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

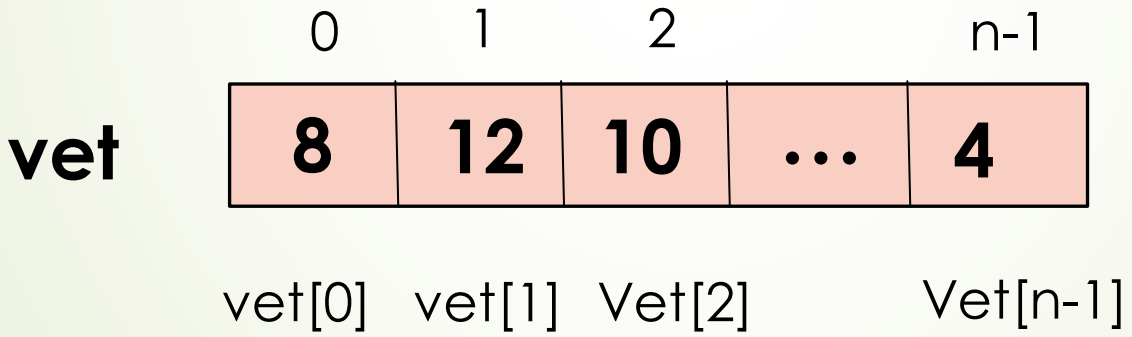
Strutture Dati
« vettore »

Struttura dati «Array» o «Vettore»

E' una struttura dati che contiene molti valori dello stesso tipo: interi, decimali, stringhe, ecc.

Per cui a differenza di una variabile che può contenere un valore alla volta, il «vettore» è una variabile complessa che può memorizzare più valori dello stesso tipo.

Per esempio, se vogliamo concettualmente rappresentare un vettore di numeri interi:



In Python usiamo le Liste

In Python, la struttura dati più simile a un array si chiama lista (**list**).

Una lista Python è una collezione ordinata e modificabile di elementi, ma non necessariamente omogenei: può contenere tipi diversi.

La scelta del nome "lista" deriva dal fatto che in Python le **list**:

- sono più flessibili degli array tradizionali,
- possono crescere o ridursi dinamicamente,
- possono contenere elementi eterogenei,
- internamente sono implementate come liste dinamiche (simili a vettori dinamici), non come array statici.

Quindi il nome "lista" enfatizza il loro uso generale e non solo la funzione di "vettore numerico".

In Python usiamo le Liste

In Python, la struttura dati più simile a un array si chiama lista (**list**).

Una lista Python è una collezione ordinata e modificabile di elementi, ma non necessariamente omogenei: può contenere tipi diversi.

La scelta del nome "lista" deriva dal fatto che in Python le **list**:

- sono più flessibili degli array tradizionali,
- possono crescere o ridursi dinamicamente,
- possono contenere elementi eterogenei,
- internamente sono implementate come liste dinamiche (simili a vettori dinamici), non come array statici.

Quindi il nome "lista" enfatizza il loro uso generale e non solo la funzione di "vettore numerico".

Esempi di Liste

numeri = [1, 2, 3, 4, 5]

Oppure

mista = [1, "ciao", 3.14, True]

Librerie esterne per l'array

Se serve un **array omogeneo e più efficiente** (per calcoli numerici o grandi quantità di dati), in Python si possono usare:

- Il modulo **array** della libreria standard ('i' indica il **tipo intero**):

```
import array
a = array.array('i', [1, 2, 3, 4])
```

- Oppure, più comunemente, **NumPy**:

```
import numpy as np
a = np.array([1, 2, 3, 4])
```

Questi sono veri array omogenei e più vicini agli array dei linguaggi tradizionali.

Esercizi con «Array» o «Vettore»

Problema ('ordini di acquisto')

Realizzare un algoritmo che riceva in input una sequenza di quantità relative agli ordini di un prodotto. Inoltre, l'utente specifica anche il prezzo unitario relativo al prodotto di cui si sono inseriti gli ordini.

L'algoritmo deve calcolare per ogni ordine il relativo costo considerando i seguenti criteri di sconto:

- Se Quantità > 10 e Quantità <= 20 allora si applica uno sconto del 5%
- Se Quantità > 20 allora si applica uno sconto del 10%

Esercizi con «Array» o «Vettore»

Problema (ordini di acquisto)

Realizzare un algoritmo che riceva in input una sequenza di quantità relative agli ordini di un prodotto. Inoltre, l'utente specifica anche il prezzo unitario relativo al prodotto di cui si sono inseriti gli ordini.

