Capitolo 2
La formalizzazione dell'informazione

Introduzione ai sistemi informatici

Il calcolatore come strumento per gestire informazione

tradizionalmente l'enfasi è su

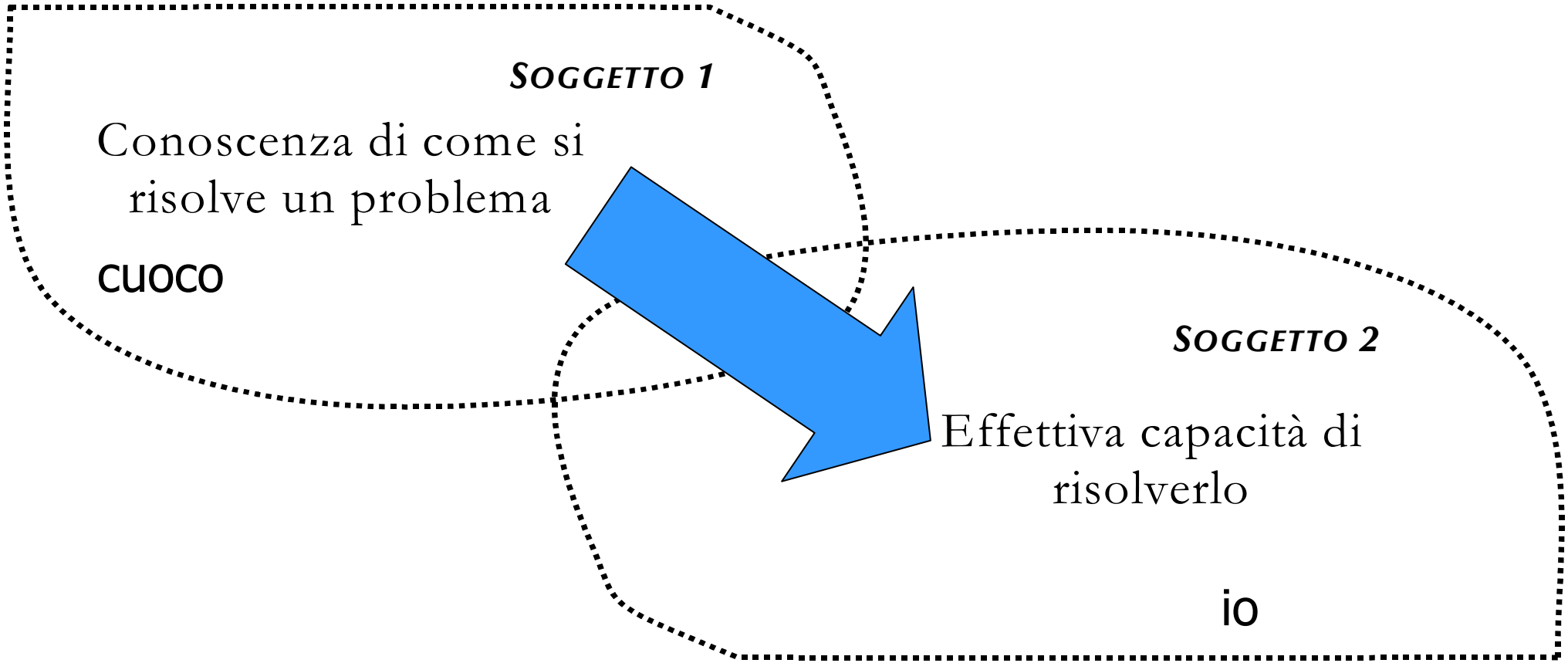
… cioè sui

sistemi di elaborazione delle informazioni …

Soluzione di un problema

Deve fare tutto lo stesso soggetto?

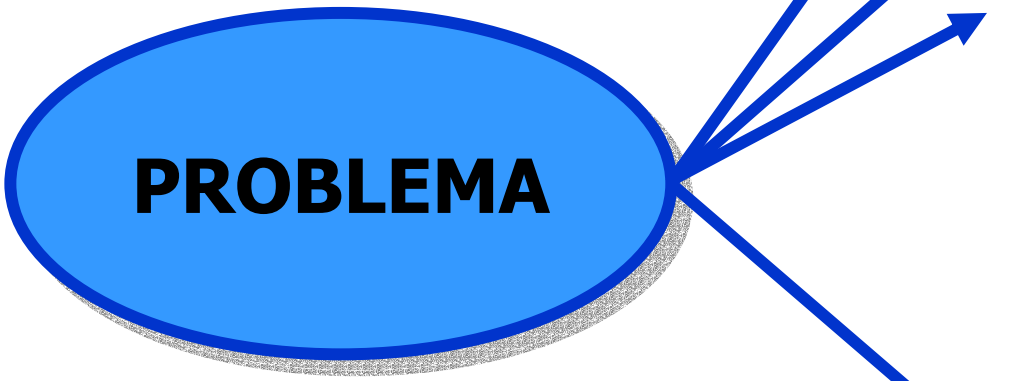
Il processo di soluzione di un problema



Introduzione

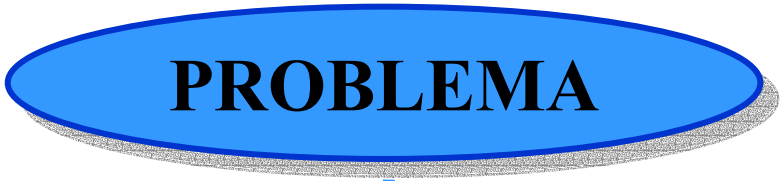
- Dalla specifica di un problema al linguaggio macchina: la fase dell'***Analisi.***
  - Comprensione del problema
  - Modellazione del problema
  - Ricerca della soluzione

Comprensione del Problema



1. **Eliminare ogni ambiguità della formulazione**
2. **Focalizzare gli obiettivi**
3. **Evidenziare**
   - **I reali obiettivi del problema**
   - **Le regole**
   - **I dati espliciti ed impliciti**
4. **Eliminare**
   - **I dettagli inutili ed ambigui**

Modellazione del problema



Modello ⟶ Rappresentazione semplificata della situazione in esame che esplicita:

- gli elementi presenti, le loro proprietà e le relazioni tra essi

Il modello può essere

- grafico
- tabellare
- simbolico

} Possono coesistere per generare un unico modello

Modello

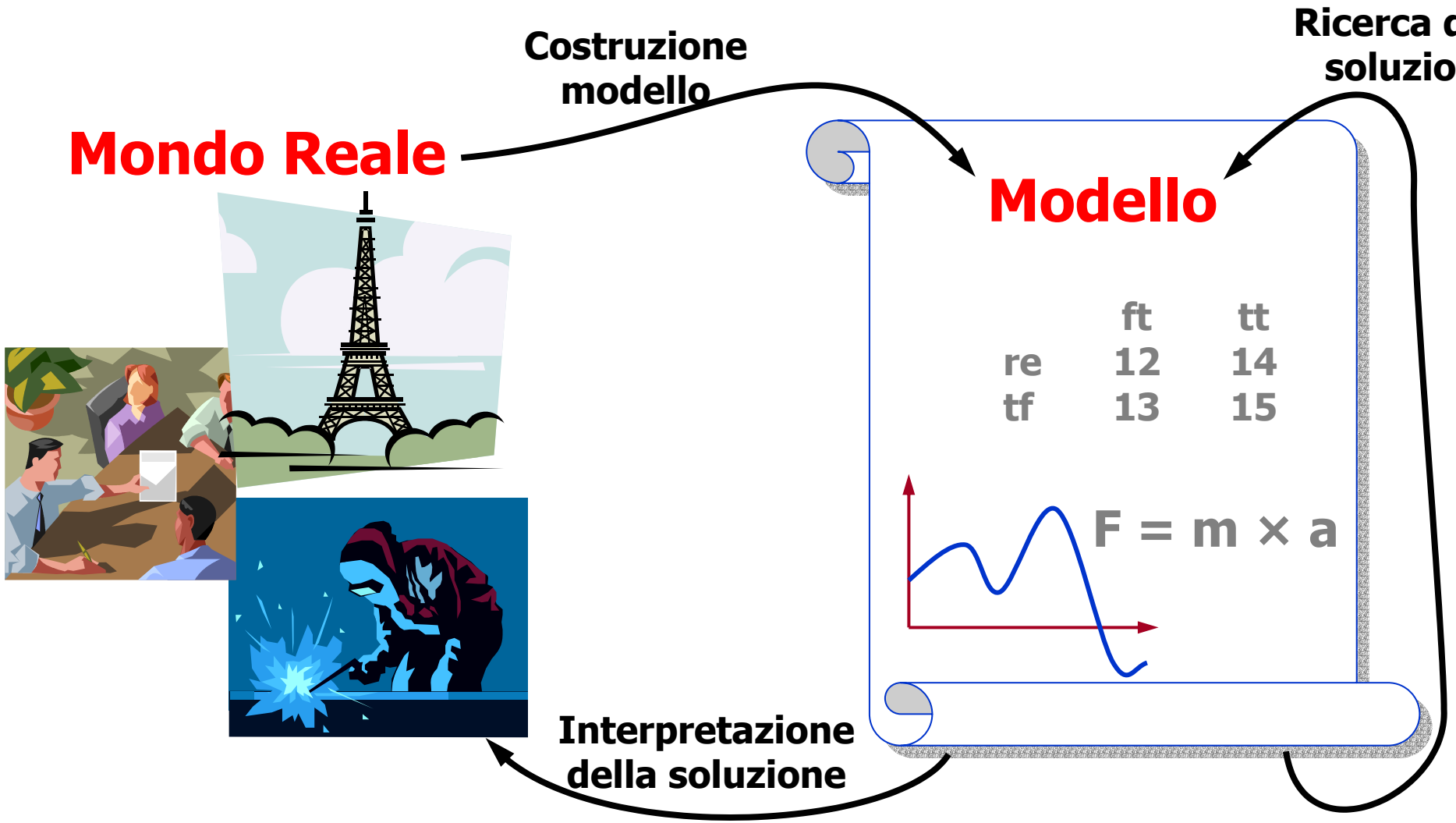
Modello

- non è sufficiente a risolvere il problema
- è un ausilio alla risoluzione del problema
- è la base per la ricerca del risultato

Importante:

la soluzione trovata sul modello deve essere interpretata correttamente al fine di poterla trasferire sulla realtà

Relazione tra Realtà e Modello



Dal problema all'algoritmo

Un esempio di problema…

- Problema
  - **Come si cucina un uovo al burro?**
- Soluzione
  - Far sciogliere in un tegamino 20 g. di burro,
  - quando il burro assume un colore dorato
  - rompere il guscio dell'uovo e
  - far scivolare delicatamente nel tegamino albume e tuorlo.
  - Salare.
  - Quando l'albume è ben rappreso spegnere il fuoco.

Dal problema alla soluzione

La soluzione è espressa come una sequenza di operazioni la cui esecuzione porta alla soluzione del problema → l'algoritmo risolutivo

Una definizione più precisa...

Si definisce *algoritmo* una *sequenza di azioni*
che trasformi i dati iniziali in un numero finito di passi,
elementari e non ambigui, per giungere al risultato finale.

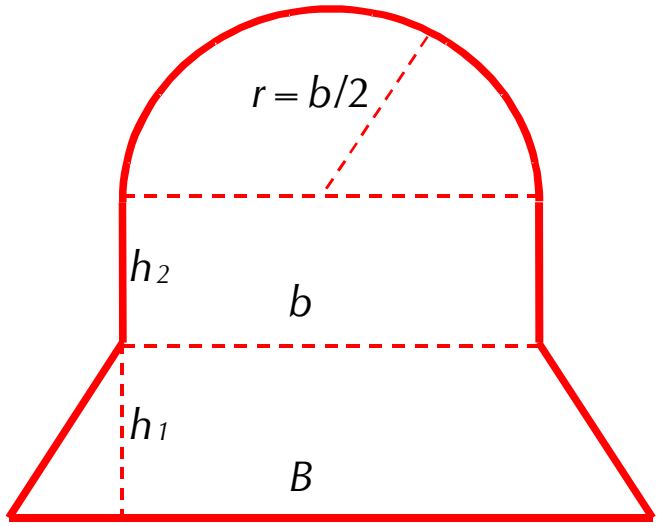
Questa sequenza di azioni è valida
per un insieme di dati iniziali ben definito
e può essere eseguita da un opportuno esecutore.

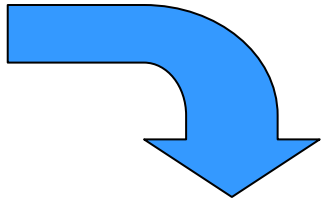
…e se non so come "rompere il guscio"?

