

# Diagrammi di Flusso

Guida Completa con Esercizi Svolti

Materiale Didattico

22 ottobre 2025

## Indice

<b>1</b>	<b>Introduzione ai Diagrammi di Flusso</b>	<b>3</b>
1.1	Cosa sono i Diagrammi di Flusso . . . . .	3
1.1.1	Perché sono importanti? . . . . .	3
1.2	Simboli Standard . . . . .	3
1.3	Regole di Base . . . . .	3
<b>2</b>	<b>Le Variabili</b>	<b>4</b>
2.1	Cosa sono le Variabili . . . . .	4
2.2	Operazioni con le Variabili . . . . .	4
2.3	Esempio Semplice: Assegnazione . . . . .	4
2.4	Esercizi Svolti con Variabili . . . . .	5
2.4.1	Esercizio 1: Assegnazione e Output . . . . .	5
2.4.2	Esercizio 2: Input e Output . . . . .	6
2.4.3	Esercizio 3: Operazioni Aritmetiche Semplici . . . . .	7
2.4.4	Esercizio 4: Somma di Due Numeri . . . . .	8
2.4.5	Esercizio 5: Calcolo della Media . . . . .	10
2.4.6	Esercizio 6: Area di un Rettangolo . . . . .	12
2.4.7	Esercizio 7: Conversione Temperature (Celsius - Fahrenheit) . . . . .	14
2.4.8	Esercizio 8: Scambio di Valori . . . . .	15
<b>3</b>	<b>Blocchi Condizionali (Selezione)</b>	<b>17</b>
3.1	Cosa sono i Blocchi Condizionali . . . . .	17
3.1.1	Concetto di Condizione . . . . .	17
3.2	Tipi di Strutture Condizionali . . . . .	17
3.2.1	Selezione Semplice (if) . . . . .	17
3.2.2	Selezione Binaria (if-else) . . . . .	18
3.2.3	Selezione Multipla (if-else if-else) . . . . .	18
3.3	Operatori di Confronto . . . . .	18
3.4	Operatori Logici . . . . .	19
3.5	Esercizi Svolti con Blocchi Condizionali . . . . .	20
3.5.1	Esercizio 1: Numero Positivo o Negativo . . . . .	20
3.5.2	Esercizio 2: Maggiore o Minore di Età . . . . .	22
3.5.3	Esercizio 3: Massimo tra Due Numeri . . . . .	23
3.5.4	Esercizio 4: Numero Pari o Dispari . . . . .	25
3.5.5	Esercizio 5: Voto Scolastico con Giudizio . . . . .	26
3.5.6	Esercizio 6: Calcolatrice Semplice . . . . .	28
3.5.7	Esercizio 7: Anno Bisestile . . . . .	30
3.5.8	Esercizio 8: Triangolo Validato e Classificazione . . . . .	32
3.6	Esercizi Proposti sui Blocchi Condizionali . . . . .	34
<b>4</b>	<b>Esercizi Proposti (Variabili e Condizioni)</b>	<b>35</b>

<b>5 Cicli (Iterazione)</b>	<b>36</b>
5.1 Cosa sono i Cicli . . . . .	36
5.1.1 Perché usare i cicli? . . . . .	36
5.2 Tipi di Cicli . . . . .	36
5.2.1 Ciclo WHILE (pre-condizionale) . . . . .	36
5.2.2 Ciclo DO-WHILE (post-condizionale) . . . . .	37
5.2.3 Ciclo FOR (a contatore) . . . . .	37
5.3 Componenti Fondamentali di un Ciclo . . . . .	39
5.4 Confronto tra i Cicli . . . . .	39
5.5 Esercizi Svolti con Cicli . . . . .	40
5.5.1 Esercizio 1: Contare da 1 a 10 (Ciclo FOR) . . . . .	40
5.5.2 Esercizio 2: Somma dei primi N numeri (Ciclo WHILE) . . . . .	41
5.5.3 Esercizio 3: Fattoriale di un Numero (Ciclo FOR) . . . . .	43
5.5.4 Esercizio 4: Validazione Input (Ciclo DO-WHILE) . . . . .	45
5.5.5 Esercizio 5: Tavola Pitagorica (Cicli Annidati) . . . . .	47
5.5.6 Esercizio 6: Contare i Numeri Pari (Ciclo WHILE) . . . . .	49
5.5.7 Esercizio 7: Media di N Numeri (Ciclo WHILE) . . . . .	51
5.5.8 Esercizio 8: Sequenza di Fibonacci (Ciclo FOR) . . . . .	53
5.6 Esercizi Proposti sui Cicli . . . . .	55
<b>6 Appendice: Convenzioni e Suggerimenti</b>	<b>56</b>
6.1 Convenzioni di Scrittura . . . . .	56
6.2 Suggerimenti per Disegnare Diagrammi . . . . .	56
6.3 Errori Comuni da Evitare . . . . .	56

# 1 Introduzione ai Diagrammi di Flusso

## 1.1 Cosa sono i Diagrammi di Flusso

I **diagrammi di flusso** (o *flowchart*) sono rappresentazioni grafiche di algoritmi o processi. Utilizzano simboli geometrici standardizzati collegati da frecce per mostrare la sequenza di operazioni da eseguire per risolvere un problema o completare un'attività.

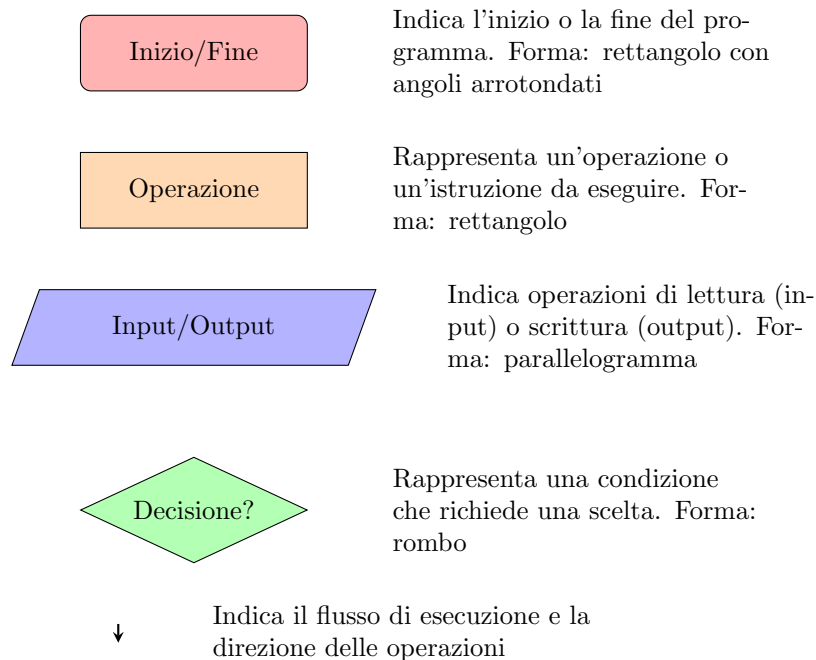
### 1.1.1 Perché sono importanti?

I diagrammi di flusso sono fondamentali nella programmazione per diversi motivi:

- **Visualizzazione:** Permettono di vedere graficamente la logica di un algoritmo
- **Progettazione:** Aiutano a pianificare il codice prima di scriverlo
- **Comunicazione:** Facilitano la condivisione delle idee con altri programmatori
- **Debug:** Rendono più semplice individuare errori logici
- **Documentazione:** Forniscono documentazione visiva del codice

## 1.2 Simboli Standard

Ogni forma geometrica in un diagramma di flusso ha un significato specifico:



## 1.3 Regole di Base

Quando si disegnano diagrammi di flusso, è importante seguire alcune regole:

1. **Un solo inizio e una sola fine:** Ogni diagramma deve avere esattamente un punto di partenza e un punto di arrivo
2. **Flusso dall'alto verso il basso:** Generalmente il flusso procede dall'alto verso il basso e da sinistra a destra
3. **Frecce chiare:** Le frecce devono indicare chiaramente la direzione del flusso
4. **Nessun incrocio di linee:** Evitare che le linee si incrocino (quando possibile)
5. **Chiarezza:** Ogni blocco deve contenere istruzioni semplici e comprensibili
6. **Decisioni binarie:** I rombi devono avere esattamente due uscite (Vero/Falso, Sì/No)

## 2 Le Variabili

### 2.1 Cosa sono le Variabili

Una **variabile** è un contenitore che memorizza un valore nella memoria del computer. Ogni variabile ha:

- **Nome:** Un identificatore univoco (es. `x`, `eta`, `somma`)
- **Tipo:** Il tipo di dato che può contenere (es. numero intero, decimale, testo)
- **Valore:** Il contenuto attuale della variabile

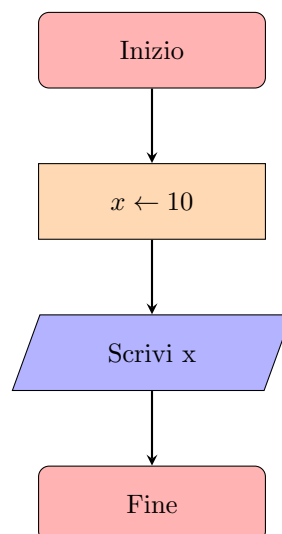
### 2.2 Operazioni con le Variabili

Le operazioni principali che si possono fare con le variabili sono:

1. **Assegnazione:** Memorizzare un valore in una variabile  
Sintassi: `variabile ← valore` o `variabile = valore`
2. **Lettura (Input):** Acquisire un valore dall'utente  
Sintassi: `Leggi variabile` o `Input variabile`
3. **Scrittura (Output):** Mostrare il valore di una variabile  
Sintassi: `Scrivi variabile` o `Output variabile`
4. **Operazioni aritmetiche:** Calcoli matematici  
Esempi: `somma ← a + b`, `prodotto ← x * y`

### 2.3 Esempio Semplice: Assegnazione

Vediamo come rappresentare l'assegnazione di un valore a una variabile:



Questo diagramma esegue le seguenti operazioni:

1. Inizia il programma
2. Assegna il valore 10 alla variabile  $x$
3. Mostra il valore di  $x$  (stampa 10)
4. Termina il programma

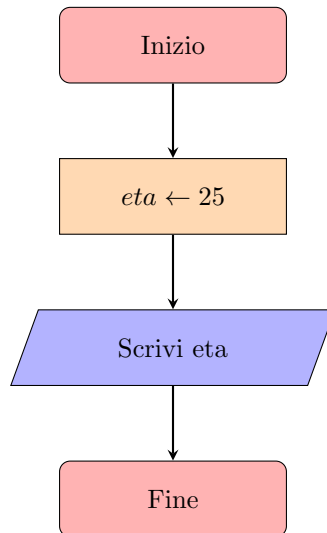
## 2.4 Esercizi Svolti con Variabili

### 2.4.1 Esercizio 1: Assegnazione e Output

#### Traccia

Creare un diagramma di flusso che assegni il valore 25 alla variabile **eta** e poi lo visualizzi.

#### Soluzione:



#### Spiegazione:

1. Il programma inizia
2. Viene assegnato il valore 25 alla variabile **eta**
3. Il valore di **eta** viene stampato (risultato: 25)
4. Il programma termina

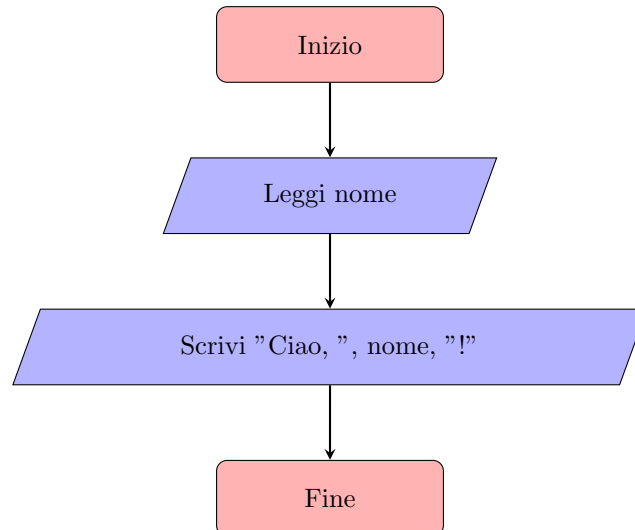
#### Codice equivalente (pseudocodice):

```
1 INIZIO
2   eta ← 25
3   SCRIVI eta
4 FINE
```

### 2.4.2 Esercizio 2: Input e Output

**Traccia**

Creare un diagramma di flusso che chieda all'utente di inserire il proprio nome e poi lo saluti stampando "Ciao, [nome]!".

**Soluzione:****Spiegazione:**

1. Il programma inizia
2. L'utente inserisce il proprio nome che viene memorizzato nella variabile `nome`
3. Viene stampato il messaggio di saluto personalizzato
4. Il programma termina

**Esempio di esecuzione:**

- Input: Mario
- Output: Ciao, Mario!

**Codice equivalente (pseudocodice):**

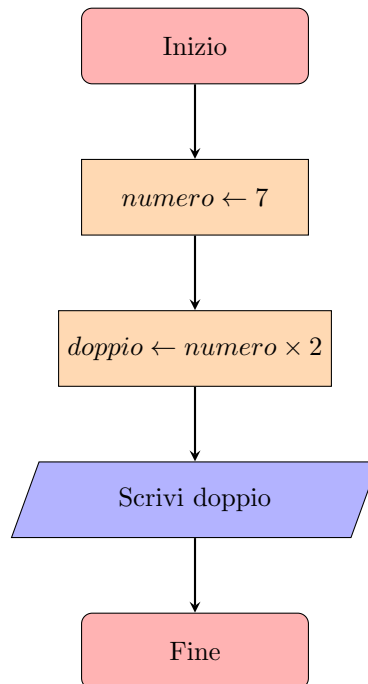
```
1 INIZIO
2     LEGGI nome
3     SCRIVI "Ciao, ", nome, "!"
4 FINE
```

### 2.4.3 Esercizio 3: Operazioni Aritmetiche Semplici

#### Traccia

Creare un diagramma di flusso che calcoli il doppio di un numero. Il programma deve assegnare il valore 7 alla variabile **numero**, calcolare il doppio e visualizzare il risultato.

