

E ora, finalmente...

Si comincia!

DD Cap.3 pp.49-74

## Sviluppo di programmi

- Per scrivere un programma C corretto bisogna:
  1. Analizzare il problema (input, output, casi estremali)
  2. Progettare una soluzione (algoritmo) tramite pseudocodice o diagramma di flusso (a blocchi)
  3. Accertarsi della correttezza del progetto
  4. Perseguire l'efficienza
  5. Scrivere il codice C (spezzando in blocchi ed usando i principi della "buona programmazione")
  6. Accertarsi della correttezza dell'esecuzione
- N.B. I primi 4 punti si eseguono senza il calcolatore!!!

## 1. Analizzare il problema

- E' necessario capire:
  - Quale sia l'**input**  
Cosa deve essere fornito dall'utente? In quale formato?  
Cosa è supposto essere noto?
  - Quale sia l'**output**  
Cosa ci si fa con i dati in input? Che problema dobbiamo risolvere?
  - Cosa accade nei **casi estremali**  
Spesso i casi estremali vanno trattati a parte

**Esempio:** soluz. di un'equaz. di 1° grado  
 $ax+b=0$

## 2. Progettare una soluzione (1)

### Pseudocodice p.50

- Linguaggio informale che aiuta a sviluppare algoritmi; fornisce una sequenza di azioni:
  - Sufficientemente ad alto livello da non entrare in dettagli implementativi
  - Sufficientemente dettagliata da non generare ambiguità
- Esempio:** ricetta della frittata (problema del livello di dettaglio)
- Ci aiuta a pensare prima di scrivere il programma, ma è facile da convertire in C
- **Attenzione:** l'ordine delle operazioni è fondamentale!

## Parentesi: flusso di controllo (1)

- che differenza c'è tra questi due frammenti di pseudocodice?

```
a=3          somma=a+b
b=5          b=5
somma=a+b   a=3
```

- Quando eseguiamo un programma le sue istruzioni vengono eseguite sequenzialmente a partire dalla prima. Quindi...

## Parentesi: flusso di controllo (2)

```
a=3
↓
b=5
↓
somma=a+b
```

	a	b	s
a=3	3	-	-
b=5	3	5	-
s=a+b	3	5	8

```
somma=a+b
↓
b=5
↓
a=3
```

	a	b	s
s=a+b	-	-	-
b=5	-	5	-
a=3	3	5	-

## 2. Progettare una soluzione (2)

### Diagrammi di flusso p.51

- Rappresentazione grafica disegnata usando simboli con significati speciali connessi tra loro da frecce, dette linee di flusso; fornisce una sequenza di azioni:
  - Sufficientemente ad alto livello da non entrare in dettagli implementativi
  - Sufficientemente dettagliata da non generare ambiguità
- Attenzione: l'ordine delle operazioni è fondamentale!
- I diagrammi di flusso "obbligano" ad utilizzare la [programmazione strutturata](#)...

## Programmazione strutturata p.51

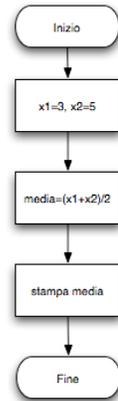
- Esecuzione sequenziale
  - istruzioni eseguite una dopo l'altra nell'ordine
- Trasferimento di controllo
  - l'istruzione successiva da eseguire non è la successiva nella sequenza
  - eccessivo uso -> problemi
- Teorema di Bohm e Jacopini
  - tutti i programmi si possono scrivere in termini di 3 strutture di controllo
    - struttura di sequenza (in C di default)
    - struttura di selezione (if, if...else, e switch)
    - struttura di iterazione (while, do...while and for)

## Il diagramma di flusso

- **Ovale:** inizio/fine (solo uscita/entrata)
- **Rettagolo:** simbolo di azione (una entrata ed una uscita)
- **Rombo:** simbolo di decisione (una entrata e due uscite)

**Esempio:** calcolare la media di 3 e 5

Assegna ad x1 il valore 3  
 Assegna ad x2 il valore 5  
 Calcola il valore di  $(x1+x2)/2$   
 Assegna tale valore a media  
 Stampa il valore di media



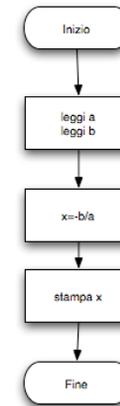
## 3. Accertarsi della correttezza(1)

**Problema:** Si leggano due interi  $a$  e  $b$ , e si risolva l'equazione  $ax+b=0$ . Sia  $b \neq 0$ .

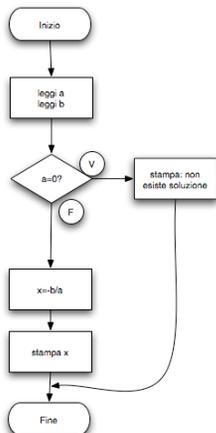
Corretto?

