MODULO 1. CONCETTI DI BASE DELL'ICT

UNITA' DI APPRENDIMENTO 1: IL LINGUAGGIO DI PROGRAMMAZIONE.

1.1 LA PROGRAMMAZIONE STRUTTURATA

Dopo avere visto come è possibile rappresentare un algoritmo utilizzando un linguaggio formale (pseudo codifica e flow-chart) siamo in grado di veder come è possibile tradurlo in linguaggio di programmazione.

Innanzitutto qualunque linguaggio di programmazione possiede il modo di rappresentare i dati e le istruzioni

I DATI

In informatica i **dati** sono caratterizzati da un **nome** e da un **tipo.**

Il **nome o identificatore** serve per identificare univocamente il dato.
E' buona norma utilizzare nomi che consentano di richiamarne immediatamente il significato.

Il **tipo** è legato al contenuto e quindi ai **valori** che il dato può assumere ed alle operazioni che possono essere eseguite su di **essi**.

A seconda di come interagiscono con il computer (e quindi con l'algoritmo e di conseguenza con il programma) i dati si classificano in:

- **dati di input**: sono quei dati forniti dall'esterno e che servono per la risoluzione del problema;
- **dati di output**:sono quelli che vengono comunicati all'esterno come soluzione del problema;
- **dati di lavoro (o di elaborazione)**: sono quelli utilizzati durante l'esecuzione del processo risolutivo (in aggiunta ai dati di input);

In funzione degli oggetti che rappresentano (quindi del loro **tipo**) i dati si classificano in:

- **dati numerici**: sono quei dati che contengono numeri e che possono essere utilizzati in formule matematiche. Tra di questi è possibile ancora distinguere:
  (a) dati numerici **interi:** sono quei dati che prevedono numeri senza cifre decimali (che in pseudocodifica indicheremo con **INT**);
  (b) dati numerici **reali:** sono quei dati che prevedono numeri con cifre decimali (che in pseudocodifica indicheremo con **REAL**);
- **dati alfanumerici**: sono quei dati che contengono sia caratteri (cifre, lettere, caratteri speciali, simboli di interpunzione, etc.) che singole cifre, ma sulle quali non è possibile effettuare alcun calcolo matematico. Tra di questi è possibile ancora distinguere:
  (a) dati alfanumerici formati da un singolo **carattere:** sono quei dati che prevedono come valore un qualunque carattere singolo (secondo lo standard ASCII) racchiuso tra apice singolo (che in pseudocodifica indicheremo con **CHAR**);
  (b dati alfanumerici formati da un insieme di caratteri (**stringhe**)**:** sono quei dati che prevedono un insieme di caratteri (secondo lo standard ASCII) racchiusi tra doppi apici (che in pseudocodifica indicheremo come **ARRAY DI CHAR**);
- **dati logici o booleani** sono quei dati che contengono i possibili valori di verità (VERO o FALSO da indicare senza l'uso dei doppi apici) (che in pseudocodifica indicheremo con **BOOL**);

In funzione della possibilità di cambiamento del loro valore i dati si classificano in:

- **dati variabili o variabili**: sono quei dati che possono cambiare il loro valore durante il processo risolutivo;
- **dati costanti o costanti**: sono quei dati che non possono cambiare il loro valore durante il processo risolutivo.

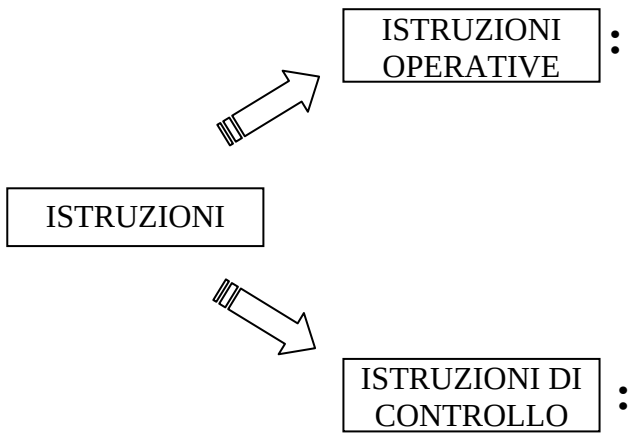
Schematizzando possiamo legare il concetto di variabile a quello di una scatola o **contenitore aperto** all'interno del quale è possibile inserire dei valori e sostituirli con degli altri purchè dello stesso tipo.
Invece possiamo legare il concetto di costante a quello di scatola o **contenitore chiuso** all'interno del quale non sarà possibile modificare il dato presente, ma se ne potrà sempre usare il contenuto