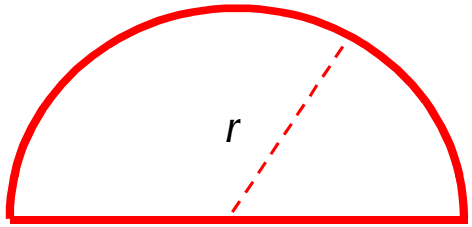
- Tutte le operazioni specificate dall'algoritmo devono essere eseguibili dall'esecutore… (in questo caso si chiamano operazioni elementari)

- … altrimenti è necessario "scomporre" il problema troppo complesso in sottoproblemi più semplici:
  - ***Rompere il guscio*** *= colpire con un gesto secco ma leggero il guscio dell'uovo con il dorso di un coltello. Tenendo verticale l'uovo, aprirne il guscio inserendo l'unghia del pollice nell'incavatura formatasi nel guscio*

Un altro esempio: l'area di una campana



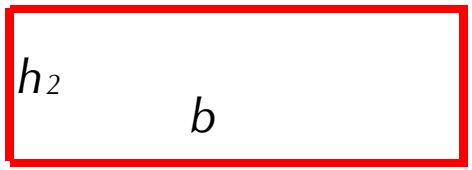




Sottoproblema 1

soluzione effettiva:

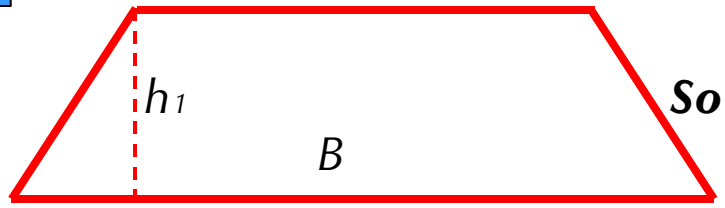
$s = \frac{1}{2} \pi r^2$



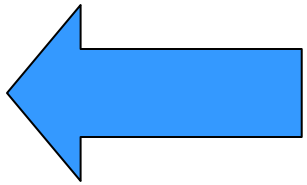
Sottoproblema 2

soluzione effettiva:

$s = b\ h_2$



Sottoproblema 3



$$s = \frac{1}{2} \pi r^2 + b\ h_2 + \frac{1}{2} (\frac{1}{2}(B-b)\ h_1) + \frac{1}{2} (\frac{1}{2}(B-b)\ h_1) + b\ h_1$$

Esempio: gestione biblioteca

- Libri disposti sugli scaffali
- La posizione di ogni libro è fissa ed è individuata da due coordinate:
  - **Scaffale** (i.e. numero dello scaffale)
  - **Posizione** nello scaffale
- La biblioteca è dotata di uno schedario (ordinato per autore/i e titolo). Ogni scheda contiene, nell'ordine:
  - **cognome** e **nome** dell'autore
  - **titolo** del libro
  - **data** di pubblicazione
  - **numero dello scaffale** in cui si trova
  - **posizione** attribuita al libro nello scaffale.

Esempio di scheda

Autore/i:	**Sciuto, Donatella** **Buonanno, Giacomo** **Fornaciari, William** **Mari, Luca**
Titolo:	**Introduzione ai Sistemi Informatici, II Edizione, 2002**
Scaffale:	**22**
Posizione:	**11**

Formulazione dell'algoritmo

1. Decidi il libro da richiedere
2. Preleva il libro richiesto

Se un passo dell'algoritmo non è direttamente comprensibile ed eseguibile dall'esecutore, occorre dettagliarlo a sua volta mediante un algoritmo.

Questo modo incrementale di procedere si dice **top-down** o anche procedimento per **raffinamenti successivi**.

Un algoritmo per il prelievo

1. Decidi il libro da richiedere
2. Cerca la scheda del libro richiesto
3. Segnati numero scaffale e posizione
4. Cerca lo scaffale indicato
5. Accedi alla posizione indicata e preleva il libro
6. Scrivi i tuoi dati sulla "scheda prestito"

Il "sotto-algoritmo"

1. Prendi la prima scheda.
2. Esamina se titolo e autore/i sono quelli cercati.
   In caso positivo la ricerca termina con successo,
   altrimenti passa alla scheda successiva e ripeti.
3. Se esaurisci le schede il libro cercato non esiste.

- Cosa succede se l'autore cercato è "Zombie Zuzzurellone"?

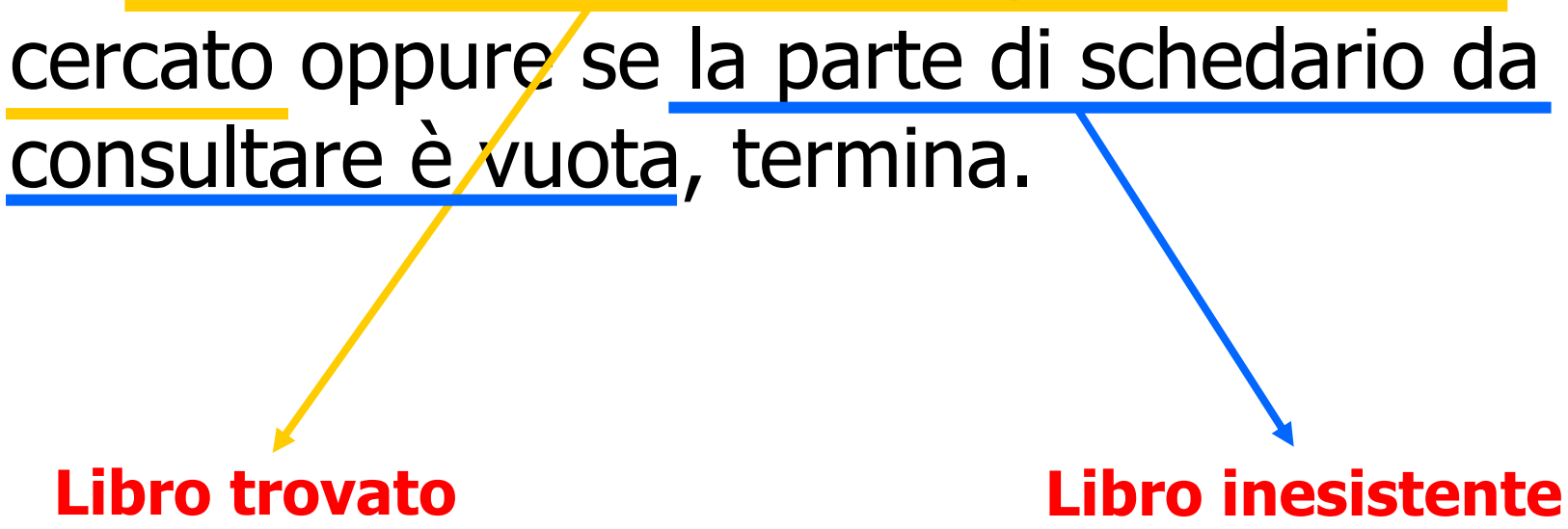
Un "sotto-algoritmo" migliore

1. Esamina la scheda centrale dello schedario.
2. Se la scheda centrale corrisponde al libro cercato allora termina.
3. Altrimenti, prosegui allo stesso modo nella metà superiore o inferiore dello schedario a seconda che il libro cercato segua o preceda quello indicato sulla scheda.

- **L'algoritmo è incompleto: c'è un'altra condizione di terminazione quando il libro non esiste.**

Revisione del passo 2

Se la scheda centrale corrisponde al libro cercato oppure se la parte di schedario da consultare è vuota, termina.



Esempio: algoritmo del risveglio

1. Alzarsi dal letto
2. Togliersi il pigiama
3. Fare la doccia
4. Vestirsi
5. Fare colazione
6. Prendere il bus per andare a scuola

NB: I passi sono eseguiti in sequenza e l'ordine delle istruzioni è essenziale per la correttezza dell'algoritmo!

Non basta organizzare i passi in sequenza

1. Alzarsi dal letto
2. Togliersi il pigiama
3. Fare la doccia
4. Vestirsi
5. Fare colazione
6. **Se piove**
   - **prendere ombrello**
7. Prendere il bus per andare a scuola

Ulteriore forma di flusso: se…altrimenti

1. Alzarsi dal letto
2. Togliersi il pigiama
3. Fare la doccia
4. Vestirsi
5. Fare colazione
6. **Se piove**
   prendere la macchina
   altrimenti
   prendere il bus

Ulteriore forma di flusso: ciclo "mentre"

1. Alzarsi dal letto
2. Togliersi il pigiama
3. Fare la doccia
4. Vestirsi
5. Fare colazione
6. Mentre piove
   restare in casa
7. Prendere il bus per andare a scuola

Le strutture di controllo: SEQUENZA

- Le istruzioni devono semplicemente essere eseguite nell'ordine in cui sono presentate

- Esempio:
  - solleva il ricevitore
  - componi il numero
  - …

Le strutture di controllo: ITERAZIONE

- Le istruzioni devono essere eseguite ripetutamente fino a che non si verifica una determinata condizione

- Esempio:
  - RIPETI
    - componi una cifra
  - FINO al completamento del numero

Le strutture di controllo: CONDIZIONE

- Le istruzioni da eseguire sono determinate dalla valutazione di una data condizione

- Esempio:
  - SE il numero è libero
  - ALLORA
    - attendi la risposta
    - conduci la conversazione
    - deponi il ricevitore
  - ALTRIMENTI
    - deponi il ricevitore

Algoritmo (definizione informale)

- Sequenza **finita** di istruzioni,
- **comprensibili** da un **esecutore** (si può trattare di uno strumento automatico),
- che descrive come **realizzare un compito** (come risolvere un "problema").

- Alcuni esempi
  - Istruzioni di montaggio di un elettrodomestico
  - Uso di un terminale Bancomat
  - Calcolo del massimo comune divisore di numeri naturali

Esecutori e linguaggi

- Un esecutore è definito in base a tre elementi:
  - l'insieme delle **operazioni** che è capace di compiere;
  - l'insieme delle **istruzioni** che capisce (**sintassi**);
  - quali **operazioni** associa ad ogni **istruzione** che riconosce (**semantica**).
- Il calcolatore "capisce" le istruzioni che fanno parte del **linguaggio macchina**
  - istruzioni primitive semplici (e.g. max 2 operandi)
  - attenzione all'efficienza (costi, complessità, velocità)
  - difficile e noioso da utilizzare per un programmatore

Soluzione effettiva per l'esecutore

- Se il problema è "semplice" per l'esecutore…
  - L'esecutore lo svolge direttamente
- Altrimenti…
  - Il descrittore deve scomporre il problema in sottoproblemi finchè l'algoritmo non è espresso esclusivamente tramite operazioni elementari
- La soluzione si dice *effettiva* se l'esecutore è in grado di:
  - interpretarla
  - compiere le azioni (in un tempo finito!)

Proprietà di un'azione elementare

- **Finitezza**
  - l'azione deve concludersi in un tempo finito
- **Osservabilità**
  - l'azione deve avere un effetto osservabile, cioè deve produrre qualcosa
- **Riproducibilità**
  - a partire dallo stesso stato iniziale, la stessa azione deve produrre sempre lo stesso risultato

Dal problema alla soluzione automatica

- **Specifiche dei requisiti:**

  descrizione precisa e **corretta** dei requisiti (*verificabilità*) ---> cosa?
- **Progetto:** procedimento con cui si individua la soluzione ---> come?
- **Soluzione:** algoritmo

Proprietà degli algoritmi

- **Correttezza**
  - L'algoritmo perviene alla soluzione del compito cui è preposto, senza difettare di alcun passo fondamentale
- **Efficienza**
  - L'algoritmo perviene alla soluzione del problema usando la minima quantità di risorse fisiche
    - tempo di esecuzione, memoria, ...

Proprietà di un algoritmo

- **Univocità**
  - Non deve esistere alcun grado di libertà da parte del processore nell'esecuzione di ogni azione
- **Effettività**
  - Le operazioni prescritte dall'algoritmo devono poter essere eseguite in *tempo finito*
- **Ingresso**
  - Un algoritmo è corretto se porta ad un risultato coerente per ogni possibile scelta dei dati in ingresso
- **Uscita**
  - L'algoritmo deve fornire uno o più dati in uscita
- **Terminazione**
  - L'esecuzione di un algoritmo deve terminare in un *numero finito di passi*

Esempio

Data la seguente ricetta, composta da un insieme di azioni elementari, **si verifichi se rispetta le proprietà a cui un algoritmo deve sottostare**:

Biancomangiare

Dosi per 4 persone:
1 litro di latte; 200 gr. di zucchero; 200 gr. di amido; scorza di limone.

Sciogliete l'amido in un pochino di latte. Mescolate lo zucchero al latte rimanente ed unite un po' di scorza di limone grattugiata. Unite l'amido sciolto e mescolate il tutto. Fate cuocere a fuoco basso, rimescolando continuamente finché la crema non si addensa. Versate nelle coppette e lasciate raffreddare in frigorifero per tre ore.