#### Soluzione:



#### Spiegazione:

1. Il programma inizia
2. Viene assegnato il valore 7 alla variabile **numero**
3. Viene calcolato il doppio:  $doppio = 7 \times 2 = 14$
4. Il valore di **doppio** viene stampato (risultato: 14)
5. Il programma termina

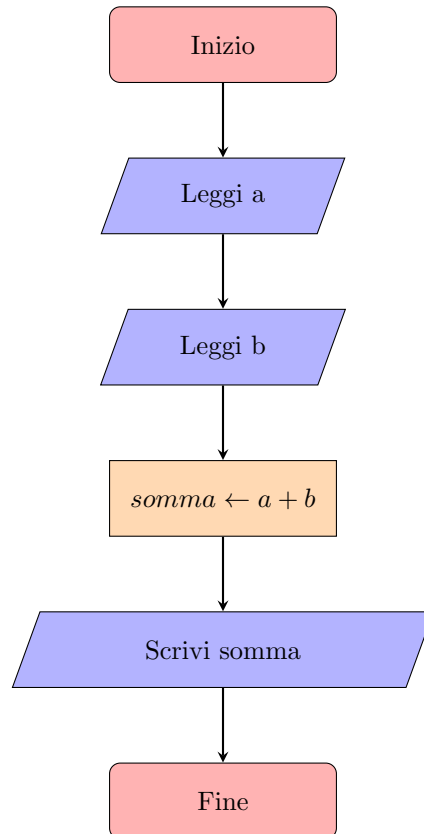
#### Codice equivalente (pseudocodice):

```
1 INIZIO
2   numero <- 7
3   doppio <- numero * 2
4   SCRIVI doppio
5 FINE
```

#### 2.4.4 Esercizio 4: Somma di Due Numeri

**Traccia**

Creare un diagramma di flusso che chieda all'utente due numeri, calcoli la loro somma e visualizzi il risultato.

**Soluzione:****Spiegazione:**

1. Il programma inizia
2. L'utente inserisce il primo numero che viene memorizzato in **a**
3. L'utente inserisce il secondo numero che viene memorizzato in **b**
4. Viene calcolata la somma:  $somma = a + b$
5. Il risultato viene stampato
6. Il programma termina

**Esempio di esecuzione:**

- Input 1: **a** = 15
- Input 2: **b** = 23
- Calcolo:  $somma = 15 + 23 = 38$
- Output: 38

**Codice equivalente (pseudocodice):**

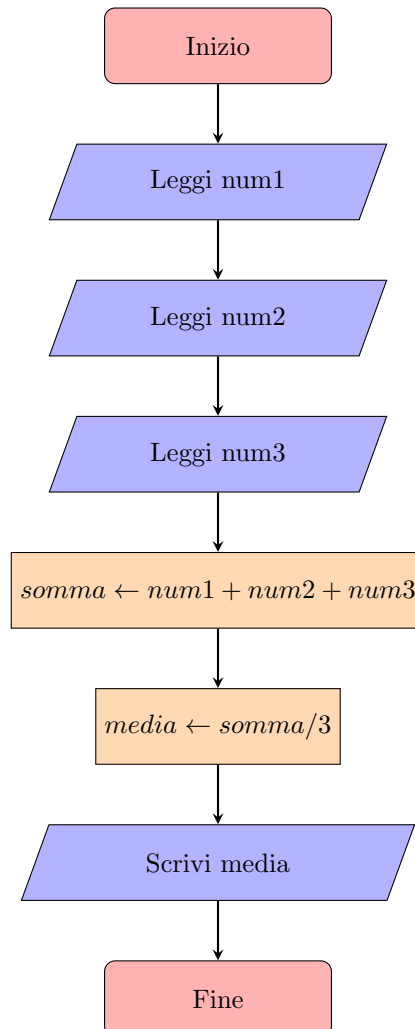


```
1 INIZIO
2   LEGGI a
3   LEGGI b
4   somma <- a + b
5   SCRIVI somma
6 FINE
```

### 2.4.5 Esercizio 5: Calcolo della Media

**Traccia**

Creare un diagramma di flusso che calcoli la media di tre numeri inseriti dall'utente e visualizzi il risultato.

**Soluzione:****Spiegazione:**

1. Il programma inizia
2. L'utente inserisce tre numeri: `num1`, `num2`, `num3`
3. Viene calcolata la somma dei tre numeri
4. Viene calcolata la media dividendo la somma per 3
5. Il risultato viene stampato
6. Il programma termina

**Esempio di esecuzione:**

- Input 1: `num1` = 8
- Input 2: `num2` = 7
- Input 3: `num3` = 9

- Calcolo somma:  $somma = 8 + 7 + 9 = 24$
- Calcolo media:  $media = 24/3 = 8$
- Output: 8

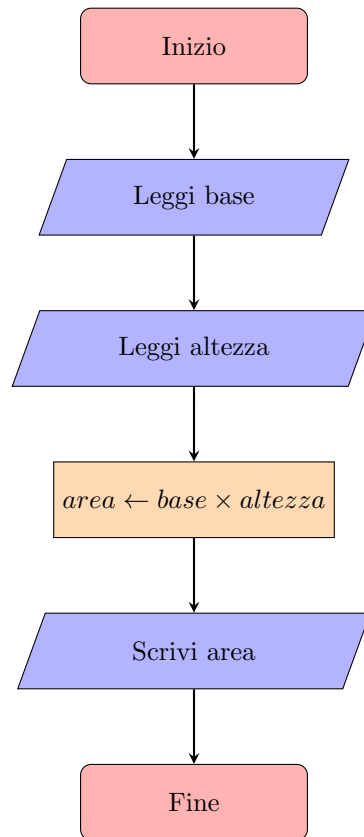
**Codice equivalente (pseudocodice):**

```
1 INIZIO
2     LEGGI num1
3     LEGGI num2
4     LEGGI num3
5     somma <- num1 + num2 + num3
6     media <- somma / 3
7     SCRIVI media
8 FINE
```

### 2.4.6 Esercizio 6: Area di un Rettangolo

**Traccia**

Creare un diagramma di flusso che calcoli l'area di un rettangolo. Il programma deve chiedere all'utente di inserire base e altezza, calcolare l'area e visualizzare il risultato.

**Soluzione:****Spiegazione:**

1. Il programma inizia
2. L'utente inserisce la base del rettangolo
3. L'utente inserisce l'altezza del rettangolo
4. Viene calcolata l'area:  $area = base \times altezza$
5. Il risultato viene stampato
6. Il programma termina

**Esempio di esecuzione:**

- Input 1: `base = 5`
- Input 2: `altezza = 8`
- Calcolo:  $area = 5 \times 8 = 40$
- Output: L'area del rettangolo è: 40

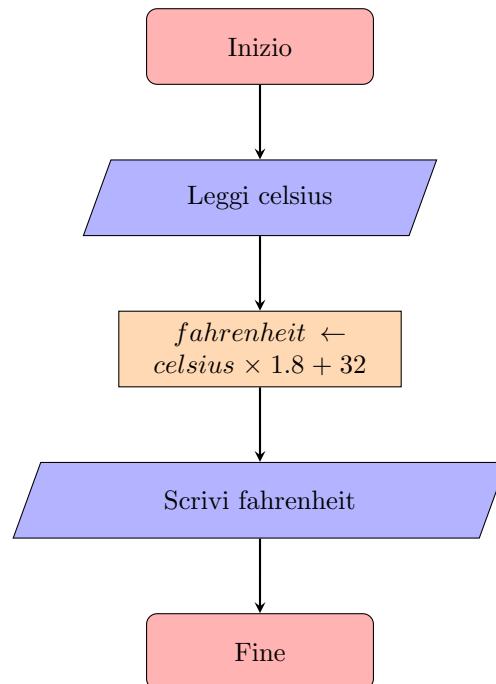
**Codice equivalente (pseudocodice):**

```
1 INIZIO
2   LEGGI base
3   LEGGI altezza
4   area <- base * altezza
5   SCRIVI "L'area del rettangolo e': ", area
6 FINE
```

### 2.4.7 Esercizio 7: Conversione Temperature (Celsius - Fahrenheit)

**Traccia**

Creare un diagramma di flusso che converta una temperatura da gradi Celsius a gradi Fahrenheit.  
Formula:  $F = C \times 1.8 + 32$

**Soluzione:****Spiegazione:**

1. Il programma inizia
2. L'utente inserisce la temperatura in gradi Celsius
3. Viene applicata la formula di conversione:  $fahrenheit = celsius \times 1.8 + 32$
4. Il risultato in gradi Fahrenheit viene stampato
5. Il programma termina

**Esempio di esecuzione:**

- Input: `celsius = 25`
- Calcolo:  $fahrenheit = 25 \times 1.8 + 32 = 45 + 32 = 77$
- Output: `77°F`

**Nota matematica:** La formula di conversione deriva dalla relazione lineare tra le due scale: il punto di congelamento dell'acqua è  $0^{\circ}\text{C}$  ( $32^{\circ}\text{F}$ ) e il punto di ebollizione è  $100^{\circ}\text{C}$  ( $212^{\circ}\text{F}$ ).

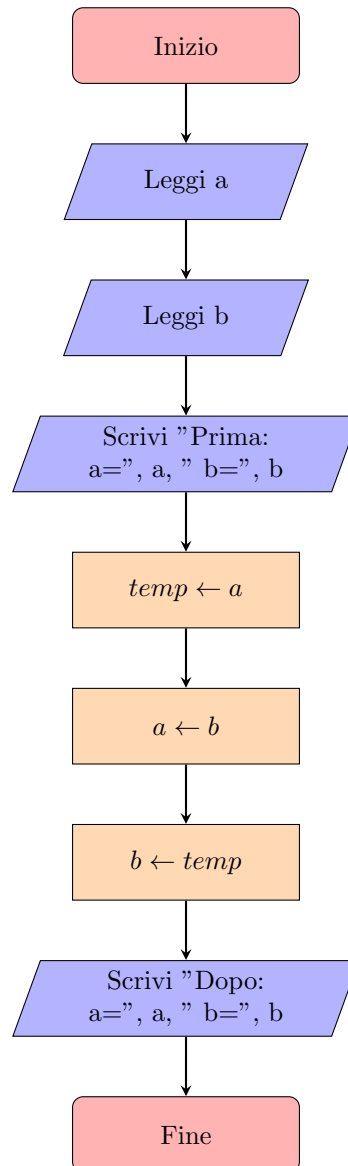
**Codice equivalente (pseudocodice):**

```
1 INIZIO
2   LEGGI celsius
3   fahrenheit <- celsius * 1.8 + 32
4   SCRIVI fahrenheit, " gradi Fahrenheit"
5 FINE
```

### 2.4.8 Esercizio 8: Scambio di Valori

**Traccia**

Creare un diagramma di flusso che scambi i valori di due variabili **a** e **b**. Utilizzare una variabile temporanea **temp**.

**Soluzione:****Spiegazione:**

1. Il programma inizia
2. L'utente inserisce i valori di **a** e **b**
3. Vengono stampati i valori iniziali
4. Il valore di **a** viene salvato nella variabile temporanea **temp**
5. Il valore di **b** viene copiato in **a**
6. Il valore di **temp** (che conteneva il vecchio valore di **a**) viene copiato in **b**
7. Vengono stampati i valori dopo lo scambio

8. Il programma termina

**Esempio di esecuzione:**

- Input 1:  $a = 10$
- Input 2:  $b = 20$
- Output iniziale: Prima:  $a=10$   $b=20$
- Scambio:
  - $temp = 10$  (salvo  $a$ )
  - $a = 20$  (copio  $b$  in  $a$ )
  - $b = 10$  (copio  $temp$  in  $b$ )
- Output finale: Dopo:  $a=20$   $b=10$

**Perché serve la variabile temporanea?**

Senza la variabile `temp`, quando copiamo  $b$  in  $a$ , perdiamo il valore originale di  $a$  e non possiamo più assegnarlo a  $b$ .

**Codice equivalente (pseudocodice):**

```
1 INIZIO
2   LEGGI a
3   LEGGI b
4   SCRIVI "Prima: a=", a, " b=", b
5   temp <- a
6   a <- b
7   b <- temp
8   SCRIVI "Dopo: a=", a, " b=", b
9 FINE
```



## 3 Blocchi Condizionali (Selezione)

### 3.1 Cosa sono i Blocchi Condizionali

I **blocchi condizionali** (o strutture di selezione) permettono al programma di prendere decisioni e seguire percorsi diversi in base al valore di una condizione. Sono fondamentali per creare programmi che reagiscono in modo diverso a situazioni diverse.

#### 3.1.1 Concetto di Condizione

Una **condizione** è un'espressione che può essere vera o falsa. Esempi:

- $x > 10$  (vero se  $x$  è maggiore di 10, altrimenti falso)
- $eta \geq 18$  (vero se  $eta$  è maggiore o uguale a 18)
- $nome = "Mario"$  (vero se nome è esattamente "Mario")
- $a \neq b$  (vero se  $a$  è diverso da  $b$ )

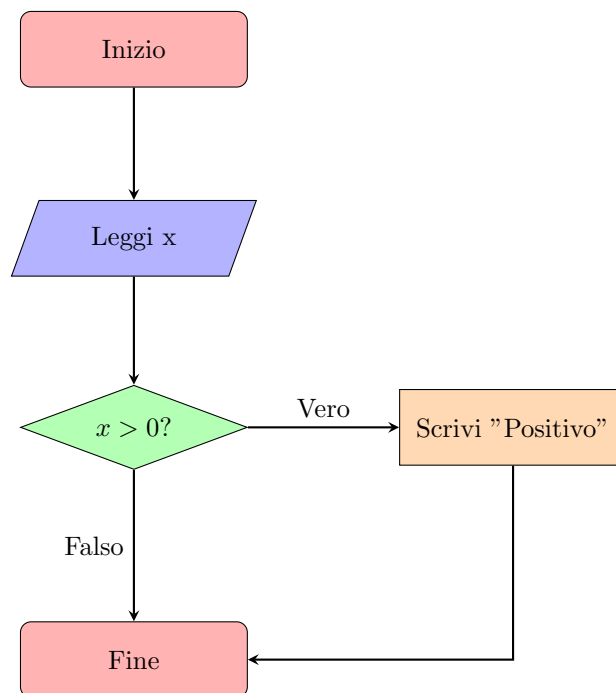
### 3.2 Tipi di Strutture Condizionali

#### 3.2.1 Selezione Semplice (if)

Esegue un blocco di istruzioni solo se la condizione è vera.

