

Introduzione alla programmazione

Algoritmi e diagrammi di flusso



F. Corno, A. Lioy, M. Rebaudengo

Sviluppo del software

- **problema**
- **idea (soluzione)**
- **algoritmo (soluzione formale)**
- **programma (traduzione dell'algoritmo in una forma comprensibile da un elaboratore elettronico)**
- **test**
- **documentazione**

Algoritmi

Per *algoritmo* si intende il meccanismo di soluzione di un problema.

Stabilisce la sequenza di operazioni da effettuare per portare a termine un determinato compito.

Algoritmi

Risolvere un problema significa individuare un procedimento (*algoritmo*) che, eseguito, fornisce informazioni finali (*risultati*) dipendenti da informazioni iniziali (*dati*).

Algoritmi

Il procedimento risolutivo è costituito da un insieme di passi (*istruzioni*), ognuno dei quali specifica una operazione elementare.

Algoritmi

Affinchè un procedimento possa definirsi algoritmo occorre che sia non ambiguo, eseguibile e finito cioè progettato in modo tale che la sua esecuzione richieda un tempo finito.

Algoritmi

L'esecuzione di un algoritmo si traduce in una successione di operazioni che vengono effettuate nel tempo.

L'esecuzione di un algoritmo evoca un *processo sequenziale* cioè una serie di eventi che occorrono uno dopo l'altro, ognuno con un inizio ed una fine ben identificabile.

Algoritmo

In genere comprende 3 fasi:

- **caricamento dei dati (input)**
- **elaborazione**
- **visualizzazione dei dati (output).**

Esecuzione di un algoritmo

Vengono eseguite in sequenza le operazioni che lo costituiscono.

Esistono algoritmi che prevedono:

- **una sequenza di esecuzione unica**
- **sequenze di esecuzione multiple**

Esempio: sequenza di esecuzione unica

Dato il valore di X, calcolare: $Y = 5 X + 3$

Sequenza di esecuzione:

- 1. ricevo il valore di X**
- 2. multiplico X per 5 (sia Z il risultato)**
- 3. sommo 3 a Z (sia Y il risultato)**
- 4. visualizzo Y**

Esempio: sequenze di esecuzione multiple

Dato il valore di X, calcolare la radice quadrata di X+5.

Sequenza di esecuzione:

- 1. ricevo il valore di X**
- 2. sommo 5 a X (sia Y il risultato)**
- 3a. se Y è positivo o nullo, calcolo la sua radice quadrata e la visualizzo**
- 3b. se Y è negativo, indico che è impossibile calcolare la sua radice quadrata**

Programma

Sequenza di operazioni svolte da un elaboratore per risolvere un problema.

Il programma corrisponde alla realizzazione software di un algoritmo in un certo linguaggio di programmazione.

Programma = Dati + Istruzioni

Programma: sequenza di operazioni svolte da un elaboratore per risolvere un problema.

Dati: la descrizione dei dati e' effettuata attraverso *dichiarazioni* e *definizioni*.

Istruzioni: le operazioni effettuate in un programma elaborano sostanzialmente dei dati.

Dati

I dati vengono gestiti attraverso:

- **Costanti**
- **Variabili scalari**
- **Vettori**
- **Matrici.**

Variabili scalari

Entita' con *memoria* in grado di contenere i *dati* (cioe' i valori) di un determinato *tipo*.

Ad ogni variabile e' associato un *nome* unico (*identificatore*).

Assegnazione

Operazione che rappresenta l'attribuzione di un valore (*dato*) ad una variabile.

Il valore assegnato puo' essere:

- Letto da terminale
- Una costante numerica (ad es. $\text{NUM} \leftarrow 1$)
- Il contenuto di un'altra variabile: (ad es. $\text{NUM2} \leftarrow \text{NUM1}$)
- Il risultato di un'espressione aritmetica o logica tra costanti e/o variabili (ad es. $\text{A} \leftarrow \text{B} * 2 - 5$).

Assegnazioni

- Il simbolo di assegnazione corrisponde ad un $=$, ma non indica ne' una equazione ne' una identita'. Indica la copia di un valore in una variabile.
- La variabile che compare nella parte sinistra di un'assegnazione puo' comparire anche nella parte destra (ad es. $NUM \leftarrow NUM+1$). Cio' significa che, in base al valore attuale, viene calcolato un nuovo valore ed assegnato alla variabile.

Vettori

Sono variabili che consentono di aggregare sotto un unico identificatore, insieme di valori, che possono essere usati singolarmente mediante un indice.

I vettori possono essere utilizzati come le variabili singole (scalari), specificando la cella (o posizione o elemento del vettore) mediante un *indice* secondo la notazione seguente

identificatore[indice]

Matrici

Le matrici sono una generalizzazione dei vettori e sono caratterizzati da avere dimensione maggiore di 1.

Esempio:

MATRICE [indiceriga] [indicecolonna]

CUBO [x] [y] [z]

Esempio

Calcolare l'importo di una fattura:

- **Cerca la corretta aliquota IVA sulla tabella**
- **Moltiplica l'importo netto per l'aliquota trovata**
- **Somma il risultato all'importo netto.**

Esempio

Specificare un algoritmo per effettuare il prodotto di due numeri interi col metodo delle addizioni successive, cioè $X * Y = X + X + X + X + \dots + X$.