Alcuni concetti

- **Algoritmo**
  descrizione di come si risolve un problema
- **Programma**
  algoritmo scritto in modo che possa essere eseguito da un calcolatore (**linguaggio di programmazione**)
- **Linguaggio macchina**
  linguaggio **effettivamente** "compreso" da un calcolatore, caratterizzato da
  - istruzioni primitive semplici (e.g. max 2 operandi)
  - attenzione all'efficienza (costi, complessità, velocità)
  - difficile e noioso da utilizzare per un programmatore
- Compito dell'informatico è **produrre algoritmi** (cioè capire la sequenza di passi che portano alla soluzione di un problema) e **codificarli in programmi** (cioè renderli comprensibili al calcolatore)

Il problema e la soluzione

- Prima di affrontare la soluzione occorre capire esattamente il problema
- Non serve saper risolvere il problema sbagliato
  - In questo corso supporremo che il problema sia ben noto è chiaramente formulato e ci concentreremo sul **formulare una soluzione**
  - **Spesso in pratica è più difficile capire esattamente la natura del problema che non trovarne una soluzione!** (**Requirements engineering**)

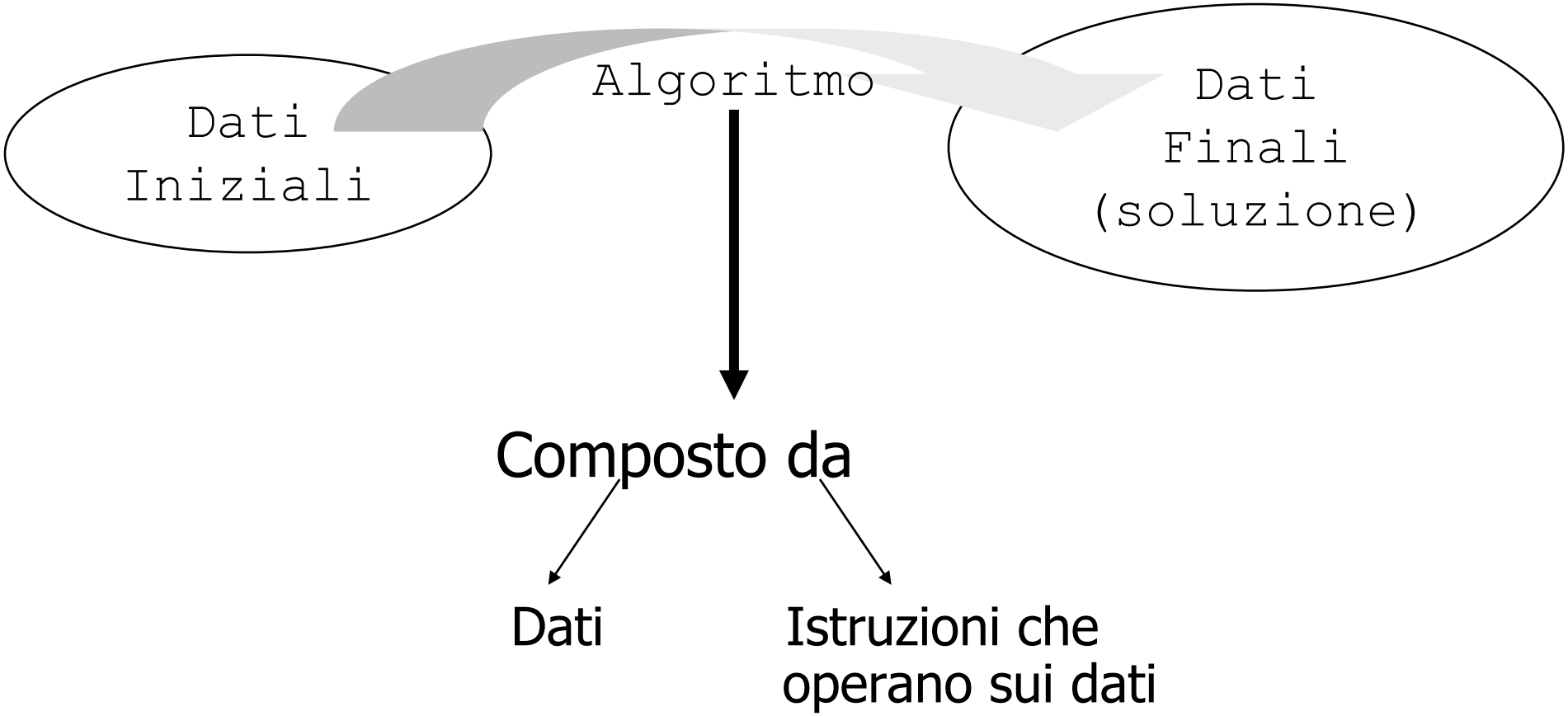
Algoritmi

Formalizzazione

Codifica degli algoritmi

- Algoritmo formulato per essere comunicato tra esseri umani
  - sintetico e intuitivo
  - codificato in linguaggi informali o semi-formali (linguaggio naturale, diagrammi di flusso, ...)
- Algoritmo formulato per essere eseguito automaticamente
  - preciso ed eseguibile
  - codificato in linguaggi comprensibili dagli esecutori automatici (linguaggio macchina o linguaggio di programmazione di alto livello)

Algoritmo = dati + istruzioni



Dati e istruzioni

➢ Tipi di dati

- Numeri naturali o interi o reali (1, -2, 0.34)
- Caratteri alfanumerici (A, B, ..)
- Dati logici o booleani (Vero, Falso)
- Array o vettore di n elementi ({1,2,3})

➢ Istruzioni

- Operazioni di Input/Output (es. *leggi*, *scrivi*)
- Operazioni Aritmetico-logiche (es. *max = A + B*)
- Strutture di controllo (es. *SE*, *RIPETI*)

Criteri di classificazione dei dati

- **Visibilità da parte dell'utente**
  - visibile (di ingresso o uscita)
  - trasparente (dati temporanei di supporto)
- **Variabilità nel tempo**
  - costanti
  - variabili (acquisizione dall'esterno o assegnazione)
- **Struttura**
  - elementari (interi, alfanumerici, booleani, …)
  - strutturati (array, matrici, …)

Operazioni elementari

- Operazioni aritmetiche e assegnamenti di valori a singole variabili
  - Es. C ← (A + B)
- Condizioni sul valore di singole variabili
  - se (A > B) allora … altrimenti …
- Lettura e scrittura di variabili
  - "Leggi A" oppure "Stampa B"

Rappresentazione degli Algoritmi

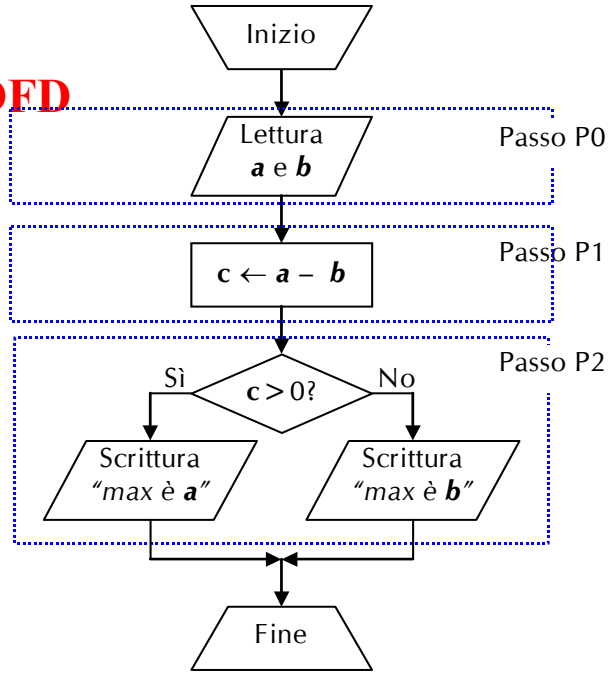
- 1. Linguaggio naturale
- 2. Diagramma a blocchi
- 3. Pseudo codice
- 4. Linguaggio di programmazione

Rappresentare gli algoritmi

Linguaggio naturale

- Sollevare il ricevitore
- Attendere il segnale di linea libera
- Comporre il numero
- …

DFD



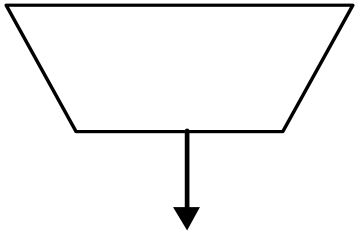
Pseudo codice

```
Input A,B
Tot ← 0
While A!=0 Do
        Tot ← Tot + B
        A ← A – 1
Output Tot
```

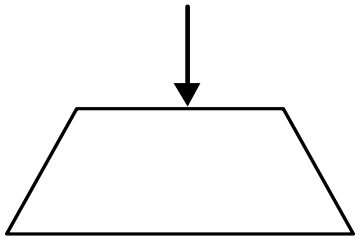
Ling programmazione

```
#include <stdo.h>
Int main (void){
        puts("ciao mondo!");
        return Exit_success;
}
```

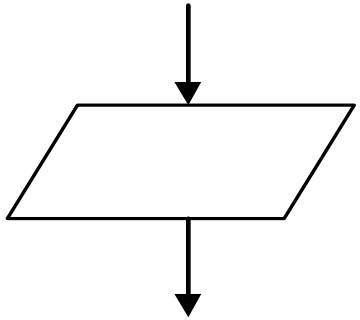
Diagrammi di flusso



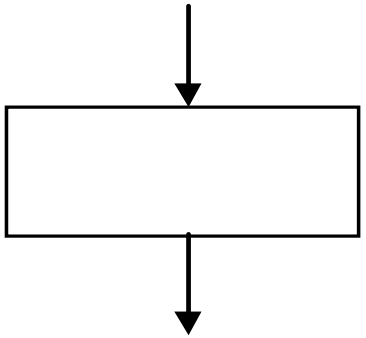
Inizio



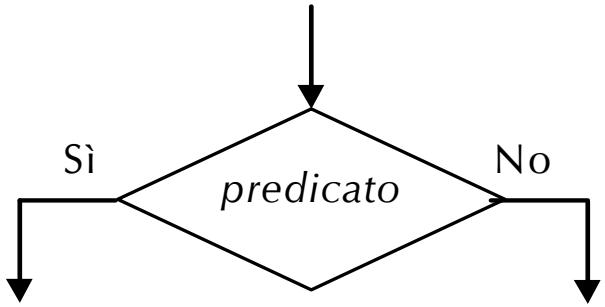
Fine



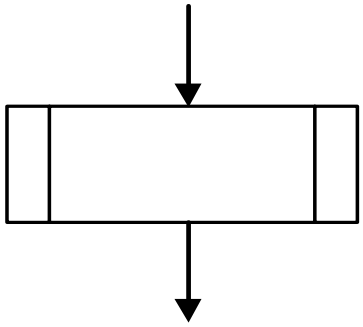
Operazioni
di ingresso/uscita



Elaborazione



Selezione a due vie



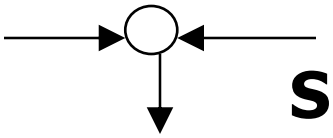
Sottoprogramma

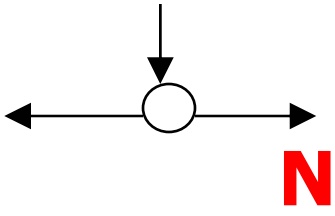
Diagrammi di flusso

I blocchi sono collegati fra loro da archi orientati.

- L'arco identifica la sequenza delle operazioni

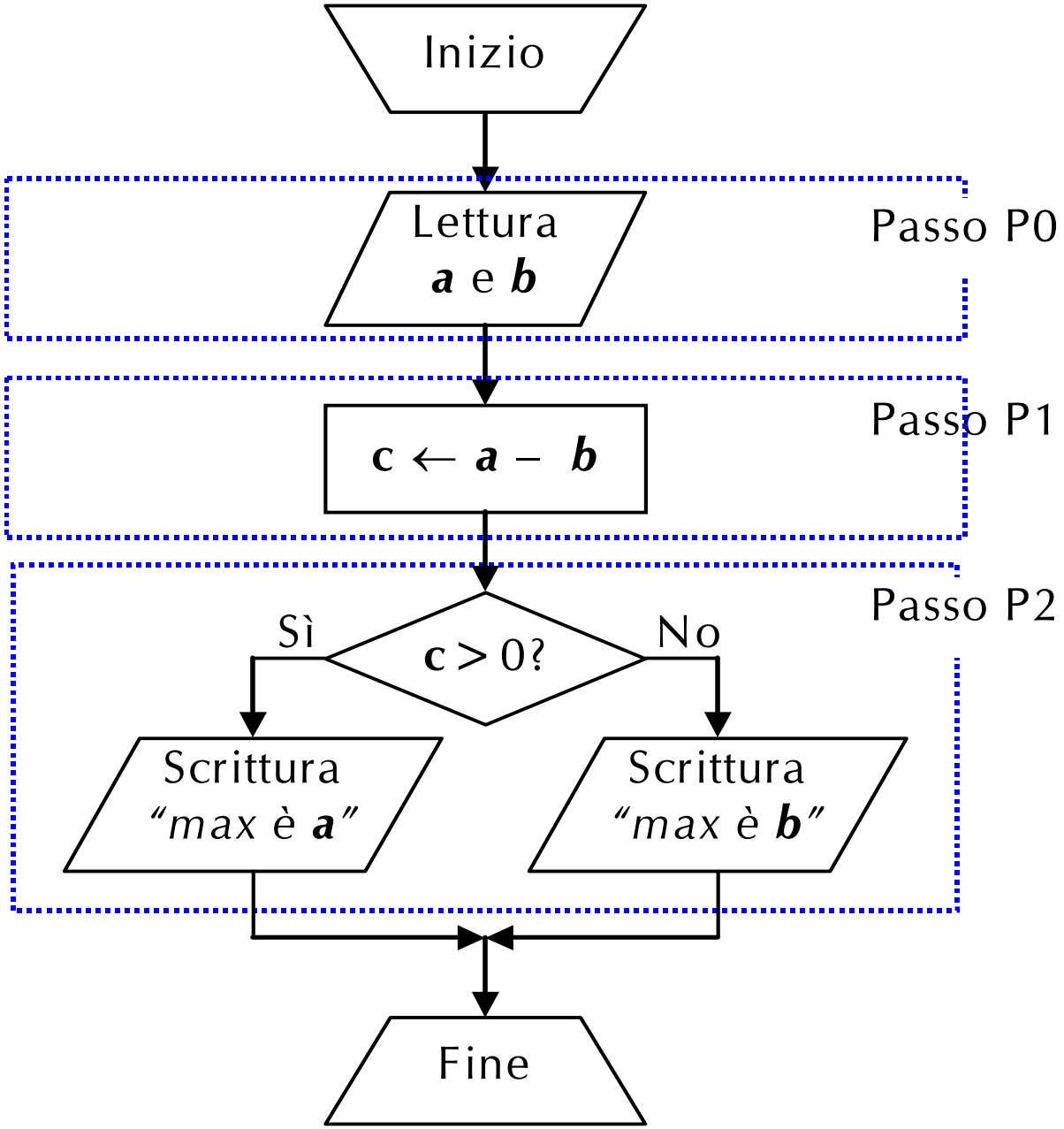
- La freccia identifica il flusso della esecuzione