Se  $a=0$ ? Caso estremo da trattare a parte...

Introduciamo il concetto di **selezione**



## 3. Accertarsi della correttezza (2)



Leggi a  
 Leggi b  
 Se  $(a=0)$   
 allora  
 stampa: "non esiste soluzione"  
 altrimenti  
 $x=-b/a$   
 stampa x

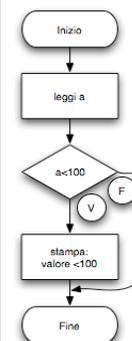
Ora il progetto è corretto!

**N.B.** l'indentazione!!!

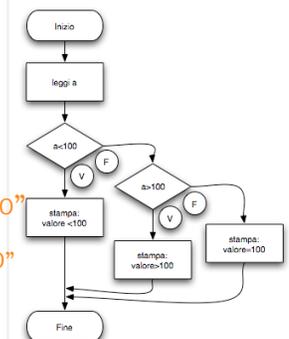
## Un altro esempio

**Problema:** Si legga un intero  $a$ , e si dica se esso è minore di 100.

**Problema:** Si legga un intero  $a$ , e si dica se esso è minore, maggiore o uguale a 100.



Leggi a  
 Se  $(a < 100)$   
 allora  
 stampa: "valore <100"  
 altrimenti  
 Se  $(a > 100)$   
 allora  
 stampa: "valore >100"  
 altrimenti  
 stampa: "valore=100"



## Esercizi (selezione)

Pseudocodice e diagramma a blocchi per la soluzione dei seguenti problemi:

- Si legga un intero e si stampi 1 se questo è dispari, e 0 se è pari.
- Dati in input 2 interi, si scriva se il primo sia  $<$ ,  $=$  o  $>$  del secondo.
- Si simuli il seguente gioco tra 2 giocatori: ogni giocatore sceglie un colore tra bianco, rosso, nero. Il bianco batte il rosso, il rosso batte il nero, il nero batte il bianco.

## La selezione in C p.53

In C la selezione tra diverse scelte si codifica tramite il comando **if**:

Se il voto non è inferiore a 18  
allora visualizza: Promosso

si scrive in C:

```
if (voto >= 18)
    printf("promosso");
```

Se c'è un'alternativa si usa **if... else**:

```
if (voto >= 18)
    printf("promosso");
else
    printf("bocciato");
```

Alternativamente: `(voto >= 18)?printf("promosso"):printf("bocciato");`  
e si parla di **operatore condizionale**.

## If...else nidificati

I comandi **if...else** possono essere messi in cascata:

```
if (voto >= 27)
    printf("ottimo");
else
    if (voto >= 23)
        printf("discreto");
    else
        if (voto >= 18)
            printf("sufficiente");
        else
            printf("insufficiente");
```

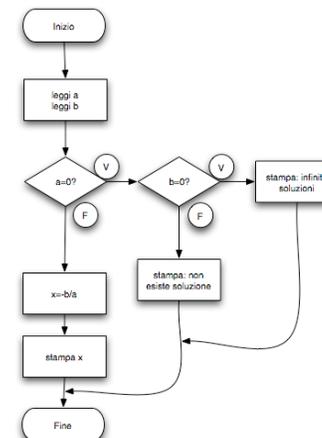
Attenzione all'indentazione!

