capitolo 11 – La programmazione strutturata

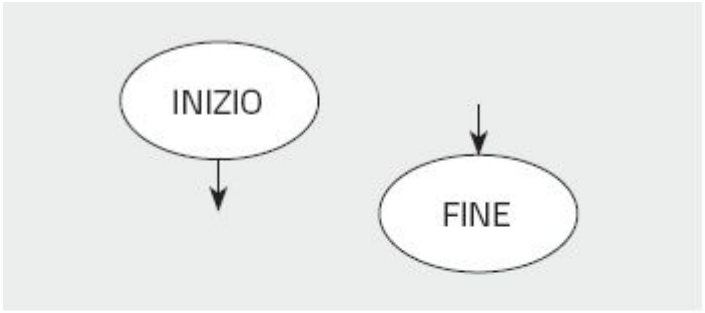
1. Gli algoritmi e i diagrammi di flusso
2. Le strutture fondamentali dei programmi
3. La struttura condizionale
4. Le strutture iterative
5. La tabella di traccia

11.1 – Gli algoritmi e i diagrammi di flusso

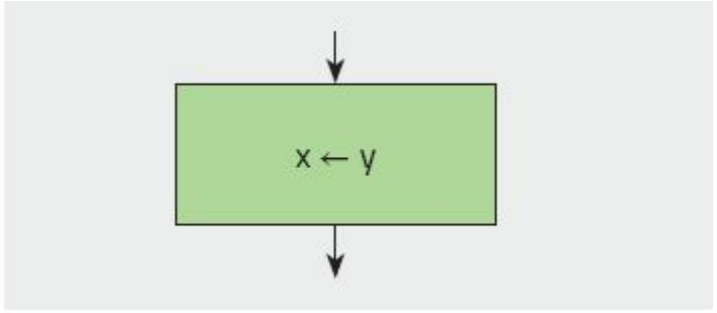
Il diagramma di flusso (o a blocchi) rappresenta graficamente un algoritmo.

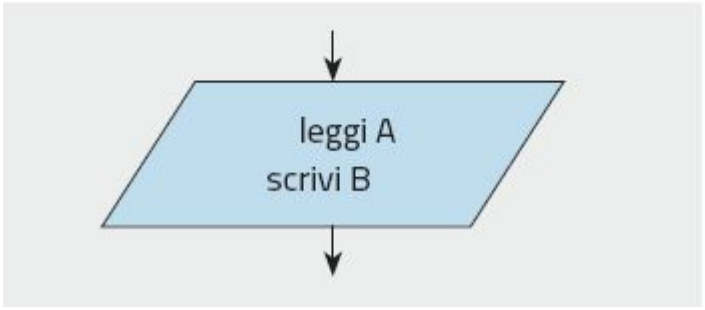
Qualsiasi algoritmo si può rappresentare usando solo **quattro tipi di blocchi**:

OVALE
blocco iniziale e finale



RETTANGOLO
blocco di azione/elaborazione





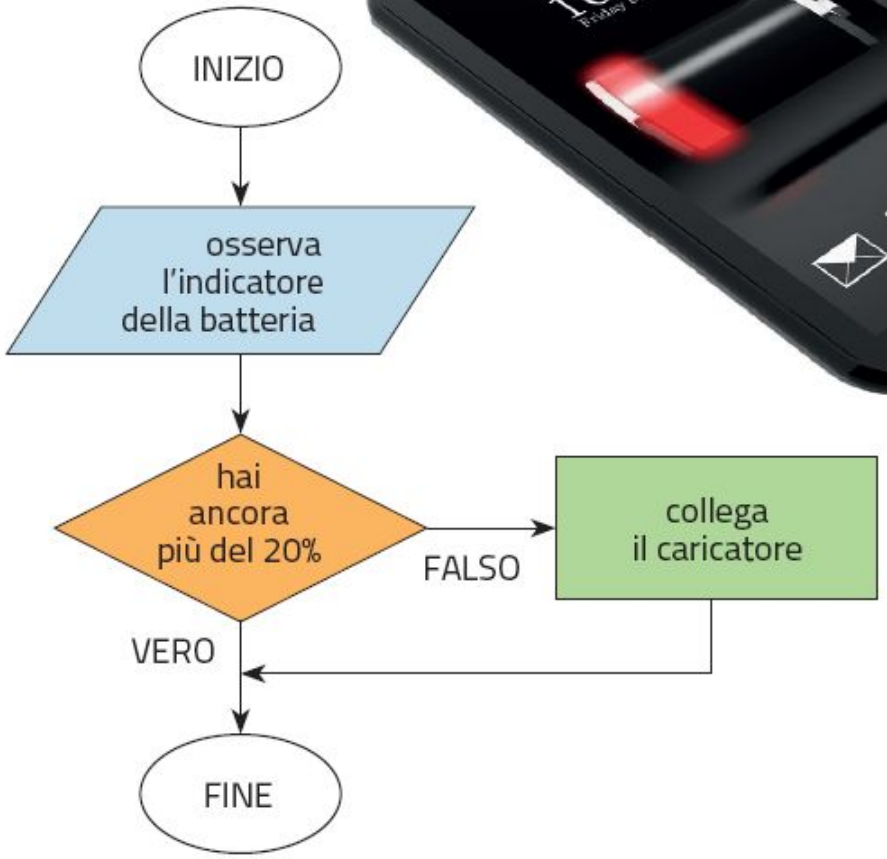
PARALLELOGRAMMA
blocco di lettura/scrittura

ROMBO
blocco di controllo/test

11.1 – Gli algoritmi e i diagrammi di flusso

Il diagramma di flusso (o a blocchi) rappresenta graficamente un algoritmo.

