# DISPENSE DI PROGRAMMAZIONE

# Modulo 1 – Parte IV

- **ESERCITAZIONE SULLE NOTAZIONI PER DESCRIVERE ALGORITMI: Algoritmi elementari (scambio, massimo, etc.) (RICADE NEL MODULO 6)**

- **Decomposizione di problemi attraverso sequenze, selezioni nidificate, relazioni di ricorrenza (iterazione e ricorsione)**

- **La strategia TOP – DOWN per la progettazione di algoritmi (RICADE NEL MODULO 5)**

Ha collaborato alla editazione Neglia Vincenzo

# ALGORITMI DI BASE – SCAMBIO

**Come si fa a scambiare il valore di due variabili ?**
**Si abbiano 2 variabili x,y**
**vogliamo fare in modo che**
**il valore di y sia assunto da x e**
**il valore di x sia assunto da y.**

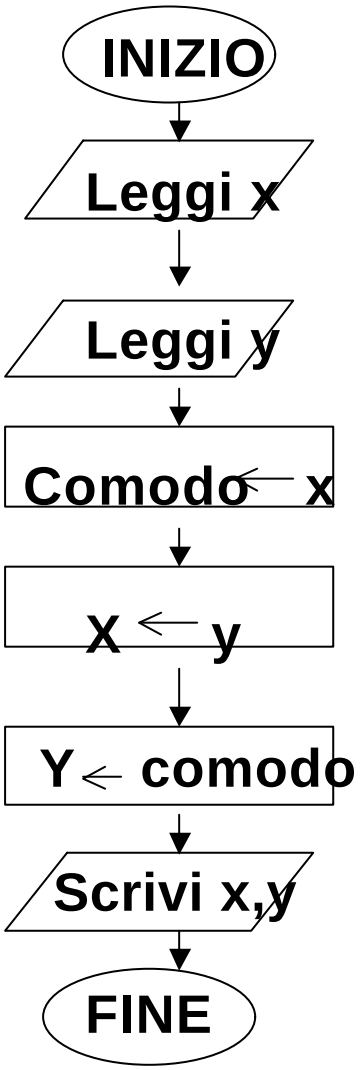
| | | | | |
|---|---|---|---|---|
| **x** | 721 | **y** | 463 | **Prima** |
| **x** | 463 | **y** | 721 | **Dopo** |

# USO DELL'ASSEGNAMENTO

**BISOGNA RICORRERE ALL'USO DI UNA VARIABILE AUSILIARIA PER NON PERDERE IL VALORE DI UNA DELLE DUE VARIABILI DA SCAMBIARE:**

**comodo ⟵ x** salva in comodo il valore di x
**x ⟵ y** assegna ad x il valore di y
**y ⟵ comodo** assegna ad y il valore salvato

