



Rappresentazione degli algoritmi

1 Diagramma di flusso

Utilizzare il diagramma di flusso per rappresentare gli algoritmi che risolvono i seguenti problemi (vedere le slide '1.1 Concetti Base dell'Informatica: Algoritmi'). Il problema 7 é di tipo non numerico.

Problema 1 - Contare fino a 100 (vedere Figura 1).

ANALISI dell'algoritmo:

1. deve usare in Ingresso una variabile A inizializzata a 0 ($A = 0$) per contare fino a 100;
2. deve eseguire un ciclo per incrementare di 1 il valore della variabile A , ossia $A = A + 1$;
3. deve controllare se la variabile ha raggiunto il valore desiderato per poter terminare, ossia $A = 100$;
4. altrimenti deve tornare al passo 2.

Variabili in Ingresso: variabile A per effettuare il conteggio, inizializzata a 0.

Variabili in Uscita: nessuna.

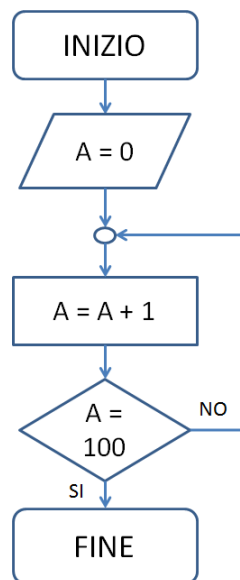


Figura 1: Problema 1

Problema 2 - Dati due numeri $N1$ e $N2$ in Ingresso stabilire quale dei due sia il maggiore. Nel caso in cui i due numeri siano uguali comunicarlo con un apposito messaggio (vedere Figura 2).

Variabili in Ingresso: variabili $N1$ e $N2$.

Variabili in Uscita: messaggio che può indicare il numero maggiore, il numero minore, o se sono uguali.

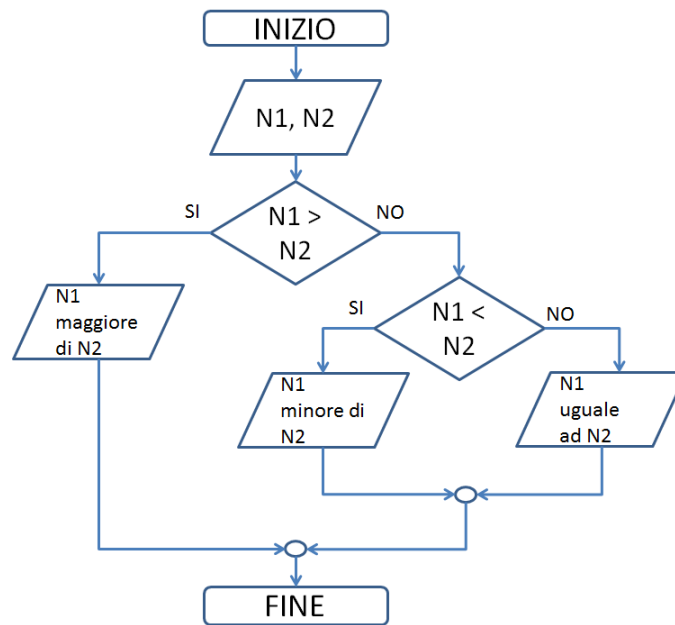


Figura 2: Problema 2

VERIFICA dell'algoritmo descritto in Figura 2 al variare dei valori delle variabili in Ingresso:

- Caso 1:
 1. Siano $N1 = 2$ e $N2 = 5$.
 2. L'esito della prima condizione ($N1 > N2$) è falsa, di conseguenza si esegue la seconda condizione.
 3. L'esito della seconda condizione ($N1 < N2$) è vera, quindi in Uscita si avrà il messaggio 'N1 minore di N2';
 4. L'algoritmo termina.
- Caso 2:
 1. Siano $N1 = 5$ e $N2 = 2$.
 2. L'esito della prima condizione ($N1 > N2$) è vera, di conseguenza in Uscita si avrà il messaggio 'N1 maggiore di N2';
 3. L'algoritmo termina.
- Caso 3:
 1. Siano $N1 = 5$ e $N2 = 5$.
 2. L'esito della prima condizione ($N1 > N2$) è falsa, di conseguenza si esegue la seconda condizione.

