

FONDAMENTI DI INFORMATICA

Prof. PIER LUCA MONTESSORO

Prof. ELIO TOPPANO

Facoltà di Ingegneria
Università degli Studi di Udine

Introduzione alla programmazione strutturata

Nota di Copyright

Questo insieme di trasparenze (detto nel seguito slide) è protetto dalle leggi sul copyright e dalle disposizioni dei trattati internazionali. Il titolo ed i copyright relativi alle slides (ivi inclusi, ma non limitatamente, ogni immagine, fotografia, animazione, video, audio, musica e testo) sono di proprietà degli autori prof. Pier Luca Montessoro e ing. Elio Toppano, Università degli Studi di Udine.

Le slide possono essere riprodotte ed utilizzate liberamente dagli istituti di ricerca, scolastici ed universitari afferenti al Ministero della Pubblica Istruzione e al Ministero dell'Università e Ricerca Scientifica e Tecnologica, per scopi istituzionali, non a fine di lucro. In tal caso non è richiesta alcuna autorizzazione.

Ogni altro utilizzo o riproduzione (ivi incluse, ma non limitatamente, le riproduzioni su supporti magnetici, su reti di calcolatori e stampe) in toto o in parte è vietata, se non esplicitamente autorizzata per iscritto, a priori, da parte degli autori.

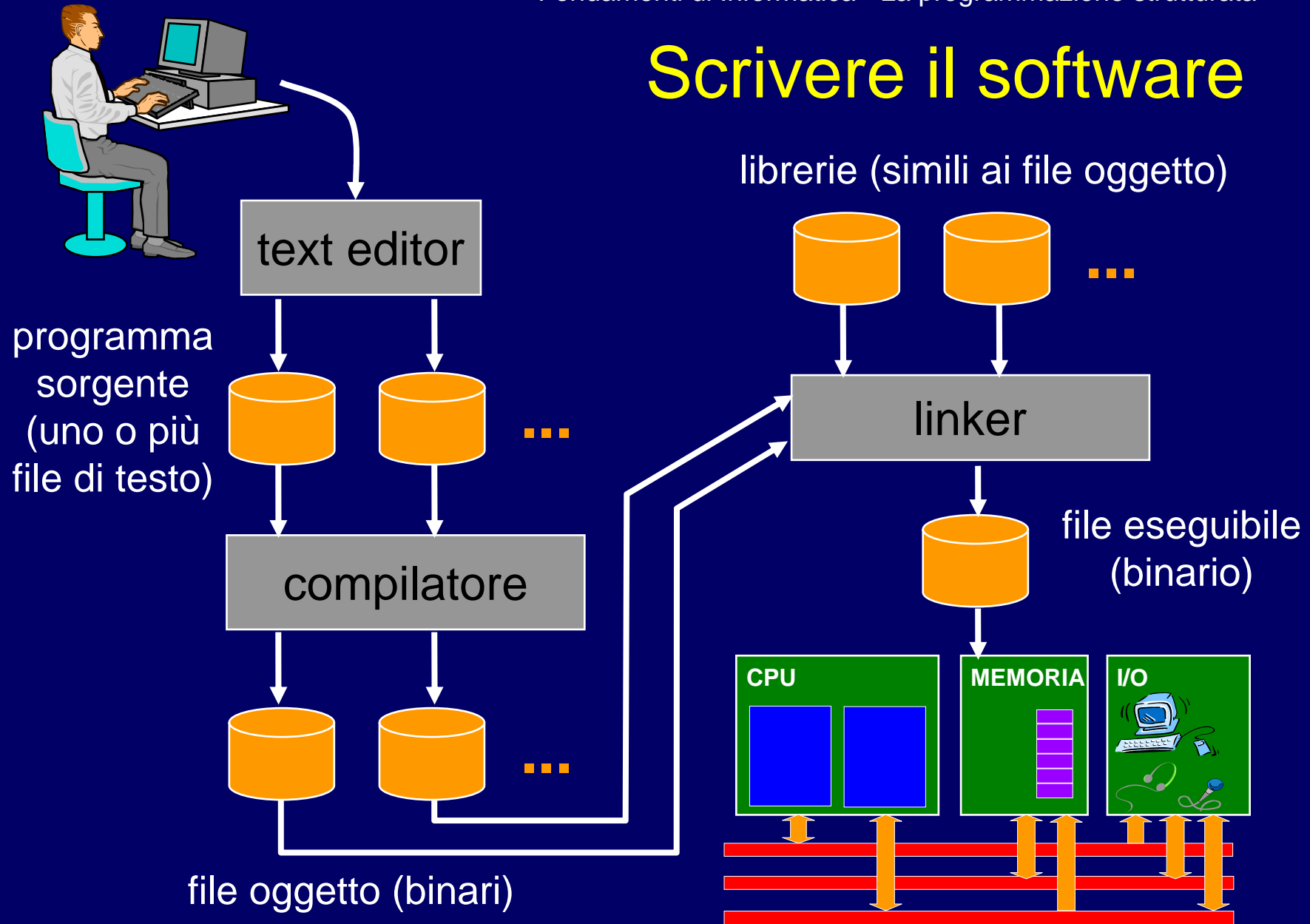
L'informazione contenuta in queste slide è ritenuta essere accurata alla data della pubblicazione. Essa è fornita per scopi meramente didattici e non per essere utilizzata in progetti di impianti, prodotti, reti, ecc. In ogni caso essa è soggetta a cambiamenti senza preavviso. L'autore non assume alcuna responsabilità per il contenuto di queste slide (ivi incluse, ma non limitatamente, la correttezza, completezza, applicabilità, aggiornamento dell'informazione).

In ogni caso non può essere dichiarata conformità all'informazione contenuta in queste slide.

In ogni caso questa nota di copyright e il suo richiamo in calce ad ogni slide non devono mai essere rimossi e devono essere riportati anche in utilizzi parziali.

Scrivere il software

Scrivere il software



Compilatore e linker

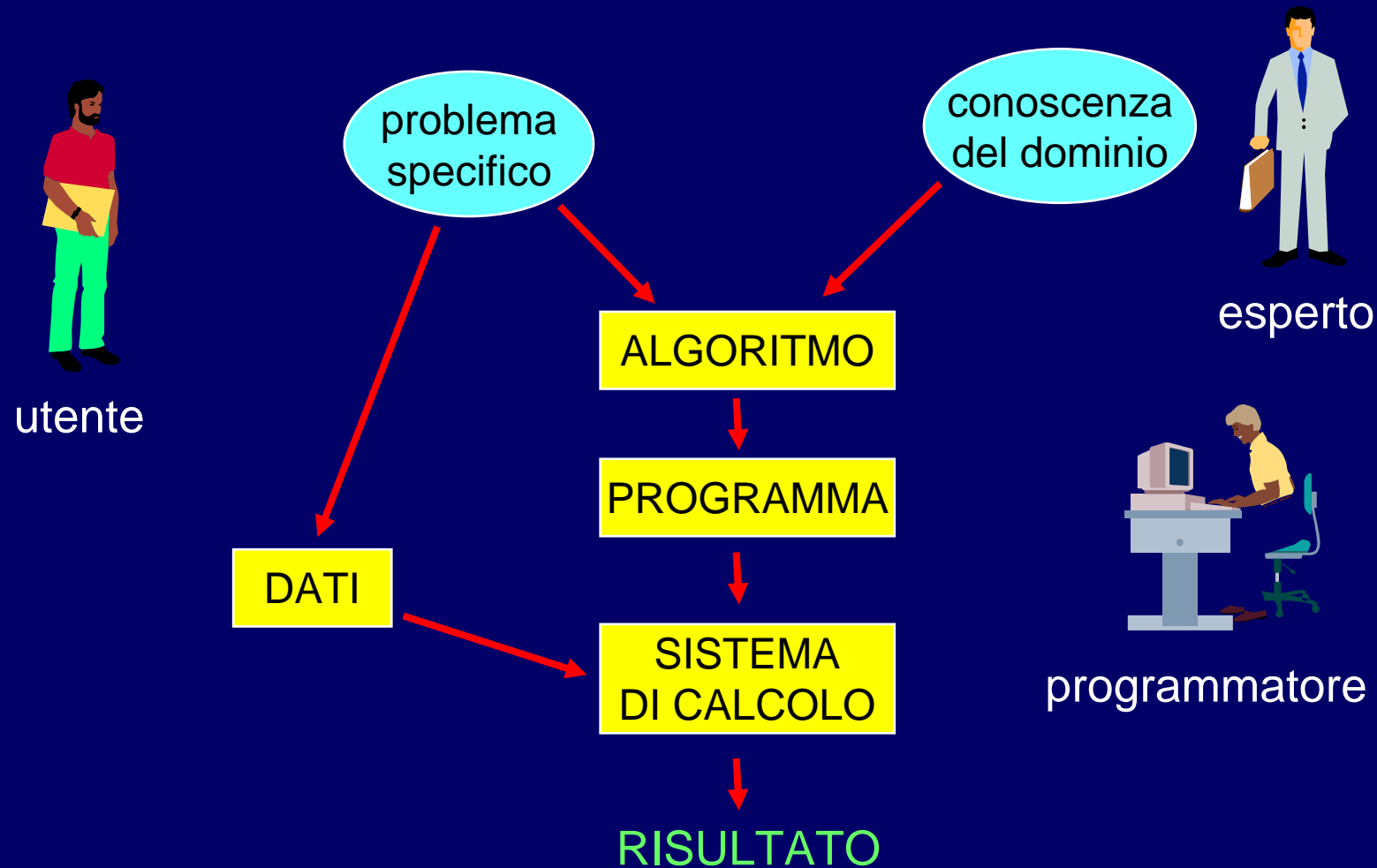
- Il **compilatore** traduce il programma sorgente (scritto in linguaggio “ad alto livello” in lunghe sequenze di istruzioni in linguaggio “macchina”)
- Il **linker** “collega” al programma compilato le sequenze di istruzioni già scritte e rese disponibili al programmatore mediante le “librerie”