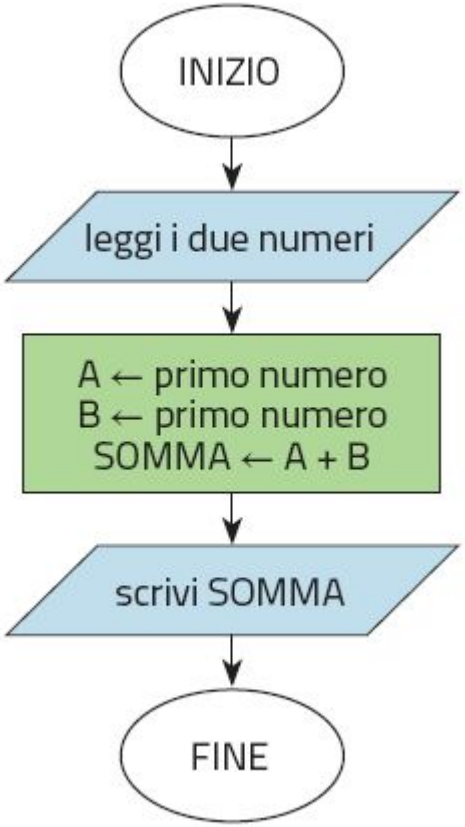
Esempio: un algoritmo per la messa in carica di un telefonino



11.1 – Gli algoritmi e i diagrammi di flusso

Il diagramma di flusso (o a blocchi) rappresenta graficamente un algoritmo.

Esempio: un semplice algoritmo matematico per la **somma di due numeri** dati come input



1. il programma inizia;
2. il programma legge i due numeri;
3. chiama i due numeri rispettivamente A e B;
4. somma i loro valori e chiama il risultato SOMMA;
5. produce come output il valore di SOMMA;
6. il programma termina.

- nel programma A, B e SOMMA sono **variabili**
- il simbolo ← indica l'**assegnazione** del loro valore

11.2 – Le strutture fondamentali dei programmi

La programmazione strutturata usa tre sole strutture di base: la sequenza di istruzioni, la selezione (struttura condizionale) e il ciclo (struttura iterativa).

La **programmazione strutturata** si basa sul **teorema di Böhm-Jacopini**.

Questo teorema implica che **qualsiasi algoritmo si può esprimere usando soltanto le tre strutture fondamentali:**

- la **sequenza**, di istruzioni;
- la **selezione** o *struttura condizionale*;
- il **ciclo** o *struttura iterativa*.

11.2 – Le strutture fondamentali dei programmi

La programmazione strutturata usa tre sole strutture di base: la sequenza di istruzioni, la selezione (struttura condizionale) e il ciclo (struttura iterativa).

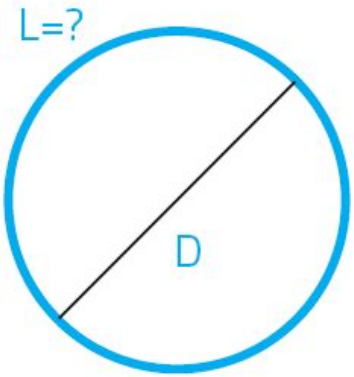
Nella **struttura sequenziale** le istruzioni sono eseguite una dopo l'altra nell'ordine in cui le ha scritte il programmatore.

Le **istruzioni** possono essere di tre tipi:

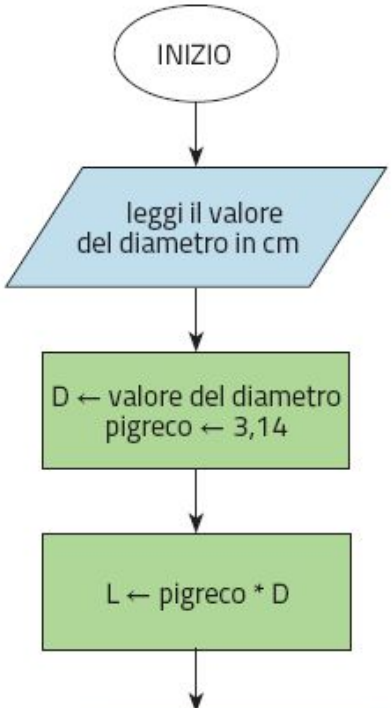
- **lettura**: legge il valore di una variabile da un'unità di input;
- **scrittura**: invia il contenuto di una variabile a un'unità di output;
- **assegnazione**: fa assumere a una variabile il valore ottenuto con un calcolo.

11.2 – Le strutture fondamentali dei programmi

La sequenza è la struttura più semplice: le istruzioni sono eseguite una dopo l'altra nell'ordine in cui le ha scritte il programmatore.



Esempio: dato il diametro **D**, trovare la lunghezza **L** della circonferenza.



1. Leggi il numero di input.

2. Assegna il suo valore a **D**.

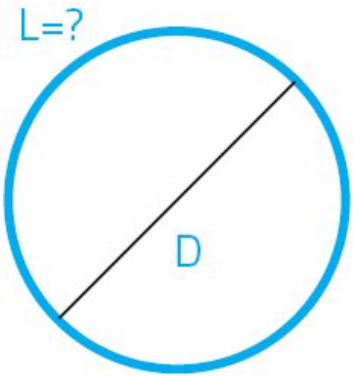
3. Assegna a **pigreco** il valore 3,14.

4. Assegna a **L** il valore trovato moltiplicando **pigreco** per **D**.

5. Scrivi un messaggio di output che contiene il valore finale di **L**.

11.2 – Le strutture fondamentali dei programmi

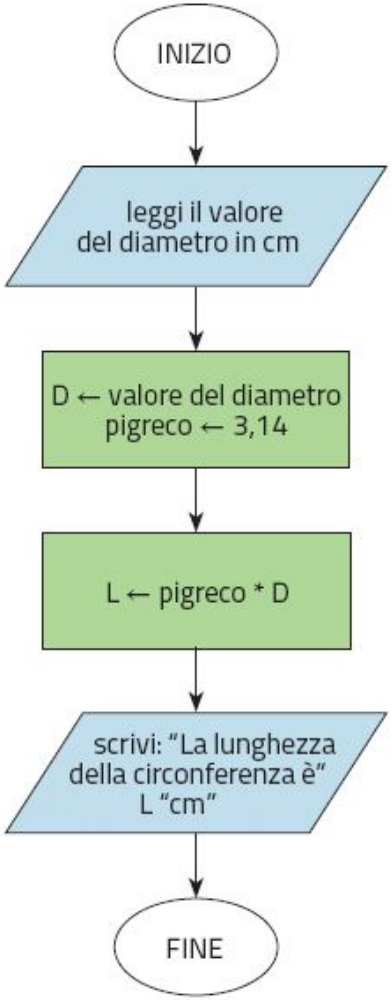
La sequenza è la struttura più semplice: le istruzioni sono eseguite una dopo l'altra nell'ordine in cui le ha scritte il programmatore.



