STRUTTURE DI CONTROLLO

Ing. Daniele Corti

© copyright
all rights reserved

Copyright © Ing. Daniele Corti 2013

www.ingdanielecorti.it

Tutti i diritti sono riservati a norma di legge e a norma delle convenzioni internazionali.

Ver.2.0 – 02-09-2020

PREREQUISITI

- ✓ Algoritmi e loro rappresentazioni.
- ✓ Algebra di Boole e operazioni logiche.

OBIETTIVI

- ✓ Saper utilizzare la sequenza e la selezione nella risoluzione dei problemi.

ARGOMENTI

- ✓ La programmazione strutturata.
- ✓ Teorema di Böhm-Jacopini.
- ✓ La sequenza.
- ✓ La selezione binaria.

CAP 4 – STRUTTURE DI CONTROLLO

LA PROGRAMMAZIONE STRUTTURATA

La programmazione strutturata è un paradigma (metodologia) di programmazione nata nella seconda metà degli anni ’60 del XX secolo. I concetti di base di questa metodologia si possono ritrovare in tutti gli altri paradigmi successivi, compresi i linguaggi di nuova concezione orientati agli oggetti.

Un **paradigma di programmazione** è uno stile fondamentale di programmazione, ovvero un insieme di strumenti concettuali forniti da un linguaggio di programmazione per la stesura dei programmi.

Questa tecnica è stata introdotta per superare i limiti della programmazione basata sul **salto incondizionato** (o **GOTO**) alla base dei primi linguaggi di programmazione. Il codice di questi primi linguaggi era addirittura incomprensibile da parte dell’autore stesso a tal punto che fu chiamato **spaghetti code**, per indicare la sua natura ingarbugliata.

STRUTTURE DI CONTROLLO DI BASE

In un algoritmo una struttura di controllo è un gruppo di istruzioni con un unico punto d’ingresso e un unico punto d’uscita. Le strutture di controllo servono a specificare come si deve comportare il programma, cioè permettono il controllo dell’esecuzione dell’algoritmo. Un programma, infatti, non è detto che debba essere una sequenza di istruzioni da eseguire una dopo l’altra. Può capitare di dove prendere delle decisioni che fanno cambiare il corso dell’esecuzione.

La programmazione strutturata è una tecnica di programmazione che permette la costruzione di algoritmi basandosi solamente sulle seguenti tre strutture di controllo:

- **Struttura di Sequenza**.
- **Struttura di Selezione (o alternativa)**.
- **Struttura di Iterazione (o ciclo, ripetizione)**.

Con questa tecnica di programmazione, la descrizione degli algoritmi risulta essere più chiara, facile, leggibile e comprensibile. Tale codice può, anche, essere compreso agevolmente e modificato in un secondo tempo, anche da un altro programmatore che non sia il suo creatore.
Fra i linguaggi tipici del paradigma strutturato possiamo citare il Pascal, il C, il Cobol e l'Algol.

TEOREMA DI BÖHM-JACOPINI

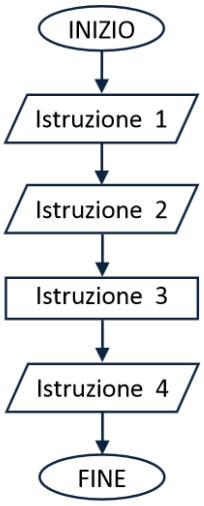
I linguaggi di programmazione strutturati iniziarono a emergere all'inizio degli anni '70 del XX secolo, grazie alle brillanti idee di due matematici italiani, Corrado Böhm e Giuseppe Jacopini. Essi giunsero alla seguente tesi, chiamata **teorema di Böhm-Jacopini**:

Un qualsiasi algoritmo può essere espresso in un linguaggio di programmazione che disponga soltanto delle tre strutture di controllo: sequenza, selezione e iterazione.

Questi sono, quindi, i mattoni di base con i quali è possibile costruire qualsiasi algoritmo.