Interprete

- Un interprete legge il programma sorgente e lo esegue man mano che lo traduce (istruzione per istruzione)
- A differenza del compilatore, non genera un file contenente il codice eseguibile, ma ritraduce il sorgente ogni volta
- È utilizzato soprattutto per i linguaggi di comandi dei sistemi operativi

Progettare il software

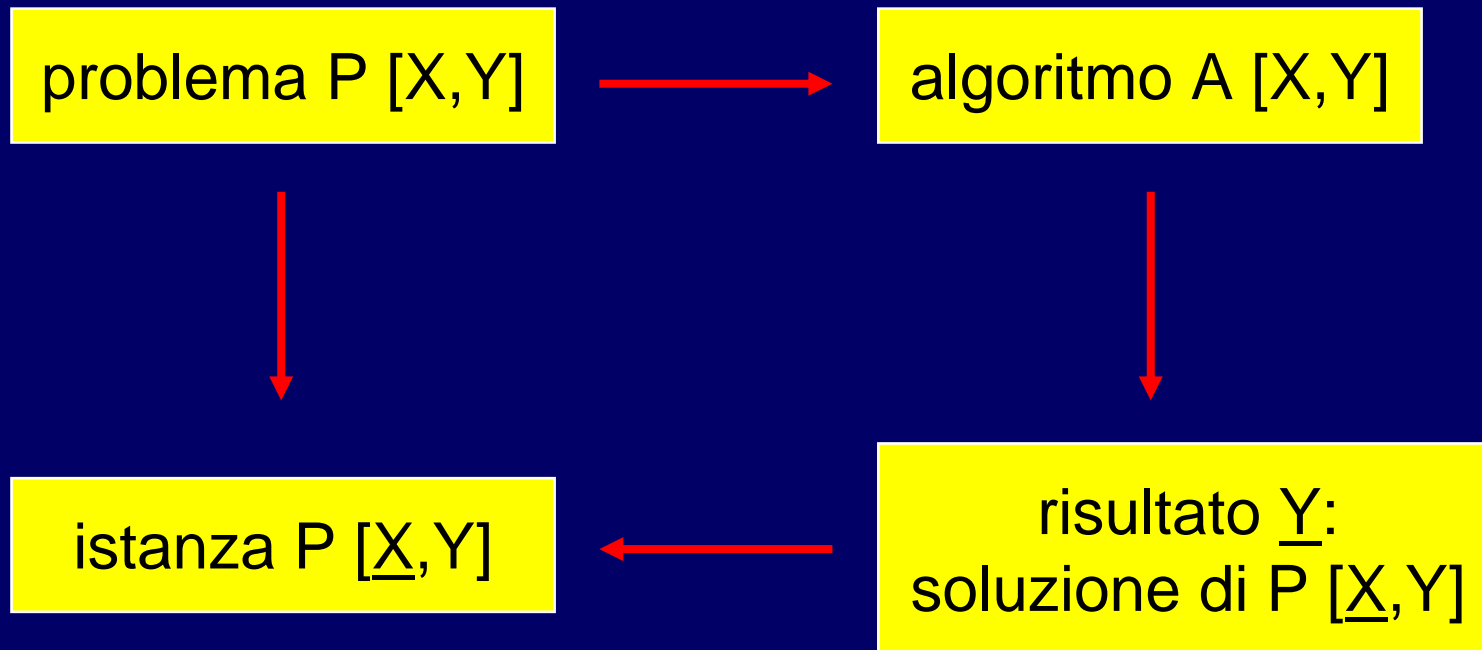
Lo scenario di riferimento



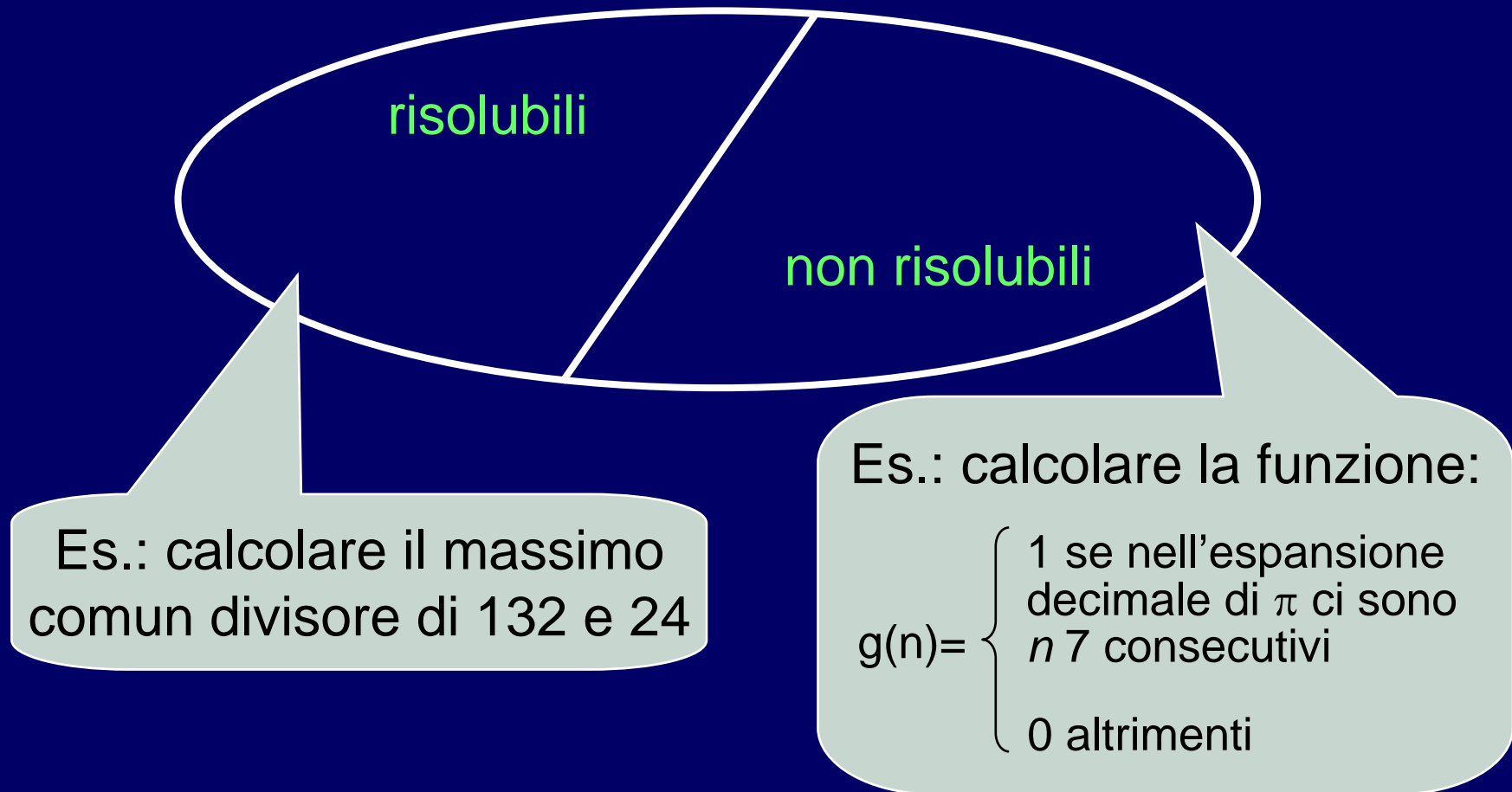
Metodi risolutivi e algoritmi

- **Metodo risolutivo:**
 - Insieme di (oper)azioni (algebriche, logiche, materiali, ecc.) che, eseguite ordinatamente, permettono di ottenere il risultato voluto a partire dalle informazioni a nostra disposizione (dati di ingresso)
- **Algoritmo:**
 - metodo risolutivo che soddisfa le proprietà di:
ESEGUIBILITÀ, NON-AMBIGUITÀ,
FINITEZZA

Problemi, algoritmi e dati

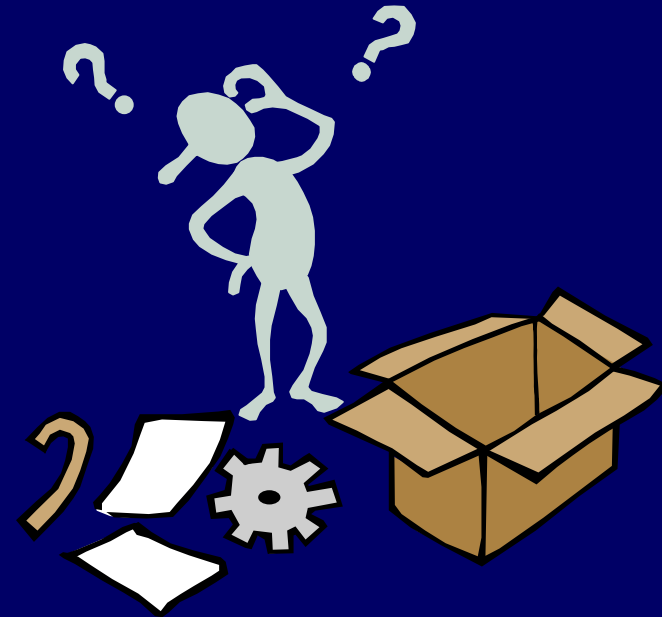


Problemi



Esecutori ed eseguibilità

- L'algoritmo deve essere espresso in termini di (oper)azioni che l'esecutore è in grado di compiere!!!



