- Algoritmo
- Linguaggio macchina
- Linguaggi simbolici
- Compilatori ed Interpreti

Nel nostro vivere quotidiano risolviamo tanti problemi, piccoli o grandi che siano, sempre avendo ben chiaro:

- cosa occorre fare
- su che cosa agire

anche se , quasi sempre, procediamo meccanicamente in quanto trattasi di problemi che affrontiamo con una certa ripetitività

Effettuare i versamenti postali
Selezionare un canale TV
Preparare la cena
Lavare l'auto
……

Nell’ esigenza di delegare la risoluzione di un problema ad altre persone, nel caso siano impreparate a farlo, si rende necessario enunciare loro, in modo *non ambiguo*, una dopo l’altra le operazioni da compiere.

In effetti generiamo a voce ***l’algoritmo*** risolutore del problema

- Accendi il televisore
- Prendi il telecomando
- Individua il tasto nro…
- Pigia il tasto
- Attendi un attimo
- Goditi la partita

Algoritmo

La descrizione , mediante un opportuno linguaggio, di un insieme **finito** di operazioni che, eseguite in un ordine logico prestabilito, permettono la risoluzione di un certo problema

Esempio di algoritmo:

Produrre un documento al PC

- Digitare il testo
- Correggere eventuali errori
- Centrare il titolo
- Giustificare il testo
- Inserire numerazione pagina
- Salvare documento
- Stampare

Ordine fisico delle operazioni
==
Ordine Logico

ma non sempre

Ogni algoritmo per essere realizzato ha bisogno di un *esecutore* che:

- sequenzi l'ordine logico di esecuzione delle operazioni
- le sappia interpretare
- le sappia eseguire

L'esecutore è l'entità che deve realizzare il compito attuando la sequenza di passi

Accendere la brace

Aspettare che la fiamma sia viva

Deporre la carne sulla bistecchiera

Specifica del problema

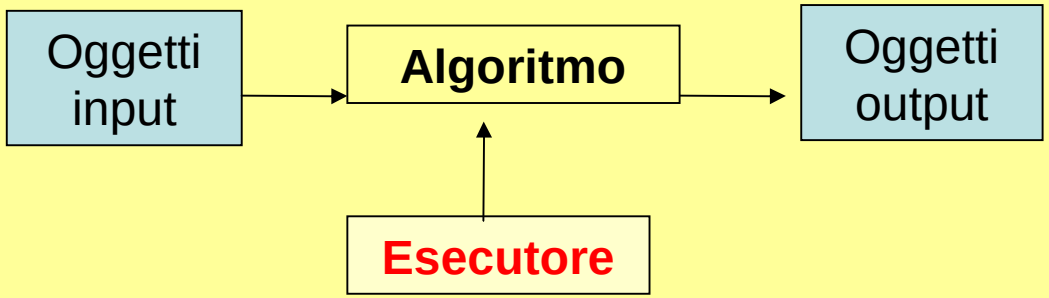
Il progetto di un algoritmo parte dalla individuazione della specifica o traccia del problema da risolvere.

- Descrizione del compito da risolvere
- Indica cosa si deve fare, non come farlo
- Non fa riferimento esplicito ad alcun esecutore
- E' diretta al progettista (programmatore)

Oggetti di ingresso e di uscita

- Ogni compito presuppone oggetti da cui partire (INPUT da elaborare) e degli oggetti da produrre (Output,risultati)

- La specifica deve indicare quali sono questi oggetti di ingresso e di uscita



Evitare specifiche incomplete o ambigue

Esempio:

Traccia	Verniciare un vecchio infisso
Ingresso	Vernice verde, pennello, carta vetro
Uscita	Infisso verniciato
Cosa fare	Pulire e levigare l’infisso con carta vetro Stendere una prima mano di pittura Lasciare asciugare Stendere una seconda mano di pittura

Esempio:

Traccia	Determinare l’area di un cerchio
Ingresso	Raggio, raggio del cerchio:numero reale
Uscita	Area, area del cerchio:numero reale
Cosa fare	3,14 * raggio^2

Casi di test

- A partire dalla specifica (traccia) individuare casi concreti di dati in ingresso da cui derivare i corrispondenti dati in uscita
- Derivazione delle uscite attese
- Utilità dei casi di test
  - Riducono l'ambiguità della specifica
  - Consentono il test dell'algoritmo

Esempio:

Traccia	Divisione fra due numeri
Ingresso	Num1,Num2, dividendo e divisore
Uscita	Valore: valore della divisione
Cosa fare	Num1/Num2

Casi di test:

Num1 =10	Num2=2	Valore=5
Num1 =10	Num2=0	Valore=∞

Ridefinizione traccia

Divisione fra due numeri con divisore diverso da 0

Sequenza statica e sequenza dinamica

Un algoritmo è una sequenza finita di passi di elaborazione

- Sequenza con cui i passi (istruzioni) sono scritti (sequenza statica o lessicografica)
- Sequenza con cui i passi sono eseguiti dall'esecutore (sequenza dinamica)

Le sequenze dinamiche dipendono dalla sequenza statica e dai dati di ingresso

Il linguaggio dell'esecutore

Un algoritmo di solito è formulato nel linguaggio di chi lo progetta.

→ Problemi di comprensione per chi (esecutore) deve eseguire le operazioni descritte.

Si pensi ad una ricetta scritta (algoritmo) in lingua italiana e ad un esecutore ,un cuoco cinese, che ***non conosca*** la lingua italiana (traduzione)

La descrizione della sequenza di operazioni deve essere formulata in un linguaggio comprensibile all'esecutore

↓

Il Programma

Nel caso della macchina informatica::

Oggetti iniziali

Oggetti finali → **INFORMAZIONI**

Esecutore → **ELABORATORE**

Algoritmo → **Programma**

A fronte di algoritmi formulati nel linguaggio di chi li progetta (incomprensibile ,anzi sconosciuto, per l'esecutore macchina informatica)…….

…si rende necessario tradurre l'algoritmo in una sequenza di istruzioni formulate nel linguaggio che la macchina utilizza;

Solo in tal modo l'elaboratore sarà in grado di scandire, interpretare ed eseguire le istruzioni

Le sole operazioni che un processore riconosce (e sa eseguire) sono quelle del suo set di “istruzioni”.

Processori di etichetta diversa hanno set di istruzioni diversi.

La formulazione delle istruzioni deve avvenire secondo un linguaggio (detto ***linguaggio macchina***) che tenendo conto dell’architettura della macchina e del suo set di istruzioni esprima in forma binaria ***cosa operare*** e ***su cosa operare***

Le istruzioni del linguaggio macchina permettono il pieno utilizzo delle risorse interne del calcolatore (***accesso alla memoria, ai registri della cpu,…..***) per eseguire le operazioni

Programmare in linguaggio macchina: ~~Pazzesco~~

Richiede al programmatore, di esprimere l'algoritmo conoscendo:

-l'architettura della macchina
-il set di istruzioni di cui dispone il processore
-l'utilizzo di stringhe di soli simboli "0" ed "1"
-la pianificazione della memoria centrale

Nell'ottica attuale delle plurimarche, quanti linguaggi macchina dovrebbe conoscere un programmatore?

La soluzione: I *linguaggi di programmazione* (simbolici)

svincolano il programmatore da tutte le precedenti problematiche.

Sono tali:
-linguaggio assembler
-linguaggi di alto livello di tipo interprete (Basic,Clipper)
-linguaggi di alto livello di tipo compilativo (Fortran, Cobol, Pascal, SQL,....)

Linguaggio assembler (assembly language)

Ignora il formato binario proprio del linguaggio macchina esprimendo :

- le operazioni (codici operativi) con nomi mnemonici (ADD, TRA, JMP) invece che con valori binari
- gli indirizzi (locazioni) di memoria su cui operare con nomi simbolici (A, B, Primo,..)