SEQUENZA

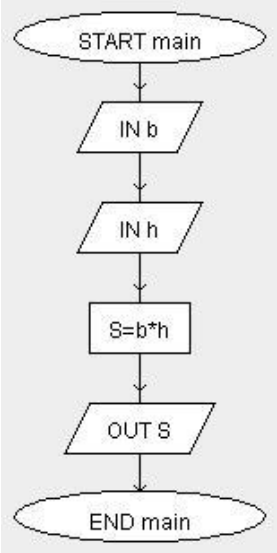
Come suggerisce il nome, una **sequenza** è un elenco di istruzioni che vengono eseguite una dopo l'altra, in successione.

Struttura di sequenza	
Flow-chart	**Pseudolinguaggio**
	INIZIO Istruzione 1 Istruzione 2 Istruzione 3 Istruzione 4 FINE

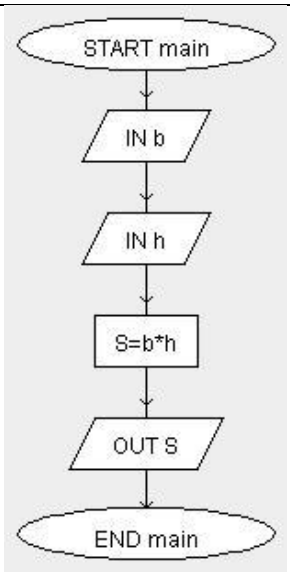
Gli algoritmi più semplici (come quello mostrato qui sopra) sono costituiti da un'unica sequenza di istruzioni.

ESEMPIO sequenza

Dati la misura della base **b** e dell'altezza **h** di un rettangolo, calcolarne l'area **S**.

Flow-chart	**Pseudolinguaggio**	**Num**	**Tabella di traccia** (b=5; h=3)			
			b	**h**	**S**	**OUT**
	PROG main	1				
	IN b	2	5			
	IN h	3		3		
	ASSIGN S=b*h	4			15	
	OUT S	5				15
	END PROG	6				

Un esempio di sequenza è quella mostrata qui sotto e che serve per calcolare area e perimetro di un cerchio data la misura del raggio:




```
PROG main
   IN raggio
   ASSIGN Area=PI*raggio*raggio
   ASSIGN Perimetro=2*PI*raggio
   OUT Area
   OUT Perimetro
END PROG //main
```

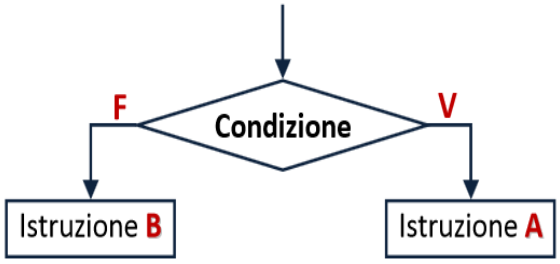
Notiamo di passaggio che la moltiplicazione è stata indicata con l'asterisco *, usando la convenzione usata nella maggior parte dei linguaggi di programmazione. Allo stesso modo è stato utilizzato il punto decimale, invece della virgola, nella scrittura di pi greco (3.14), perché questa è la convenzione che viene di solito usata nei linguaggi di programmazione.

SELEZIONE BINARIA

In un algoritmo si può verificare che le istruzioni da eseguire siano diverse a seconda dei dati elaborati. L'esecutore deve operare una scelta fra due possibili alternative.

La selezione binari o alternativa consente di fare la scelta tra due possibili alternative. Per effettuare tale scelta l'esecutore controlla il grado di verità di una **condizione** posta in un test. Se la condizione risulta **vera** l'esecutore esegue l'istruzione A, altrimenti se è **falsa** esegue l'istruzione B. Naturalmente, sia nel blocco vero che nel blocco falso possono essere eseguite una o più istruzione. Nel seguente schema, per semplicità abbiamo indicato una sola istruzione.