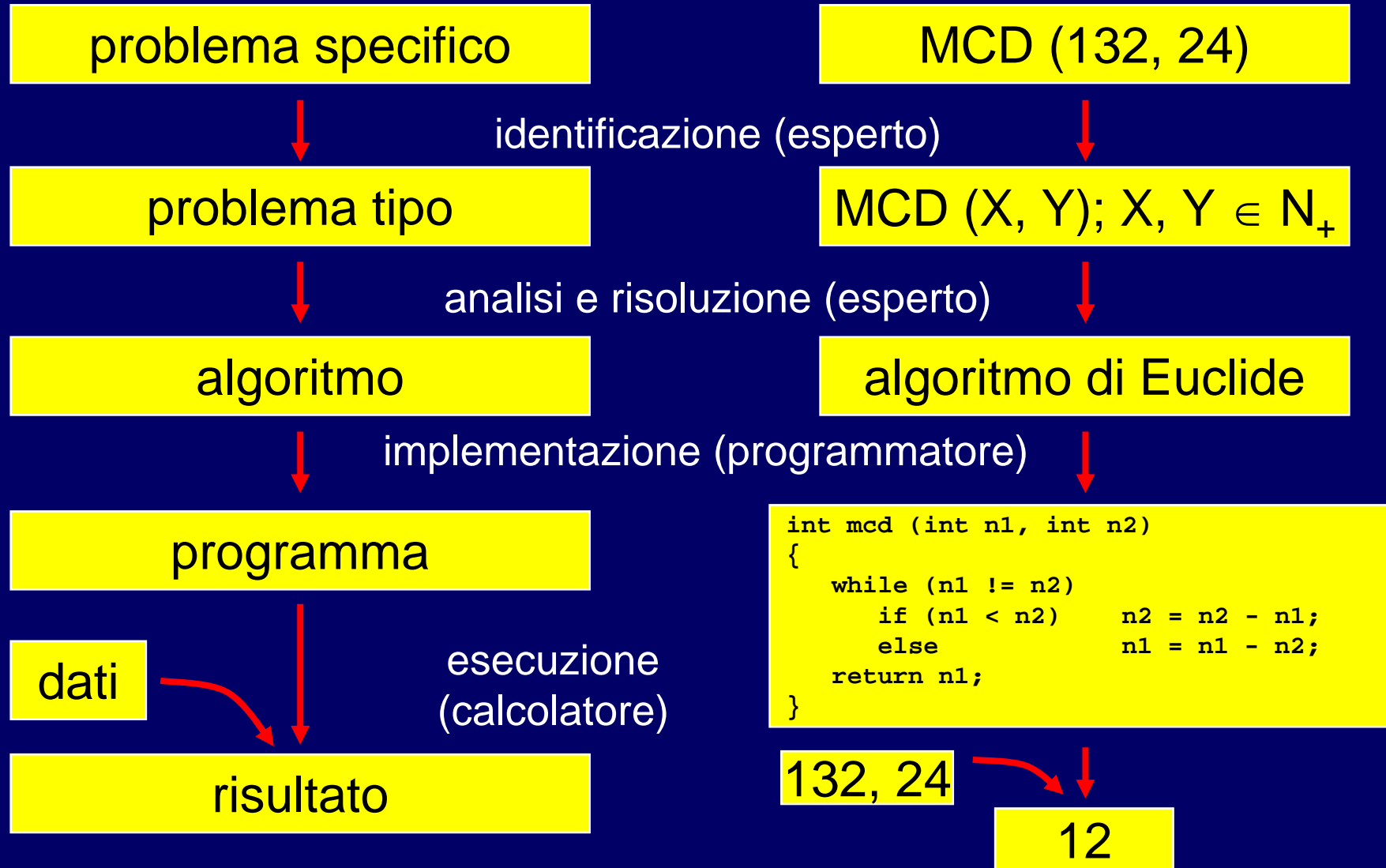
Linguaggio di programmazione

- Insieme di regole per la descrizione formale di un algoritmo eseguibile da un calcolatore
 - lessico: insieme dei termini disponibili
 - sintassi: forma delle frasi
 - semantica: significato delle frasi

Programma

- Descrizione di un algoritmo in un linguaggio di programmazione
- È composto da un numero finito di istruzioni
- Ogni istruzione descrive una (oper)azione

Esempio



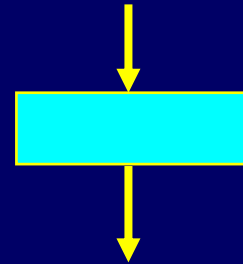
Progetto di un programma

- Specifica dell'algoritmo
 - specifica dei DATI (Tipi di Dati Astratti)
 - specifica della STRUTTURA DI CONTROLLO (diagrammi di flusso mediante schemi a blocchi)
- Implementazione
 - traduzione dei DATI (Tipi di Dati Concreti)
 - traduzione della STRUTTURA DI CONTROLLO (assegnamento, meccanismi di controllo)

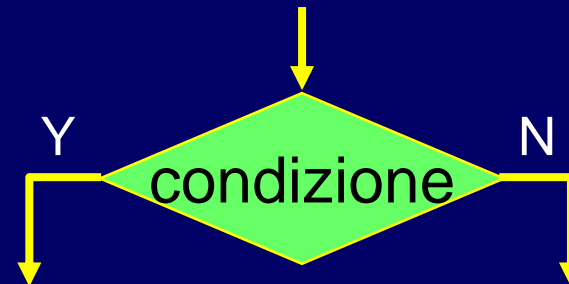
Programmazione strutturata

Schemi a blocchi

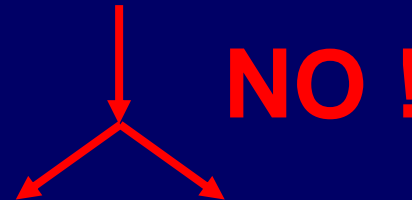
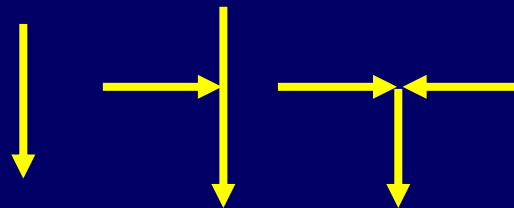
- Blocco funzionale