Esempio: dato il diametro **D**, trovare la lunghezza **L** della circonferenza.

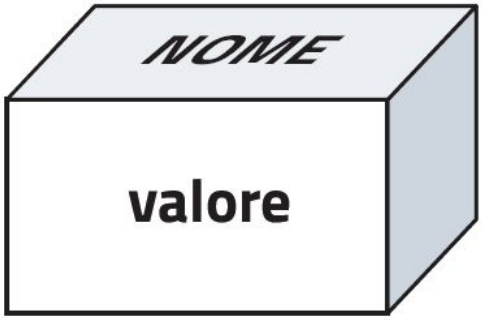
- **D** e **L** sono **variabili**, ossia **spazi di memoria RAM** in cui è registrato un **valore che può cambiare** durante l'esecuzione del programma.

- **pigreco** invece è una **costante**, ossia uno spazio di memoria che registra un **valore che non cambia**.



11.2 – Le strutture fondamentali dei programmi

Una variabile è come una «scatola di memoria» su cui è scritto un nome, mentre il valore della variabile è il contenuto della scatola.



Nei programmi per computer **ciò che importa di una variabile è il valore** che essa ha in ogni dato momento durante l'esecuzione.

All'inizio di ogni algoritmo bisogna **inizializzare le variabili**, cioè far sì che abbiano un valore (introdotto come input o assegnato con un'istruzione).

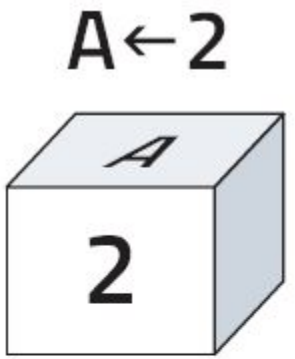
Il valore di una variabile può poi cambiare a ogni esecuzione dell'algoritmo, o anche nel corso dell'algoritmo stesso.

Una **costante** invece, una volta inizializzata, **non subisce mai modifiche**.

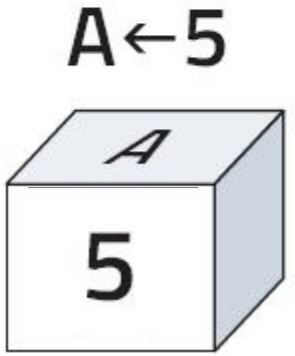
11.2 – Le strutture fondamentali dei programmi

Una variabile è come una «scatola di memoria» su cui è scritto un nome, mentre il valore della variabile è il contenuto della scatola.

Il simbolo ← indica l'operazione di **assegnazione**: alla variabile posta a sinistra della freccia si assegna il valore che si trova a destra della freccia.



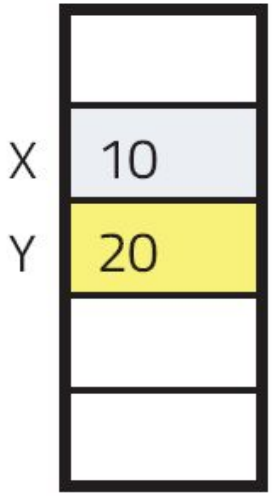
Assegnare un nuovo valore a una variabile significa **sovrascrivere** il valore che era presente nel suo spazio di memoria.



11.2 – Le strutture fondamentali dei programmi

Una variabile è come una «scatola di memoria» su cui è scritto un nome, mentre il valore della variabile è il contenuto della scatola.

Come si deve procedere, per esempio, se si vogliono **scambiare tra loro i valori delle due variabili X e Y** (*swapping*) nella memoria RAM del computer?

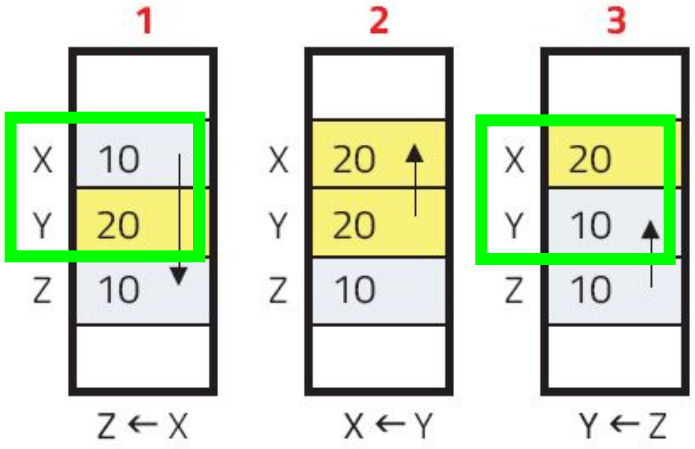


Bisogna fare ricorso a una terza **variabile ausiliaria Z**.

1. prima si assegna a **Z** il valore di **X**;

2. poi si assegna a **X** il valore di **Y**;

3. infine si assegna a **Y** il valore di **Z**.



11.3 – La struttura condizionale

La selezione o struttura condizionale fa eseguire istruzioni diverse a seconda che una condizione sia vera oppure falsa.

Il risultato del **test**, o **selezione**, dice al computer quale tra le due strade alternative seguire nei passaggi successivi del programma.

Nei diagrammi di flusso la struttura condizionale si rappresenta con il **rombo**.

Nei linguaggi di programmazione la struttura condizionale si esprime usando le parole-chiave **if** («se la condizione è vera…») ed **else** («altrimenti…»).

11.3 – La struttura condizionale

La selezione o struttura condizionale fa eseguire istruzioni diverse a seconda che una condizione sia vera oppure falsa.