N.B. Il concetto di scatola o **contenitore** va legato al concetto di **cella di memoria** presente in un elaboratore ed utilizzato da un programma in esecuzione.

Le ISTRUZIONI

Abbiamo detto che ogni algoritmo è un insieme finito di istruzioni (azioni o passi) da fornire all'esecutore per poter trasformare secondo un preciso processo risolutivo (descritto dall'algoritmo) i dati iniziali o di **input** nei dati finali o di **output**.

Tra le istruzioni possibili distinguiamo:



Sono le istruzioni che corrispondono ad azioni direttamente eseguibili dall'elaboratore (in realtà dal microprocessore o CPU) e che possono riguardare:

- **Istruzione di assegnazione**
- **Istruzione di input**
- **Istruzione di output**
- **Dichiarazione di variabili**

Sono le istruzioni che permettono di controllare il flusso delle istruzioni di un algoritmo eventualmente instradando percorsi differenti durante l'esecuzione, in funzione del verificarsi o meno di determinate eventualità. Esse sono

1. **istruzione (o costrutto) di SEQUENZA;**
2. **istruzione (o costrutto) di SELEZIONE;**
3. **istruzione (o costrutto) di ITERAZIONE**

Con il termine **programmazione strutturata** si intende la programmazione che considera l'algoritmo come un insieme di blocchi di azioni o istruzioni ognuno fornito di un solo ingresso e di una sola uscita (blocchi di istruzioni ONE IN - ONE OUT).
Ciascun blocco è isolato dagli altri nel senso che non è possibile saltare dall'interno di un blocco all'interno di un altro.

Due matematici italiani Corrado **BÖHM** e Giuseppe **IACOPINI** nel 1966 formularono un importantissimo teorema che non dimostreremo che affermava:
"Un qualunque algoritmo scritto secondo le regole della programmazione a salti (ossia non strutturato) *per quanto complesso, può sempre essere trasformato in un algoritmo ad esso equivalente che utilizzi esclusivamente tre istruzioni di controllo fondamentali: sequenza, selezione ed iterazione (ossi a strutturato)"*

N.B. Due algoritmi si dicono **equivalenti** quando ottengono a partire dagli stessi dati iniziali, gli stessi dati finali utilizzando processi risolutivi diversi.

Il fondamentale teorema di **BÖHM-JACOPINI** ci assicura che ogni algoritmo può essere espresso con le sole tre strutture di controllo fondamentali sequenza, selezione e iterazione.

Inoltre ogni struttura di controllo dell'algoritmo, immaginata mediante il formalismo dei flow-chart, deve essere un blocco con una sola freccia in entrata ed una sola freccia in uscita.
Ogni blocco interno ad una struttura di controllo non è detto che sia semplice (singola istruzione) ma può essere a sua volta una struttura.

ISTRUZIONI OPERATIVE: Istruzione di assegnazione

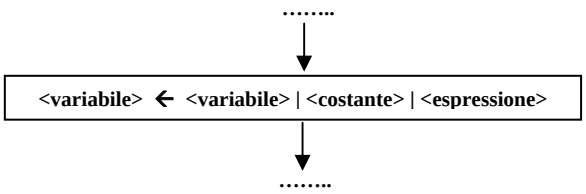
L'istruzione di assegnazione presente in tutti i linguaggi di programmazione consente di attribuire un valore ad una variabile.
Essa è in pratica un operatore di tipo binario identificato dal simbolo ← che viene detto **operatore di assegnazione**.