- Blocco di decisione



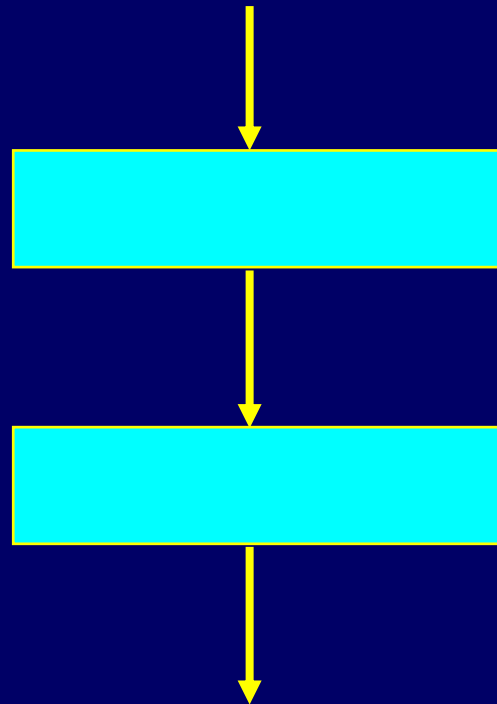
- Archi

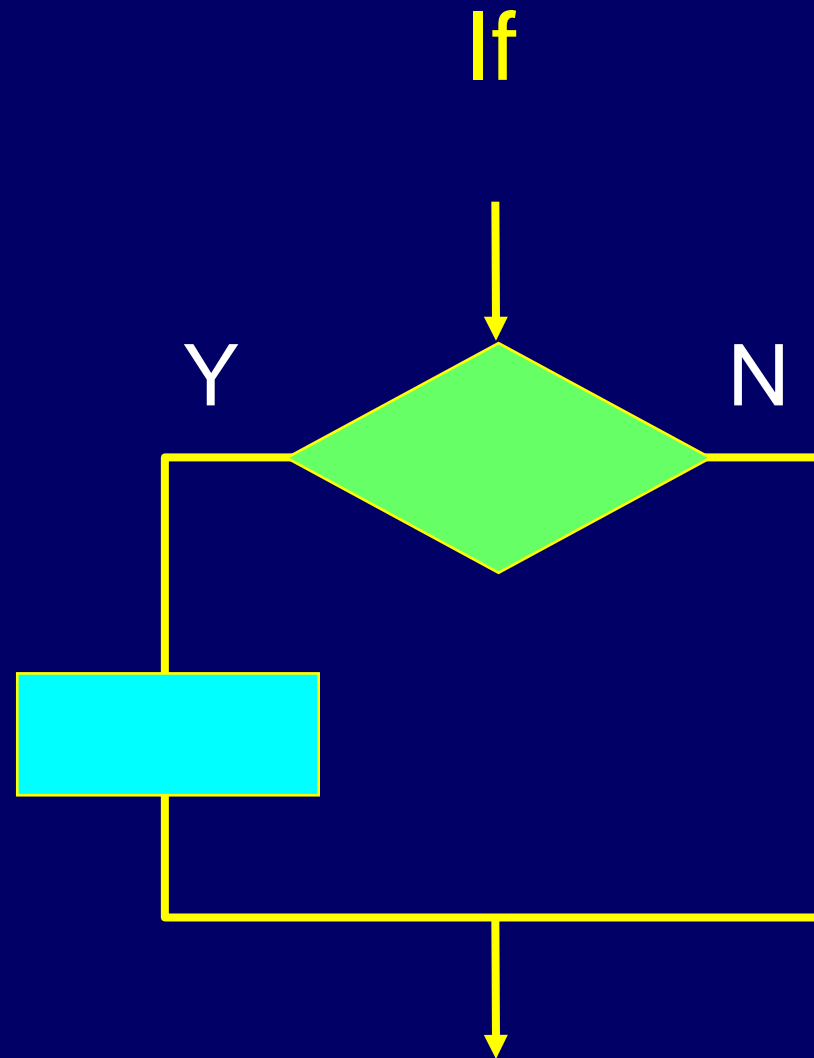


Schemi a blocchi strutturati

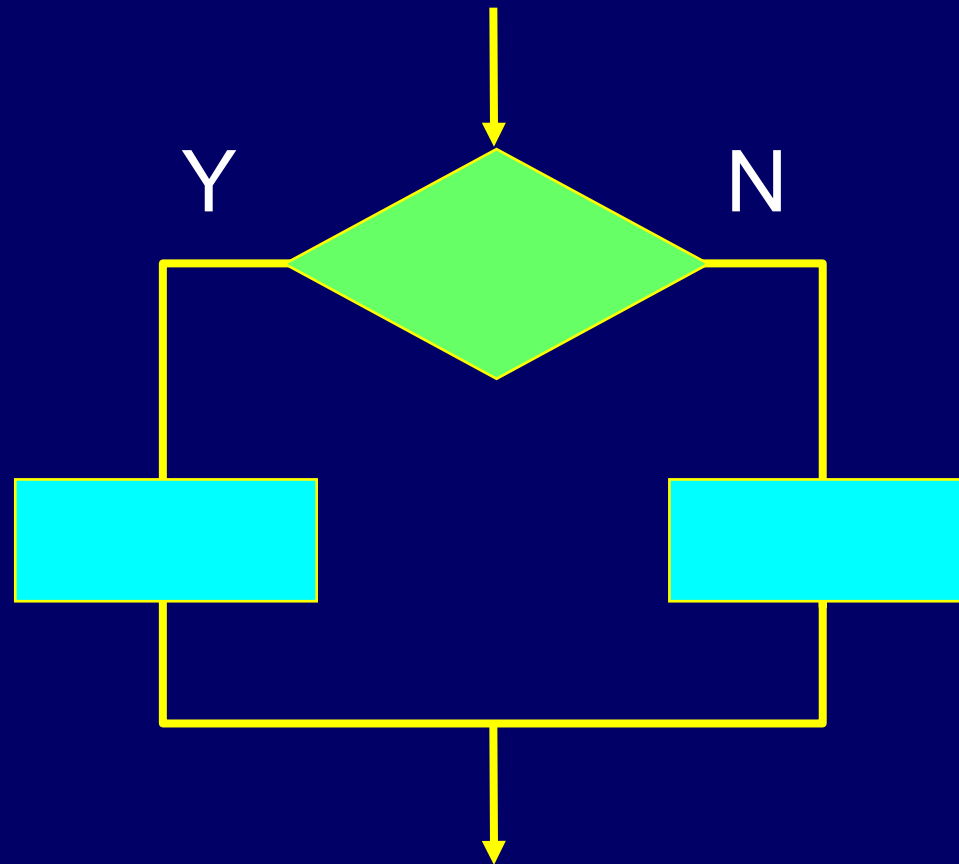
- Impongono severe restrizioni nella costruzione dei diagrammi di flusso
- Si basano su poche strutture di base con un solo ingresso e una sola uscita
- Tali strutture sono sufficienti per descrivere qualsiasi algoritmo risolutore di un problema risolubile (teorema di Böhm-Jacopini)