**Sintassi:**

```
1 SE condizione ALLORA
2     istruzioni
3 FINE SE
```



### 3.2.2 Selezione Binaria (if-else)

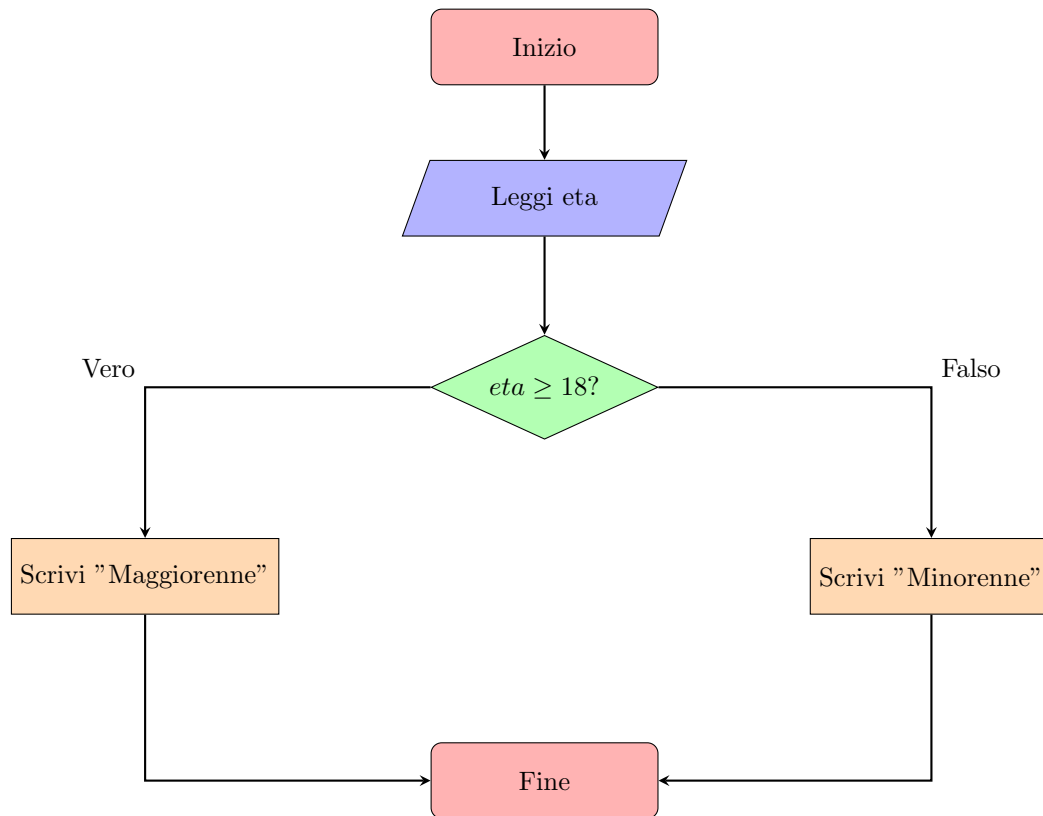
Esegue un blocco se la condizione è vera, altrimenti esegue un blocco alternativo.

**Sintassi:**

```

1 SE condizione ALLORA
2     istruzioni_vero
3 ALTRIMENTI
4     istruzioni_falso
5 FINE SE

```



### 3.2.3 Selezione Multipla (if-else if-else)

Permette di verificare più condizioni in sequenza.

**Sintassi:**

```

1 SE condizione1 ALLORA
2     istruzioni1
3 ALTRIMENTI SE condizione2 ALLORA
4     istruzioni2
5 ALTRIMENTI
6     istruzioni3
7 FINE SE

```

## 3.3 Operatori di Confronto

Operatore	Significato	Esempio
<code>= o ==</code>	Uguale a	$x = 5$
<code>≠ o !=</code>	Diverso da	$x \neq 0$
<code>&lt;</code>	Minore di	$x < 10$
<code>&gt;</code>	Maggiore di	$x > 0$
<code>≤</code>	Minore o uguale	$x \leq 100$
<code>≥</code>	Maggiore o uguale	$x \geq 18$

### 3.4 Operatori Logici

Per combinare più condizioni:

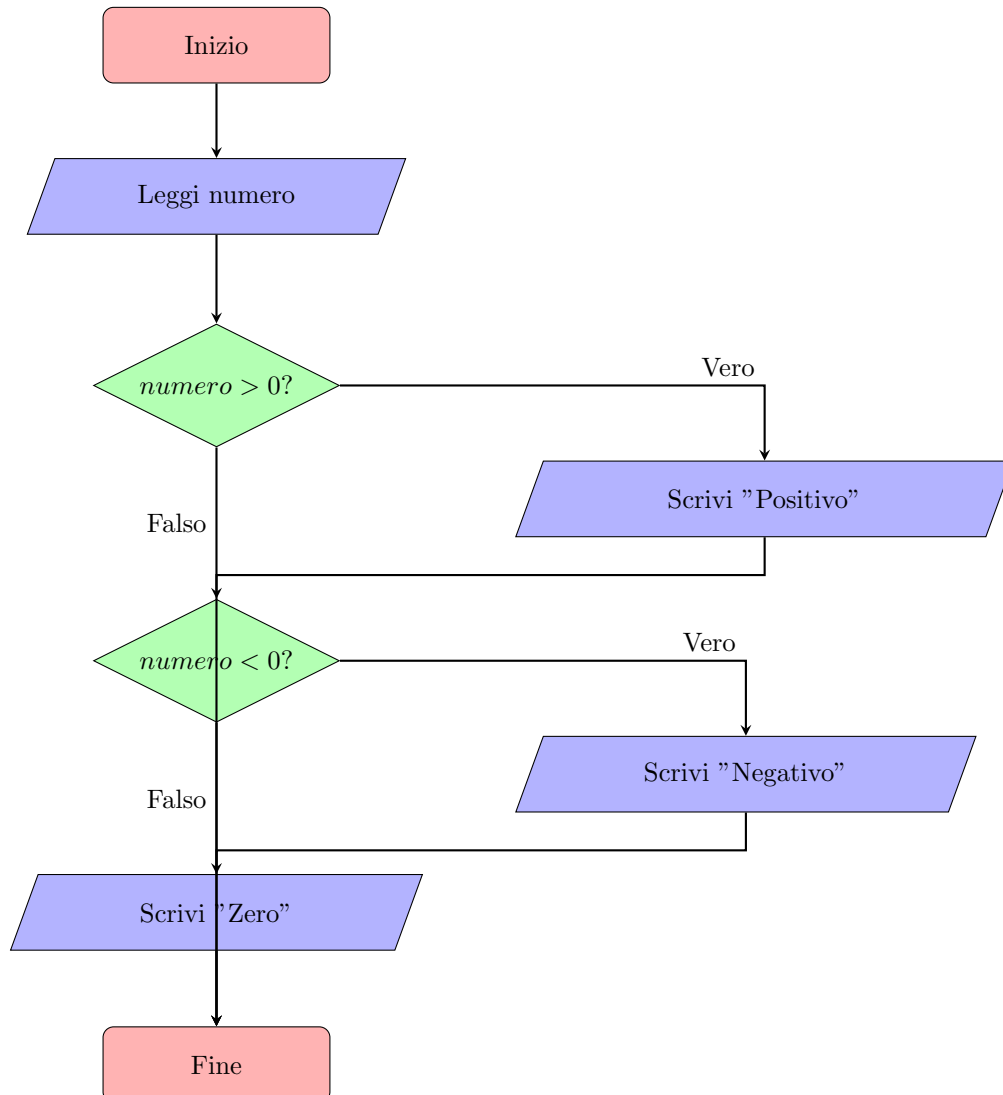
Operatore	Significato	Esempio
AND (E)	Entrambe vere	$(x > 0)$ AND $(x < 10)$
OR (O)	Almeno una vera	$(x < 0)$ OR $(x > 100)$
NOT (NON)	Negazione	NOT $(x = 0)$

### 3.5 Esercizi Svolti con Blocchi Condizionali

#### 3.5.1 Esercizio 1: Numero Positivo o Negativo

**Traccia**

Creare un diagramma di flusso che legga un numero e stampi se è positivo, negativo o zero.

**Soluzione:****Spiegazione:**

1. Il programma legge un numero
2. Prima condizione: verifica se il numero è maggiore di 0
  - Se VERO: stampa "Positivo" e termina
  - Se FALSO: passa alla seconda condizione
3. Seconda condizione: verifica se il numero è minore di 0
  - Se VERO: stampa "Negativo" e termina
  - Se FALSO: il numero deve essere zero, stampa "Zero"

**Esempi di esecuzione:**

- Input: 5 → Output: Positivo
- Input: -3 → Output: Negativo
- Input: 0 → Output: Zero

**Codice equivalente (pseudocodice):**

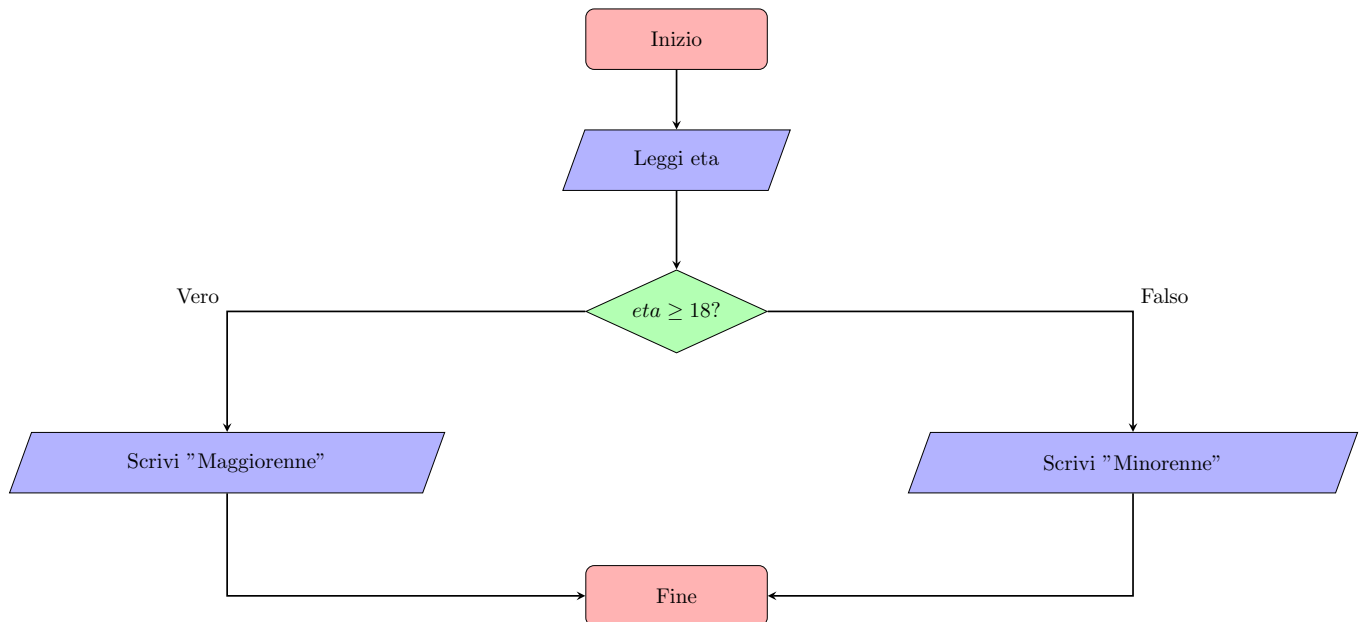
```
1 INIZIO
2   LEGGI numero
3   SE numero > 0 ALLORA
4     SCRIVI "Positivo"
5   ALTRIMENTI SE numero < 0 ALLORA
6     SCRIVI "Negativo"
7   ALTRIMENTI
8     SCRIVI "Zero"
9   FINE SE
10 FINE
```

### 3.5.2 Esercizio 2: Maggiore o Minore di Età

#### Traccia

Creare un diagramma di flusso che chieda l'età di una persona e stampi se è maggiorenne (età  $\geq$  18) o minorenne.

#### Soluzione:



#### Spiegazione:

1. Il programma chiede di inserire l'età
2. Verifica se l'età è maggiore o uguale a 18
3. Se VERO: stampa "Maggiorenne"
4. Se FALSO: stampa "Minorenne"
5. Il programma termina

#### Esempi di esecuzione:

- Input: età = 20 → Output: Maggiorenne
- Input: età = 16 → Output: Minorenne
- Input: età = 18 → Output: Maggiorenne

#### Codice equivalente (pseudocodice):

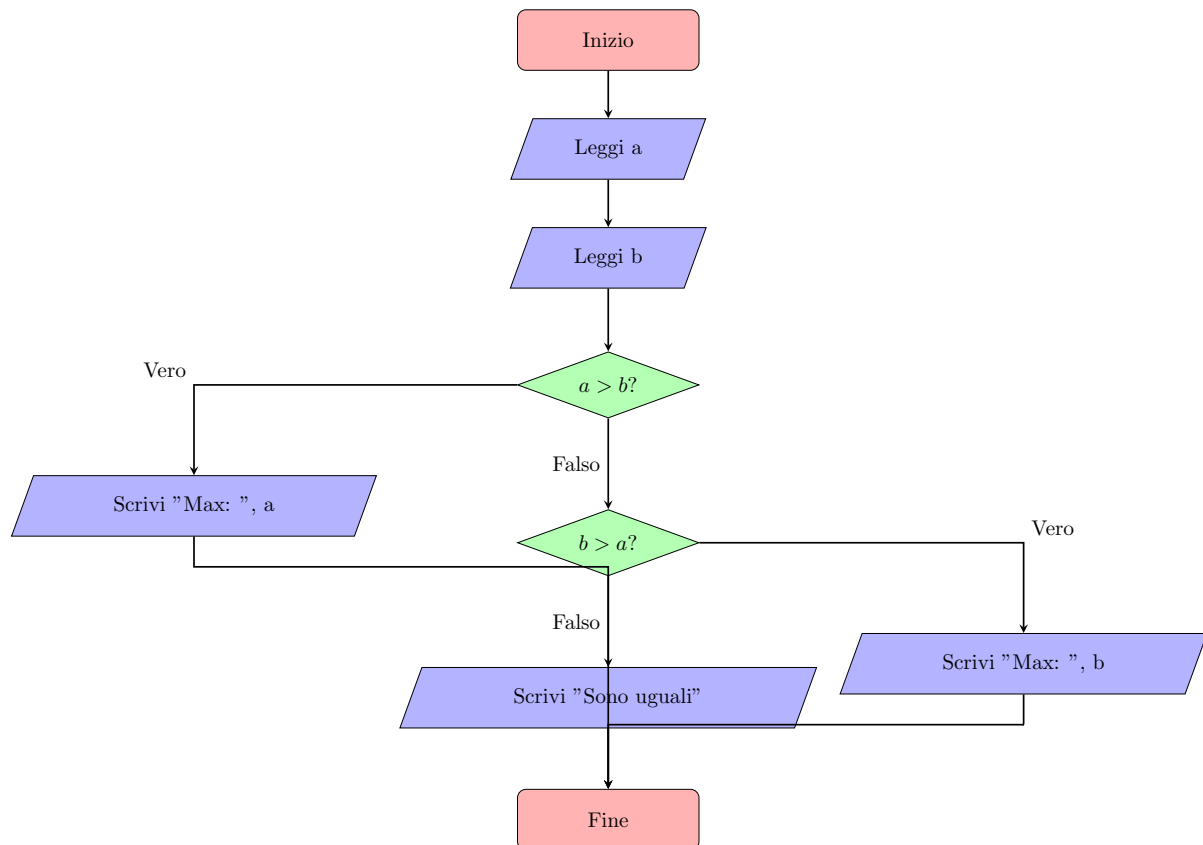
```

1 INIZIO
2   LEGGI età
3   SE età >= 18 ALLORA
4     SCRIVI "Maggiorenne"
5   ALTRIMENTI
6     SCRIVI "Minorenne"
7   FINE SE
8 FINE
  
```

### 3.5.3 Esercizio 3: Massimo tra Due Numeri

**Traccia**

Creare un diagramma di flusso che legga due numeri e stampi il maggiore. Se sono uguali, stampare un messaggio appropriato.