La sua pseudocodifica è la seguente

<variabile> ← <variabile> | <costante> | <espressione>

il primo termine a sinistra contiene sempre una variabile mentre quello a destra può essere una costante, un'altra variabile oppure un'espressione.

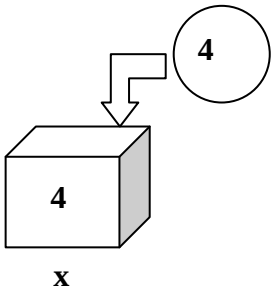
Nel flow-chart è rappresentata da un rettangolo al cui interno è scritta la pseudocodifica dell'istruzione



Casi possibili di assegnazione:

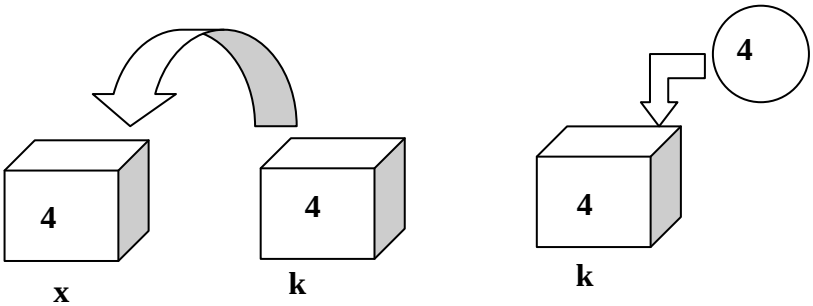
1) <variabile> ← <costante>

Es. **x ← 4**



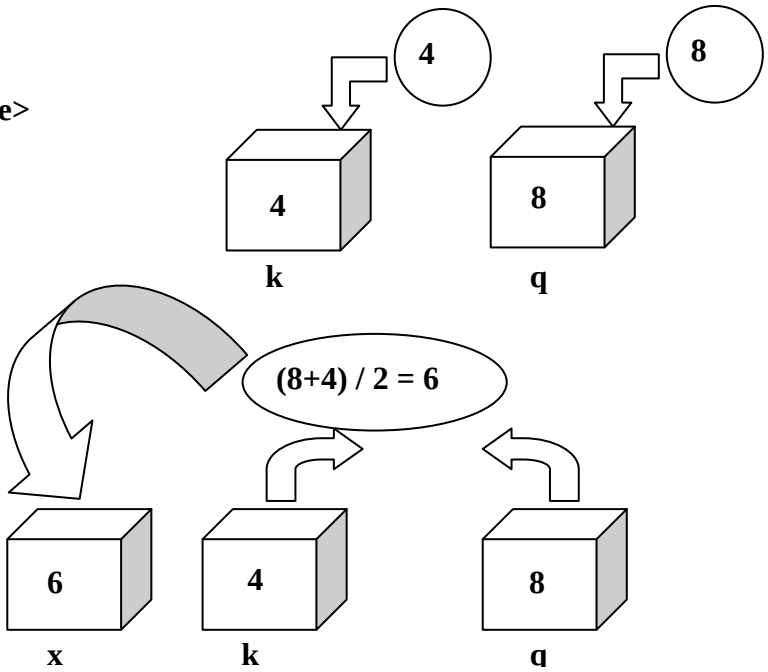
2) <variabile1> ← <variabile2>

Es. **k ← 4**
x ← k



3) <variabile> ← <espressione>

Es. **k ← 4**
q ←8
x ← (k + q) / 2



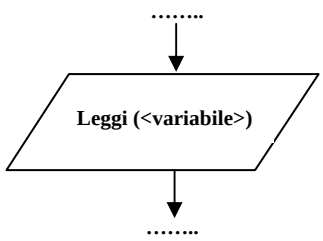
ISTRUZIONI OPERATIVE: Istruzione di input

L'istruzione di input presente in tutti i linguaggi di programmazione consente di attribuire dall'esterno un valore ad una variabile.