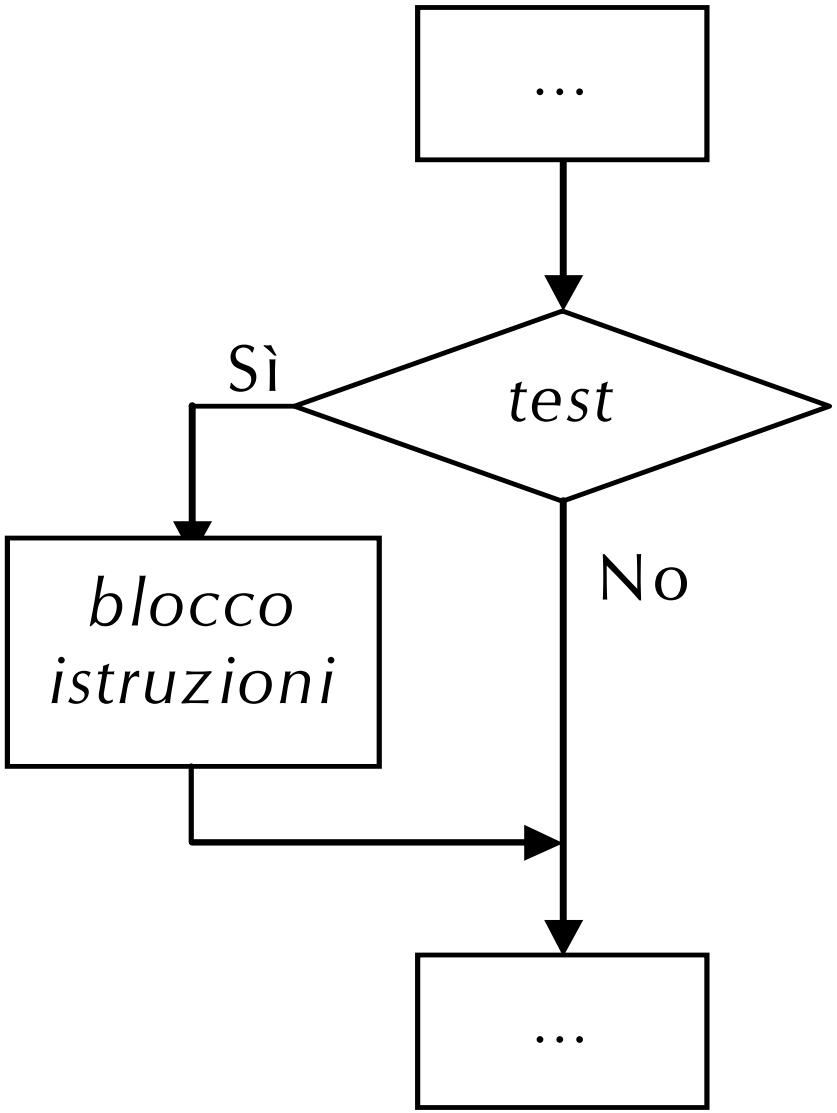
Esempio

- **Esempio**: dati in ingresso due numeri A e B, si calcoli e stampi il maggiore.

