

FONDAMENTI DI PROGRAMMAZIONE

PROGRAMMAZIONE STRUTTURATA IN C

Dott. Federico Concone

federico.concone@unipa.it

Informazioni utili

- Ricevimento:
 - Edificio 6, 3° Piano, Laboratorio Intelligenza Artificiale e Sistemi Distribuiti;
- Orario ricevimento:
 - Da concordare tramite e-mail;
- Testo consigliato:
 - Deitel, Deitel, Il linguaggio C – Fondamenti e tecniche di programmazione 8 Ed., Pearson, Italia, 2016
- Link materiale:
 - goo.gl/mrcWL7

Informazioni utili

- COMUNICAZIONE PER GLI **STUDENTI ISCRITTI AD ANNI SUCCESSIVI AL PRIMO**;
- ISCRIZIONE ALLA PROVA IN **ITINERE DEL 18/04/2018** CON DOCENTE DI RIFERIMENTO **MARCO ORTOLANI**:
 - FONDAMENTI DI PROGRAMMAZIONE IN C;
 - ARCHITETTURE AVANZATE DEI CALCOLATORI;
- PER POTERSI ISCRIVERE **INVIARE UNA E-MAIL** CON I SEGUENTI CAMPI:
 1. *Destinatari:* marco.ortolani@unipa.it e federico.concone@unipa.it;
 2. *Oggetto:* iscrizione alla prova in itinere di giorno 18/04/2018;
 3. *Corpo:* Nome, Cognome, Matricola, Anno di corso;
- LA PROVA DEL 19/04/2018 E' RISERVATA AI FUORI CORSO;

Sommario

- Strumenti di supporto alla programmazione
- Pseudocodice
- Diagrammi di flusso
- Strutture di controllo
 - Strutture di condizione
 - Strutture di iterazione
- Sintassi alternative
- Errori comuni
- Esercizi

Strumenti di supporto alla programmazione

- Prima di scrivere un programma che risolva un certo problema, è ***necessario capire il problema stesso e progettare attentamente una soluzione;***
- La soluzione a qualsiasi problema prevede l'esecuzione di una serie di azioni svolte in un ordine preciso (*algoritmo*);
- I programmatore alle prime armi tendono a “complicare” il programma introducendo azioni prive di regole (*spaghetti programming*);
- Vi sono diversi strumenti che aiutano il programmatore a progettare correttamente un algoritmo:
 - Pseudocodice;
 - Diagrammi di flusso;

Pseudocodice

- Cos'è?

"Pseudocode is an artificial and informal language that helps you develop algorithms";
- Lo pseudocodice è particolarmente utile per progettare algoritmi che possono essere facilmente tradotti in programmi C;
- Caratteristiche:
 - E' molto simile al linguaggio parlato;
 - I programmi scritti in pseudocodice *non vengono eseguiti sul calcolatore*;
 - E' composto solamente di ***action statements***;
- Gli action statement vengono eseguiti quando lo pseudocodice viene tradotto in C (o in qualsiasi altro linguaggio di programmazione);

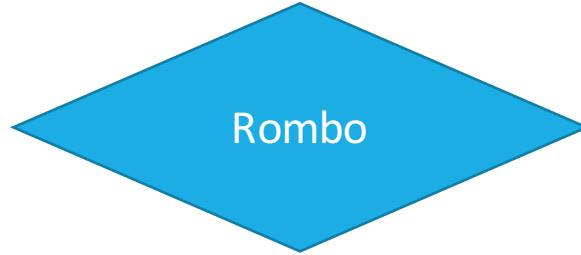
Diagrammi di flusso

- I diagrammi di flusso:

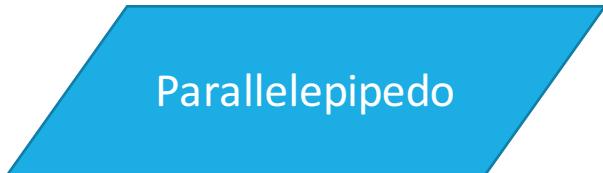
- sono uno strumento alternativo per progettare un algoritmo;
- sono una *rappresentazione grafica* dell'algoritmo;
- vengono descritti da simboli special-purpose;



Rettangolo



Rombo



Parallelepipedo

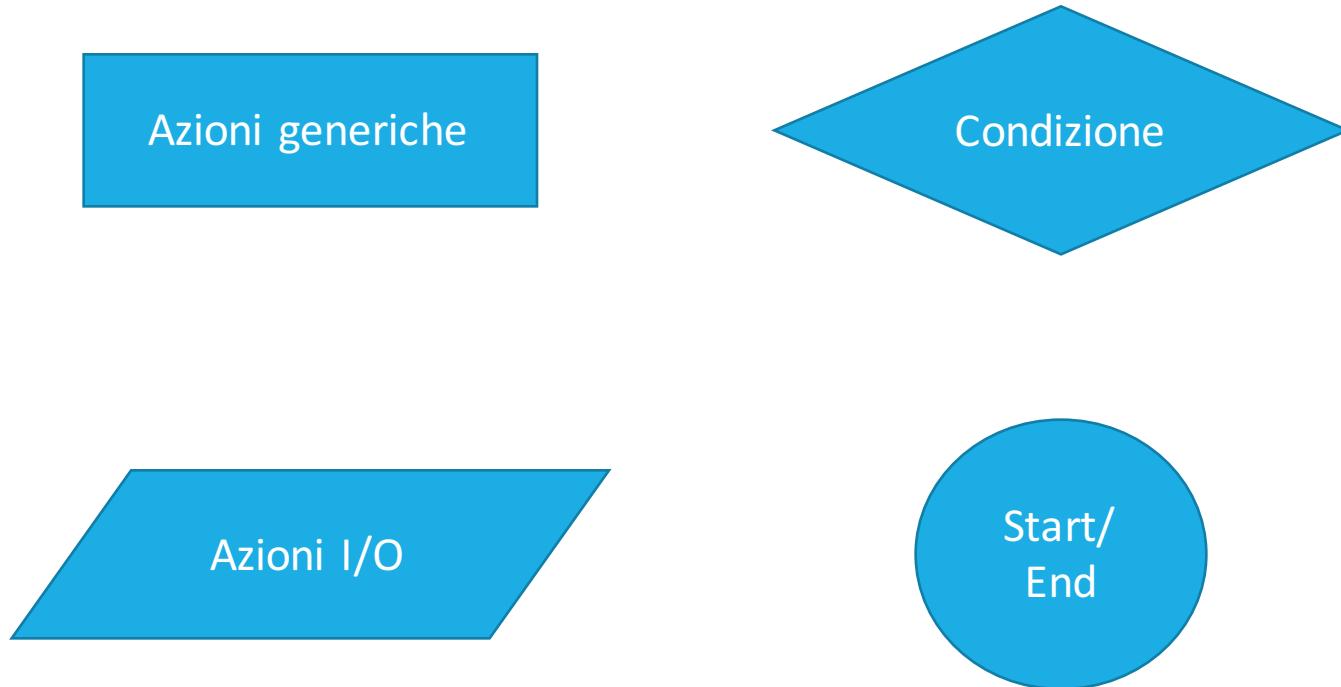


Cerchio

Diagrammi di flusso

- I diagrammi di flusso:

- sono uno strumento alternativo per progettare un algoritmo;
- sono una *rappresentazione grafica* dell'algoritmo;
- vengono descritti da simboli special-purpose;



Strutture di controllo

- Il linguaggio C fornisce 4 tipi di *strutture di selezione* e 3 *strutture di ripetizione*:

- **If** statement;
- **If...else** statement;
- **Nested If...else** statement;
- **Switch** statement;



Strutture di condizione

Strutture di controllo

- Il linguaggio C fornisce 4 tipi di *strutture di selezione* e 3 *strutture di ripetizione*:

- **If** statement;
- **If...else** statement;
- **Nested If...else** statement;
- **Switch** statement;



Strutture di condizione

- **While** statement;
- **Do...while** statement;
- **For** statement;

