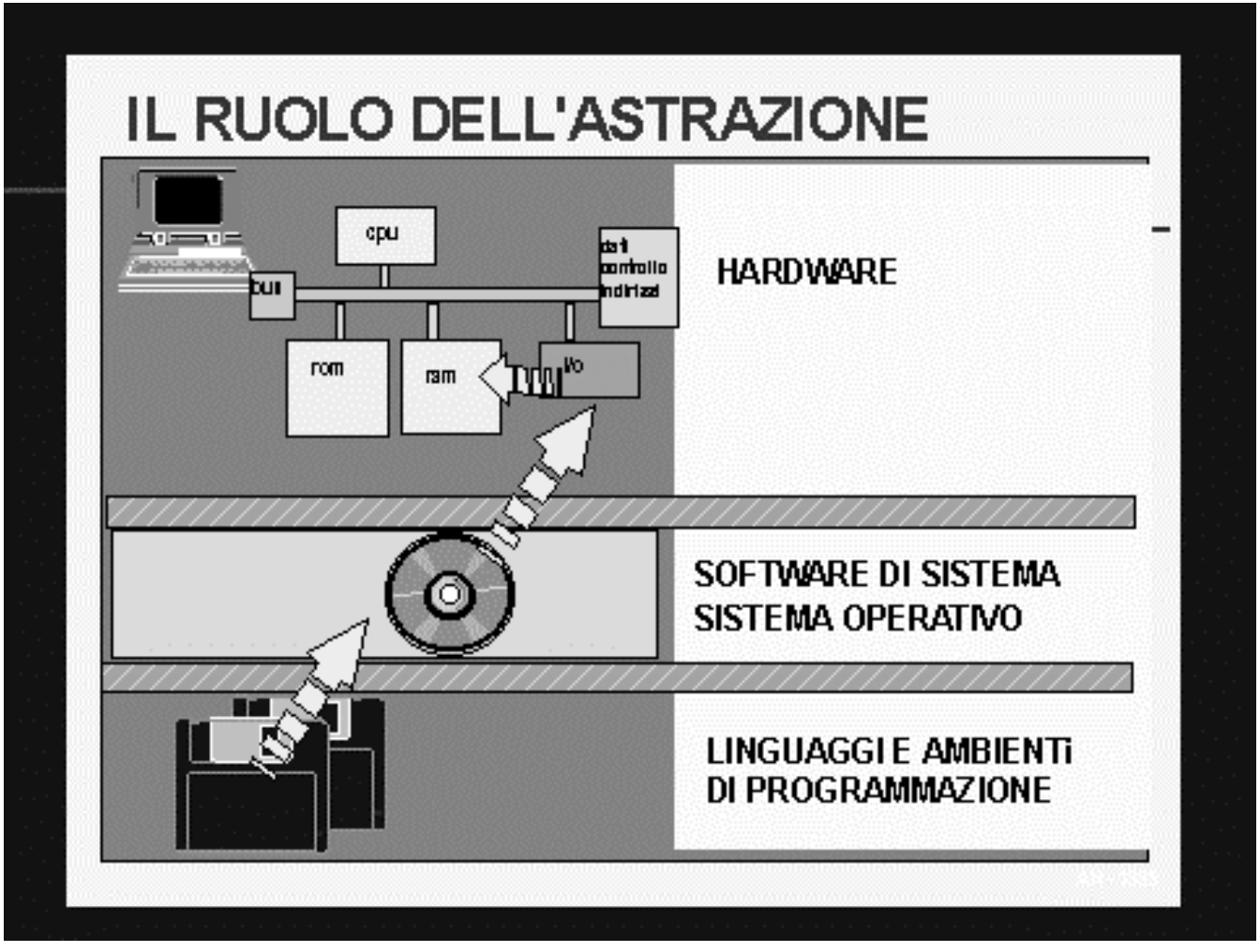
  Ma abbiamo visto che non tutti i programmi codificano algoritmi …

- **Un programma si compone di più istruzioni scritte nel linguaggio di programmazione.**

  - Quali ***parole chiave*** vi sono in un linguaggio?
  - Quali sono i **meccanismi di combinazione** delle parole chiave?
  - E soprattutto, a quale ***macchina*** si riferisce?



Astrazione: come si vede la “macchina”



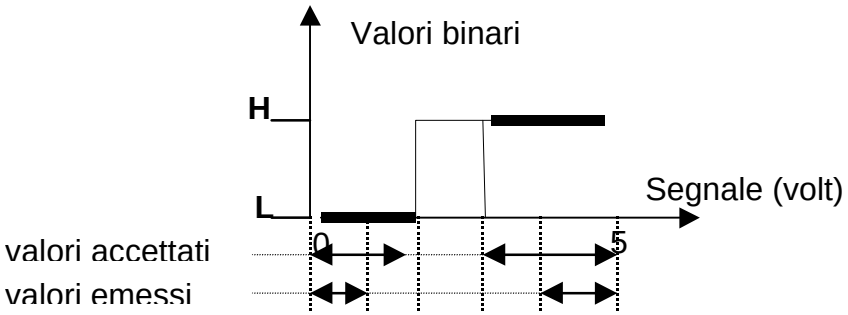
Linguaggio macchina

- **E’ il linguaggio di un computer**
- **Il linguaggio macchina è direttamente eseguibile dall'elaboratore, senza alcuna intermediazione**
- **Computer con architetture interne (CPU) differenti hanno linguaggi macchina differenti**
- **Pertanto, un programma scritto nel linguaggio macchina di un computer non è eseguibile su di un computer con un’architettura differente**

  (Si dice che non è portabile!)