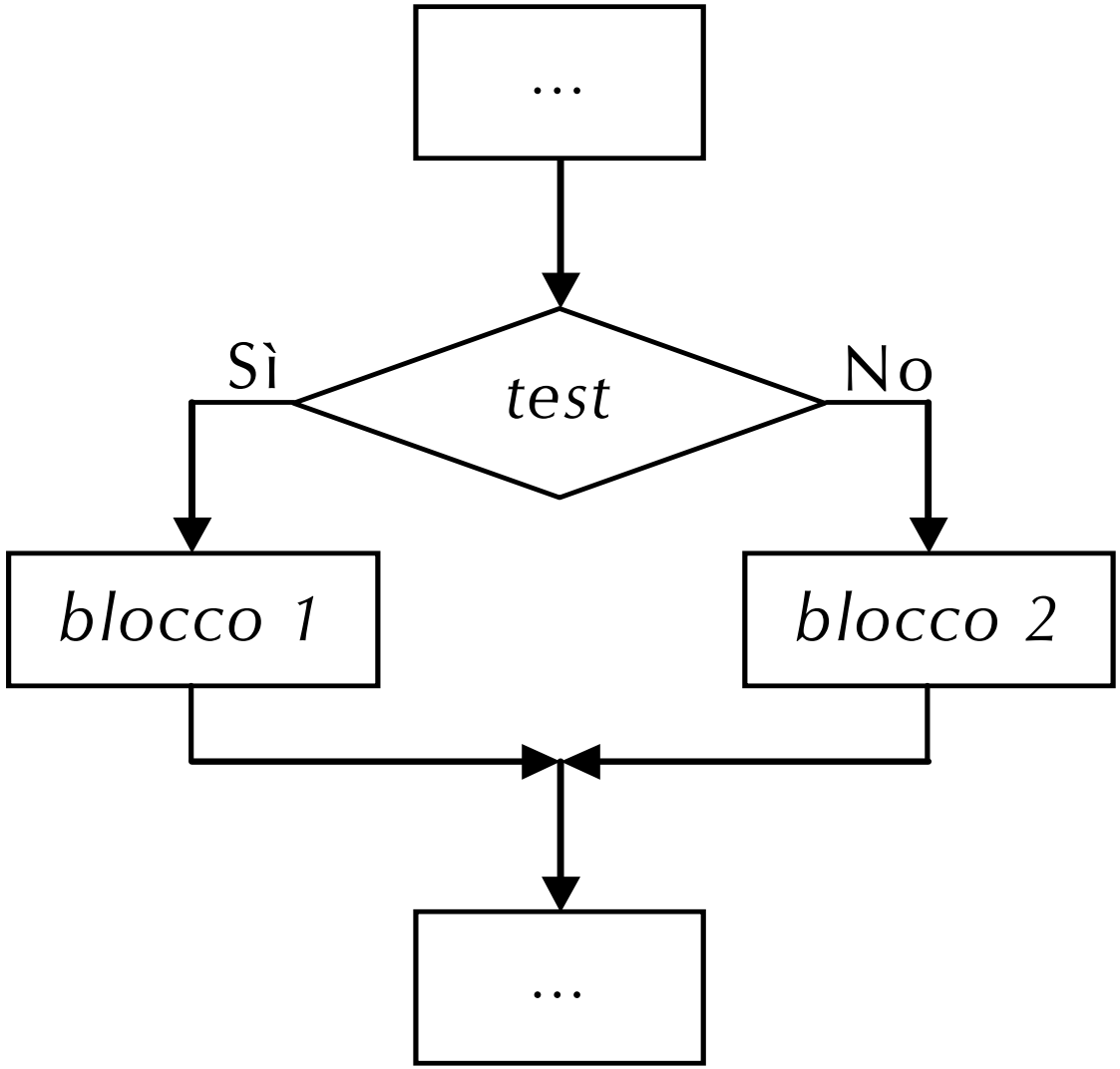


Le strutture di controllo

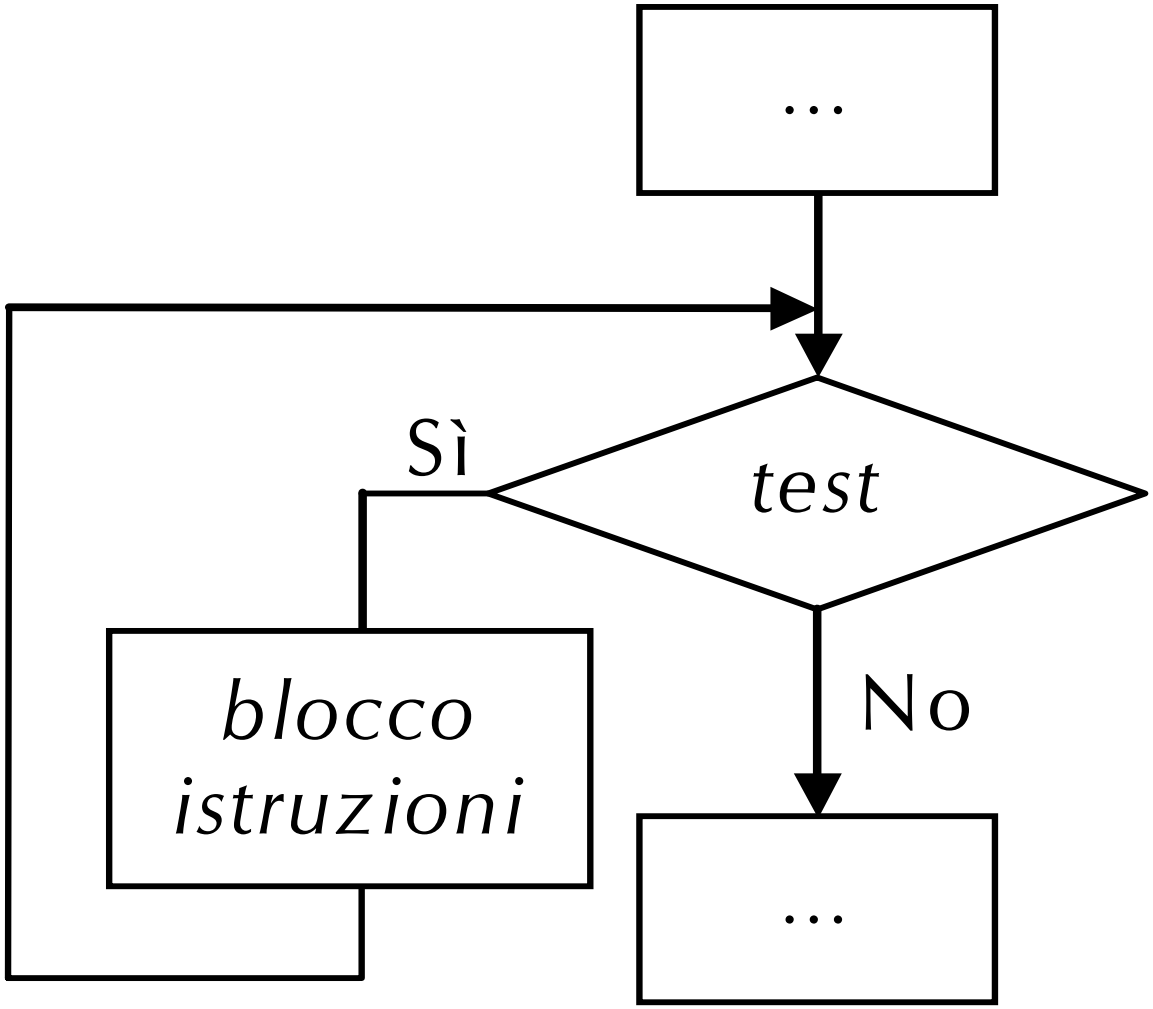
Selezione semplice



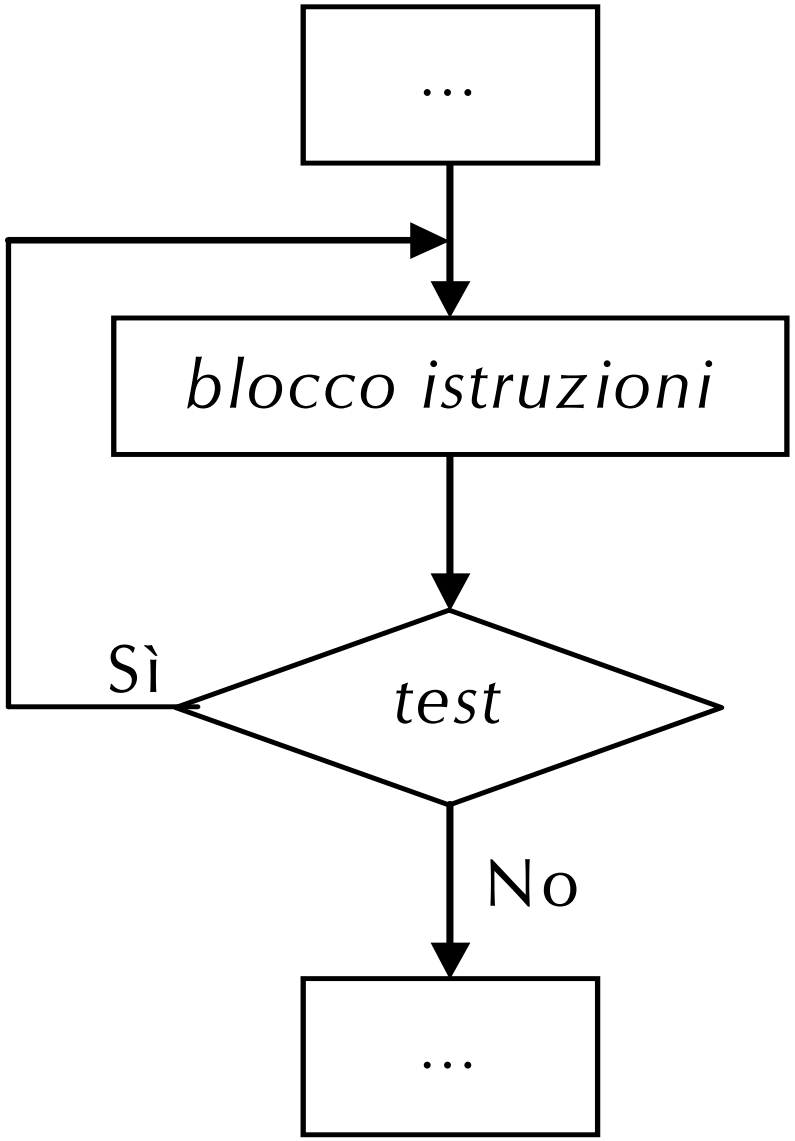
Selezione a due vie



Ciclo a condizione iniziale



Ciclo a condizione finale



Alcuni esempi di algoritmo

Esempio: il prodotto di due interi positivi

- Leggi W
- Leggi Y
- Somma W a se stesso Y volte
- Scrivi risultato

Prodotto di due interi positivi

1 **Leggi W**
2 **Leggi Y**
3 **SP = 0**
4 **NS = Y**
5 **SP = SP + W**
6 **NS = NS - 1**
7 **NS = 0?**
Se NO: torna a 5
8 **Z = SP**
9 **Scrivi Z**

- Procedimento **sequenziale**
- **Non ambiguo**
- Formulazione **generale**
- Prevede **tutti i casi**
(che succede se Y < 0?)

Strumenti per la descrizione degli algoritmi

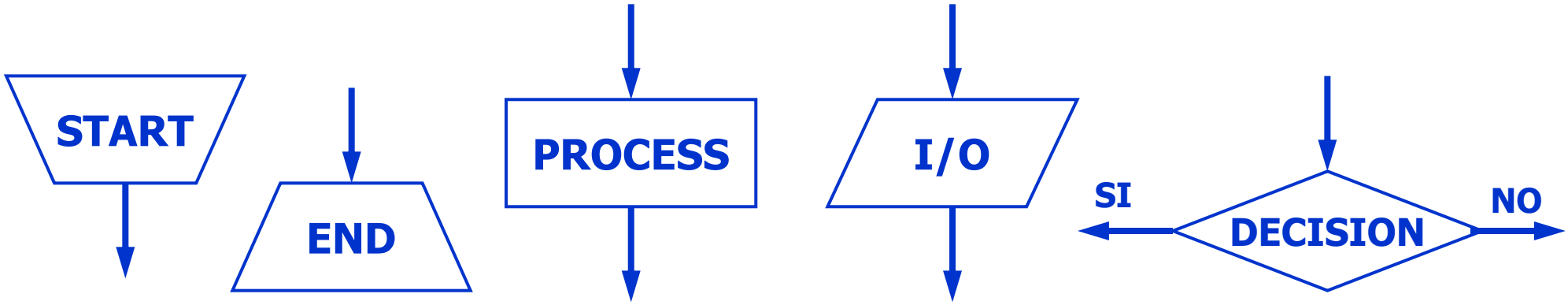
- Semi-formali (specifiche iniziali, ancora intelligibili solo all'essere umano)

- Formali (programmi da eseguire):

  linguaggi di programmazione

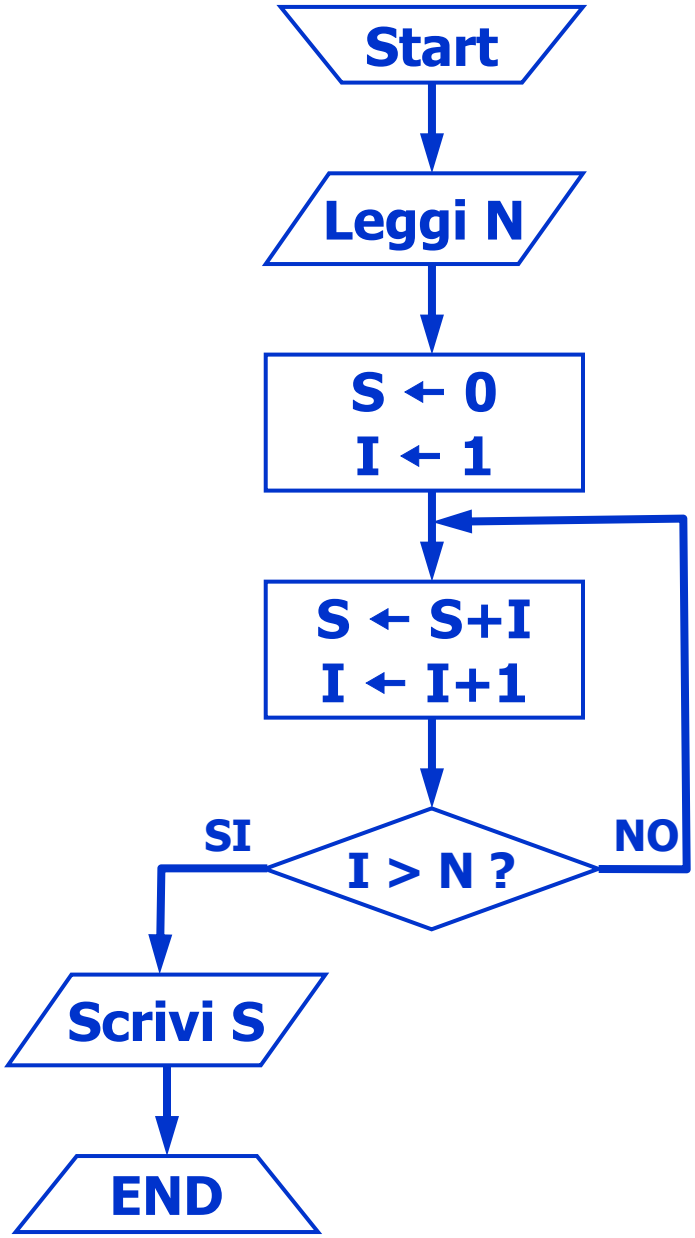
Linguaggi semi-formali

- **pseudo-codice:**
  - **IF** A>0 **THEN** A=A+1 **ELSE** A=0
- **diagrammi di flusso**
  (flow chart, schemi a blocchi):



Esempio

Calcolare e poi stampare la somma dei primi N numeri naturali

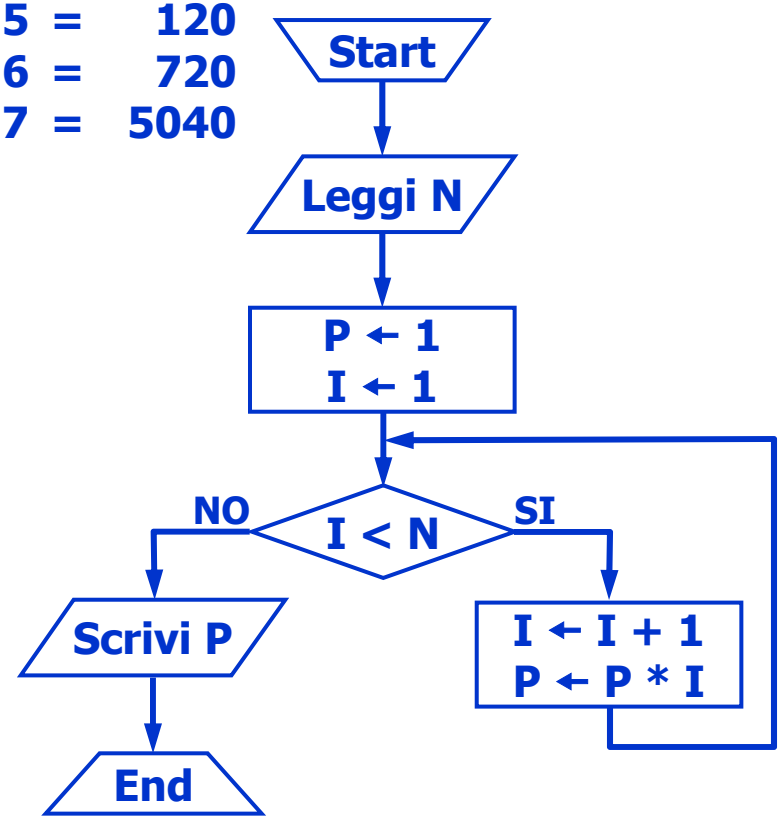


Esercizio

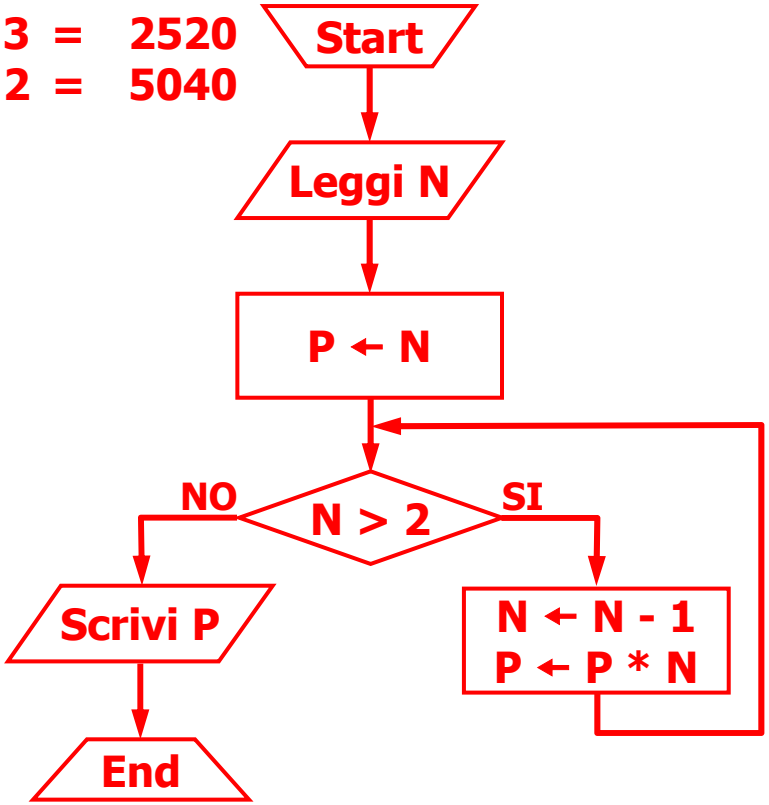
- L'esecutore deve leggere un intero N e restituire il fattoriale di questo numero, cioè il valore ottenuto da N x (N-1) x (N-2) x … x 1.
- Scrivere l'algoritmo immaginando che i dati di ingresso siano sempre corretti (cioè sempre maggiori di zero).
- Modificare l'algoritmo in modo da considerare anche la possibilità che siano inseriti valori inferiori a 1.

Diverse alternative (e.g. 7!)