Sequenza

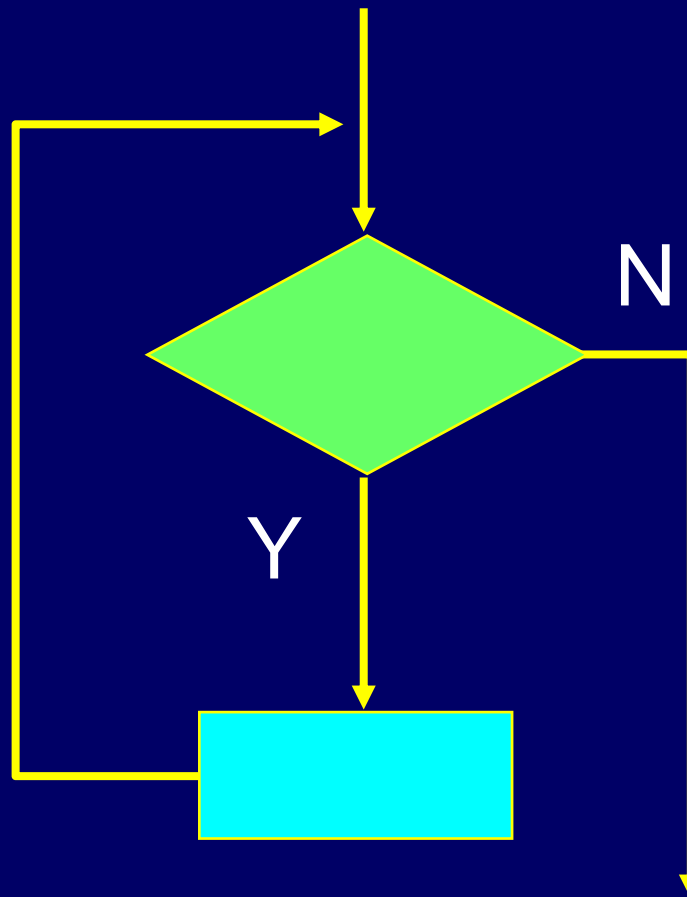




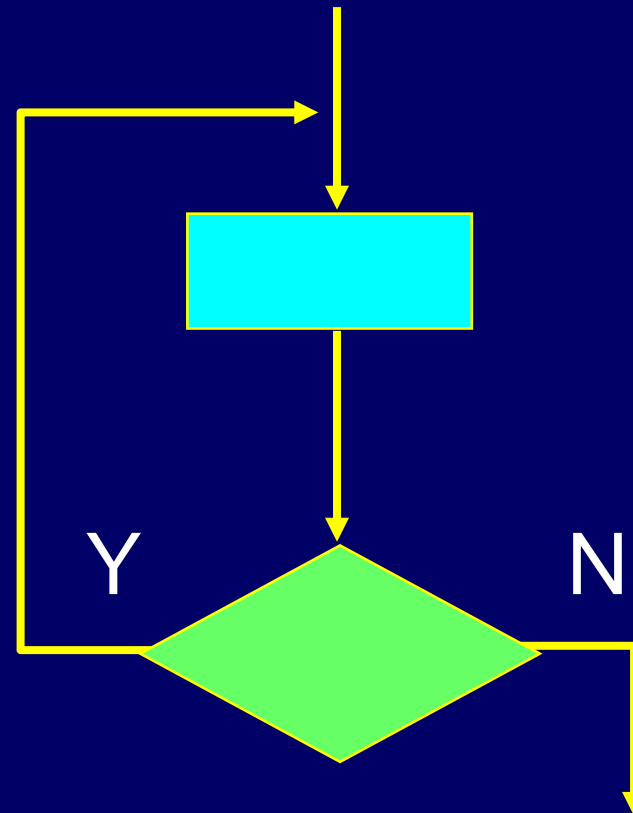
If - else



Ciclo while

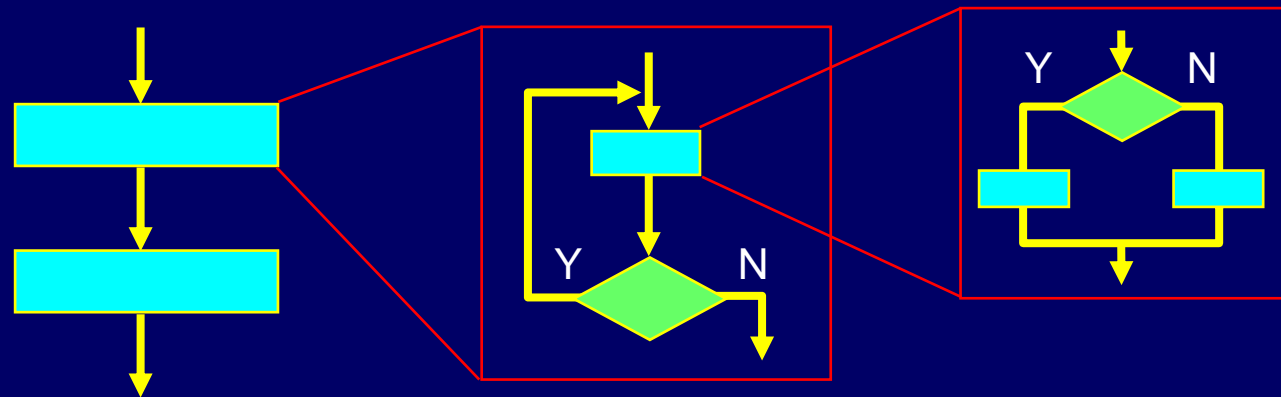


Ciclo do - while (o repeat)



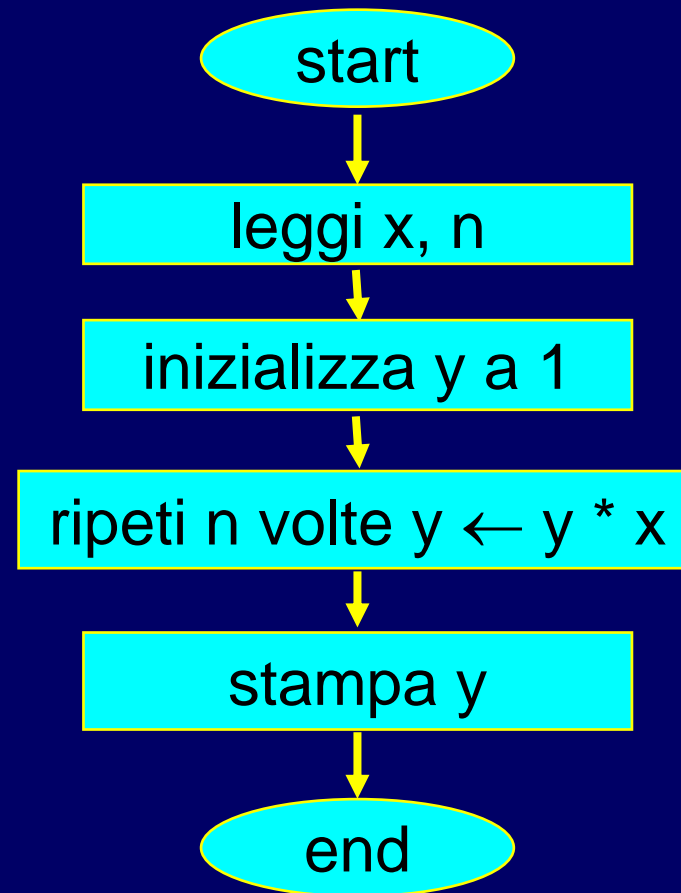
Sviluppo “top-down” dei programmi

- Scomposizioni successive in sottoproblemi
- Sostituzione nel diagramma di flusso dei blocchi con costrutti via via di maggior dettaglio