Esempio

Specificare un algoritmo per effettuare il prodotto di due numeri interi col metodo delle addizioni successive, cioè $X * Y = X + X + X + X + \dots + X$.

Si chiami X il valore del moltiplicando ed Y il valore del moltiplicatore e sia M il risultato.

- **Si inizializza il valore di M a 0.**
- **Si ripetono le seguenti operazioni fintanto che Y è diverso da 0:**
 - **Si sommi il valore di X al valore M e si chiami il risultato ancora M ($M = M + X$).**
 - **Si sottragga 1 dal valore di Y, e si chiami Y ancora il risultato ($Y = Y - 1$).**
- **Sia M il risultato del prodotto.**

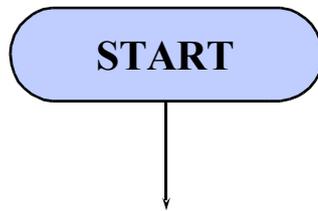
Formalizzazione di una soluzione

- **diagrammi di flusso**
- **alberi algoritmici**
- **pseudo-linguaggio**
- **...**

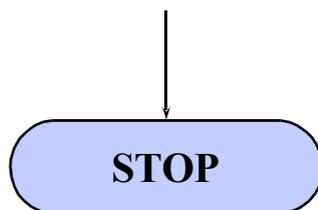
Diagrammi di flusso (flow-chart)

- **metodo per descrivere in modo formale un algoritmo**
- **blocchi base per descrivere azioni e decisioni (solo binarie)**
- **archi orientati per descrivere la sequenza di svolgimento delle azioni**

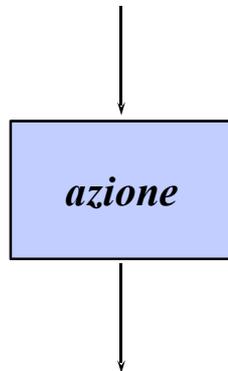
Blocco di inizio



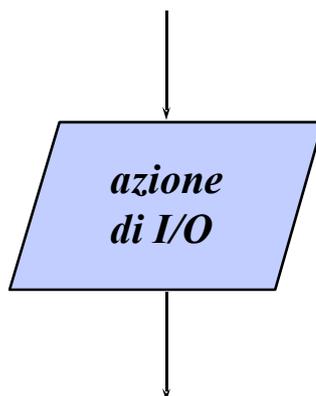
Blocco di fine



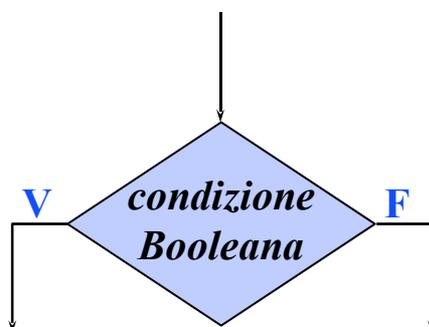
Blocco di azione



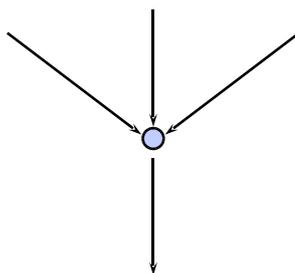
Blocco di Input/Output



Blocco di decisione binaria



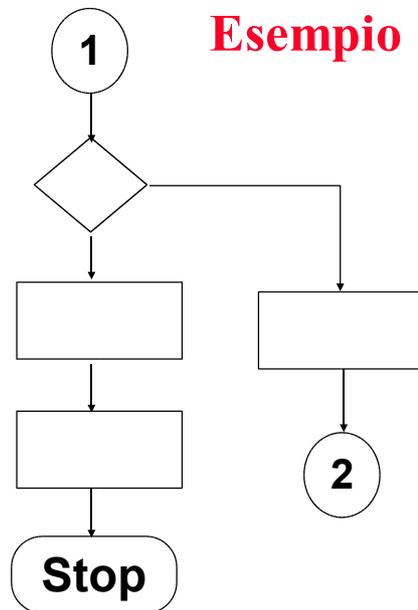
Connettore



Regole

- **uno ed un solo blocco START**
- **uno ed un solo blocco STOP**
- **tutti gli archi devono avere origine e fine in un blocco**

Esempio



Esempio

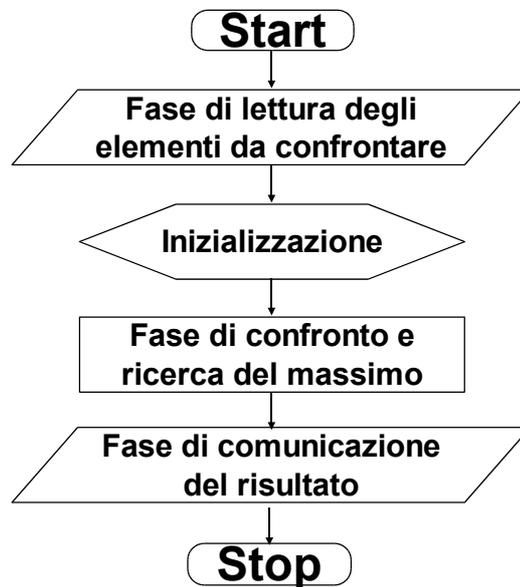
Rappresentare con un diagramma di flusso l'algoritmo che ricerca il massimo tra quattro numeri.