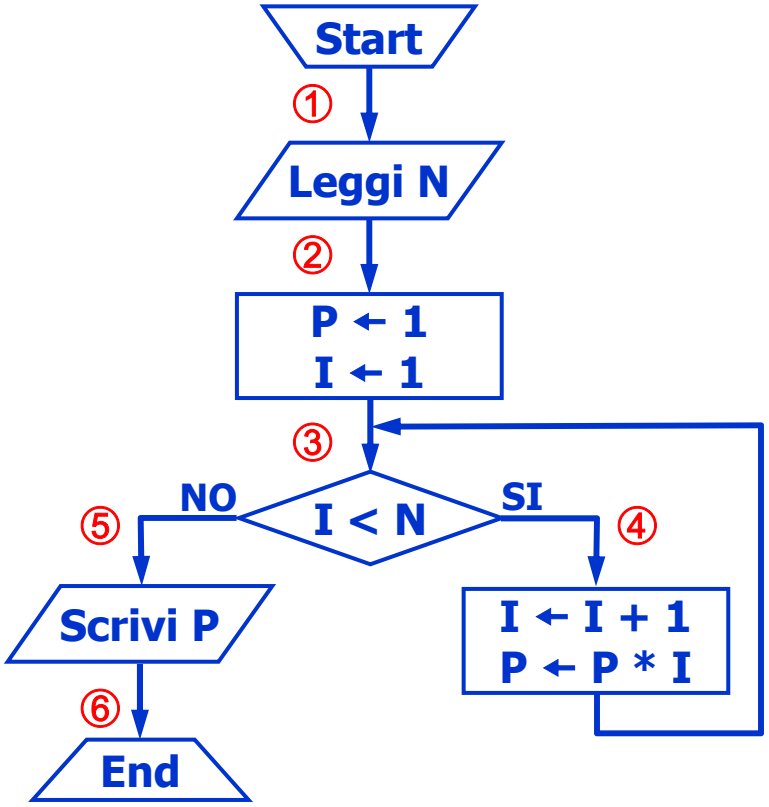
```
  1 *  2 =    2
  2 *  3 =    6
  6 *  4 =   24
 24 *  5 =  120
120 *  6 =  720
720 *  7 = 5040
```




```
  7 *  6 =   42
 42 *  5 =  210
210 *  4 =  840
840 *  3 = 2520
2520 *  2 = 5040
```



"Tracciato" dell'esecuzione

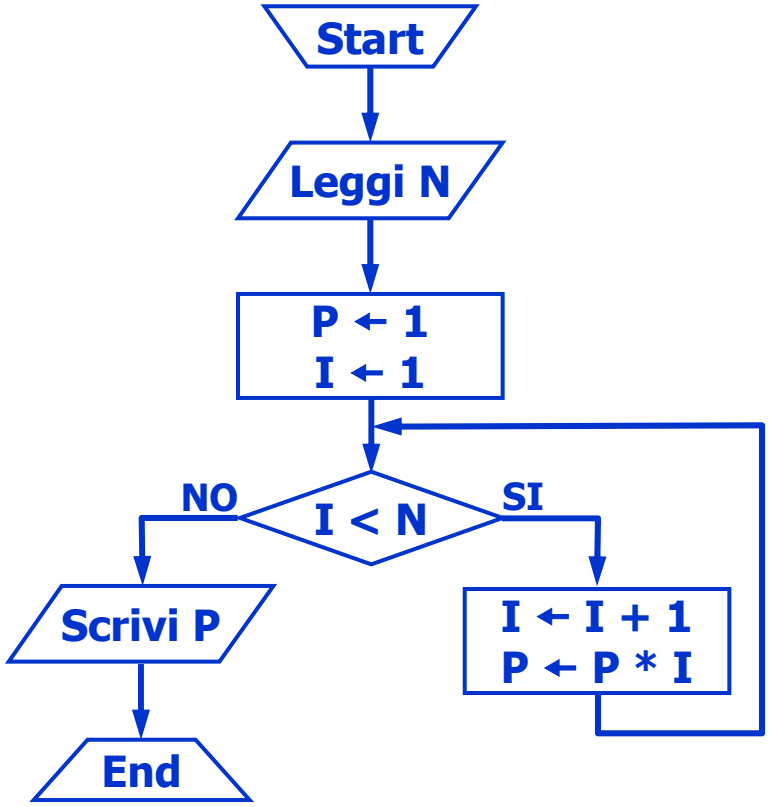


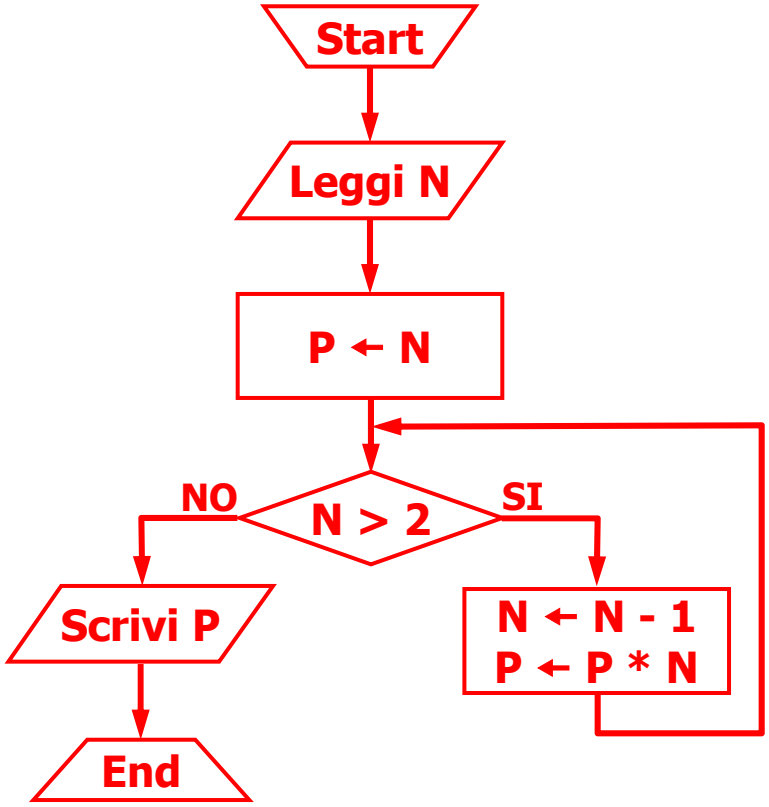
Ipotizziamo di calcolare 4!

T	pos	N	I	P	note
t_1	①	??	??	??	
t_2	②	4	??	??	
t_3	③	4	1	1	I < N
t_4	④	4	1	1	
t_5	③	4	2	2	I < N
t_6	④	4	2	2	
t_7	③	4	3	6	I < N
t_8	④	4	3	6	
t_9	③	4	4	24	I = N
t_{10}	⑤	4	4	24	
t_{11}	⑥	4	4	24	

Attenzione: il valore iniziale non è 0!!
Può non esserci oppure esserci ma non essere noto.

Le alternative sono "diverse"?



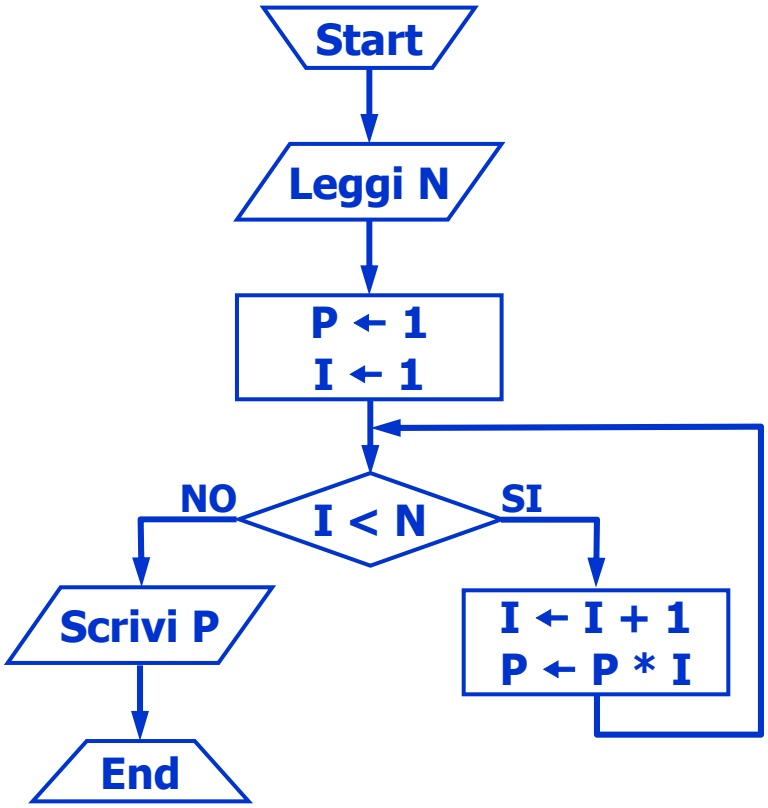


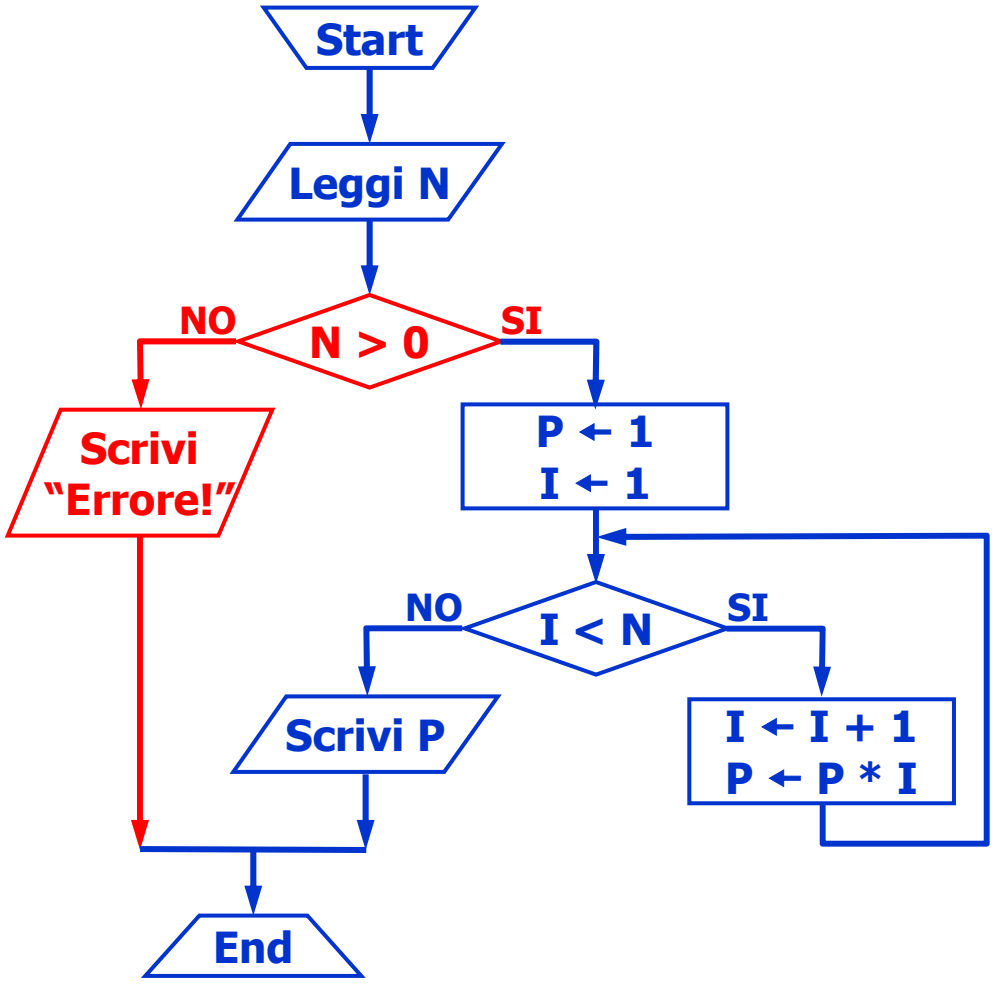
Cosa succede se il dato in ingresso non rispetta le specifiche (N > 0)?
Per esempio, che risultato restituisce l'esecutore per N = 0 ? e per N = -4 ?

N = 0 ⇒ P = 1
N = -4 ⇒ P = 1

N = 0 ⇒ P = 0
N = -4 ⇒ P = -4

Come gestire le "eccezioni"





Algoritmo per il caso "normale". Come lo modifico per gestire anche i casi che non erano stati previsti?

Esercizio

- L'esecutore deve leggere un numero N indicato da un utente esterno e deve poi calcolare ed infine stampare la somma di tutti i numeri compresi tra 0 e N.
- Si presti attenzione al fatto che il numero indicato dall'utente può essere positivo, negativo e, al limite, anche uguale a zero.
- Per esempio, se il numero indicato dall'utente esterno fosse 5, il risultato generato dall'esecutore dovrebbe essere 15 (che corrisponde a 0+1+2+3+4+5); se fosse invece -7, il risultato dovrebbe essere -28 (che corrisponde a (-1)+(-2)+(-3)+(-4)+(-5)+(-6)+(-7)).

Esercizio

- L'esecutore deve leggere una sequenza di numeri naturali (i.e. interi positivi strettamente maggiori di zero) e calcolarne (per poi stamparlo) il minimo.
- La sequenza si interrompe non appena viene introdotto un numero negativo oppure uguale a zero.
- Per esempio, data la sequenza 5, 1, 2, 3, 4, -5, il risultato dovrebbe essere: "**Il valore minimo è 1**".