# ALGORITMO DI SCAMBIO

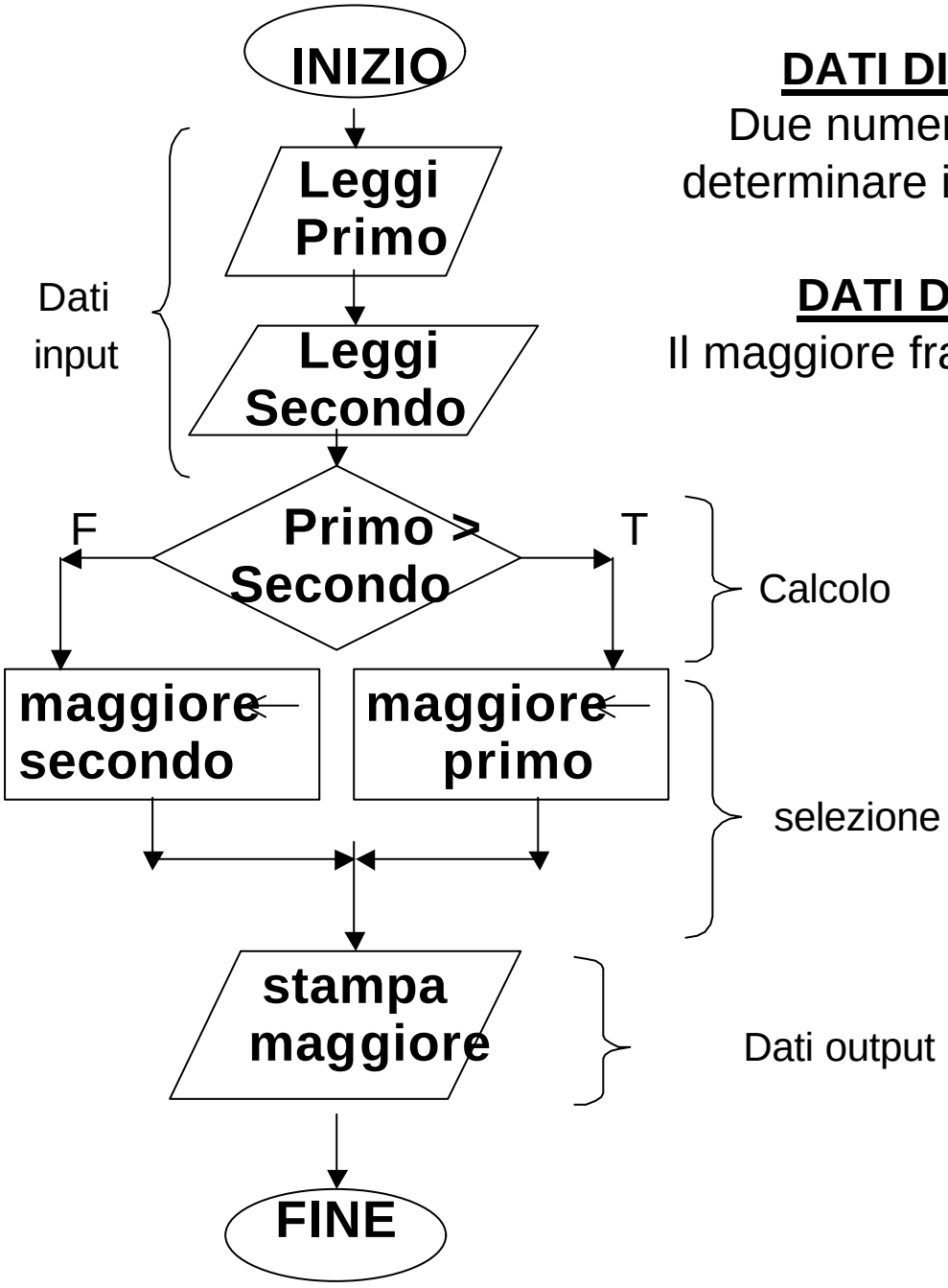



```
BEGIN
    Leggi(x);
    Leggi(y);
    Comodo:=x;
    x:=y;
    y:=comodo;
    scrivi(x,y)
END
```

# APPLICAZIONI DELL'ALGORITMO DI SCAMBIO: Problemi di ORDINAMENTO

# RICERCA DEL MASSIMO

## DETERMINARE IL MAGGIORE FRA DUE NUMERI DATI




**DATI DI INGRESSO**

Due numeri dei quali determinare il maggiore

**DATI DI USCITA**

Il maggiore fra i due numeri

Qual è il dominio di definizione di questo algoritmo?
Come si comporta quando i due numeri sono uguali?
INPUT: all'esterno → calcolatore (memoria)
OUTPUT: dal calcolatore → l'esterno

PSEUCODIFICA

```
BEGIN
    leggi primo;
    leggi secondo;
    IF primo > secondo THEN
                maggiore ← primo
            ELSE  maggiore ← secondo;
    scrivi maggiore
END
```