3. L'esito della seconda condizione ($N1 < N2$) é falsa, quindi in Uscita si avrà il messaggio 'N1 uguale a N2';
4. L'algoritmo termina.

Problema 3 - Calcolare l'area A di una superficie irregolare quale triangolo T e rettangolo R , data la base b e l'altezza a in Ingresso per calcolare l'area del triangolo $A = \frac{b \times a}{2}$ o l'area del rettangolo $A_R = b \times a$ (vedere Figura 3).

ANALISI dell'algoritmo:

1. deve usare una variabile A per l'area;
2. deve usare in Ingresso tre variabili: altezza a , base b e un flag f che indica il tipo di superficie irregolare (sia $f = 1$ per triangolo, $f = 3$ per rettangolo);
3. deve controllare il tipo di superficie per calcolare l'area corrispondente;
4. in caso di flag corretto deve calcolare l'area e terminare l'algoritmo.
5. altrimenti deve terminare l'algoritmo.

Variabili in Ingresso: variabili $a = 3$, $b = 2$ e $f = 1$.

Variabili in Uscita: nessuna.

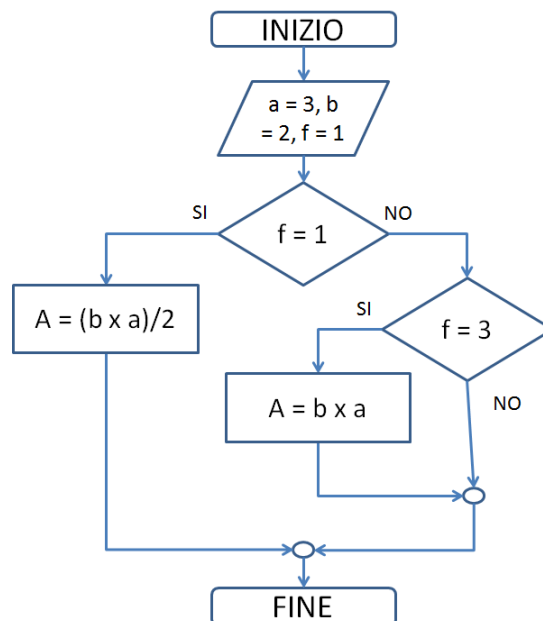


Figura 3: Problema 3

Per esercizio, effettuare la verifica dell'algoritmo considerando $f = 3$; modificare il diagramma di flusso in modo da avere in Uscita un messaggio di errore in caso di flag errato o il valore dell'area A .

Problema 4 - Calcolare il massimo di una sequenza di numeri interi detti N numero degli elementi della sequenza ed x elemento della sequenza. Fornire il massimo max (vedere Figura 4).

Variabili in Ingresso: variabili N e x .

Variabili in Uscita: max .

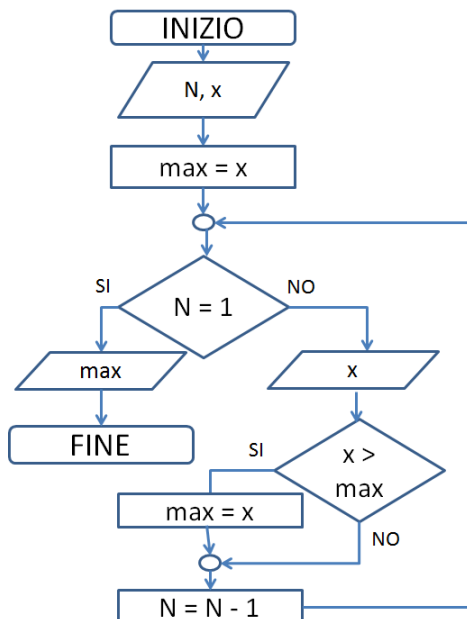


Figura 4: Problema 4

Per esercizio, effettuare la verifica dell'algoritmo considerando la sequenza di numeri interi 2, 0, 5, 4, 6, 10, 3 ed $N = 7$; dire cosa succede se la condizione nel primo controllo é del tipo $N = 0$; modificare il diagramma di flusso inserendo il controllo $N = 0$.