Rappresentazione delle informazioni con segnali digitali binari

La grandezza fisica che si utilizza (segnale elettrico di tensione) assume solo **due valori discreti** (binaria)



L'elemento tecnologico base per la realizzazione di circuiti digitali è il **transistore** il cui funzionamento può essere modellato (in modo molto semplice) come il funzionamento di un interruttore (*aperto* o *chiuso*), quindi con due stati fisici, cui corrispondono 2 opportune tensioni (in genere 0V e 5V)..

BIT (*binary digit*) = cifra binaria. (unità di informazione elementare)
Un bit può assumere due valori che possono essere associati ai simboli:

L(ow)	**H(igh)**	*aspetto fisico del segnale*
0	**1**	*aspetto aritmetico*
false	**true**	*aspetto logico*

Terminologia e «unita' di misura»

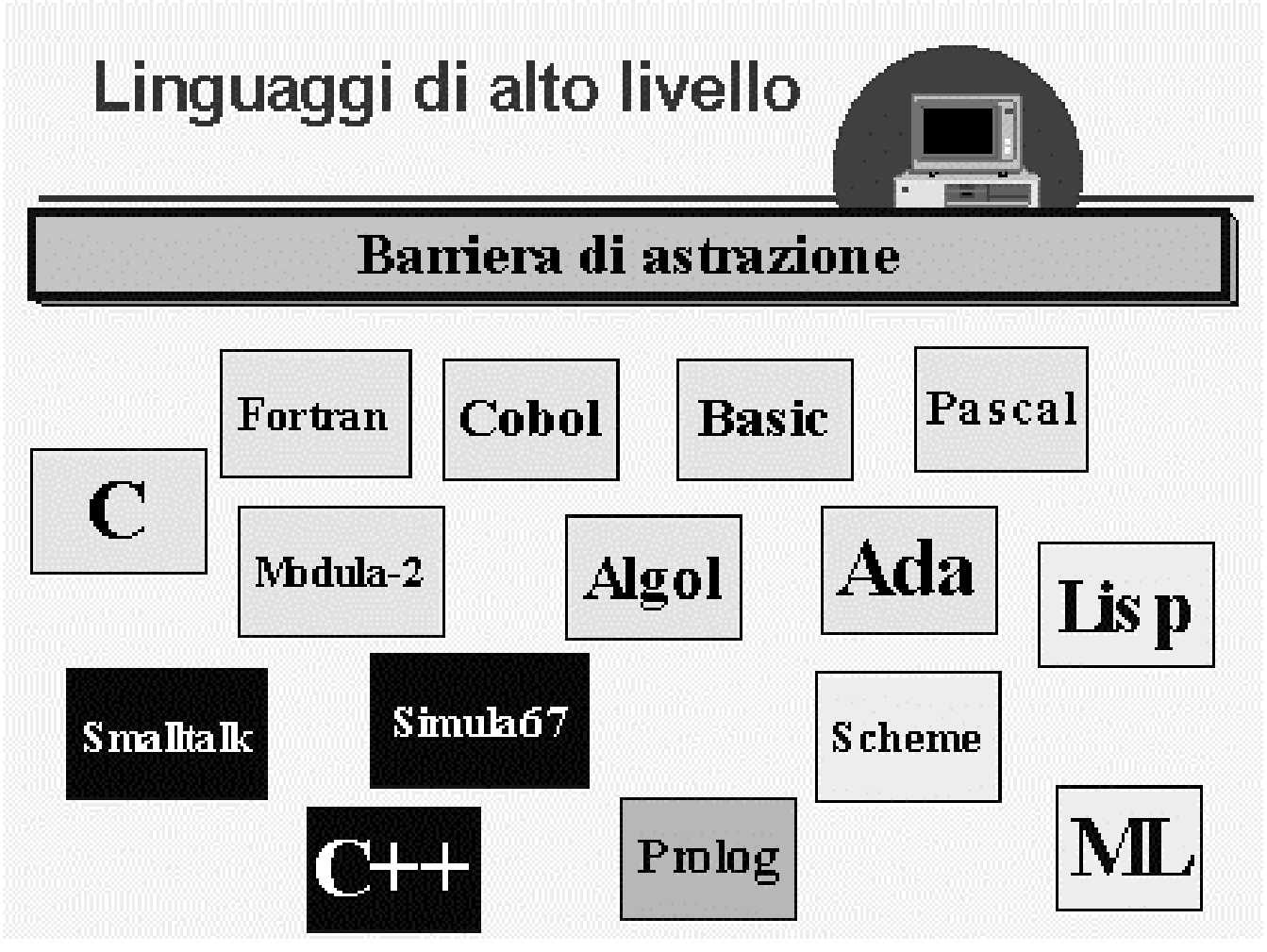
1 cifra	= bit	1 Kilo	= 2^{10} = 1024	$\cong 10^3$
8 bit	= byte	1 Mega	= 2^{20} = 1048576	$\cong 10^6$
16 bit	= word (parola)	1 Giga	= 2^{30}	$\cong 10^9$
32 bit	= double word			
64 bit	= quad word			

