INGEGNERIA PER LA SOSTENIBILITA' INDUSTRIALE - UnivPM

Fondamenti di Informatica

Prof. Ing. Loris Penserini, PhD

elpense@gmail.com

https://orcid.org/0009-0008-6157-0396

Materiale:

https://github.com/penserini/Lezioni_UnivPM.git

Operatori e Variabili

Variabile

Rappresenta un contenitore nel quale parcheggiare dei dati che dovranno essere utilizzati nelle operazioni dell'Algoritmo. Per cui, una variabile è individuata da un identificatore o nome della variabile, e contiene dati che variano durante l'esecuzione dell'Algoritmo.

Su una variabile (var) si possono eseguire queste operazioni:

DICHIARAZIONE: si dichiara il nome «var» e il tipo di dato che contiene, per es. «intero»

ASSEGNAZIONE: si assegna un valore preciso « var = 12 »

UTILIZZO: si utilizza la variabile per risolvere espressioni o per stampare in output un risultato: « var2 = var + 5 » oppure « stampa var »

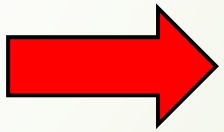
Una variabile a run-time rappresenta una cella di memoria, per cui nella fase di dichiarazione l'esecutore apprende quanto spazio allocare alle variabili a seconda del tipo di dato che contengono.

Campo d'azione

In molti linguaggi strutturati/oggetti, le parentesi graffe individuano i «blocchi di istruzioni», per cui una variabile definita all'interno di un blocco ha validità (o visibilità) solo in quel blocco o nei sotto-blocchi.

Lo spazio di validità (o visibilità) di una variabile all'interno di un programma è anche detto «scope» o «campo d'azione».

```
{
    int a = 1;
    {
        a = a + 5;
        int b = a;
        System.out.println(''b_1 = '' + b);
    }
    a = a  + 1;
    int b = a;
    System.out.println (''b_2 = '' + b);
}
```



b_1 = 6

b_2 = 7

Campo d'azione in Python

In Python, le variabili definite all'interno di un blocco **if**, **for**, **while**, o **else** non creano un nuovo scope.

Questo significa che le variabili dichiarate dentro questi blocchi sono visibili anche al di fuori di essi, purché appartengano alla stessa funzione (o al livello globale).

Questa «troppa» libertà è anche causa di facili errori:

ExScope.py > ...

```
1   a = 4; b = 7
2   a = a + b
3   if a > b:
4       c = 1
5       a = a + c
6       var = print("DENTRO IF: a + c = ", a)
7
8   a = c + 3
9   print("FUORI IF: a + c = ", a)
```

