4 - ESEMPI; LO PSEUDOCODICE (O "LINGUAGGIO DI PROGETTO")

Qui di seguito daremo alcuni esempi di algoritmi costruiti secondo i principi della "programmazione strutturata".

Accanto a ciascun diagramma di flusso scriveremo anche la corrispondente versione in
pseudocodice (o **pseudolinguaggio**, o **linguaggio di progetto**):
si tratta di una specie di **"linguaggio di programmazione semplificato", non standard**
(noi ne diamo infatti una *nostra* versione, non necessariamente coincidente con altre)
a partire dal quale è poi facile tradurre l'algoritmo in un vero e proprio linguaggio comprensibile dal computer, come ad esempio il linguaggio PASCAL o il linguaggio C o un qualsiasi altro.

… Anche se poi, traducendo lo *pseudocodice* in vero e proprio linguaggio, occorre tenere conto

- delle rigide regole sintattiche di quel linguaggio
- delle risorse proprie del linguaggio adottato (tanto per fare un esempio,
  la RIPETI … FINCHE' … , con la quale si ha uscita dal ciclo quando la condizione è verificata,
  va realizzata, in linguaggio C, attraverso le parole chiave DO … WHILE …
  ma tenendo conto che questa struttura del linguaggio C prevede si esca dal ciclo
  quando la condizione diventa FALSA, e *non* quando diventa *vera*!
  (… Ovviamente, basterà sostituire alla condizione in gioco la sua contraria per sistemare le cose …)
- dell'eventuale presenza, in quel linguaggio, di comode varianti. Tanto per fare due esempi:
  a) nel linguaggio PASCAL accanto alle due strutture iterative con controllo finale e con controllo iniziale, abbiamo pure una struttura iterativa "enumerativa" FOR … DO … che ordina al computer di ripetere l'istruzione, o il blocco di istruzioni, per un numero prefissato di volte
  b) sempre in PASCAL, abbiamo anche una struttura di "selezione multipla" (la CASE … OF …)

- ESEMPIO 1

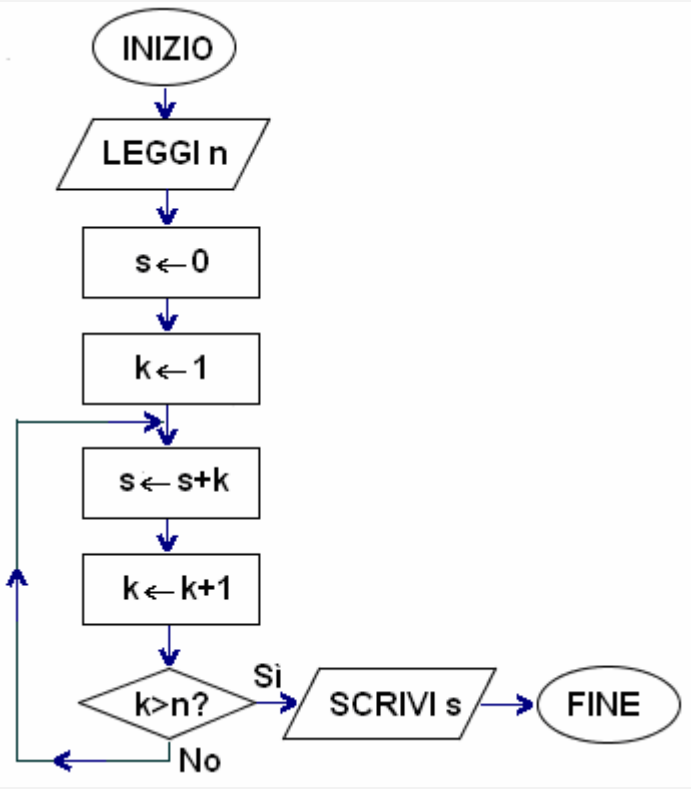
  Il seguente algoritmo serve per
  sommare i primi n interi positivi, ossia eseguire la somma $1+2+3+...+n$, con n letto in ingresso (facciamo finta di non sapere che per questo calcolo c'è l'apposita "formula di Gauss").

  Abbiamo utilizzato la "**freccia a sinistra**" $\leftarrow$ per indicare l'**istruzione di assegnazione**:
  ad esempio, $\mathbf{s \leftarrow 0}$ significa "assegna alla variabile s il valore 0" (metti nella "scatoletta" s il valore 0);
  $\mathbf{k \leftarrow k+1}$ significa "assegna alla variabile k il valore che essa aveva precedentemente, aumentato di 1" (incrementa di 1 il valore della variabile k).

  In linguaggio Pascal, l'assegnazione viene indicata con **:=** *e in linguaggio C con* **=** *senza i "puntini"*

  Osserviamo che la variabile s fa da "**accumulatore**": essa ha il ruolo di "somma parziale",
  e via via, a forza di "accumulare" contributi, si porta a diventare la somma di tutti quanti gli addendi.