Linguaggi di alto livello

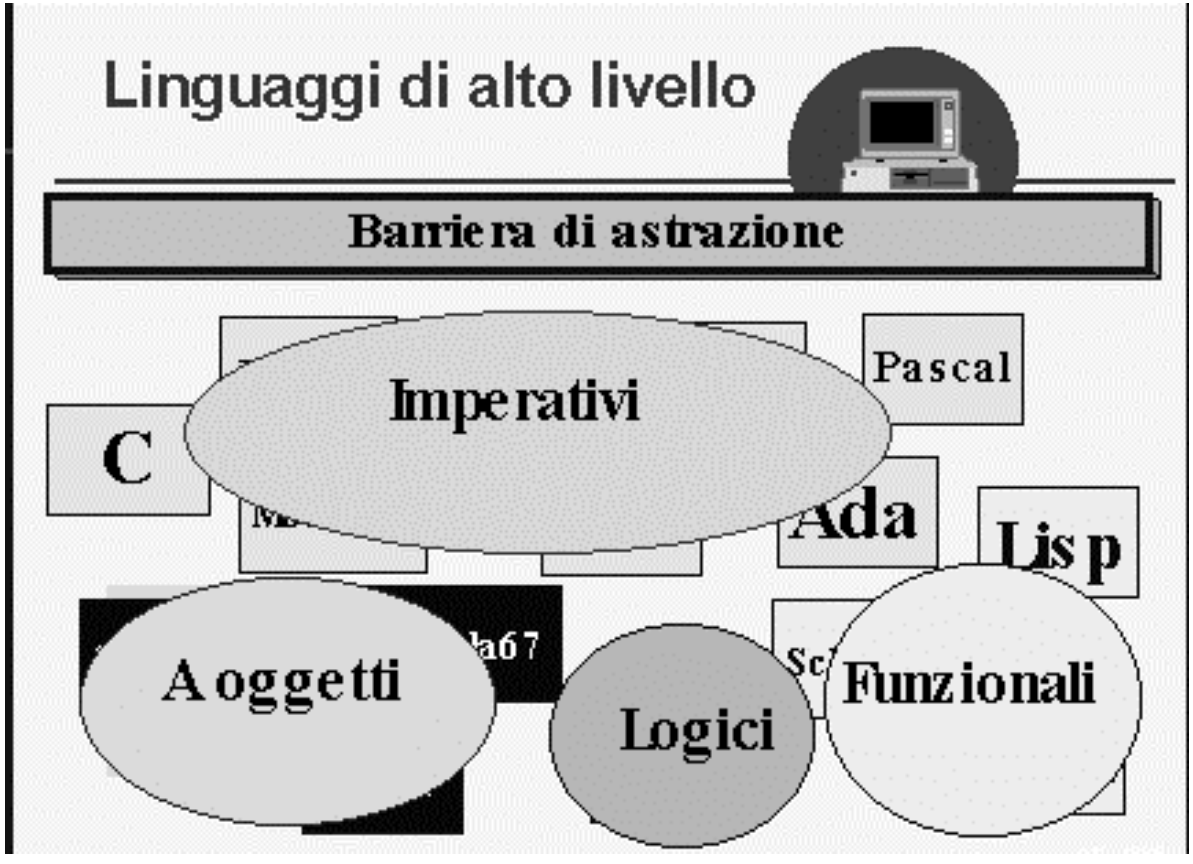
- Si basano su una “macchina” le cui “mosse” non sono quelle della macchina hardware
  $\rightarrow$ realizzano una “***macchina virtuale***”
- Supportano concetti ed astrazioni
- Promuovono metodologie per agevolare lo sviluppo del software da parte del programmatore
- Hanno capacità espressive molto superiori rispetto a quelle del linguaggio macchina

- **Esistono centinaia di linguaggi di programmazione**!
  (anche se pochi sono in uso)

Linguaggi di alto livello (*cont*.)