## VARIANTE

```
IF primo > secondo
            THEN scrivi primo
            ELSE scrivi secondo
```

# DATI X E Y, CALCOLARE IL PRODOTTO X*Y USANDO L'OPERAZIONE DI SOMMA.

$x * y = \underbrace{x + x + x}_{y \text{ volte}}$

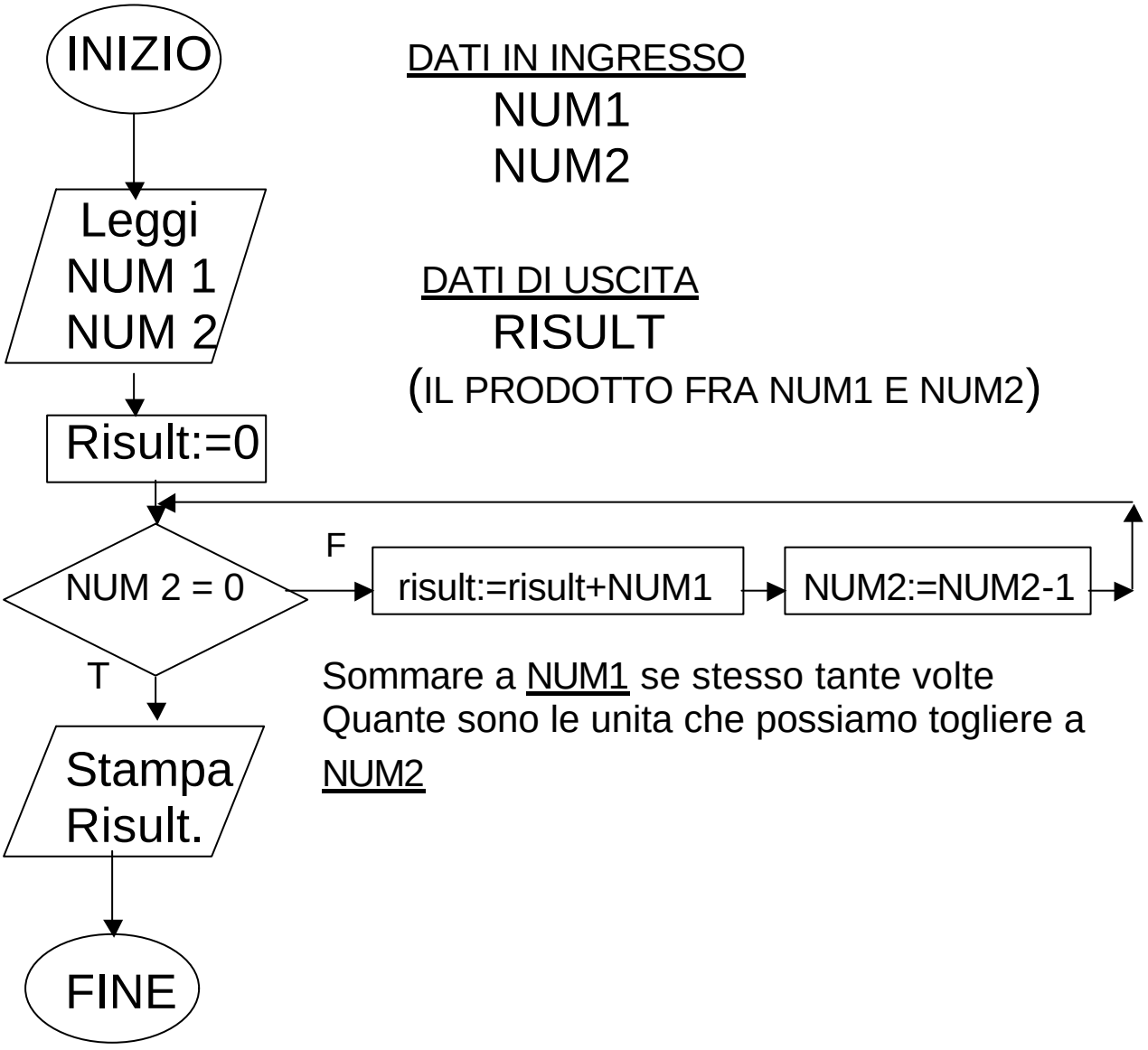
$5 * 3 = 5 + 5 + 5$

**Problemi:**

- **Somma di due numeri interi (tramite incremento unitario)**
- **Sottrazione di due numeri interi ( tramite decremento unitario)**
- **Divisione di due numeri interi (tramite sottrazioni successive)**

# RAPPRESENTARE TRAMITE UN GRAFO DI FLUSSO STRUTTURATO IL PROCESSO DI

## "Moltiplicazione di due numeri interi"




DATI IN INGRESSO
NUM1
NUM2

DATI DI USCITA
RISULT
(IL PRODOTTO FRA NUM1 E NUM2)

Sommare a NUM1 se stesso tante volte
Quante sono le unita che possiamo togliere a NUM2

## Qual è il dominio di definizione di questo algoritmo ?

# LA PROGETTAZIONE DI ALGORITMI

## La strategia TOP-DOWN

**QUANDO SI HA A CHE FARE CON PROBLEMI SEMPLICI**

**Es. Massimo fra 2 numeri**

**L' ALGORITMO È FACILMENTE INDIVIDUABILE.**

**QUANDO SI DEVE RISOLVERE UN PROBLEMA COMPLESSO**

**Es. Gestione c/c bancari**

**…….**

**RIDURRE LA COMPLESSITÀ DEL PROBLEMA**

- **Scomposizione in sotto-problemi più semplici (Individuare struttura e relazioni)**
- **Progettare un algoritmo per ciascun singolo sotto-problema. Se, a sua volta,il sottoproblema è complesso, va riapplicato un passo di scomposizione.**

