Struttura di un ALGORITMO

↓

Tabelle dei DATI

1

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

N.B. Va utilizza una RIGA per ogni VARIABILE

2

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

N.B. Va utilizza una RIGA per ogni VARIABILE

3

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

N.B. Va utilizza una RIGA per ogni VARIABILE o COSTANTE

(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 oppure il punto decimale;

Esempio: 12,34 -45,00 +32,1 ma anche 12.34 -45.00 +32.1

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:
STATICA per indicare l'allocazione della variabile di tipo statico (ossia costante per tutta la durata dell'algoritmo)
DINAMICA per indicare l'allocazione della variabile di tipo dinamico (ossia modificabile nel corso dell'esecuzione dell'algoritmo)

N.B. in questa prima fase useremo variabili esclusivamente STATICHE per le quali, quindi, occorre valorizzare con STATICA la colonna "Tipo Allocazione" ad eccezione delle costanti per le quali scriveremo sempre STATICA per il tipo di allocazione (vengono sempre allocate così)

(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 STATICA di tipo REAL 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 del nome o identificatore assegnato alla variabile oppure alla costante.

(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 <u>non valide</u> i seguenti: Pigreco PI+GRECO PI&GRECO

Struttura di un ALGORITMO

PSEUDODODIFICA

ALGORITMO <Nome_Algoritmo>

.....
.....

AMBIENTE GLOBALE: le variabili, le costanti ed i tipi di dato dichiarati in questa zona saranno visibili (ossia **utilizzabili**) non solo dalla PROCEDURA main ( ), ma anche DA TUTTI I SOTTOPROGRAMMI (PROCEDURE e/o FUNZIONI) eventuali in cui l'algoritmo è stato eventualmente suddiviso.

N.B. In questa sezione NON è possibile inserire nessun altro tipo di istruzione

PROCEDURA main ( )

.....
.....

AMBIENTE LOCALE : le variabili, le costanti ed i tipi di dato qui dichiarati saranno visibili (ossia **utilizzabili**) **SOLO ALL'INTERNO** del SOTTOPROGRAMMA (in questo caso la PROCEDURA main ( ))

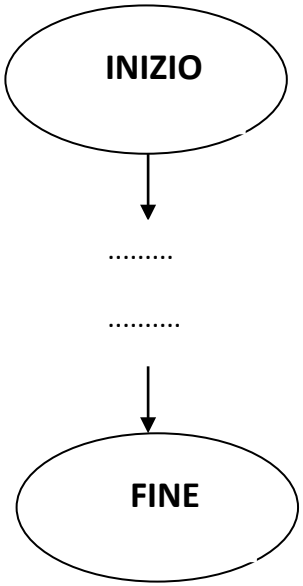
INIZIO

.....
.....
.....
.....
.....
.....

CORPO DEL SOTTOPROGRAMMA: contiene **TUTTE LE ISTRUZIONI** (istruzioni OPERATIVE ed istruzioni DI CONTROLLO) che descrivono i processo risolutivo del SOTTOPROGRAMMA (in questo caso la PROCEDURA main()) **TRANNE** le dichiarazioni delle variabili e la definizione delle costanti

FINE

FLOWCHART



Struttura del programma sorgente
*(che deriva dall'**ALGORITMO**)*

↓

CODIFICA C

[*] main.c

```
 1  #include <stdio.h>
 2  #include <stdlib.h>
 3
 4
 5
 6  int main(int argc, char *argv[])
 7  {
 8
 9
10
11
12
13
14
15
16
17
18  system("PAUSE");
19  return 0;
20  }
```

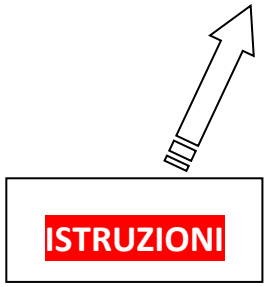
AMBIENTE GLOBALE: le variabili, le costanti ed i tipi di dato dichiarati in questa zona saranno visibili (ossia **utilizzabili**) non solo dalla FUNZIONE main ( ) , ma anche DA TUTTI I SOTTOPROGRAMMI (PROCEDURE e/o FUNZIONI) eventuali in cui l'algoritmo è stato eventualmente suddiviso.

AMBIENTE LOCALE : le variabili, le costanti ed i tipi di dato qui dichiarati saranno visibili (ossia **utilizzabili**) **SOLO ALL'INTERNO** del SOTTOPROGRAMMA (PROCEDURA o FUNZIONE) (in questo caso la FUNZIONE main ( ))

N.B. Comunque DOPO la parentesi graffa di apertura

CORPO DEL SOTTOPROGRAMMA(PROCEDURA o FUNZIONE): contiene **TUTTE LE ISTRUZIONI** (istruzioni OPERATIVE ed istruzioni DI CONTROLLO) che descrivono il processo risolutivo del SOTTOPROGRAMMA (in questo caso la FUNZIONE main()) **TRANNE** le dichiarazioni delle variabili e la definizione delle costanti

LE ISTRUZIONI



A

ISTRUZIONI OPERATIVE :

Sono le istruzioni che **corrispondono ad azioni direttamente eseguibili dall'elaboratore** (in realtà dal microprocessore o CPU) ma che non permettono in alcun modo di controllare il flusso del processo risolutivo

Di questo tipo di istruzioni fanno parte:

1. **La DICHIARAZIONE delle variabili e delle costanti**
2. **Le istruzioni di INPUT/OUTPUT**
3. **L'istruzione di ASSEGNAZIONE (o di ASSEGNAMENTO)**

B

ISTRUZIONI DI CONTROLLO :

Sono le istruzioni che **permettono di controllare il flusso del processo risolutivo di un algoritmo, modificandolo opportunamente** .

Di questo tipo di istruzioni fanno parte:

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

A. ISTRUZIONI OPERATIVE (ALGORITMO)

1. la dichiarazione delle VARIABILI

PSEUDODODIFICA

******* Dove va posizionata:

Nell'ambiente LOCALE (da evitare assolutamente l'utilizzo di variabili globali)

********Come va scritta:

<Nome_variabile> : <Tipo_variabile> →

Al momento usaremo solo **TIPI DI DATO SEMPLICE**:

INT	per numeri interi positivi e negativi (anche con segno)
REAL	per numeri decimali positivi e negativi (anche con segno)
CHAR	per singolo carattere codifica ASCII
BOOL	per valore booleano VERO o FALSO

N.B. E' possibile dichiarare più variabili dello stesso tipo separando i nomi con le virgole

****

ALGORITMO Esempio

…….
…….

PROCEDURA main ( )

a : INT
b, c : REAL
d : CHAR
e : BOOL

INIZIO

……..
……..
……..
……..
……..

FINE

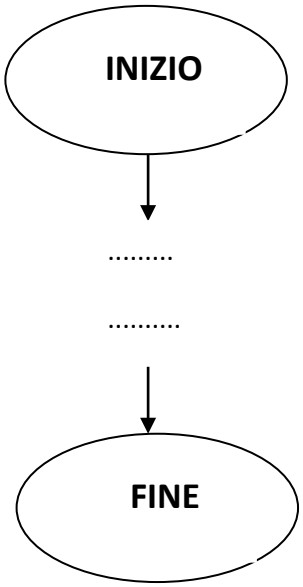
AMBIENTE GLOBALE: le variabili, le costanti ed i tipi di dato dichiarati in questa zona saranno visibili (ossia **utilizzabili**) non solo dalla PROCEDURA main ( ), ma anche DA TUTTI I SOTTOPROGRAMMI (PROCEDURE e/o FUNZIONI) eventuali in cui l'algoritmo è stato eventualmente suddiviso.

N.B. In questa sezione NON è possibile inserire nessun altro tipo di istruzione

AMBIENTE LOCALE : le variabili, le costanti ed i tipi di dato qui dichiarati saranno visibili (ossia **utilizzabili**) **SOLO ALL'INTERNO** del SOTTOPROGRAMMA (in questo caso la PROCEDURA main ( ))

CORPO DEL SOTTOPROGRAMMA: contiene **TUTTE LE ISTRUZIONI** (istruzioni OPERATIVE ed istruzioni DI CONTROLLO) che descrivono il processo risolutivo del SOTTOPROGRAMMA (in questo caso la PROCEDURA main()) **TRANNE** le dichiarazioni delle variabili e la definizione delle costanti

FLOWCHART



A. ISTRUZIONI OPERATIVE (PROGRAMMA SORGENTE)

1. La dichiarazione delle VARIABILI

CODIFICA C

[*] main.c

```c
#include <stdio.h>
#include <stdlib.h>



int main(int argc, char *argv[])
{
int a;

float b, c;

char d;

int e;              //Il tipo di dati BOOL non esiste in C



system("PAUSE");
return 0;
}
```

AMBIENTE GLOBALE: le variabili, le costanti ed i tipi di dato dichiarati in questa zona saranno visibili (ossia **utilizzabili**) non solo dalla FUNZIONE main ( ) , ma anche DA TUTTI I SOTTOPROGRAMMI (PROCEDURE e/o FUNZIONI) eventuali in cui l'algoritmo è stato eventualmente suddiviso.