Struttura di selezione	

	SE condizione ALLORA
	Istruzione A
	ALTRIMENTI
	Istruzione B
	
	Oppure
	
	IF condizione THEN
	Istruzione A
	ELSE
	Istruzione B

La **selezione** corrisponde in italiano all'istruzione SE... ALLORA. Una selezione si dice *semplice* o *a una via* quando prevede l'esecuzione di un solo blocco di istruzioni nel caso in cui la condizione sia vera. Nel caso di condizione falsa, la selezione semplice non esegue nessuna istruzione.

```
SE condizione
ALLORA
        AZIONI_1
FINE SE
```

Un esempio di selezione semplice è:

SE piove ALLORA prendi l'ombrello

Se la condizione è falsa (cioè *se NON piove*), non viene prevista nessuna istruzione.

In una selezione *a due vie*, invece, viene eseguito qualcosa sia nel caso in cui la condizione sia vera sia nel caso in cui la condizione sia falsa. La selezione a due vie viene tradotta in italiano con SE... ALLORA.... ALTRIMENTI....

SE condizione
ALLORA
    AZIONI_1
ALTRIMENTI
    AZIONI_2
FINE SE

Un esempio di selezione a due vie è:

SE il semaforo è rosso ALLORA *passa* ALTRIMENTI *fermati*

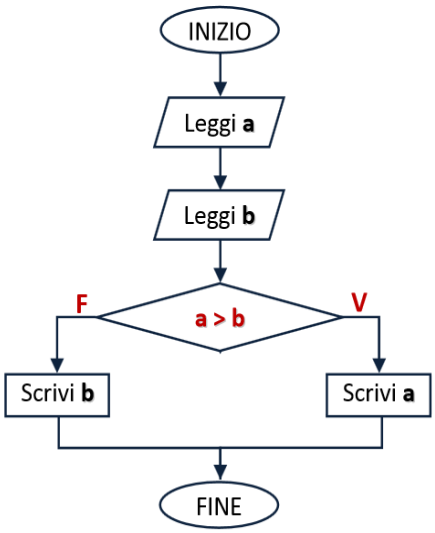
In questo caso l'istruzione passa viene eseguita se la condizione è vera (cioè *SE il semaforo è rosso*), mentre l'istruzione fermati viene eseguita se la condizione è falsa.

ESEMPIO struttura di selezione

Dati due numeri **a** e **b,** calcolare il maggiore.

Max fra due numeri

Flow-chart



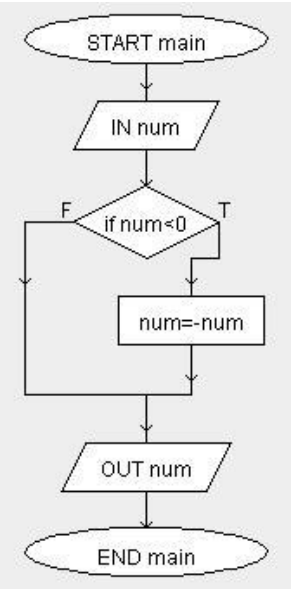
Pseudolinguaggio

n°	Istruzione
1	INIZIO
2	Leggi il numero *a*
3	Leggi il numero *b*
4	*Se* a > b
5	*allora* Scrivi *a*
6	*altrimenti* Scrivi *b*
7	FINE

Trace table (Input: a=3 ; b=5)

n°	a > b	a	b
1	INIZIO		
2		3	
3			5
4	Falso		
5			
6			5
7	FINE		

ESEMPIO Acquisire un numero e renderlo positivo se è negativo.