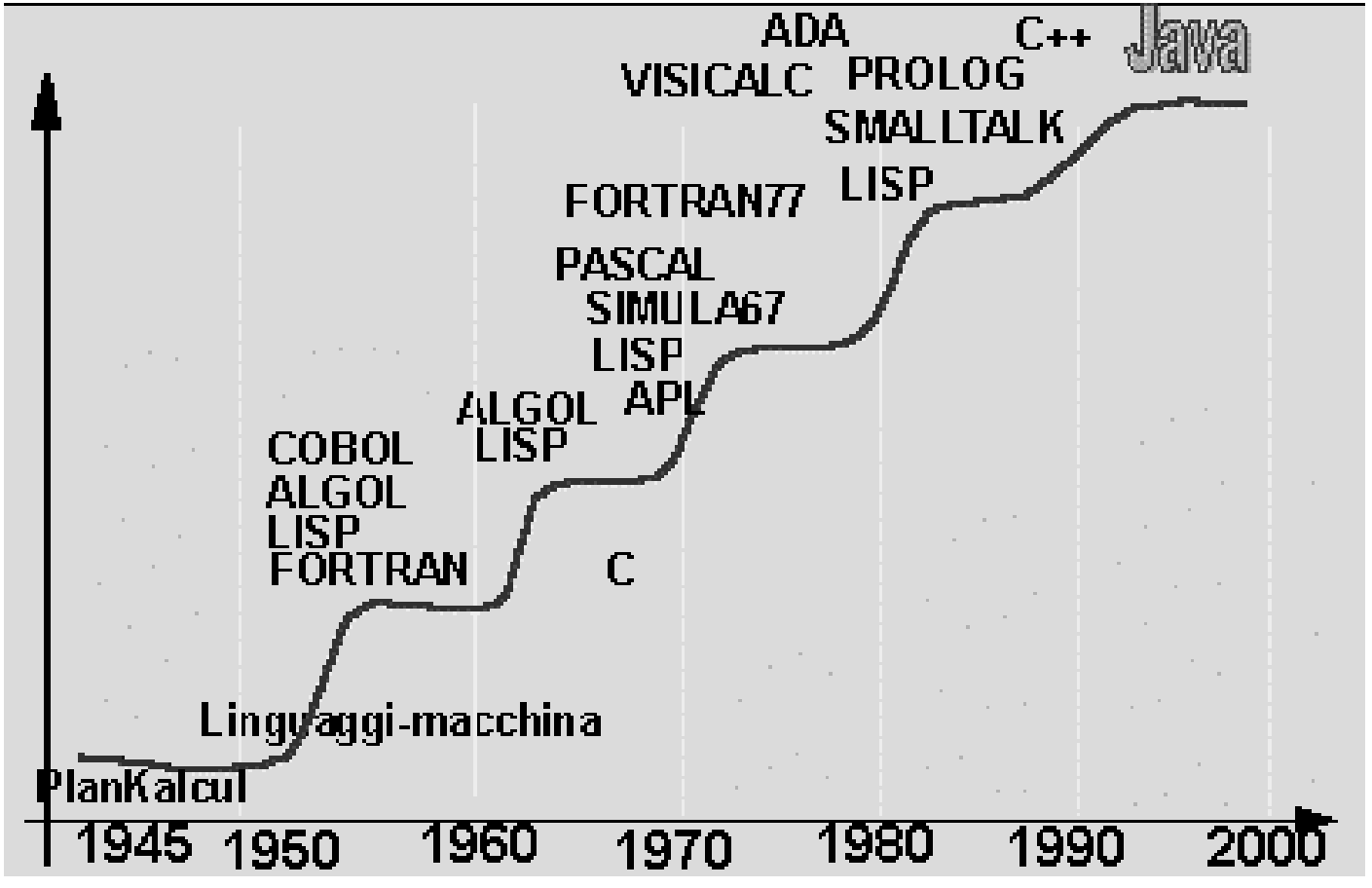


Categorie di linguaggi ad alto livello



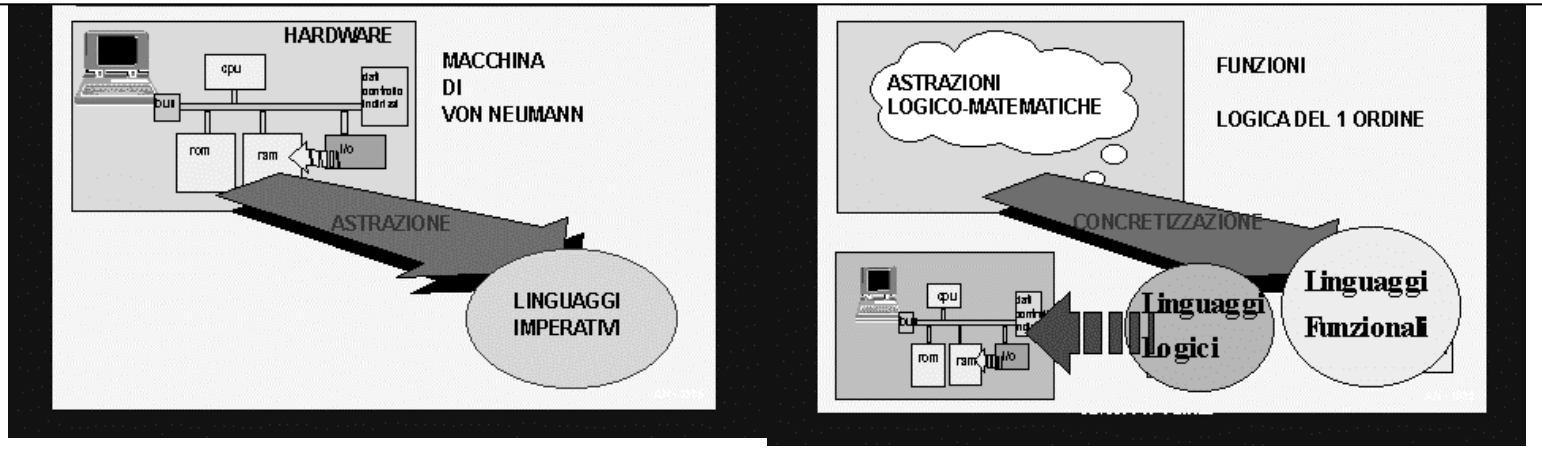
Ogni categoria determina uno specifico stile di programmazione!