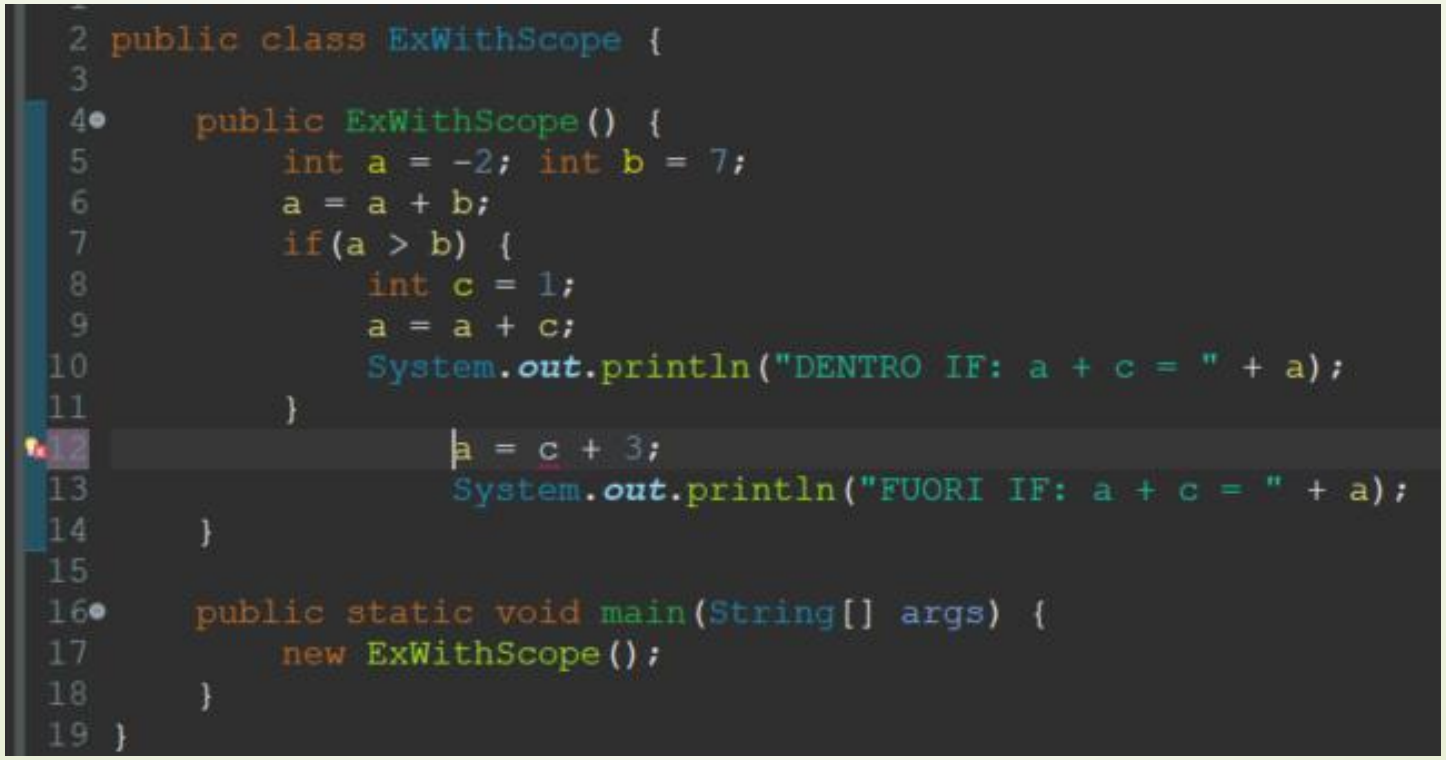
OUTPUT

```
DENTRO IF: a + c =  12
FUORI IF: a + c =  4
```

Cosa accadrebbe se fosse a = -2 e b = 7?

Lo stesso esempio con Java

In Java, che al contrario di Python, è tipizzato, è obbligatorio dichiarare le variabili prima di utilizzarle. Per cui questo errore viene segnalato subito al momento della scrittura del codice (indipendentemente dal valore di **a**).



Tipi di Dati Numerici (Java)

I tipi di dati che può contenere una variabile sono in genere simili per tutti i linguaggi, anche se poi ogni linguaggio può avere delle variazioni. Per esempio in Java non esiste il tipo «*unsigned*».

Numeri interi

Tipo	Dimensione	Valori
Byte	8 Bit	Da -128 a 127
Short	16 Bit	Da -32.768 a 32.767
Int	32 Bit	Da -2.147.483.648 a 2.147.483.647
Long	64 Bit	Da - 2^{63} a 2^{63}-1

Numeri in virgola mobile

Tipo	Dimensione	Precisione
Float	32 Bit	Singola
Double	64 Bit	Doppia

Tipi di Dati Testuali e Booleani (Java)

I tipi di dati che può contenere una variabile sono in genere simili per tutti i linguaggi, anche se poi ogni linguaggio può avere delle variazioni.

Dati Testuali

Tipo	Dimensione	Note
Char	16 Bit	Codifica Unicode
String	variabile	Si possono usare sequenze di escape
boolean	---	Valori: true o false

Tipi di Dati Numerici (Python)

Anche in Python non esiste il tipo «*unsigned*». Tuttavia può essere simulato utilizzando la libreria matematica «numpy»:

```
import numpy as np
x = np.uint8(255)  # 'x' varia da 0 a 255 → 8 bit
```

Tipo	Descrizione	Bit / Range	Signed / Unsigned	Note
int	Numero intero	Illimitato (dipende solo dalla memoria	Sempre signed	Python gestisce automaticamente la dimensione.
float	Numero in virgola mobile (IEEE 754)	64 bit (precisione doppia)	signed	Valori con parte decimale o esponenziale.
complex	Numero complesso (a + bj)	a e b sono di tipo FLOAT (64bit + 64bit)	signed (real e imag)	Ha parte reale e
bool	Valore logico (True o False)	1 bit (internamente int)	signed	In realtà è un sotto-tipo di

Tipi di Dati in Flowgorithm

I tipi di dati che si possono utilizzare in Flowgorithm (p.35 - manuale):

Integer

Real

String

Boolean

Operatori in Flowgorithm

Gli operatori matematici e logici per il confronto, possono variare a seconda del tipo di linguaggio.