È orientato alla macchina (non portabilità**)**

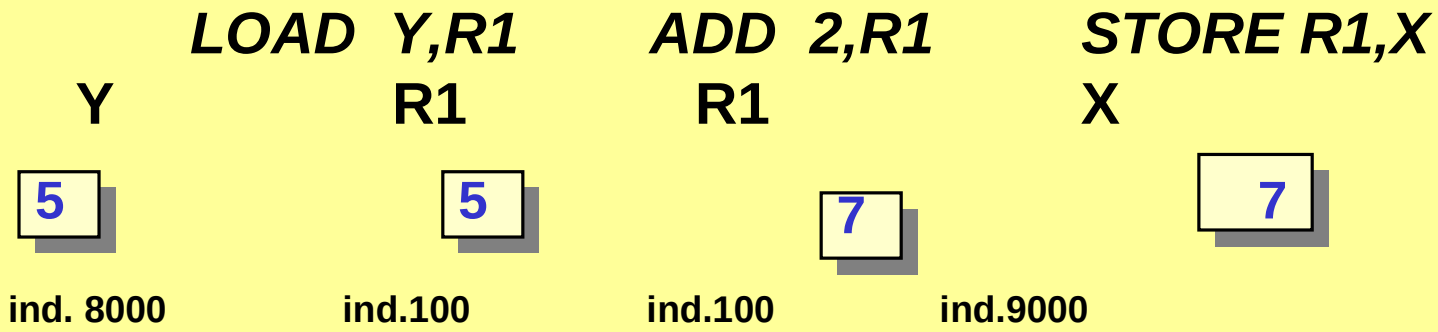
Non è strumento ideale per elaborazioni di una certa complessità

Un esempio di sequenza istruzioni assembler

```
LOAD Y,R1
ADD   2,R1
STORE  R1,X
```

X,Y,R1 sono i nomi simbolici di 3 locazioni(registri) di memoria

il significato delle singole istruzioni:
-carica il contenuto del registro Y nel registro R1
-aumenta di due il contenuto del registro R1
-scarica il contenuto del registro R1 nel registro X



I Linguaggi imperativi di alto livello (o simbolici o evoluti)

Rappresentano la naturale evoluzione della programmazione a tutto vantaggio di chi programma.

Rendono la programmazione semplice e veloce; consentono al programmatore di concentrarsi esclusivamente al problema da risolvere.

Sono definiti come "***linguaggi legati ai problemi***".

Sono standard, ovvero non legati alla specifica macchina , il che assicura la portabilità delle applicazioni.

FORTRAN	RPG	JAVA
COBOL	BASIC	ALGOL
PASCAL	HTML	C
.....	.....	.....

I linguaggi evoluti sono standard

Fortran	Basic	Prolog	Ada	Java
Algol 60	Algol 68	Modula 2	C++	HTML
Cobol	Pascal			SQL

Per evidenziare la notevole semplificazione consentita da questi linguaggi

```
X= Y + 2
```

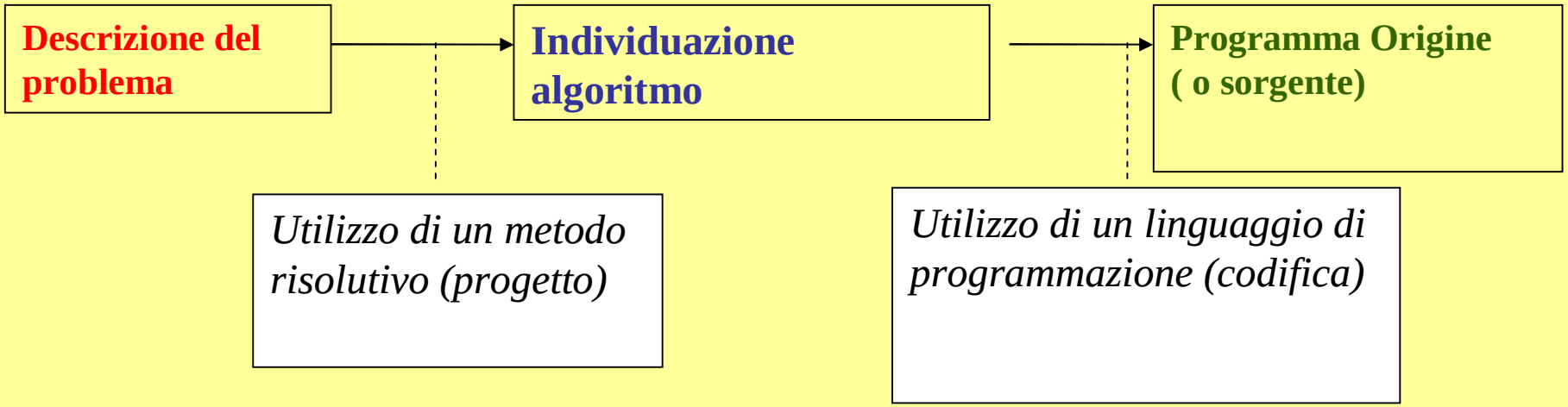
```
LOAD Y,R1
ADD   2,R1
STORE  R1,X
```