Evoluzione dei linguaggi



Perché esistono tanti linguaggi?

- **Contesto applicativo:**
  - Scientifico: **Fortran**
  - Gestionale: **Cobol**
  - Sistemi Operativi: **C**
  - Applicazioni di rete: **Java**
- **Modello di partenza:**



Ambiente di programmazione

- È l'insieme degli strumenti (*tool*) che consentono la codifica, 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 ad alto livello sia comprensibile (e quindi eseguibile) da un calcolatore, ***occorre tradurlo*** **dal linguaggio di programmazione originario al linguaggio comprensibile al calcolatore** (**linguaggio macchina**)
- Questa operazione viene normalmente svolta da speciali programmi, detti **traduttori**

Processo di traduzione

Programma

```
main()
{ int A;
  ...
  A=A+1;
  if ...
```

Traduzione

```
00100101
   ...
11001..
1011100 ...
```

I traduttori convertono il testo dei programmi scritti in un particolare linguaggio di programmazione (***programmi sorgenti***) nella corrispondente rappresentazione in linguaggio macchina (***programmi eseguibili***)

Tipi di Traduttori

Due categorie di traduttori:

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

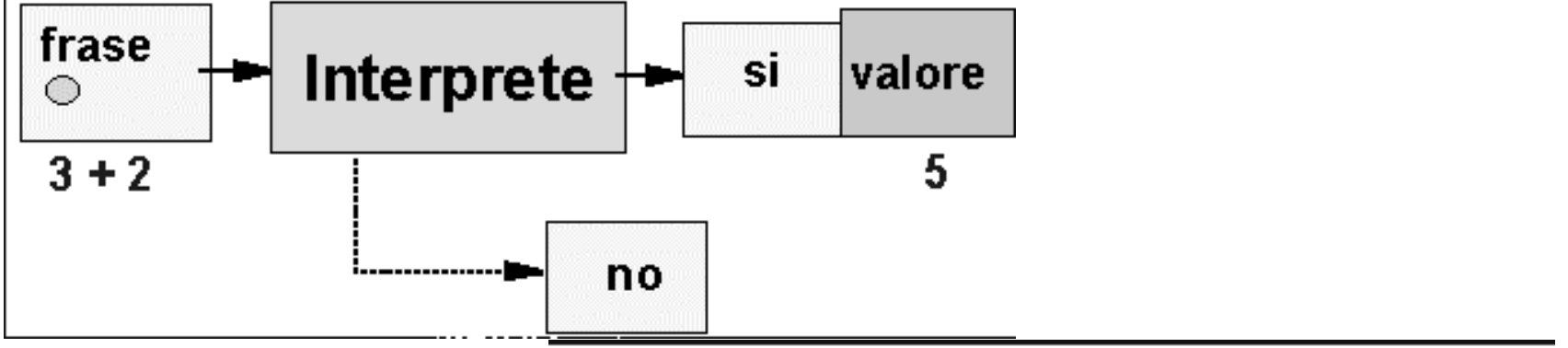
Quindi,

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

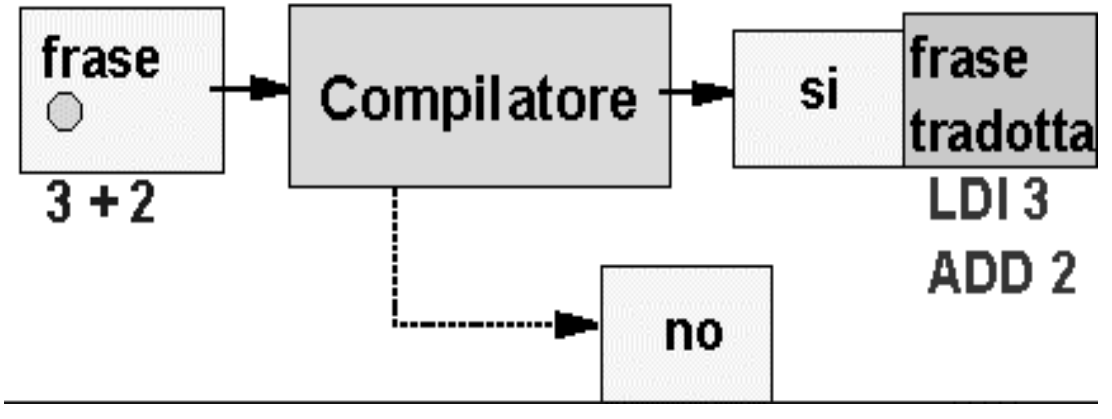
  Ad ogni attivazione dell'interprete su di una particolare istruzione segue l'esecuzione dell'istruzione stessa**.**

Compilatore e Interprete

- Riconoscimento (e valutazione)



- Riconoscimento (e traduzione)



Compilatore e Interprete (*cont*.)