**La PROGETTAZIONE di un ALGORITMO per un problema COMPLESSO non può essere affrontata in blocco**

**Occorre SCOMPORRE il PROBLEMA ORIGINARIO in SOTTOPROBLEMI Più SEMPLICI**

**Aumenta il numero complessivo di problemi da risolvere, ma ne diminuisce la difficoltà di ciascuno.**

**Affrontare un sottoproblema alla volta.**

**TOP alto livello di descrizione**

**↓**

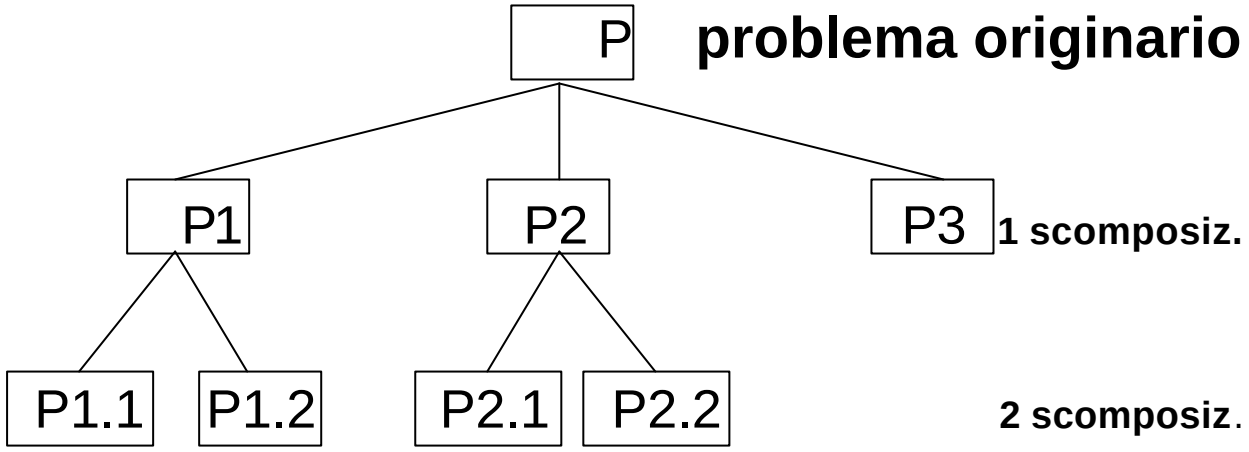
**DOWN basso livello di descrizione**

**IL PROCESSO DI SCOMPOSIZIONE DOVRÀ CONTINUARE FINO A CHE IL LIVELLO DI DETTAGLIO RAGGIUNTO È "SUFFICIENTEMENTE" ESPRESSO IN TERMINI DI AZIONI ELEMENTARI PER L'ESECUTORE.**

**UN PROBLEMA E' DETTO PRIMITIVO QUANDO E' ASSOCIATO AD UNA AZIONE BASICA O PRIMITIVA OPPURE AD UNA SEQUENZA DI AZIONI PRIMITIVE.**

**UN PROBLEMA COMPLESSO DOVRÀ ESSERE SCOMPOSTO IN SOTTO-PROBLEMI FINO A TROVARE UN INSIEME DI SOTTOPROBLEMI PRIMITIVI CHE RISULTI EQUIVALENTE AL PROBLEMA DI PARTENZA.**

# TOP–DOWN
# o “per raffinamenti successivi”



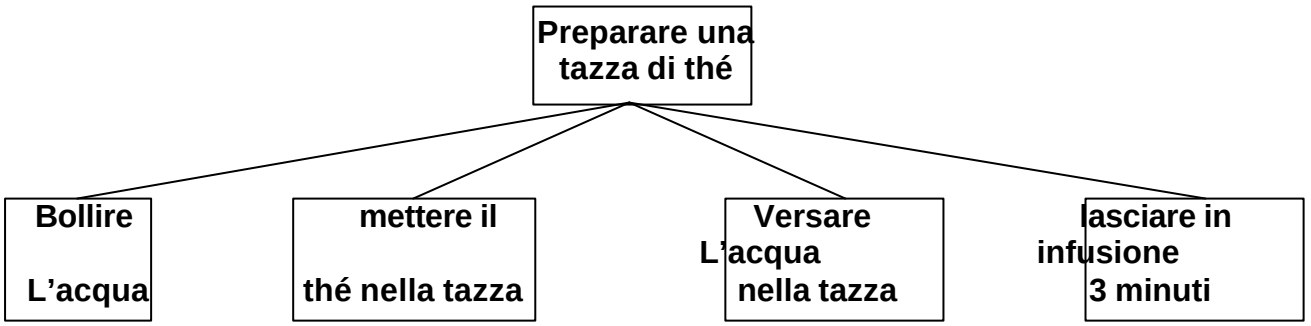

**E’ una rappresentazione ad albero: albero dello sviluppo top-down**

**Ad ogni passo di scomposizione ci si allontana dal linguaggio (ad alto livello) naturale e ci si avvicina alla descrizione nel linguaggio di programmazione.**