Si utilizzano le seguenti variabili:

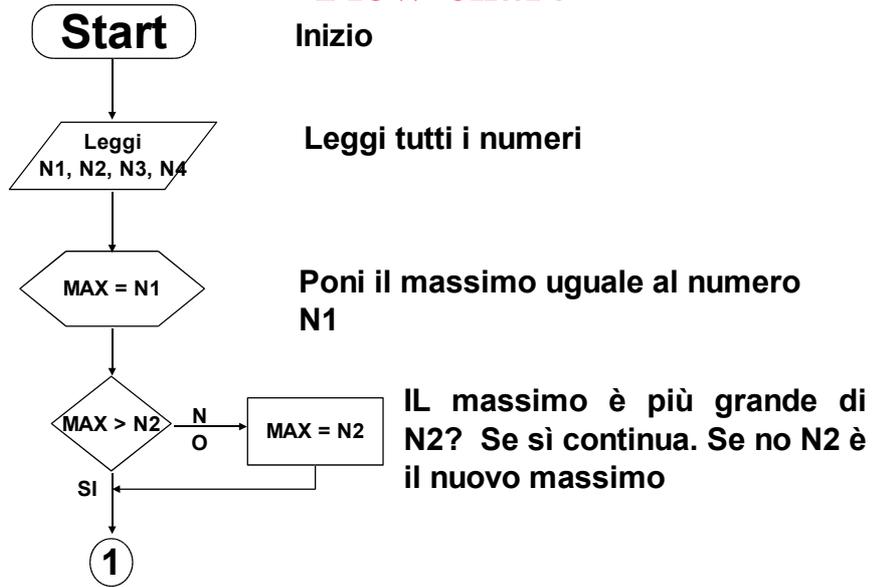
MAX per memorizzare il numero massimo

N1, N2, N3 e N4 per memorizzare i 4 dati del problema.

Flow Chart

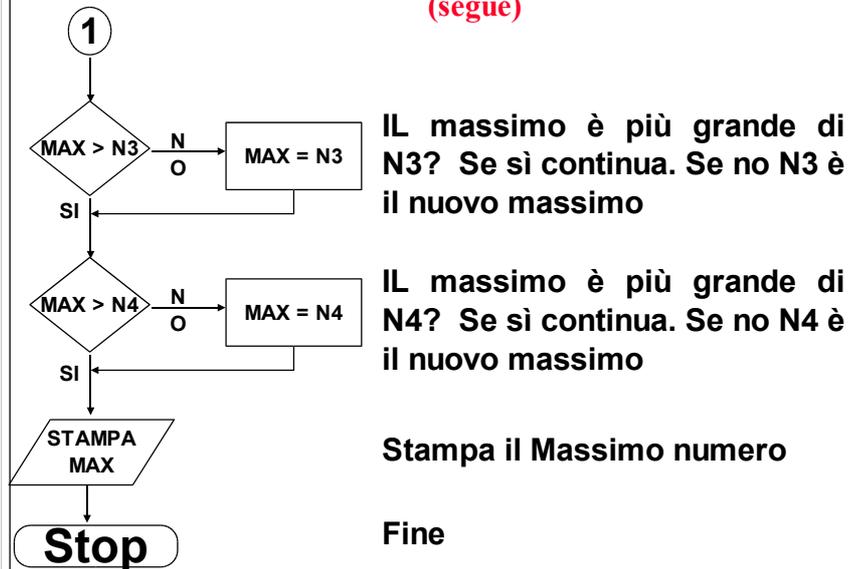


Flow-chart



Flow-chart

(segue)



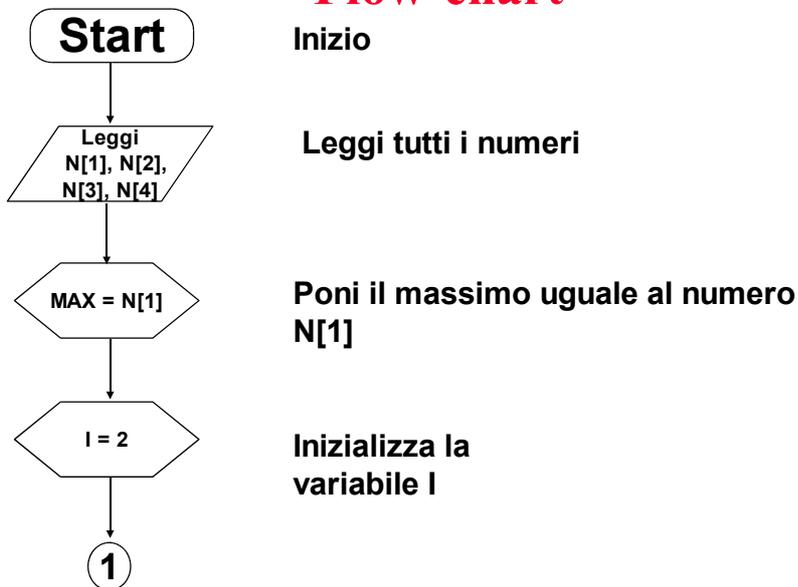
Seconda Versione

L'algorithmo precedente è una soluzione rigida al problema da risolvere.