Operator	C Family	BASIC Family	Mathematics (Unicode)
Equality	==	=	=
Inequality	!=	<>	≠
Less Than or Equal	<=	<=	≤
Greater Than Or Equal	>=	>=	≥
Logical Not	!	not	¬
Logical And	&&	and	∧
Logical Or	||	or	∨
Multiply	*	*	×
Divide	/	/	÷
Modulo	%	mod	(no symbol)

Operatori in Python

Categoria	Esempi principali
Aritmetici	`+ - * / // % **`
Confronto	`== != > < >= <=`
Logici	`and or not`
Bitwise	`&`
Appartenenza / Identità	`in not in is is not`

Operatori di Appartenenza in Python

Operatore	Descrizione	Esempio	Risultato
`in`	Vero se un elemento è contenuto in una sequenza	`'a' in 'casa'`	`True`
`not in`	Vero se non è contenuto	`'z' not in 'casa'`	`True`
`is`	Vero se due variabili puntano allo stesso oggetto	`a is b`	`True/False`
`is not`	Vero se non puntano allo stesso oggetto	`a is not b`	`True/False`

Stringhe in Python

Stringhe

In Python, dichiarare una variabile stringa è molto semplice: basta assegnare del testo a una variabile usando apici singoli ('...') o doppi apici ("...").

In Python non serve dichiarare il tipo (come **string** o **str**): il tipo viene determinato automaticamente. Se vuoi sapere che tipo ha una variabile, puoi usare type():

```
print(type(nome))  # Output: <class 'str'>
```

Esempi di Stringhe

```
# Esempi di variabili stringa
nome = "Luca"
saluto = 'Ciao a tutti!'
frase = "Oggi è una bella giornata."

# Puoi anche usare le triple virgolette per stringhe su più righe
testo_lungo = """Questa è
una stringa
su più righe."""

# Concatenazione semplice
saluto = "Ciao " + nome + "!"
print(saluto)  # Output: Ciao Luca!
```

Usare f-string

Le **f-string** (introdotte da Python 3.6) permettono di inserire variabili direttamente nel testo:

```
nome = "Luca"
eta = 25
messaggio = f"Ciao {nome}, hai {eta} anni!"
print(messaggio)
```

Le f-string sono potenti: puoi anche inserire espressioni, ad esempio:

```
print(f"Tra 5 anni avrai {eta + 5} anni.")
```

Formattare numeri decimali

Con **f-string**, puoi controllare quanti decimali mostrare:

```
prezzo = 12.3456
print(f"Prezzo: {prezzo:.2f} €")   # Output: Prezzo: 12.35 €
```

:.2f significa:

- **:** inizia la formattazione
- **.2** due cifre decimali
- **f** numero in formato "float"

Realizzare Algoritmi con procedimento iterativo

Project Work

Problema (pari/dispari)

Realizzare un algoritmo che dato in input un qualsiasi numero intero positivo verifica se è dispari o pari e ne comunichi il risultato.

PW: Soluzione informale

Problema (pari/dispari)

Realizzare un algoritmo che dato un numero intero positivo verifica se è dispari o pari e ne comunichi il risultato.

Idea (informale)

Un numero pari è divisibile per 2, per cui tolgo 2 al numero inserito tante volte fino a quando ho resto 1 o 0. Se il resto è 0 allora il numero iniziale inserito sarà un numero pari e viceversa.