DIAGRAMMA DI FLUSSO



LINGUAGGIO DI PROGETTO

INIZIO

LEGGI n

s ← 0

k ← 1

RIPETI
{ s ← s+k

k ← k + 1 }

FINCHE' k>n
SCRIVI s
FINE

La **coppia di GRAFFE** ci serve per evidenziare che quelle istruzioni "fanno blocco", "vanno insieme".

Anche nel linguaggio C le graffe vengono utilizzate con questa funzione.

Invece in linguaggio PASCAL al posto delle graffe si impiegano (se necessario) due indicatori di "inizio blocco" e "fine blocco": BEGIN e END.

- ESEMPIO 2

L'algoritmo che segue legge un intero >2 in ingresso e stabilisce se è o non è un numero primo.

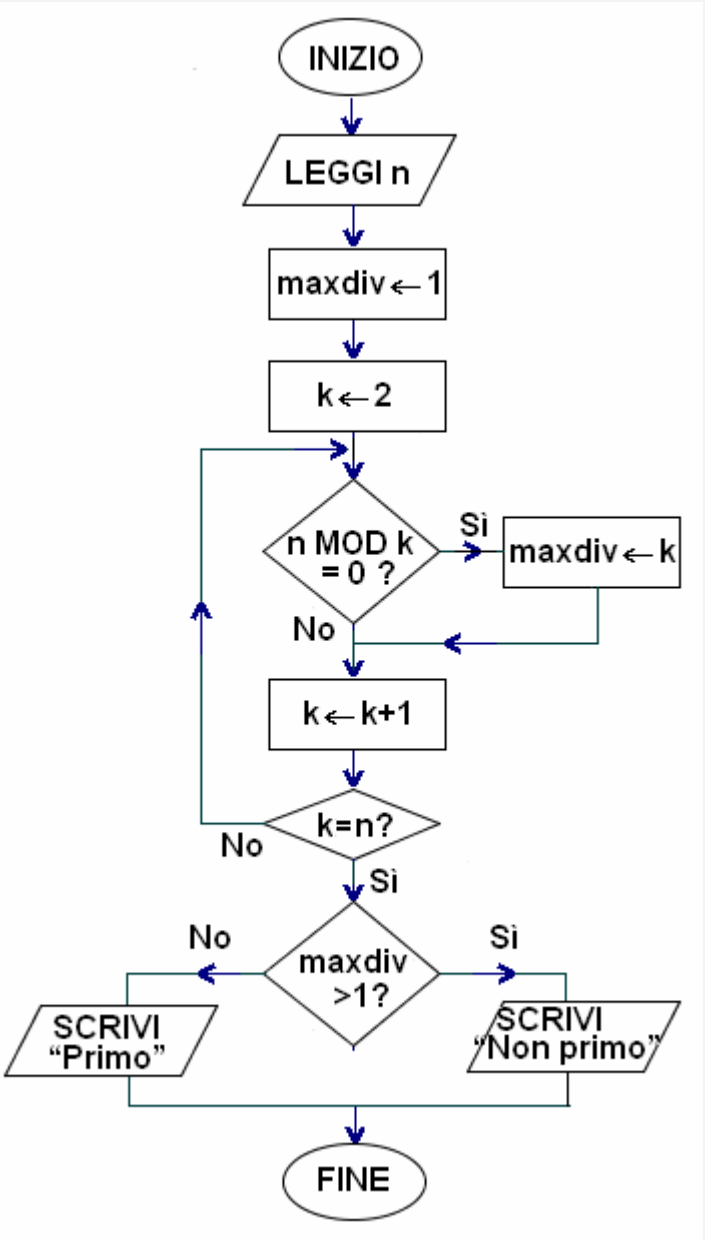
Ricordiamo che
l'operatore MOD dà il resto della divisione intera (mentre DIV ne dà il quoziente intero);
ad esempio, 47 MOD 5 = 2 perché il resto della divisione intera 47:5 è 2 (mentre 47 DIV 5 = 9).
Se a, b sono due interi, a è divisibile per b se e solo se a MOD b = 0.

La variabile maxdiv immagazzina il "massimo fra i divisori trovati fino a quel momento"
e diventerà alla fine il massimo fra i divisori di n, inferiori ad n.
Se, al termine dell'elaborazione, nella "scatoletta" maxdiv ci sarà un numero >1,
vorrà dire che è stato trovato un divisore diverso sia dall'unità che dal numero n stesso,
per cui se ne concluderà che n non è primo.

Il diagramma di flusso contiene due strutture di selezione:
- la prima offre una sola alternativa (SE … ALLORA … senza "altrimenti"),
- la seconda ne offre classicamente due (SE … ALLORA … ALTRIMENTI …)

Controlla tu stesso che con questo algoritmo,
se il numero n dato in ingresso *non* fosse >2,
si rimarrebbe intrappolati in un *ciclo senza fine*.