Problema 5 - Convertire la temperatura dalla scala Fahrenheit F a Celsius C e viceversa. Fornire il risultato (vedere Figura 5). Si considerino le seguenti elaborazioni: $C = \frac{5}{9}(F - 32)$, $F = \frac{9}{5}(C) + 32$.

ANALISI dell'algoritmo:

1. deve usare in Ingresso tre variabili: variabile F per la temperatura in Fahrenheit, variabile C per la temperatura in Celsius e una variabile flag f per indicare la conversione desiderata (sia $f = 1$ per Fahrenheit, $f = 3$ per Celsius);
2. deve controllare il valore del flag;
3. in caso di flag errato scrivere un messaggio 'conversione errata' e terminare l'algoritmo;
4. in caso di flag corretto effettuare l'elaborazione, fornire il valore della temperatura e terminare l'algoritmo.

Variabili in Ingresso: variabili C , F e f .

Variabili in Uscita: F o C , o messaggio di errore.

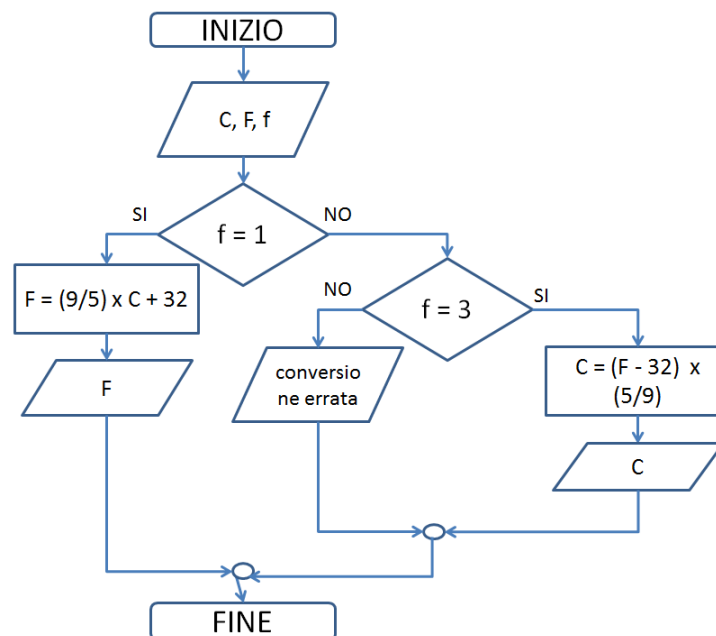


Figura 5: Problema 5

Per esercizio, effettuare la verifica dell'algoritmo considerando i seguenti valori delle variabili in Ingresso:

- $C = 0$, $F = 0$, $f = 1$;
- $C = 0$, $F = 0$, $f = 3$;
- $C = 10$, $F = 0$, $f = 1$;
- $C = 0$, $F = 10$, $f = 3$.

Modificare il diagramma di flusso:

- per evitare l'operazione di moltiplicazione nella conversione da Fahrenheit a Celsius quando $C = 0$.

- per evitare l'operazione di moltiplicazione nella conversione da Celsius a Fahrenheit quando $F = 0$.

Problema 6 - Data una lista di N elementi con indice da 1 a N dire il tipo di operazione gestita dall'algoritmo descritto in Figura 6.