L'algoritmo deve calcolare per ogni ordine il relativo costo considerando i seguenti criteri di sconto:

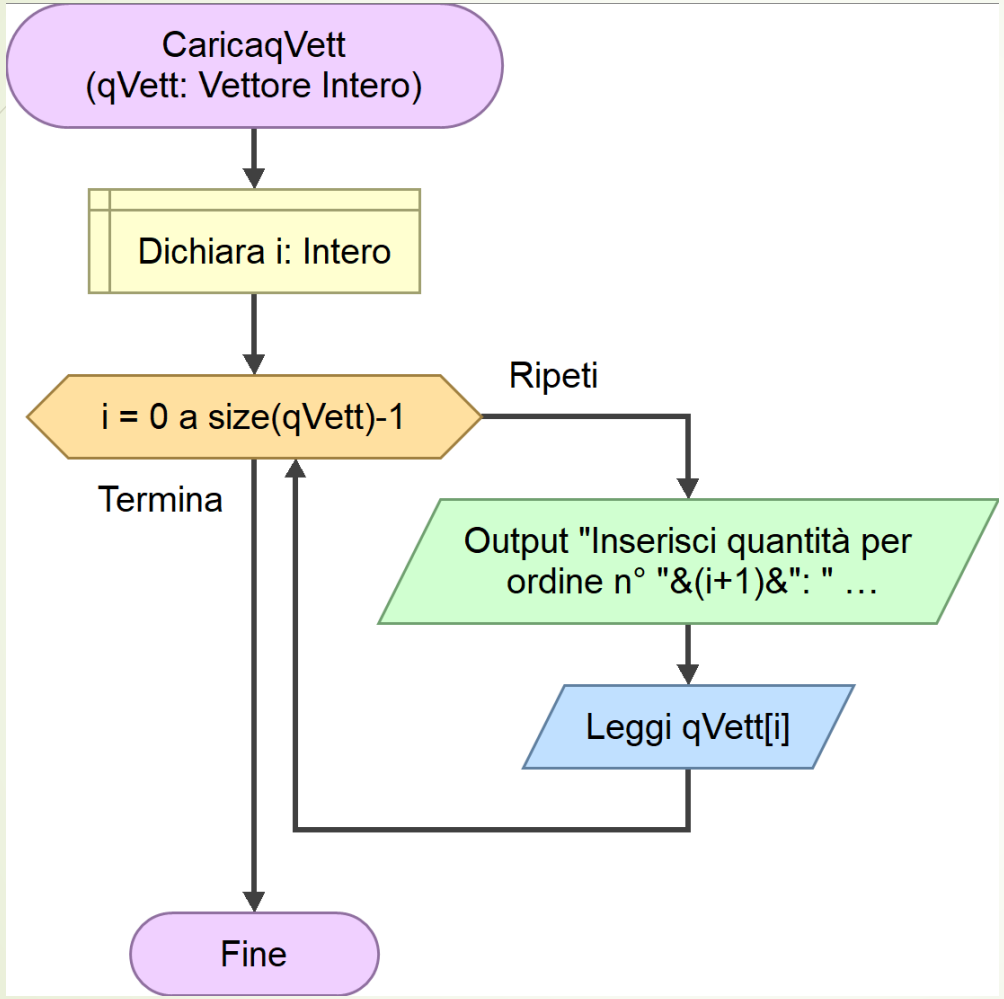
- Se Quantità > 10 e Quantità <= 20 allora si applica uno sconto del 5%
- Se Quantità > 20 allora si applica uno sconto del 10%