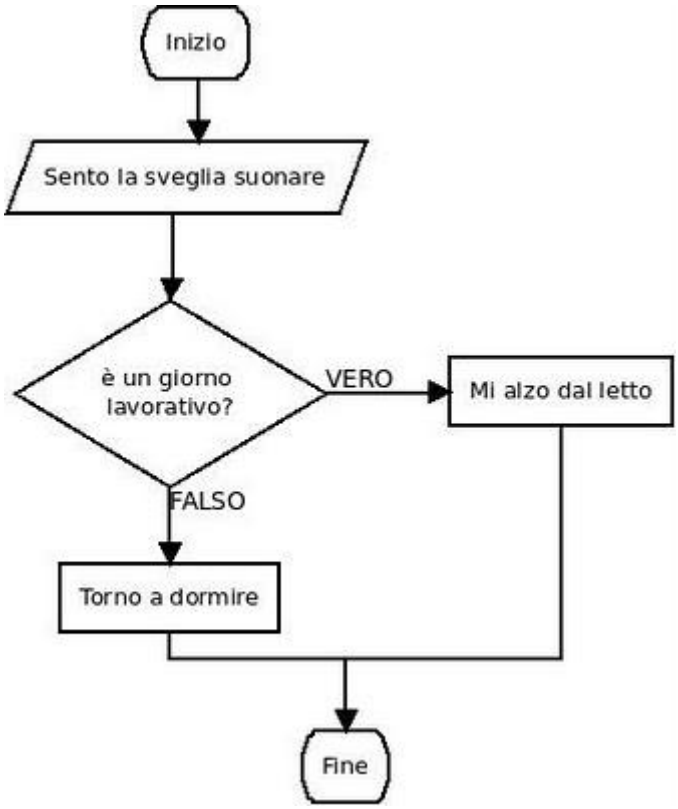


Si osservi il metodo usato per rendere positivo il numero (se negativo):

num = - num

ESEMPI

ESEMPIO: L'ESECUTORE È L'ESSERE UMANO.



Che molto semplicemente descrive la situazione in cui alla mattina suona la sveglia e bisogna capire se alzarsi dal letto oppure no.

Esercizio 1

Dato in input un numero, calcolarne il cubo se è maggiore di 0, il quadrato in caso contrario.

Esercizio 2

Calcolare l'area di un rettangolo e quella di un quadrato

Esercizio3. Buon giorno.

Descrivi le azioni che vengono eseguite al mattino dal suono della sveglia fino all'uscita di casa.

Esercizio4. Secchi d'acqua.

Sono presenti due secchi con capacità rispettivamente di 4 e 3 litri. Si riempia il primo secchio (quello da 4 litri) con soli 2 litri d'acqua. Descrivi con un algoritmo la soluzione.

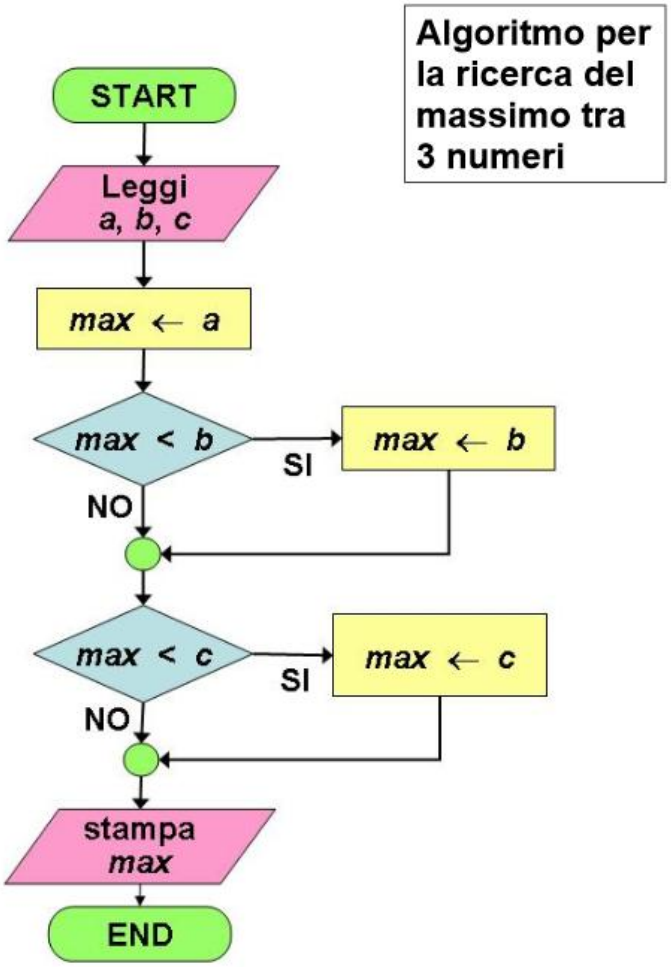
Esercizio 5. Pizza Margherita.