AMBIENTE LOCALE : le variabili, le costanti ed i tipi di dato qui dichiarati saranno visibili (ossia **utilizzabili**) **SOLO ALL'INTERNO** del SOTTOPROGRAMMA (PROCEDURA o FUNZIONE) (in questo caso la FUNZIONE main ( ))

N.B. Comunque DOPO la parentesi graffa di apertura

CORPO DEL SOTTOPROGRAMMA(PROCEDURA o FUNZIONE):
contiene **TUTTE LE ISTRUZIONI**
(istruzioni OPERATIVE ed istruzioni DI CONTROLLO) che descrivono il processo risolutivo del SOTTOPROGRAMMA (in questo caso la FUNZIONE main())
TRANNE le dichiarazioni delle variabili e la definizione delle costanti

A. ISTRUZIONI OPERATIVE (ALGORITMO)
1. La dichiarazione delle COSTANTI

PSEUDODODIFICA

******* Dove va posizionata:

Nell'ambiente GLOBALE **(poiché il suo valore non deve cambiare, può essere messa a fattor comune)**

********Come va scritta:

<Nome_costante> <Valore_Costante> →

- NON SI DEVE DICHIARARE IL TIPO DI UNA COSTANTE (verrà automarticamente desunto dal valore assegnato
- NON SI POSSONO DICHIARARE ASSIEME PIU' COSTANTI (ogni costante va dichiarata su un proprio rigo)

****

```
ALGORITMO Esempio
PIGRECO    3.14
MAXVOTO 10
SI         'Y'
NO         'N'
FALSO      0
VERO       1

PROCEDURA main ( )

……..
……..

INIZIO

……..
……..
……..
……..
……..

FINE
```

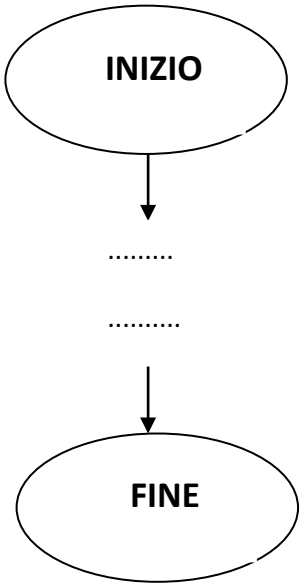
AMBIENTE GLOBALE: le variabili, le costanti ed i tipi di dato dichiarati in questa zona saranno visibili (ossia **utilizzabili**) non solo dalla PROCEDURA main ( ), ma anche DA TUTTI I SOTTOPROGRAMMI (PROCEDURE e/o FUNZIONI) eventuali in cui l'algoritmo è stato eventualmente suddiviso.

N.B. In questa sezione NON è possibile inserire nessun altro tipo di istruzione

AMBIENTE LOCALE : le variabili, le costanti ed i tipi di dato qui dichiarati saranno visibili (ossia **utilizzabili**) **SOLO ALL'INTERNO** del SOTTOPROGRAMMA (in questo caso la PROCEDURA main ( ))

CORPO DEL SOTTOPROGRAMMA: contiene **TUTTE LE ISTRUZIONI** (istruzioni OPERATIVE ed istruzioni DI CONTROLLO) che descrivono il processo risolutivo del SOTTOPROGRAMMA (in questo caso la PROCEDURA main()) **TRANNE** le dichiarazioni delle variabili e la definizione delle costanti

FLOWCHART



A. ISTRUZIONI OPERATIVE (PROGRAMMA SORGENTE)
1. La dichiarazione delle COSTANTI

↓

CODIFICA C

[*] main.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #define PIGRECO 3.14
5  #define MAXVOTO 10
6  #define SI 'Y'
7  #define NO 'N'
8  #define FALSO 0
9  #define VERO 1
10
11 int main(int argc, char *argv[])
12 {
13
14
15
16
17
18 system("PAUSE");
19 return 0;
20 }
```

AMBIENTE GLOBALE: le variabili, le costanti ed i tipi di dato dichiarati in questa zona saranno visibili (ossia **utilizzabili**) non solo dalla FUNZIONE main ( ) , ma anche DA TUTTI I SOTTOPROGRAMMI (PROCEDURE e/o FUNZIONI) eventuali in cui l'algoritmo è stato eventualmente suddiviso.

AMBIENTE LOCALE : le variabili, le costanti ed i tipi di dato qui dichiarati saranno visibili (ossia **utilizzabili**) **SOLO ALL'INTERNO** del SOTTOPROGRAMMA (PROCEDURA o FUNZIONE) (in questo caso la FUNZIONE main ( ))

N.B. Comunque DOPO la parentesi graffa di apertura

CORPO DEL SOTTOPROGRAMMA (PROCEDURA o FUNZIONE):
contiene **TUTTE LE ISTRUZIONI**
(istruzioni OPERATIVE ed istruzioni DI CONTROLLO) che descrivono il processo risolutivo del SOTTOPROGRAMMA (in questo caso la FUNZIONE main())
TRANNE le dichiarazioni delle variabili e la definizione delle costanti

A. ISTRUZIONI OPERATIVE (ALGORITMO)
2. Le istruzioni di INPUT/OUTPUT

PSEUDODODIFICA

ALGORITMO Esempio
PIGRECO 3.14
MAXVOTO 10
SI 'Y'
NO 'N'
FALSO 0
VERO 1

PROCEDURA main ( )
a : **INT**
b, c : **REAL**
d : **CHAR**
e : **BOOL**
INIZIO
Leggi(a)
Leggi(b)
Leggi(c)
Leggi(d)
Leggi(e)

Scrivi (a)
Scrivi (b)
Scrivi ("Il valore e': ")
Scrivi(c)
Scrivi(d)
Scrivi(e)

Scrivi(**PIGRECO**)
Scrivi(**MAXVOTO**)
Scrivi(**SI**)
Scrivi(**NO**)
Scrivi(**VERO**)
Scrivi(**FALSO**)
FINE

******* Dove va posizionata:

Istruzione Leggi(...)

Nel CORPO DELLA PROCEDURA
-ogni volta che dall'esterno (verso l'interno) va acquisito un valore per una variabile

********Come va scritta:

Leggi (<Nome_Variabile>) →

ATTENZIONE!
NON SI PUO' LEGGERE UNA COSTANTE

NON SI POSSONO LEGGERE ASSIEME PIU' VARIABILI

********NOTE:

a) SOLO l'iniziale della parola Leggi in maiuscolo

****

******* Dove va posizionata:

Istruzione Scrivi(...)

Nel CORPO DELLA PROCEDURA
-ogni volta che dall'interno (verso l'esterno) va comunicato il valore di una variabile

********Come va scritta:

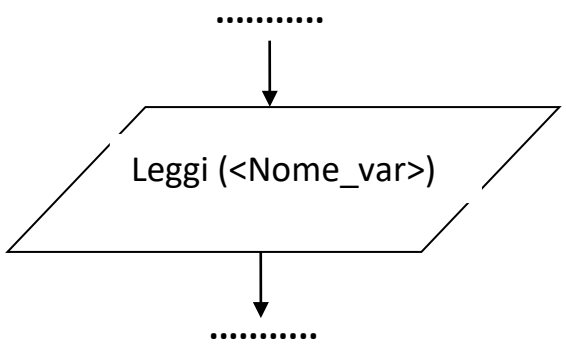
Scrivi (<Nome_Variabile> | <Nome_Costante>)

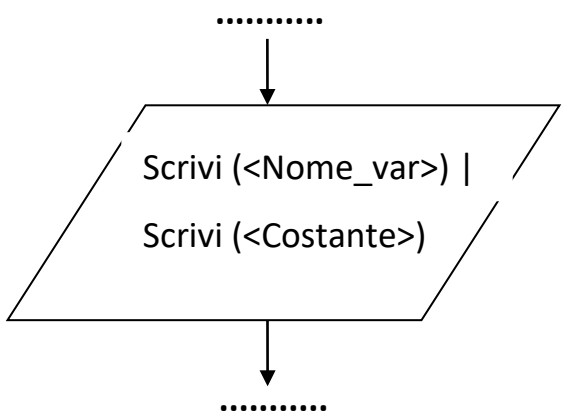
ATTENZIONE!
- NON SI POSSONO SCRIVERE ASSIEME PIU' VARIABILI
- NON SI POSSONO SCRIVERE ASSIEME PIU' COSTANTI

********NOTE:

a) SOLO l'iniziale della parola Scrivi in maiuscolo

FLOWCHART





A. ISTRUZIONI OPERATIVE (PROGRAMMA SORGENTE)
2. Le istruzioni di INPUT/OUTPUT

↓

CODIFICA C

[*] esempio-main.c

```c
#include <stdio.h>
#include <stdlib.h>

#define PIGRECO 3.14
#define MAXVOTO 10
#define SI 'Y'
#define NO 'N'
#define FALSO 0
#define VERO 1

int main(int argc, char *argv[])
{
int a;
float b,c;
char d;
int e;                  //il tipo di dati BOOL non esiste in C

scanf("%d", &a);        //La funzione scanf ammette due parametri:
scanf("%f", &b);        //il primo, racchiuso tra doppi apici, indica
scanf("%f", &c);        //    il formato della variabile da leggere ossia
scanf("%c", &d);        //    %d intera, %f reale, %c carattere
scanf("%d", &e);        //il secondo l'indirizzo (&) della variabile da leggere

printf("%d", a);
printf("%f", b);
printf("Il valore della variabile e': ");
printf("%f", c);
printf("%c", d);
printf("%d", e);