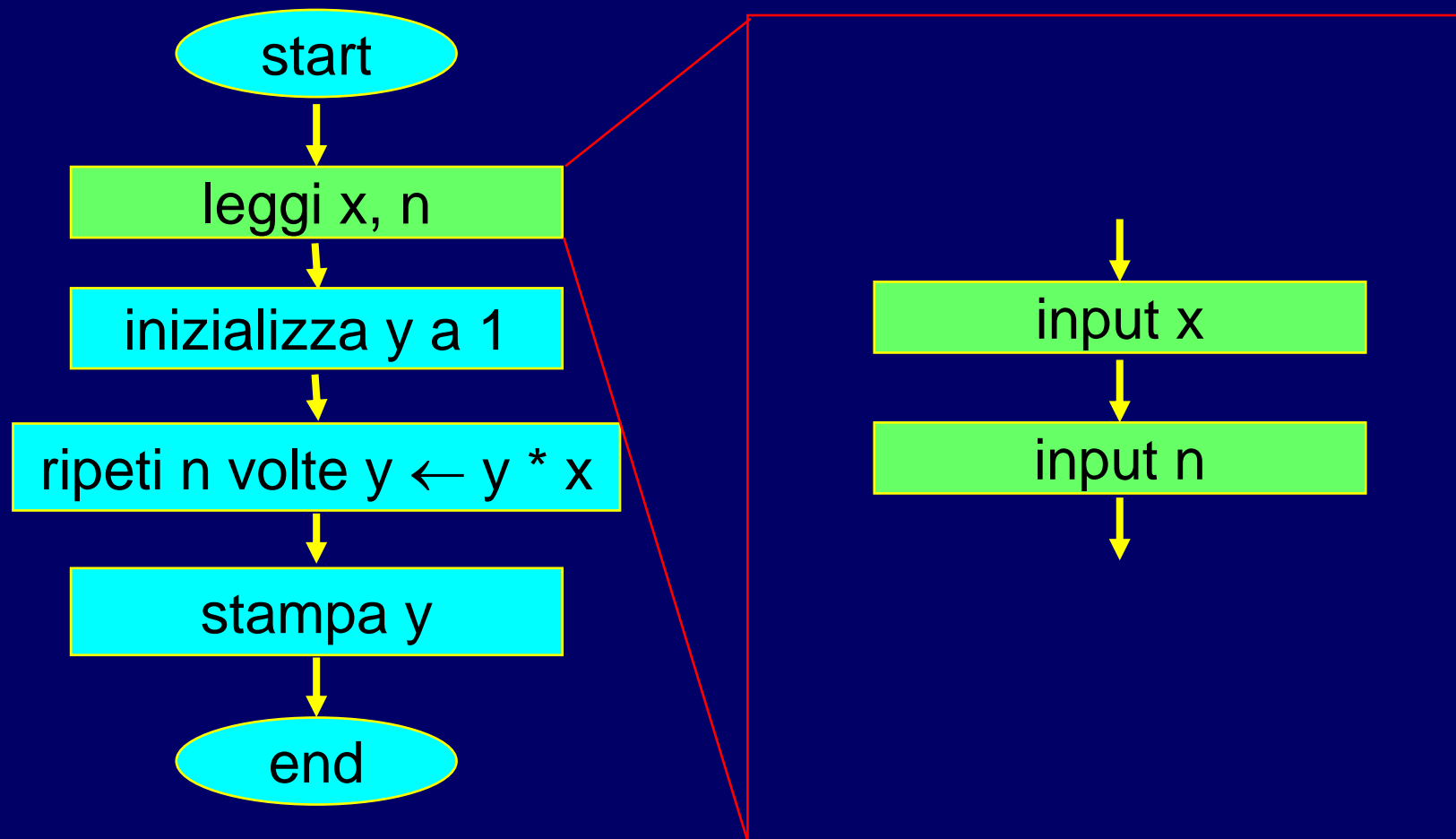


Esempio

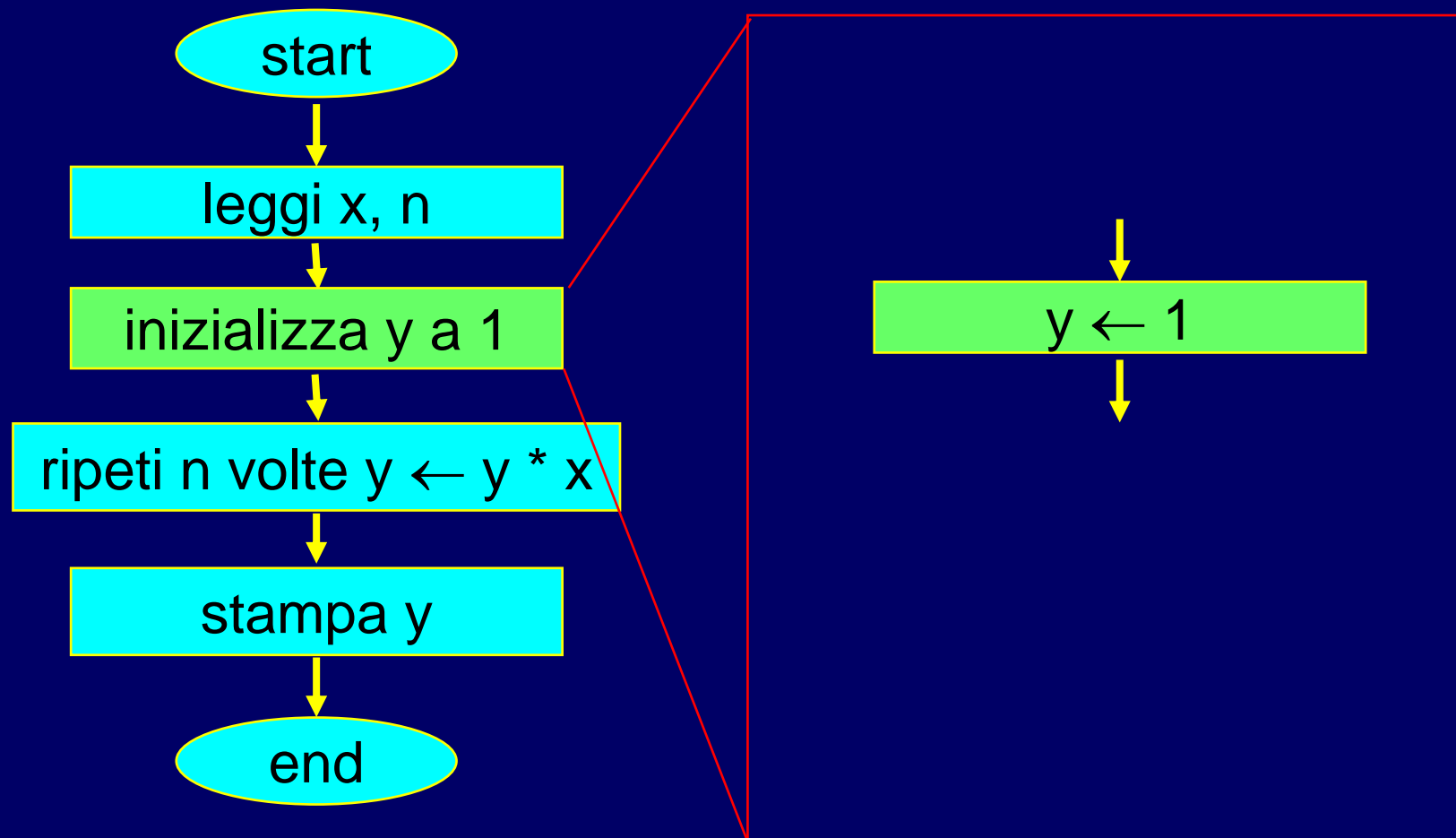
- Calcolo di $y = x^n$



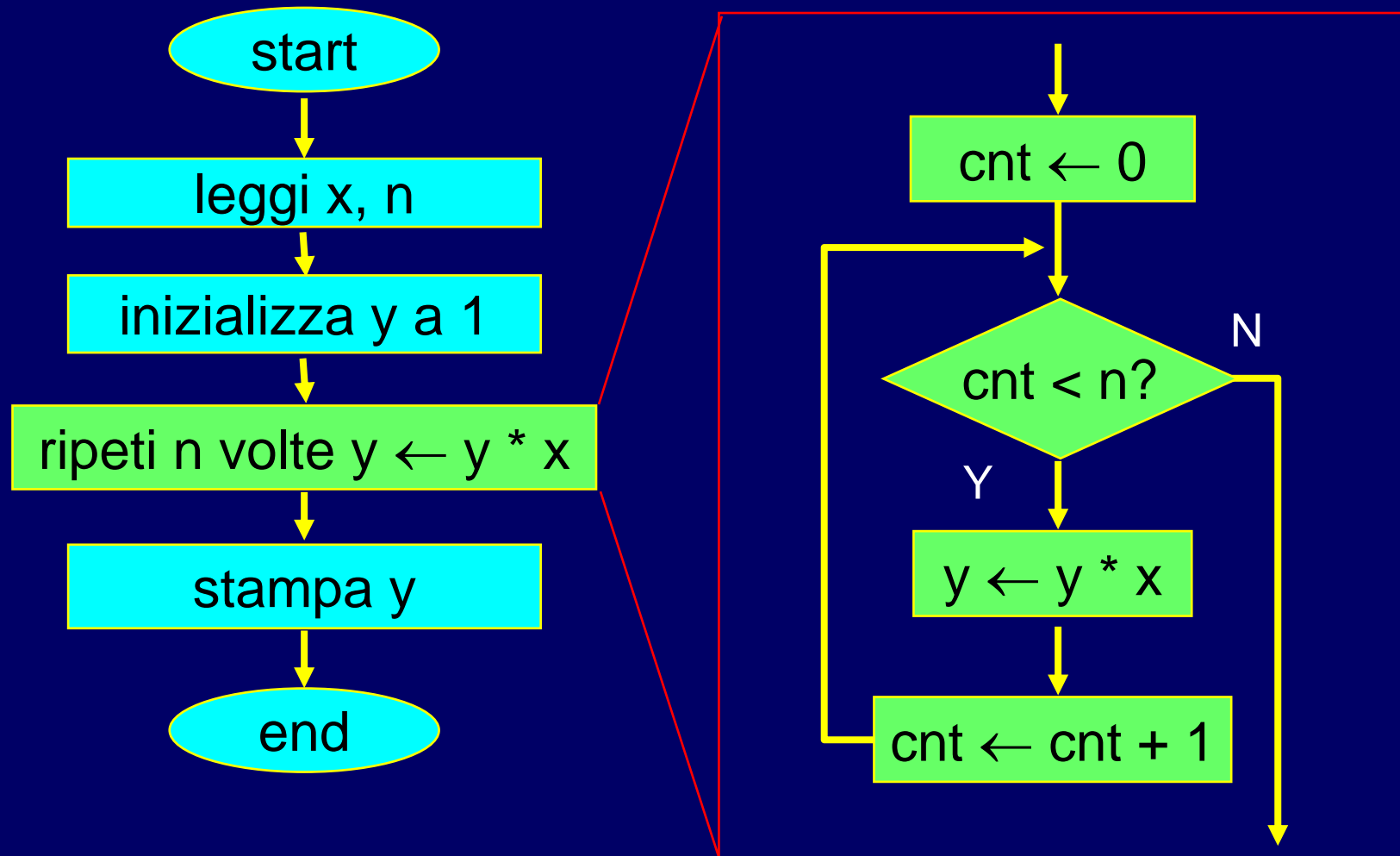
Esempio



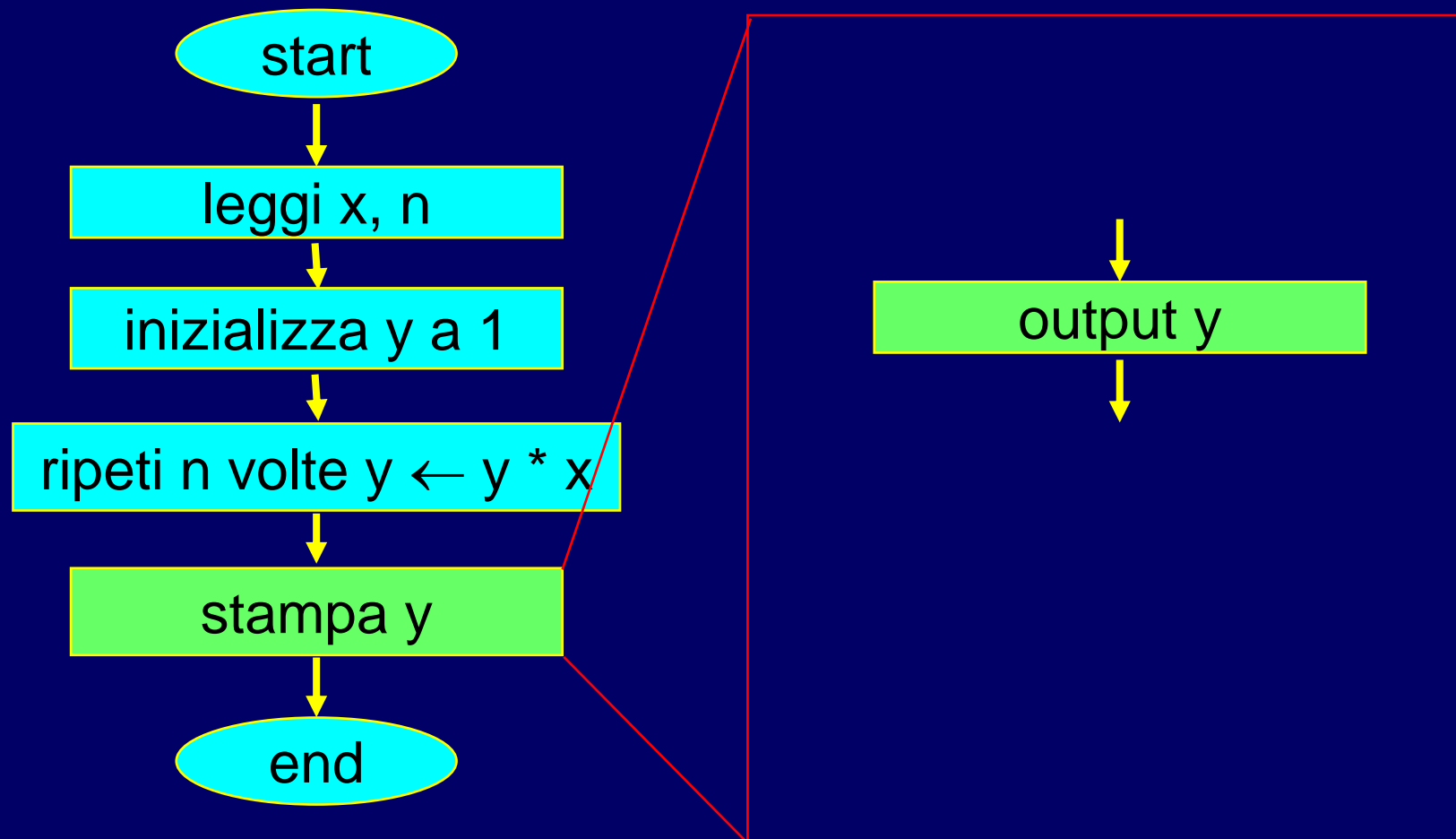
Esempio



Esempio



Esempio

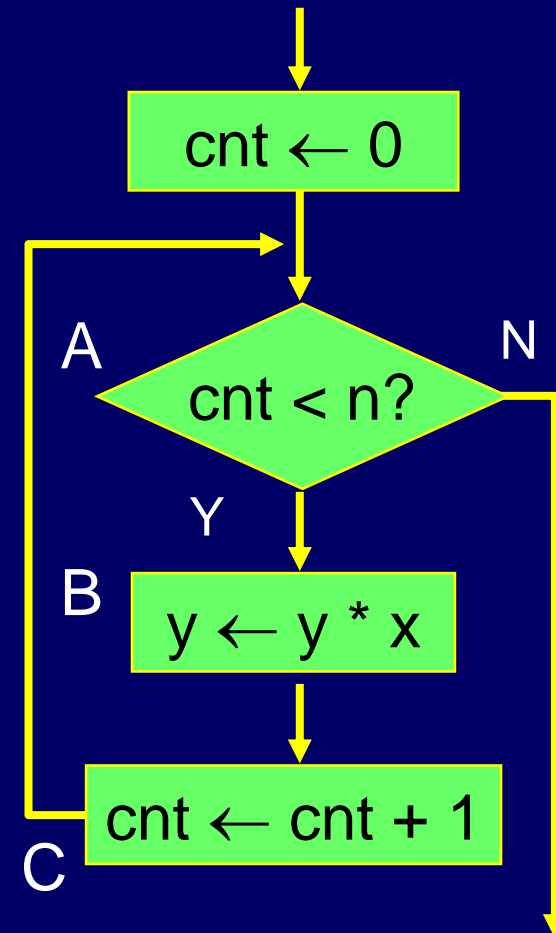