Esercizio

- L'esecutore deve leggere una sequenza di numeri naturali (i.e. interi positivi strettamente maggiori di zero) e calcolarne (per poi stamparli) il massimo, il minimo e la media di questa sequenza.
- La sequenza si interrompe non appena viene introdotto un numero negativo oppure uguale a zero.
- Per esempio, data la sequenza 5, 1, 2, 3, 4, -5, il risultato dovrebbe essere:
  "Il massimo è 5, il minimo è 1, la media è 3"

Esercizio

- L'esecutore deve leggere un numero N indicato da un utente esterno, questo numero indica la lunghezza della sequenza di numeri che vengono poi inseriti dallo stesso utente (per esempio, se un utente vuole inserire 20 numeri, prima indica il numero 20, poi specifica i 20 numeri che compongono la sua sequenza, in questo modo egli introdurrà 21 numeri).
- Di questi numeri, l'esecutore deve calcolare e poi stampare il massimo, il minimo e la media.
- Per esempio, data la sequenza 5, 1, 2, 3, 4, -5, il risultato dovrebbe essere:
  "**massimo = 4, minimo = -5, media = 1**"

Esercizio

- L'esecutore deve leggere un intero che rappresenta l'anno, e stampare
  - la scritta “true” se l’anno in esame è bisestile
  - la scritta “false” se l’anno non è bisestile.
- Si ricordi che un anno è bisestile se:
  - è divisibile per 4 ma non per 100
  - oppure se è divisibile per 400.
- Per esempio 1900 e 2100 non sono anni bisestili, mentre 1996 e 2000 lo sono.

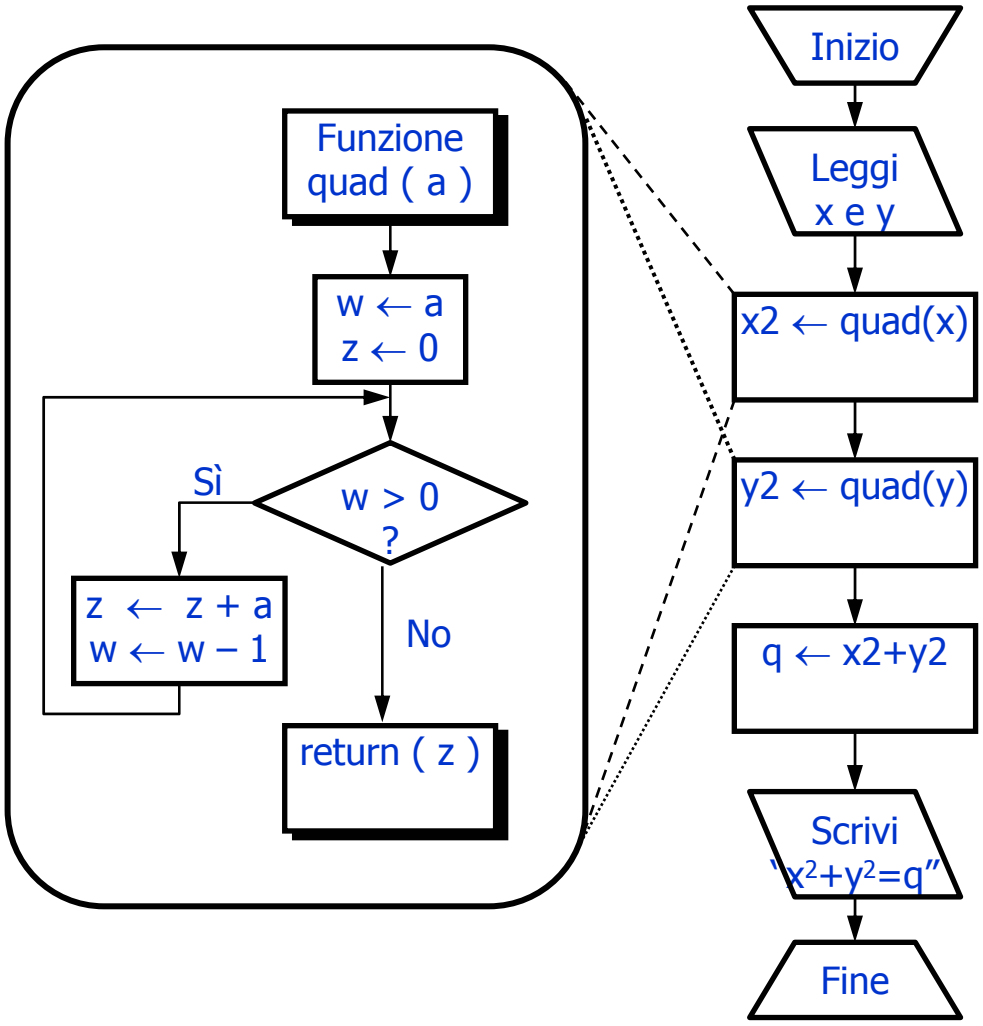
Esercizio

- L'esecutore deve leggere tre numeri interi che rappresentano una data in termini di giorno, mese e anno, e deve stampare il numero di giorni trascorsi dall'inizio dell'anno.
- Scrivere l'algoritmo immaginando che i dati di ingresso siano sempre corretti.

I dati

- Ogni variabile è caratterizzata dal suo *tipo*.
  - Tipi predefiniti: numeri, caratteri, booleani, …
  - Altri tipi: stringhe, date, …
- Variabili strutturate:
  - Vettori (o array)
  - Record

Sottoprogrammi




```
int quad (int a)
   /* restituisce a2 /
{  int w, z;
   w = a;  z = 0;
   while (w > 0)
   {  z = z + a;
      w = w – 1;
   }
   return (z);
}

main() /* q = x2 + y2 */
{ int x,y,x2,y2,q;
  scanf("%d %d",&x,&y);
  x2 = quad(x);
  y2 = quad(y);
  q = x2+y2;
  printf("%d", q);
}
```

Uso degli array

```
Dati
  n=100 intero
  f[ ]  vettore di
interi
  w, z  interi positivi
Risoluzione
  …
  w ← 1
  z ← 0
  finché (w ≤ n) ripeti
    z ← z + f[w]
    w ← w + 1
  fine ciclo
  scrivi z
  …
```

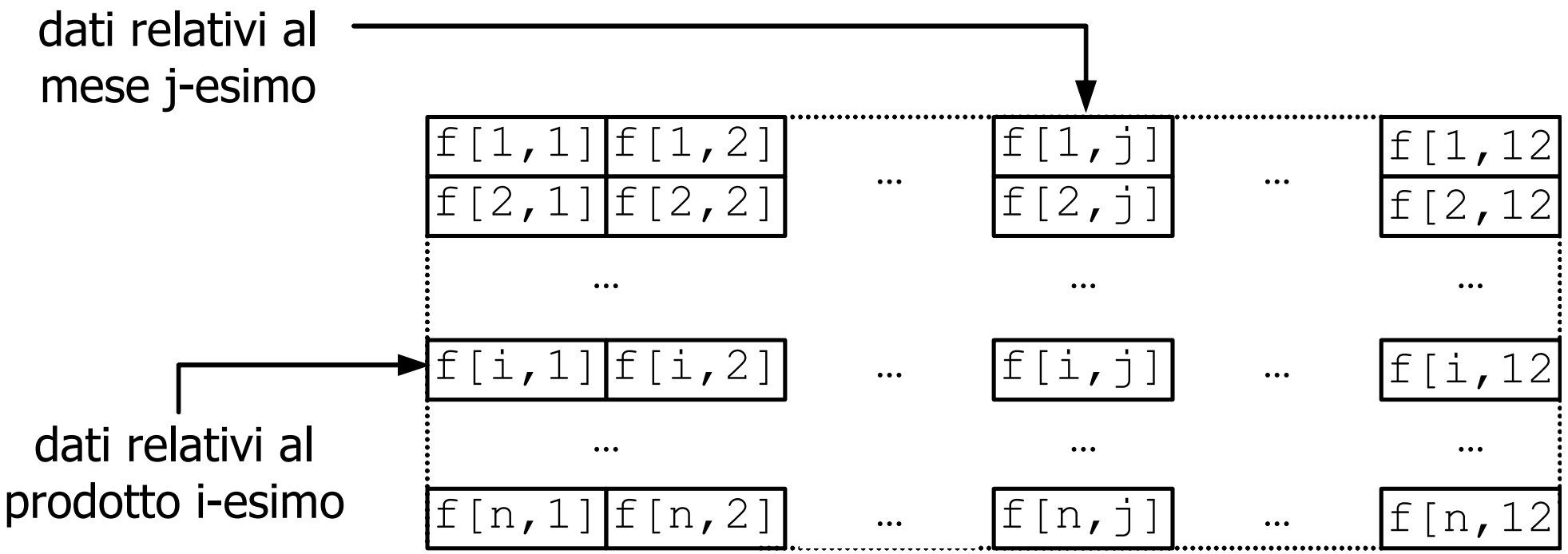
```
main() /* C */
{
  int f[100];
  int w, z;

  …
  w = 0;
  z = 0;
  while (w ≤ 99)
  { z = z + f[w];
    w = w + 1;
  }
  printf("%d", z);
  …
}
```

```
'Basic

dim f(100) as integer
dim w as integer
dim z as integer
…
w = 1
z = 0
while w <= 100
  z = z + f(w)
  w = w + 1
wend
print z
…
```

Un esempio di matrice



Esempio di variabili strutturate

```
struct prodotto /* C */
{ char nome[DIMNOME];
  int  fatturato;
};
main()  /* operazioni su prodotti */
{ struct prodotto p;
  …
  p.fatturato = … ;/* assegna un valore */
         /* al fatturato del prodotto p */
}
```

```
type prodotto ‘Basic
  nome as string * DIMNOME
  fatturato as integer
end type

dim p as prodotto
…
p.fatturato = …
```

Array di strutture

```
struct prodotto /* C */
{ char nome[DIMNOME];
  int  fatturato;
};
main()  /* operazioni su prodotti */
{ struct prodotto p[100];
  …
  p[5].fatturato = … ;
    /* assegna un valore al fatturato */
    /* del prodotto p di indice 5 */
}
```

```
type prodotto ‘Basic
  nome as string * DIMNOME
  fatturato as integer
end type

dim p(100) as prodotto
…
p(5).fatturato = …
```

Tabella

Rappresentazione sotto forma di tabella dell'array di record definito e utilizzato nei frammenti di codice riportato nella slide precedente.
Il numero riportato alla sinistra di ogni riga rappresenta l'indice dell'array (in questo caso a partire da 0 e quindi in accordo alla sintassi C) ed è utilizzato per accedere al contenuto del corrispondente record.

	PRODOTTI	
	Nome	**Fatturato**
0	HT5231	129 000
1	HT5441	105 000
2	NS221	144 000
3	NS321	123 000
4	NS222	133 000
5	NS322	136 000
6	HT5321	139 000
7	HT5442	136 000
…	… … …	… … …
98	RX521	183 000
99	RX522	175 000

Il modello relazionale

STUDENTI

Matr	Cognome	Nome	Nato_il	Nato_a
3571	Banfi	Alessandro	19/02/1982	Milano
999	Bosio	Umberto	27/01/1983	Aosta
2805	Castelnuovo	Andrea	06/05/1982	Torino
3719	Colpi	Marco	15/01/1983	Genova
773	Izzo	Stefania	08/10/1982	Firenze
3672	Librandi	Silvia	12/03/1983	Bologna
1539	Longoni	Mauro	05/02/1983	Venezia
3500	Matta	Vera	26/04/1982	Roma
1886	Merlo	Andrea	05/05/1983	Trento
1427	Morelli	Riccardo	14/04/1982	Trieste
2608	Ornaghi	Gabriele	09/09/1982	Perugia
3711	Panico	Andrea	29/05/1982	Pescara
1940	Poretti	Stefania	20/02/1982	Ancona
1814	Quaglia	Andrea	13/08/1982	Napoli
1662	Salmoiraghi	Veronica	19/09/1982	Cagliari
2744	Sterlocchi	Elena	29/06/1982	Palermo
3024	Tarantola	Marcello	17/06/1982	Reggio Calabria
3527	Valentini	Samuele	10/07/1982	Bari
3615	Venturi	Anita	28/07/1982	Potenza
681	Zaccaretti	Carolina	23/02/1983	Campobasso

ISCRITTI

Matr	Codice
2805	IG06
3527	BA03
1940	IG10
773	IG11
1539	IG05
1940	IG03
3672	ICT3
681	ICT2
1886	IG05
1940	ICT1
3500	BA08
1886	IG01
3024	BA01
3719	IG10
3672	IG08
773	ICT2
3719	IG06
1814	ICT2
2744	BA09
2744	IG03
2805	IG09

CORSI

Codice	Titolo	Settore	Tipo	Crediti
BA01	Analisi Matematica I	MAT/05	Base	7.5
BA02	Analisi Matematica II	MAT/05	Base	7.5
BA03	Elettromagnetismo	FIS/01	Base	5
BA04	Fond. Meccanica Teorica e Applicata	ING-IND/13	Affine	5
ICT1	Fond. Informatica I (laboratorio)	ING-INF/05	Affine	4
IG01	Elettrotecnica	ING-IND/31	Affine	5
IG02	Fisica Tecnica	ING-IND/10	Affine	5
ICT2	Fond. Informatica II	ING-INF/05	Base	6
IG03	Fond. Automatica	ING-INF/04	Caratt.	5
…	… … …	…	…	…
IG04	Economia Organizzazione Aziendale	ING-IND/35	Caratt.	10
IG05	Gestione Produzione Industriale	ING-IND/17	Caratt.	10
BA09	Ricerca Operativa	MAT/09	Base	5
ICT3	Produzione Assistita Calcolatore	ING-IND/16	Caratt.	5
IG09	Sistemi di Controllo di Gestione	ING-IND/35	Caratt.	5
IG10	Logistica Industriale	ING-IND/17	Caratt.	5
IG11	Gestione Aziendale	ING-IND/35	Caratt.	5
IG12	Gestione della Qualità	ING-IND/17	Caratt.	5