La sua pseudocodifica è la seguente

Leggi (<variabile>)

Nel flow-chart è rappresentata da un parallelogramma al cui interno è scritta la pseudocodifica dell'istruzione

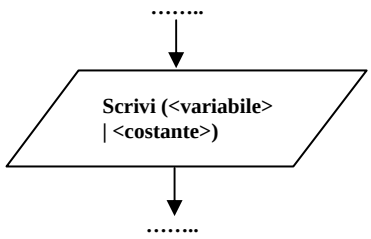


L'istruzione di ouput presente in tutti i linguaggi di programmazione consente di comunicare all'esterno il valore ad una variabile o di una costante

La sua pseudocodifica è la seguente

Scrivi (<variabile>|>costante>)

Nel flow-chart è rappresentata da un parallelogramma al cui interno è scritta la pseudocodifica dell'istruzione



Dichiarazione di una variabile

L'istruzione di dichiarazione di una variabile, presente in quasi tutti i linguaggi di programmazione, consente di definire all'interno di un algoritmo un dato di input, di output o di lavoro che sia variabile.
Non si applica nel caso di costanti che non devono essere dichiarate esplicitamente all'interno dell'algoritmo, ma che verranno implicitamente definite una volta dettagliate all'interno della tabella dei dati di lavoro (o elaborazione).

La sua pseudocodifica è

<variabile1> [<variabile2>,…<variabilen>] : <tipo>

N.B. Nel flow-chart le dichiarazioni di variabili NON SI RAPPRESENTANO

N.B. LA PRIMA COSA DA FARE, PRIMA DI PROCEDERE CON IL DETTAGLIO DELLE ISTRUZIONI DELL'ALGORITMO, È REDIGERE LE SEGUENTI TABELLE (VEDI PAGINA SEGUENTE) CHE IN PRATICA FORNISCONO, DOPO LA FASE DI ANALISI DEL PROBLEMA, L'ELENCO DI TUTTE LE VARIABILI, COSTANTI NECESSARIE ALLO SVOLGIMENTO DEL PROCESSO RISOLUTIVO.

TABELLE RIASSUNTIVE PER L'ANALISI DEI DATI

DATI DI INPUT DEL PROBLEMA PRINCIPALE (PROCEDURA MAIN)				
Nome variabile (1)	**Tipo dati (2)**	**Tipo Allocazione (3)**	**Valori ammessi (4)**	**Descrizione (5)**

DATI DI OUTPUT DEL PROBLEMA PRINCIPALE (PROCEDURA MAIN)				
Nome variabile	**Tipo dati**	**Tipo Allocazione**	**Valori ammessi**	**Descrizione**

DATI DI ELABORAZIONE (LAVORO) DEL PROBLEMA PRINCIPALE (PROCEDURA MAIN)				
Nome variabile oppure **nome costante (6)**	**Tipo dati**	**Tipo Allocazione**	**Valori ammessi**	**Descrizione**

(1) Il nome di una variabile deve essere scritto tutto con caratteri minuscoli senza utilizzare spazi e caratteri speciali (quali &, *,#, @). In caso di nomi di variabili costituiti da più di una parola è possibile utilizzare il carattere '-' (trattino) oppure '_' (underscore).
Esempio: Sono nomi di variabili valide i seguenti: a, pippo, somma-2, somma_2
Sono nomi di variabili non valide i seguenti: A, Pippo, somma 2, somma$2

(2) I tipi utilizzabili sono i tipi di dato semplice (un solo valore possibile per una variabile in un certo istante durante 'esecuzione dell'algoritmo) o tipo di dato strutturato o struttura dati (più valori possibili per una variabile in un certo istante durante l'esecuzione dell'algoritmo)
I tipi di dato semplice che utilizzeremo sono:
INT: per i numeri interi relativi (eventualmente con il segno) ma senza parte decimale; i valori possibili per questo tipo di variabile sono i normali interi numeri scritti nella modalità consueta;
Esempio: 12 -45 +32