E' sempre conveniente prevedere delle soluzioni il più possibile elastiche, che permettano di risolvere il maggior numero possibile di problemi analoghi.

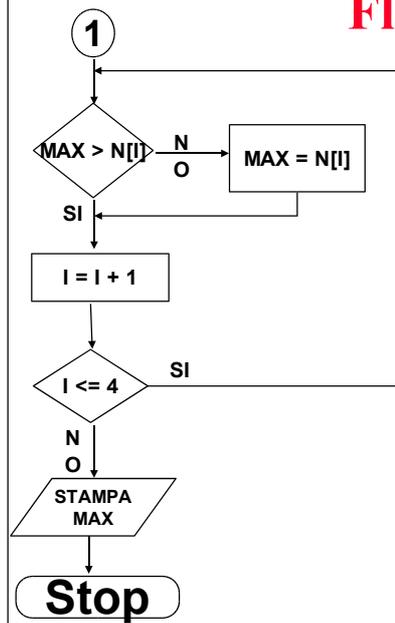
In questa seconda versione i dati del problema sono memorizzati in un vettore di nome N . Ogni elemento è univocamente caratterizzato da un indice I . Ad esempio il secondo numero è accessibile come $N[2]$.

Flow-chart



Flow-chart

(segue)



Il massimo è più grande di N [I]? Se sì continua. Se no N[I] è il nuovo massimo

Ho confrontato tutti e 4 i dati?

Stampa il Massimo numero

Fine

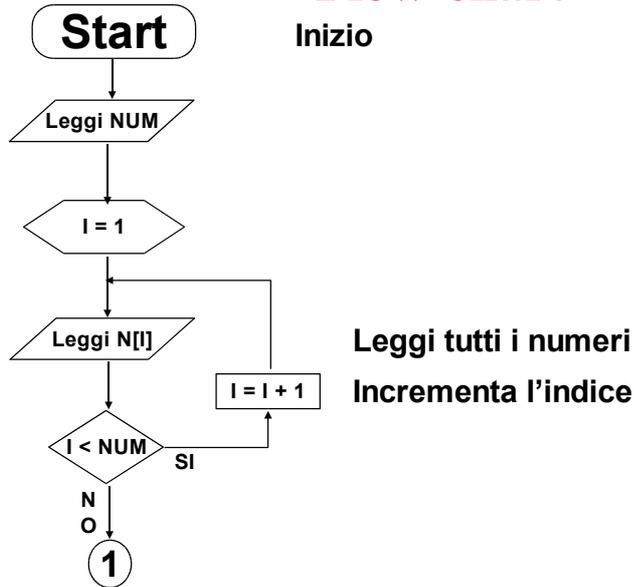
Terza Versione

Può essere scomodo avere fisso il numero di elementi da confrontare.

Per rendere generale l'algoritmo si vuole realizzare una soluzione che calcoli il massimo numero tra NUM dati.

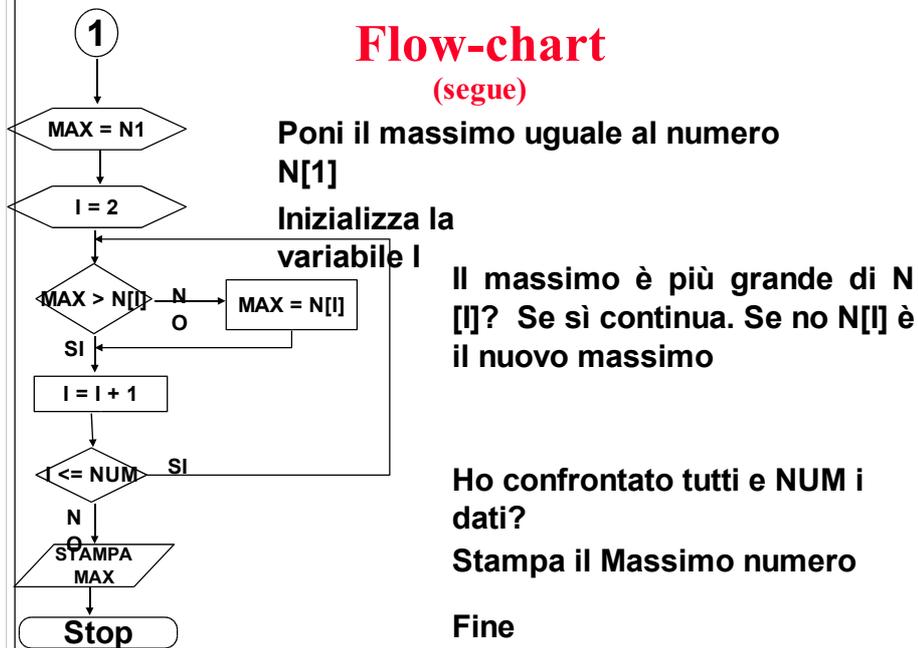
Il valore NUM è un parametro del problema.