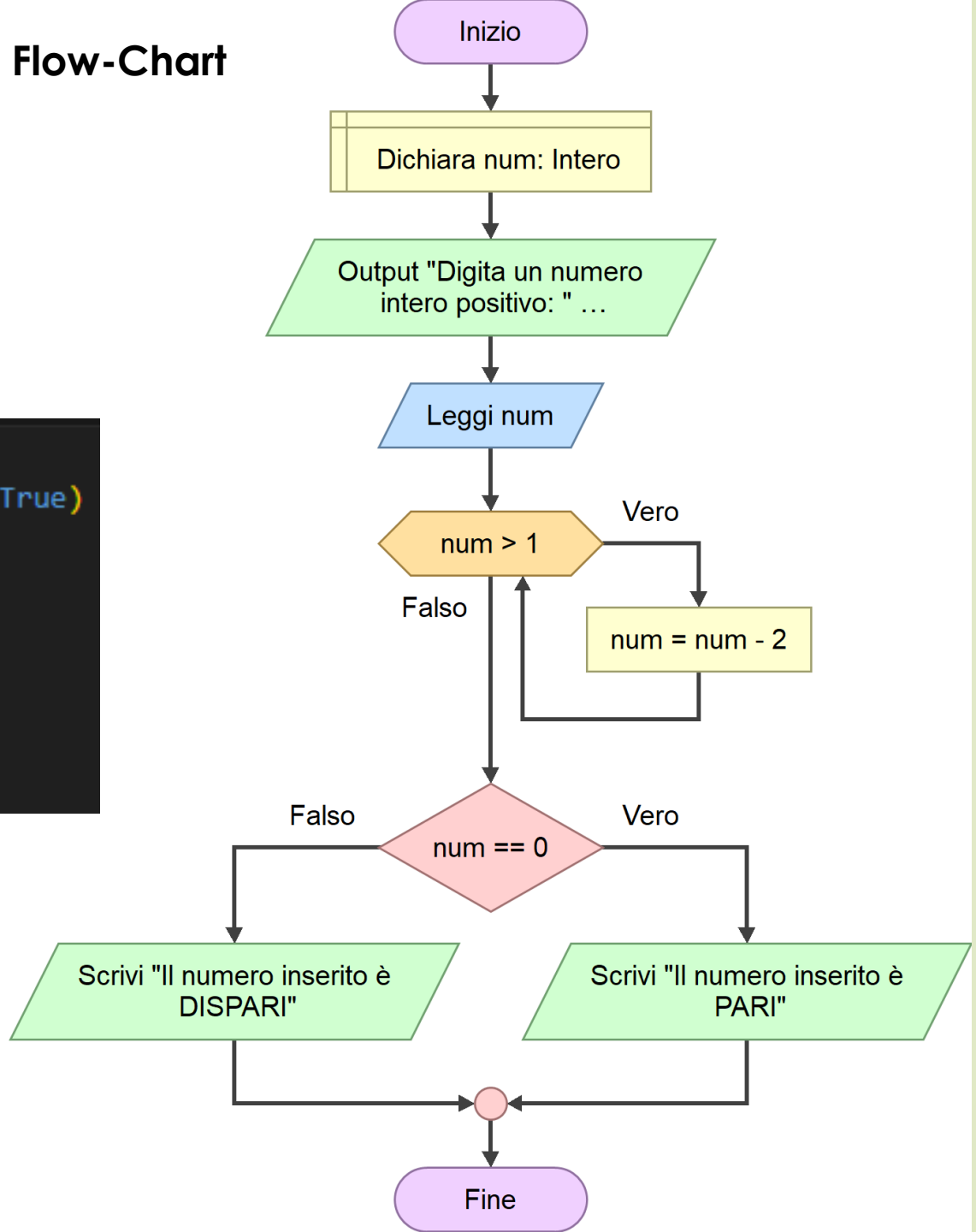
PW: Soluzione

Codice Python

pari_dispari.py > ...

```python
print("Digita un numero intero positivo: ", end='', flush=True)
num = int(input())
while num > 1:
    num = num - 2
if num == 0:
    print("Il numero inserito è PARI")
else:
    print("Il numero inserito è DISPARI")
```

Flow-Chart



Progetto della soluzione: Java

PariDispari.java

```java
import java.io.BufferedReader;

public class PariDispari {

    public static void main(String[] args) throws IOException{
        int num = 0;
        String str;

        InputStreamReader input = new InputStreamReader(System.in);
        BufferedReader tastiera = new BufferedReader(input);

        System.out.print("Digita un numero intero positivo: ");
        System.out.print("");
        str = tastiera.readLine();
        num = Integer.parseInt(str);

        while(num > 1) {
            num = num - 2;
        }
        if(num == 0) {
            System.out.println("Il numero inserito è PARI");
        } else {
            System.out.println("Il numero inserito è DISPARI");
        }
    }
}
```

Project Work

Problema (min)

Realizzare un algoritmo che riceva in input una serie di temperature e produca in output la temperatura minima.

L'inserimento deve continuare fino a che l'utente decida di uscire.

PW: Soluzione informale

Problema (min)

Realizzare un algoritmo che riceva in input una serie di temperature e produca in output la temperatura **minima**.