REAL: per i numeri reali (eventualmente con il segno) con parte decimale; ; i valori possibile per questo tipo di variabile sono i normali numeri reali scritti nella modalità consueta utilizzando la virgola decimale;
Esempio: 12,34 -45,00 +32,1 ma anche 12 -45 +32

CHAR: per il singolo carattere alfanumerico secondo la codifica ASCII estesa: i valori possibili per questo tipo di variabile sono i caratteri racchiusi da due apici singoli;
Esempio: 'a' 'M' '$' '@' ';' 'b'|

BOOL: per il singolo valore di verità dell'algebra booleana: i valori possibili per questo tipo di variabile sono i seguenti: VERO oppure FALSO;

(3) Da valorizzare con:
STAT per indicare l'allocazione della variabile di tipo statico (ossia costante per tutta la durata dell'algoritmo)
DIN per indicare l'allocazione della variabile di tipo dinamico (ossia modificabile nel corso dell'esecuzione dell'algoritmo)
N.B. per questa prima fase useremo variabili esclusivamente STATICHE per le quali, quindi, occorre valorizzare con STAT la colonna "Tipo Allocazione" ad eccezione delle costanti per le quali scriveremo N.A. ossia non applicabile nella medesima colonna.

(4) In questa colonna occorre indicare, utilizzando se possibile il linguaggio della matematica, le condizioni che i valori di quella variabile devono rispettare nel corso dell'algoritmo.
Esempio: 'Se a è la variabile STAT di tipo INT che rappresenta il primo coefficiente di un'equazione di secondo grado è ovvio che dovrà essere scritta, nella colonna "Valori ammessi", la condizione a≠0

(5) Da valorizzare indicando brevemente ma in modo significativo la spiegazione del significato della variabile.

(6) Il nome di una costante deve essere scritto tutta con caratteri MAIUSCOLI senza utilizzare spazi e caratteri speciali (quali &, *,#, @). In caso di nomi di costanti costituiti da più di una parola è possibile utilizzare il carattere '-' (trattino) oppure '_' (underscore).
Esempio: Sono nomi di costanti valide i seguenti: PIGRECO PI-GRECO PI_GRECO
Sono nomi di costanti non valide i seguenti: Pigreco PI+GRECO PI&GRECO

ISTRUZIONI DI CONTROLLO:

a) L'istruzione o costrutto di "**Sequenza"**:

La sequenza è il costrutto più semplice tra i tre costrutti fondamentali della programmazione strutturata. Si utilizza quando le azioni ossia le istruzioni devono essere seguite ordinatamente una dopo l'altra senza alcuna possibilità di scelta.

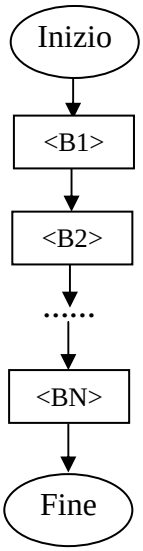
Pseudocodifica istruzione di Sequenza":

INIZIO
<B1>
<B2>
…..
<BN>
FINE

<B1>, <B2>…<BN> possono essere **blocchi semplici** o **blocchi composti** di istruzioni
Un *blocco semplice* è ad esempio una istruzione di assegnazione, una istruzione di I/O (input/output) oppure una sola tra le altre istruzioni di controllo.
Un *blocco composto* è invece l'insieme di più blocchi semplici.

N.B. Per questo motivo anche un intero algoritmo strutturato (costituito solo attraverso i tre costrutti fondamentali di sequenza, selezione ed iterazione) è in realtà una struttura sequenziale.