Si descriva la procedura per la preparazione di una pizza margherita.

Dopo essersi procurati la ricetta della pizza ci si procura tutto il necessario, quindi si preparano la pasta e gli ingredienti. Si accende il forno (fuoco), si stende la pasta e la si farcisce con pomodoro e mozzarella. Quando il forno è alla temperatura giusta, si inforna la pizza e si attende la cottura. Quando la pizza è cotta, si estrae dal forno, si taglia e si mangia!

ESEMPIO

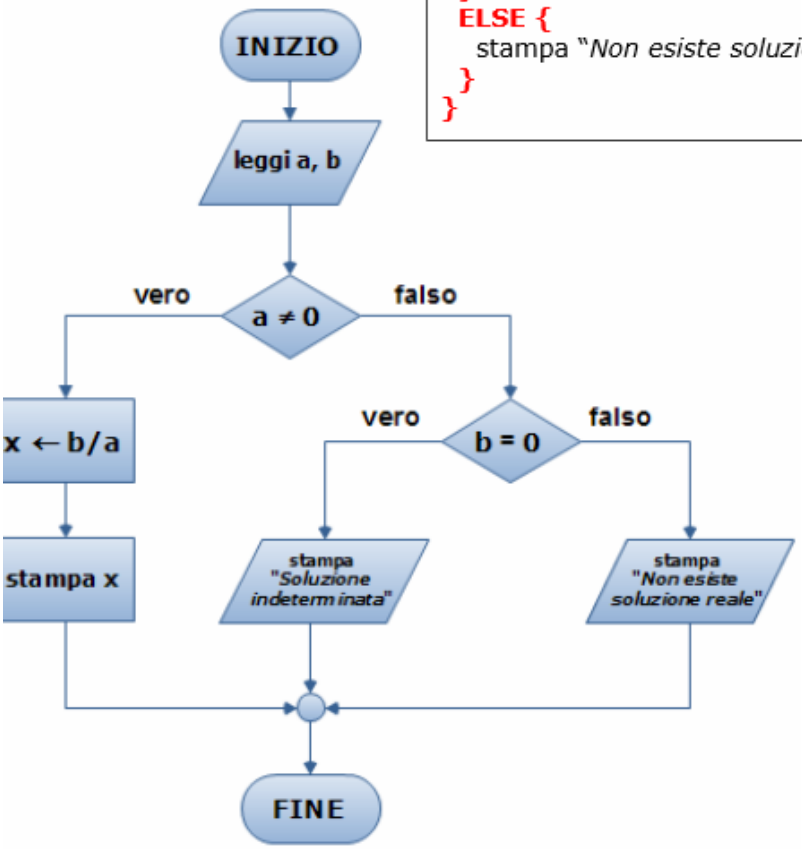
Si scriva un algoritmo per la determinazione del massimo fra tre numeri interi.



Esercizio:

Scrivere lo pseudo-codice e rappresentarlo con un flow chart dell'algoritmo per la risoluzione di un'equazione di primo grado.

```
leggi  a, b
IF (a ≠ 0) {
   x = b/a
   stampa x
}
ELSE {
  IF (b = 0) {
    stampa "Soluzione indeterminata"
  }
  ELSE {
    stampa "Non esiste soluzione"
  }
}
```



13

ESEMPI

PROBLEMA 0 – es pilota

Noto il prezzo pieno del biglietto (per un evento culturale), si vuole calcolare e visualizzare a video il prezzo del biglietto a seconda dell'età del cliente, secondo la seguente tabella.