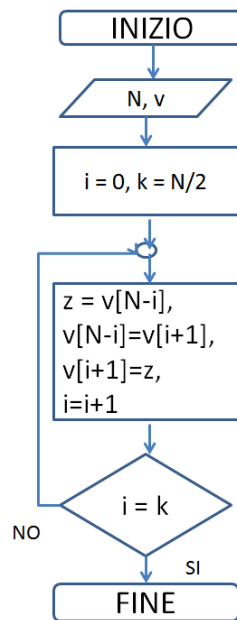


Figura 6: Problema 6

Si osserva quanto segue:

- ogni volta che il ciclo viene svolto, l'elemento $(i + 1)$ -esimo del vettore viene scambiato con l'elemento $(N - i)$ -esimo;
- il ciclo è effettuato un numero di volte pari a $\frac{N}{2}$, ossia metà dell'insieme;
- alla prima iterazione con $i = 0$ e $k = \frac{N}{2}$, l'algoritmo inverte l'ultimo elemento del vettore con il primo elemento dello stesso;
- il ciclo va da $i = 0$ a $i = k = \frac{N}{2}$.

Si può dire che l'algoritmo inverte l'ordine degli elementi del vettore.

VERIFICA dell'algoritmo:

Siano $N = 7$ e $v = [1, 2, 5, 6, 0, 10, 7]$.

- prima iterazione $i = 0$, a $k = \frac{7}{2} = 3$:
 1. $z = v[7]$, ossia $z = 7$;
 2. $v[7] = v[1]$, ossia $v[7] = 1$;
 3. $v[1] = z$, ossia $v[1] = 7$;
 4. $i = 1$;
 5. a termine della prima iterazione si ha $v = [7, 2, 5, 6, 0, 10, 1]$.
- seconda iterazione $i = 1$, a $k = \frac{7}{2} = 3$:
 1. $z = v[6]$, ossia $z = 10$;
 2. $v[6] = v[2]$, ossia $v[6] = 2$;

3. $v[2] = z$, ossia $v[2] = 10$;
 4. $i=2$;
 5. a termine della seconda iterazione si ha $v = [7, 10, 5, 6, 0, 2, 1]$.
- terza iterazione $i = 2$, a $k = \frac{7}{2} = 3$:
 1. $z = v[5]$, ossia $z = 0$;
 2. $v[5] = v[3]$, ossia $v[5] = 5$;
 3. $v[5] = z$, ossia $v[3] = 0$;
 4. $i=3$;
 5. a termine della terza iterazione si ha $v = [7, 10, 0, 6, 5, 2, 1]$.

Problema 7 - Replicare le operazioni di prelievo o deposito effettuate da una persona al bancomat (vedere Figura 7).

ANALISI dell'algoritmo:

1. deve chiedere in Ingresso la password;
2. deve controllare la password;
3. deve chiedere in Ingresso il tipo di operazione e importo;
4. in caso di deposito deve aggiungere l'importo al saldo;
5. in caso di prelievo deve controllare il saldo:
 - se l'importo $>$ del saldo, deve formulare un messaggio di errore;
 - se l'importo $<$ del saldo, deve sottrarre l'importo al saldo e fornire il saldo.
6. deve chiedere se si vuole procedere con altra operazione;
7. altrimenti deve terminare.

Variabili in Ingresso: password, tipo di operazione, importo.

Variabili in Uscita: errore o contante.