Flow-chart istruzione di "Sequenza"



b) Istruzione o costrutto di "**Selezione"**:

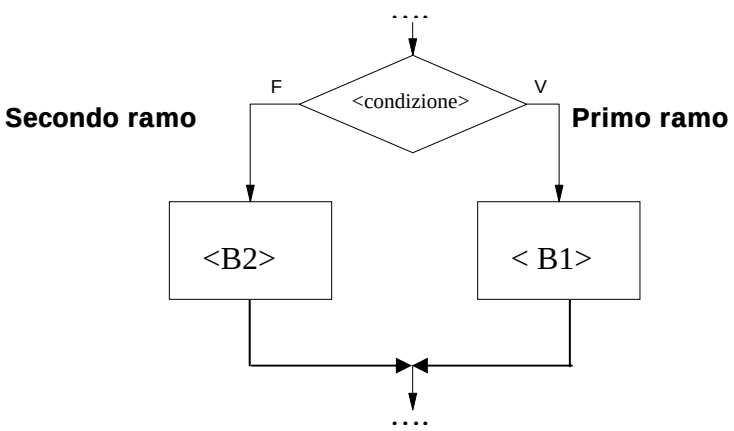
La selezione ci permette di instradare uno, due o più percorsi (vie o strade o rami) rispetto al flusso dell'algoritmo entrante.
In un primo momento distinguiamo tra le istruzioni di selezione che consentono *effettuare una scelta* fra due possibili alternative: la selezione unaria e la selezione binaria
Per effettuare la scelta dobbiamo *valutare una condizione booleana* (che ovviamente può assumere solo due possibili valori: vero o falso)

Pseudocodifica "Selezione binaria" o "a 2 vie":

….
SE <condizione>
ALLORA
<B1>
ALTRIMENTI
<B2>
FINE SE
….

Anche in questo caso <B1> e <B2> possono essere **blocchi semplici** o **blocchi composti**.
Se la condizione è VERA si esegue il blocco <B1>.
Se la condizione è FALSA si esegue il blocco <B2>.

Flow-chart "Selezione binaria" o "a 2 vie":

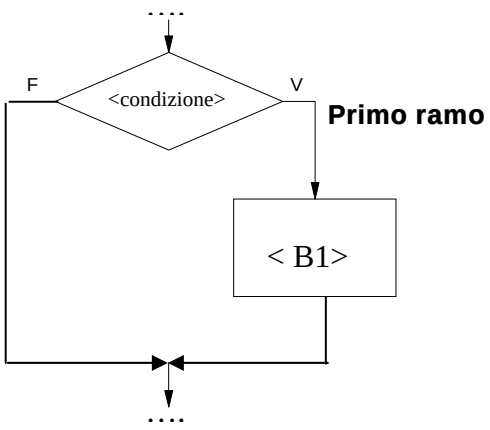


Pseudocodifica "Selezione unaria" o "a 1 via":

…
SE <condizione>
ALLORA
<B1>
FINE SE
….

Anche in questo caso <B1> può essere un **blocco semplice** o **blocco composto**.
Se la condizione è VERA si esegue il blocco <B1>.
Se la condizione è FALSA si prosegue normalmente**.**

Flow-chart "Selezione unaria" o "a 1 via":



Esiste anche un'istruzione di selezione che permette *di effettuare una scelta* fra n possibili valori ed instradare di conseguenza n possibili percorsi distinti rispetto al flusso dell'algoritmo entrante.
Per effettuare la scelta dobbiamo *valutare il valore di una variabile o di una espressione*

Pseudocodifica istruzione di "Selezione multipla o n-ria":