- Sebbene in linea di principio un qualsiasi linguaggio possa essere tradotto sia mediante **compilatori** sia mediante **interpreti**, nella pratica si tende verso una differenziazione già a livello di linguaggio:
  - Tipici linguaggi interpretati: **Basic**, **Javascript, Perl**, …
  - Tipici linguaggi compilati: **C**, **Fortran**, **Pascal**, **ADA**, …

    (*NOTA:* **Java** *costituisce un caso particolare*, anche se si tende a considerarlo interpretato)

Compilatore e Interprete (*cont*.)

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

- Un programma interpretato è più **portabile** di un programma compilato
  (***portabile*** = può essere eseguito su piattaforme diverse)

Ambiente di programmazione

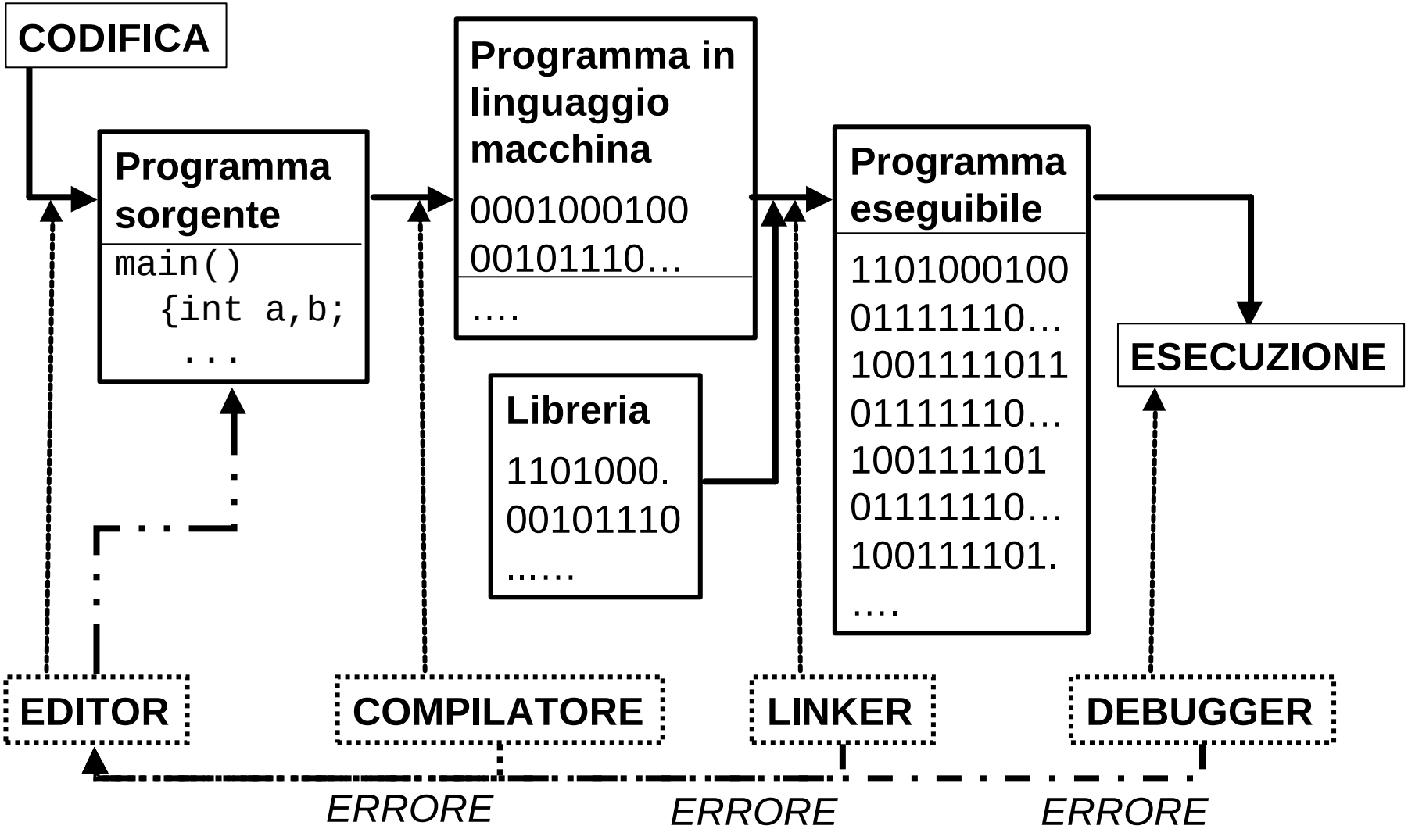
- ***Editor*:** serve per creare file che contengono testi (cioè sequenze di caratteri). In particolare, l'editor consente di scrivere il *programma sorgente*.
- ***Compilatore*:** opera la traduzione di un programma sorgente scritto in un linguaggio ad alto livello in un *programma oggetto*.
- ***Linker*** ("*collegatore*"): nel caso in cui la costruzione del programma oggetto richieda l'unione di più moduli (compilati separatamente), provvede a collegarli per formare un unico *programma eseguibile.*

- ***Debugger*** (*"spulciatore"*): consente di eseguire passo-passo un programma, controllando via via quel che succede, al fine di scoprire ed eliminare errori non rilevati in fase di compilazione.