Esempio: calcolare il **valore assoluto** di un numero

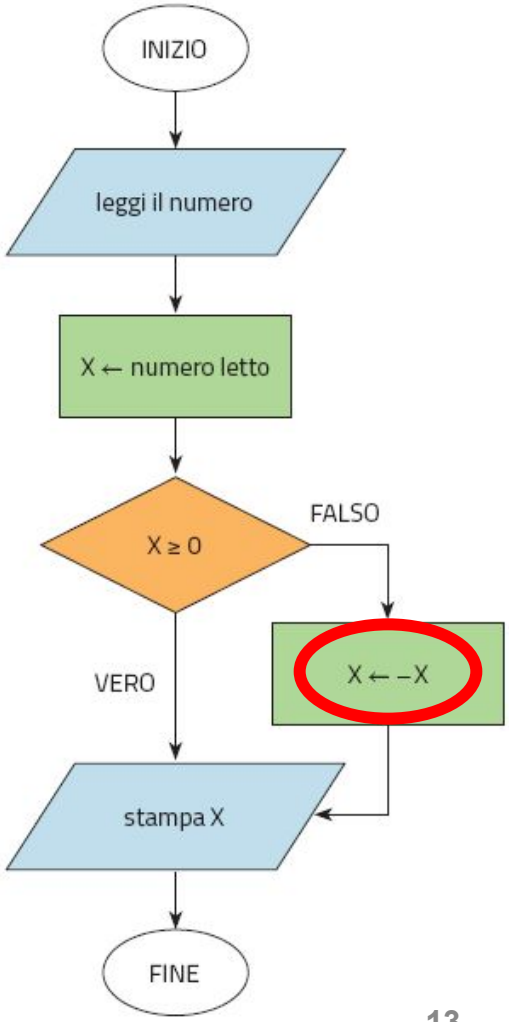
$|X| = X$ se $X \geq 0$, mentre $|X| = -X$ se $X < 0$

1. Leggi il numero e assegna il suo valore a X.

2. Se X è maggiore o uguale a zero, mostra come risultato il valore di X.

3. Altrimenti, cioè se invece X è minore di zero, mostra come risultato il valore di −X.

NB: nel caso FALSO si stampa X dopo averne cambiato il valore, assegnandogli quello di −X.



11.3 – La struttura condizionale

Esempio: risolvere un'**equazione di 2º grado** che ha la forma $ax^2 + bx + c = 0$

soluzioni (o radici):

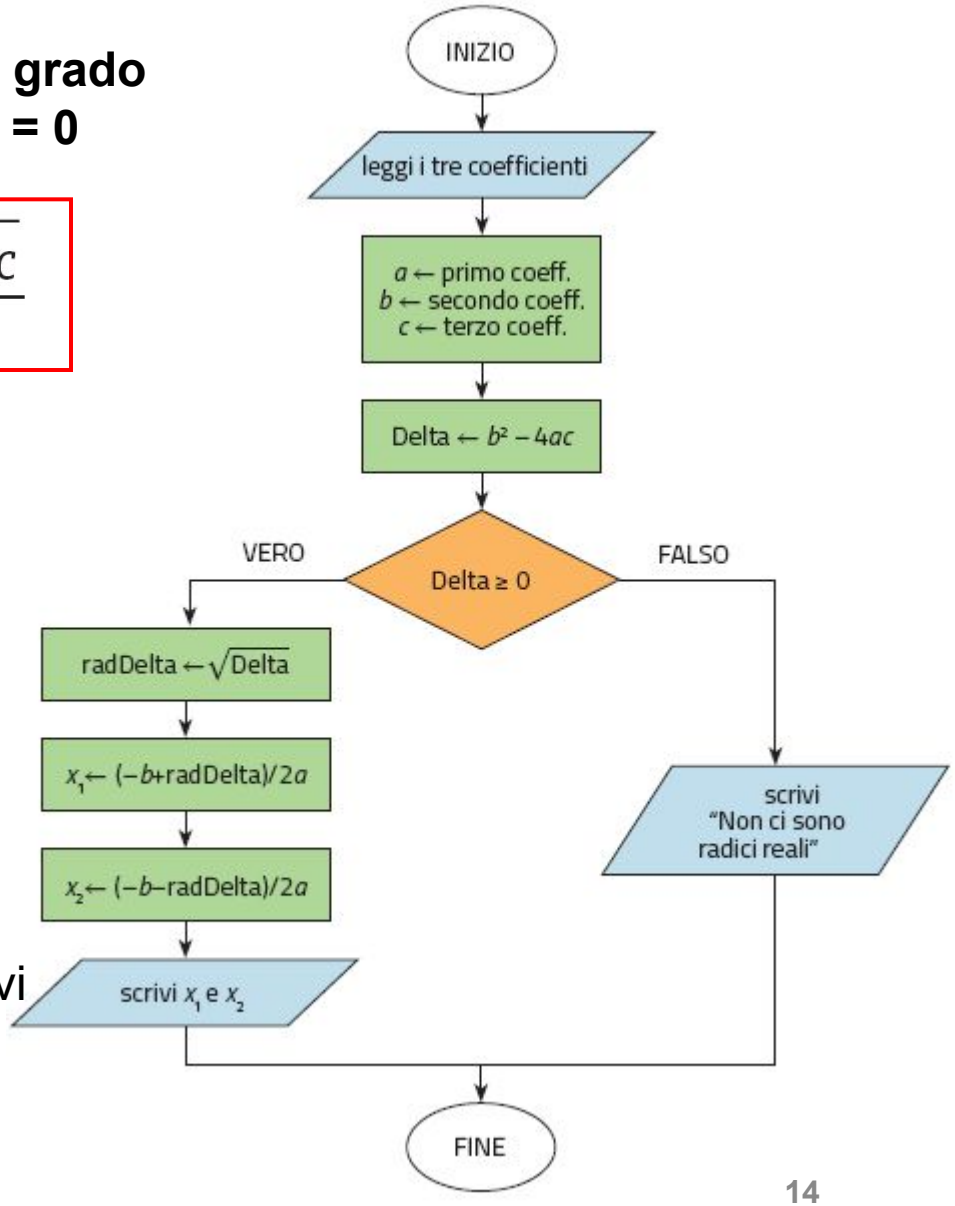
$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

1. Leggi i tre coefficienti *a*, *b* e *c*.

2. Calcola il discriminante $\Delta = b^2 - 4ac$.

3. **Se** Δ è maggiore o uguale a zero, calcola con la formula le due soluzioni, cioè x_1 e x_2, e producile come output.

4. **Altrimenti**, cioè se Δ è negativo, scrivi «Non ci sono radici reali».



11.3 – La struttura condizionale

La selezione o struttura condizionale fa eseguire istruzioni diverse a seconda che una condizione sia vera oppure falsa.

Il risultato del **test**, o **selezione**, dice al computer quale tra le due strade alternative seguire nei passaggi successivi del programma.