**Soluzione:****Spiegazione:**

1. Il programma legge due numeri **a** e **b**
2. Prima verifica se  $a > b$ :
  - Se VERO: **a** è il maggiore, lo stampa e termina
  - Se FALSO: procede con la seconda verifica
3. Seconda verifica se  $b > a$ :
  - Se VERO: **b** è il maggiore, lo stampa e termina
  - Se FALSO: i numeri sono uguali, stampa messaggio

**Esempi di esecuzione:**

- Input:  $a=10, b=5 \rightarrow$  Output: Max: 10
- Input:  $a=3, b=8 \rightarrow$  Output: Max: 8
- Input:  $a=7, b=7 \rightarrow$  Output: Sono uguali

**Codice equivalente (pseudocodice):**

```
1 INIZIO
2   LEGGI a
3   LEGGI b
4   SE a > b ALLORA
5       SCRIVI "Max: ", a
6   ALTRIMENTI SE b > a ALLORA
7       SCRIVI "Max: ", b
8   ALTRIMENTI
9       SCRIVI "Sono uguali"
10  FINE SE
11 FINE
```

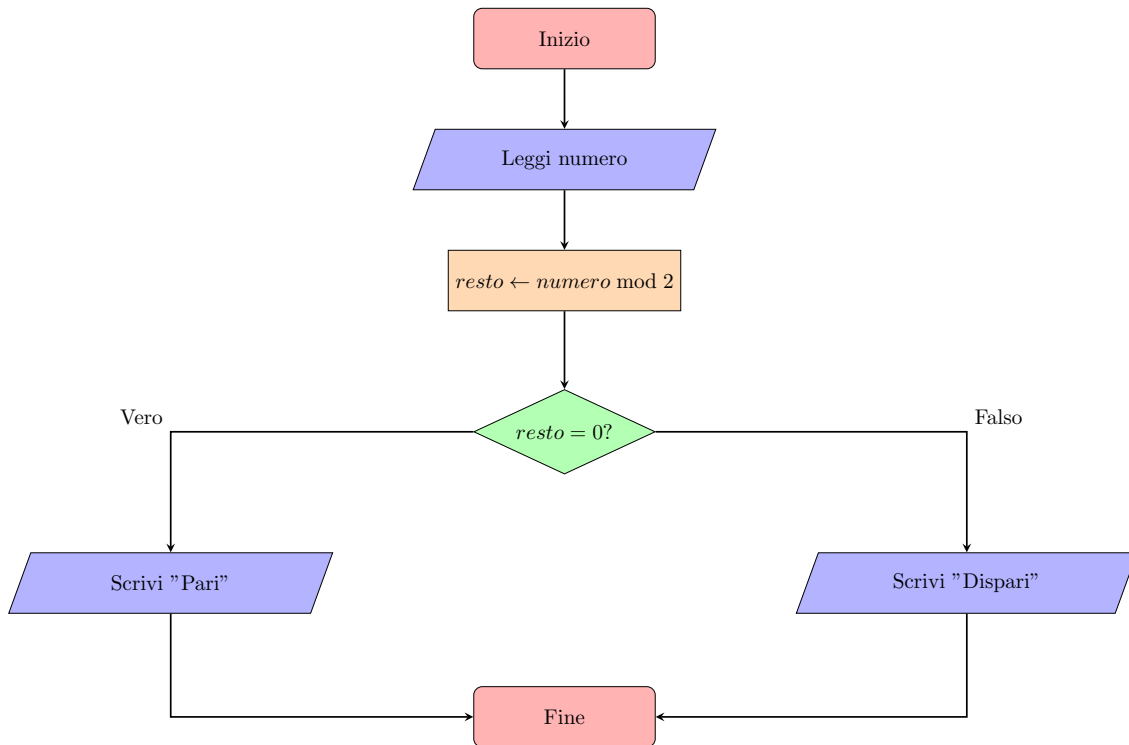


### 3.5.4 Esercizio 4: Numero Pari o Dispari

#### Traccia

Creare un diagramma di flusso che determini se un numero è pari o dispari. Un numero è pari se il resto della divisione per 2 è zero.

#### Soluzione:



#### Spiegazione:

1. Il programma legge un numero
2. Calcola il resto della divisione per 2 usando l'operatore modulo (mod)
3. Verifica se il resto è uguale a 0:
  - Se VERO: il numero è pari, stampa "Pari"
  - Se FALSO: il numero è dispari, stampa "Dispari"

**Concetto matematico:** Un numero è pari se è divisibile per 2 senza resto. L'operatore modulo (mod) restituisce il resto della divisione.

#### Esempi di esecuzione:

- Input: 8 →  $8 \bmod 2 = 0$  → Output: Pari
- Input: 15 →  $15 \bmod 2 = 1$  → Output: Dispari
- Input: 0 →  $0 \bmod 2 = 0$  → Output: Pari

#### Codice equivalente (pseudocodice):

```

1 INIZIO
2   LEGGI numero
3   resto ← numero MOD 2
4   SE resto = 0 ALLORA
5     SCRIVI "Pari"
6   ALTRIMENTI
7     SCRIVI "Dispari"
8   FINE SE
9 FINE
  
```

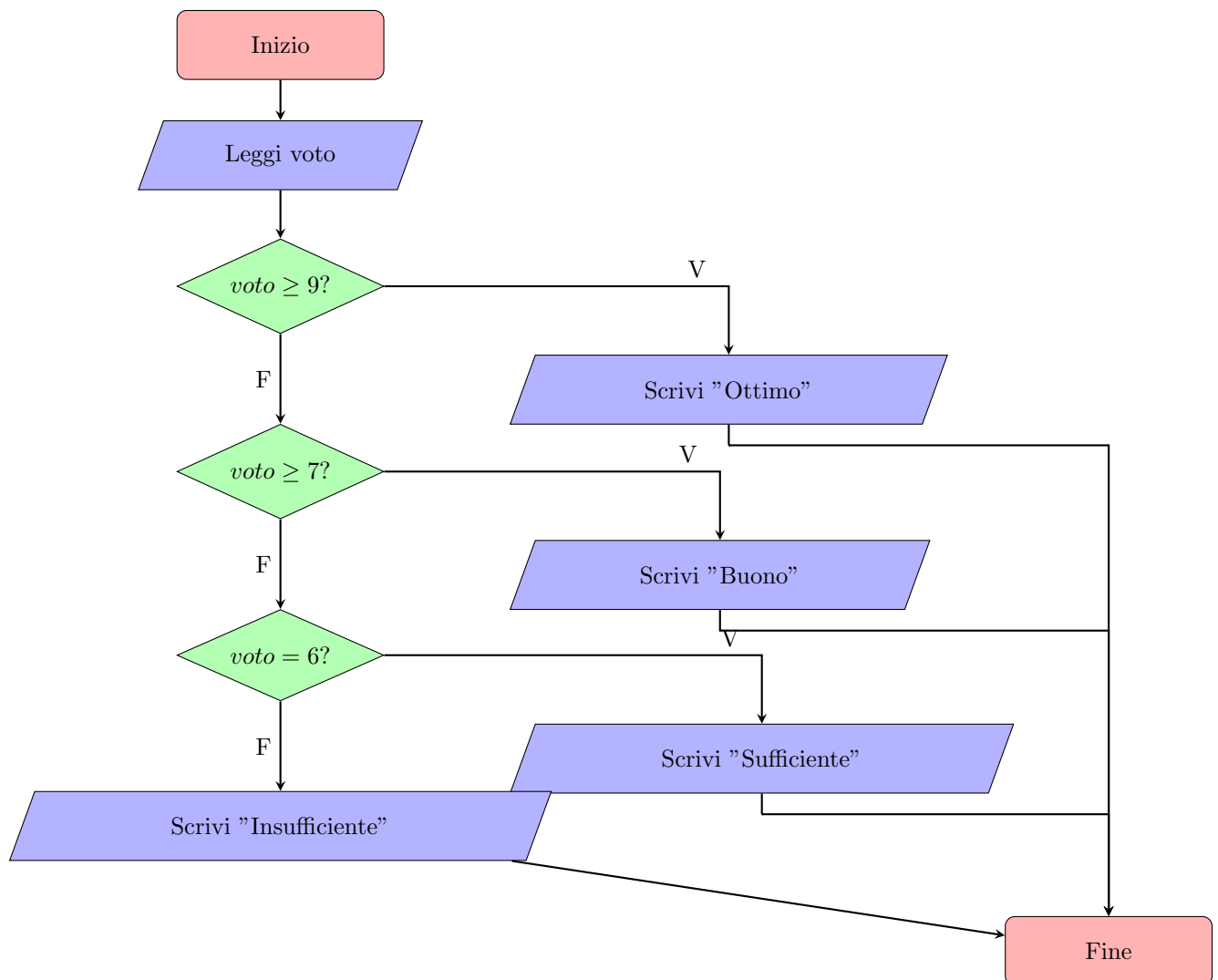
### 3.5.5 Esercizio 5: Voto Scolastico con Giudizio

#### Traccia

Creare un diagramma di flusso che legga un voto (0-10) e stampi il giudizio corrispondente:

- 9-10: Ottimo
- 7-8: Buono
- 6: Sufficiente
- 0-5: Insufficiente

#### Soluzione:



#### Spiegazione:

1. Il programma legge il voto
2. Verifica in sequenza le condizioni dall'alto verso il basso:
  - Se  $voto \geq 9$ : stampa "Ottimo"
  - Altrimenti, se  $voto \geq 7$ : stampa "Buono"
  - Altrimenti, se  $voto = 6$ : stampa "Sufficiente"
  - Altrimenti: stampa "Insufficiente"

**Esempi di esecuzione:**

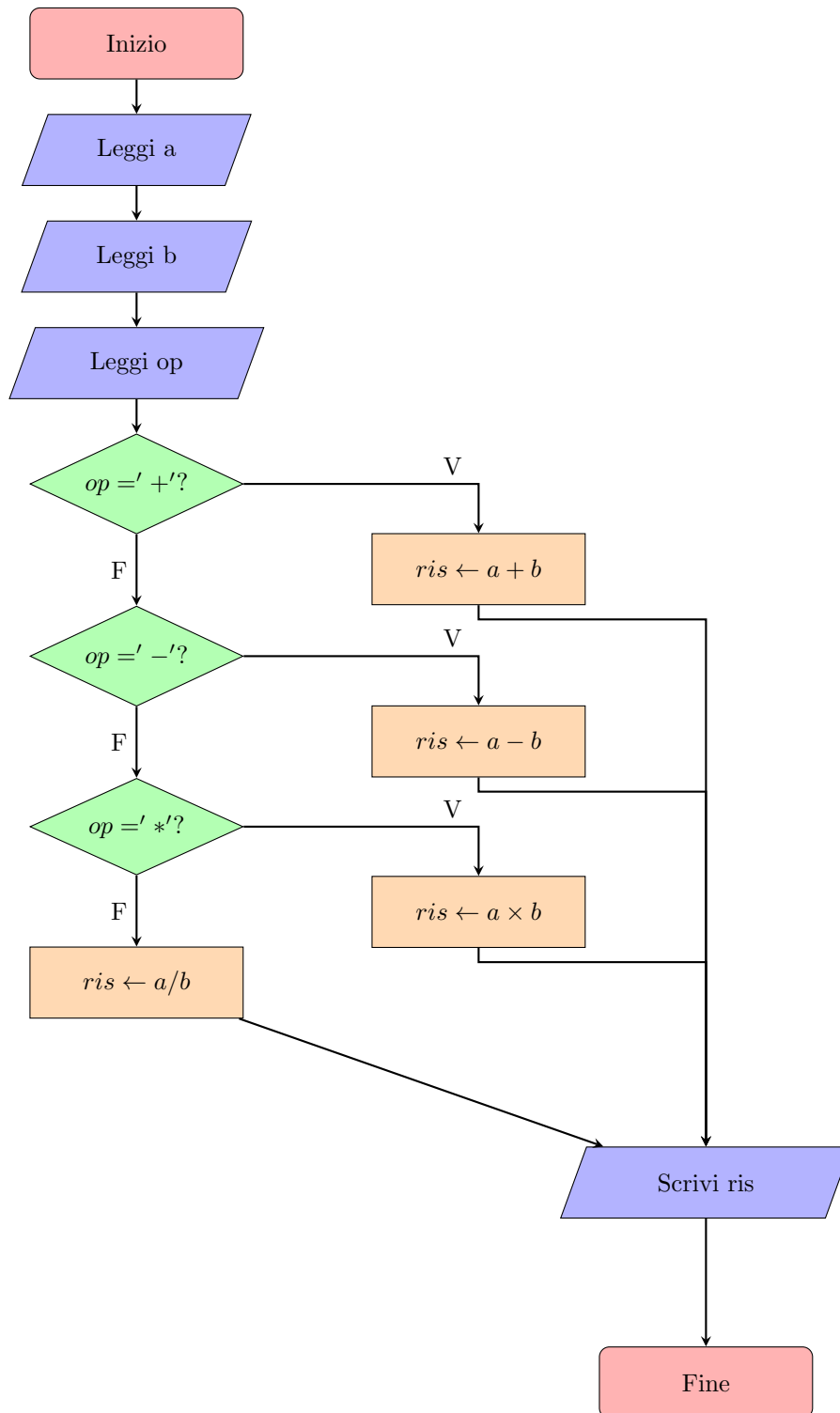
- Input: 10 → Output: Ottimo
- Input: 8 → Output: Buono
- Input: 6 → Output: Sufficiente
- Input: 4 → Output: Insufficiente

**Codice equivalente (pseudocodice):**

```
1 INIZIO
2   LEGGI voto
3   SE voto >= 9 ALLORA
4     SCRIVI "Ottimo"
5   ALTRIMENTI SE voto >= 7 ALLORA
6     SCRIVI "Buono"
7   ALTRIMENTI SE voto = 6 ALLORA
8     SCRIVI "Sufficiente"
9   ALTRIMENTI
10    SCRIVI "Insufficiente"
11  FINE SE
12 FINE
```

**3.5.6 Esercizio 6: Calcolatrice Semplice****Traccia**

Creare un diagramma di flusso per una calcolatrice che legga due numeri e un'operazione (+, -, \*, /) ed esegua il calcolo corrispondente.