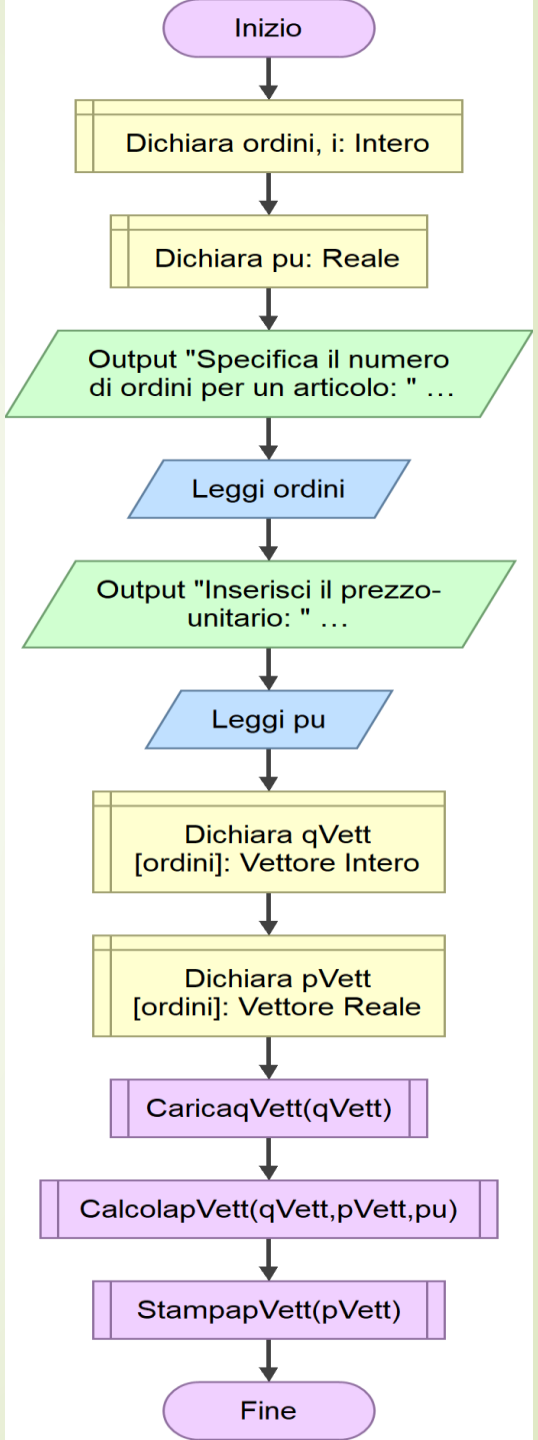
Idea (informale)

Si utilizzano due vettori: «qVett» per memorizzare le quantità e «pVett» per memorizzare i prezzi con relativi sconti degli ordini inseriti.

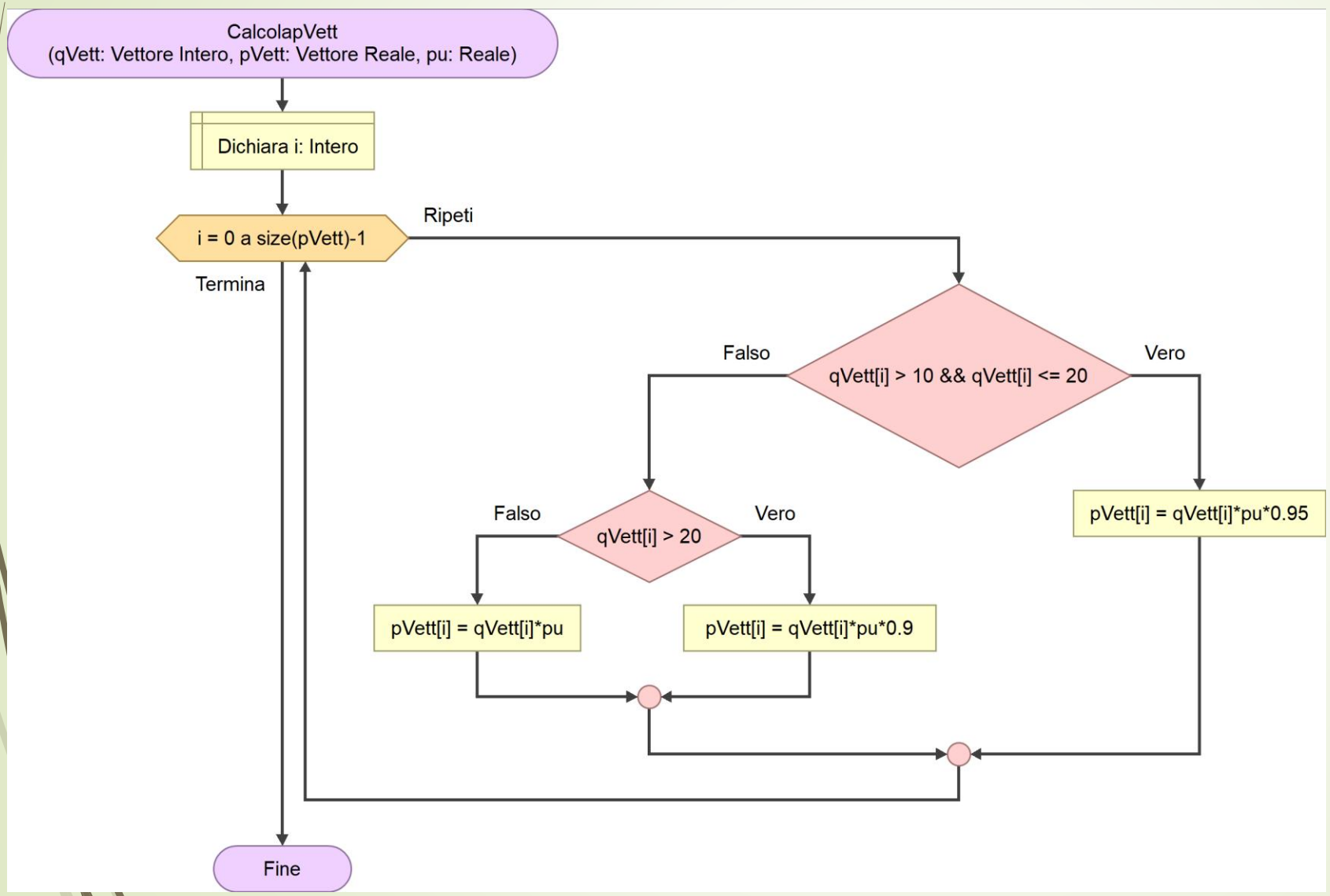
Si utilizzano 3 sottoprogrammi per: inserire dati in «qVett», calcolare «pVett» e stampare i risultati.