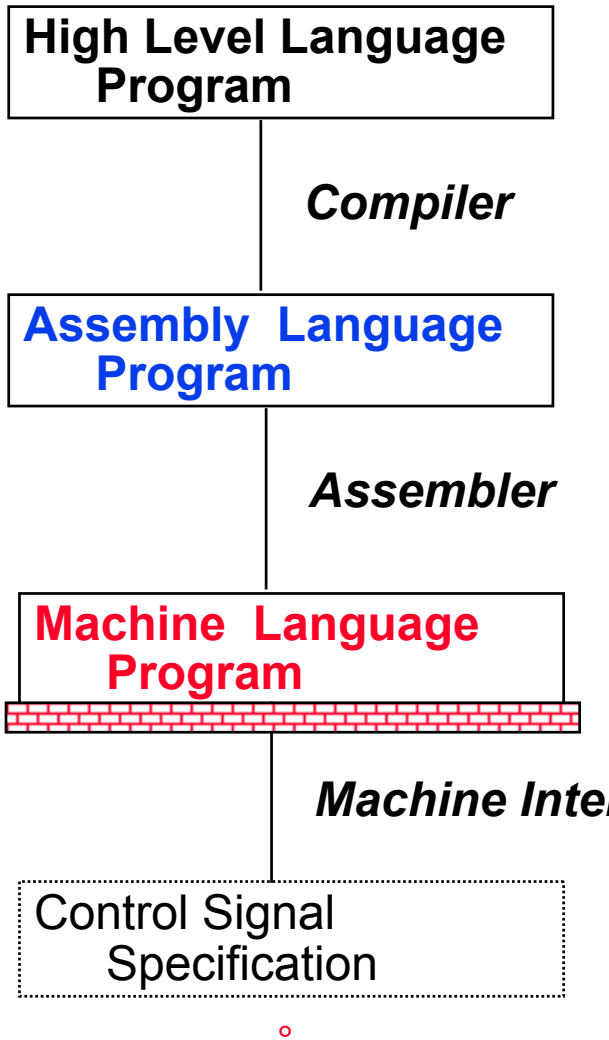
- ***Interprete*:** traduce ed esegue direttamente ciascuna istruzione del *programma sorgente*, istruzione per istruzione.

  È alternativo al compilatore.

Sviluppo ed esecuzione di un programma



Levels of Representation




```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

```
lw$15, 0($2)
lw$16, 4($2)
sw     $16, 0($2)
sw     $15, 4($2)
```

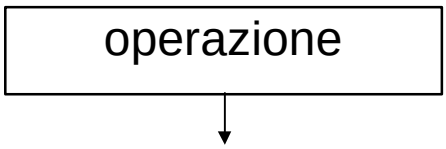
```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```

ALUOP[0:3] <= InstReg[9:11] & MASK

Schemi a blocchi

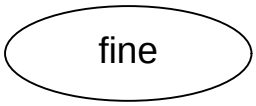
Gli **elementi base,** di tipo grafico, che vengono adottati per la descrizione di algoritmi sono:

- **blocco esecutivo**



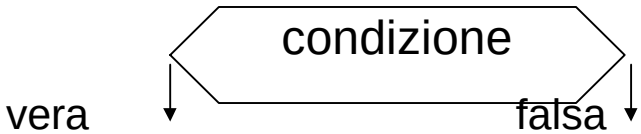
- **blocco di inizio dell'esecuzione**

- **blocco di terminazione**

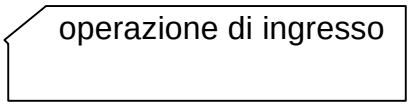


- **flusso di controllo delle operazioni**

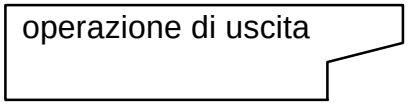
- **blocco condizionale**



- **blocco di ingresso dati**



- **blocco di uscita dati**



Schemi a blocchi

Come in quasi tutti i linguaggi, negli schemi a blocchi le variabili vengono rappresentate tramite nomi simbolici

L'uso di nomi simbolici per le variabili (operandi) consente di descrivere operazioni da eseguire di volta in volta sui diversi valori assegnabili a tali variabili. In tal modo è possibile scrivere algoritmi di

- una **variabile** può essere definita come un «contenitore» di valori

  proprietà

  una variabile non può essere «vuota», cioè ad una variabile è sempre associato un valore che può essere significativo o non significativo

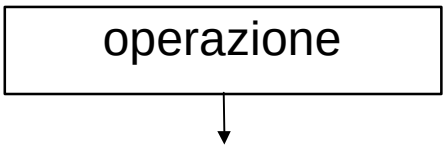
Ovviamente al momento dell'esecuzione di operazioni che «usano» la variabile, essa deve avere un valore significativo. Una variabile ha un valore significativo dopo che è stata effettuata un'operazione di ingresso dati che le attribuisce un valore acquisito dall'esterno, oppure un'operazione di calcolo che le assegna il risultato di un'espressione.