**Soluzione:****Spiegazione:**

1. Il programma legge due numeri **a** e **b**

2. Legge l'operazione desiderata `op`
3. Verifica quale operazione è stata richiesta:
  - Se '+': calcola  $ris = a + b$
  - Se '-': calcola  $ris = a - b$
  - Se '\*': calcola  $ris = a \times b$
  - Altrimenti (assume '/'): calcola  $ris = a/b$
4. Stampa il risultato

**Esempio di esecuzione:**

- Input: `a=10, b=5, op='*'`
- Calcolo:  $ris = 10 \times 5 = 50$
- Output: 50

**Nota:** In un programma reale, bisognerebbe verificare che il divisore non sia zero prima di eseguire la divisione.

**Codice equivalente (pseudocodice):**

```
1 INIZIO
2   LEGGI a, b, op
3   SE op = '+' ALLORA
4     ris <- a + b
5   ALTRIMENTI SE op = '-' ALLORA
6     ris <- a - b
7   ALTRIMENTI SE op = '*' ALLORA
8     ris <- a * b
9   ALTRIMENTI
10    ris <- a / b
11  FINE SE
12  SCRIVI ris
13 FINE
```

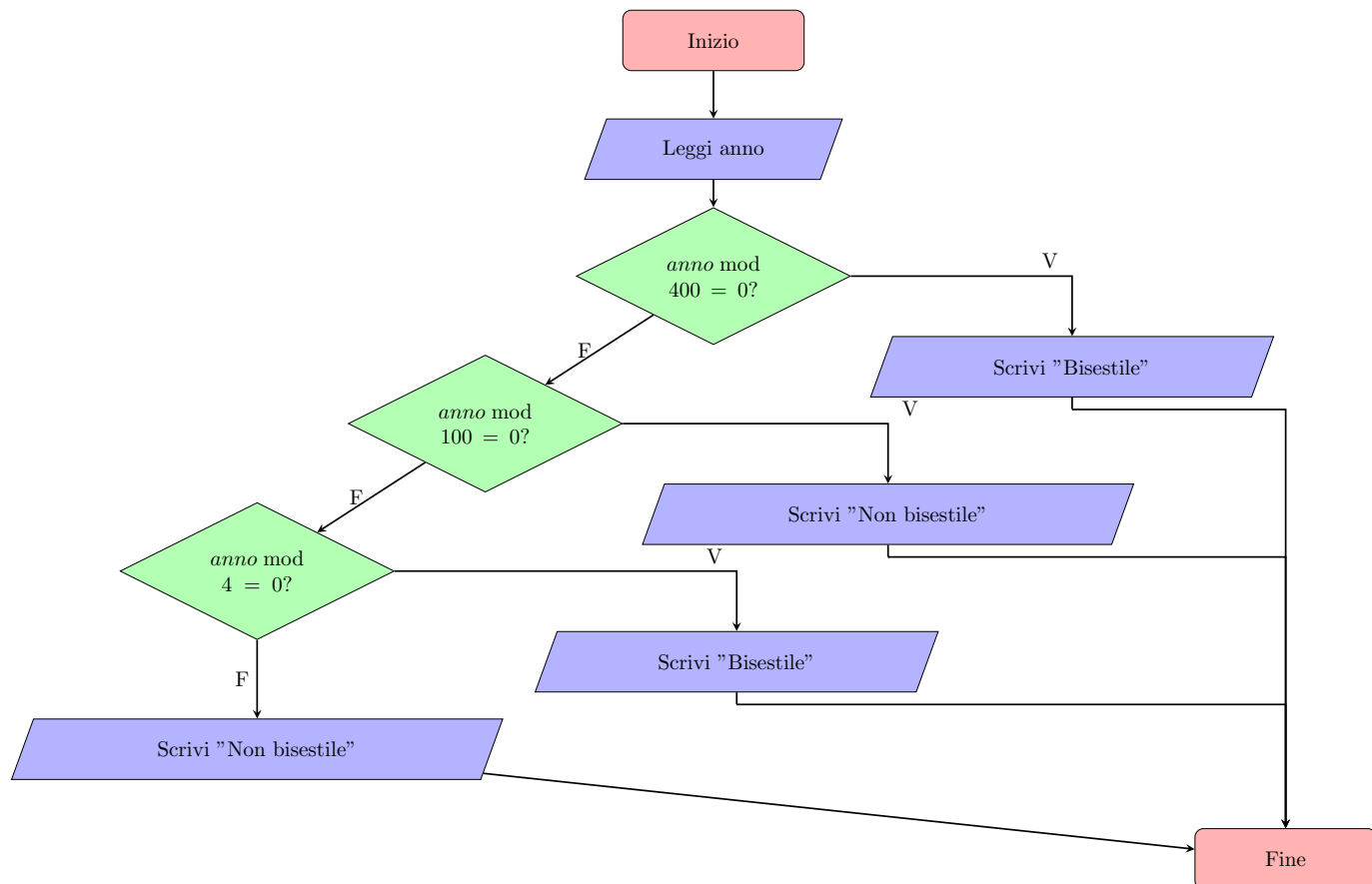
### 3.5.7 Esercizio 7: Anno Bisestile

#### Traccia

Creare un diagramma di flusso che determini se un anno è bisestile. Un anno è bisestile se:

- È divisibile per 4 E non è divisibile per 100
- OPPURE è divisibile per 400

#### Soluzione:



#### Spiegazione della regola:

1. Se l'anno è divisibile per 400 → è sempre bisestile (es. 2000)
2. Altrimenti, se è divisibile per 100 → NON è bisestile (es. 1900)
3. Altrimenti, se è divisibile per 4 → è bisestile (es. 2024)
4. Altrimenti → NON è bisestile (es. 2023)

#### Esempi di esecuzione:

- Input: 2024 →  $2024 \bmod 4 = 0$  (e non per 100) → **Bisestile**
- Input: 2000 →  $2000 \bmod 400 = 0$  → **Bisestile**
- Input: 1900 →  $1900 \bmod 100 = 0$  (ma non per 400) → **Non bisestile**
- Input: 2023 →  $2023 \bmod 4 \neq 0$  → **Non bisestile**

#### Codice equivalente (pseudocodice):

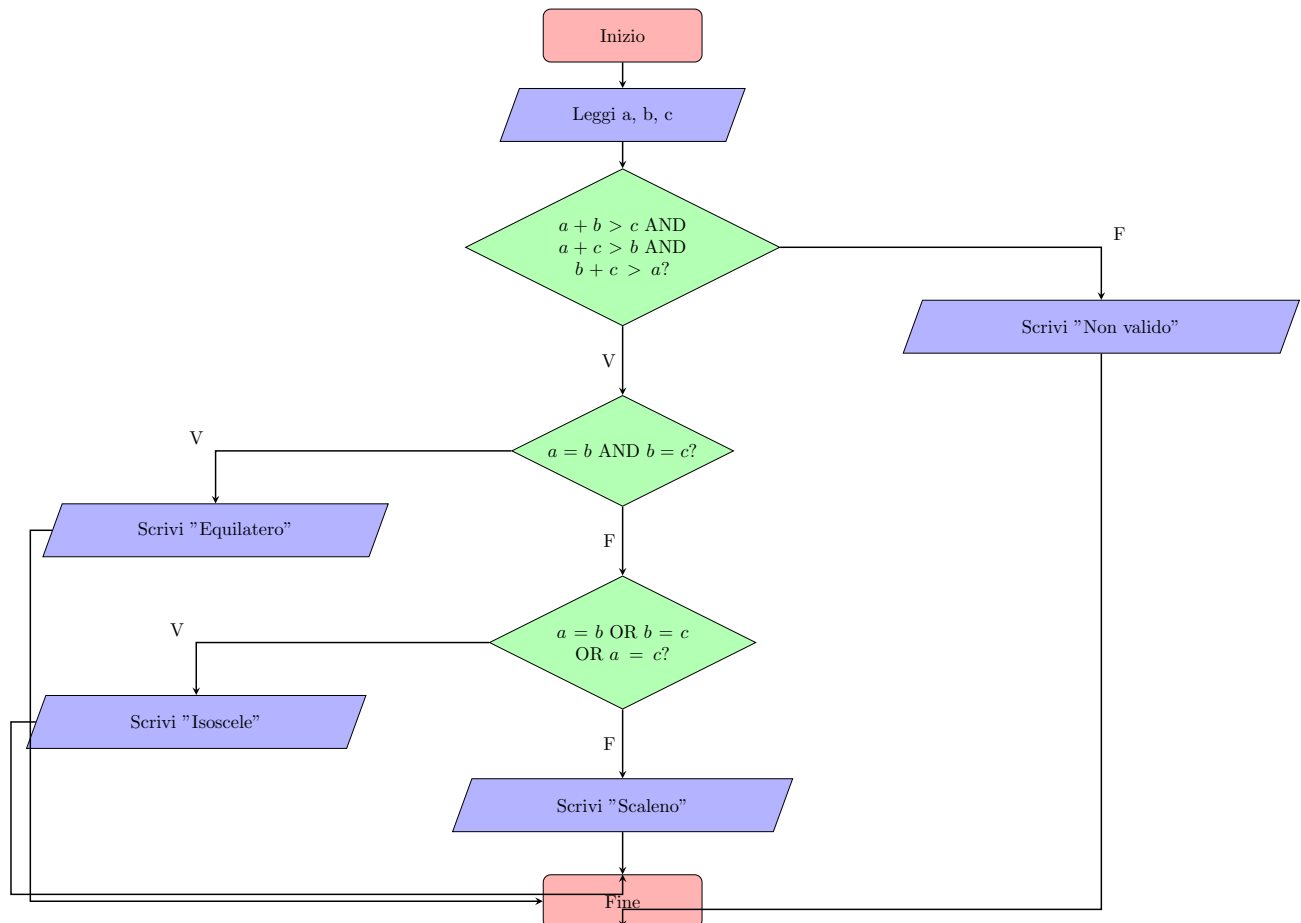
```
1 INIZIO
2   LEGGI anno
3   SE anno MOD 400 = 0 ALLORA
4     SCRIVI "Bisestile"
5   ALTRIMENTI SE anno MOD 100 = 0 ALLORA
6     SCRIVI "Non bisestile"
7   ALTRIMENTI SE anno MOD 4 = 0 ALLORA
8     SCRIVI "Bisestile"
9   ALTRIMENTI
10    SCRIVI "Non bisestile"
11  FINE SE
12 FINE
```

### 3.5.8 Esercizio 8: Triangolo Validato e Classificazione

#### Traccia

Creare un diagramma di flusso che, dati tre lati, verifichi se formano un triangolo valido e lo classifichi come equilatero, isoscele o scaleno. Un triangolo è valido se la somma di due lati qualsiasi è maggiore del terzo.

#### Soluzione:



#### Spiegazione:

1. Il programma legge i tre lati **a**, **b**, **c**
2. Verifica se formano un triangolo valido (disuguaglianza triangolare):
  - Se NO: stampa "Non valido" e termina
  - Se SÌ: procede con la classificazione
3. Verifica se tutti i lati sono uguali → "Equilatero"
4. Altrimenti verifica se almeno due lati sono uguali → "Isoscele"
5. Altrimenti → "Scaleno" (tutti i lati diversi)

#### Esempi di esecuzione:

- Input: a=5, b=5, c=5 → Equilatero
- Input: a=5, b=5, c=7 → Isoscele
- Input: a=3, b=4, c=5 → Scaleno



- Input:  $a=1, b=2, c=10 \rightarrow$  Non valido ( $1+2 \not\geq 10$ )

**Codice equivalente (pseudocodice):**

```
1 INIZIO
2   LEGGI a, b, c
3   SE (a+b>c) AND (a+c>b) AND (b+c>a) ALLORA
4     SE a=b AND b=c ALLORA
5       SCRIVI "Equilatero"
6     ALTRIMENTI SE a=b OR b=c OR a=c ALLORA
7       SCRIVI "Isoscele"
8     ALTRIMENTI
9       SCRIVI "Scaleno"
10    FINE SE
11  ALTRIMENTI
12    SCRIVI "Non valido"
13  FINE SE
14 FINE
```

### 3.6 Esercizi Proposti sui Blocchi Condizionali

Per consolidare le conoscenze sui blocchi condizionali, prova a risolvere i seguenti esercizi:

1. **Sconto sul prezzo:** Creare un diagramma che applichi uno sconto del 10% se il prezzo è maggiore di 100€, altrimenti nessuno sconto
2. **Massimo tra tre numeri:** Creare un diagramma che determini il maggiore tra tre numeri
3. **Ordinamento di due numeri:** Creare un diagramma che ordini due numeri in ordine crescente
4. **Divisibilità:** Creare un diagramma che verifichi se un numero è divisibile per 3 e per 5 contemporaneamente
5. **Equazione di secondo grado:** Creare un diagramma che, dati i coefficienti  $a$ ,  $b$ ,  $c$ , determini se l'equazione  $ax^2 + bx + c = 0$  ha soluzioni reali, controllando il discriminante  $\Delta = b^2 - 4ac$
6. **Categoria peso:** Creare un diagramma che, dato il peso di una persona, la classifichi come sottopeso ( $< 50$  kg), normopeso ( $50 - 80$  kg) o sovrappeso ( $> 80$  kg)
7. **Segno del prodotto:** Creare un diagramma che, dati due numeri, determini se il loro prodotto è positivo, negativo o zero senza calcolare il prodotto
8. **Conversione voto:** Creare un diagramma che converta un voto in centesimi (0-100) in trentesimi (0-30), con lode se il voto è maggiore o uguale a 99
9. **Accesso a servizio:** Creare un diagramma che verifichi se un utente può accedere a un servizio: deve avere età  $\geq 18$  E (saldo  $\geq 10$  O abbonamento attivo)
10. **Stagione:** Creare un diagramma che, dato il numero del mese (1-12), stampi la stagione corrispondente

#### Suggerimento

Quando risolvi esercizi con condizioni multiple, ricorda:

- AND: Entrambe le condizioni devono essere vere
- OR: Almeno una delle condizioni deve essere vera
- Usa le parentesi per chiarire la priorità delle operazioni
- Disegna prima i casi limite (valori estremi) per verificare la logica

## 4 Esercizi Proposti (Variabili e Condizioni)

Per consolidare le conoscenze acquisite, prova a risolvere i seguenti esercizi creando i relativi diagrammi di flusso:

1. Creare un diagramma che calcoli il perimetro di un rettangolo (formula:  $P = 2 \times (base + altezza)$ )
2. Creare un diagramma che calcoli l'area di un cerchio dato il raggio (formula:  $A = \pi \times r^2$ , usa  $\pi \approx 3.14$ )
3. Creare un diagramma che converta una temperatura da Fahrenheit a Celsius (formula:  $C = (F - 32)/1.8$ )
4. Creare un diagramma che calcoli la media ponderata di due voti con i rispettivi pesi
5. Creare un diagramma che calcoli il resto della divisione tra due numeri interi senza usare l'operatore modulo
6. Creare un diagramma che calcoli l'IVA (22%) su un prezzo e il prezzo finale
7. Creare un diagramma che, dati i tre lati di un triangolo, calcoli il perimetro e l'area usando la formula di Erone

### Nota

**Prossimo argomento:** Nella prossima sezione tratteremo i cicli (while, do-while, for) per eseguire operazioni ripetute.

## 5 Cicli (Iterazione)

### 5.1 Cosa sono i Cicli

I **cicli** (o strutture iterative) permettono di ripetere un blocco di istruzioni più volte, fino a quando una determinata condizione è soddisfatta. Sono fondamentali per automatizzare operazioni ripetitive senza dover scrivere lo stesso codice molte volte.