printf("%f", PIGRECO); //La funzione printf ammette due parametri:
printf("%d", MAXVOTO); //il primo, racchiuso tra doppi apici, deve contenere
printf("%c", SI);      //    il formato(i) della(e) variabile(i)/costante(i)
printf("%c", NO);      //    da scrivere ossia %d intera, %f reale, %c carattere
printf("%d", FALSO);   //    (ma può contenere anche altri caratteri)
printf("%d", VERO);    //il secondo indica il nome della(e) variabile(i) o
                       //    della(e) costante(i) da scrivere
system("PAUSE");
return 0;
}
```

AMBIENTE GLOBALE

AMBIENTE LOCALE

CORPO DEL SOTTOPROGRAMMA
(in questo caso FUNZIONE main())

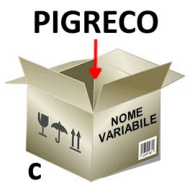
PSEUDODODIFICA

ALGORITMO Esempio
PIGRECO 3.14
PROCEDURA main ( )
a, b, sum : INT
c : REAL
INIZIO

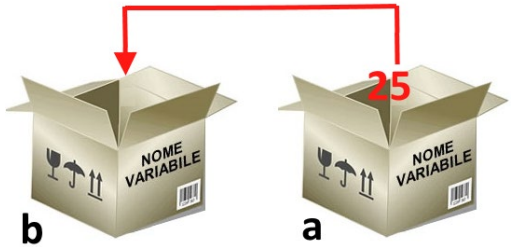
a ← 25

c ← PIGRECO

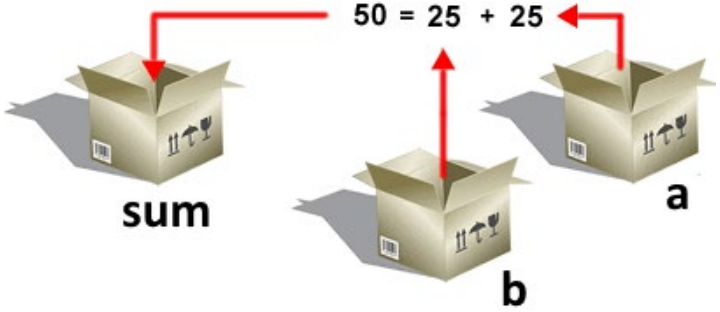
PIGRECO



b ← a



sum ← a + b



Scrivi(a)
Scrivi(b)
Scrivi(sum)
Scrivi(c)

FINE

Tale istruzione permette di assegnare valori alle variabili DIRETTAMENTE NEL CORSO DELL'ESECUZIONE del processo risolutivo
(valorizzazione ALTERNATIVA di una variabile rispetto l'istruzione **Leggi**)

******* Dove va posizionata:

Nel CORPO DELLA PROCEDURA
-ogni volta che occorre modificare il valore una variabile

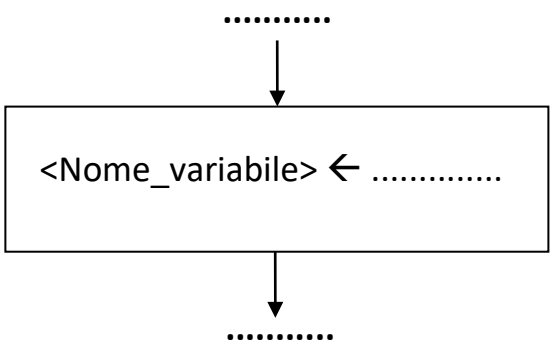
********Come va scritta:

<Nome_Variabile> ← <Valore_Costante> | <Nome_Variabile> | < Espressione>

********NOTE:

- La freccia ha la punta rivolta verso sinistra

FLOWCHART



OPERATORI

PSEUDODODIFICA

FLOWCHART

PREMESSA: Per potere esegure l'istruzione di assegnazione (←) è necessario specificare gli operatori (simboli e funzionamento) possibili che variano a seconda del tipo posseduto dalla variabile cui si vuole assegnare quel determinato valore.

OPERATORI ARITMETICI

a) Tra due dati di tipo INT sono possibili le seguenti operazioni

ALGORITMO Esempio
PROCEDURA main ( )
a, b, c : INT
INIZIO
a ← 15
b ← 6
c ← a + b /* **SOMMA** TRA DUE INTERI – c varrà 21 */
Scrivi(c)
c ← a - b /* **DIFFERENZA** TRA DUE INTERI - – c varrà 9 */
Scrivi(c)
c ← a * b /* **PRODOTTO** TRA DUE INTERI - – c varrà 90 */
Scrivi(c)
c ← a / b /* **DIVISIONE** (CON VIRGOLA) TRA DUE INTERI - – c varrà 2.5 */
Scrivi(c)
c ← a **DIV** b /* **DIVISIONE (SENZA VIRGOLA)** TRA DUE INTERI - – c varrà 2 */
Scrivi(c)
c ← a **%** b /* **MODULO** TRA DUE INTERI - – c varrà 3 */
Scrivi(c)
c ← b ^ 2 /* **POTENZA CON BASE ED ESPONENTE INTERI** - – c varrà 36 */
oppure
c ← b^2 /* **POTENZA CON BASE ED ESPONENTE INTERI** - – c varrà 36 */
oppure
c ← b*b /* **POTENZA CON BASE ED ESPONENTE INTERI** - – c varrà 36 */
Scrivi(c)

FINE

b) Tra due dati di tipo REAL sono possibili le seguenti operazioni

```
ALGORITMO Esempio
PROCEDURA main ( )
a, b, c : REAL
INIZIO
a ← 15.6
b ← 6.5
c ← a + b          /* SOMMA TRA DUE REALI – c varrà 22.1 */
Scrivi(c)
c ← a - b          /* DIFFERENZA TRA DUE REALI - – c varrà 9.1 */
Scrivi(c)
c ← a * b          /* PRODOTTO TRA DUE REALI - – c varrà 101.4 */
Scrivi(c)
c ← a / b          /* DIVISIONE (SEMPRE CON VIRGOLA) TRA DUE REALI - – c varrà 2.4 */
Scrivi(c)
FINE
```

c) Tra due dati di tipo CHAR sono possibili le seguenti operazioni

```
ALGORITMO Esempio
PROCEDURA main ( )
a, b: CHAR
offset: INT
INIZIO
a ← 'x'
b ← 'm'
/* N.B. tale operazione ha senso solo se il primo carattere SEGUE nella tavola ASCII il secondo */
offset ← a – b - 1        /* DIFFERENZA TRA DUE CARATTERI */
Scrivi(offset)    /* offset o spiazzamento ossia il numero di caratteri presenti tra i due */
FINE
```

CONNETTIVI LOGICI

d) Tra due dati di tipo BOOL sono possibili le seguenti operazioni

```
ALGORITMO Esempio
PROCEDURA main ( )
p, q, r: BOOL
INIZIO
p ← VERO
q ← FALSO
r ← p AND q        /* CONGIUNZIONE TRA DUE ENUNCIATI BOOLEANI*/
Scrivi(r)          /* l'enunciato composto r in questo caso risulterà FALSO */
r ← p OR q         /* DISGIUNZIONE TRA DUE ENUNCIATI BOOLEANI */
Scrivi(r)          /* l'enunciato composto r in questo caso risulterà VERO */
r ← NOT p          /* NEGAZIONE DI UN ENUNCIATO BOOLEANO */
Scrivi(r)          /* l'enunciato composto r in questo caso risulterà FALSO */
FINE
```

A. ISTRUZIONI OPERATIVE (PROGRAMMA SORGENTE)
3. L'istruzione di ASSEGNAZIONE

↓

CODIFICA C

[*] esempio-main.c

```c
#include <stdio.h>
#include <stdlib.h>

#define PIGRECO 3.14

int main(int argc, char *argv[])
{
int a, b, sum;
float c;

a = 25;
c = PIGRECO;

b = a;

sum = a + b;

printf("a = %d\n", a);
printf("b = %d\n", b);
printf("sum = %d\n", sum);
printf("c = %f\n", c);

system("PAUSE");
return 0;
}
```

AMBIENTE GLOBALE

AMBIENTE LOCALE

CORPO DEL SOTTOPROGRAMMA
(in questo caso FUNZIONE main())

OPERATORI

CODIFICA C

OPERATORI ARITMETICI

a) Tra due dati di tipo int sono possibili le seguenti operazioni

esempio-main.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>       /* libreria contenente le funzioni matematiche */

int main(int argc, char *argv[]) // PROCEDURA main()
{                                // INIZIO
int a,b,c;

a = 15;
b= 6;

/* SOMMA TRA DUE INTERI - c varrà 21 */
c = a + b;
printf("c varra' %d\n", c);
/* DIFFERENZA TRA DUE INTERI - - c varrà 9 */
c = a - b;
printf("c varra' %d\n", c);
/* PRODOTTO TRA DUE INTERI - - c varrà 90 */
c = a * b;
printf("c varra' %d\n", c);
/* N.B. DIVISIONE (CON VIRGOLA) TRA DUE INTERI - - c varrà SEMPRE 2 */
c = a / b;
printf("c varra' %d\n", c);
/* DIVISIONE (SENZA VIRGOLA) TRA DUE INTERI - - c varrà 2 */
c = a / b;
printf("c varra' %d\n", c);
/* MODULO TRA DUE INTERI - - c varrà 3 */
c = a % b;
printf("c varra' %d\n", c);
/* POTENZA CON BASE ED ESPONENTE INTERI - - c varrà 36 */
c = pow(b, 2);              /* Uso della funzione pow() */
printf("c varra' %d\n", c);

system("PAUSE");
return 0;
}                                // FINE
```

b) Tra due dati di tipo float sono possibili le seguenti operazioni

esempio-main.c

```c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) // PROCEDURA main()
{                                // INIZIO
float a,b,c;

a = 15.6;
b = 6.5;


/* SOMMA TRA DUE REALI - c varrà 22.100000 */
c = a + b;
printf("c varra' %f\n", c);     // N.B. %f da origine in stampa SEMPRE
                                // a 6 cifre dopo il punto decimale
/* DIFFERENZA TRA DUE REALI - - c varrà 9.100000 */
c = a - b;
printf("c varra' %f\n", c);
/* PRODOTTO TRA DUE REALI - - c varrà 101.400000 */
c = a * b;
printf("c varra' %f\n", c);
/* DIVISIONE (SEMPRE CON VIRGOLA) TRA DUE REALI - - c varrà 2.400000 */
c = a / b;
printf("c varra' %f\n", c);

system("PAUSE");
return 0;
}                                // FINE
```

c) Tra due dati di tipo char sono possibili le seguenti operazioni

esempio-main.c

```c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) // PROCEDURA main()
{                                // INIZIO
char a, b;
int offset;

a = 'x';
b = 'm';

/* N.B. Tale operazione ha senso solo se il primo carattere SEGUE
   nella tavola ASCII il secondo */
/* DIFFERENZA TRA DUE CARATTERI */
offset = a - b - 1;
/* offset o spiazzamento = il numero di caratteri presenti tra i due */
printf("offset = %d\n", offset);

system("PAUSE");
return 0;
}                                // FINE
```

CONNETTIVI LOGICI

d) Tra due dati di tipo int (N.B. da gestire come BOOL) sono possibili le seguenti operazioni

esempio-main.c

```c
#include <stdio.h>
#include <stdlib.h>

#define FALSO 0
#define VERO 1

int main(int argc, char *argv[]) // PROCEDURA main()
{                                // INIZIO
int p, q, r;    //N.B. Il tipo BOOL in C non esiste

p = VERO;
q = FALSO;

r = p && q;     /* CONGIUNZIONE TRA DUE ENUNCIATI BOOLEANI*/
/* l'enunciato composto r in questo caso risulterà FALSO */
printf ("enunciato composto r sara' %d\n", r);

r = p || q;     /* DISGIUNZIONE TRA DUE ENUNCIATI BOOLEANI */
/* l'enunciato composto r in questo caso risulterà VERO */
printf ("enunciato composto r sara' %d\n", r);

r = !p;     /* NEGAZIONE DI UN ENUNCIATO BOOLEANO */
/* l'enunciato composto r in questo caso risulterà FALSO */
printf ("enunciato composto r sara' %d\n", r);

system("PAUSE");
return 0;
}                                // FINE
```

PSEUDODODIFICA

FLOWCHART

La sequenza è il costrutto più semplice tra i tre costrutti fondamentali della programmazione strutturata.