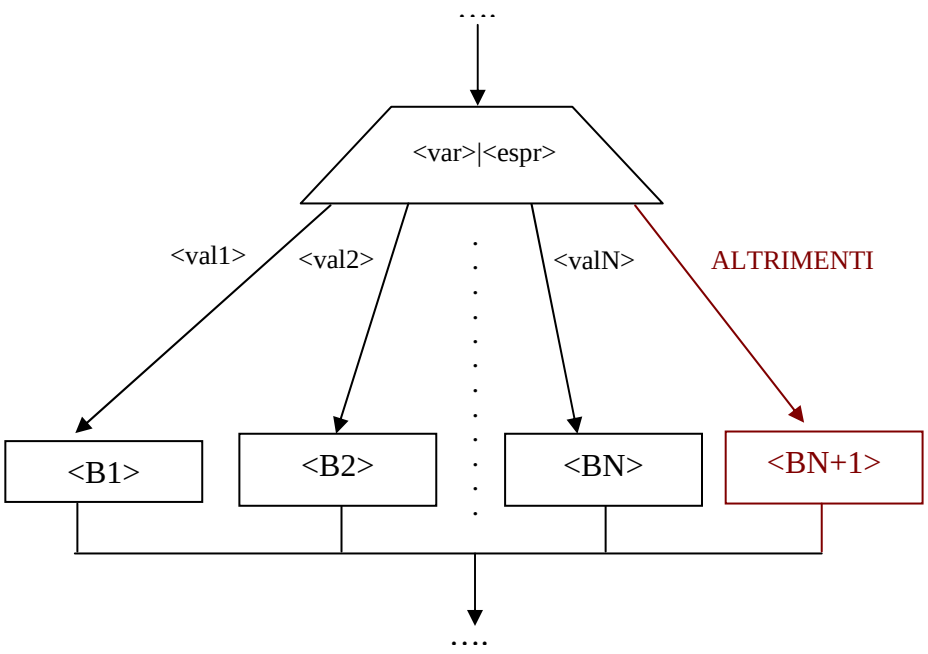
….
NEL CASO CHE <var> | <espr> **SIA**
<val1> **:** <B1>
<val 2> **:** <B2>
…… : ……
<val N> **:** <BN>
[ALTRIMENTI : <BN + 1>**]**
FINE CASO
….
N.B. e' possibile avere anche liste di valori separati da virgole

<*var*> | <*espr*> può essere una variabile oppure una espressione che può assumere un unico valore numerico intero o reale oppure un singolo carattere ma non può mai essere espresso come risultato di una condizione logica booleana.
<*valoreX*> può essere

- una costante di uguale valore a <var> | <espr>;
- un valore unico;
- una lista di valori;

N.B. E' opportuno ricordare che <valoreX> devono essere dello stesso tipo di <var> | <espr>

Flow-chart istruzione di "Selezione multipla o n-aria"



b) Istruzione o costrutto di "**iterazione"**:

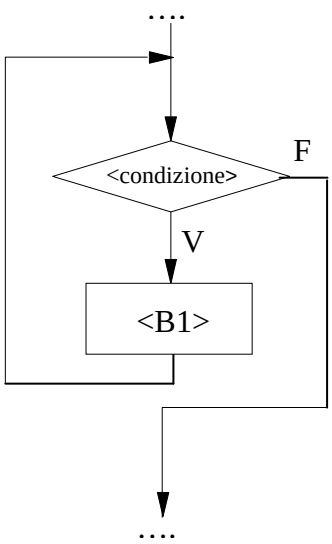
L'istruzione di iterazione viene utilizzata quando una istruzione (o un gruppo di istruzioni.) deve essere eseguita finchè non si verifica una determinata condizione logica booleana.

Pseudocodifica istruzione di "Iterazione con controllo in testa (o PRE-condizionale)":

....
MENTRE <condizione> **ESEGUI**
<B1>
FINE MENTRE
....

Anche in questo caso B1 può essere un **blocco semplice** o un **blocco composto.**
Se la condizione è VERA si esegue il blocco B1.
Se la condizione è FALSA si arresta il processo iterativo.
Se la condizione è inizialmente FALSA il ciclo non viene mai eseguito

Flow-chart istruzione di "Iterazione con controllo in testa (o PRE-condizionale)":

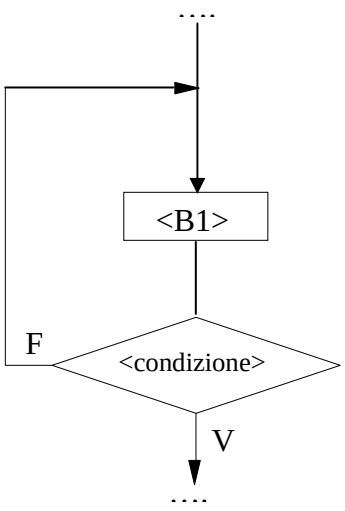


Pseudocodifica istruzione di "Iterazione con controllo in coda (o POST-condizionale)":

RIPETI
<B1>
FINCHE' <condizione>

Anche in questo caso <B1> può essere un **blocco semplice** o un **blocco composto.**
Se la condizione è FALSA si esegue il blocco <B1>.
Se la condizione è VERA si arresta il processo iterativo.