## Esempio

**Problema:** Si leggano due interi  $a$  e  $b$ , e si risolva l'equazione  $ax+b=0$ .

- Attenzione a tutte le possibilità ('sta volta  $b$  può essere 0):
- $a \neq 0$ :  $x = -b/a$
  - $a = 0$  ma  $b \neq 0$ : non esistono sol.
  - $a = 0$  e  $b = 0$ : infinite sol.

```
Leggi a; Leggi b;
se (a=0) allora
    se (b=0) allora
        stampa: "non esiste soluzione"
    altrimenti
        stampa: "infinite soluzioni"
altrimenti
    x = -b/a
    stampa x
```

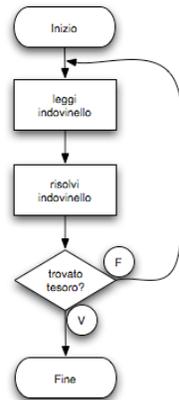


## Esempio: caccia al tesoro

**Problema:** dato un primo indovinello, trova i successivi finché non raggiungi il tesoro

Non sappiamo quanti siano gli indovinelli, ma quando avremo trovato il tesoro sapremo che avremo finito

Introduciamo il concetto di **ciclo**

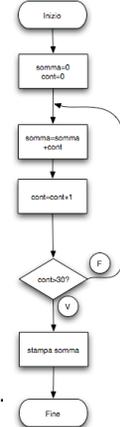


## Esempio: somma dei primi 5 interi

E se ne dobbiamo sommare 30?

Somma=0; cont=0;  
 Finché cont non è 30  
 Somma=Somma+cont  
 cont=cont+1  
 (Eseguite le istr. con cont=30)  
 Stampa Somma

Correttezza ok.  
 Efficienza?

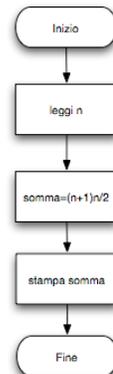


## 4. Perseguire l'efficienza

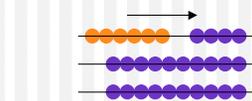
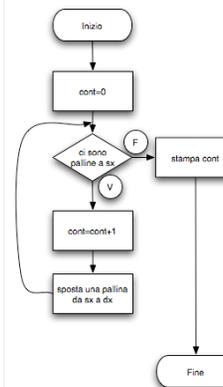
$$1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 =$$

In generale:  $1+2+\dots+n=(n+1)n/2$ .

Leggi n  
 Somma=(n+1)\*n/2  
 Stampa n



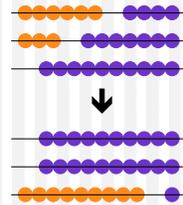
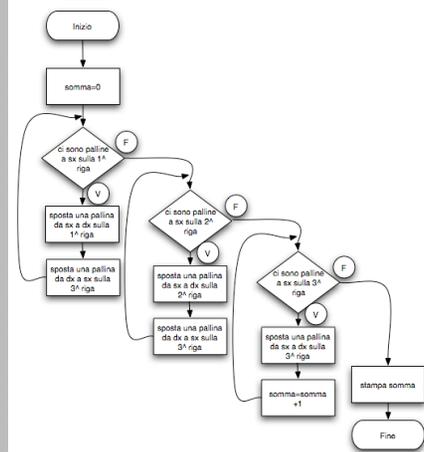
## Esempio: contare col pallottoliere



Un altro esempio di ciclo

cont=0  
 Finché ci sono palline a sx  
 cont=cont+1  
 sposta una pallina da sx a dx  
 (non ci sono più palline a sx)  
 stampa cont

## Esempio: sommare col pallottoliere



Cicli concatenati

## Esercizi (cicli)

Pseudocodice e diagramma di flusso:

- Si prenda in input un intero  $n$  e si dia in output il valore di  $n!$
- Si prenda in input un intero  $n$  e si stampino in output i primi  $n$  interi all'indietro (da  $n$  ad 1)
- Si calcoli la somma di 30 numeri interi inseriti dall'utente
- Si prenda in input un intero  $n$  e si calcoli la somma di  $n$  numeri interi inseriti dall'utente

## I cicli in C p.58

I cicli in C si possono codificare in diversi modi.

Uno di essi è il comando **while**:

Finché cont non supera 30

aggiungi cont a somma

si esprime in C come:

```
while (cont <=30)
    somma=somma+cont;
```

Finché ci sono palline a sx

cont=cont+1

sposta una pallina da sx a dx

si esprime in C come:

```
while (sx>0)
{
    cont=cont+1;
    sx=sx-1;
    dx=dx+1;
}
```

In C le istruzioni multiple sono racchiuse tra `{ }`