#### 5.1.1 Perché usare i cicli?

I cicli sono essenziali quando dobbiamo:

- Eseguire la stessa operazione su molti dati
- Contare o sommare valori
- Cercare un elemento in una sequenza
- Validare input fino a quando è corretto
- Elaborare dati fino a una condizione di stop

### 5.2 Tipi di Cicli

Esistono tre tipi principali di cicli, ognuno adatto a situazioni diverse:

#### 5.2.1 Ciclo WHILE (pre-condizionale)

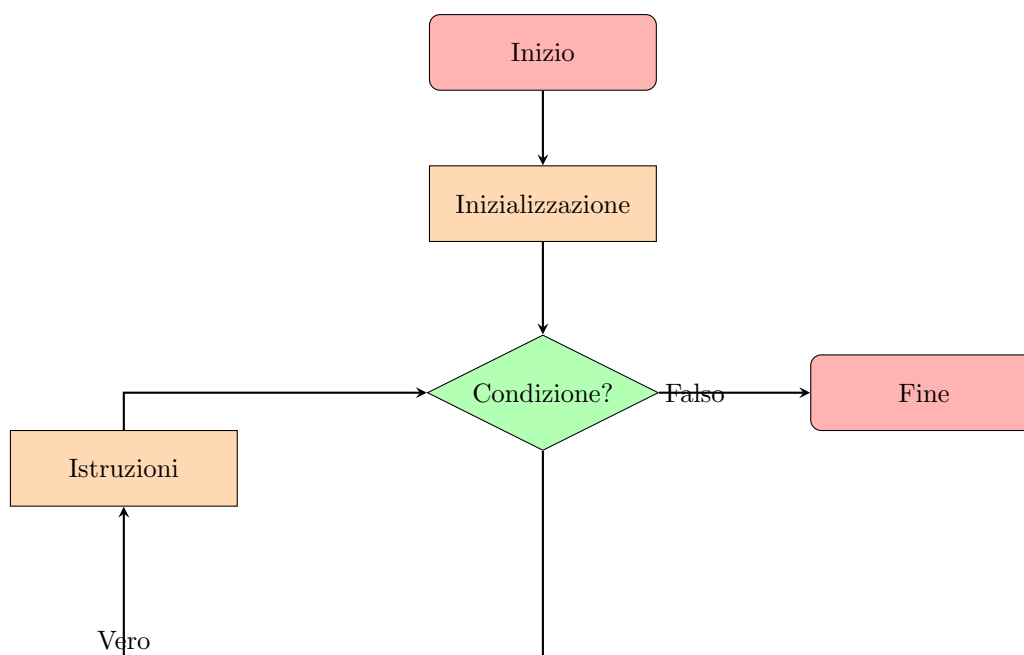
Il ciclo **while** verifica la condizione *prima* di eseguire il blocco di istruzioni. Se la condizione è falsa fin dall'inizio, il blocco non viene mai eseguito.

**Sintassi:**

```
1 MENTRE condizione vera FARE
2   istruzioni
3   TORNA A MENTRE
```

**Caratteristiche:**

- Controllo *prima* dell'esecuzione
- Può eseguire 0 o più iterazioni
- Usato quando non si sa a priori quante volte ripetere



### 5.2.2 Ciclo DO-WHILE (post-condizionale)

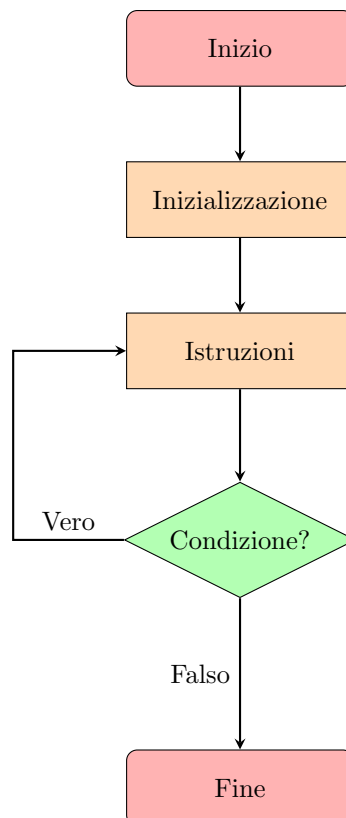
Il ciclo **do-while** esegue il blocco di istruzioni *prima* di verificare la condizione. Garantisce almeno un'esecuzione del blocco.

**Sintassi:**

```
1 FARE
2   istruzioni
3 MENTRE condizione
```

**Caratteristiche:**

- Controllo *dopo* l'esecuzione
- Esegue almeno 1 volta (anche se la condizione è falsa)
- Usato quando si deve eseguire almeno una volta (es. menu, validazione input)



### 5.2.3 Ciclo FOR (a contatore)

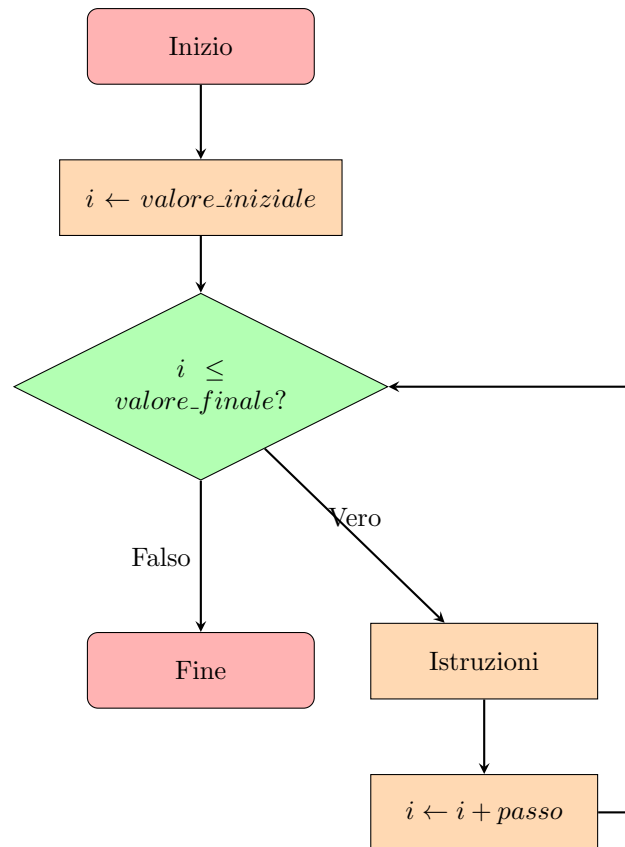
Il ciclo **for** è usato quando si conosce esattamente il numero di iterazioni da eseguire. Utilizza una variabile contatore.

**Sintassi:**

```
1 PER i DA valore_iniziale A valore_finale [PASSO incremento] FARE
2   istruzioni
3 FINE PER
```

**Caratteristiche:**

- Numero di iterazioni noto a priori
- Gestisce automaticamente il contatore
- Usato per iterare su intervalli numerici



### 5.3 Componenti Fondamentali di un Ciclo

Ogni ciclo ben strutturato deve avere:

1. **Inizializzazione:** Preparare le variabili prima del ciclo
2. **Condizione di controllo:** Determina se continuare o terminare
3. **Corpo del ciclo:** Le istruzioni da ripetere
4. **Aggiornamento:** Modificare le variabili per avvicinarsi alla fine

#### Attenzione: Cicli Infiniti!

Un **ciclo infinito** si verifica quando la condizione di uscita non diventa mai falsa. Questo causa il blocco del programma!

#### Esempio di errore:

```
1 i <- 0
2 MENTRE i < 10 FARE
3     SCRIVI i
4     // ERRORE: manca i <- i + 1
5 FINE MENTRE
```

Assicurati sempre che il ciclo modifichi le variabili della condizione!

### 5.4 Confronto tra i Cicli

Tipo	Controllo	Quando usarlo	Iterazioni
WHILE	Prima	Numero iterazioni sconosciuto	0 o più
DO-WHILE	Dopo	Almeno una esecuzione necessaria	1 o più
FOR	Prima	Numero iterazioni noto	0 o più

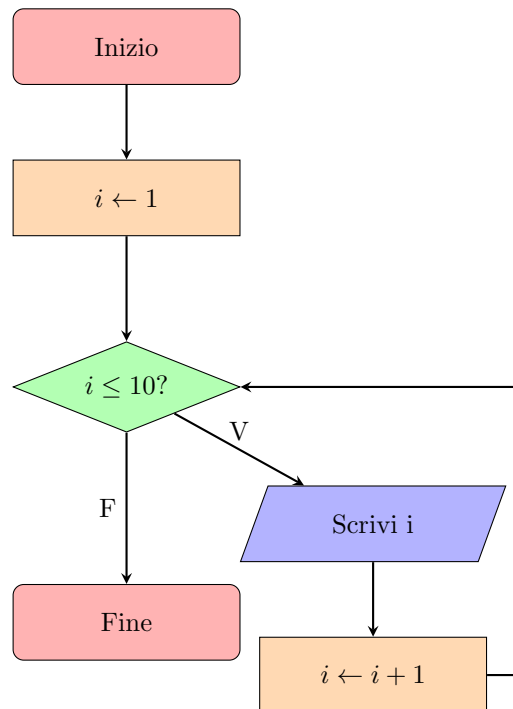
## 5.5 Esercizi Svolti con Cicli

### 5.5.1 Esercizio 1: Contare da 1 a 10 (Ciclo FOR)

#### Traccia

Creare un diagramma di flusso che stampi i numeri da 1 a 10 utilizzando un ciclo FOR.

#### Soluzione:



#### Spiegazione:

1. Inizializza il contatore  $i$  a 1
2. Verifica se  $i \leq 10$ :
  - Se VERO: stampa il valore di  $i$ , incrementa  $i$  di 1 e torna alla verifica
  - Se FALSO: esce dal ciclo e termina

#### Traccia di esecuzione:

- Iterazione 1:  $i = 1$ , stampa 1,  $i$  diventa 2
- Iterazione 2:  $i = 2$ , stampa 2,  $i$  diventa 3
- ... (continua fino a  $i = 10$ )
- Iterazione 10:  $i = 10$ , stampa 10,  $i$  diventa 11
- Verifica:  $11 \leq 10$  è FALSO  $\rightarrow$  esce dal ciclo

**Output:** 1 2 3 4 5 6 7 8 9 10

**Codice equivalente (pseudocodice):**

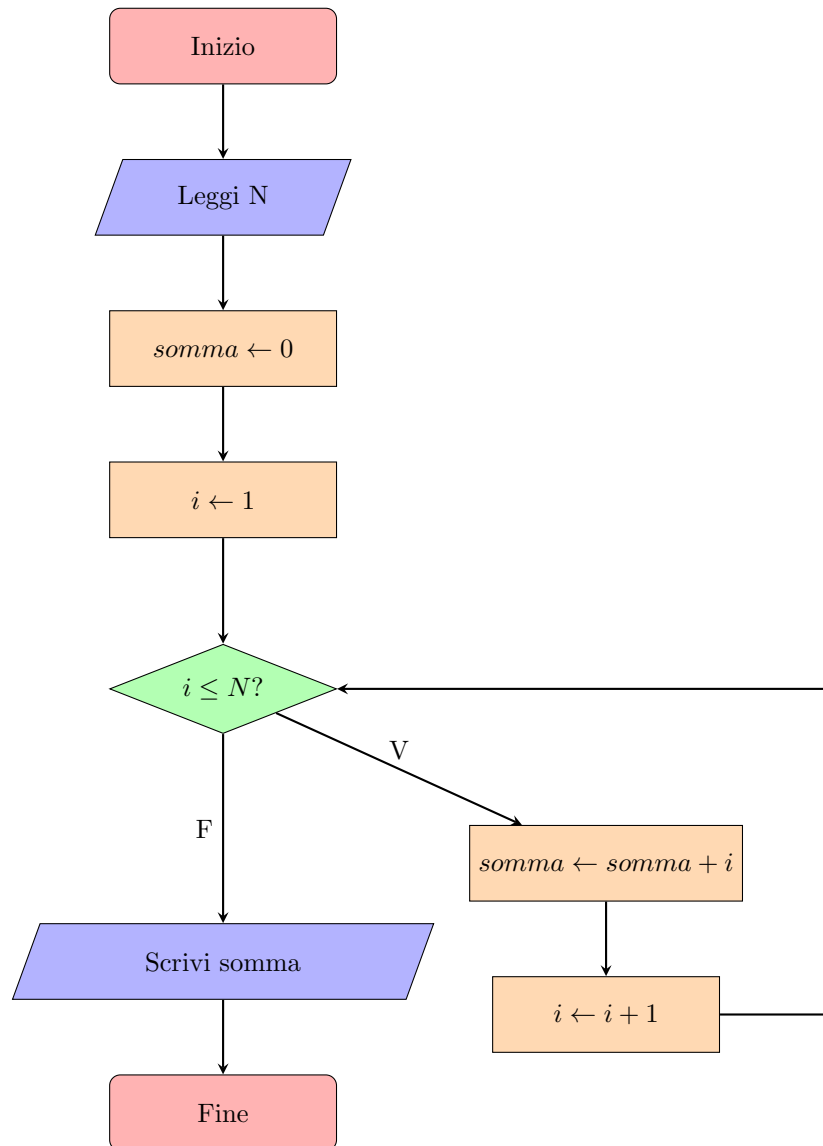
```

1 INIZIO
2   PER i DA 1 A 10 FARE
3     SCRIVI i
4   FINE PER
5 FINE
  
```



**5.5.2 Esercizio 2: Somma dei primi N numeri (Ciclo WHILE)****Traccia**

Creare un diagramma di flusso che calcoli la somma dei primi N numeri naturali (da 1 a N), dove N è inserito dall'utente.