**Si potrebbe usare anche un linguaggio lineare.**

# La strategia TOP-DOWN consiste nel:

# TRASFORMARE UN PROBLEMA IN UNA GERARCHIA DI PROBLEMI DI DIFFICOLTÀ DECRESCENTE




**Inizio**

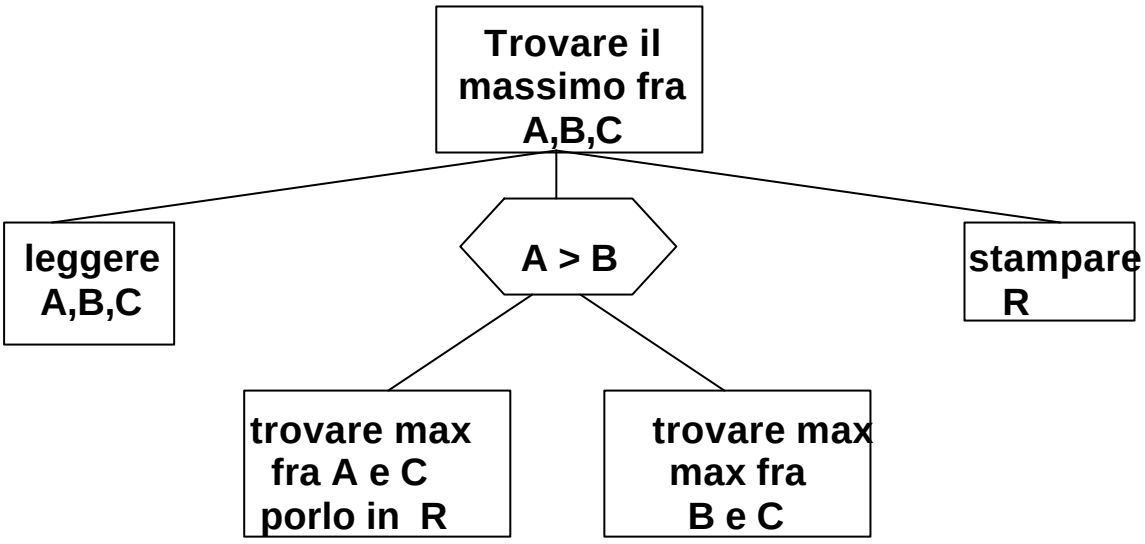
**1 1.1 riempire d'acqua il bollitore**
**1.2 accendere il gas**
**1.3 aspettare che l'acqua bolla**
**1.4 spegnere il gas**
**2 2.1 aprire la scatola del thé**
**2.2 prendere un sacchetto-filtro**
**2.3 chiudere la scatola del thé**
**2.4 mettere il filtro nella tazza**
**3 3.1 versare l'acqua bollente nella tazza finchè non sia piena**
**4 4.1 aspettare 3 minuti**
**4.2 estrarre il sacchetto-filtro**

**I sottoproblemi: P1,P2,….. sono fra loro INDIPENDENTI (dalla progettazione degli altri),**
**ovvero**
**per ciascuno di essi ha senso progettare un algoritmo**