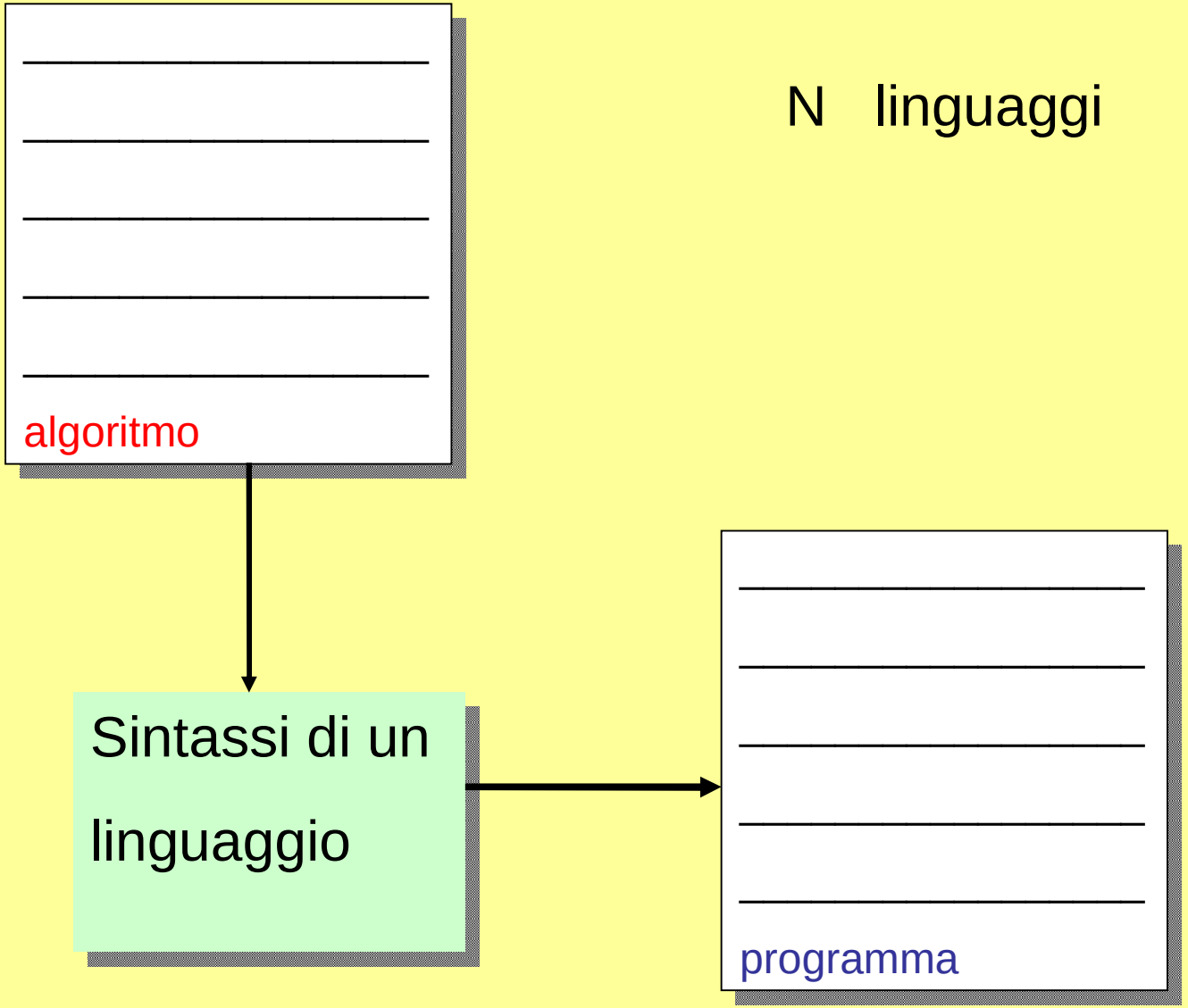
Algoritmo → Programma

*Se **l'algoritmo*** descrive la sequenza di azioni risolutoria di un problema (nel linguaggio di chi la progetta, es. lingua italiana)…….