Nei diagrammi di flusso la struttura condizionale si rappresenta con il **rombo**.

Nei linguaggi di programmazione la struttura condizionale si esprime usando le parole-chiave **if** («se la condizione è vera…») ed **else** («altrimenti…»).

11.3 – La struttura condizionale

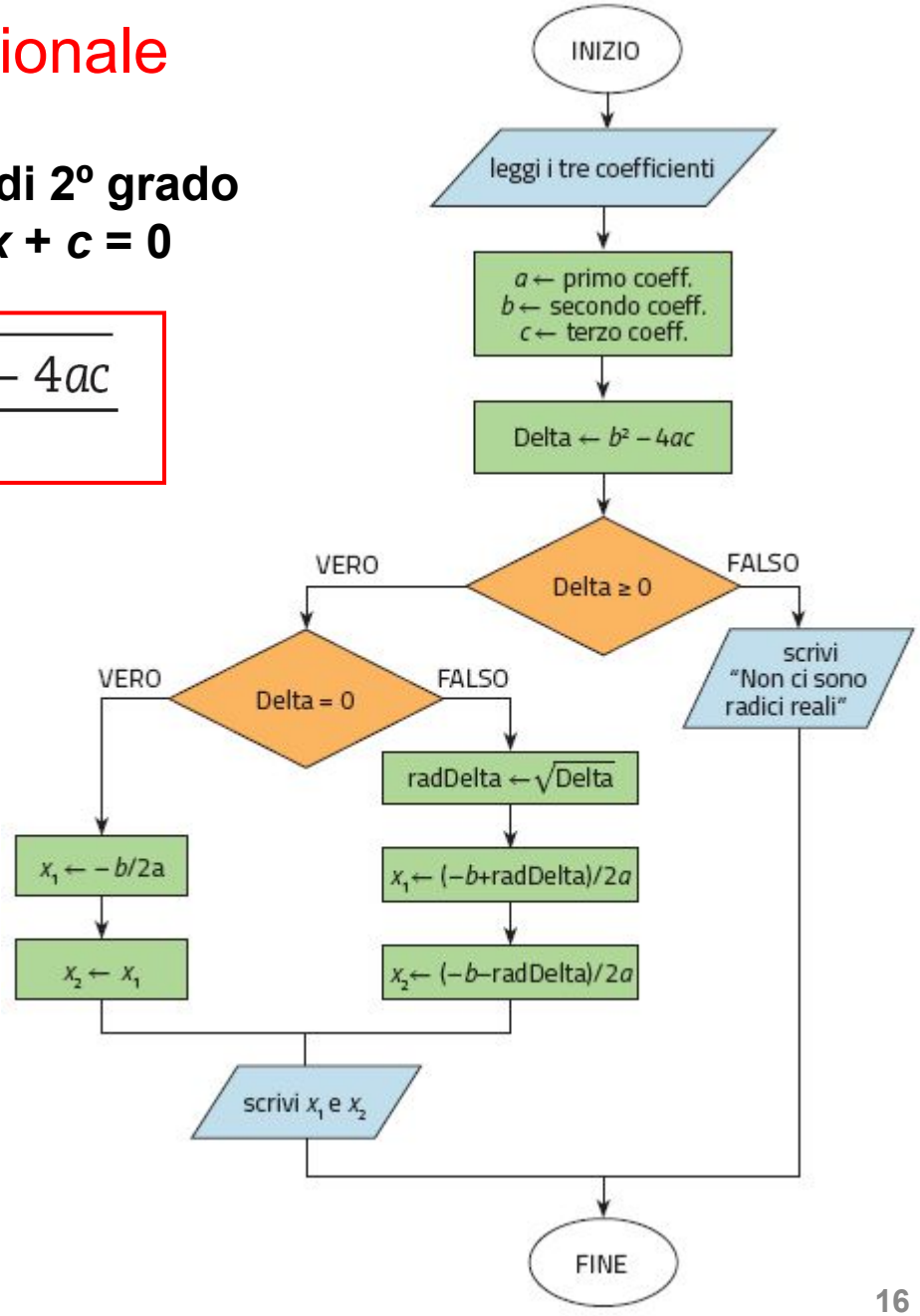
Esempio: risolvere un'**equazione di 2º grado** che ha la forma $ax^2 + bx + c = 0$

soluzioni (o radici):

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Versione alternativa dell'algoritmo, che esplicita tutti i casi significativi per il valore del discriminante Δ.

In questo caso ci sono due selezioni, con una **struttura nidificata** dentro l'altra.



11.3 – La struttura condizionale

La selezione o struttura condizionale fa eseguire istruzioni diverse a seconda che una condizione sia vera oppure falsa.

Il risultato del **test**, o **selezione**, dice al computer quale tra le due strade alternative seguire nei passaggi successivi del programma.

Nei diagrammi di flusso la struttura condizionale si rappresenta con il **rombo**.

Nei linguaggi di programmazione la struttura condizionale si esprime usando le parole-chiave **if** («se la condizione è vera…») ed **else** («altrimenti…»).

11.3 – La struttura condizionale

La selezione o struttura condizionale fa eseguire istruzioni diverse a seconda che una condizione sia vera oppure falsa.

Il risultato del **test**, o **selezione**, dice al computer quale tra le due strade alternative seguire nei passaggi successivi del programma.

Nei diagrammi di flusso la struttura condizionale si rappresenta con il **rombo**.

Nei linguaggi di programmazione la struttura condizionale si esprime usando le parole-chiave **if** («se la condizione è vera…») ed **else** («altrimenti…»).

11.3 – La struttura condizionale

La selezione o struttura condizionale fa eseguire istruzioni diverse a seconda che una condizione sia vera oppure falsa.

Il risultato del **test**, o **selezione**, dice al computer quale tra le due strade alternative seguire nei passaggi successivi del programma.

Nei diagrammi di flusso la struttura condizionale si rappresenta con il **rombo**.

Nei linguaggi di programmazione la struttura condizionale si esprime usando le parole-chiave **if** («se la condizione è vera…») ed **else** («altrimenti…»).