L'istruzione (o costrutto) di SEQUENZA stabilisce che le istruzioni (o passi o azioni elementari) del processo risolutivo dell'algoritmo devono essere eseguite ordinatamente una dopo l'altra così come stabilito dal risolutore senza alcuna possibilità di scelta.

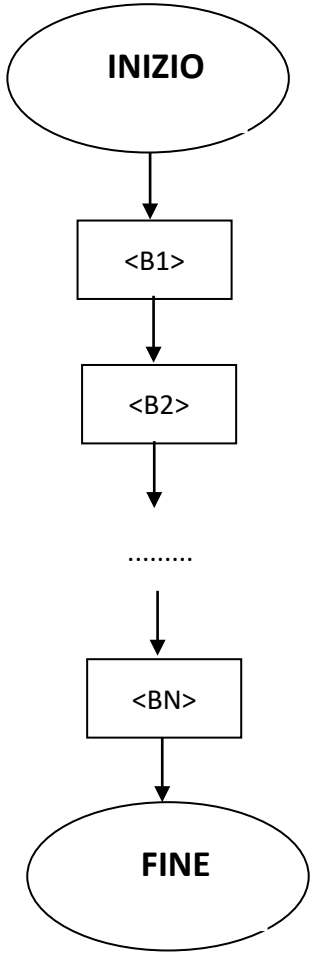
ALGORITMO Esempio
.....
PROCEDURA main ( )
.....
INIZIO
<B1>
<B2>
…..
<BN>
FINE

B1, B2 … BN possono essere **blocchi semplici** o **blocchi composti**.

Un *blocco semplice* è ad esempio UNA istruzione di assegnazione, UNA istruzione di I/O (input/output) oppure UNA delle prossime istruzioni(o costrutti) di controllo che illustreremo

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) può essere visto in realtà COME UN'UNICA STUTTURA SEQUENZIALE.



B. ISTRUZIONI DI CONTROLLO (PROGRAMMA SORGENTE)
1. L'istruzione (o COSTRUTTO) di SEQUENZA

↓

CODIFICA C

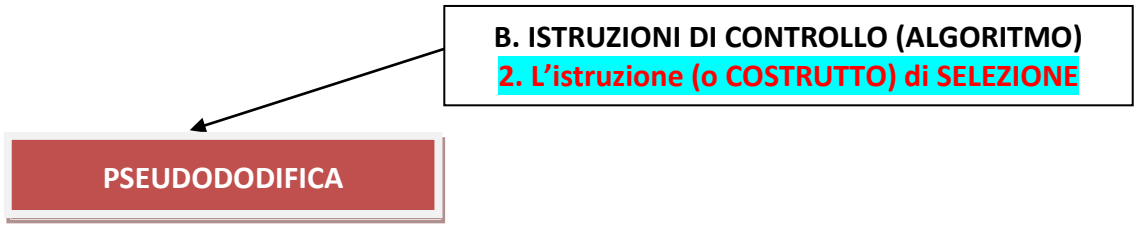
[*] esempio-main.c

```c
#include <stdio.h>
#include <stdlib.h>



int main(int argc, char *argv[])
{

<B1>;  //Primo Blocco semplice o composto di istruzioni C
<B2>;  //secondo Blocco semplice o composto di istruzioni C
....
<BN>;  //N-esimo Blocco semplice o composto di istruzioni C

system("PAUSE");
return 0;
}
```

AMBIENTE GLOBALE

AMBIENTE LOCALE

CORPO DEL SOTTOPROGRAMMA
(in questo caso FUNZIONE main())



L'istruzione (o costrutto) di SELEZIONE ci permette di instradare uno, due o più percorsi (vie o strade o rami) all'interno del flusso del processo risolutivo dell'algoritmo.

Ne esistono 3 tipologie:

- **L'istruzione (o costrutto) di SELEZIONE a UNA VIA o UNARIA**
- **L'istruzione (o costrutto) di SELEZIONE a DUE VIE o BINARIA**

Il loro funzionamento si basa **sulla logica booleana** ossia sulla valutazione del valore di verità di un enunciato semplice o composto (VERO o FALSO)

- **L'istruzione (o costrutto) di SELEZIONE a PIU' VIE o N-ARIA**

Il suo funzionamento si basa sulla **valutazione** del valore assunto da una **variabile** o da una **espressione**

PSEUDODODIFICA

FLOWCHART

➢ L'istruzione (o costrutto) di SELEZIONE a UNA VIA o UNARIA

PREMESSA

Tale istruzione permette di instradare UN SOLO percorso alternativo all'interno del flusso del processo risolutivo di un algoritmo.

******* Dove va posizionata:

Nel CORPO DELLA PROCEDURA

-ogni volta che occorre effettuare una scelta al massimo tra due possibilità

********Come va scritta:

```
SE (<condizione_logica>)
   ALLORA
      <B1>
FINE SE
```

LOGICA FUNZIONAMENTO

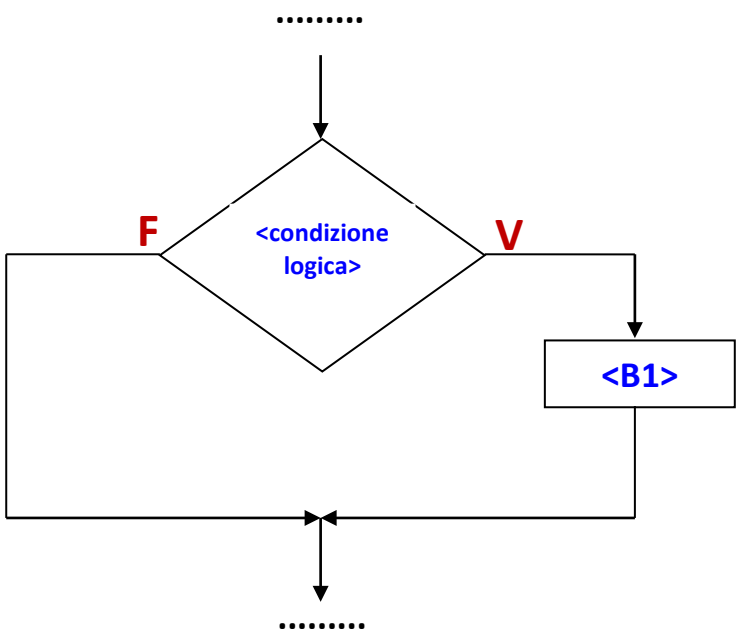
Se <condizione_logica> è:

- VERA verrà eseguito il blocco di istruzioni semplice o composto <B1>

- FALSA il flusso dell'esecuzione proseguirà normalmente

********NOTE:

- Occorre rispettare l'allineamento (INDENTAZIONE) proposto



B. ISTRUZIONI DI CONTROLLO (PROGRAMMA SORGENTE)
2. L'istruzione (o COSTRUTTO) di SELEZIONE

↓

CODIFICA C

- **L'istruzione (o costrutto) di SELEZIONE a UNA VIA o UNARIA**

[*] esempio-main.c

```
1   #include <stdio.h>
2   #include <stdlib.h>
3
4
5
6   int main(int argc, char *argv[])
7   {
8
9   if (<condizione_logica>)
10     {
11     <B1>;     // <B1> blocco di istruzioni semplice o composto
12     }
13
14  system("PAUSE");
15  return 0;
16  }
```

AMBIENTE GLOBALE

AMBIENTE LOCALE

CORPO DEL SOTTOPROGRAMMA
(in questo caso FUNZIONE main())

PSEUDODODIFICA

FLOWCHART

➢ L'istruzione (o costrutto) di SELEZIONE a DUE VIE o BINARIA

PREMESSA

Tale istruzione permette di instradare DUE PERCORSI alternativi all'interno del flusso del processo risolutivo di un algoritmo.

******* Dove va posizionata:

Nel CORPO DELLA PROCEDURA

-ogni volta che occorre effettuare una scelta al massimo tra due possibilità

********Come va scritta:

SE (<condizione_logica>)
ALLORA
<B1>
ALTRIMENTI
<B2>
FINE SE

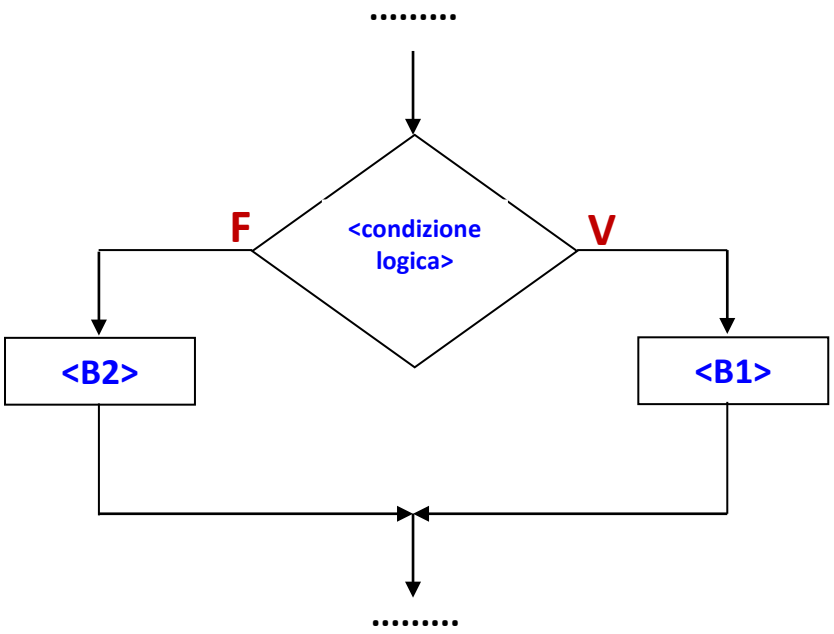
LOGICA FUNZIONAMENTO

Se <condizione_logica> è:

- **VERA verrà eseguito il blocco di istruzioni semplice o composto <B1>**
- **FALSA verrà eseguito il blocco di istruzioni semplice o composto <B2>**

********NOTE:

- Occorre rispettare l'allineamento (INDENTAZIONE) proposto



B. ISTRUZIONI DI CONTROLLO (PROGRAMMA SORGENTE)
2. L'istruzione (o COSTRUTTO) di SELEZIONE

↓

CODIFICA C

➢ L'istruzione (o costrutto) di SELEZIONE a DUE VIE o BINARIA

[*] esempio-main.c

```
1   #include <stdio.h>
2   #include <stdlib.h>
3
4
5
6   int main(int argc, char *argv[])
7   {
8
9   if (<condizione_logica>)
10     {
11     <B1>;     // <B1> blocco di istruzioni semplice o composto
12     }
13  else
14     {
15     <B2>;     // <B2> blocco di istruzioni semplice o composto
16     }
17
18  system("PAUSE");
19  return 0;
20  }
```

AMBIENTE GLOBALE

AMBIENTE LOCALE

CORPO DEL SOTTOPROGRAMMA
(in questo caso FUNZIONE main())

PSEUDODODIFICA

FLOWCHART

➢ L'istruzione (o costrutto) di SELEZIONE a PIU' VIE o N-ARIA

PREMESSA

Tale istruzione permette di instradare PIU' (DI DUE) PERCORSI alternativi all'interno del flusso del processo risolutivo di un algoritmo.