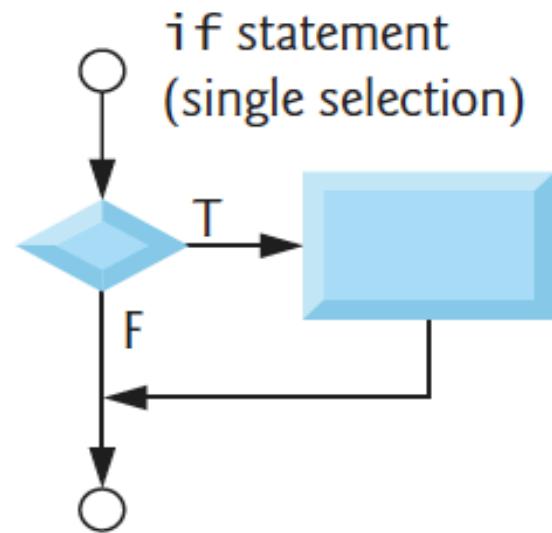


Strutture di iterazione

Strutture di condizione: IF

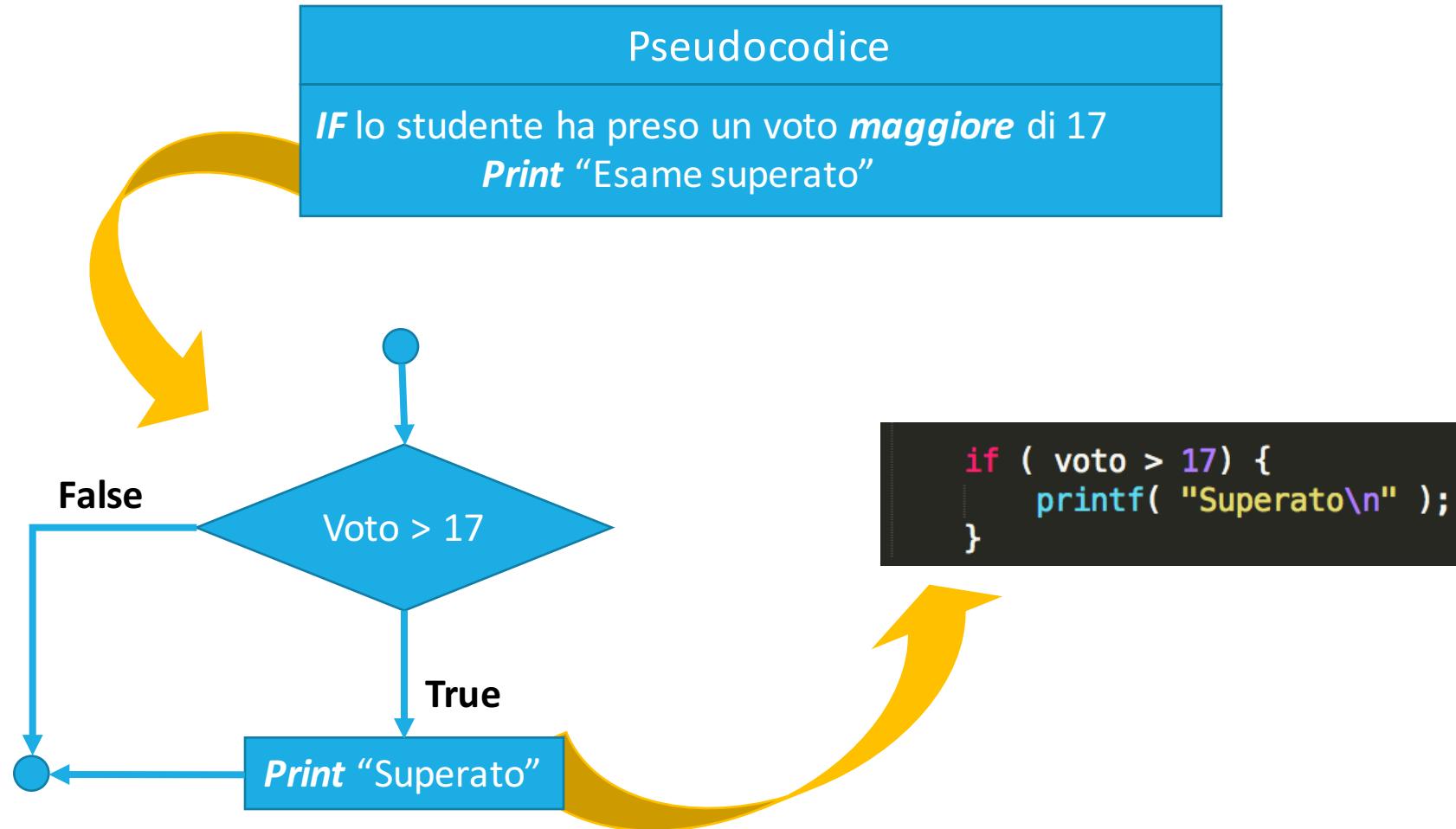
- Lo statement **if** *seleziona o ignora un'azione sulla base di una condizione*;

```
if (expression) {  
    statement  
}
```



Strutture di condizione: IF

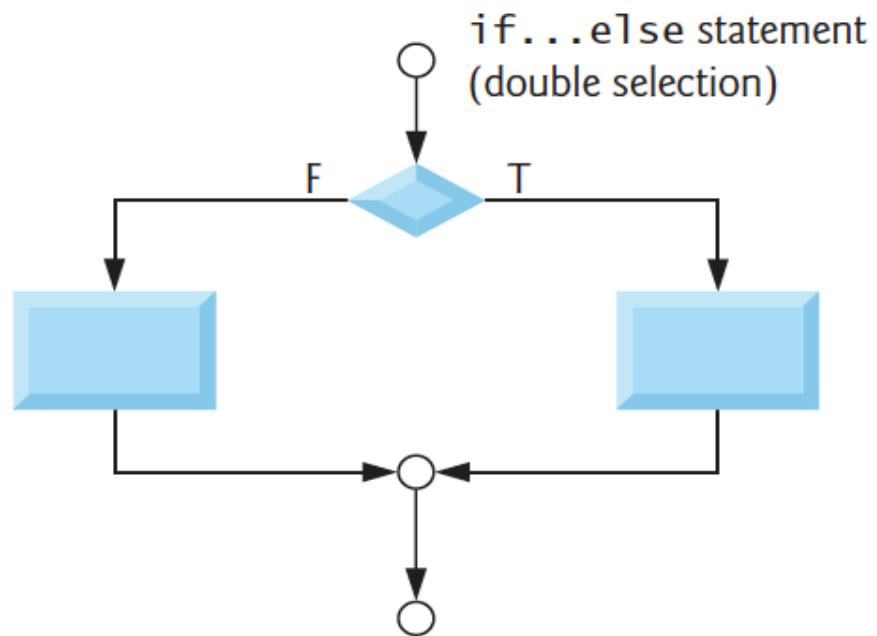
- Lo statement **if** *seleziona o ignora un'azione sulla base di una condizione*;



Strutture di condizione: IF...ELSE

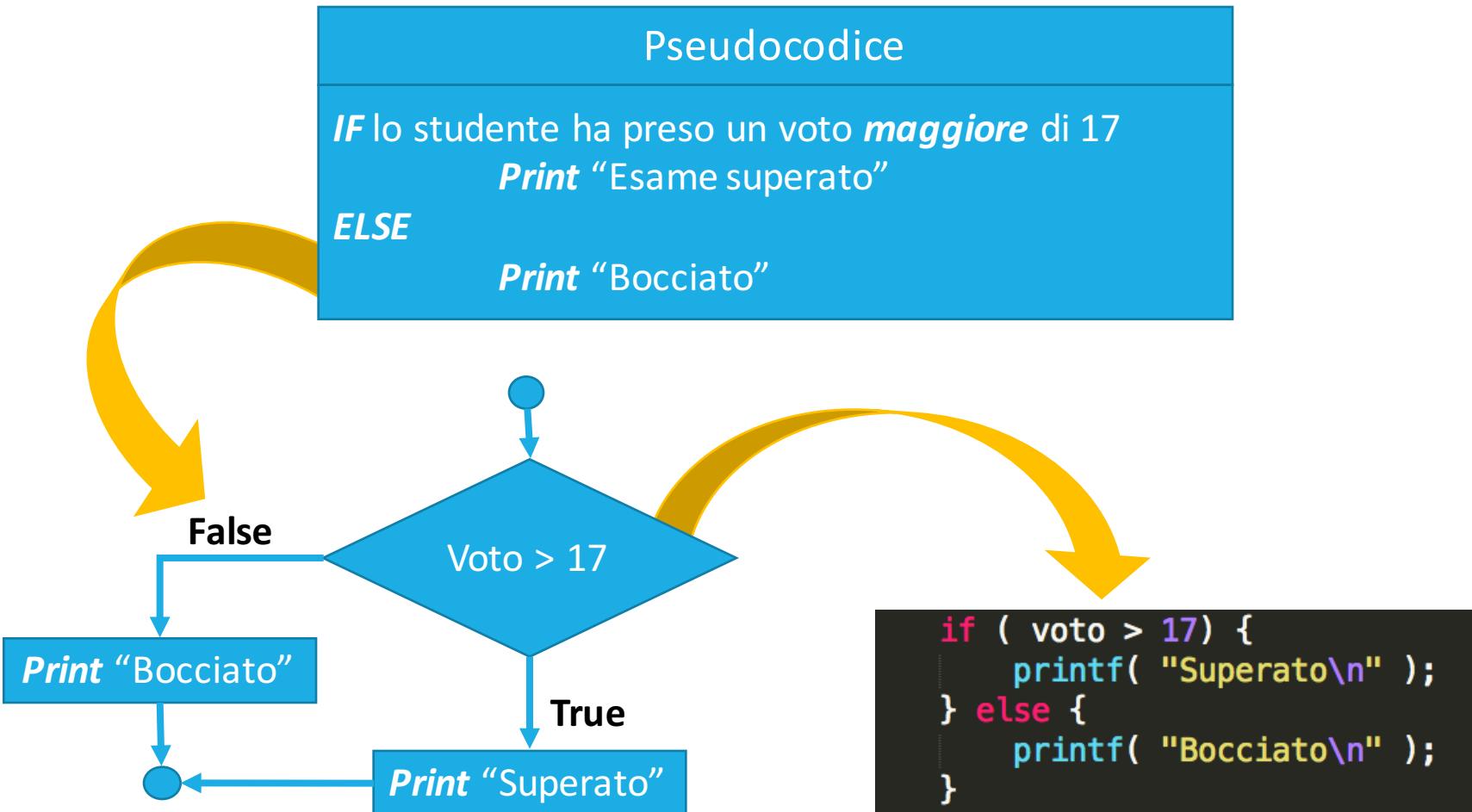
- Lo statement *if...else* **seleziona un'azione tra due alternative sulla base di una condizione**;

```
if (expression) {  
    statement_1  
} else {  
    statement_2  
}
```