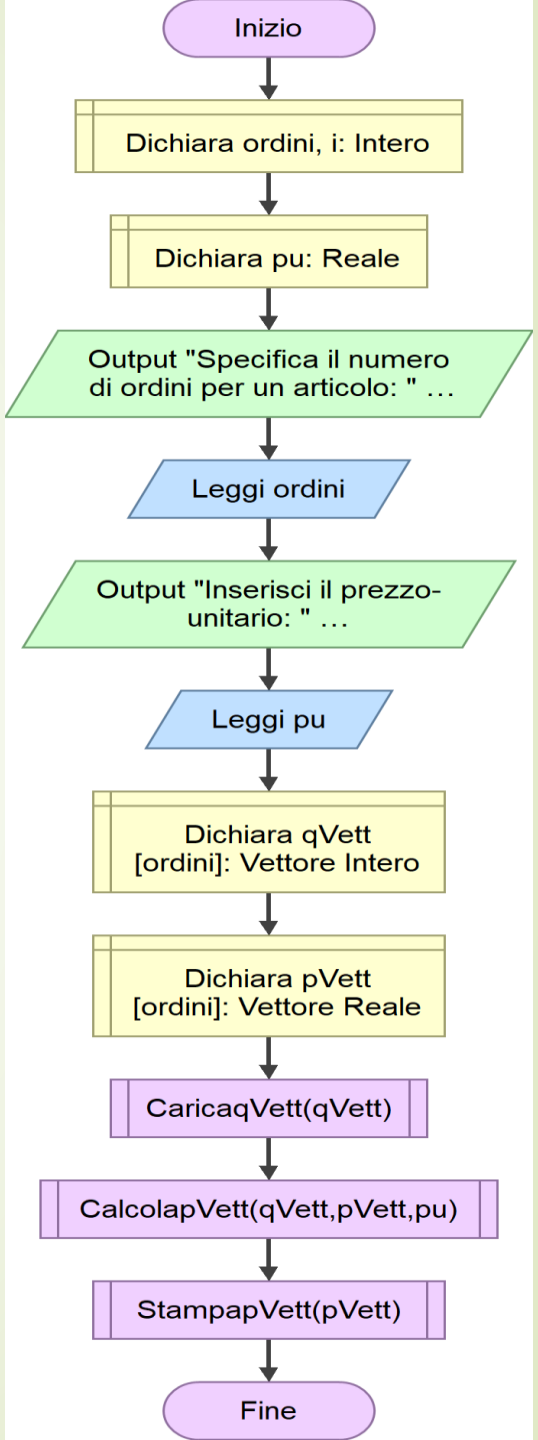
Esercizi con «Array» o «Vettore»