******* Dove va posizionata:

Nel CORPO DELLA PROCEDURA

- ogni volta che occorre effettuare una scelta tra più possibilità

********Come va scritta:

NEL CASO CHE (<nome_var> | <espr>) SIA
<valore_1> : <B1>
<valore_2> : <B2>
.....
<valore_N> : <BN>
[ALTRIMENTI : <BN+1>]
FINE CASO

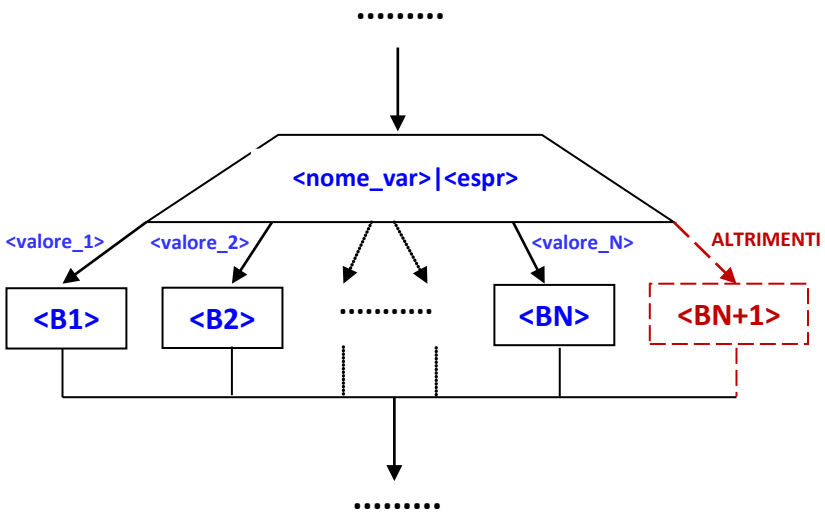
LOGICA FUNZIONAMENTO

Se <nome_var> | <espr> assume:

- **<valore_1> verrà eseguito il blocco di istruzioni <B1>**
- **<valore_2> verrà eseguito il blocco di istruzioni <B2>**

……..

- **<valore_N> verrà eseguito il blocco di istruzioni <BN>**
- **ALTRIMENTI verrà eseguito il blocco di istruzioni <BN+1>**



********NOTE:

- <nome_var> PUO' ESSERE di tipo INTERO o CARATTERE
- <espr> DEVE ASSUMERE UN VALORE INTERO
- CONSEGUENTEMENTE <valore_1>, <valore_2>, ... <valore_N> POSSONO ESSERE VALORI DI TIPO INTERO o CARATTERE
- Il ramo OPZIONALE ALTRIMENTI viene attivato SOLO PER valori DIVERSI da tutti quelli dettagliati esplicitamente
- nel caso in cui più valori diano origine all'esecuzione dello stesso blocco di istruzioni PER EVITARE RIPETIZIONI è possibile accorparli ossia scrivere l'istruzione di selezione ennaria così:

```
NEL CASO CHE (<nome_var> | <espr>) SIA
    <valore_1> :
    <valore_2> :
    <valore_3> : <B1>
    <valore_4> :
    <valore_5> : <B2>
    <valore_6> : <B3>
    ……................
    <valore_N> : <BN>
[ALTRIMENTI : <BN+1>]
FINE CASO
```

B. ISTRUZIONI DI CONTROLLO (PROGRAMMA SORGENTE)
2. L'istruzione (o COSTRUTTO) di SELEZIONE

↓

CODIFICA C

➢ L'istruzione (o costrutto) di SELEZIONE a PIU' VIE o N-ARIA

[*] esempio-main.c

```
1   #include <stdio.h>
2   #include <stdlib.h>
3
4
5
6   int main(int argc, char *argv[])
7   {
8
9   switch (<nome_var>|<espr>)
10     {
11     case <valore_1>:
12        {
13        <B1>;       // <B1> blocco di istruzioni semplice o composto
14        break;
15        }
16     case <valore_2>:
17        {
18        <B2>;       // <B2> blocco di istruzioni semplice o composto
19        break;
20        }
21     .....
22     .....
23     case <valore_N>:
24        {
25        <BN>;       // <BN> blocco di istruzioni semplice o composto
26        break;
27        }
28     default:
29        {
30        <BN+1>;     // <BN+1> blocco di istruzioni semplice o composto
31        break;      // N.B. l'ultimo break dell'istruzione può essere omesso!
32        }
33     }
34
35  system("PAUSE");
36  return 0;
37  }
```

AMBIENTE GLOBALE

AMBIENTE LOCALE

CORPO DEL SOTTOPROGRAMMA
(in questo caso FUNZIONE main())

➢ L'istruzione (o costrutto) di SELEZIONE ad UNA VIA o UNARIA

PSEUDODODIFICA

```
ALGORITMO Esempio
PROCEDURA main ( )
a, b : INT
INIZIO
Leggi (a)
Leggi (b)
SE (a ≥ b)
   ALLORA
      Scrivi(a)
FINE SE
FINE
```

N.B. Per usare tale istruzione dovremo conoscere ed utilizzare sia gli **OPERATORI DI CONFRONTO** sia i **CONNETTIVI LOGICI**

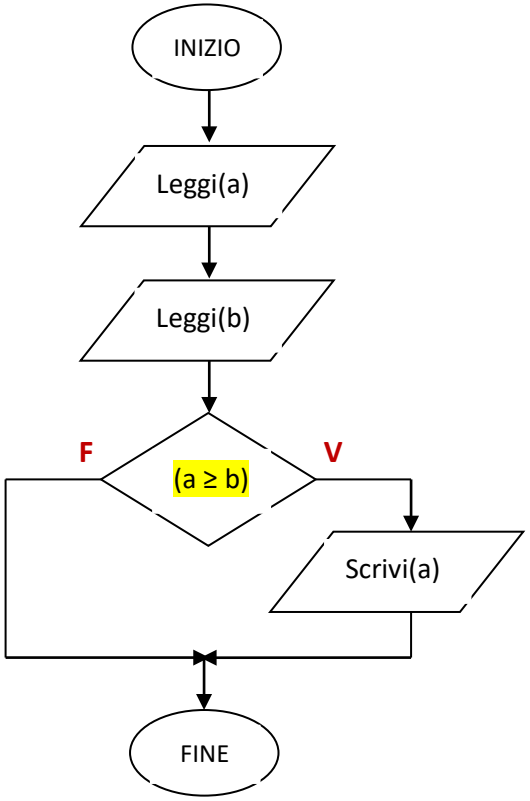
OPERATORI DI CONFRONTO

Lettura	PSEUDOCODIFICA	Linguaggio C
MAGGIORE	>	>
MAGGIORE o UGUALE	≥	>=
MINORE	<	<
MINORE o UGUALE	≤	<=
UGUALE	=	==
DIVERSO	≠	!=

OPERATORI BOOLEANI (CONNETTIVI LOGICI)

Lettura	PSEUDOCODIFICA	Linguaggio C
CONGIUNZIONE logica	AND	&&
DISGIUNZIONE logica	OR	||
NEGAZIONE logica	NOT	!

FLOWCHART



- **L'istruzione (o costrutto) di SELEZIONE ad UNA VIA o UNARIA**

esempio-main.c

```c
#include <stdio.h>
#include <stdlib.h>



int main(int argc, char *argv[])
{
int a, b;                                   // a,b : INT
                                            // INIZIO
scanf("%d", &a);                            // Leggi(a)
scanf("%d", &b);                            // Leggi(b)

if (a >= b)                                 // SE (a >= b)
   {                                        //    ALLORA
   printf("%d\n", a);                       //       Scrivi(a)
   }                                        // FINE SE

system("PAUSE");
return 0;
}                                           // FINE
```