Flow-chart istruzione di "Iterazione con controllo in coda (o POST-condizionale)":



Mettiamo a confronto i due costrutti iterativi introdotti:

COSTRUTTO ITERAZIONE MENTRE	**COSTRUTTO ITERAZIONE RIPETI …. FINCHE'**
1) La condizione viene controllata **prima** di eseguire il blocco <B1> (PRE-condizionale); 2) Può anche **non eseguire mai** il blocco <B1>; 3) Esegue il blocco <B1> fino a che la condizione in testa è **VERA**;	1) La condizione viene controllata **dopo** avere seguito il blocco <B1> (POST-condizionale); 2) Esegue il blocco <B1> **almeno una volta**; 3) Esegue il blocco <B1> fino a che la condizione in coda è **FALSA**;

OSS: In molti casi conosciamo in partenza quante volte deve essere ripetuto un ciclo. Quando si conosce quante volte deve essere ripetuta un'azione è conveniente utilizzare il costrutto iterativo determinato o indicizzato

Pseudocodifica istruzione di "Iterazione determinato o indicizzato":

PER <indice> ← <inizio> **[INDIETRO] A** <fine> **ESEGUI**
<B1>
INCREMENTA <indice>
[**DECREMENTA** <indice>]
FINE PER

Anche in questo caso <B1> può essere un **blocco semplice** o un **blocco composto.**

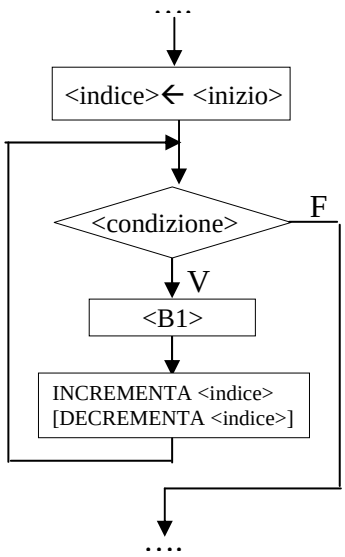
Oppure in modo equivalente:

PER <indice> ← <inizio> **[INDIETRO] A** <fine> **ESEGUI**
<B1>
<indice> ← <indice> + 1
[<indice> ← <indice> - 1]
FINE PER

Anche in questo caso <B1> può essere un **blocco semplice** o un **blocco composto.**

N.B. Tale tipo di costrutto iterativo viene utilizzato quando si conoscono a priori il numero di iterazioni richieste.

Flow-chart istruzione di "Iterazione determinato o indicizzato":



Appare evidente che tale tipo di costrutto equivale ad uno iterativo con controllo in testa e che quest'ultimo equivale ad un costrutto di selezione + costrutto iterativo con controllo in coda

Esempio

PER i ← 1 **A** 5 **ESEGUI**
    <B1>
    INCREMENTA i
FINE PER

Costrutto iterazione determinato o indicizzato

equivale ⟺

i ← 1

MENTRE (i <= 5) **ESEGUI**
    <B1>
    i ← i + 1
FINE MENTRE

Costrutto iterazione con controllo in testa o per vero

equivale ⟺

i ← 1

SE (i <= 5)
  ALLORA
      RIPETI
        <B1>
        i ← i + 1
      FINCHE' NOT (i <= 5)
FINE SE

Costrutto iterazione con controllo in coda o per falso

1.2 IL LINGUAGGIO DI PROGRAMMAZIONE

1.3 DATI STRUTTURATI

1.4 LE STRUTTURE INFORMATIVE