Flow-chart



Flow-chart

(segue)



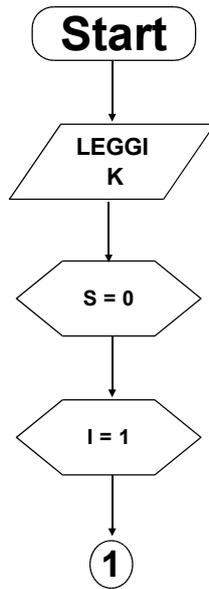
Esempio

Rappresentare con un diagramma di flusso l'algoritmo che effettua la somma di **K** numeri.

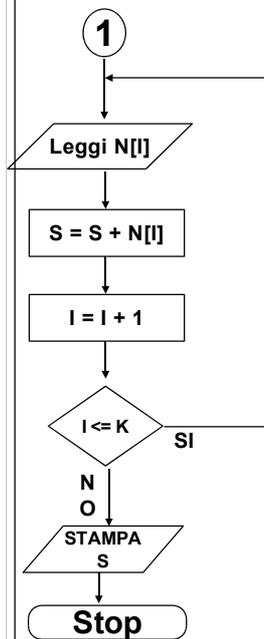
Flow chart



Flow chart



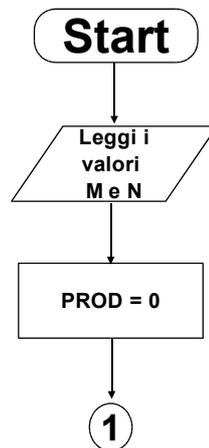
Flow-chart (segue)



Esercizio

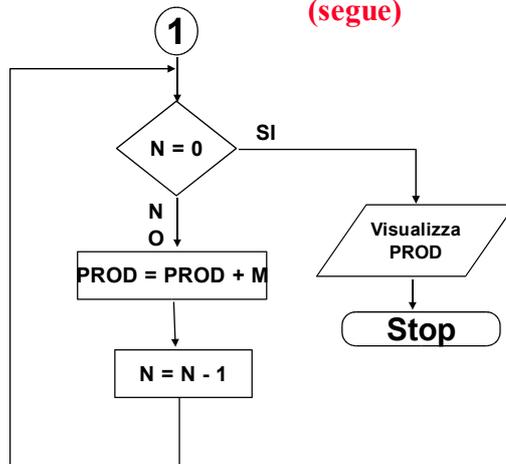
Realizzare il diagramma di flusso dell'algoritmo che risolve il prodotto di due numeri col metodo delle addizioni successive.

Soluzione



Soluzione

(segue)



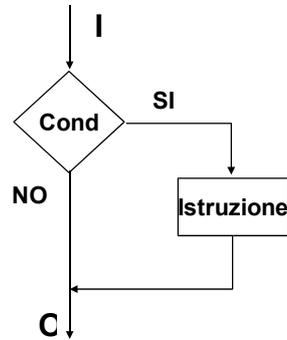
Strutture elementari

Nella programmazione esistono alcuni costrutti elementari che permettono di svolgere specifiche operazione a seconda se una condizione è soddisfatta oppure no.

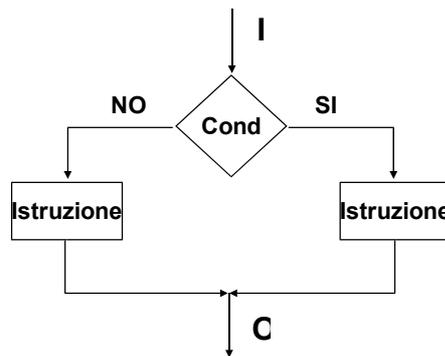
Tali costrutti elementari sono:

- **blocchi condizionali**
- **cicli while do**
- **cicli repeat until.**

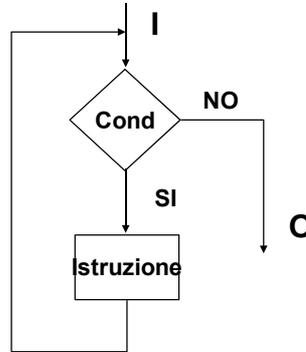
Blocco Condizionale: IF THEN



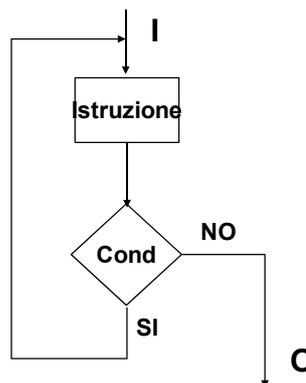
Blocco condizionale IF THEN ELSE



Ciclo WHILE DO



Ciclo REPEAT UNTIL



Esempio

Scrivere un algoritmo per calcolare la media tra N numeri.