AMBIENTE GLOBALE

AMBIENTE LOCALE

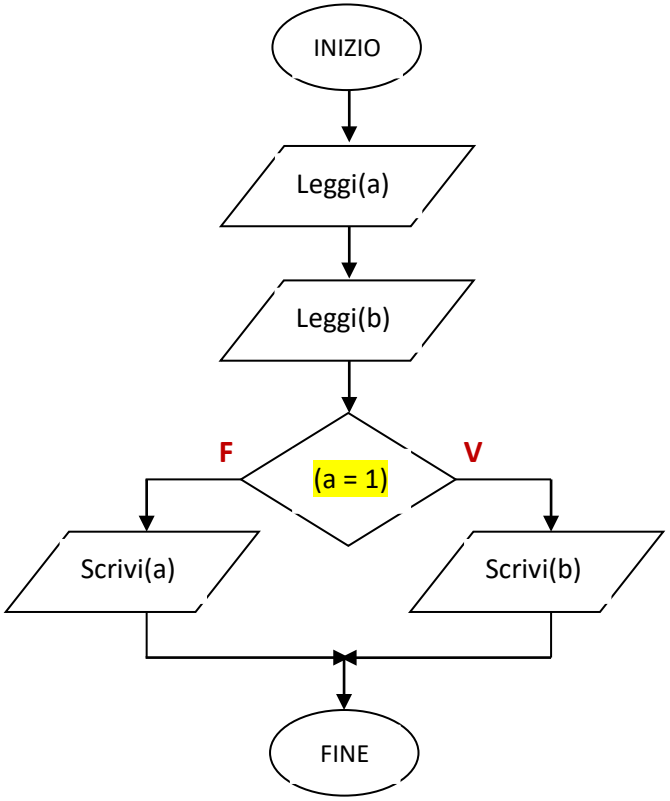
SECONDO ESEMPIO

➢ **L'istruzione (o costrutto) di SELEZIONE a DUE VIE o BINARIA**

PSEUDODODIFICA

```
ALGORITMO Esempio
PROCEDURA main ( )
a, b: INT
INIZIO
Leggi (a)
Leggi (b)
SE (a = 1)
   ALLORA
      Scrivi(b)
   ALTRIMENTI
      Scrivi(a)
FINE SE
FINE
```

FLOWCHART



➢ L'istruzione (o costrutto) di SELEZIONE a DUE VIE o BINARIA

esempio-main.c

```c
#include <stdio.h>
#include <stdlib.h>



int main(int argc, char *argv[])
{
int a, b;                               // a,b : INT
                                        // INIZIO
scanf("%d", &a);                        // Leggi(a)
scanf("%d", &b);                        // Leggi(b)

if (a == 1)                             // SE (a = 1)
   {                                    //    ALLORA
   printf("%d\n", b);                   //       Scrivi(b)
   }
else                                    //    ALTRIMENTI
   {
   printf("%d\n", a);                   //       Scrivi(a)
   }                                    // FINE SE

system("PAUSE");
return 0;
}                                       // FINE
```

AMBIENTE GLOBALE

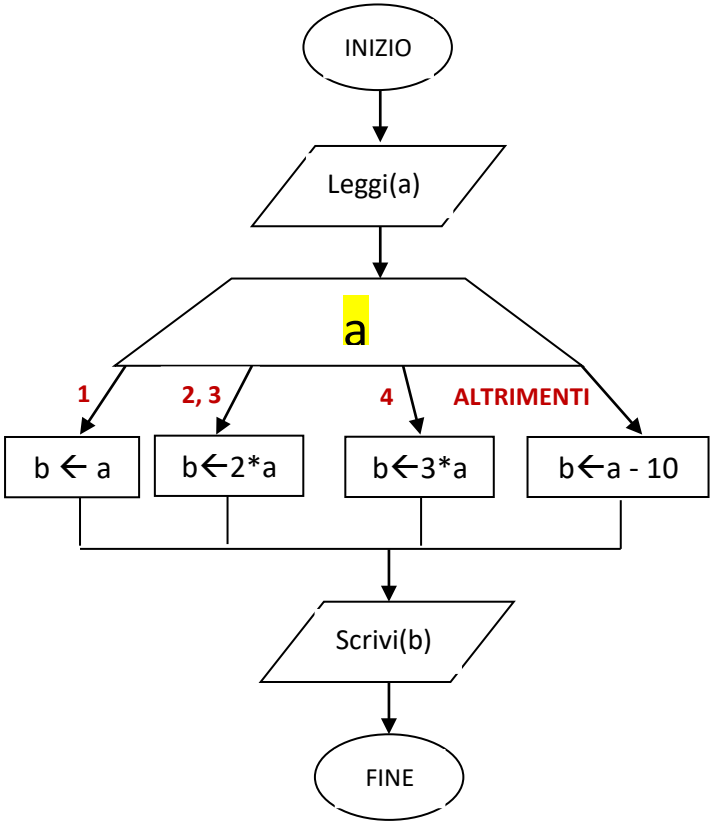
AMBIENTE LOCALE

➢ L'istruzione (o costrutto) di SELEZIONE a PIU' VIE o N-ARIA

PSEUDODODIFICA

```
ALGORITMO Esempio
PROCEDURA main ( )
a, b: INT
INIZIO
Leggi (a)
NEL CASO CHE (a) SIA
   1: b ← a          /* se a = 1 allora b sarà anch'essa uguale a 1 */
   2:
   3: b ← 2*a        /* se a = 2 oppure a = 3 allora b = 4 oppure b = 6 */
   4: b ← 3*a        /* se a = 3 allora b sarà uguale a 12 */
ALTRIMENTI:
   b ← a – 10        /* se a assume qualunque valore diverso da 1, 2, 3 e 4
                        verrà visualizzato il valore di a diminuito di 10 unità */
FINE CASO
Scrivi(b)
FINE
```

FLOWCHART



CODIFICA C

- **L'istruzione (o costrutto) di SELEZIONE a PIU' VIE o N-ARIA**

esempio-main.c

```c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
int a, b;                               // a,b : INT
                                        // INIZIO
scanf("%d", &a);                        // Leggi(a)

switch (a)                              // NEL CASO CHE (a) SIA
   {                                    //
   case 1:                              // 1 :b <-- a
      {
      b = a;
      break;
      }
   case 2:                              // 2 :
   case 3:                              // 3 : b <-- 2*a
      {
      b = 2*a;
      break;
      }
   case 4:                              // 4: b <-- 3*a
      {
      b = 3*a;
      break;
      }
   default:                             // ALTRIMENTI: b <-- a - 10
      {
      b = a - 10;
      }
   }                                    // FINE CASO

printf("%d\n", b);                      // Scrivi(b)

system("PAUSE");
return 0;
}                                       // FINE
```

AMBIENTE GLOBALE

AMBIENTE LOCALE

B. ISTRUZIONI DI CONTROLLO (ALGORITMO)
3. L'istruzione (o COSTRUTTO) di ITERAZIONE

PREMESSA

L'istruzione (o costrutto) di **ITERAZIONE** ci permette di eseguire più volte un blocco di istruzioni (sempice o complesso) finchè non si verifica una determinata condizione.

Esistono **TRE TIPOLOGIE** di **CICLI** che posso **CLASSIFICARE** secondo **TRE PUNTI DI VISTA**

1° **Rispetto a DOVE SI VERIFICA la condizione logica ossia in altre parole il TEST**

- L'istruzione (o costrutto) di ITERAZIONE MENTRE
- L'istruzione (o costrutto) di ITERAZIONE PER
- L'istruzione (o costrutto) di ITERAZIONE RIPETI

→ **CICLI PRECONDIZIONALI (o con controllo in TESTA)**

→ **CICLI POSTCONDIZIONALI (o con controllo in CODA)**

2° **Rispetto alla LOGICA DI CICLO (ossia per quale valore di verità ciclano e per quali no)**

- L'istruzione (o costrutto) di ITERAZIONE MENTRE
- L'istruzione (o costrutto) di ITERAZIONE PER
- L'istruzione (o costrutto) di ITERAZIONE RIPETI

→ **CICLA x VERO (esce per FALSO)**

→ **CICLA x FALSO (esce per VERO)**

3° **Rispetto alla possibilità di CONOSCERE A PRIORI IL NUMERO DI ITERAZIONI**

- L'istruzione (o costrutto) di ITERAZIONE MENTRE
- L'istruzione (o costrutto) di ITERAZIONE RIPETI
- L'istruzione (o costrutto) di ITERAZIONE PER

→ **NON DEVO conoscere A PRIORI il numero di iterazioni necessario. Possono essere usati IN QUALUNQUE CASO**

→ **DEVO conoscere A PRIORI il numero di iterazioni necessario. Possono essere usati SOLO IN QUESTO CASO**

B. ISTRUZIONI DI CONTROLLO (ALGORITMO)

3. L'istruzione (o COSTRUTTO) di ITERAZIONE MENTRE

PSEUDODODIFICA

FLOWCHART

- **L'istruzione (o costrutto) di ITERAZIONE MENTRE**

LOGICA FUNZIONAMENTO

MENTRE il valore di verità della <condizione> risulta
- VERO esegui il blocco di istruzioni <B1>
- FALSO esci dal ciclo.

********Come va scritta:

MENTRE (<condizione>) ESEGUI
<B1>
FINE MENTRE

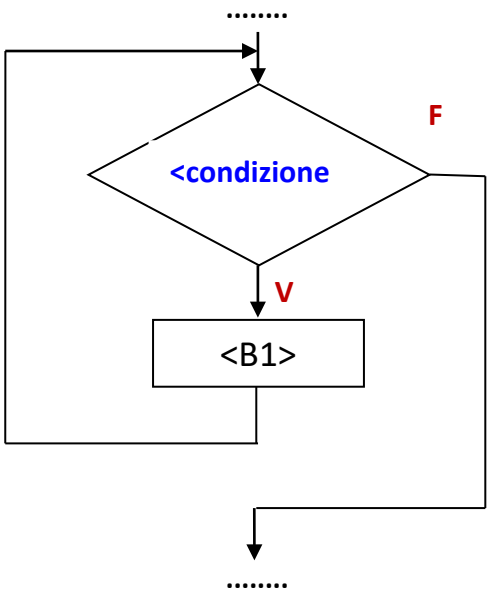
********NOTE:

- La condizione nel ciclo MENTRE viene eseguita in testa (CICLO PRECONDIZIONALE)

- Il ciclo MENTRE cicla per condizione VERA e termina per condizione FALSA

- Il blocco di istruzioni all'interno del ciclo MENTRE PUO' ANCHE NON ESSERE MAI ESEGUITO (se la condizione risulta inizialmente FALSA)

- E' sottinteso che qualcuna delle istruzioni del blocco <B1> dovrà cambiare il valore di verità della condizione del ciclo MENTRE per evitare il LOOP INFINITO