11.3 – La struttura condizionale

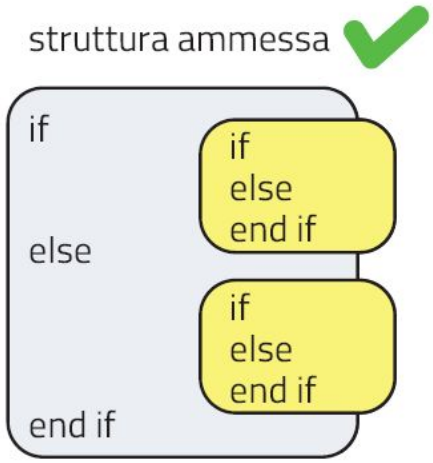
La selezione o struttura condizionale fa eseguire istruzioni diverse a seconda che una condizione sia vera oppure falsa.

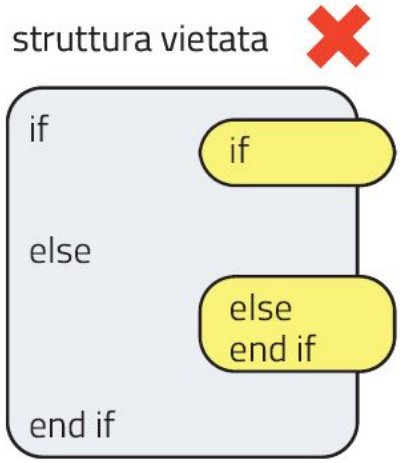
Nei linguaggi di programmazione la selezione si può scrivere così:

```
if (condizione da verificare)
        istruzioni da eseguire se la condizione è vera
else
        istruzioni da eseguire se la condizione è falsa
end if
```

Una selezione può contenere, fra le azioni condizionate, altre selezioni **nidificate** nella prima.

Attenzione però a nidificare le strutture in modo coerente dal punto di vista logico!





11.3 – La struttura condizionale

La selezione o struttura condizionale fa eseguire istruzioni diverse a seconda che una condizione sia vera oppure falsa.

Il risultato del **test**, o **selezione**, dice al computer quale tra le due strade alternative seguire nei passaggi successivi del programma.

Nei diagrammi di flusso la struttura condizionale si rappresenta con il **rombo**.

Nei linguaggi di programmazione la struttura condizionale si esprime usando le parole-chiave **if** («se la condizione è vera…») ed **else** («altrimenti…»).

11.3 – La struttura condizionale

Le condizioni semplici da controllare nei test di selezione si esprimono usando gli operatori relazionali tra le variabili.

Gli **operatori relazionali** sono:

=	uguale
≠	diverso
>	maggiore
<	minore
≥	maggiore o uguale
≤	minore o uguale

Esempio: la condizione **A > B** scritta in un blocco a forma di rombo significa «se è vero che il valore della variabile A è maggiore di quello della variabile B».

11.3 – La struttura condizionale

Nelle condizioni composte da controllare nei test di selezione si usano anche gli operatori logici (o booleani) AND e OR.

Se ognuna delle due condizioni C_1 e C_2 può essere vera (V) o falsa (F), allora gli operatori **AND** e **OR** sono definiti da queste **tabelle di verità**:

C_1	C_2	C_1 AND C_2
V	V	V
V	F	F
F	V	F
F	F	F

La condizione **C_1 AND C_2** è vera soltanto se sono vere *sia* C_1 *sia* C_2.

C_1	C_2	C_1 OR C_2
V	V	V
V	F	V
F	V	V
F	F	F

Invece, perché **C_1 OR C_2** sia vera, basta che sia vera *almeno una* tra C_1 e C_2.

La congiunzione **AND** impone vincoli più restrittivi rispetto alla disgiunzione **OR**.

11.3 – La struttura condizionale

Nelle condizioni composte da controllare nei test di selezione si usano anche gli operatori logici (o booleani) AND e OR.

C_1	C_2	C_1 AND C_2
V	V	V
V	F	F
F	V	F
F	F	F

C_1	C_2	C_1 OR C_2
V	V	V
V	F	V
F	V	V
F	F	F

Esempio:

«Rispondi se è un amico AND è pomeriggio.»

Se un amico ti chiama di mattina o di sera non risponderai, e anche di pomeriggio non risponderai, se chi chiama non è un amico.

«Rispondi se è un amico OR è pomeriggio.»

In questo caso risponderai molte più volte: sempre se è un amico e sempre di pomeriggio.

11.3 – La struttura condizionale

Nelle condizioni composte da controllare nei test di selezione si usano anche gli operatori logici (o booleani) AND, OR e NOT.

L'operatore logico **NOT** nega la condizione a cui è applicato, con questa **tabella di verità**:

C	NOT C
V	F
F	V

Esempio:

«Rispondi se NOT è mattina.»

significa che risponderai a tutte le chiamate, escluse soltanto quelle che arrivano al mattino.

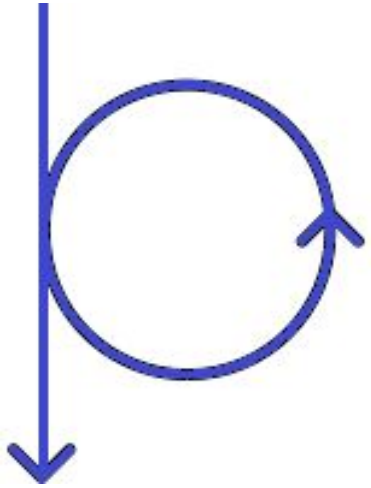
11.4 – Le strutture iterative

L'iterazione o ciclo è l'esecuzione ripetuta di un gruppo di istruzioni.

Con la **sequenza** e la **selezione**, l'**iterazione** è la terza struttura caratteristica della **programmazione strutturata**.

Un **algoritmo iterativo** richiede di **eseguire ripetutamente, in modo ciclico, uno stesso gruppo di istruzioni** con dati diversi.

L'iterazione può avvenire per un numero di volte predefinito oppure fino al verificarsi di una data condizione.



11.4 – Le strutture iterative

Il ciclo di tipo for permette di iterare per un numero prestabilito di volte.

```
for (condizione da verificare)
        istruzioni da eseguire se la condizione è vera
        poi torna all'istruzione for per l'iterazione successiva
end for
```

La **condizione** del ciclo dipende da una variabile intera detta **contatore**, perché conta il numero delle iterazioni eseguite.