Pseudo-codice

Leggo il numero di dati e lo metto in NUM

Inizializzo la variabile SOMMA a 0

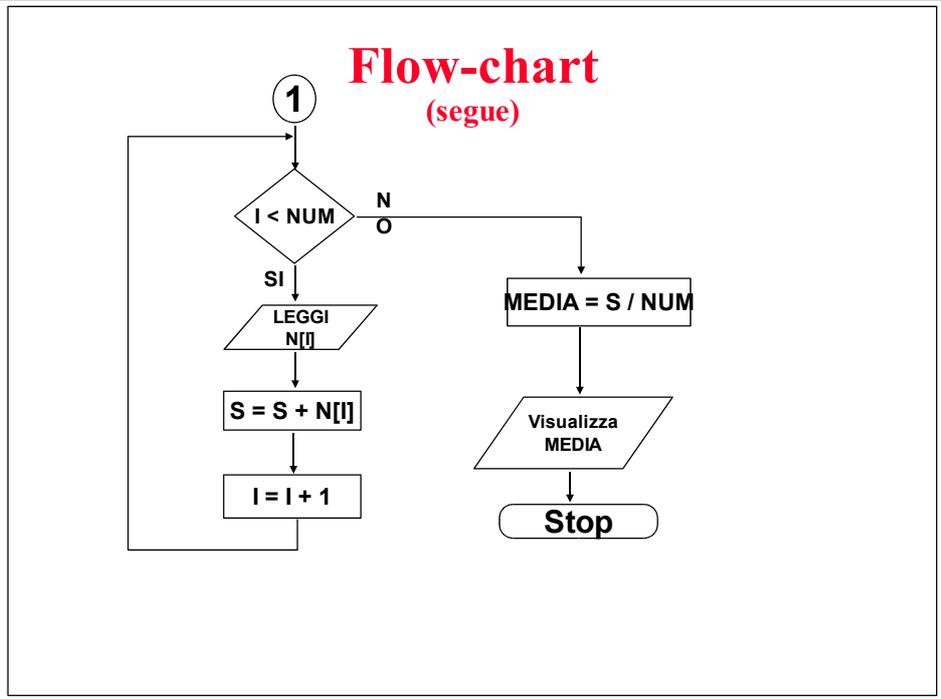
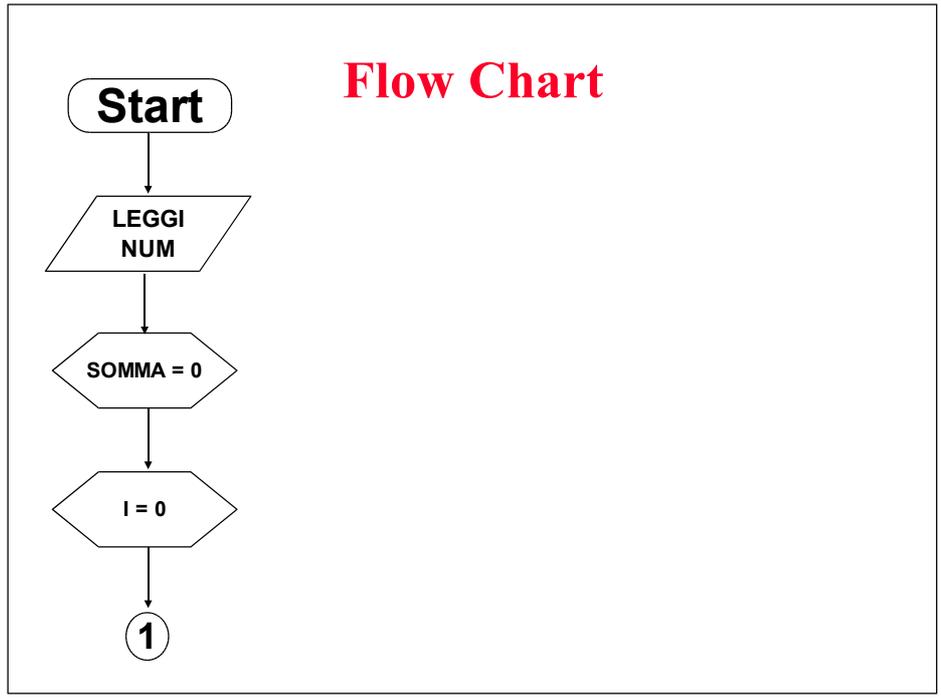
Inizializzo la variabile I a 0

While (I è minore di NUM)

- **Leggo N(I)**
- **SOMMA = SOMMA + N[I]**
- **I = I + 1**

MEDIA = SOMMA / NUM

Stampa la variabile MEDIA



Esempio

Realizzare un algoritmo che in un insieme di numeri calcola quanti sono maggiori di 5.