**ma COOPERANO in quanto utilizzano i risultati prodotti dalla soluzione dei sottoproblemi che li precedono**

**Persone diverse potrebbero occuparsi di sottoproblemi diversi.**

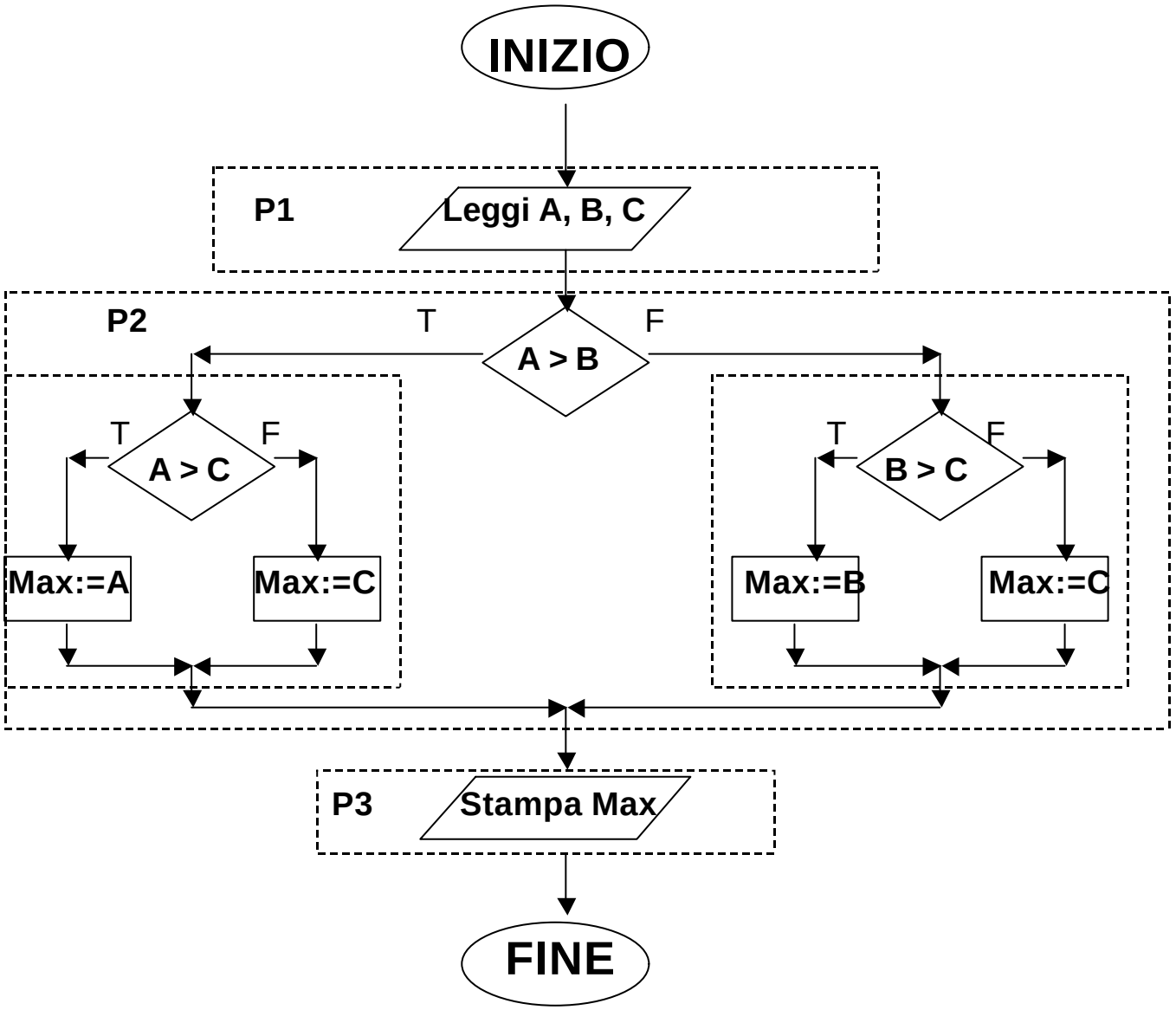
# PROBLEMA
# Trovare il più grande di tre numeri (a,b,c)




**se A>B**
**allora 2 trovare il max tra A e B**
**altrimenti 3 trovare il max tra B e C**

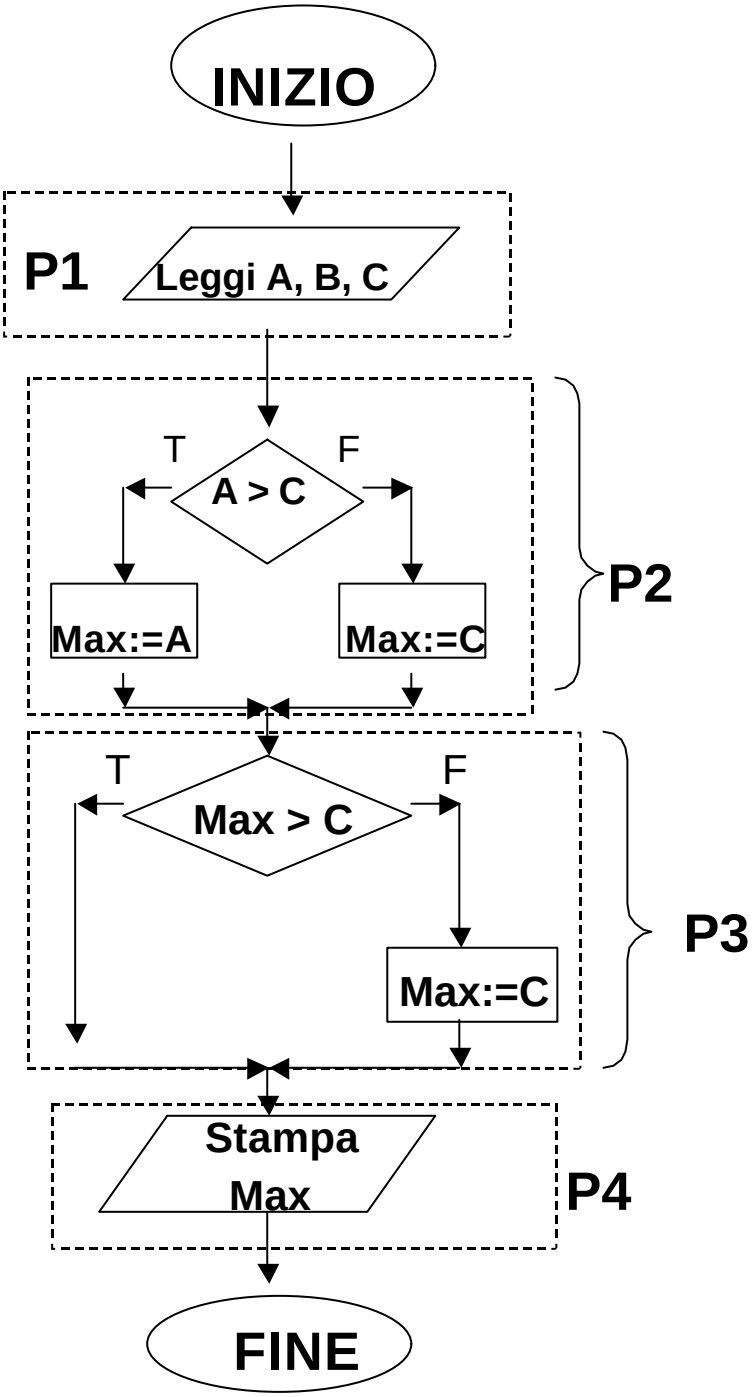
1.se A>B
allora
2.1 se A>C
2.2 allora il risultato è A
2.3 altrimenti il risultato è C
altrimenti
3.1 se B>C
3.2 allora il risultato è B
3.3 altrimenti il risultato è C