Di solito la **variabile contatore** è chiamata **i** e il suo valore viene incrementato di un'unità a ogni esecuzione del ciclo.

NB: il controllo della condizione da soddisfare *precede* il gruppo delle istruzioni da iterare; può quindi accadere che le istruzioni del ciclo non siano mai eseguite.

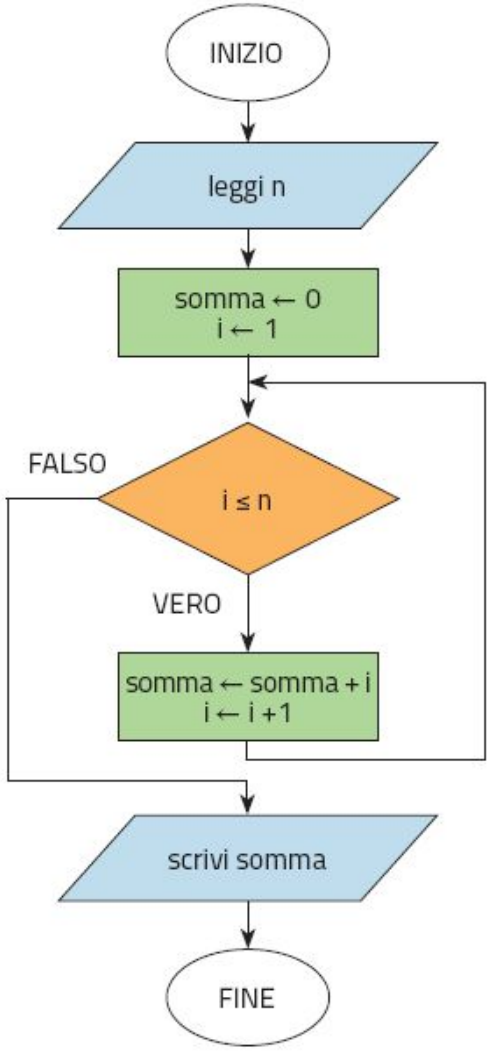
11.4 – Le strutture iterative

Il ciclo di tipo for permette di iterare per un numero prestabilito di volte.

```
for (condizione da verificare)
      istruzioni da eseguire se la condizione è vera
      poi torna all'istruzione for per l'iterazione successiva
end for
```

Esempio: algoritmo per calcolare, dato l'input **n**, la somma dei primi **n** numeri interi positivi, cioè:

1 + 2 + 3 + … + (n–1) + n .



1. Assegna come input il valore della variabile **n**.

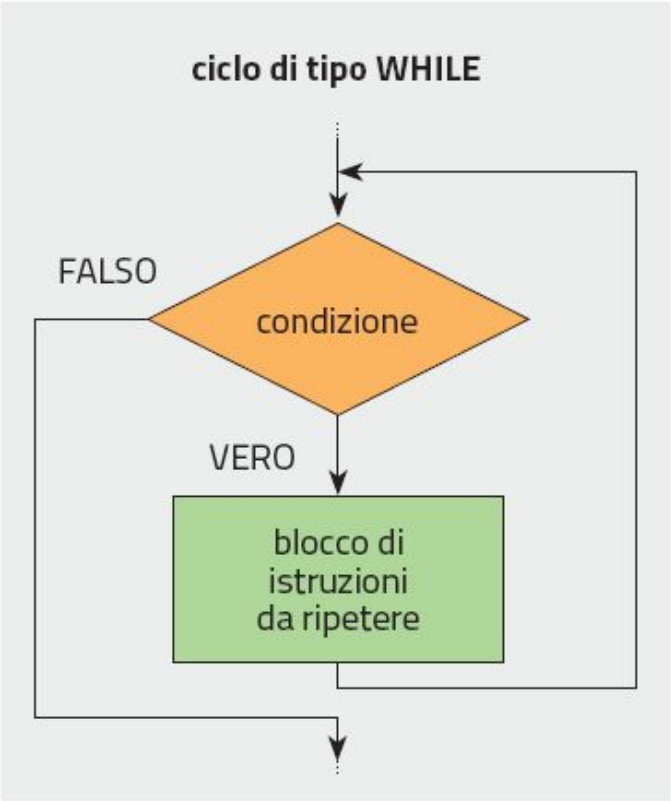
2. Inizializza la variabile **somma** con il valore 0 e il contatore **i** con il valore 1.

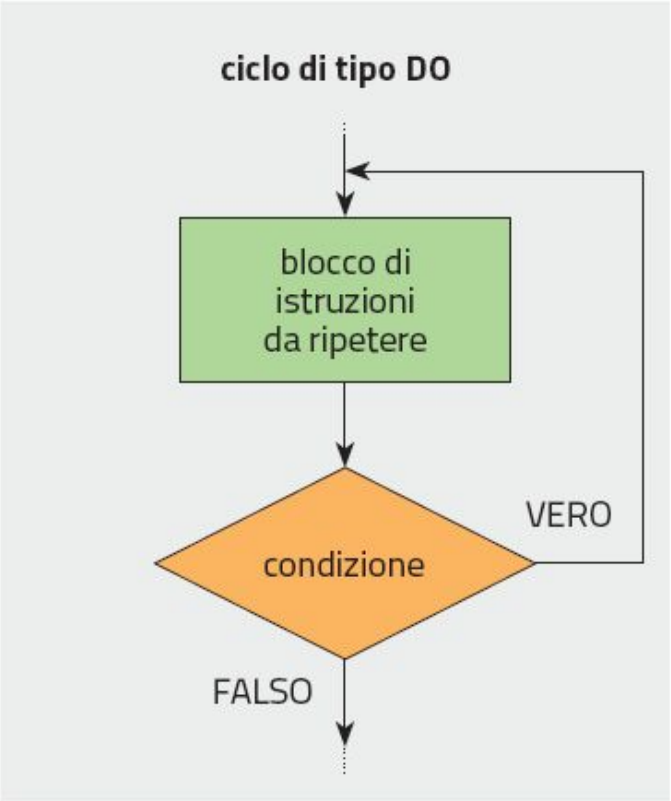
3. Se il valore di **i** è minore o uguale a **n**, aggiungi **i** a **somma**, poi incrementa di un'unità il valore di **i** e quindi itera, cioè vai a ripetere il test di controllo.