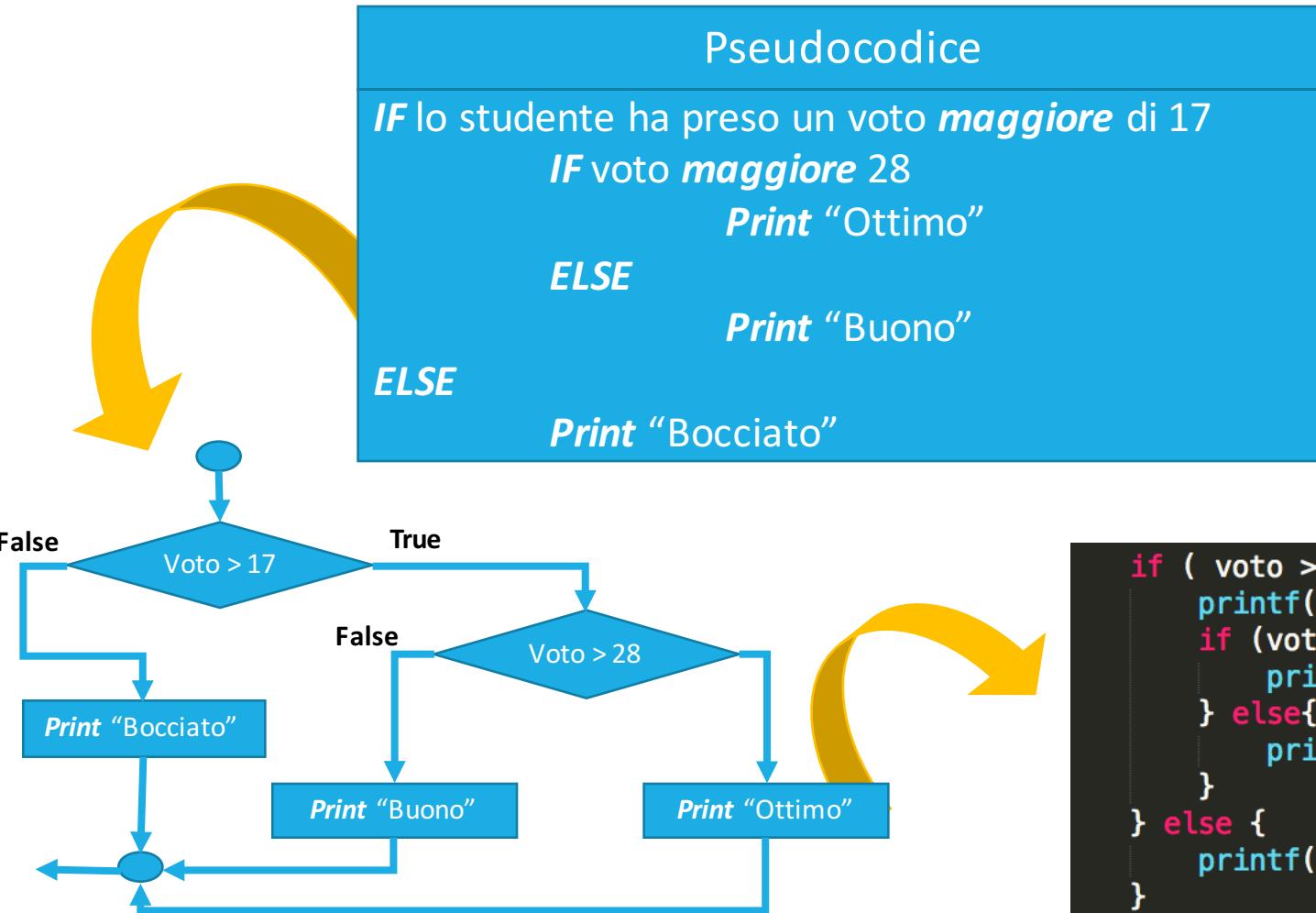
Strutture di condizione: IF...ELSE

- Lo statement *if...else* **seleziona un'azione tra due alternative sulla base di una condizione**;



Strutture di condizione: NESTED IF...ELSE

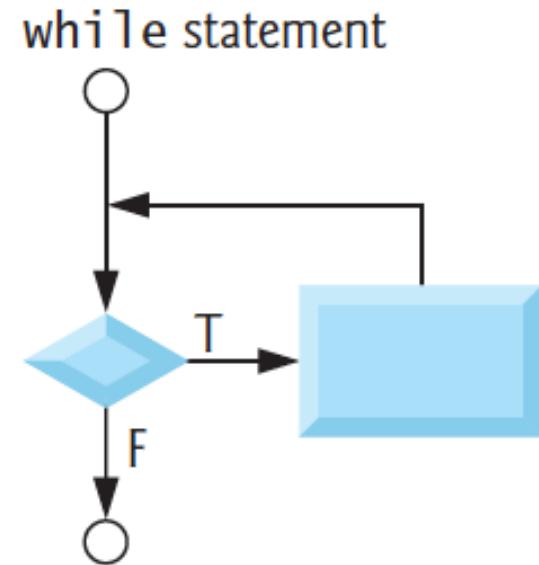
- Lo statement nested if...else seleziona un'azione tra più alternative sulla base di una condizione;



Strutture di iterazione: WHILE

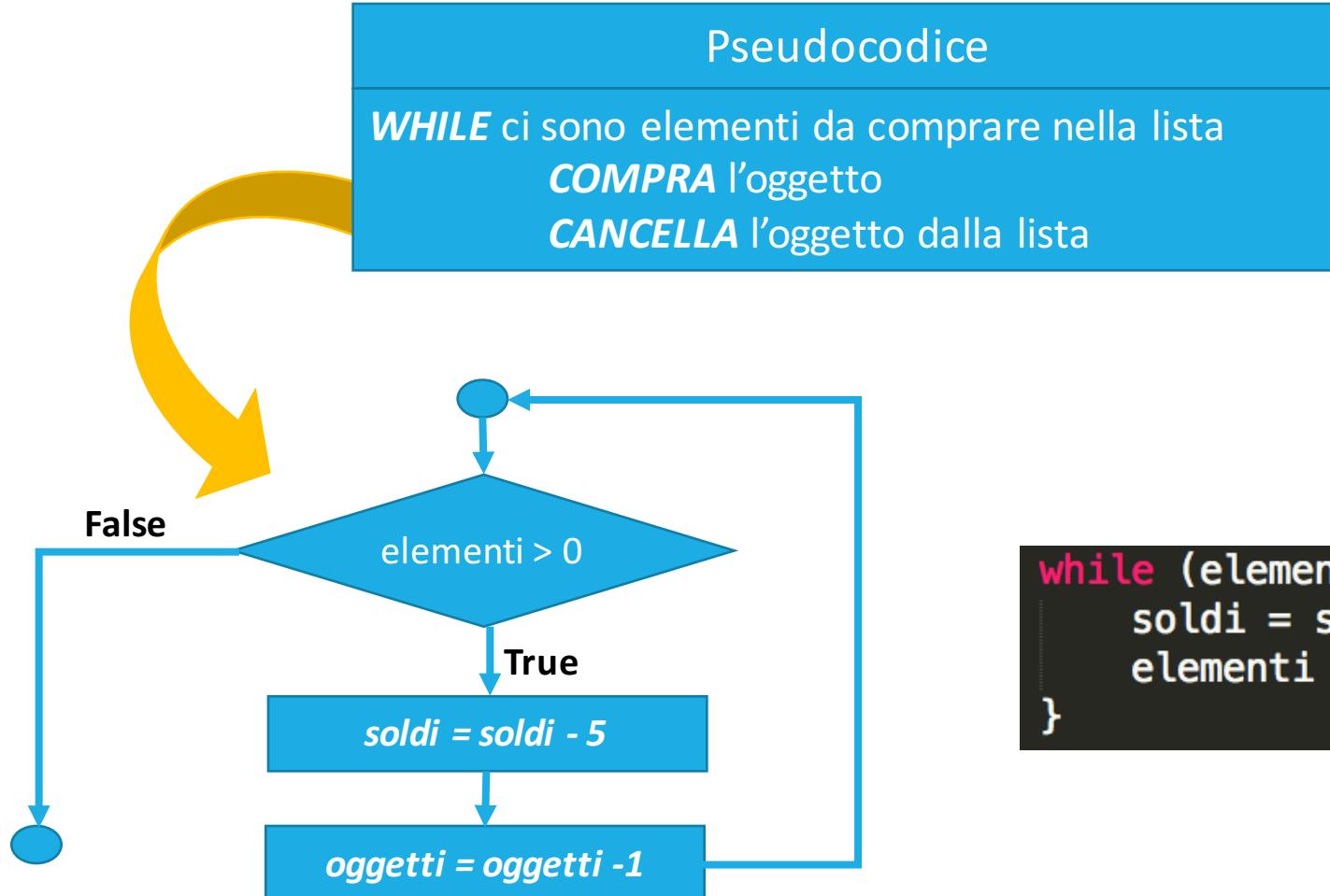
- Lo statement *while* permette di ***eseguire un'azione fin tanto che una certa condizione viene verificata;***

```
while(expression) {  
    statement  
}
```



Strutture di iterazione: WHILE

- Lo statement *while* permette di ***eseguire un'azione fin tanto che una certa condizione viene verificata;***



Sintassi alternative

- Fino ad ora abbiamo visto quale sia la sintassi dei vari costrutti di controllo;
- Esistono delle sintassi alternative che permettono di **velocizzare la stesura del codice**;
- Queste sintassi alternative potrebbero **confondere il programmatore** (non il programma) ed ottenere un output inatteso;
- E' necessario fare molta attenzione;

Omissione delle parentesi graffe

- Tra le varie sintassi, quella che potrebbe essere ritrovata più frequentemente è l'**omissione delle parentesi**;

```
if ( a > b) {  
    printf("A è maggiore di b \n");  
}
```



```
if ( a > b)  
    printf("A è maggiore di b \n");
```

Omissione delle parentesi graffe

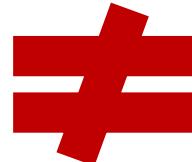
- Tra le varie sintassi, quella che potrebbe essere ritrovata più frequentemente è l'**omissione delle parentesi**;

```
if ( a > b ) {  
    printf("A è maggiore di b \n");  
}
```



```
if ( a > b)  
    printf("A è maggiore di b \n");
```

```
while ( a < 10 ) {  
    a++;  
    printf("a = %d\n", a);  
}
```



```
while ( a < 10 )  
    a++;  
    printf("a = %d\n", a);
```

Un errore comune

- Uno degli errori più comuni per i programmati alle prime armi è ***confondere l'operatore di assegnazione con l'operatore di confronto;***

| OPERATORE ASSEGNAZIONE | OPERATORE DI CONFRONTO |
|------------------------|------------------------|
| = | == |

Un errore comune

- Uno degli errori più comuni per i programmati alle prime armi è ***confondere l'operatore di assegnazione con l'operatore di confronto;***

| OPERATORE ASSEGNAZIONE | OPERATORE DI CONFRONTO |
|------------------------|------------------------|
| = | == |

if($a = b$)

Secondo voi è corretta?

Un errore comune

- Uno degli errori più comuni per i programmati alle prime armi è ***confondere l'operatore di assegnazione con l'operatore di confronto***;

| OPERATORE ASSEGNAZIONE | OPERATORE DI CONFRONTO |
|------------------------|------------------------|
| = | == |

if(a = b)

è sintatticamente corretta

Un errore comune

- Uno degli errori più comuni per i programmati alle prime armi è **confondere l'operatore di assegnazione con l'operatore di confronto**;

| OPERATORE ASSEGNAZIONE | OPERATORE DI CONFRONTO |
|------------------------|------------------------|
| = | == |

$if(a = b)$
è sintatticamente
corretta

b viene assegnato ad a
l'espressione $a=b$ prende il valore di b

Un errore comune

- Uno degli errori più comuni per i programmati alle prime armi è **confondere l'operatore di assegnazione con l'operatore di confronto**;

| OPERATORE ASSEGNAZIONE | OPERATORE DI CONFRONTO |
|------------------------|------------------------|
| = | == |

$if(a = b)$
è sintatticamente
corretta

$if (a=b) \rightarrow if (b),$
True se $b!=0$, False se $b==0$

ESERCIZIO GUIDATA

Programmazione strutturata

FONDAMENTI DI PROGRAMMAZIONE

Esercizio guidato

- *Definire e implementare* un programma C che chieda all'utente di inserire due numeri interi, x e y , e stampi in output il maggiore tra i due.