B. ISTRUZIONI DI CONTROLLO (PROGRAMMA SORGENTE)
3. L'istruzione (o COSTRUTTO) di ITERAZIONE MENTRE

↓

CODIFICA C

- **L'istruzione (o costrutto) di ITERAZIONE MENTRE**

[*] esempio-main.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(int argc, char *argv[])
5  {
6
7  .....
8
9  while (<condizione>)
10    {
11    <B1>;
12    }
13
14 .....
15
16 system("PAUSE");
17 return 0;
18 }
```

AMBIENTE GLOBALE

AMBIENTE LOCALE

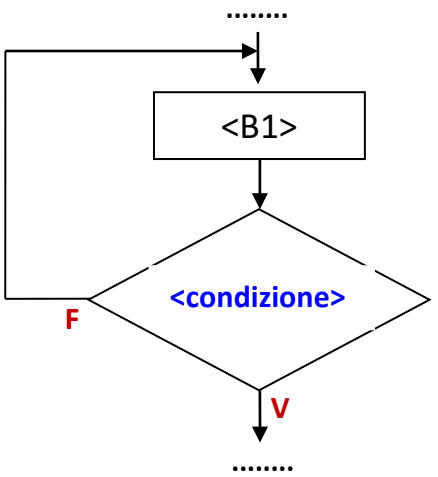
B. ISTRUZIONI DI CONTROLLO (ALGORITMO)
3. L'istruzione (o COSTRUTTO) di ITERAZIONE RIPETI

PSEUDODODIFICA

FLOWCHART

- **L'istruzione (o costrutto) di ITERAZIONE RIPETI**

LOGICA FUNZIONAMENTO
Esegui il blocco di istruzioni <B1>
- se la condizione è FALSA RIPETI l'esecuzione del blocco <B1>
- se la condizione è VERA esci dal ciclo



********Come va scritta:

RIPETI
<B1>
FINCHE' (<condizione>)

********NOTE:

- La condizione nel ciclo RIPETI viene eseguita in coda (CICLO POSTCONDIZIONALE)

- Il ciclo RIPETI cicla per condizione FALSA e termina per condizione VERA

- Il blocco di istruzioni all'interno del ciclo RIPETI VIENE ESEGUITO ALMENO UNA VOLTA

- E' sottinteso che qualcuna delle istruzioni del blocco <B1> dovrà cambiare il valore di verità della condizione del ciclo RIPETI per evitare il LOOP INFINITO

B. ISTRUZIONI DI CONTROLLO (PROGRAMMA SORGENTE)
3. L'istruzione (o COSTRUTTO) di ITERAZIONE RIPETI

↓

CODIFICA C

- **L'istruzione (o costrutto) di ITERAZIONE RIPETI**

[*] esempio-main.c

```
1   #include <stdio.h>
2   #include <stdlib.h>
3
4   int main(int argc, char *argv[])
5   {
6
7   .....
8
9   do
10     {
11     <B1>;
12     }
13  while (!<condizione>);
14  .....
15
16  system("PAUSE");
17  return 0;
18  }
```

AMBIENTE GLOBALE

AMBIENTE LOCALE

PSEUDODODIFICA

FLOWCHART

➢ L'istruzione (o costrutto) di ITERAZIONE PER

PREMESSA

Per gestire un ciclo PER occorre una variabile <indice> di tipo INTERO (detta CONTATORE)

I valori <inizio> e <fine> devono essere anch'essi INTERI (POSITIVI ma anche NEGATIVI)

Possono essere indicati attraverso variabili intere a patto che I LORO VALORI NON VENGANO IN NESSUN MODO MODIFICATI dalle istruzioni contenute nel blocco <B1>

La condizione sottointesa del ciclo PER viene ESEGUITA IN TESTA (CICLO PRECONDIZIONALE)

Il ciclo PER cicla per condizione VERA e termina per condizione FALSA

Il blocco di istruzioni all'interno del ciclo PER PUO' NON ESSERE MAI ESEGUITO (se la condizione è inizialmente FALSA)

********Come va scritta: (PER CRESCENTE)

PER <indice> ← <inizio> A <fine> ESEGUI
  <B1>
  <indice> ← <indice> + 1
FINE PER

LOGICA FUNZIONAMENTO
Esegui il blocco di istruzioni <B1> ESATTAMENTE
|<fine> - <inizio>| + 1 VOLTE

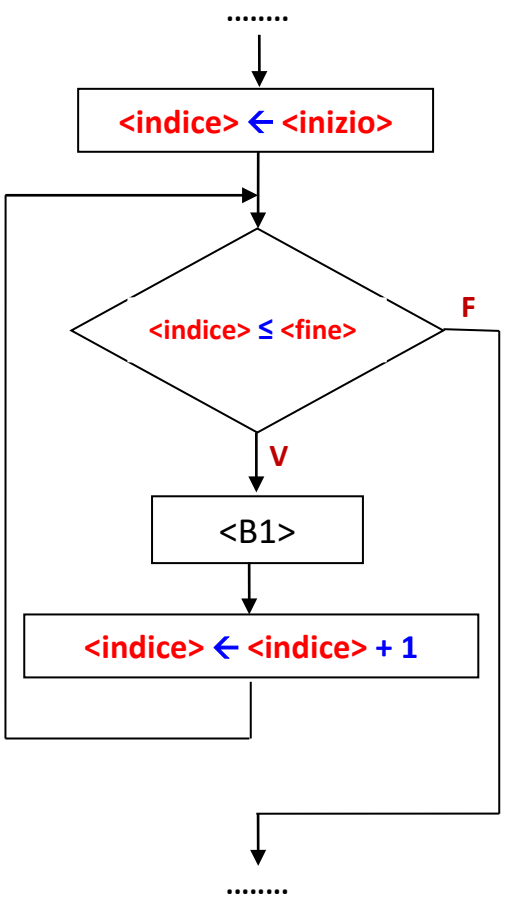
********NOTE:

- L'inizializzazione della variabile <indice> VIENE EFFETTUATA UNA VOLTA SOLA PRIMA che si inizi il ciclo

- Affinchè il ciclo PER inizi, il valore <inizio> deve essere MINORE o UGUALE al valore <fine>

- Condizione di ciclo PER CRESCENTE risulta essere <indice> ≤ <fine>

PER CRESCENTE



********Come va scritta: (PER DECRESCENTE)

PER <indice> ← <inizio> INDIETRO A <fine> ESEGUI
<B1>
<indice> ← <indice> - 1
FINE PER

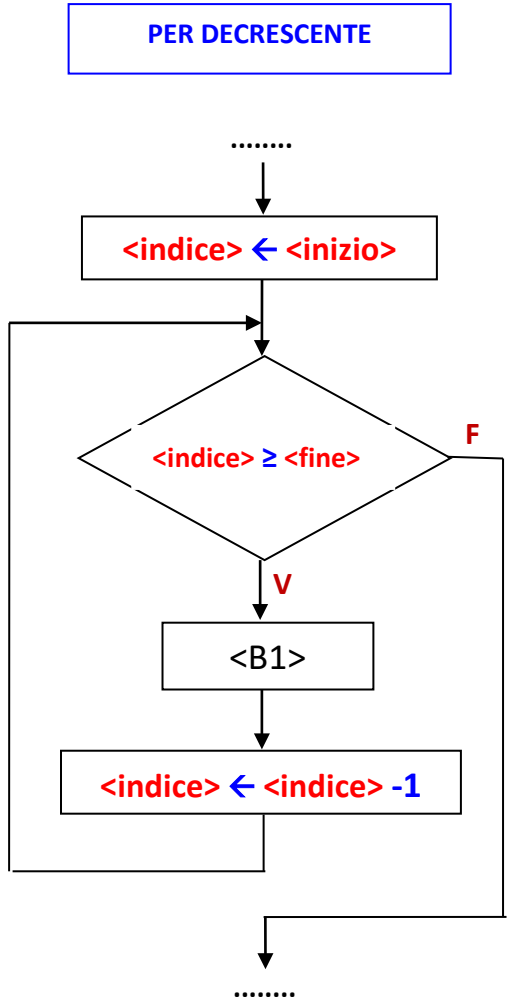
LOGICA FUNZIONAMENTO
Esegui il blocco di istruzioni <B1> ESATTAMENTE
| <inizio> - <fine> | + 1 VOLTE

********NOTE:

- L'inizializzazione della variabile <indice> VIENE EFFETTUATA UNA VOLTA SOLA PRIMA che si inizi il ciclo

- Affinchè il ciclo PER inizi, il valore <inizio> deve essere MAGGIORE o UGUALE al valore <fine>

- Condizione di ciclo Per PER DECRESCENTE risulta essere <indice> ≥ <fine>)



B. ISTRUZIONI DI CONTROLLO (PROGRAMMA SORGENTE)
3. L'istruzione (o COSTRUTTO) di ITERAZIONE PER

↓

CODIFICA C

➢ L'istruzione (o costrutto) di ITERAZIONE PER

[*] esempio-main.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(int argc, char *argv[])
5  {
6  int <indice>;
7
8  .....
9  //PER CRESCENTE
10 for (<indice> = <inizio>; <indice> <= <fine>; <indice>++)
11    {
12    <B1>;
13    }
14 .....
15
16 .....
17 //PER DECRESCENTE
18 for (<indice> = <inizio>; <indice> >= <fine>; <indice>--)
19    {
20    <B1>;
21    }
22 .....
23
24 system("PAUSE");
25 return 0;
26 }
```

Dichiarazione indice

AMBIENTE GLOBALE

AMBIENTE LOCALE

Inizializzazione

test

incremento

Forma alternativa
<indice> = <indice> + 1

Inizializzazione

test

decremento

Forma alternativa
<indice> = <indice> - 1

PRIMO ESEMPIO

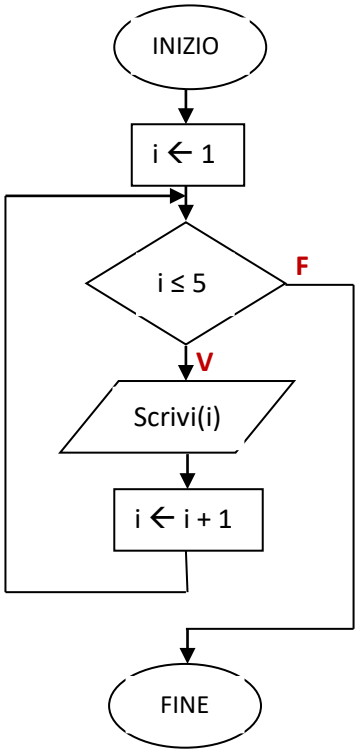
- **L'istruzione (o costrutto) di ITERAZIONE MENTRE**