Funziona?

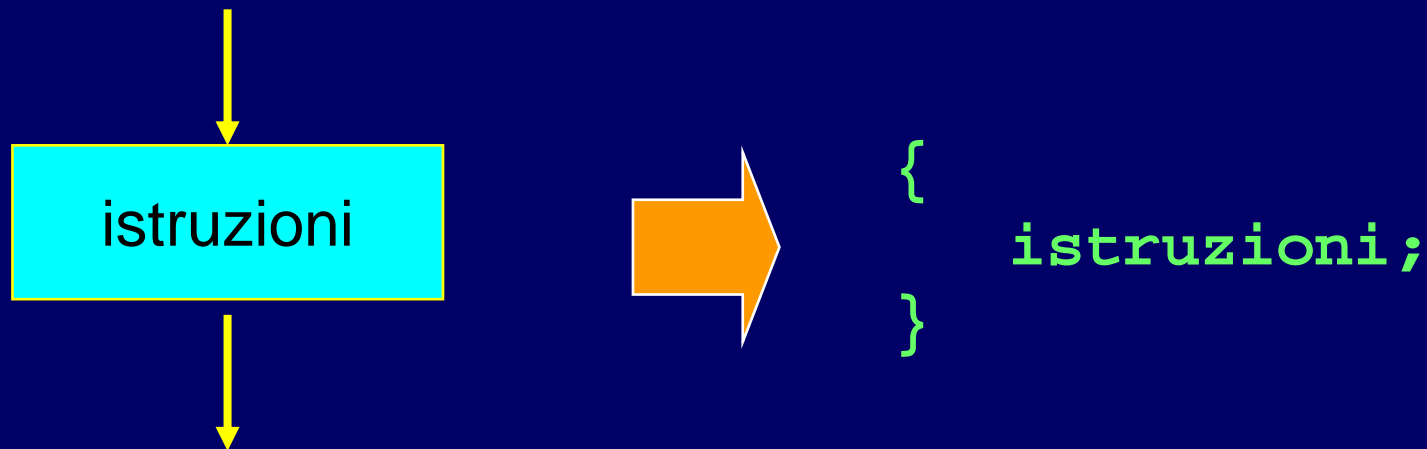
- Verifica mediante traccia della computazione
- Es. $x = 3$, $n = 2$

Funziona?

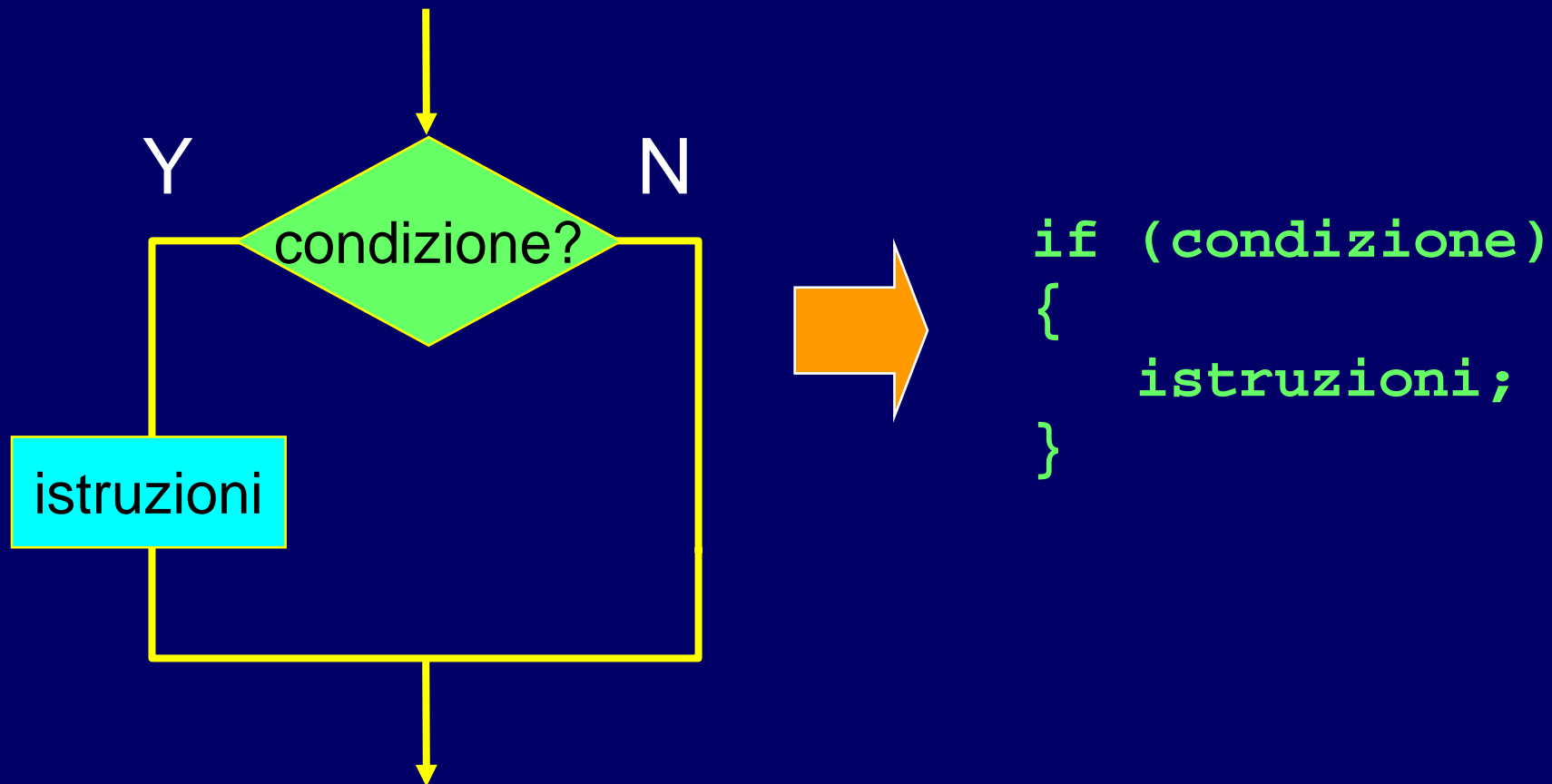
blocco	condizione	cnt	n	x	y
A	V	0	2	3	1
B	-	0	2	3	3
C	-	1	2	3	3
A	V	1	2	3	3
B	-	1	2	3	9
C	-	2	2	3	9
A	F	2	2	3	9