Età	Sconto
<=6	100%
>60	30%

DATI IN

PrezzoPieno, Età

DATI OUT

PrezzoScontato

PROCEDIMENTO

1. Acquisisco da tastiera il prezzo pieno e lo salvo in PrezzoPieno
2. Acquisisco da tastiera l'età del cliente e lo salvo in Età.
3. Verifico l'età al fine di applicare lo sconto corretto e calcolo il prezzo.
4. Stampo a video il prezzo.

Dettagli passo 3 del Procedimento:

a. Se l'età è minore o uguale a 6 lo sconto è del 100%;
b. Altrimenti se l'età è maggiore di 60 lo sconto è del 30%;
c. Altrimenti non c'è nessuno sconto.

In tutti questi tre casi dovrò calcolare il relativo sconto.

INIZIO
INPUT PREZZOPIENO
INPUT ETA
SE ETA<=6 **ALLORA**
    PREZZO=0
ALTRIMENTI SE ETA>60

PREZZO=PREZZOPIENO –
PREZZOPIENO*30/100

ALTRIMENTI

PREZZO=PREZZOPIENO

OUTPUT PREZZO

FINE

NB Le seguenti due istruzioni

INPUT A

INPUT B

se eseguite una dopo l'altra, possono essere sostituite da quest'unica istruzione

INPUT A, B

In questo modo il codice risulta, quindi, più leggero.

ES: Le seguenti operazioni vengono eseguite una dopo l'altra; analizzare, passo per passo, i valori che assumono le variabili.

Istruzione	A	B
A ← 4		
B ← 10		
A ← B – A		
B ← B – A		
A ← A*A		

PROBLEMA 1 (difficoltà max)

1. COSTRUIRE UN ALGORITMO CHE CONSENTE DI INDOVINARE UNA PASSWORD.

DATI IN: PASSWORD
DATI INTERNI: PW
PROCEDURA:
INIZIO
PW←"carlo" (assegno a PW il codice da indovinare)
INPUT PASSWORD
SE PW=PASSWORD ALLORA
    OUTPUT "indovinata!"
ALTRIMENTI
    OUTPUT "ritenta"
FINE SE
FINE

2. COSTRUIRE UN ALGORITMO CHE CONSENTE DI INDOVINARE UNA PASSWORD CON UN MAX DI TRE TENTATIVI POSSIBILI.

PROCEDURA "INDOVINA PASSWORD - VER1"
INIZIO
    DATI IN: PW
    DATI INTERNI: PWX
ESEGUI
    PWX←"carlo" (assegno a PW il codice da indovinare)
    INPUT PW
    SE PW=PWX
    ALLORA
        OUTPUT "indovinata!"
    ALTRIMENTI
        OUTPUT "ritenta per la 2a volta"
        INPUT PW
        SE PW=PWX

```
                ALLORA
                        OUTPUT “indovinata!”
                ALTRIMENTI
                        OUTPUT “ritenta per la 3a volta”
                        INPUT PW
                        SE PW=PWX
                        ALLORA
                                OUTPUT “indovinata!”
                        ALTRIMENTI
                                OUTPUT “tentativi esauriti”
                        FINE SE
                FINE SE
        FINE SE
FINE PROCEDURA
```

PROCEDURA “INDOVINA PASSWORD – VER2”

```
INIZIO
        DATI IN: PW
        DATI INTERNI: CONTA, PWX, INDOVINATA
ESEGUI
        CONTA=0
        PWX=”PIPPO”
        INDOVINATA=NO
        RIPETI
                IN(PW)
                SE(PW=PWX)
                ALLORA
                        OUT(“PUOI ACCEDERE  AL SISTEMA”)
                        INDOVINATA=SI
                ALTRIMENTI
                        OUT(“PW SBAGLIATA RITENTA”)
                FINE SE
                CONTA=CONTA+1
```

```
    FINCHE'(CONTA>=3) 'finché la condizione è false rimango nel ciclo
    SE(INDOVINATA=SI)
    ALLORA
        OUT("PUOI ACCEDERE")
    ALTRIMENTI
        OUT("ACCESSO IMPOSSIBILE")
    FINE SE
FINE PROCEDURA
```