Esercizio guidato

- *Definire e implementare* un programma C che chieda all’utente di inserire due numeri interi, x e y , e stampi in output il maggiore tra i due.
- Fasi di sviluppo:
 - **Fase 1:** analisi del problema e definizione dei dati di ingresso e di uscita;
 - **Fase 2:** descrizione dell’algoritmo con lo pseudocodice;
 - **Fase 3:** descrizione tramite diagramma di flusso;
 - **Fase 4:** test logica algoritmo;
 - **Fase 5:** traduzione del diagramma di flusso in codice C;

Esercizio guidato

- **(Fase 1)** Analisi del problema e definizione dei dati di ingresso e di uscita:

- **INPUT:**

- **x**: numero intero;
- **y**: numero intero;

- **OUTPUT:**

- Stampa il messaggio “***Il maggiore è x***”, se $x < y$;
- Stampa il messaggio “***Il maggiore è y***”, se $x < y$;
- Stampa il messaggio “***I valori sono uguali***”;

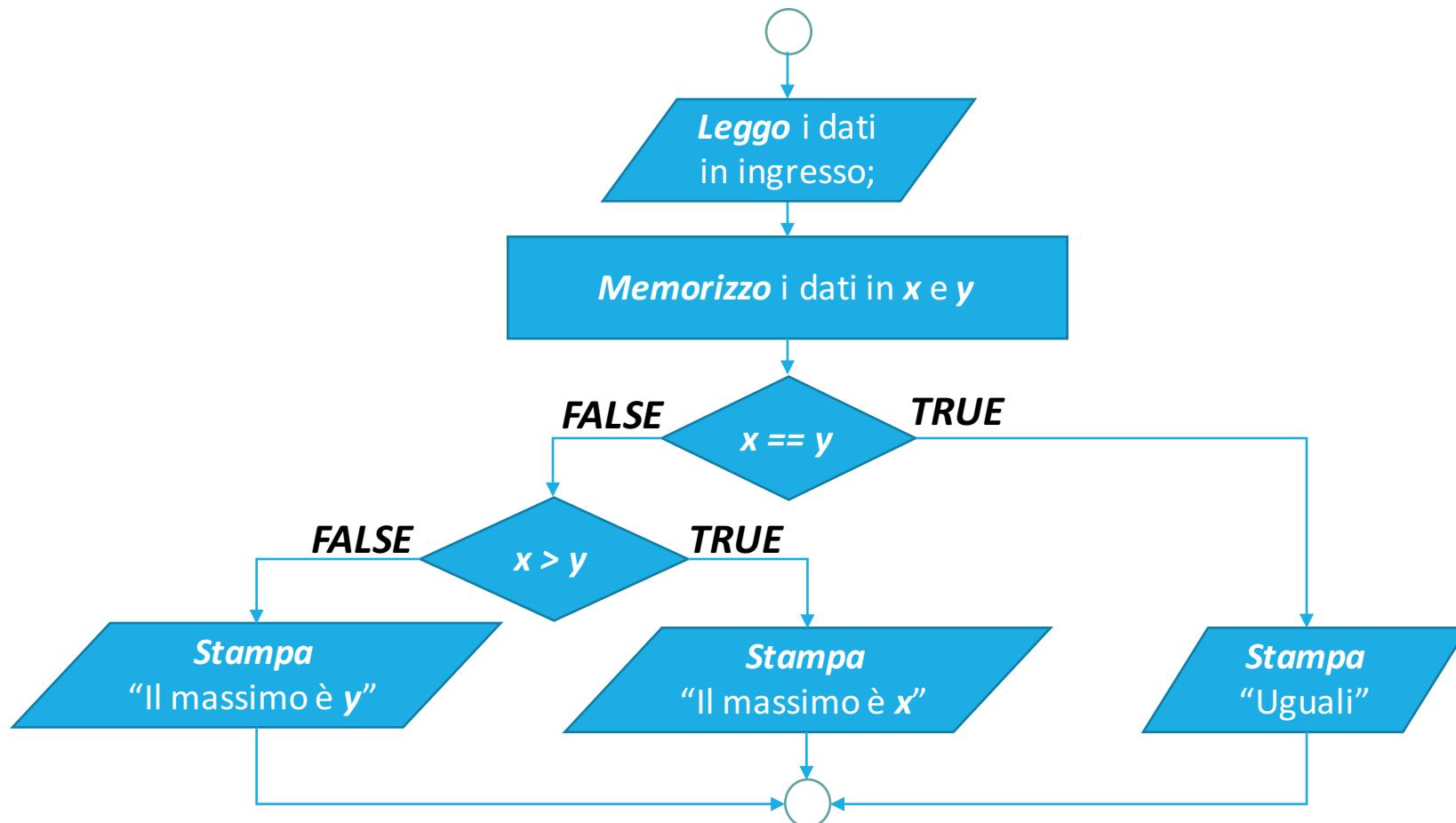
Esercizio guidato

- **(Fase 2)** Descrizione dell'algoritmo con lo pseudocodice:

1. **Leggo** i dati in ingresso;
2. **Memorizzo** i dati nelle variabili x e y ;
3. **Confronto** x e y ;
 - a. **Se** $x > y$, **stampa** “Il maggiore è x ”;
 - b. **Se** $x < y$, **stampa** “Il maggiore è y ”;
 - c. **Altrimenti, stampa** “I valori sono uguali”;
4. **Fine**

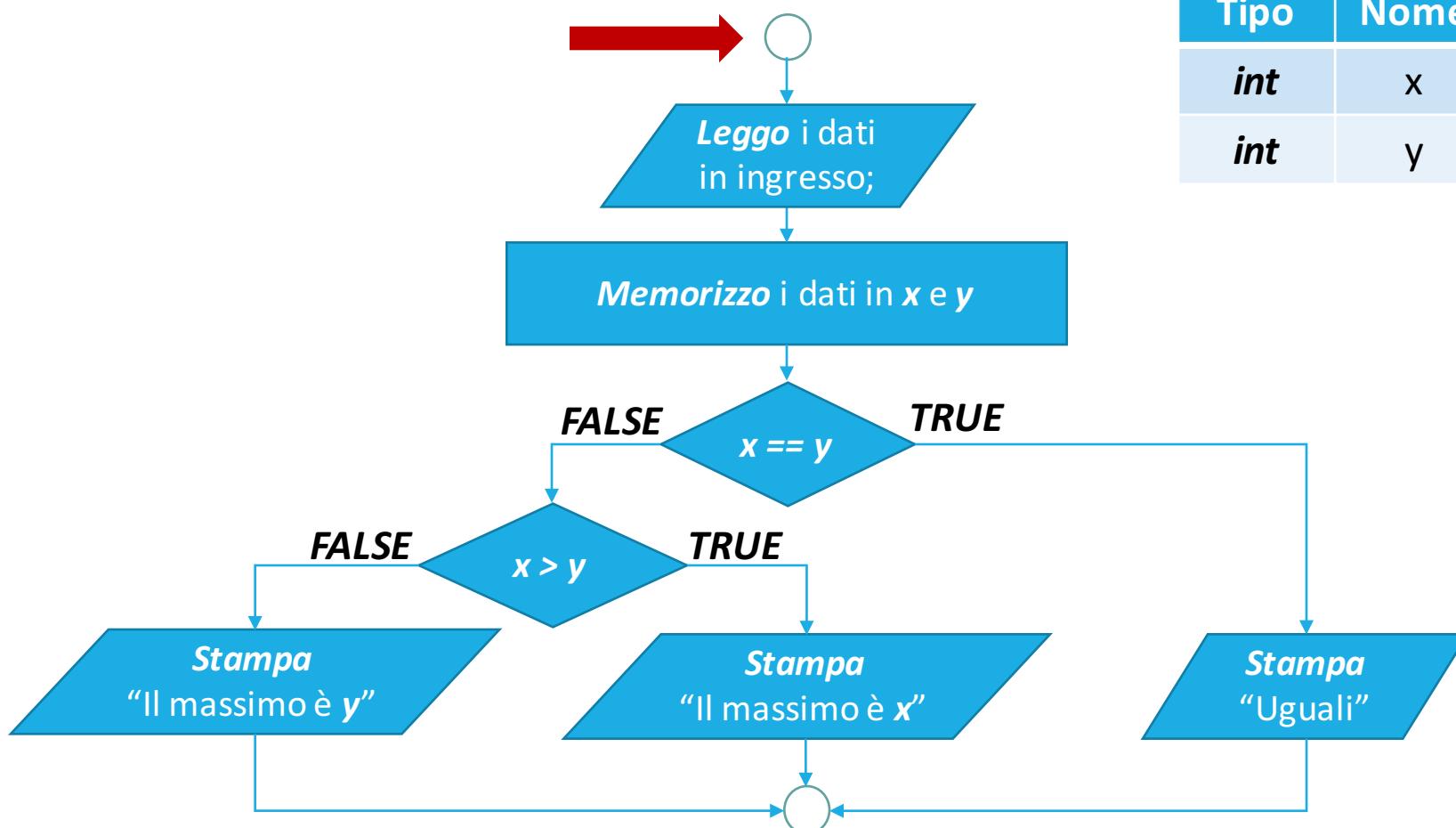
Esercizio guidato

- **(Fase 3)** Descrizione tramite diagramma di flusso:



Esercizio guidato

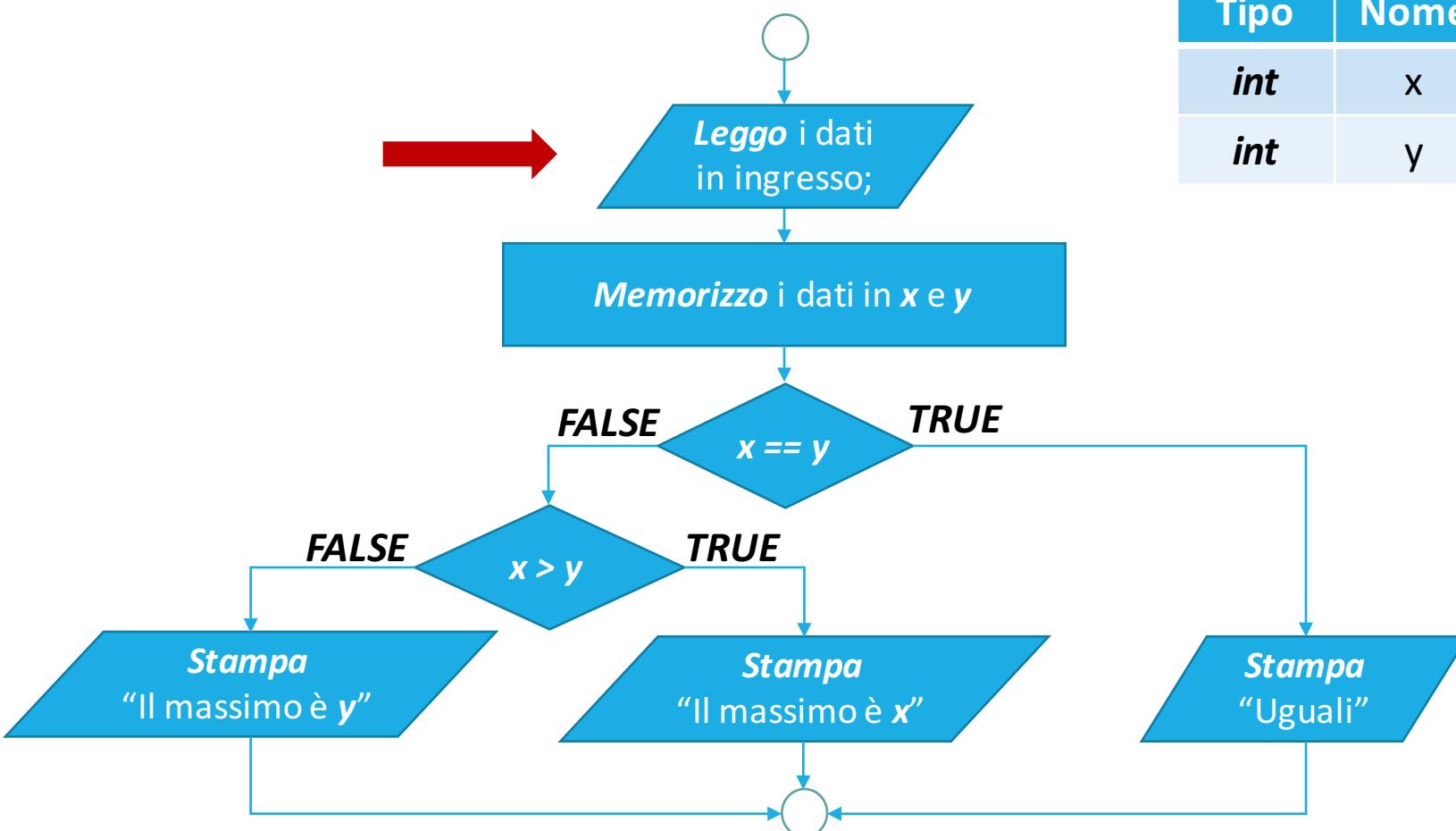
- **(Fase 4)** Test logica dell'algoritmo:



| Tipo | Nome | Valore |
|------|------|--------|
| int | x | - |
| int | y | - |

Esercizio guidato

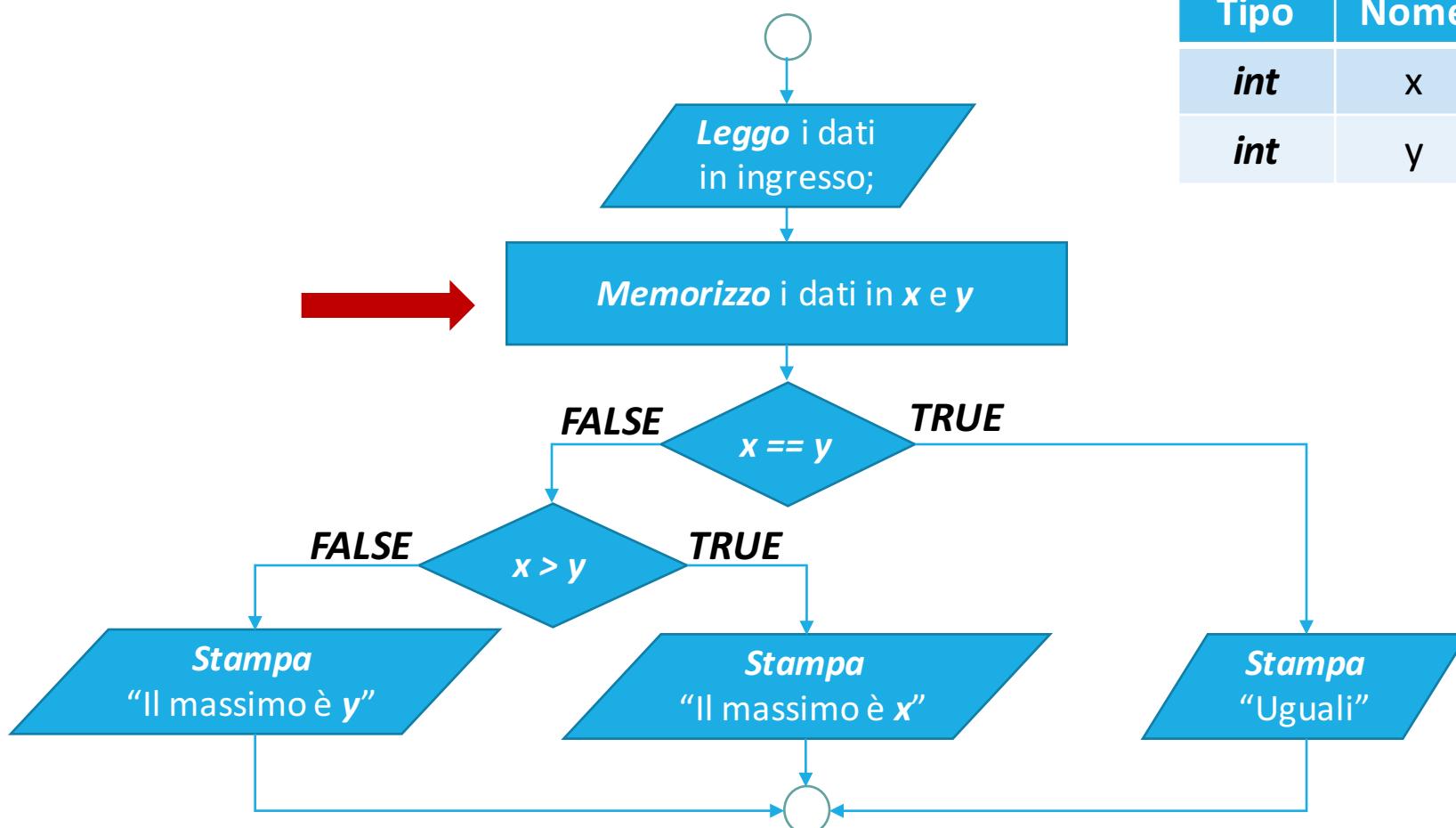
- **(Fase 4)** Test logica dell'algoritmo:



| Tipo | Nome | Valore |
|------|------|--------|
| int | x | - |
| int | y | - |

Esercizio guidato

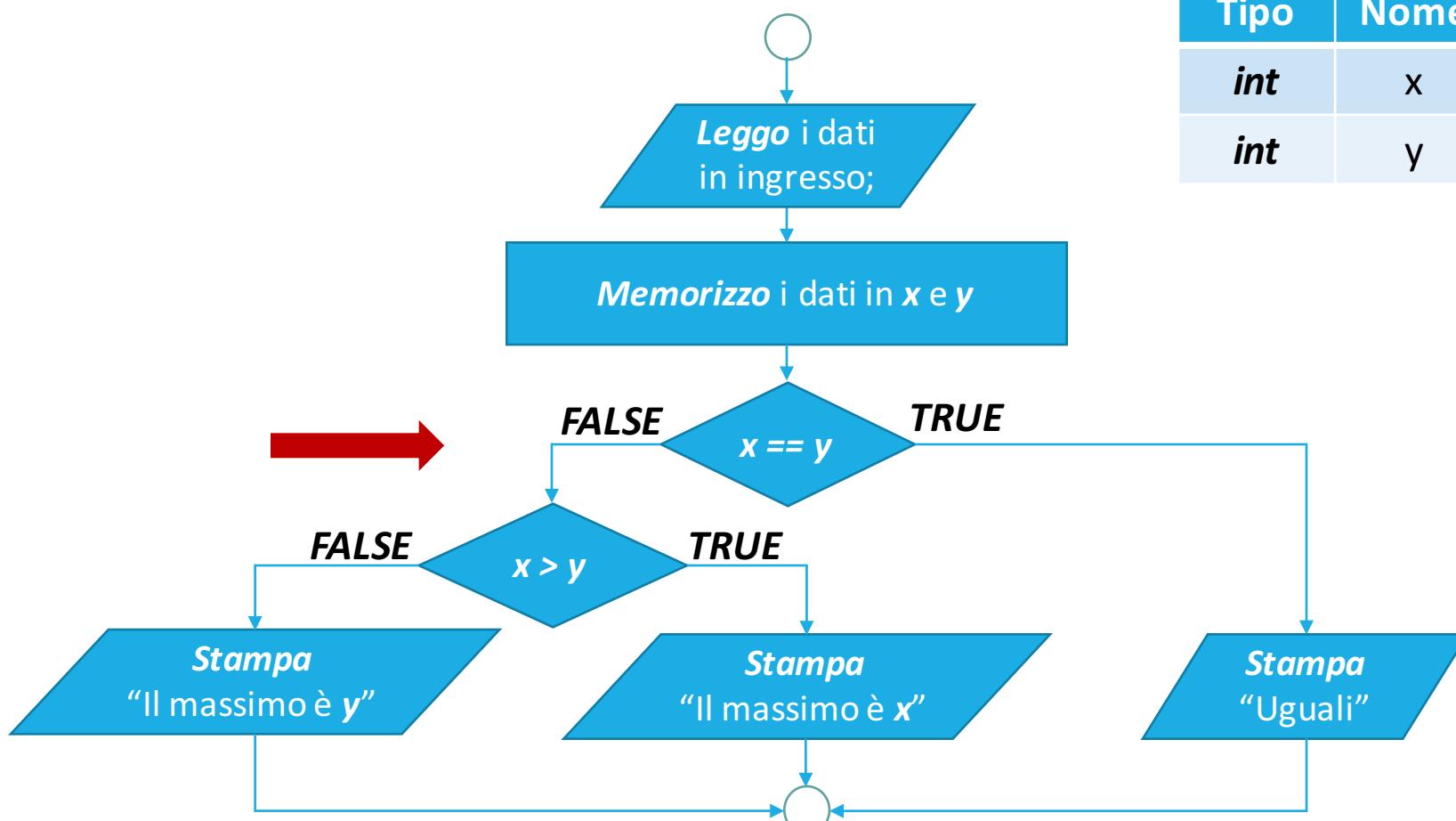
- **(Fase 4)** Test logica dell'algoritmo:



| Tipo | Nome | Valore |
|------------|------|--------|
| <i>int</i> | x | 10 |
| <i>int</i> | y | 10 |