Pseudo-codice dell'algoritmo

Leggo il numero di dati e lo metto in NUM

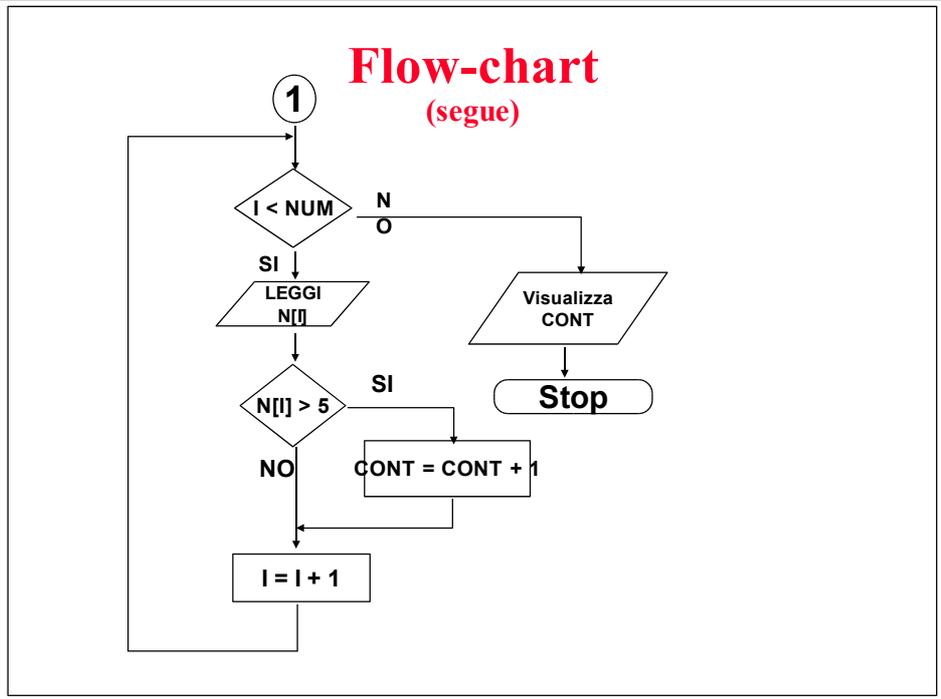
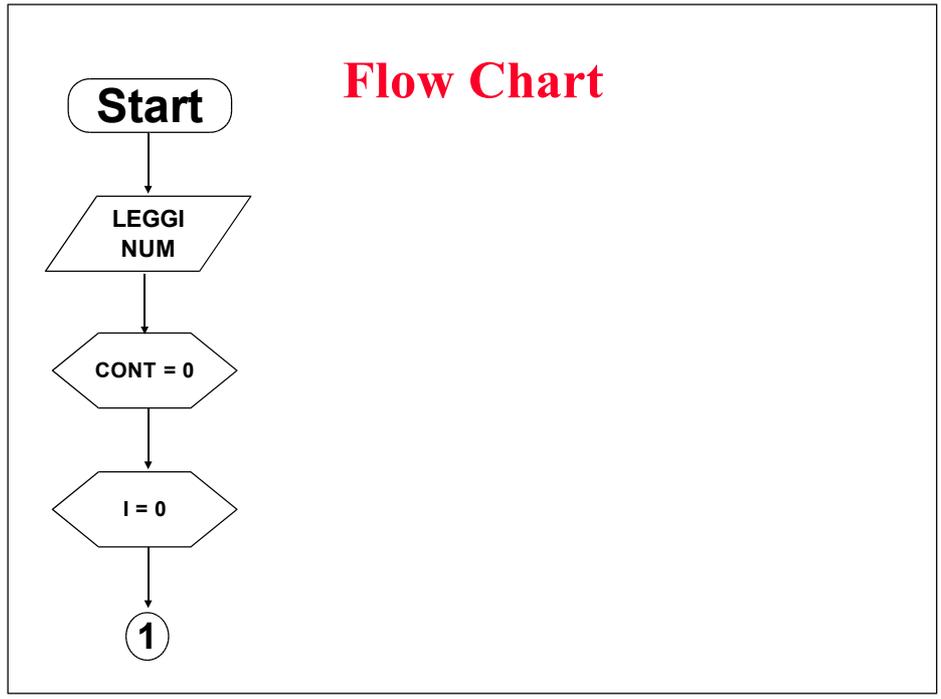
Inizializzo la variabile CONT a 0

Inizializzo la variabile I a 0

While (I è minore di NUM)

- **Leggo N[I]**
- **IF N[I] > 5 THEN CONT = CONT + 1**
- **I = I + 1**

Stampo la variabile CONT



Diagrammi di flusso strutturati

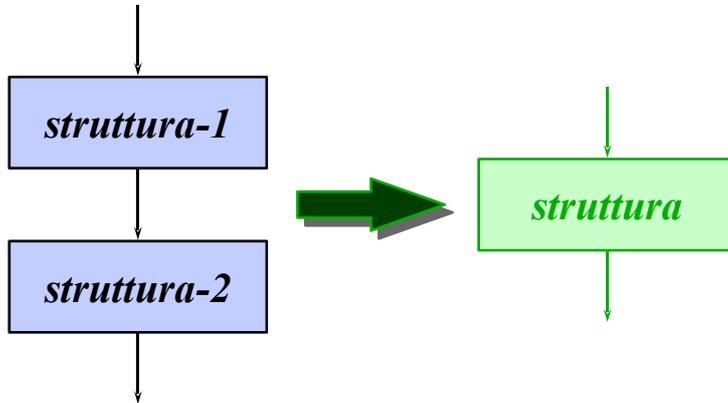
Un diagramma di flusso è detto *strutturato* se contiene solo un insieme predefinito di strutture:

- sequenza
- IF-THEN-ELSE
- WHILE
- REPEAT

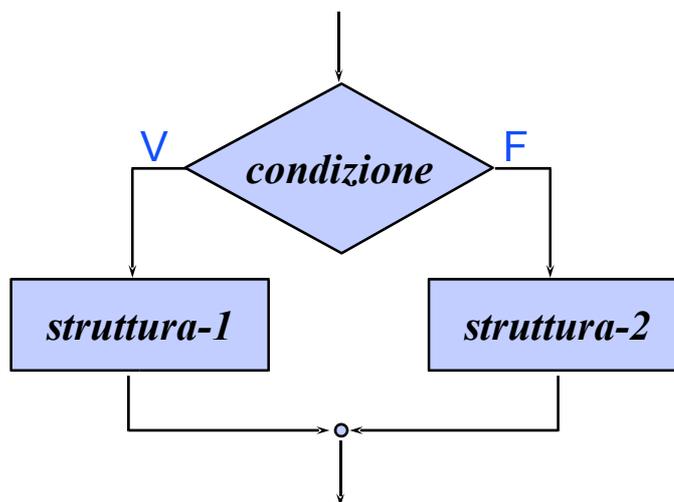
Teorema di Böhm - Jacopini

Qualunque diagramma di flusso è sempre trasformabile in un diagramma di flusso strutturato equivalente a quello dato.