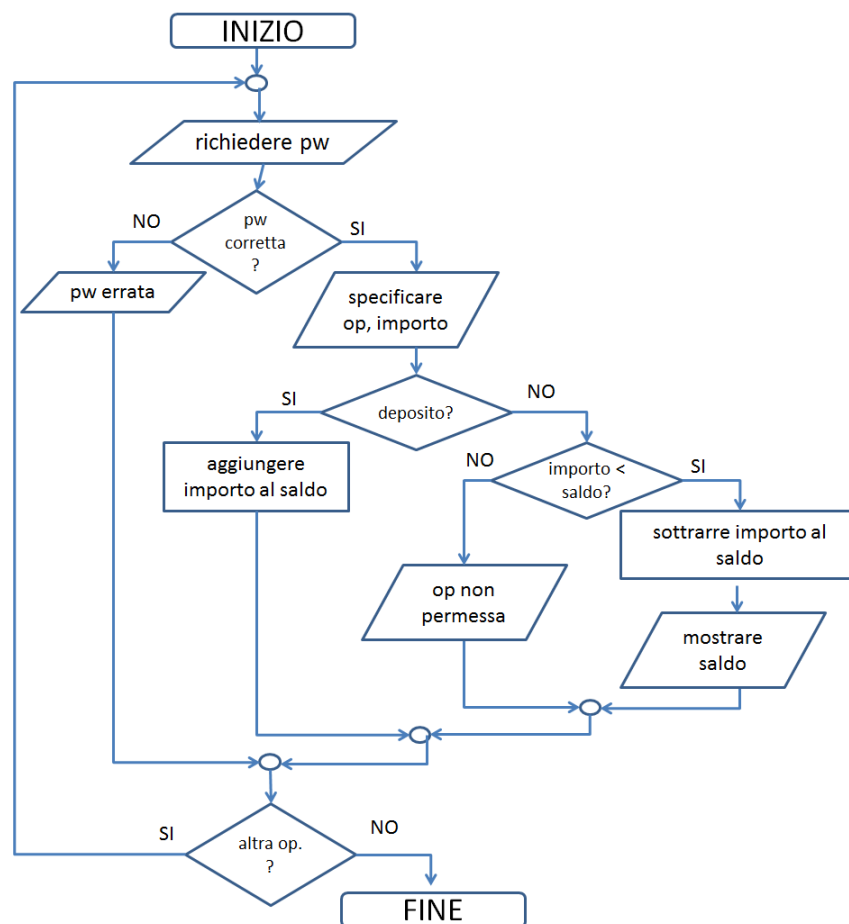


Figura 7: Problema 7

2 Pseudocodice

Utilizzare lo pseudocodice per rappresentare gli algoritmi (vedere le slide '1.1 Concetti Base dell'Informatica: Algoritmi'). In Appendice sono riportate alcune istruzioni fondamentali utili per la rappresentazione degli algoritmi.

Problema 1 - Dato un numero n , controllare se n è pari o dispari, usando il costrutto IF-THEN-ELSE. In base all'esito del controllo assegnare alla variabile m uno dei valori espressi dalla seguente funzione:

$$m(n) = \begin{cases} \frac{n}{2} & \text{if } n \text{ é pari} \\ \frac{n-1}{2} & \text{if } n \text{ é dispari} \end{cases}$$

Pseudocodice Problema 1:

```
n <- 10  
  
if n mod 2 = 0 then  
    m <- n/2  
else  
    m <- (n-1)/2
```

Si osserva che l'operazione *mod* restituisce il resto della divisione tra numeri interi: ossia $a \bmod b = r$ dove r è il resto della divisione tra a e b .

Problema 2 - Dato un numero f e un contatore c pari a 5, usando il costrutto WHILE, determinare il nuovo valore di f dato da

$$f = f \times c$$

fino a quando il contatore non é pari a zero.

Pseudocodice Problema 2:

```
f <- 1
c <- 5

while c > 0 do
  f <- f x c
  c <- c -1
```

Problema 3 - Dato un numero y e un contatore c pari a 5, usando il costrutto REPEAT-UNTIL, determinare il nuovo valore di y dato da

$$y = y + c$$

fino a quando il contatore é diverso da zero.

Pseudocodice Problema 3:

```
c <- 5
y <- 0

repeat
  y <- y + c
  c <- c -1
until c = 0
```

Problema 4 - Mettere in una variabile *sum* la somma delle prime 10 iterazioni ottenute con il costrutto FOR e restituire il valore finale.