Negli schemi a blocchi le **operazioni** e le **condizioni** vengono espresse in modo testuale ed eventualmente tramite simboli che rappresentano operatori aritmetici, di confronto, ecc.

Schemi a blocchi

Significato degli elementi

BLOCCO ESECUTIVO:



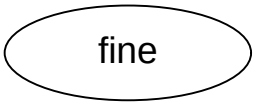
identifica un'operazione da eseguire (espressa in forma testuale al suo interno). Per semplicità di rappresentazione può anche contenere un insieme di operazioni da eseguirsi in modo sequenziale

BLOCCO DI INIZIO DELL'ESECUZIONE:

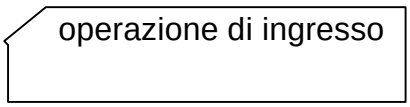
identifica il primo blocco che deve essere eseguito

BLOCCO DI TERMINAZIONE:

contrassegna una possibile terminazione dell'esecuzione dell'algoritmo



BLOCCO DI INGRESSO DATI:

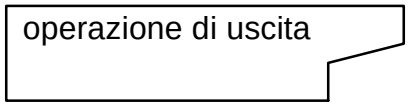


identifica un particolare blocco esecutivo che contiene una direttiva di ingresso dati, cioè un'operazione che consente di acquisire in ingresso al «programma» valori assegnandoli a variabili. Questi valori determinano la particolare *istanza* di esecuzione dell'algoritmo.
Ad esempio l'operazione di ingresso ***Leggi num*** determina l'acquisizione di un valore, inserito dall'utente, che viene assegnato alla variabile di nome simbolico ***num***.
Le direttive di ingresso consentono l'interazione tra «mondo esterno» e calcolatore (esecutore

Le direttive di ingresso più semplici e di uso più comune sono quelle che consentono di acquisire

Ci sono anche operazioni di ingresso da altri dispositivi.

BLOCCO DI USCITA DATI:



identifica un blocco esecutivo che contiene una direttiva di uscita, cioè un'operazione che consente di emettere in uscita dal «programma» frasi di testo e/o valori assunti da variabili.
Ad esempio l'operazione di uscita ***Scrivi num*** produce la visualizzazione del valore che in quel momento è contenuto nella variabile di nome simbolico ***num***.
Le direttive di uscita consentono l'interazione tra calcolatore (esecutore dell'algoritmo) e «mondo

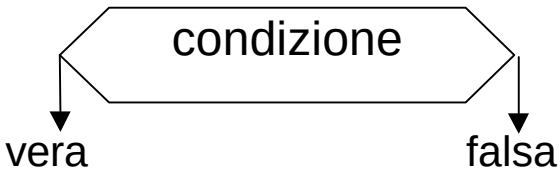
Le operazioni di uscita più semplici e di uso più comune sono quelle che consentono di

Analogamente a quanto disponibile per l'acquisizione di valori in ingresso, ci sono operazioni per l'emissione di dati in uscita su altri dispositivi.

FRECCE E BLOCCO CONDIZIONALE:

consentono di rappresentare in modo completo e univoco il flusso di controllo.

- Generalmente dopo l'esecuzione di un blocco sarà unico il blocco che costituisce il passo successivo.
  La freccia uscente da un blocco esecutivo indica l'unico blocco che segue nel flusso di controllo.
- Quando si presentano due alternative si introducono i blocchi condizionali.



Un blocco condizionale propone un flusso di controllo di selezione tra due alternative. Esso contiene una **condizione**, cioè un'espressione, che può risultare **vera** o **falsa**, a seconda dei valori assunti in quel momento da variabili. In base al risultato della valutazione di tale condizione, il flusso di esecuzione segue uno o l'altro dei due rami rappresentati da frecce uscenti dal blocco condizionale stesso.

Schemi a blocchi

Regole di composizione degli elementi di uno schema a blocchi

La composizione degli elementi ha lo scopo di definire il **flusso di controllo** dell'algoritmo, cioè di definire **in modo univoco** quali sono tutte le possibili sequenze di blocchi da eseguire.
Per garantire questa univocità occorre rispettare le seguenti regole sintattiche.

- Deve esistere un solo blocco d'inizio
- Deve esistere almeno un blocco di terminazione
- Dal blocco d'inizio e da ogni blocco esecutivo (compresi quelli di ingresso e uscita dati) deve uscire una sola freccia
- Da ogni blocco condizionale devono uscire due frecce contrassegnate dalle indicazioni vero (si) e falso (no).

Se per ogni blocco c'è un solo blocco successivo, si parla di flusso di controllo sequenziale (*sequenza*)

Il **flusso di controllo** non è più sequenziale se dopo un blocco si possono presentare diverse *alternative*, cioè diverse sequenze di operazioni da eseguire in alternativa a seconda dell'esito della valutazione di una condizione (in questo caso, come si è detto, si deve usare un *blocco condizionale*).

In **tutti** i casi – **in esecuzione** – e' **unica** la scelta del blocco successivo da eseguire

Parole chiave della lezione

- **Linguaggi di programmazione**
  - Linguaggio macchina
  - Linguaggi ad alto livello
- **Traduttori**
  - Compilatore
  - Interprete
- **Ambiente di programmazione**