4. Altrimenti produci come output il valore di **somma**.

11.4 – Le strutture iterative

I cicli di tipo while e do si usano se il numero di iterazioni non è predefinito.



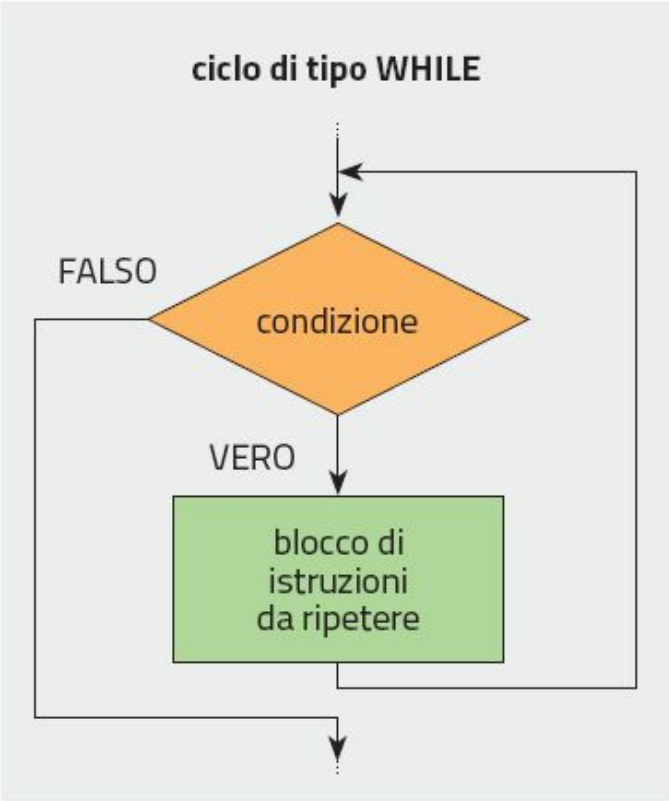


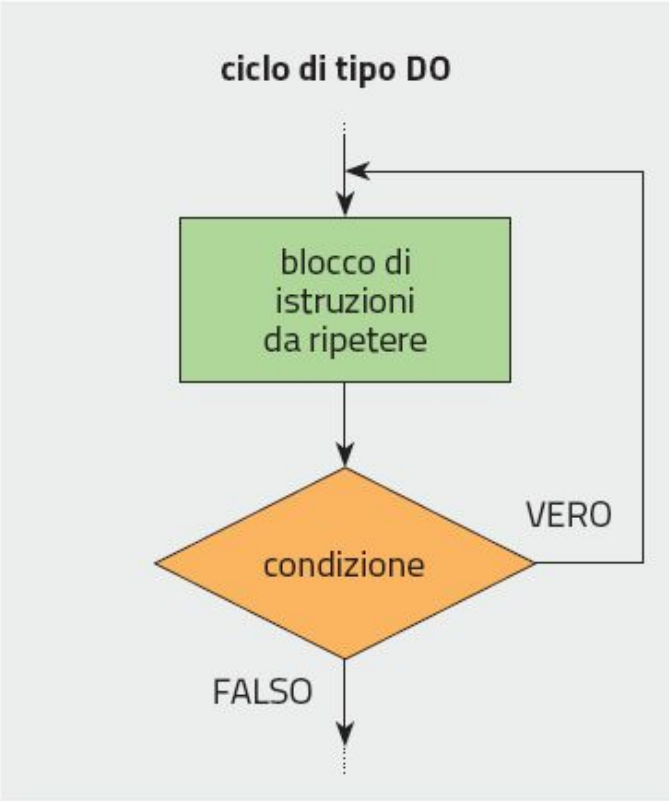
while *(condizione da verificare)*
istruzioni da eseguire
finché la condizione è vera
end while

do
istruzioni da eseguire
while *(condizione da verificare)*
se la condizione è vera, torna a **do**

11.4 – Le strutture iterative

I cicli di tipo while e do si usano se il numero di iterazioni non è predefinito.





la condizione *precede* le istruzioni da ripetere, come nel ciclo **for**

la condizione è *dopo* le istruzioni: saranno eseguite almeno una volta

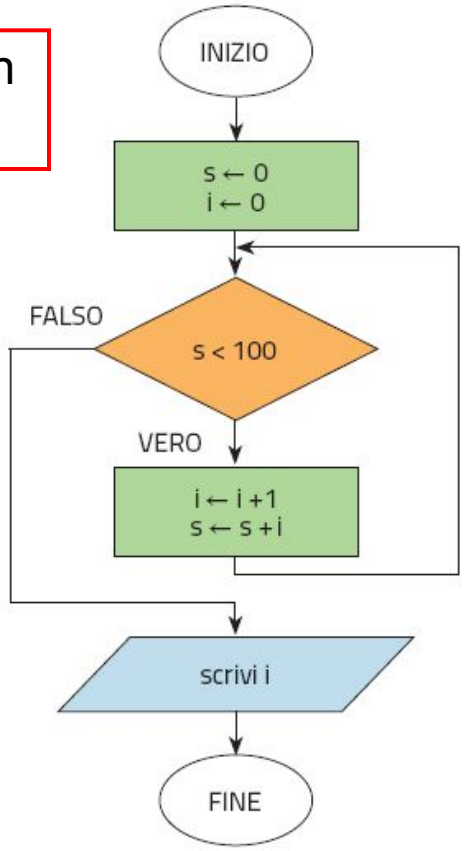
a parte questo aspetto, i due cicli sono equivalenti tra loro

11.4 – Le strutture iterative

I cicli di tipo while e do si usano se il numero di iterazioni non è predefinito.

Esempio: se sommiamo gli interi positivi (così: 1 + 2 + 3 + ...), a quale numero dovremo arrivare perché la somma ottenuta superi il valore 100?

algoritmo con **ciclo while**



1. Inizializza con il valore 0 la variabile numero intero **i** e la variabile somma **s**.

2. Controlla se il valore della somma **s** è minore del valore-soglia 100.

3. Se **s** < 100, aumenta il valore di **i** di un'unità (cioè passa al numero intero successivo), aggiungilo alla somma **s**, poi vai a eseguire di nuovo il controllo.

4. Altrimenti, cioè se **s** ≥ 100, esci dal ciclo e produci come output il valore di **i**.