PSEUDODODIFICA

Algoritmo che mostra a video i primi 5 numeri naturali

```
ALGORITMO A1
PROCEDURA main ( )
i: INT
INIZIO

/* inizializzazione variabili di controllo del ciclo */
i ← 1
/* esecuzione condizione del ciclo IN TESTA*/
MENTRE (i ≤ 5) ESEGUI
   Scrivi(i)
   /* incremento variabile di controllo del ciclo */
   i ← i + 1
FINE MENTRE
FINE
```

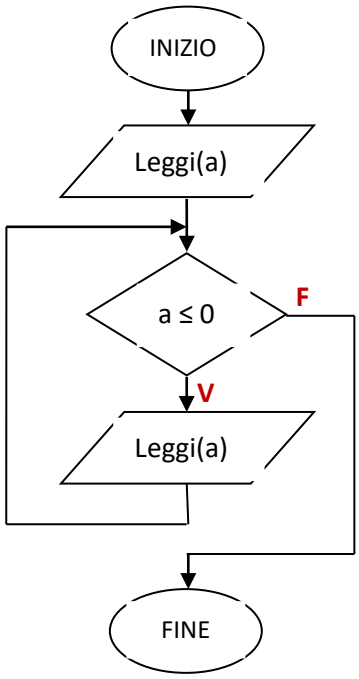
FLOWCHART



Algoritmo che esegue il controllo dell'ammissibilità dei valori che possono essere assegnati ad una variabile

In questo caso la variabile intera **a** si suppone debba essere strettamente positiva

```
ALGORITMO A2
PROCEDURA main ( )
a: INT
INIZIO
/* N.B. prima lettura variabile a IMPORTANTISSIMA*/
Leggi(a)
/* esecuzione condizione  su valori di a non ammessi*/
MENTRE (a ≤ 0) ESEGUI
   /* Ulteriore lettura variabile a */
   Leggi(a)
FINE MENTRE
FINE
```



CODIFICA C

- **L'istruzione (o costrutto) di ITERAZIONE MENTRE**

PROGRAMMA A1: che mostra a video i primi 5 numeri naturali

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])  //PROCEDURA main()
{
int i;                            // i : INT
                                  // INIZIO
i = 1;                            // i <-- 1
while (i <= 5)                    // MENTRE (i <= 5)
   {
   printf("%d\n", i);             //    Scrivi(i)
   i = i + 1;                     //    i <-- i + 1
   }                              // FINE MENTRE
system("PAUSE");
return 0;
}                                 // FINE
```

PROGRAMMA A2 che esegue il controllo dell'ammissibilità dei valori che possono essere assegnati ad una variabile. N.B. In questo caso la variabile intera **a** si suppone debba essere strettamente positiva

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])  //PROCEDURA main()
{
int a:                            // a : INT
                                  // INIZIO
fflush(stdin);
scanf("%d", &a);                  // Leggi(a)
while (a <= 0)                    // MENTRE (a < = 0) ESEGUI
   {
   fflush(stdin);
   scanf("%d", &a);               //     Leggi(a)
   }                              // FINE MENTRE

system("PAUSE");
return 0;
}                                 // FINE
```

SECONDO ESEMPIO

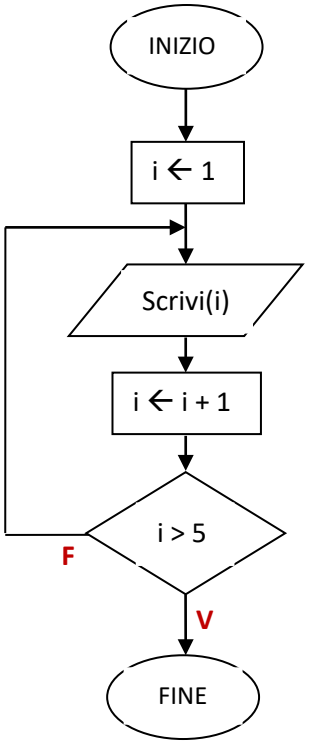
- **L'istruzione (o costrutto) di ITERAZIONE RIPETI**

PSEUDODODIFICA

Algoritmo che mostra a video i primi 5 numeri naturali

```
ALGORITMO B1
PROCEDURA main ( )
i: INT
INIZIO

/* inizializzazione variabili di controllo del ciclo */
i ← 1
/* esecuzione condizione del ciclo IN CODA*/
RIPETI
   Scrivi(i)
   /* incremento variabile di controllo del ciclo */
   i ← i + 1
FINCHE' ( i > 5)
FINE
```

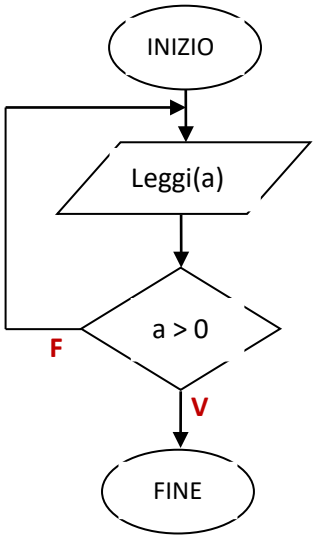
FLOWCHART



Algoritmo che esegue il controllo dell'ammissibilità dei valori che possono essere assegnati ad una variabile

In questo caso la variabile intera **a** si suppone debba essere strettamente positiva

```
ALGORITMO B2
PROCEDURA main ( )
a: INT
INIZIO

/* esecuzione condizione su valori di a non ammessi*/
RIPETI
   /* Lettura variabile a */
   Leggi(a)
FINCHE' ( a > 0 )
FINE
```



➢ L'istruzione (o costrutto) di ITERAZIONE RIPETI

PROGRAMMA B1: che mostra a video i primi 5 numeri naturali

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])  //PROCEDURA main()
{
int i;                            // i : INT
                                  // INIZIO

i = 1;                            // i <-- 1

do                                // RIPETI
   {
   printf("%d\n", i);             //    Scrivi(i)
   i = i + 1;                     //    i <-- i + 1
   }
while (i <= 5);                   // FINE RIPETI

system("PAUSE");
return 0;
}                                 // FINE
```

PROGRAMMA B2 che esegue il controllo dell'ammissibilità dei valori che possono essere assegnati ad una variabile. N.B. In questo caso la variabile intera **a** si suppone debba essere strettamente positiva

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])  //PROCEDURA main()
{
int a;                            // a : INT
                                  // INIZIO
do                                // RIPETI
   {
   fflush(stdin);
   scanf("%d", &a);               //    Leggi(a)
   }
while (a <= 0);                   // FINE RIPETI

system("PAUSE");
return 0;
}                                 // FINE
```

TERZO ESEMPIO

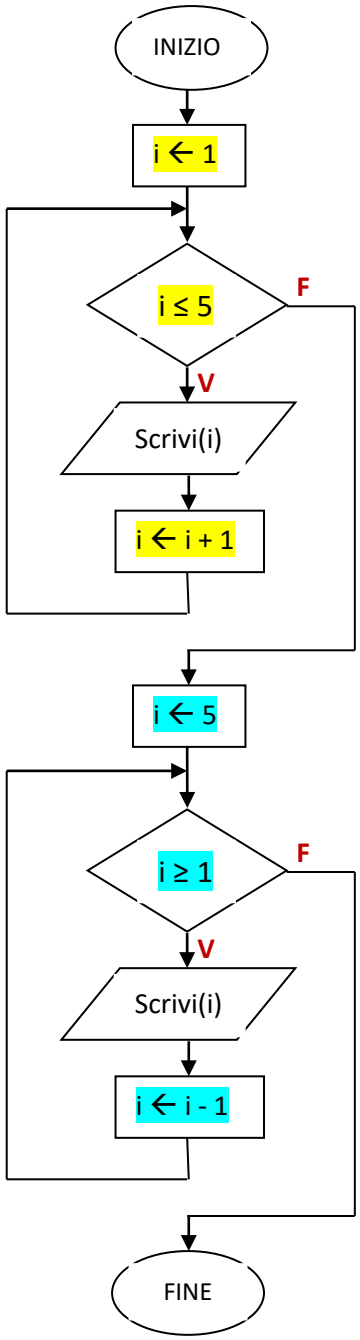
➢ L'istruzione (o costrutto) di ITERAZIONE PER

PSEUDODODIFICA

Algoritmo che mostra a video i primi 5 numeri naturali (in senso CRESCENTE e poi DECRESCENTE)

```
ALGORITMO C1
PROCEDURA main ( )
i: INT
/* PER CRESCENTE */
/* i ← 1 inizializzazione CONTATORE – TEST (i ≤ 5) */
PER i ← 1 A 5 ESEGUI
   Scrivi(i)
   i ← i + 1              /* incremento CONTATORE */
FINE PER

/* PER DECRESCENTE */
/* i ← 5 inizializzazione CONTATORE – TEST (i ≥ 1) */
PER i ← 5 INDIETRO A 1 ESEGUI
   Scrivi(i)
   i ← i - 1              /* decremento CONTATORE */
FINE PER
FINE
```

FLOWCHART



CODIFICA C

➢ L'istruzione (o costrutto) di ITERAZIONE PER

PROGRAMMA C1: che mostra a video i primi 5 numeri naturali (in senso CRESCENTE e poi DECRESCENTE)

```c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) // PROCEDURA main()
{                                // INIZIO
int i;

//for CRESCENTE
for(i=1; i<=5; i++)              // PER i <-- 1 A 5 ESEGUI
   {
   printf("%d\n", i);            //    Scrivi(i)
   }                             //    i <-- i + 1
                                 // FINE PER

//for DECRESCENTE
for(i=5; i>=1; i--)              // PER i <-- 5 INDIETRO A 1 ESEGUI
   {
   printf("%d\n", i);            //    Scrivi(i)
   }                             //    i <-- i - 1
                                 // FINE PER

system("PAUSE");
return 0;
}                                // FINE
```

Cosa verrà mostrato a video in questo caso?

```c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) // PROCEDURA main()
{                                // INIZIO
int i;

//for DECRESCENTE
for(i=5; i>=1; i--)
   {
   printf("%d\n", 6-i);
   }

system("PAUSE");
return 0;
}                                // FINE
```