…. il ***programma*** è la rappresentazione di un algoritmo mediante un "linguaggio di programmazione" (utilizzando la relativa sintassi)

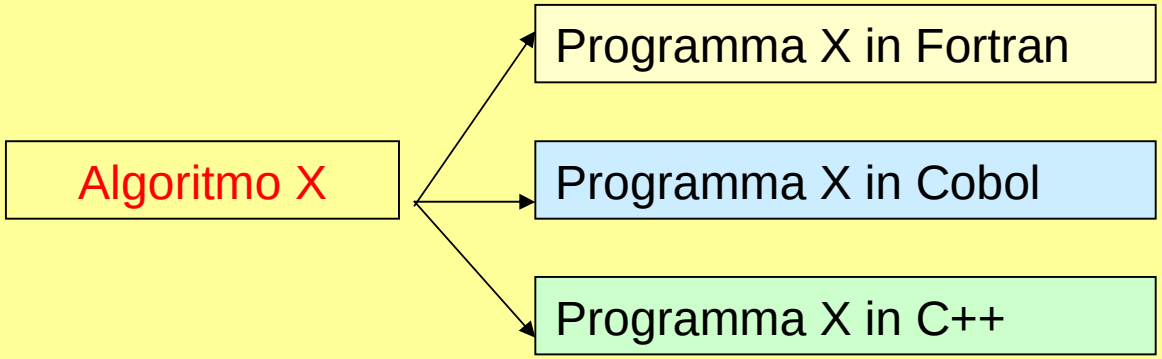
Tale operazione è la codifica del programma





N linguaggi

Il rapporto algoritmo ➔programma è del tipo ***1 a N***, nel senso che partendo da uno stesso algoritmo si possono ottenere N programmi tanti quanti sono i linguaggi disponibili



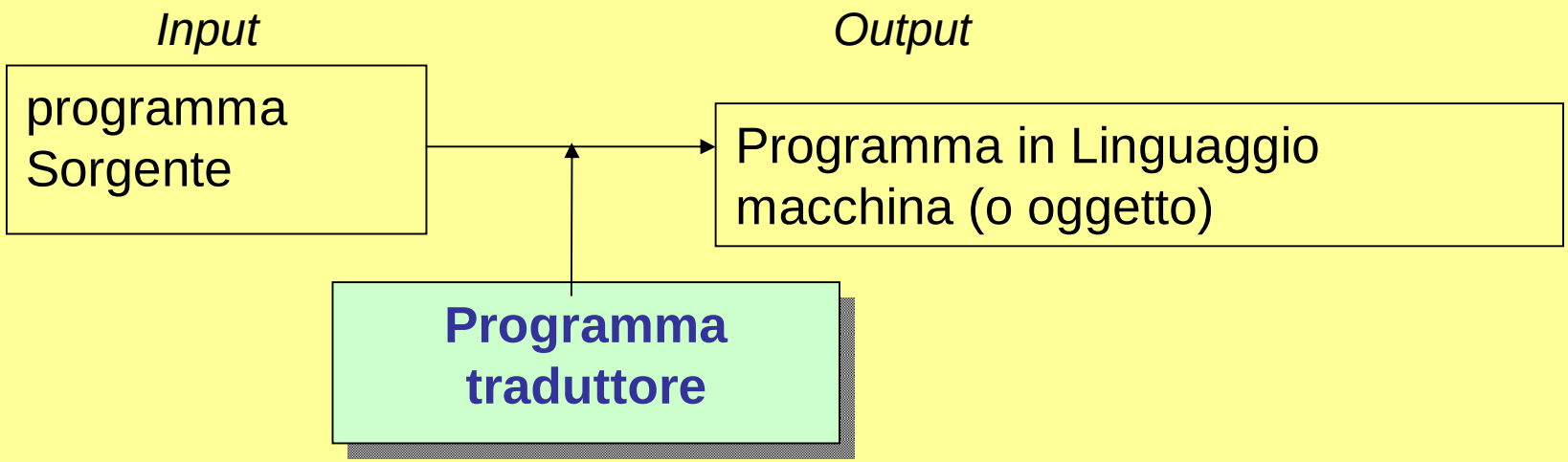
Algoritmo : sequenza finita di azioni atte a risolvere un problema

Linguaggio : insieme di frasi interpretabili da un esecutore (*automa*) in grado di comprenderne il contenuto operativo e di compiere le operazioni in esse indicate. Le frasi rispettano precise regole formali

Programma : rappresentazione di un algoritmo mediante un linguaggio. Rispetto assoluto della sintassi del linguaggio.