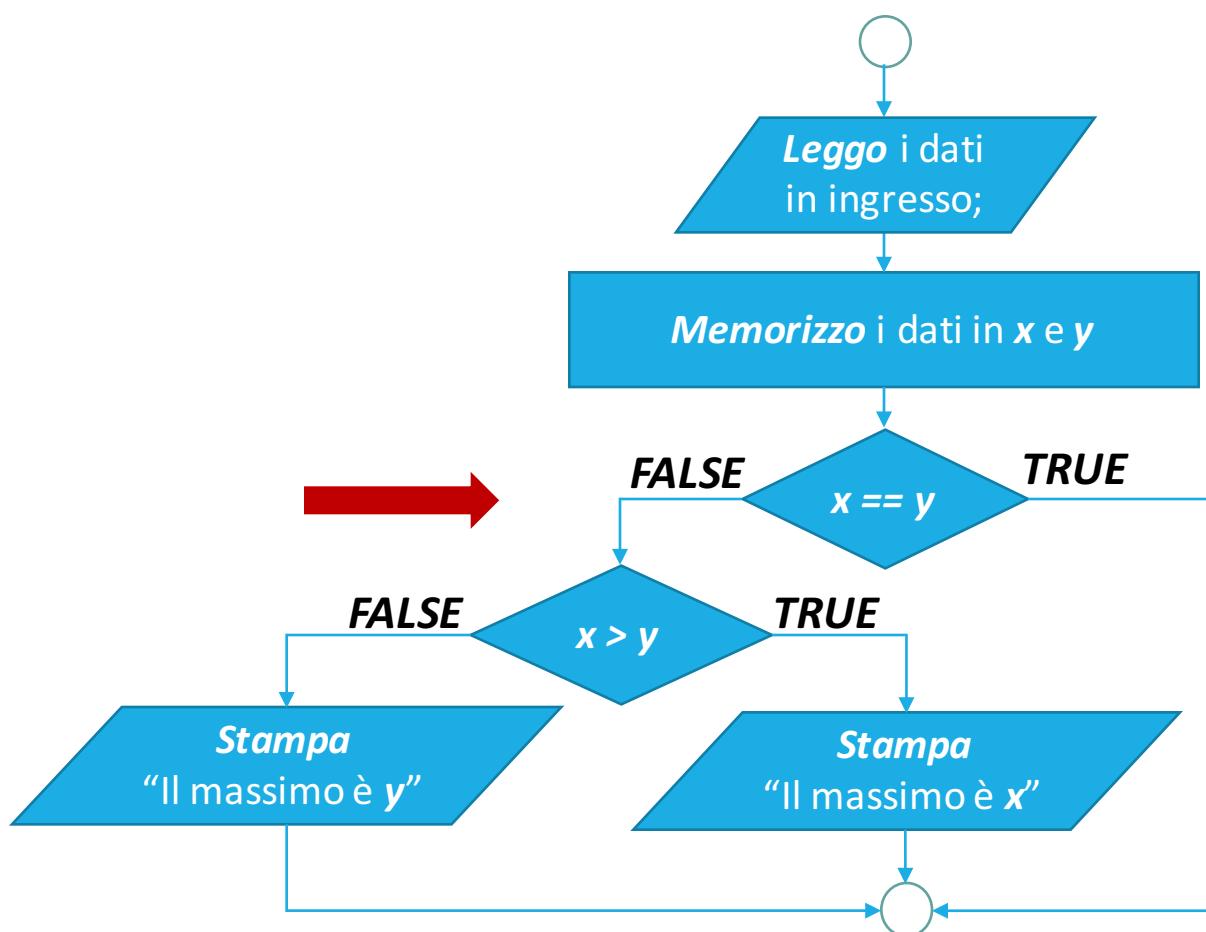
Esercizio guidato

- **(Fase 4)** Test logica dell'algoritmo:



Esercizio guidato

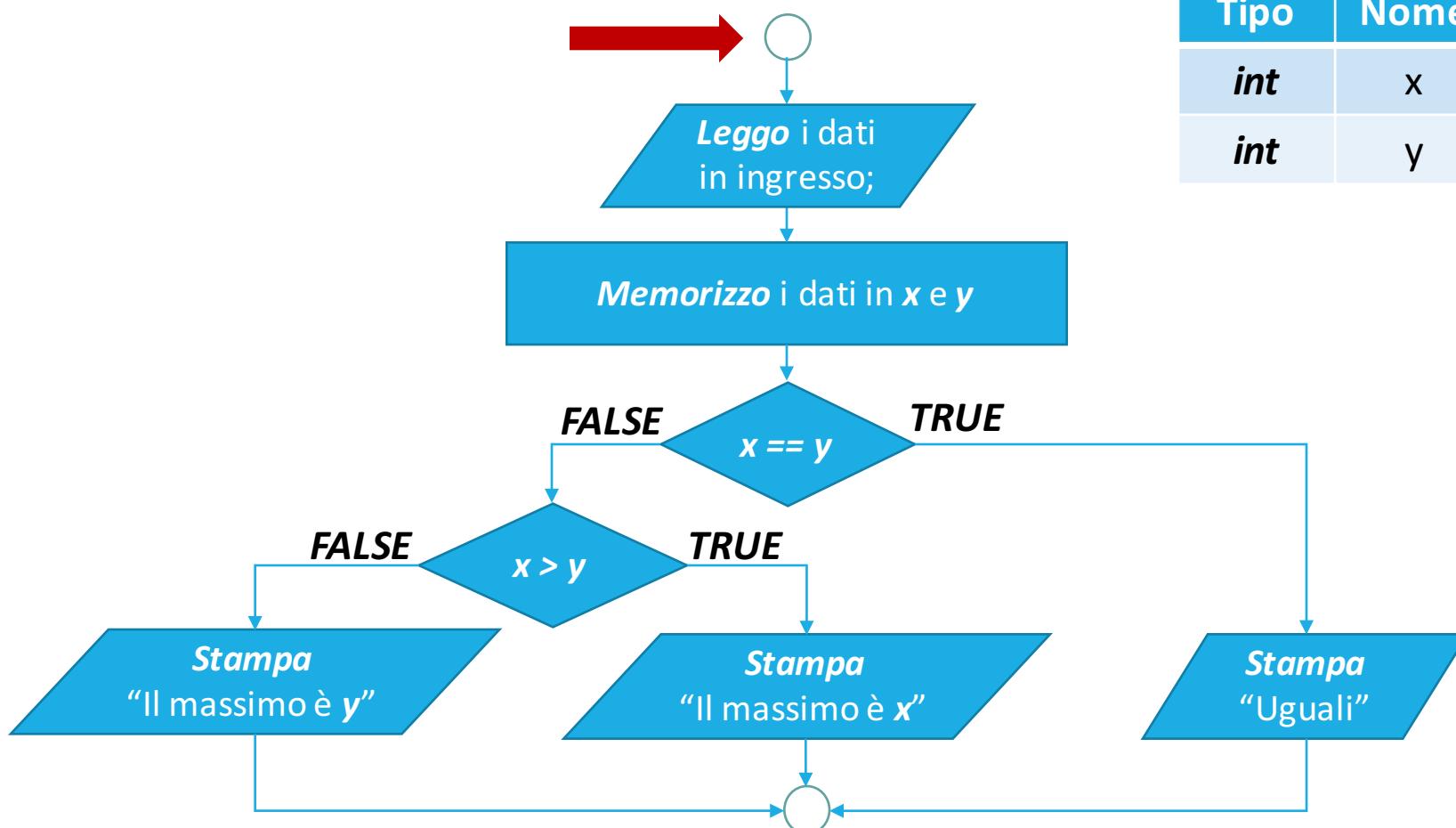
- **(Fase 4)** Test logica dell'algoritmo:



| Tipo | Nome | Valore |
|------|------|--------|
| int | x | 10 |
| int | y | 10 |

Esercizio guidato

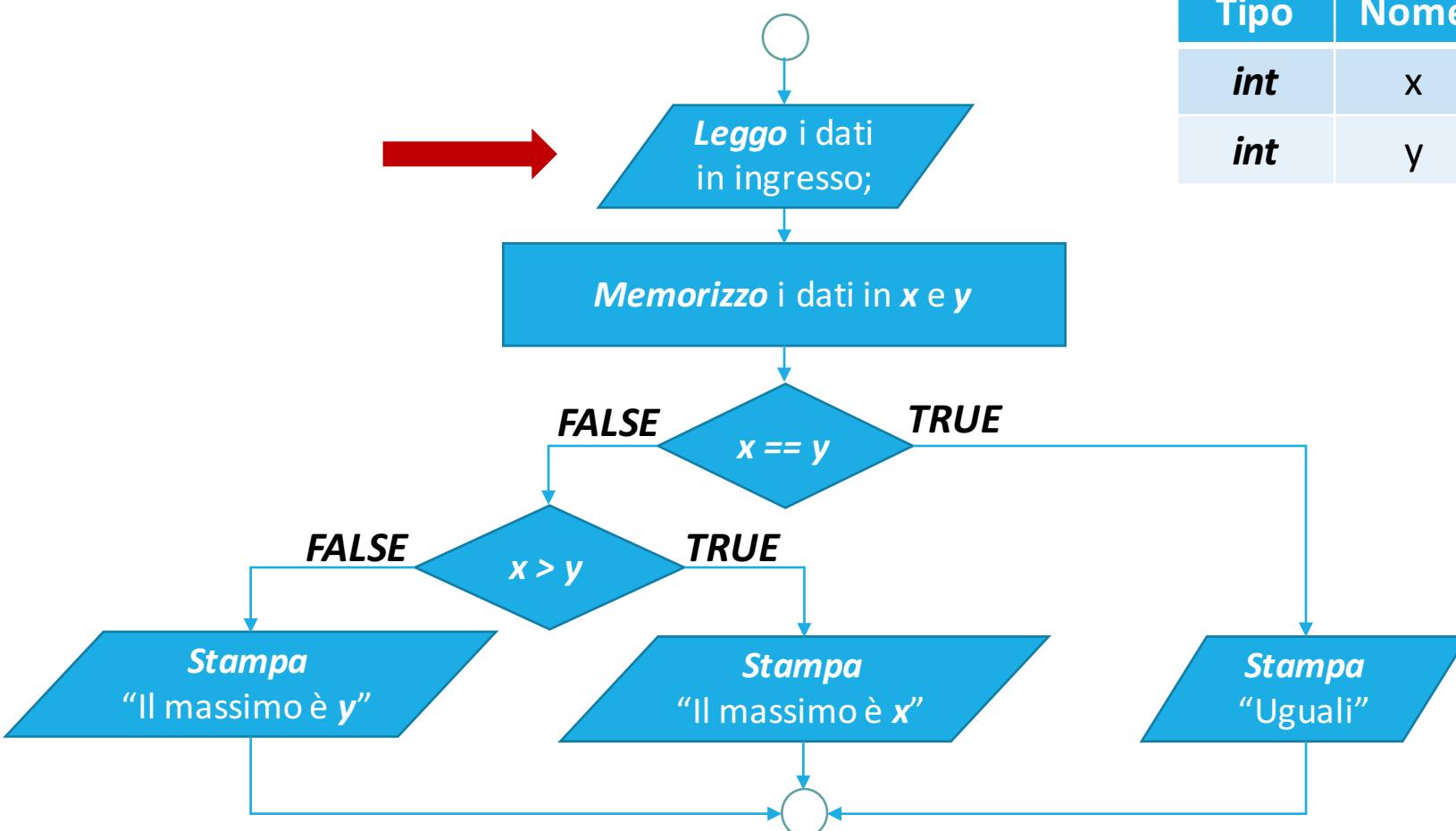
- **(Fase 4)** Test logica dell'algoritmo:



| Tipo | Nome | Valore |
|------|------|--------|
| int | x | - |
| int | y | - |