# PROBLEMA:
# Trovare il più grande di 3 numeri, A, B, e C




# Uso di strutture <u>selettive</u> "nidificate" rispecchia un metodo intuitivo di soluzione mediante confronti successivi, per separazione dei casi

# PROBLEMA:
# Trovare il più grande di 3 numeri, A, B, e C




# Uso di strutture selettive "in cascata" (metodo più sistematico di soluzione)

# SE DOVESSIMO DETERMINARE IL PIU' GRANDE FRA 100 valori?

**P 1 Trovare il più grande fra i primi due numeri dati**
**P 2.1 Mentre ci sono numeri da esaminare risolvi:**
**P 2.2 esaminare il primo numero non ancora considerato e trovare il più grande fra questo e il più grande precedentemente trovato.**

## SCOMPOSIZIONE ITERATIVA

**Si può costruire quando**
**SI INDIVIDUA UNA CATENA DI SOTTO PROBLEMI (come P2, P3, …, Pn-1)**
**CHE SODDISFANO LE SEGUENTI CONDIZIONI:**

- **I SOTTOPROBLEMI SONO UGUALI oppure DIFFERISCONO PER I DATI SU CUI AGISCONO**
- **I DATI SU CUI AGISCONO I SOTTOPROBLEMI SONO UGUALI oppure (quando sono diversi) SONO IN RELAZIONE D'ORDINE E UN PROBLEMA DIFFERISCE DAL PRECEDENTE SOLO PERCHÈ UTILIZZA IL DATO SUCCESSIVO**

# TECNICHE DI RAFFINAMENTO O DECOMPOSIZIONE

- **Suddividere il problema in parti disgiunte (sequenziale)**

- **Ripetere un numero finito di volte un passo di soluzione in modo che alla fine si abbia la soluzione completa (iterativo)**

- **Trattamento differenziato di casi differenti (selettivo)**

- **Decomposizione Ricorsiva**

**LE TIPICHE TECNICHE DI DECOMPOSIZIONE DEL METODO TOP-DOWN (FASE DI PROGETTO) TROVANO RISPONDENZA NEI COSTRUTTI DI CONTROLLO OFFERTI DAI LINGUAGGI. DI PROGRAMMAZIONE STRUTTURATI (ES. PASCAL).**

# SULLA DECOMPOSIZIONE RICORSIVA

## IL CONCETTO DI RICORSIONE

**UN OGGETTO SI DICE RICORSIVO SE E' DEFINITO IN TERMINI DI SE STESSO.**

**Esempi**

- **I NUMERI NATURALI**

a) **1 è un numero naturale**
b) **il successore di un numero naturale è un numero naturale.**

- **I GRAFI DI FLUSSO STRUTTURATI**

- **LE STRUTTURE AD ALBERO**

**QUANDO DURANTE LA SCOMPOSIZIONE DI UN PROBLEMA IN SOTTO-PROBLEMI, I SOTTO-PROBLEMI RISULTANO FORMALMENTE SIMILI AL PROBLEMA DI PARTENZA IL PROBLEMA SI DICE RISOLTO**