Codifica : Operazione che dall' algoritmo genera il programma

Un programma origine non può essere compreso dalla macchina informatica (***processore***)



Il traduttore è tipico del linguaggio usato e della macchina(processore)

Il programma in linguaggio macchina è eseguibile dal processore reale

La traduzione passa attraverso tre fasi di analisi :

- Lessicale
- Sintattica
- Semantica

-**Analisi lessicale** (*scanning*) : verifica il rispetto delle regole di aggregazione dei caratteri dell'alfabeto del linguaggio nella generazione dei simboli (identificatori, parole chiavi, numeri,operatori, stringhe) del linguaggio

Es. Aut?omobile in C++ è una cattiva aggregazione in quanto il carattere ? non è contemplato come carattere dell'alfabeto

Es. Precipitevolissimevolmente è una cattiva aggregazione in quanto eccede la lunghezza max di aggregazione

-**Analisi sintattica** (parsing): verifica la correttezza , nel rispetto della rigida sintassi del linguaggio,del concatenamento dei simboli nella costruzione delle frasi.

La semplice mancanza di una virgola, uno scorretto bilanciamento delle parentesi, possono essere fonti di errore.

-**Analisi semantica**: verifica il corretto significato della frase (*il senso realistico*).

La frase **A=”Napoli”+2**; è sintatticamente corretta in quanto rispetta la sintassi di una frase di assegnazione e calcolo, ma è ***semanticamente scorretta*** in quanto non ha senso procedere alla somma di caratteri e numeri.

Nel processo di traduzione vengono generate ed utilizzate diverse tabelle:

- dei simboli,
- dei simboli futuri
- dei riferimenti esterni
- ----------

Un programma, scritto in linguaggio ***simbolico***, è una sequenza di frasi distinguibili in:

- ***dichiarative*** o dichiarazioni
- ***esecutive*** o istruzioni

Dichiarative

Forniscono informazioni utili al programma traduttore

Esecutive

Descrivono le operazioni che devono essere effettuate.

Es. di frasi dichiarativa in ambiente Fortran:

```
Integer  A,B
Logical   Flag
Real Area
Integer  Matrice(10,10)
```

Dichiarazione del tipo di variabili semplici o strutturate

La tipologia delle frasi esecutive:

- ►di ingresso
- ►di uscita
- ►di calcolo
- ►di chiamata a sottoprogramma
- ►di controllo sequenza

```
Integer a,b     (dich.)
Real c          (dich.)
Read a          (ing.)
Read b          (ing.)
c=a/b           (calc.)
Write c         (usc.)
```

Due tecniche di traduzione

compilativa : il SW di traduzione viene detto compilatore o traduttore

interpretativa: il SW di traduzione viene detto interprete

Ne consegue che :
un esecutore di un linguaggio simbolico e' costituito dalla coppia:

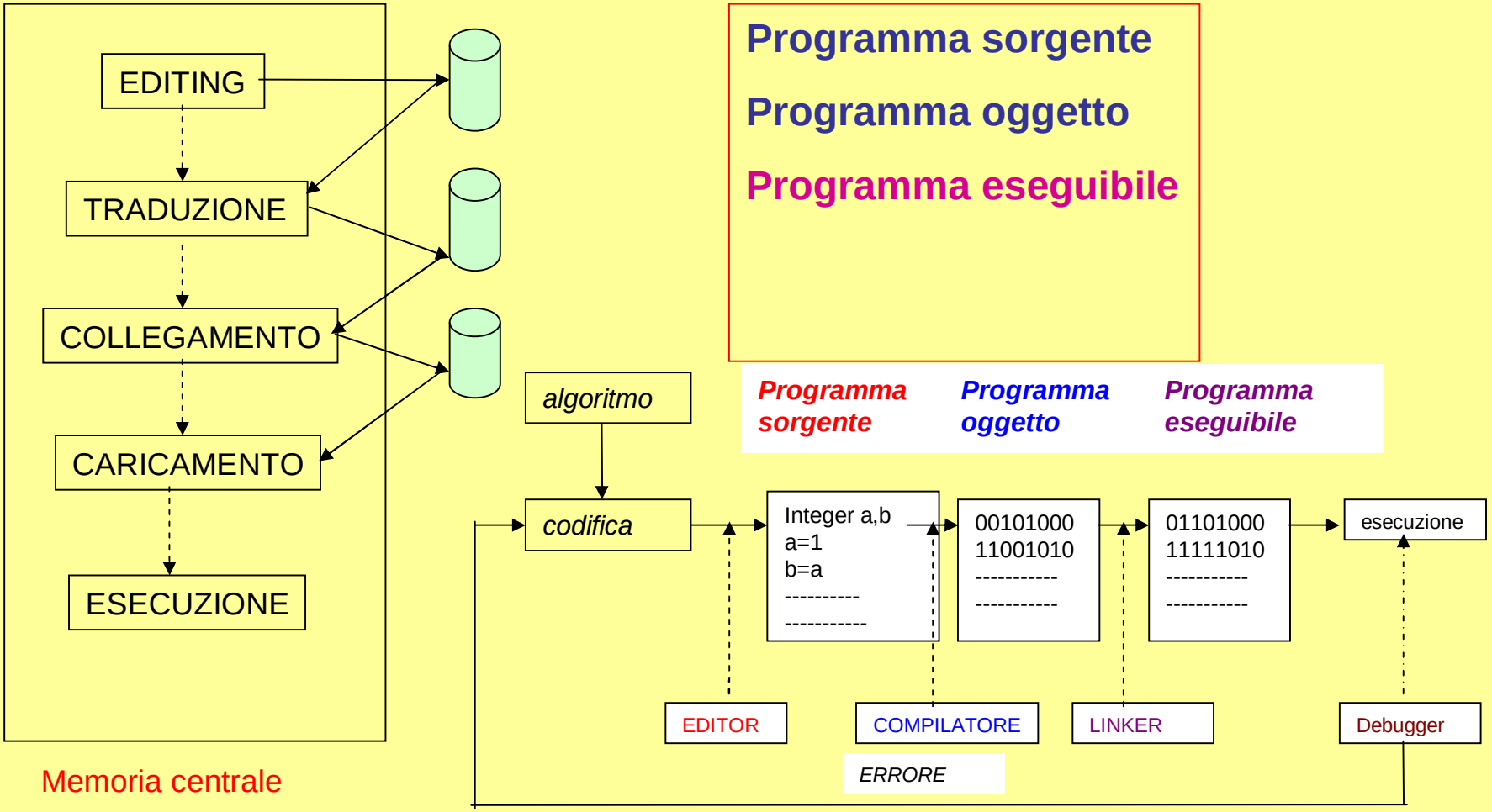
Compilatore,processore oppure

Interprete, processore

Basic, Clipper,…..	**Interpretativi**
Cobol, Fortran, ….	**Compilativi**

Tecnica compilativa

Il processore reale esegue un programma in più passi:



Editing

il programma ***origine*** (o sorgente) viene scritto con l'ausilio di un programma *editor* (videoscrittura) nel linguaggio simbolico scelto e memorizzato su supporto esterno di memoria (disco)

Traduzione

Il programma origine viene tradotto in linguaggio macchina. La traduzione è curata dal processore che, eseguendo il programma *compilatore*, che legge il programma origine,
produce il programma ***oggetto*** e lo deposita su memoria esterna

Collegamento

Il processore esegue il programma *collegatore* che preleva dalla memoria di massa i moduli da collegare (mettere insieme) e genera il programma ***eseguibile*** con deposito su supporto esterno

Caricamento

Il processore esegue il programma *caricatore* che posiziona il programma eseguibile in memoria centrale

Esecuzione

Il programma esiste in memoria centrale nelle vesti di linguaggio macchina e viene eseguito dal processore con lettura dei dati di input (se richiesti) e produzione in output dei risultati

Tecnica interpretativa

- Il processore esegue il programma simbolico in un unico passo in cui esegue il programma interprete
- Il programma interprete legge ,dal supporto su cui e' memorizzata, una istruzione alla volta del programma origine e ne produce le azioni elaborative richieste
- Il processore esegue il programma origine non eseguendo direttamente le sue istruzioni (che non e' in grado di capire) ma agisce sulle istruzioni dell'interprete che interpretano le istruzioni del programma origine

Non esiste passaggio: Origine → Oggetto → Eseguibile

La tecnica interpretativa:

- eseguendo il programma origine *passo* dopo *passo* potrebbe essere particolarmente utile nella fase di messa a punto di un programma
- non produce la traduzione in linguaggio macchina del programma sorgente; necessario ogni volta eseguire l'attività dell'interprete
- non consente l'utilizzo di sottoprogrammi
- più lenta della tecnica compilativa