Esercizio guidato

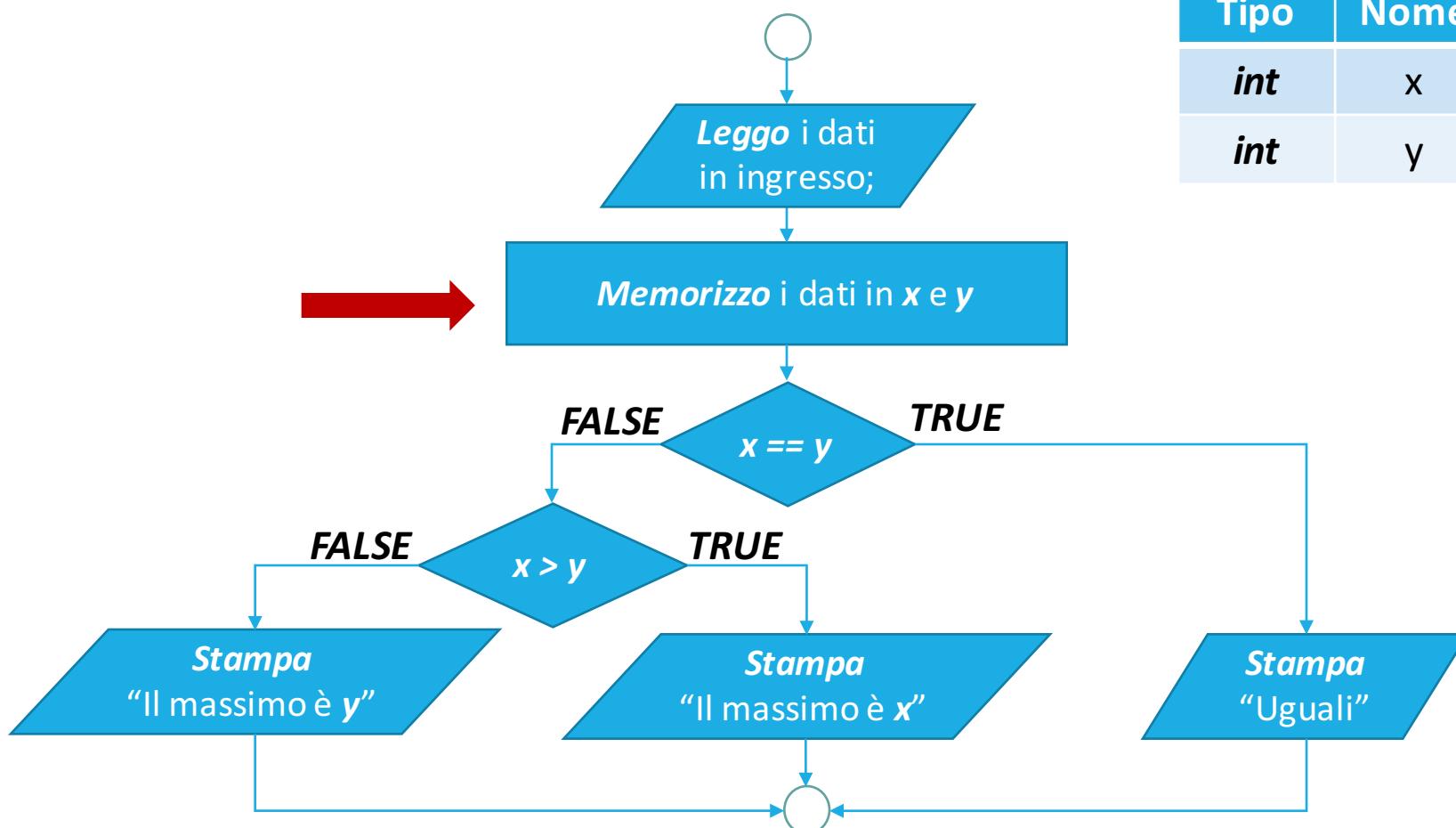
- **(Fase 4)** Test logica dell'algoritmo:



| Tipo | Nome | Valore |
|------|------|--------|
| int | x | - |
| int | y | - |

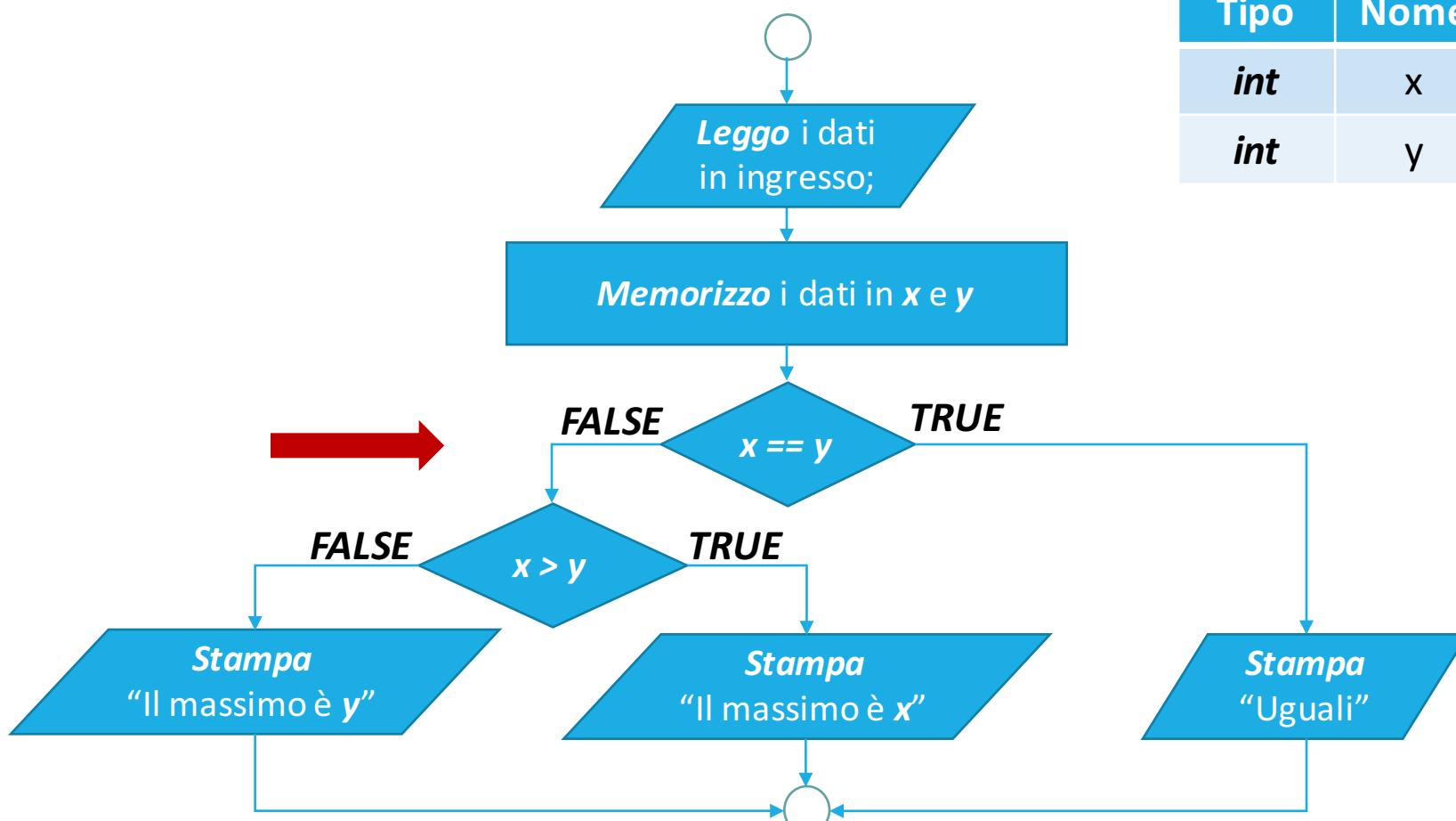
Esercizio guidato

- **(Fase 4)** Test logica dell'algoritmo:



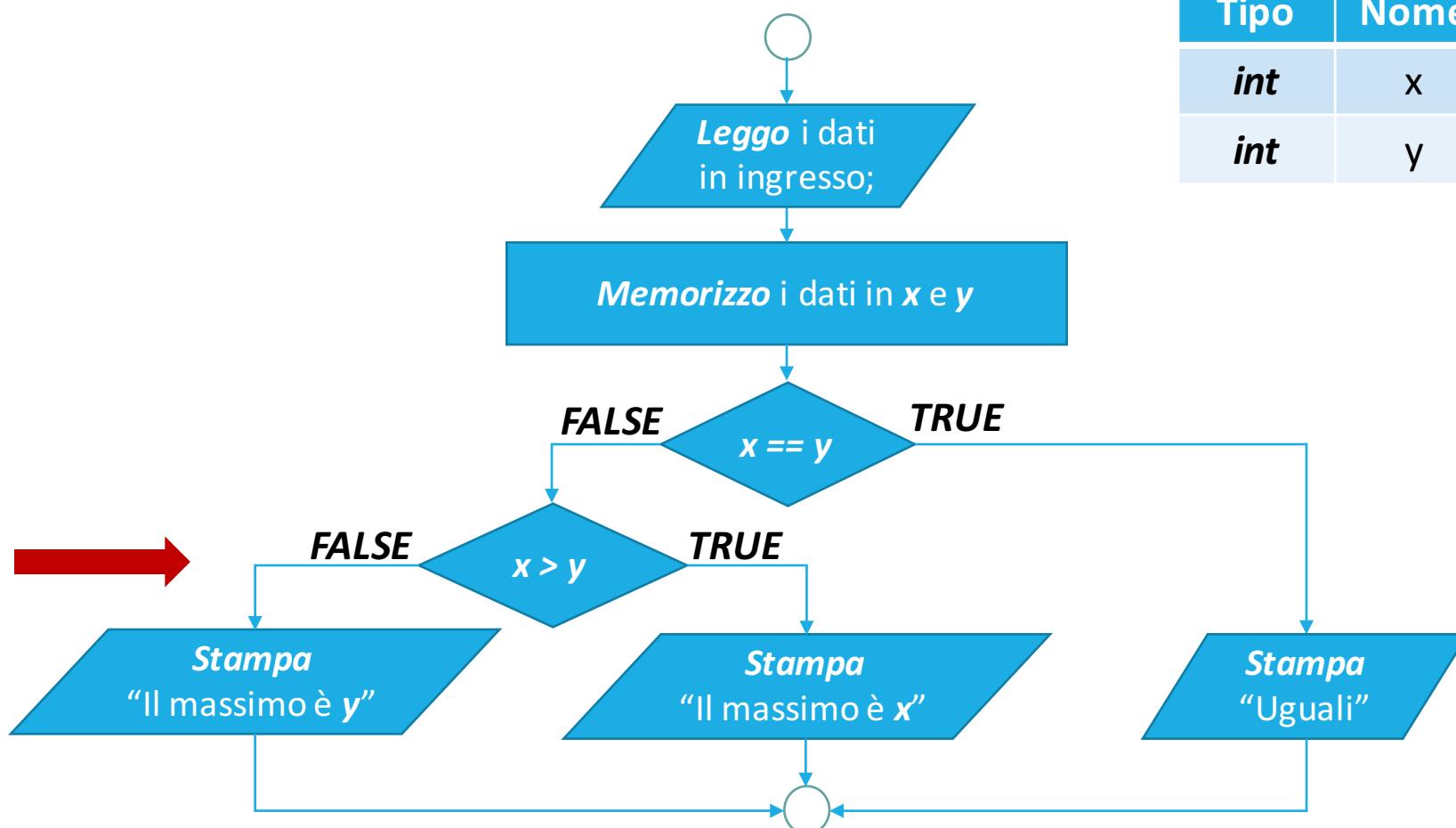
Esercizio guidato

- **(Fase 4)** Test logica dell'algoritmo:



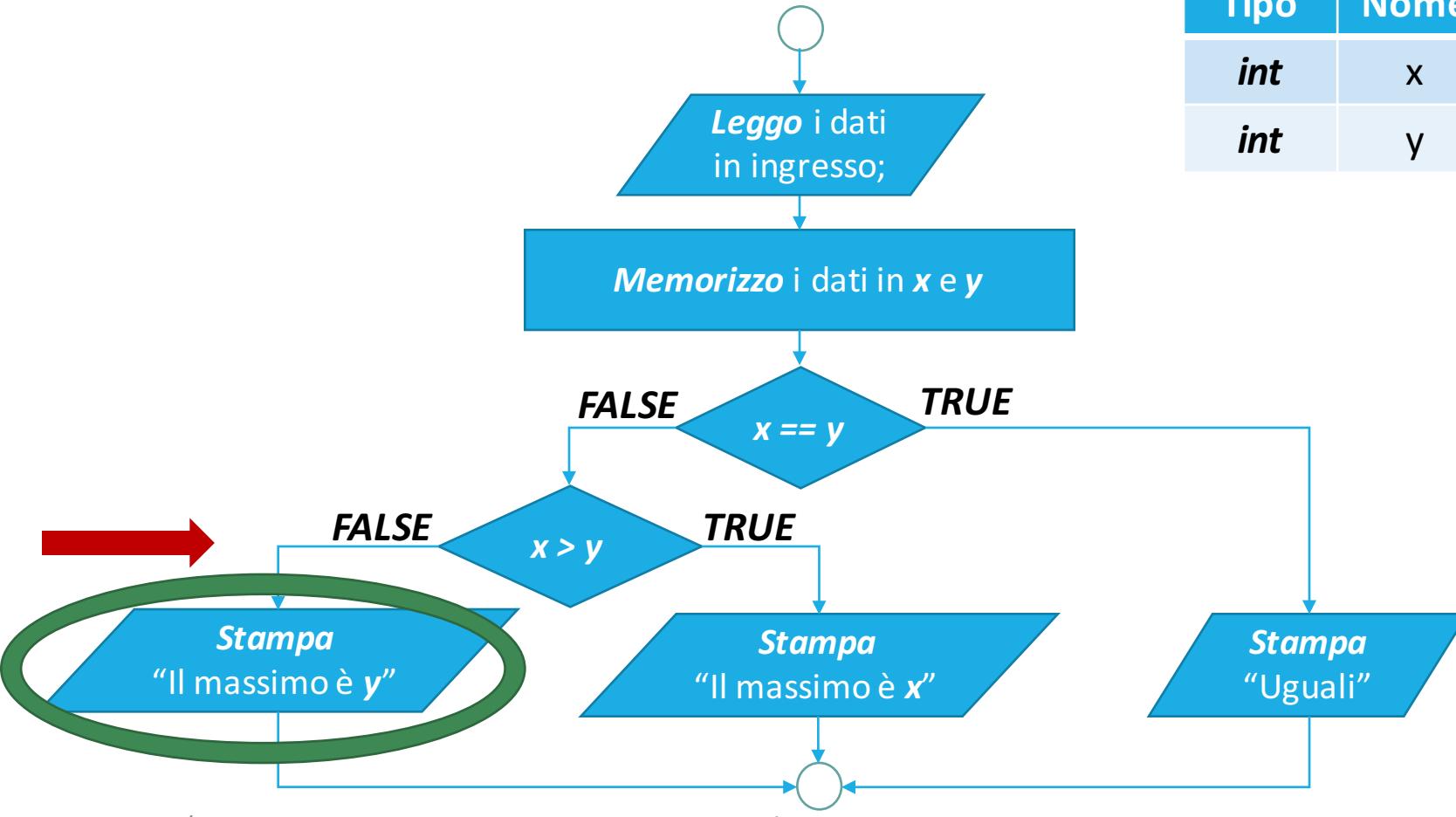
Esercizio guidato

- **(Fase 4)** Test logica dell'algoritmo:



Esercizio guidato

- **(Fase 4)** Test logica dell'algoritmo:



Esercizio guidato

- **(Fase 5)** Traduzione del diagramma di flusso in codice C :

```
5  #include<stdio.h>
6
7  int main(void){
8      int x,y;
9
10     printf("Inserire il primo numero: ");
11     scanf("%d", &x);
12
13     printf("Inserire il secondo numero: ");
14     scanf("%d", &y);
15
16     if( x==y ){
17         printf("I numeri sono uguali!\n");
18     }else{
19         if(x > y){
20             printf("Il maggiore è %d\n", x);
21         } else {
22             printf("Il maggiore è %d\n", y);
23         }
24     }
25     return 0;
26 }
```

ESERCIZI

Programmazione strutturata in C

FONDAMENTI DI PROGRAMMAZIONE

Esercizio 1



10 min

Dato il seguente programma C:

```
#include<stdio.h>

int main(void){
    int i = 7;
    int j = 7;

    if( i == 1 )
        if( j == 2)
            printf("%d\n", i = i + j);
    else
        printf("%d\n", i = i - j);

    printf("%d\n", i);

    return 0;
}
```



1. *Dire che cosa viene stampato in output;*
2. *Fornire una spiegazione.*

Esercizio 2



10 min

Dato il seguente programma C:



```
#include<stdio.h>

int main(void){
    unsigned int i = 10;
    int count = 0;

    while ( i < 0 ) {
        count = count + 1;
        printf("%d\n", count);
        i = i - 1;
    }

    return 0;
}
```

1. **Identificare gli errori** (c'è almeno un errore);
2. **Compilare ed eseguire** il codice corretto;

Esercizio 3



20 min



Un numero è pari se è divisibile per 2.

Definire e implementare un programma C che chieda all'utente di inserire un intero n e che conti quanti numeri pari vi sono nell'intervallo che va da 1 a n.

Esempio. Se $n=10$ allora il programma dovrà restituire 5

Esercizio 4



15 min

Dire cosa fa il seguente programma C:



```
#include<stdio.h>

int main(void){
    unsigned int i=0, valore;
    unsigned int risultato = 1;

    printf("Inserire un valore: \n");
    scanf("%d", &valore);

    while(i<=valore){
        if (i==0)
            risultato = risultato * 1;
        else
            risultato = risultato * i;
        i++;
    }

    printf("Risultato: %d\n", risultato);
    return 0;
}
```

HOMEWORKS

FONDAMENTI DI PROGRAMMAZIONE

HOMEWORKS



Definire e implementare un programma C che chieda all'utente di inserire un intero n e che calcoli la somma degli interi da 1 a n.

Esempio. Se $n=5$ allora il programma dovrà restituire $1+2+3+4+5 = 15$

HOMEWORKS



Un numero naturale è primo se è maggiore di 1 e se è divisibile per 1 e per sé stesso.

Definire e implementare un programma C che chieda all'utente di inserire un numero naturale n e dica se il numero è primo o no.

HOMEWORKS



Definire e implementare un programma C che legga un valore intero n e:

- *sommi gli interi da n a $2n$ se n non è negativo;*
- *sommi gli interi da $2n$ a n se n è negativo;*

*Scrivere l'algoritmo e il relativo codice in due versioni:
una che utilizzi esclusivamente cicli for, l'altra che
utilizzi esclusivamente cicli while.*

HOMEWORKS



Un palindromo è un numero o una frase di testo che, letta al contrario, rimane invariata. Ad esempio, ciascuno dei seguenti numeri interi a cinque cifre è un palindromo: 12321, 55555, 45554 e 11611.

Definire e implementare un programma C che verifichi se un numero intero a 5 cifre è palindromo.

Suggerimento: Utilizzare la divisione e l'operatore modulo per separare il numero nelle singole cifre.

HOMEWORKS



Definire e implementare un programma C per determinare il primo ed il secondo classificato di una gara dei 100 metri a cui partecipano quattro corridori.

Il programma dovrà chiedere all'utente:

- l'inserimento dell'identificativo (un numero intero tra 1 e 255) di ogni partecipante;*
- il tempo impiegato dallo stesso espresso in secondi (per esempio, 9.878).*

Al termine dell'inserimento, il programma dovrà stampare a video l'identificativo ed il tempo del vincitore e del secondo classificato.

Si assuma che non ci siano due corridori che abbiano impiegato esattamente lo stesso tempo e che nessun corridore impieghi più di 20 secondi.