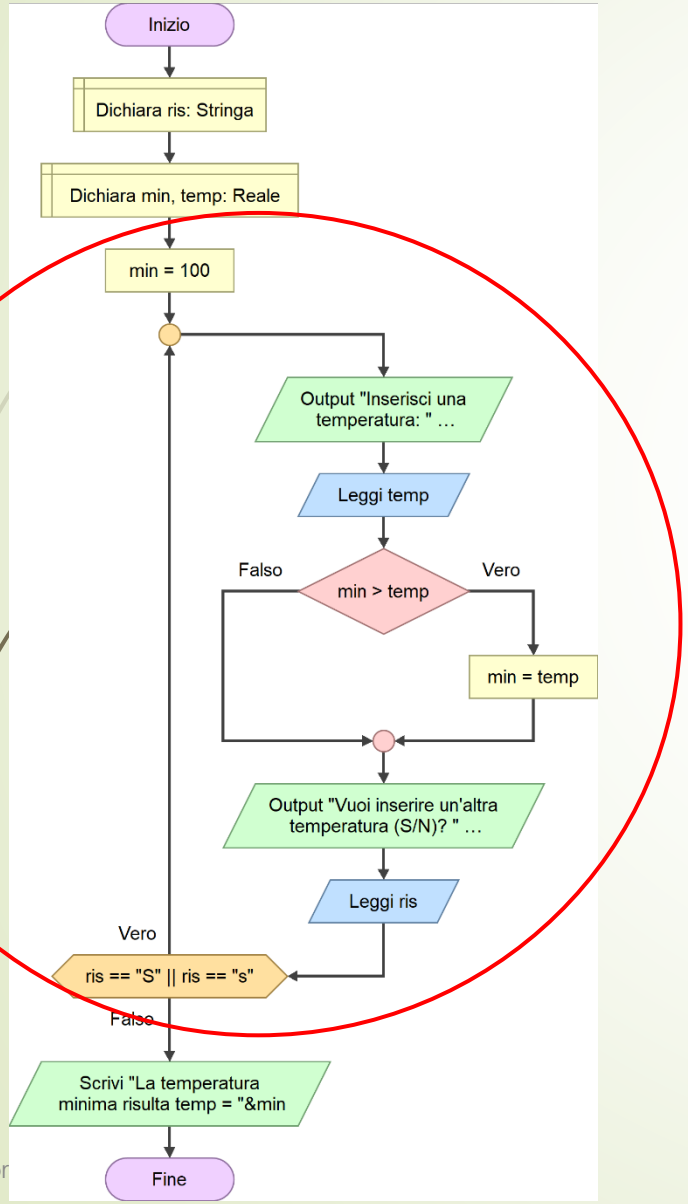
L'inserimento deve continuare fino a che l'utente decida di uscire.

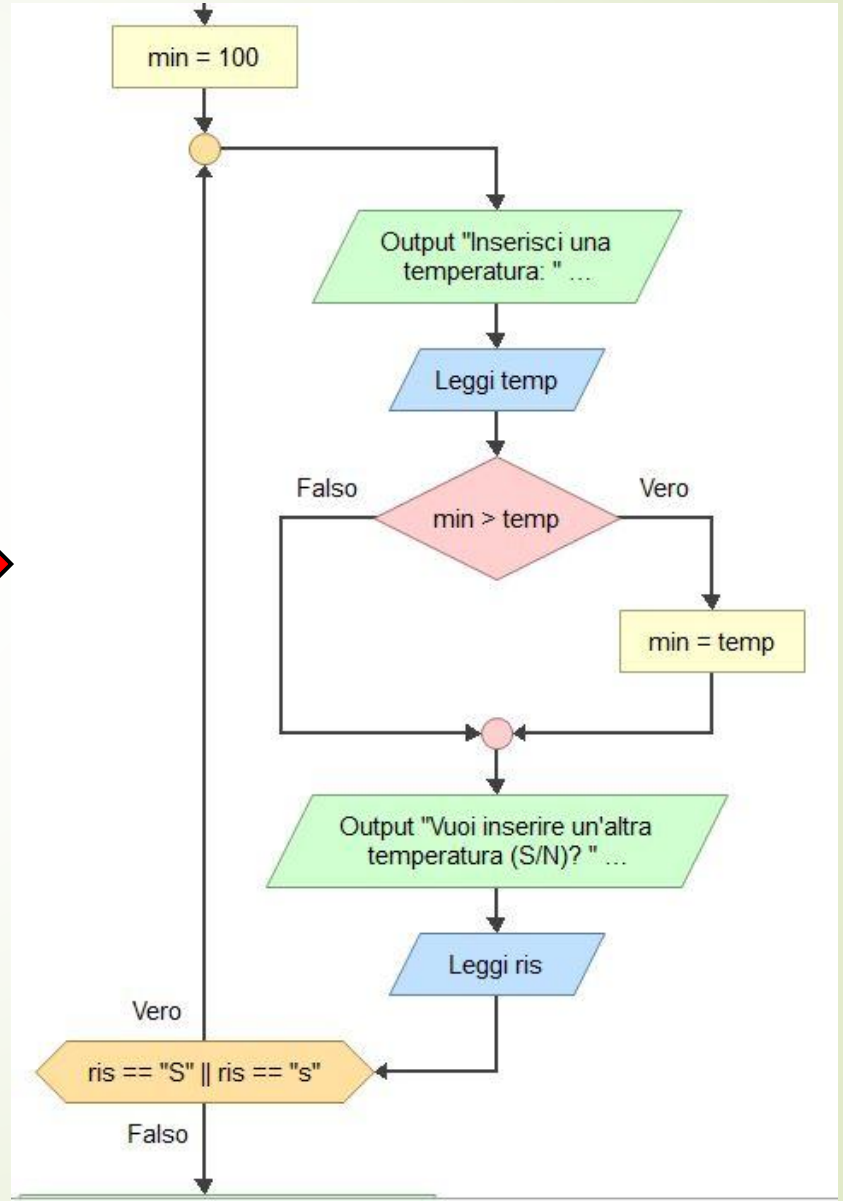
Idea (informale)