Pseudocodice Problema 4:

```
sum <- 0

for i <- 1 to 10 do
  sum <- sum + i

write sum
```

3 Appendice

3.1 Costrutto condizionale

3.1.1 IF-THEN-ELSE

```
if <condizione> then <blocco1> else <blocco2>
```

Se la *< condizione >* é verificata, si esegue il contenuto del *< blocco1 >*, altrimenti quello del *< blocco2 >*. Figura 8 mostra l'istruzione rappresentata con il diagramma di flusso.

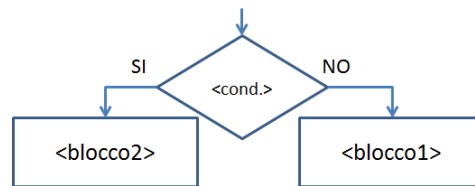


Figura 8: IF-THEN-ELSE

La parte *else* e relativo *< blocco2 >* del costrutto sono opzionali.

```
if <condizione> then <blocco1>
```

3.2 Costrutti iterativi

3.3 WHILE

```
while <condizione> do  
    <blocco>
```

Esprime un ciclo. Se la *< condizione >* é soddisfatta, si esegue il contenuto del *< blocco >*. La *< condizione >* é verificata prima di ogni iterazione. L'iterazione si ferma quando la *< condizione >* non é più vera.

3.3.1 REPEAT-UNTIL

```
repeat  
    <blocco>  
until <condizione>
```

Esprime un ciclo. Se la *< condizione >* non é soddisfatta, si esegue il contenuto del *< blocco >*. La *< condizione >* é verificata dopo ogni iterazione. L'iterazione si ferma quando la *< condizione >* é vera. Figura 9 mostra l'istruzione rappresentata con il diagramma di flusso.

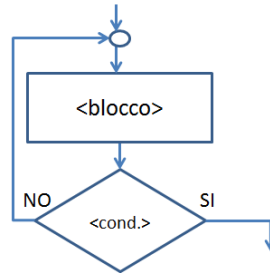


Figura 9: REPEAT-UNTIL

3.3.2 FOR

```
for <condizione> do
  <blocco>
```

Esprime un ciclo. Se la *< condizione >* é soddisfatta, si esegue il contenuto del *< blocco >*. La *< condizione >* é verificata prima di ogni iterazione. La *< condizione >* può essere del tipo indicato di seguito:

```
<variabile> = <espressione1> to <espressione2>
```

In questo caso la *< condizione >* esprime il valore di una *< variabile >* il cui valore iniziale é indicato da *< espressione1 >*, mentre quello finale da *< espressione2 >*. Ad ogni iterazione viene incrementato di 1 il valore della *< variabile >* fino a quando quest'ultima non raggiunge il valore finale indicato da *< espressione2 >*. Figura 10 mostra l'istruzione rappresentata con il diagramma di flusso.

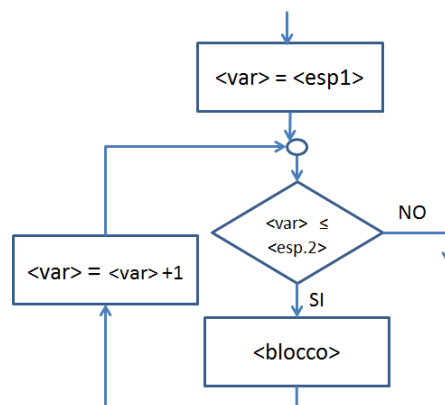


Figura 10: FOR-DO

3.4 Costrutti di Ingresso e Uscita

3.4.1 WRITE

```
write <variabile>
```

L'elaboratore invia al dispositivo di uscita (per esempio il monitor) il valore della $\langle \textit{variabile} \rangle$.

3.4.2 READ

`read <variabile>`

L'elaboratore riceve dal dispositivo di ingresso (per esempio la tastiera) il valore specificato dall'utente assegnato alla $\langle \textit{variabile} \rangle$.