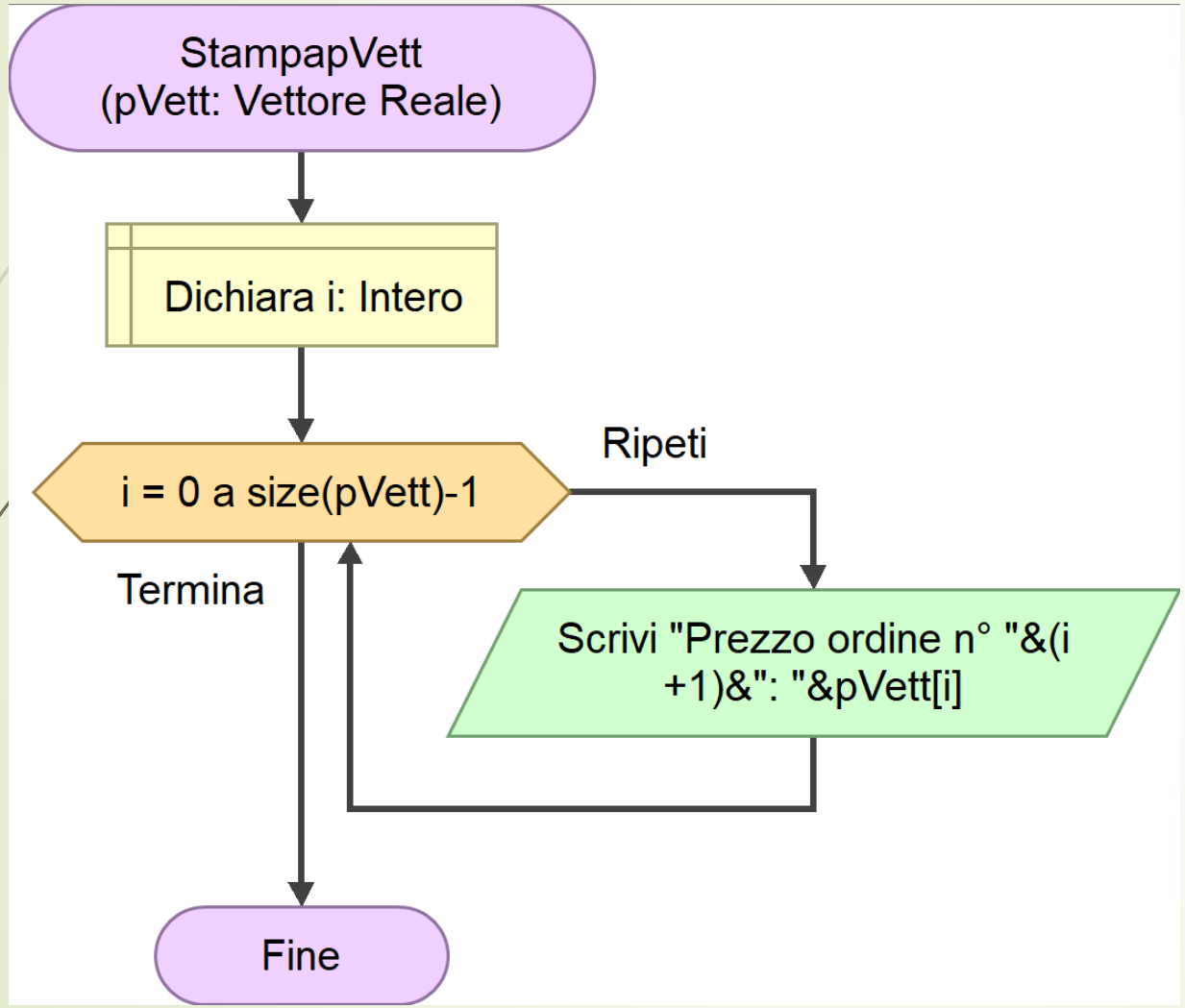


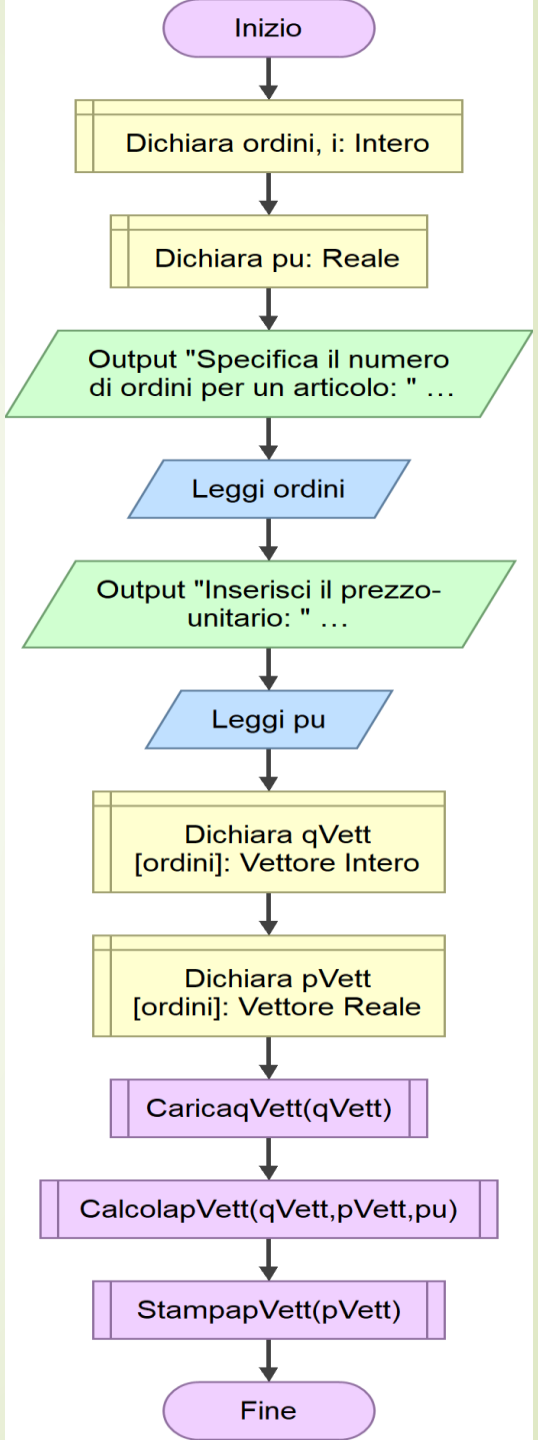
Esercizi con «Array» o «Vettore»





Esercizi con «Array» o «Vettore»





Esempio in Python

Esempio di OUTPUT

```
Specifica il numero di ordini per un articolo: 4
Inserisci il prezzo-unitario (€): 45

QUANTITA' ORDINATE
Inserisci quantità per ordine nº 1: 12
Inserisci quantità per ordine nº 2: 3
Inserisci quantità per ordine nº 3: 4
Inserisci quantità per ordine nº 4: 56

CALCOLO PREZZI DEGLI ORDINI INCLUSIVI DI EVENTUALI TIPOLOGIE DI SCONTO
Prezzo ordine nº 1: 513.0 €
Prezzo ordine nº 2: 135.0 €
Prezzo ordine nº 3: 180.0 €
Prezzo ordine nº 4: 2268.0 €
```

vettoreOrdini.py > ...

```python
'''
Calcola lo sconto sull'ordine di un prodotto in base alle quantità ordinate:
10 < Q <= 20 applica il 5% di sconto
Q > 20       applica il 10% di sconto
1 <= Q <= 10 il prezzo rimane PU*Q senza sconto

USEREMO COME SINONIMI 'LISTA' e 'VETTORE'
'''
def calcolapVett(qVett, pVett, pu):
    for i in range(0, len(pVett), 1):
        if qVett[i] > 10 and qVett[i] <= 20:
            pVett[i] = qVett[i] * pu * 0.95