Realizzo l'input dentro un ciclo e, ad ogni inserimento, testo con una domanda per l'utente se vuole continuare ad inserire altre temperature.

Inizializzo una variabile «min» ad un valore iniziale alto, così da non perturbare la ricerca del minimo sull'insieme dato. Così «min» servirà da accumulatore del minimo relativo, operando un confronto tutte le volte che l'utente inserisce una temperatura. Al termine del ciclo, o fase di inserimento, «min» conterrà il minimo assoluto.

PW: Progetto della soluzione





PW: Realizzazione in Python

minimo_temp.py > ...

```
1    min = 100
2    while True:    #This simulates a Do Loop
3        print("Inserisci una temperatura °C: ", end='', flush=True)
4        temp = float(input())
5        if min > temp:
6            min = temp
7        print("Vuoi inserire un'altra temperatura (S/N)? ", end='', flush=True)
8        ris = input()
9        if not(ris == "S" or ris == "s"): break   #Exit loop
10   print("La temperatura minima risulta temp = " + str(min) + " °C")
```

Se gli input sono le temperature di fusione dei metalli, l'algoritmo continua a funzionare correttamente?

PW: Realizzazione in JAVA

Min.java

```java
import java.io.*;

public class Min {

    public static void main(String[] args) throws IOException{
        double min = 100.00;
        double temp;
        String str;
        String ris = "S";

        InputStreamReader input = new InputStreamReader(System.in);
        BufferedReader tastiera = new BufferedReader(input);

        do {
            System.out.print("Inserisci una temperatura: ");
            str = tastiera.readLine();
            temp = Double.parseDouble(str);
            if(min > temp) {
                min = temp;
            }
            System.out.println("Vuoi inserire un'altra temperatura (S/N)? ");
            ris = tastiera.readLine();

        } while(ris.equals("S") || ris.equals("s"));

        System.out.println("La temperatura minima risulta temp = " + min);
    }
}
```

Project Work

Problema (min e max)

Realizzare un algoritmo che riceva in input una serie di temperature e produca in output la temperatura **minima** e quella **massima**.

L'utente deve fornire in input il numero totale (ntemp) dei valori di temperature da inserire. L'algoritmo deve funzionare per qualsiasi valore di temperatura inserito.

PW: Soluzione informale

Problema (min e max)

Realizzare un algoritmo che riceva in input una serie di temperature e produca in output la temperatura minima e quella massima.