DIAGRAMMA DI FLUSSO



LINGUAGGIO DI PROGETTO

INIZIO

LEGGI n

maxdiv ← 1

k ← 2

RIPETI

{ SE n MOD k = 0 ALLORA maxdiv ← k

k ← k+1 }

FINCHE' k = n

SE maxdiv > 1 ALLORA
SCRIVI "Non primo"
ALTRIMENTI
SCRIVI "Primo"

FINE

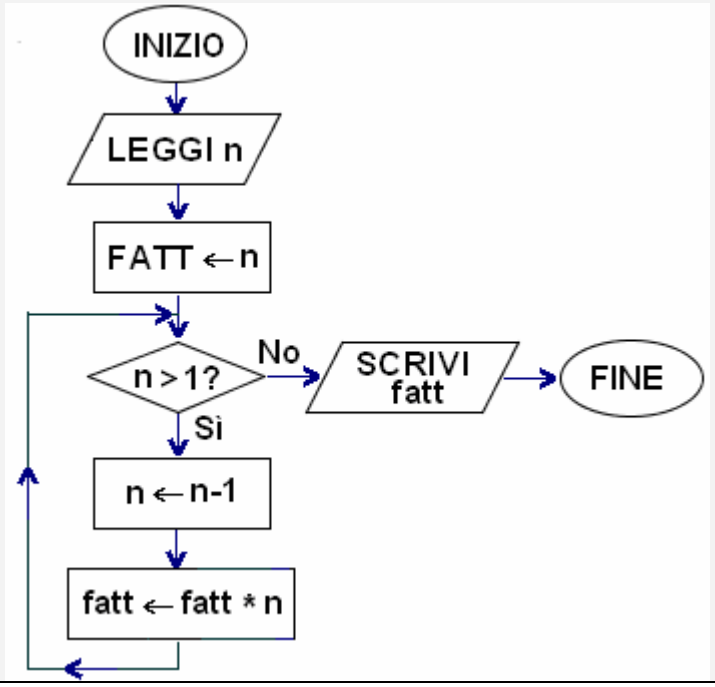
❑ ESEMPIO 3

Il seguente algoritmo calcola il **fattoriale** $\mathbf{n! = n \cdot (n-1) \cdot (n-2) \cdot ... \cdot 2 \cdot 1}$ di un intero $n \geq 1$ letto in ingresso.

n, se è maggiore di 1, viene fatto decrescere sempre di un'unità a partire dal valore iniziale,
e una variabile fatt, inizializzata ad n, viene moltiplicata per ogni nuovo valore di n
generando così, via via, il fattoriale desiderato.

Affinché il procedimento funzionasse anche con n = 1,
valore per il quale il decremento di n NON va effettuato nemmeno una volta,
in questo algoritmo abbiamo utilizzato la STRUTTURA ITERATIVA A CONTROLLO INIZIALE,
che in linguaggio di progetto abbiamo tradotto con FINTANTOCHE' … ESEGUI …

DIAGRAMMA DI FLUSSO



LINGUAGGIO DI PROGETTO

INIZIO

LEGGI n

fatt ← n

FINTANTOCHE' n > 1 ESEGUI

{ n ← n – 1

fatt ← fatt * n }

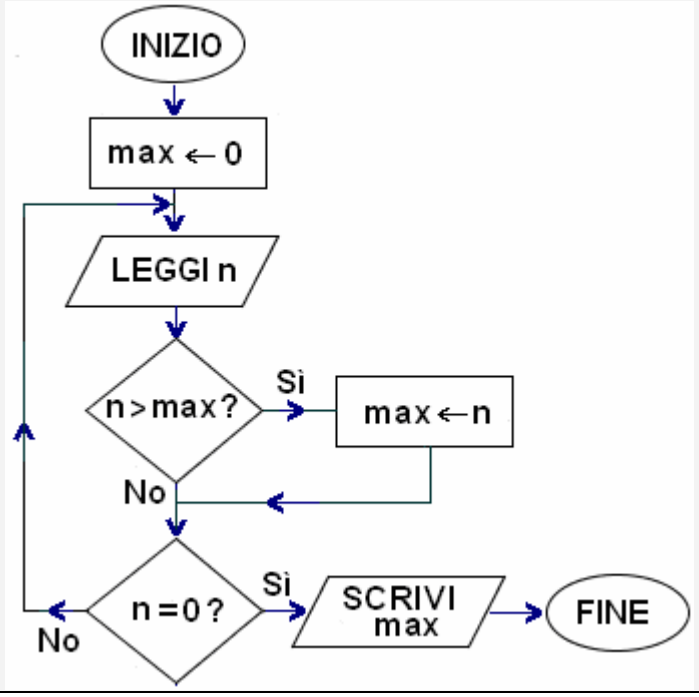
SCRIVI fatt
FINE

Nel nostro linguaggio di progetto la moltiplicazione è indicata con l'asterisco *, come nei linguaggi Pascal e C.

❑ ESEMPIO 4

Si leggono in input più numeri positivi, anche eventualmente tantissimi, e se ne stabilisce il massimo.
La fine della sequenza numerica in ingresso viene segnalata comunicando un "**numero sentinella**": lo 0.

DIAGRAMMA DI FLUSSO



LINGUAGGIO DI PROGETTO

INIZIO

max ← 0

RIPETI

{ LEGGI n

SE n > max ALLORA max ← n }

FINCHE' n=0
SCRIVI max
FINE