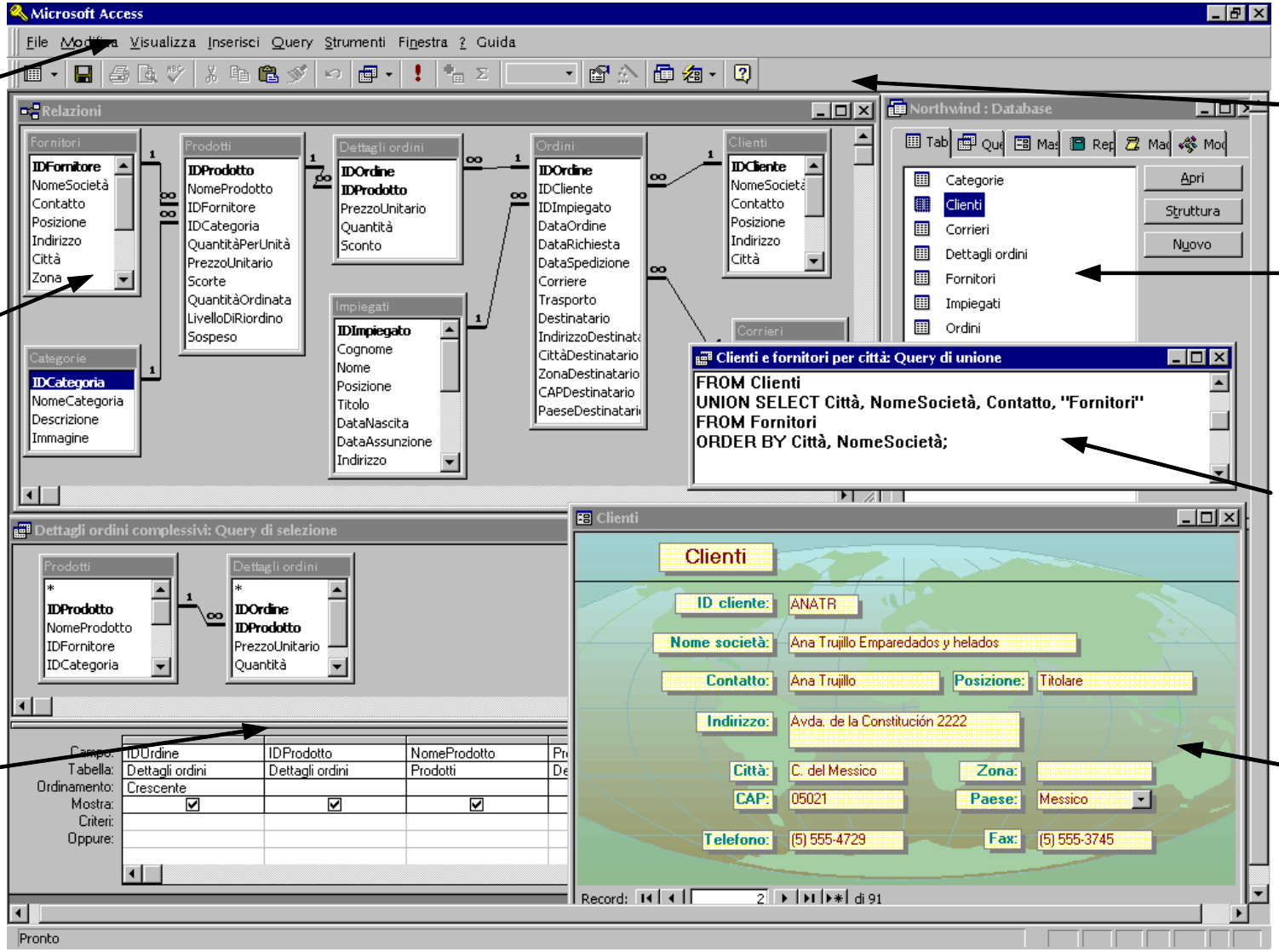
Studenti (Matr, Cognome, Nome, Nato_il, Nato_a);
Corsi (Codice, Titolo, Settore, Tipo, Crediti);
Iscritti (Matr, Codice).

Linguaggio di accesso a un DBMS

Tale linguaggio deve disporre di tre funzionalità di base:

- **Data Definition Language** (**DDL**): definizione della struttura del DB (schemi delle tabelle);
- **Data Manipulation Language** (**DML**): manipolazione dei dati (inserimenti e cancellazioni di record);
- **Query Language** (**QL**): realizzazione di interrogazioni.

Ambiente per lo sviluppo e l'utilizzo di un DB relazionale



Menu comandi

Barra strumenti

Schema delle relazioni

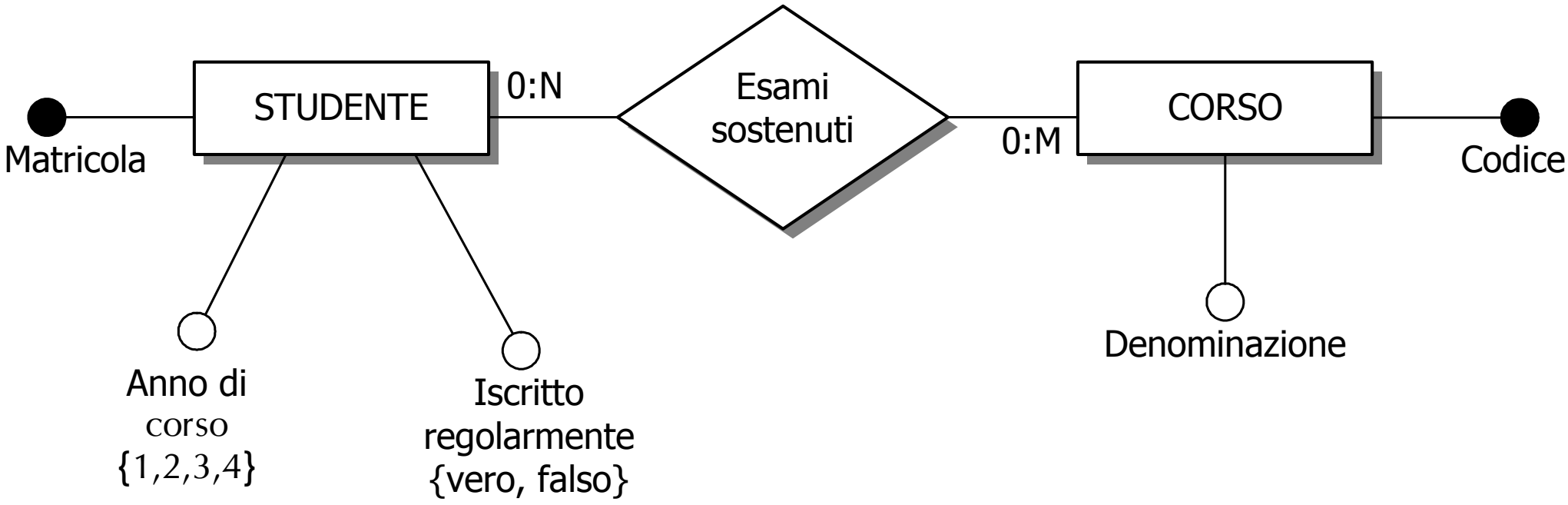
Elementi del DB (Tabelle, Query, ...)

Struttura di una query SQL

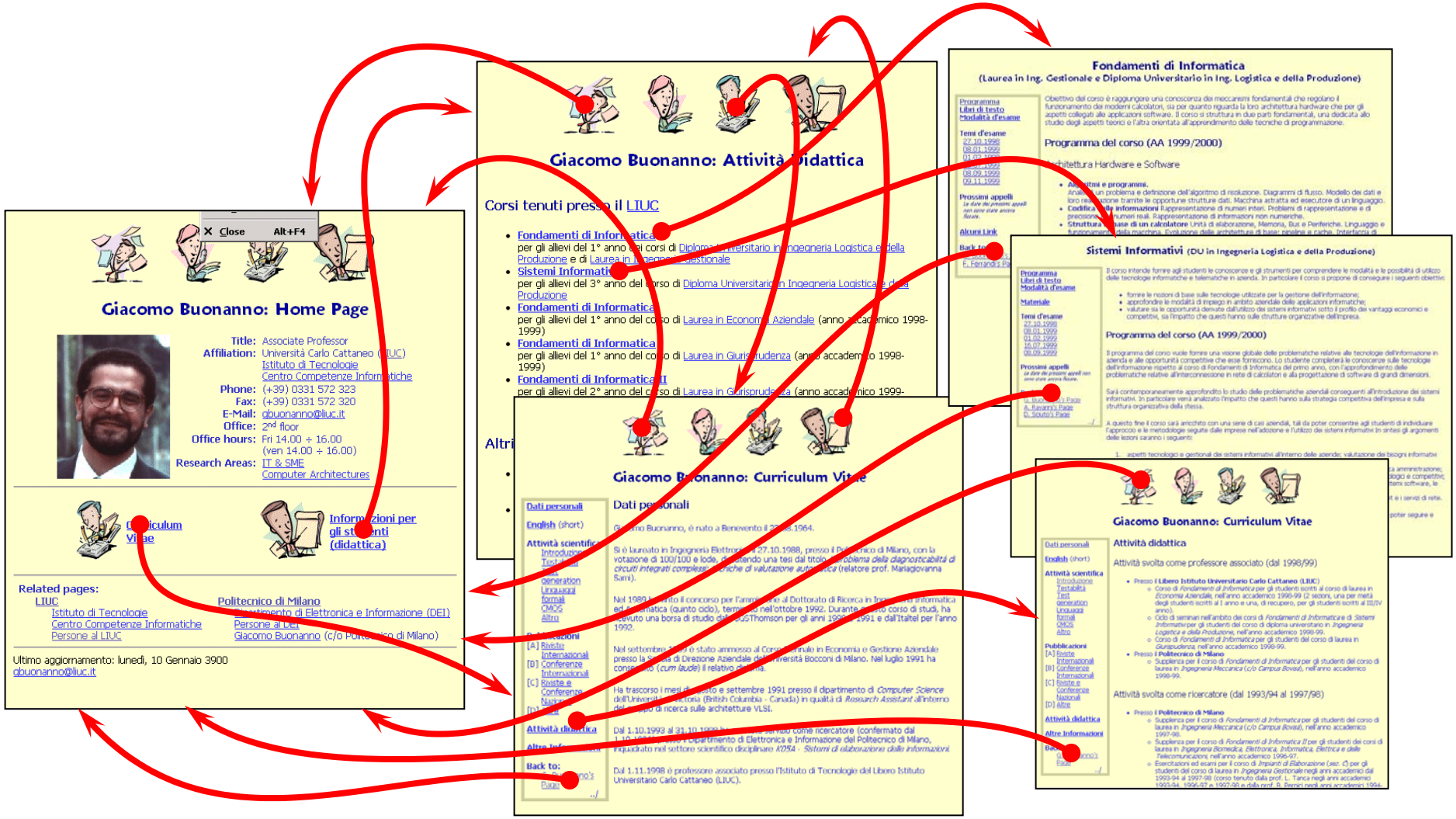
Struttura di una query by example

Scheda di un cliente

Modello concettuale dei dati



Ipertesto


Giacomo Buonanno: Home Page
Title: Associate Professor
Affiliation: Università Carlo Cattaneo (LIUC)
Istituto di Tecnologie
Centro Competenze Informatiche
Phone: (+39) 0331 572 323
Fax: (+39) 0331 572 320
E-Mail: gbuonanno@liuc.it
Office: 2nd floor
Office hours: Fri 14.00 ÷ 16.00
(ven 14.00 ÷ 16.00)
Research Areas: IT & SME
Computer Architectures
Informazioni per gli studenti (didattica)
Related pages:
LIUC
Istituto di Tecnologie
Centro Competenze Informatiche
Persone al LIUC
Politecnico di Milano
Persone al DEI
Giacomo Buonanno
Ultimo aggiornamento: lunedì, 10 Gennaio 3900
gbuonanno@liuc.it
Giacomo Buonanno: Attività didattica
Corsi tenuti presso il LIUC
Fondamenti di Informatica
Sistemi Informativi
Fondamenti di Informatica
Fondamenti di Informatica II
Giacomo Buonanno: Curriculum Vitae
Dati personali
Attività scientifica
Attività didattica
Fondamenti di Informatica
(Laurea in Ing. Gestionale e Diploma Universitario in Ing. Logistica e della Produzione)
Programma del corso (AA 1999/2000)
Sistemi Informativi (DU in Ingegneria Logistica e della Produzione)
Programma del corso (AA 1999/2000)
Giacomo Buonanno: Curriculum Vitae
Attività didattica
Attività svolta come professore associato (dal 1998/99)
Attività svolta come ricercatore (dal 1993/94 al 1997/98)