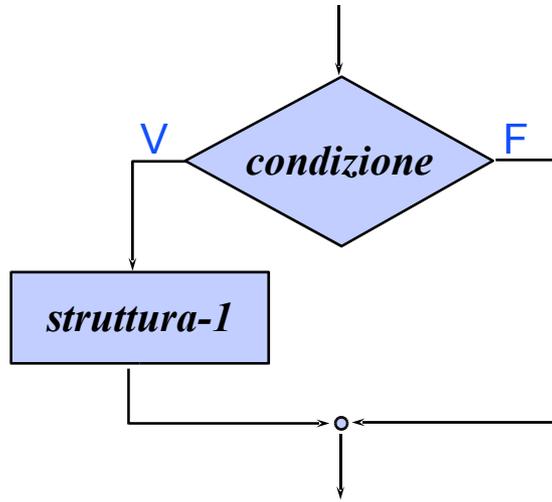
Sequenza



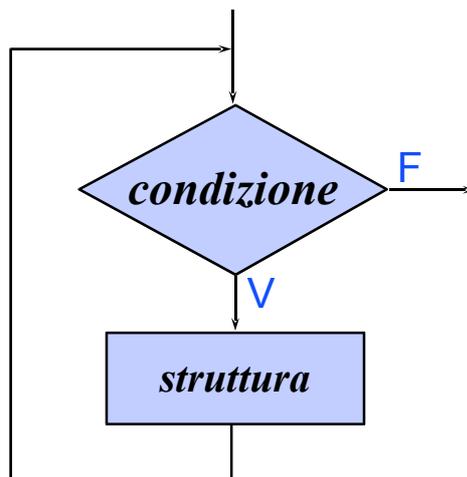
If-Then-Else



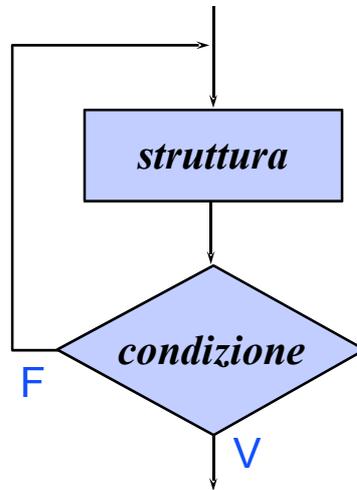
If-Then



While-Do



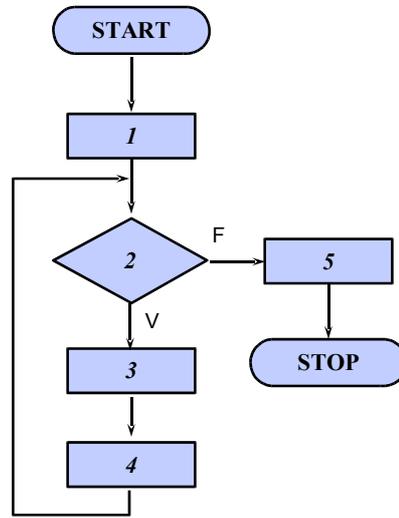
Repeat-Until



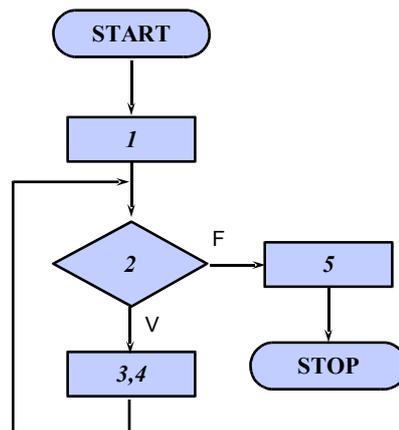
Verifica di strutturazione

- etichettare ogni blocco
- sostituire ad ogni insieme strutturato un blocco con l'unione delle etichette
- se si è fatta almeno una sostituzione, ripetere il passo precedente
- se alla fine si ottiene un diagramma lineare, allora il diagramma originale è strutturato

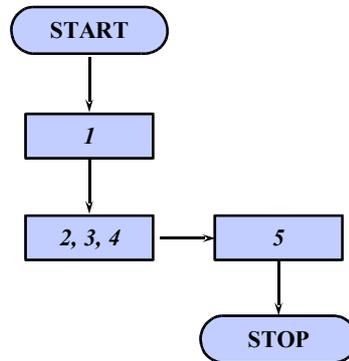
Esempio: diagramma strutturato



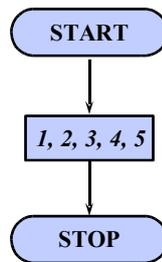
Esempio: diagramma strutturato



Esempio: diagramma strutturato



Esempio: diagramma strutturato



**Esempio:
diagramma
non
strutturato**