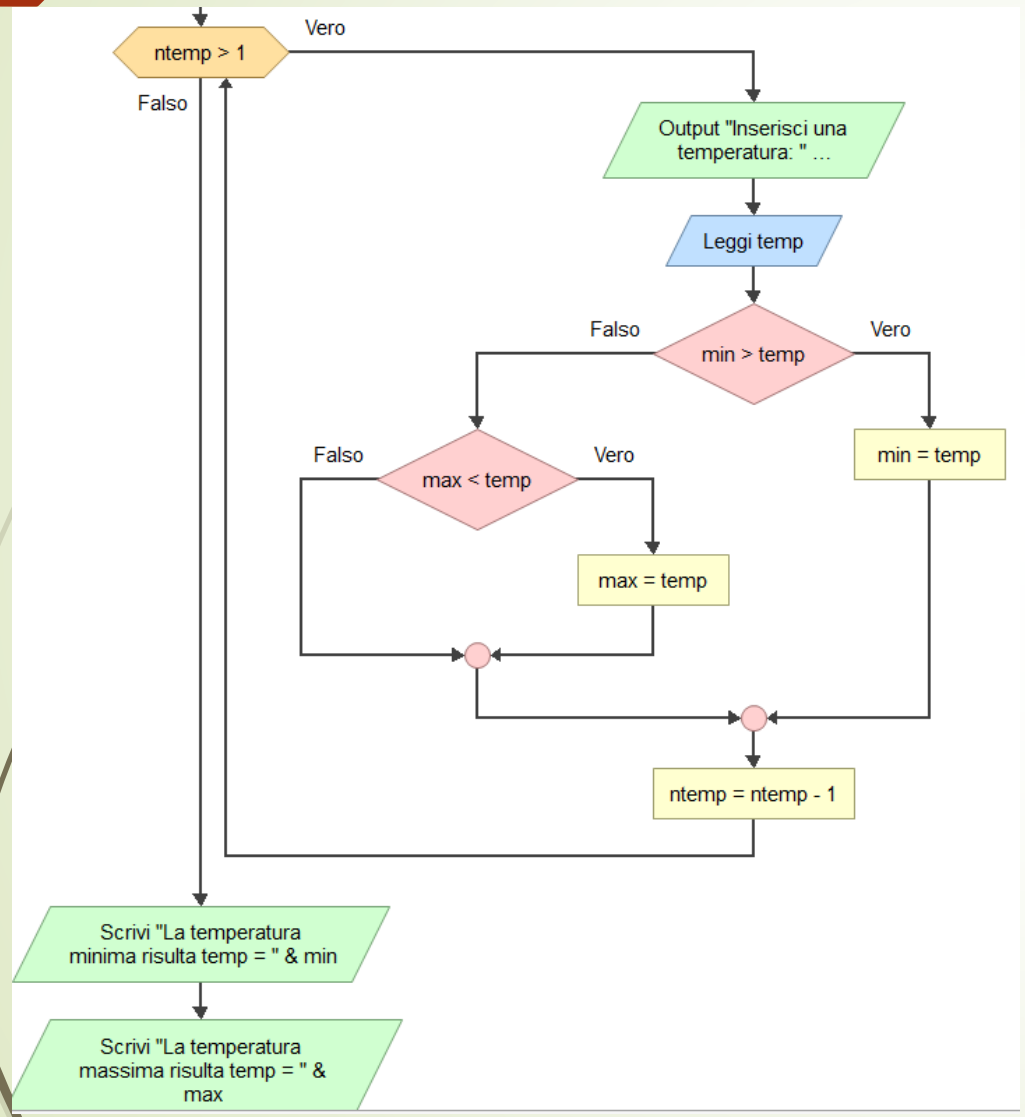
L'utente deve fornire, a priori come input, il numero totale (ntemp) dei valori di temperature da inserire. L'algoritmo deve funzionare per qualsiasi valore di temperatura inserito.