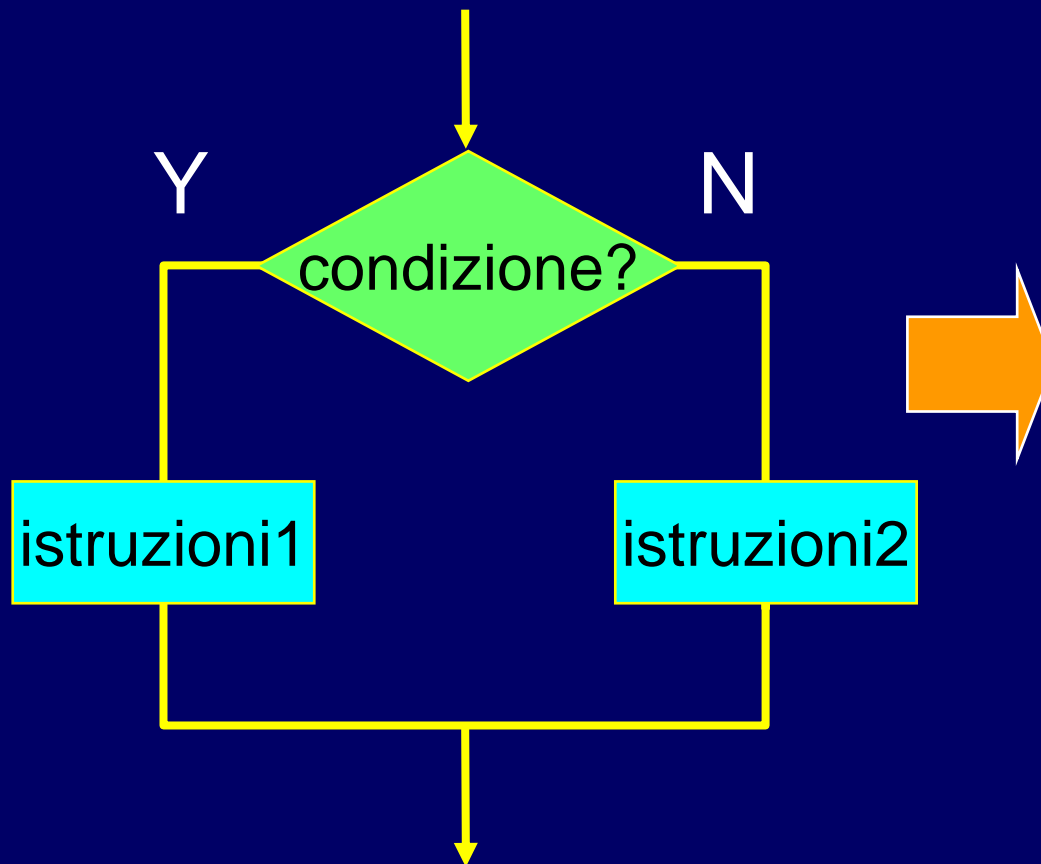
Dai diagrammi di flusso al C



Dai diagrammi di flusso al C

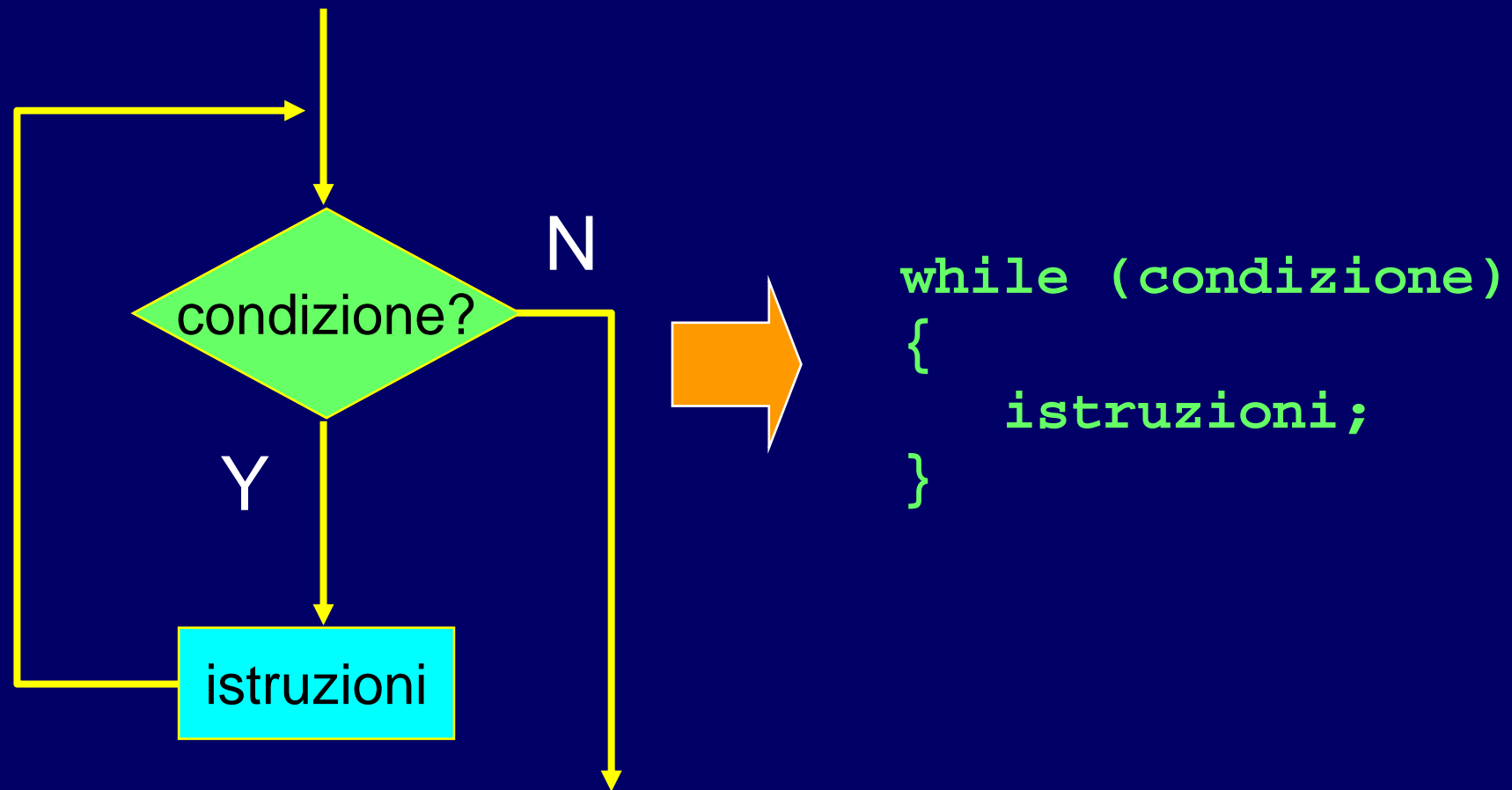


Dai diagrammi di flusso al C

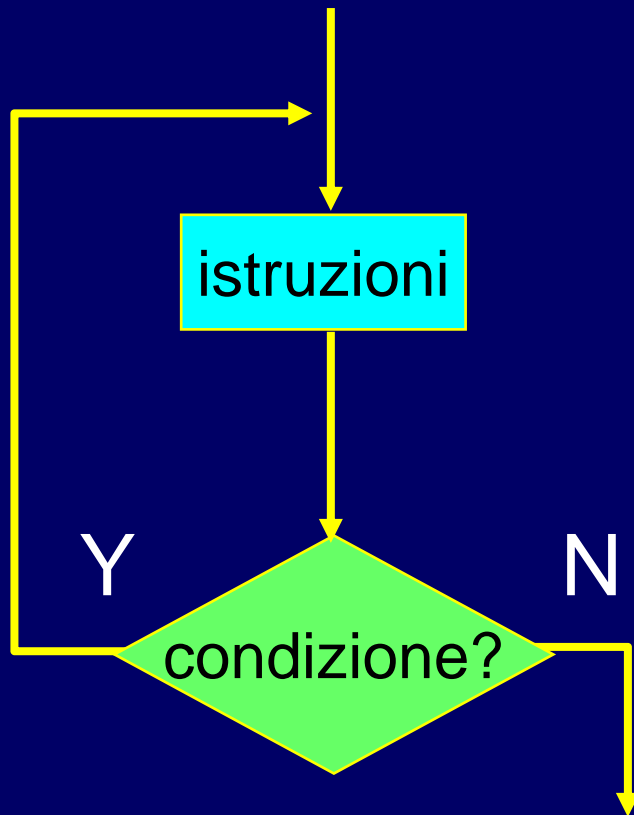


```
if (condizione)
{
    istruzioni1;
}
else
{
    istruzioni2;
}
```

Dai diagrammi di flusso al C



Dai diagrammi di flusso al C



```
do
{
    istruzioni;
} while (condizione);
```

Esempio

- Calcolo di
 $y = x^n$

```
input x;  
input n;  
  
y <- 1;  
cnt <- 0;  
while (cnt < n)  
{  
    y <- y * x;  
    cnt <- cnt + 1;  
}  
  
output y;
```