        else:
            if qVett[i] > 20:
                pVett[i] = qVett[i] * pu * 0.9
            else:
                pVett[i] = qVett[i] * pu

def caricaqVett(qVett):
    print("")
    print("QUANTITA' ORDINATE")
    for i in range(0, len(qVett), 1):
        print("Inserisci quantità per ordine nº " + str(i + 1) + ": ", end='', flush=True)
        qVett[i] = int(input())

def stampapVett(pVett):
    print("")
    print("CALCOLO PREZZI DEGLI ORDINI INCLUSIVI DI EVENTUALI TIPOLOGIE DI SCONTO")
    for i in range(0, len(pVett), 1):
        print("Prezzo ordine nº " + str(i + 1) + ": " + str(pVett[i]) + " €")

# Main
print("Specifica il numero di ordini per un articolo: ", end='', flush=True)
ordini = int(input())
print("Inserisci il prezzo-unitario (€): ", end='', flush=True)
pu = float(input())
qVett = [0] * (ordini) # produce una lista = [0,0,...,0] di lunghezza pari a 'ordini'
pVett = [0] * (ordini) # produce una lista = [0,0,...,0] di lunghezza pari a 'ordini'

caricaqVett(qVett)
calcolapVett(qVett, pVett, pu)
stampapVett(pVett)
```

GRAZIE!