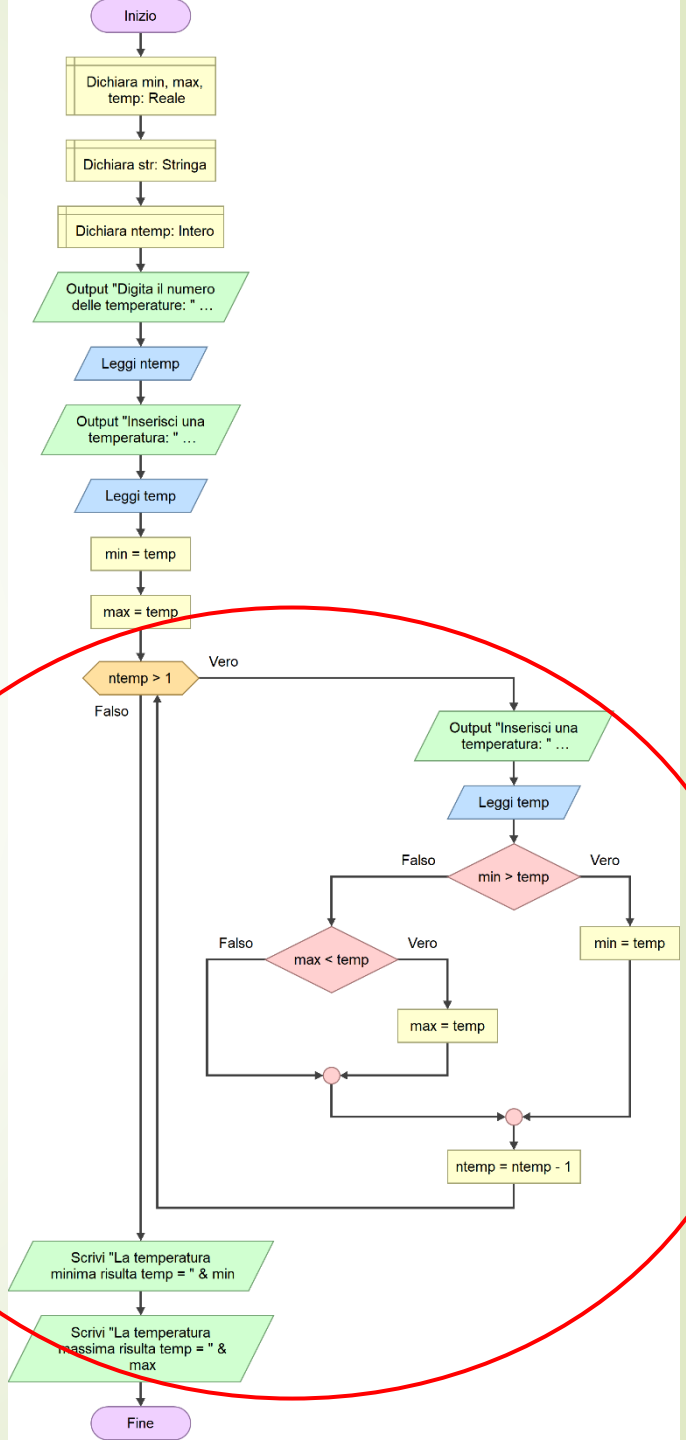
Idea (informale)

L'input «ntemp» determina il numero dei cicli. Utilizzo il primo inserimento di «temp» per inizializzare sia «min» che «max», che serviranno da accumulatori del minimo e del massimo relativi, operando un confronto tutte le volte che l'utente inserisce una temperatura.

Al termine del ciclo, o fase di inserimento, «min» e «max» conterranno il minimo e il massimo assoluti.

PW: Progetto della soluzione





PW: Realizzazione in Python

min_max_temp.py > ...

```python
print("Digita il numero delle temperature: ", end='', flush=True)
ntemp = int(input())
print("Inserisci una temperatura °C: ", end='', flush=True)
temp = float(input())
min = temp
max = temp
while ntemp > 1:
    print("Inserisci una temperatura °C: ", end='', flush=True)
    temp = float(input())
    if min > temp:
        min = temp
    else:
        if max < temp:
            max = temp
    ntemp = ntemp - 1
print("La temperatura minima risulta temp = ", min, "°C")
print("La temperatura massima risulta temp = " + str(max) + " °C")
```

PW: Realizzazione in Java

MinMax.java

```java
import java.io.*;

public class MinMax {

    public static void main(String[] args) throws IOException{
        double min,max;
        double temp;
        int ntemp = 0;
        String str;

        InputStreamReader input = new InputStreamReader(System.in);
        BufferedReader tastiera = new BufferedReader(input);

        System.out.print("Digita il numero di temperature: ");
        str = tastiera.readLine();
        ntemp = Integer.parseInt(str);

        System.out.print("");
        System.out.print("Inserisci una temperatura: ");
        str = tastiera.readLine();
        min = Double.parseDouble(str);
        max = min;

        while(ntemp > 1) {
            System.out.print("Inserisci una temperatura: ");
            str = tastiera.readLine();
            temp = Double.parseDouble(str);
            if(min > temp) {
                min = temp;
            } else if(max < temp) {
                max = temp;
            }
            ntemp = ntemp -1;
        }
        System.out.println("La temperatura minima risulta temp = " + min);
        System.out.println("La temperatura massima risulta temp = " + max);
    }
}
```

Project Work

Problema (potenza)

Realizzare un algoritmo che riceva in input un numero reale, la «base», e un numero intero positivo, l' «esponente», e calcoli e stampi l'elevamento a potenza. Ovviamente senza utilizzare eventuali operatori già preconfezionati del linguaggio adottato. Per esempio:

base = 3

esponente = 2 **=>** **$3^2 = 9$**

GRAZIE!