**Soluzione:****Spiegazione:**

1. L'utente inserisce il valore di N
2. Inizializza  $somma = 0$  e  $i = 1$
3. Finché  $i \leq N$ :
  - Aggiunge  $i$  alla somma
  - Incrementa  $i$  di 1
4. Stampa il risultato finale

**Esempio di esecuzione (N=5):**

- Input:  $N = 5$

- Inizializzazione:  $somma = 0, i = 1$
- Iter. 1:  $somma = 0 + 1 = 1, i = 2$
- Iter. 2:  $somma = 1 + 2 = 3, i = 3$
- Iter. 3:  $somma = 3 + 3 = 6, i = 4$
- Iter. 4:  $somma = 6 + 4 = 10, i = 5$
- Iter. 5:  $somma = 10 + 5 = 15, i = 6$
- Verifica:  $6 \leq 5$  è FALSO  $\rightarrow$  Output: 15

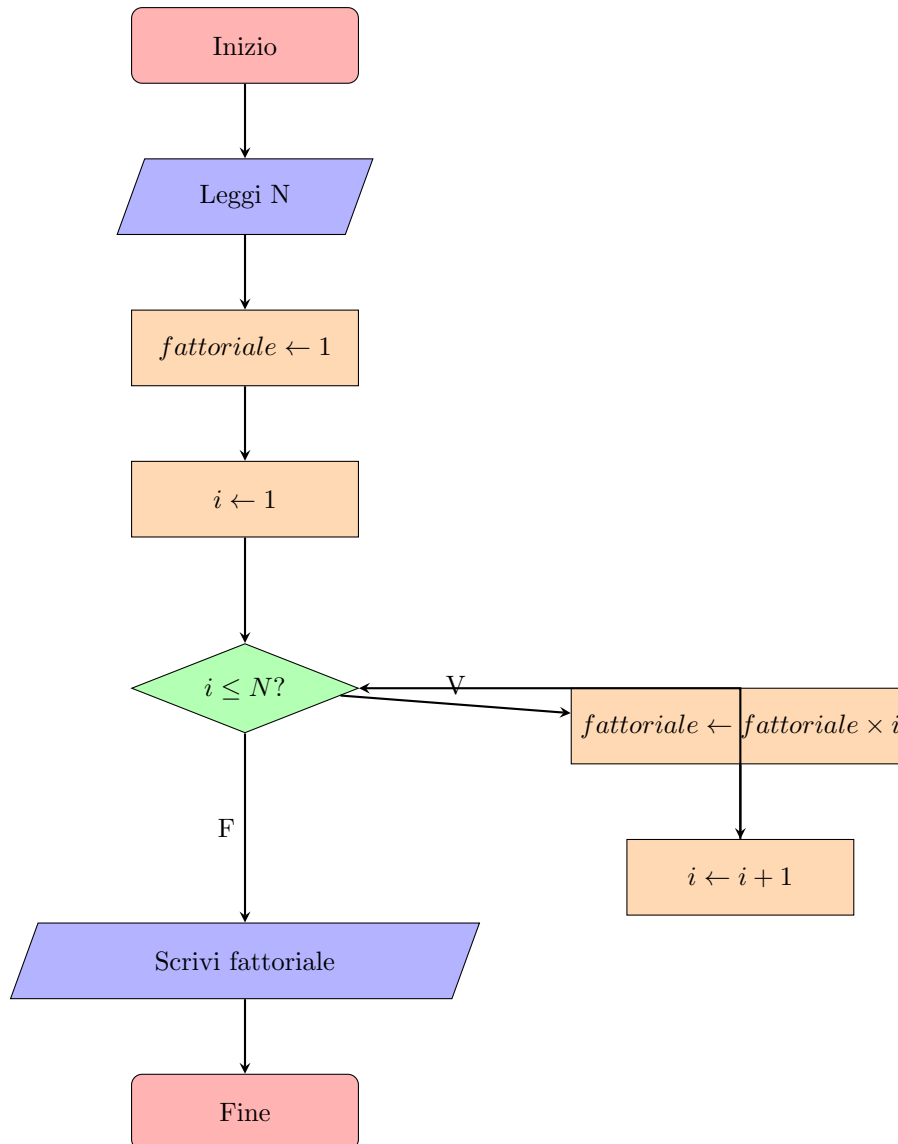
**Formula matematica:**  $\sum_{i=1}^N i = \frac{N(N+1)}{2}$

**Codice equivalente (pseudocodice):**

```
1 INIZIO
2   LEGGI N
3   somma <- 0
4   i <- 1
5   MENTRE i <= N FARE
6       somma <- somma + i
7       i <- i + 1
8   FINE MENTRE
9   SCRIVI somma
10 FINE
```

**5.5.3 Esercizio 3: Fattoriale di un Numero (Ciclo FOR)****Traccia**

Creare un diagramma di flusso che calcoli il fattoriale di un numero  $N$  ( $N! = 1 \times 2 \times 3 \times \dots \times N$ ).

**Soluzione:****Spiegazione:**

1. L'utente inserisce il numero  $N$
2. Inizializza  $fattoriale = 1$  (elemento neutro della moltiplicazione)
3. Per ogni  $i$  da 1 a  $N$ :
  - Moltiplica  $fattoriale$  per  $i$
4. Stampa il risultato

**Esempio di esecuzione (N=5):**

- Input:  $N = 5$

- Inizializzazione:  $fattoriale = 1$
- Iter. 1:  $fattoriale = 1 \times 1 = 1$
- Iter. 2:  $fattoriale = 1 \times 2 = 2$
- Iter. 3:  $fattoriale = 2 \times 3 = 6$
- Iter. 4:  $fattoriale = 6 \times 4 = 24$
- Iter. 5:  $fattoriale = 24 \times 5 = 120$
- Output: 120 ( $5! = 120$ )

**Nota:** Il fattoriale cresce molto rapidamente:  $10! = 3,628,800$

**Codice equivalente (pseudocodice):**

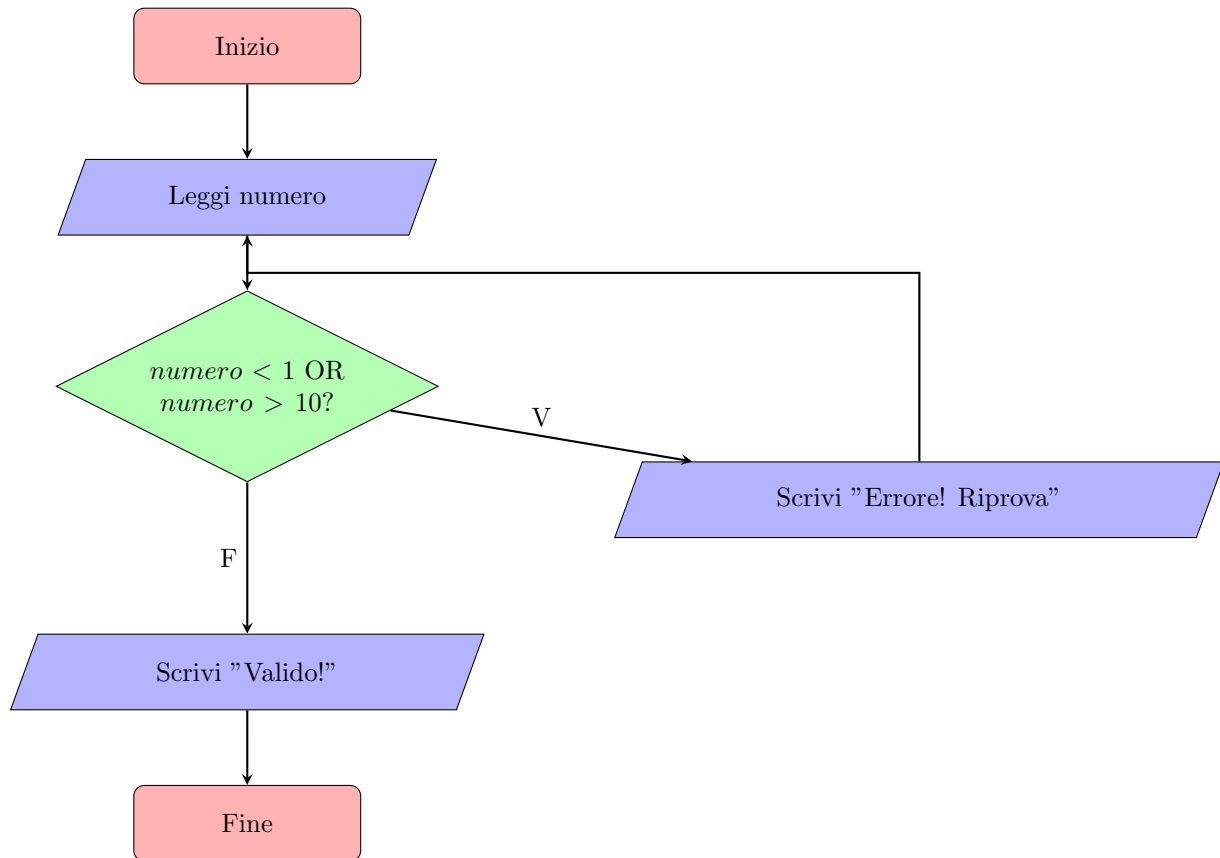
```
1 INIZIO
2   LEGGI N
3   fattoriale <- 1
4   PER i DA 1 A N FARE
5       fattoriale <- fattoriale * i
6   FINE PER
7   SCRIVI fattoriale
8 FINE
```

#### 5.5.4 Esercizio 4: Validazione Input (Ciclo DO-WHILE)

##### Traccia

Creare un diagramma di flusso che chieda all'utente di inserire un numero compreso tra 1 e 10. Se il numero non è valido, continua a chiederlo fino a quando non viene inserito un valore corretto.

##### Soluzione:



##### Spiegazione:

1. Il programma chiede di inserire un numero
2. Verifica se il numero è fuori dall'intervallo  $[1, 10]$ :
  - Se VERO (numero non valido): stampa messaggio di errore e torna all'input
  - Se FALSO (numero valido): stampa "Valido!" e termina

**Perché DO-WHILE?** Almeno un tentativo di input deve essere fatto, quindi il ciclo DO-WHILE è perfetto per la validazione.

##### Esempio di esecuzione:

- Tentativo 1: utente inserisce 15 → "Errore! Riprova"
- Tentativo 2: utente inserisce -3 → "Errore! Riprova"
- Tentativo 3: utente inserisce 7 → "Valido!" → fine

##### Codice equivalente (pseudocodice):

```

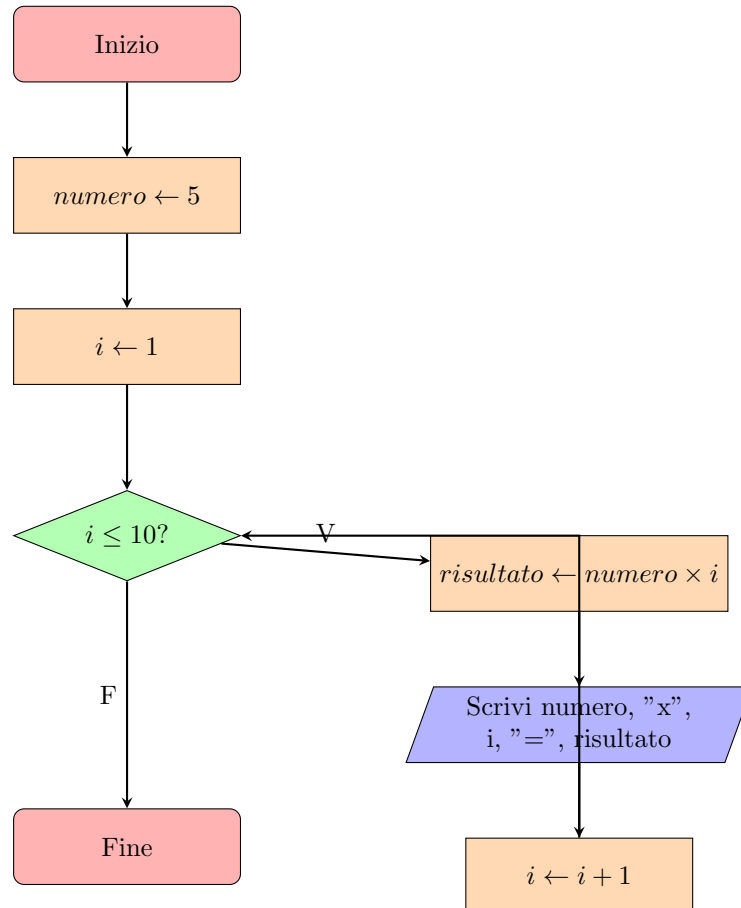
1 INIZIO
2   FARE
3     LEGGI numero
4     SE numero < 1 OR numero > 10 ALLORA

```

```
5      SCRIVI "Errore! Riprova"
6      FINE SE
7      MENTRE numero < 1 OR numero > 10
8      SCRIVI "Valido!"
9  FINE
```

**5.5.5 Esercizio 5: Tavola Pitagorica (Cicli Annidati)****Traccia**

Creare un diagramma di flusso che stampi la tavola pitagorica del 5 ( $5 \times 1$ ,  $5 \times 2$ , ...  $5 \times 10$ ) usando un ciclo FOR.

**Soluzione:****Spiegazione:**

1. Imposta il numero base (5)
2. Per ogni  $i$  da 1 a 10:
  - Calcola  $risultato = numero \times i$
  - Stampa l'operazione e il risultato

**Output:**

5 x 1 = 5  
5 x 2 = 10  
5 x 3 = 15  
5 x 4 = 20  
5 x 5 = 25  
5 x 6 = 30  
5 x 7 = 35  
5 x 8 = 40  
5 x 9 = 45  
5 x 10 = 50

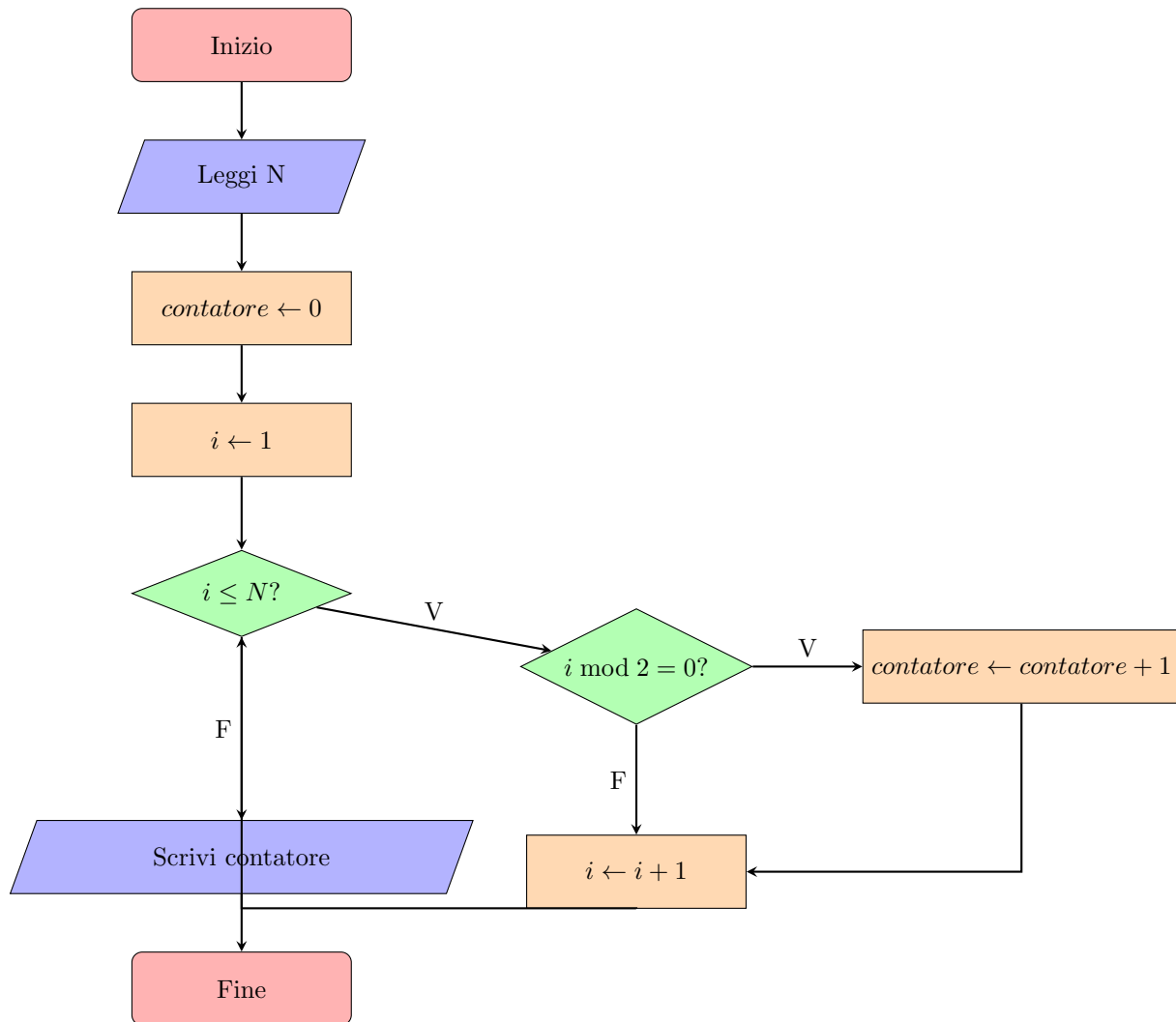
**Codice equivalente (pseudocodice):**

```
1 INIZIO
2     numero <- 5
3     PER i DA 1 A 10 FARE
4         risultato <- numero * i
5         SCRIVI numero, " x ", i, " = ", risultato
6     FINE PER
7 FINE
```



**5.5.6 Esercizio 6: Contare i Numeri Pari (Ciclo WHILE)****Traccia**

Creare un diagramma di flusso che conti quanti numeri pari ci sono tra 1 e N (dove N è inserito dall'utente).