## RICORSIVAMENTE

**L'USO DELLA RICORRENZA E DI RELAZIONI DI RICORRENZA CONSENTE, IN MATEMATICA, DI DARE DEFINIZIONI IN TERMINI DI SE STESSE.**

**ESEMPIO**

**LA FUNZIONE FATTORIALE :**

$$N! = N * (N-1) * (N-2) *\ldots *2 * 1$$

**OPPURE RICORSIVAMENTE:**

$$N! = \begin{cases} 1 & \text{se } N = 1 \\ N * (N-1)! & \text{Se } N > 1 \end{cases}$$

- **UNA DEFINIZIONE RICORSIVA DEFINISCE UNA ENTITA' IN TERMINI DI VERSIONI PIU' SEMPLICI DELLA STESSA ENTITA' CHE SI STA DEFINENDO.**

- **UN PROBLEMA DI ORDINE N E' DEFINITO IN TERMINI DEL MEDESIMO PROBLEMA DI ORDINE INFERIORE**

- **E' INDISPENSABILE CHE NELLA DEFINIZIONE CI SIA IL <u>LIVELLO ASSIOMATICO</u>: E' GRAZIE AD ESSO CHE POSSONO RICOSTRUIRSI I LIVELLI SUCCESSIVI**

**ESEMPI**

- **IL PRODOTTO DI DUE NUMERI x ED y:**

$$X * Y = \begin{cases} X & \text{se } Y = 1 \\ X + X * (Y - 1) & \text{se } Y > 1 \end{cases}$$

- **INVERSIONE DI UNA STRINGA:**

**AMOR-R(AMO)-RO(AM)-ROM(A)-ROMA**

**E' POSSIBILE SCOMPORRE UN PROBLEMA FACENDO USO DI RELAZIONI DI RICORRENZA.**

- **SE E' NOTO L'ORDINE DEL PROBLEMA, LA SCOMPOSIZIONE PUO' ATTUARSI MEDIANTE LA REGOLA DI COPIATURA SINO AD AVERE PROBLEMI PRIMITIVI (OVVERO SINO A RAGGIUNGERE IL LIVELLO ASSIOMATICO)**

- **LA REGOLA CONSISTE NEL SOSTITUIRE AD OGNI OCCORRENZA DEL PROBLEMA LA SCOMPOSIZIONE RICORSIVA.**

**PROBLEMA**
**CALCOLARE IL FATTORIALE DEL NUMERO NATURALE N**

**E' POSSIBILE PROGETTARE SIA UN ALGORITMO ITERATIVO CHE UNO RICORSIVO.**

# IMPORTANTE

- **UN ALGORITMO ESPRESSO RICORSIVAMENTE TERMINA SEMPRE.**

- **UNA FUNZIONE (O UN PROBLEMA) ESPRIMIBILE RICORSIVAMENTE PUO' RISOLVERSI ITERATIVAMENTE.**

- **UNA FUNZIONE ESPRIMIBILE RICORSIVAMENTE E' COMPUTABILE (ESISTE UN METODO SOLUTIVO TIPO ALGORITMO)**

- **OGNI FUNZIONE COMPUTABILE PER MEZZO DI UN PROGRAMMA E' RICORSIVA**

# AI MECCANISMI DI SCOMPOSIZIONE CORRISPONDONO I DIFFERENTI LIVELLI DI COMPLESSITÀ DEGLI ALGORITMI:

- **LA FORMULA**
- **LA SEQUENZA DI FORMULE CON RISULTATI INTERMEDI**
- **GLI ALGORITMI CONDIZIONALI**
- **GLI ALGORITMI ITERATIVI**
- **GLI ALGORITMI RICORSIVI**

# IN SINTESI:

# APPROCCIO TOP-DOWN STRUMENTO CONCETTUALE PER LA COSTRUZIONE DI ALGORITMI

**SI PARTE DAL LIVELLO PIÙ ALTO FINO A GIUNGERE, ATTRAVERSO LA DEFINIZIONE DELLA STRUTTURA, AL LIVELLO PIÙ BASSO DI DETTAGLIAMENTO**

**IL PROCESSO DI SCOMPOSIZIONE / RAFFINAMENTO TERMINA QUANDO TUTTI I SOTTO-PROBLEMI SONO PRIMITIVI**

**PER MOLTI ALGORITMI BASTA SCENDERE DI DUE O TRE LIVELLI. QUANTO PIÙ VASTO E COMPLESSO È IL PROBLEMA TANTO PIÙ SARÀ NECESSARIO SCENDERE DI LIVELLO.**

**UNO STESSO PROBLEMA PUÒ ESSERE COMPOSTO IN DIFFERENTI MODI!**