**Soluzione:****Spiegazione:**

1. L'utente inserisce N
2. Inizializza il contatore a 0 e  $i$  a 1
3. Per ogni numero da 1 a N:
  - Se il numero è pari (resto della divisione per 2 è zero), incrementa il contatore
  - Passa al numero successivo
4. Stampa quanti numeri pari sono stati trovati

**Esempio di esecuzione (N=10):**

- Input:  $N = 10$
- Numeri pari trovati: 2, 4, 6, 8, 10

- Output: 5 (ci sono 5 numeri pari tra 1 e 10)

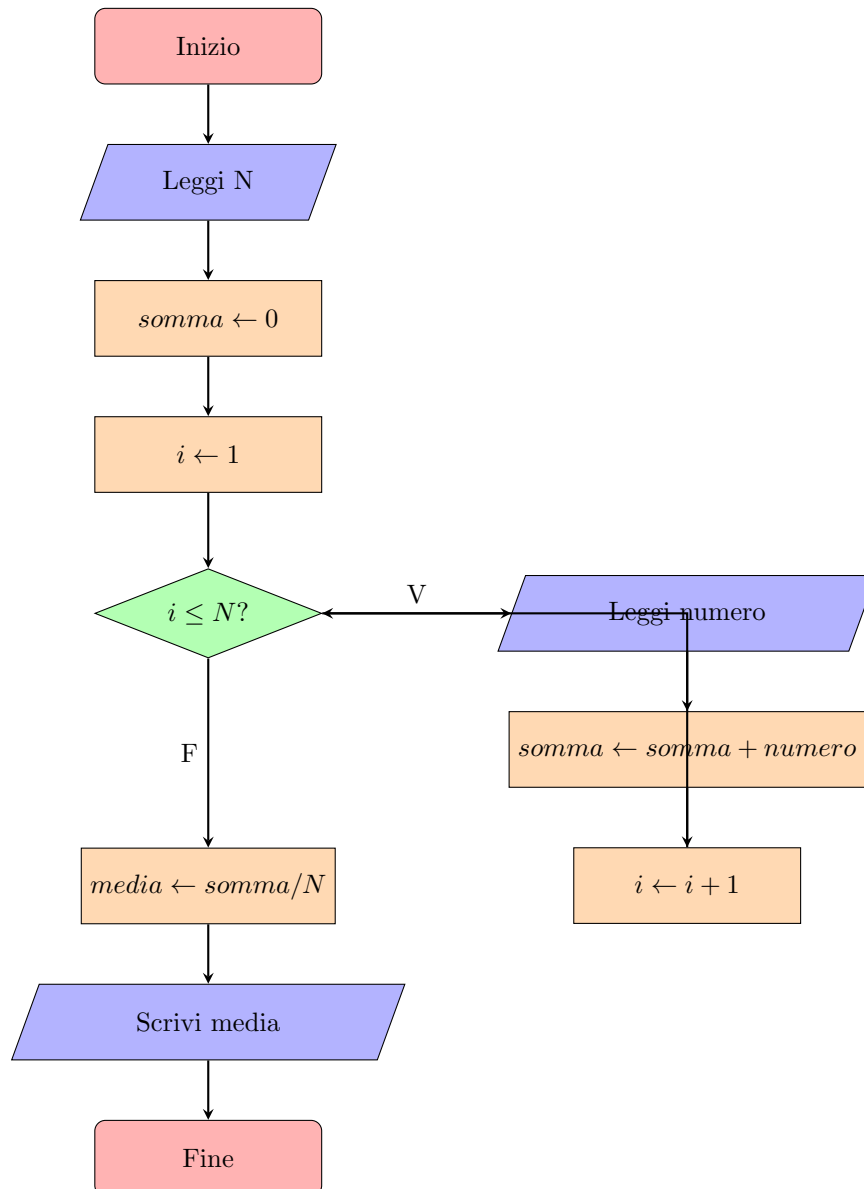
**Formula diretta:** Per N pari:  $\frac{N}{2}$ , per N dispari:  $\frac{N-1}{2}$

**Codice equivalente (pseudocodice):**

```
1 INIZIO
2   LEGGI N
3   contatore <- 0
4   i <- 1
5   MENTRE i <= N FARE
6     SE i MOD 2 = 0 ALLORA
7       contatore <- contatore + 1
8     FINE SE
9     i <- i + 1
10  FINE MENTRE
11  SCRIVI contatore
12 FINE
```

**5.5.7 Esercizio 7: Media di N Numeri (Ciclo WHILE)****Traccia**

Creare un diagramma di flusso che calcoli la media di N numeri inseriti dall'utente. Prima chiede quanti numeri verranno inseriti, poi li legge uno alla volta e infine calcola la media.

**Soluzione:****Spiegazione:**

1. Chiede quanti numeri verranno inseriti (N)
2. Inizializza la somma a 0
3. Per N volte:
  - Legge un numero
  - Lo aggiunge alla somma
4. Calcola la media:  $media = somma / N$
5. Stampa la media

**Esempio di esecuzione:**

- Input:  $N = 4$
- Input numeri: 8, 6, 7, 9
- Somma:  $8 + 6 + 7 + 9 = 30$
- Media:  $30/4 = 7.5$
- Output: 7.5

**Codice equivalente (pseudocodice):**

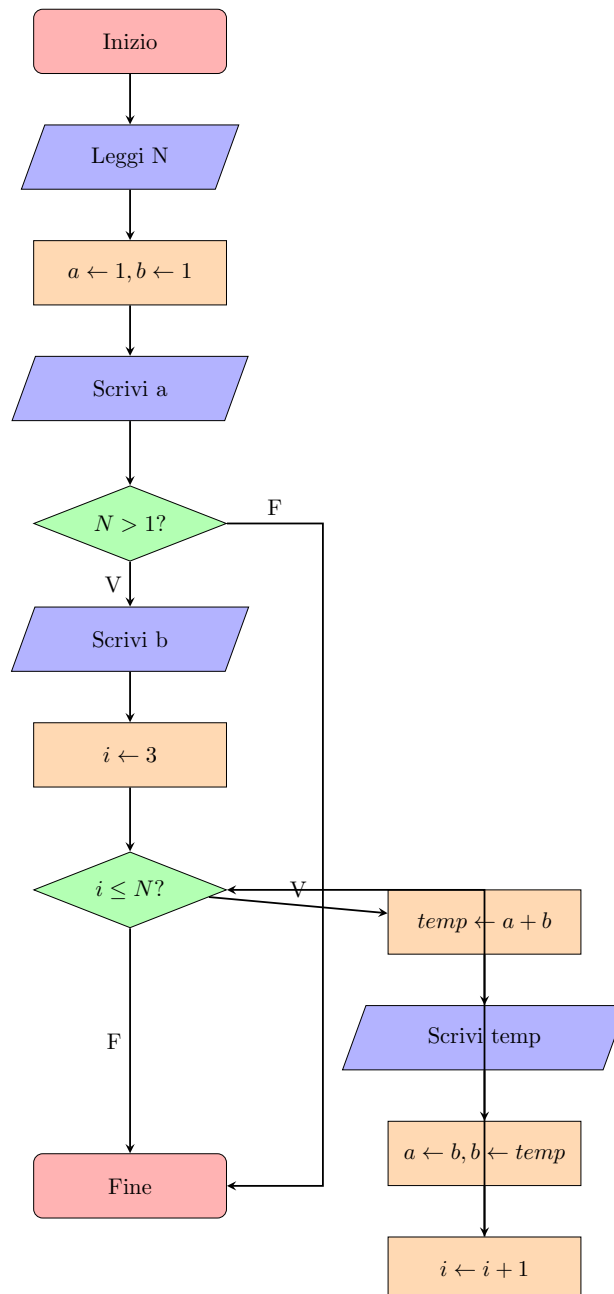
```
1 INIZIO
2     LEGGI N
3     somma <- 0
4     i <- 1
5     MENTRE i <= N FARE
6         LEGGI numero
7         somma <- somma + numero
8         i <- i + 1
9     FINE MENTRE
10    media <- somma / N
11    SCRIVI media
12 FINE
```

## 5.5.8 Esercizio 8: Sequenza di Fibonacci (Ciclo FOR)

## Traccia

Creare un diagramma di flusso che stampi i primi N numeri della sequenza di Fibonacci, dove ogni numero è la somma dei due precedenti (1, 1, 2, 3, 5, 8, 13, ...).

## Soluzione:



## Spiegazione:

1. Chiede quanti numeri di Fibonacci stampare
2. Inizializza i primi due numeri:  $a = 1, b = 1$
3. Stampa il primo numero
4. Se  $N > 1$ , stampa anche il secondo numero
5. Per i numeri successivi (da 3 a N):

- Calcola il nuovo numero:  $temp = a + b$
- Stampa  $temp$
- Aggiorna:  $a = b$  e  $b = temp$

**Esempio di esecuzione (N=8):**

- Output: 1, 1, 2, 3, 5, 8, 13, 21
- Calcoli:  $1 + 1 = 2$ ,  $1 + 2 = 3$ ,  $2 + 3 = 5$ ,  $3 + 5 = 8$ ,  $5 + 8 = 13$ ,  $8 + 13 = 21$

**Codice equivalente (pseudocodice):**

```
1 INIZIO
2   LEGGI N
3   a <- 1, b <- 1
4   SCRIVI a
5   SE N > 1 ALLORA
6     SCRIVI b
7     PER i DA 3 A N FARE
8       temp <- a + b
9       SCRIVI temp
10      a <- b
11      b <- temp
12    FINE PER
13  FINE SE
14 FINE
```

## 5.6 Esercizi Proposti sui Cicli

Per consolidare le conoscenze sui cicli, prova a risolvere i seguenti esercizi:

1. **Numeri dispari:** Creare un diagramma che stampi tutti i numeri dispari da 1 a 50 (Ciclo FOR)
2. **Conto alla rovescia:** Creare un diagramma che conti alla rovescia da 10 a 0 (Ciclo FOR)
3. **Somma fino a zero:** Creare un diagramma che continui a leggere numeri e sommarli finché l'utente non inserisce 0, poi stampi la somma totale (Ciclo WHILE)
4. **Massimo di N numeri:** Creare un diagramma che trovi il numero più grande tra N numeri inseriti dall'utente (Ciclo WHILE)
5. **Password:** Creare un diagramma che chieda una password e continui a richiederla finché non viene inserita quella corretta "1234" (Ciclo DO-WHILE)
6. **Potenza:** Creare un diagramma che calcoli  $base^{esponente}$  usando un ciclo (es.  $2^5 = 2 \times 2 \times 2 \times 2 \times 2$ ) (Ciclo FOR)
7. **Divisori:** Creare un diagramma che stampi tutti i divisori di un numero N (Ciclo FOR)
8. **Numero primo:** Creare un diagramma che verifichi se un numero è primo (divisibile solo per 1 e se stesso) (Ciclo WHILE)
9. **MCD:** Creare un diagramma che calcoli il Massimo Comun Divisore tra due numeri usando l'algoritmo di Euclide (Ciclo WHILE)
10. **Quadrato di asterischi:** Creare un diagramma che stampi un quadrato di asterischi di dimensione  $N \times N$  (Cicli annidati FOR)

### Suggerimenti per i Cicli

- **Inizializzazione:** Imposta sempre i valori iniziali delle variabili prima del ciclo
- **Condizione di uscita:** Assicurati che la condizione diventi falsa prima o poi
- **Aggiornamento:** Ricorda di modificare le variabili nel corpo del ciclo
- **Test con casi limite:** Prova il ciclo con 0, 1 e molte iterazioni
- **Traccia manuale:** Esegui il ciclo passo-passo su carta per verificare la logica
- **Cicli annidati:** Fai attenzione a usare variabili diverse per cicli esterni e interni

## 6 Appendice: Convenzioni e Suggerimenti

### 6.1 Convenzioni di Scrittura

- **Nomi delle variabili:** Usare nomi significativi (es. `prezzo`, `eta`, `somma`)
- **Assegnazione:** Si può usare sia  $\leftarrow$  che  $=$  (es.  $x \leftarrow 5$  oppure  $x = 5$ )
- **Operatori matematici:**
  - Somma:  $+$
  - Sottrazione:  $-$
  - Moltiplicazione:  $\times$  o  $*$
  - Divisione:  $/$
  - Modulo (resto):  $\%$  o *mod*
- **Operatori di confronto:**  $=, \neq, <, >, \leq, \geq$

### 6.2 Suggerimenti per Disegnare Diagrammi

1. **Pianifica prima di disegnare:** Scrivi prima l'algoritmo in pseudocodice
2. **Mantieni la semplicità:** Un blocco = un'operazione semplice
3. **Usa spaziature uniformi:** Mantieni distanze regolari tra i blocchi
4. **Allinea i blocchi:** Cerca di mantenere un allineamento verticale
5. **Etichetta le frecce:** Nelle decisioni, indica sempre Vero/Falso o Sì/No
6. **Verifica il flusso:** Segui mentalmente il percorso per verificare la correttezza
7. **Testa con esempi:** Prova il diagramma con valori concreti

### 6.3 Errori Comuni da Evitare

- Dimenticare di inizializzare le variabili prima di usarle
- Confondere l'operatore di assegnazione ( $\leftarrow$ ) con quello di confronto ( $=$ )
- Non specificare la direzione Vero/Falso nei rombi decisionali
- Creare cicli infiniti senza condizione di uscita
- Lasciare percorsi senza sbocco (che non arrivano alla fine)
- Usare lo stesso nome per variabili diverse
- Fare operazioni con variabili di tipo incompatibile