

# PreLab4

**Quando il gioco si fa duro....**

**Informatica, Corso B - D. Tamascelli**

# n-uple: dichiarazione-accesso

- Dichiarazione:
- serve un modo per distinguere dichiarazione di variabili da dichiarazione di n-uple...
  - ...e comunque per definire una n-upla serve un'informazione in più: la dimensione
  - il tipo, ovviamente, dovrà restare

```
//Dichiarazione di un array di 5 interi  
int pippo[5];
```

```
//Dichiarazione di un array di 7 float  
float poldo[7];
```

```
//Dichiarazione e inizializzazione di un array di 3 double  
double pluto[] = {1.,2.,3.};  
//pluto viene creato come vettore di 3 double  
//e riempito con i valori dati.
```

semplice no?!

# n-uple: dichiarazione-accesso

## Accesso:

- ogni elemento di una n-upla è moralmente una normale variabile
- ma come si chiama?  
 $x_i \rightarrow x[i], i = 0, 1, \dots, n - 1$

- Quindi:

I.  $x_i = 5 \rightarrow x[i] = 5$

II.  $x_j = x_i \rightarrow x[j] = x[i]$

III. Si comincia a contare da 0



```
//Ogni elemento di un array si usa  
//come una normale variabile  
pippo[0] = 4;  
pippo[1] = 3;  
pippo[2] = 2;  
pippo[3] = pippo[2]-1;  
pippo[4] = pippo[3]-1;
```

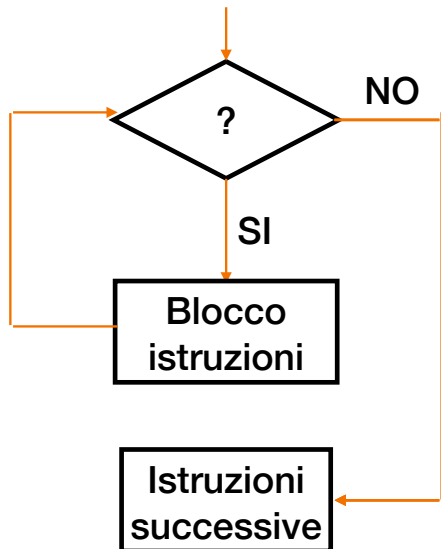
[...]: operatore di accesso....

...stessa semantica, a questo livello, di un pedice...

...usatelo così!

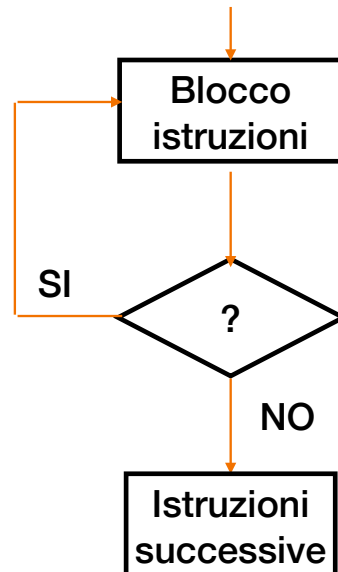
# Iterazione: varianti sul tema

Iterazione precondizionale



```
while(cond){  
    //Blocco  
}
```

Iterazione postcondizionale



```
do{  
    //Blocco  
}while(cond);
```

Il blocco istruzioni viene eseguito **ALMENO** una volta

Esempio: acquisizione input da tastiera con controllo:

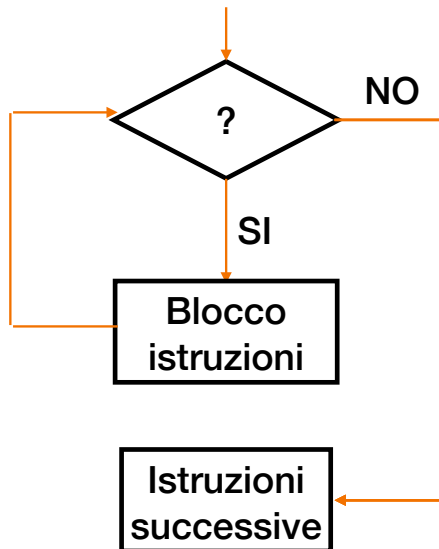
"Inserire due numeri interi positivi"

I due interi devono essere letti almeno una volta prima di controllarli ed eventualmente richiederli.

```
do{  
    cout << "Inserire a , b > 0" << endl;  
    cin >> a >> b;  
} while(a <=0 or b<=0);
```

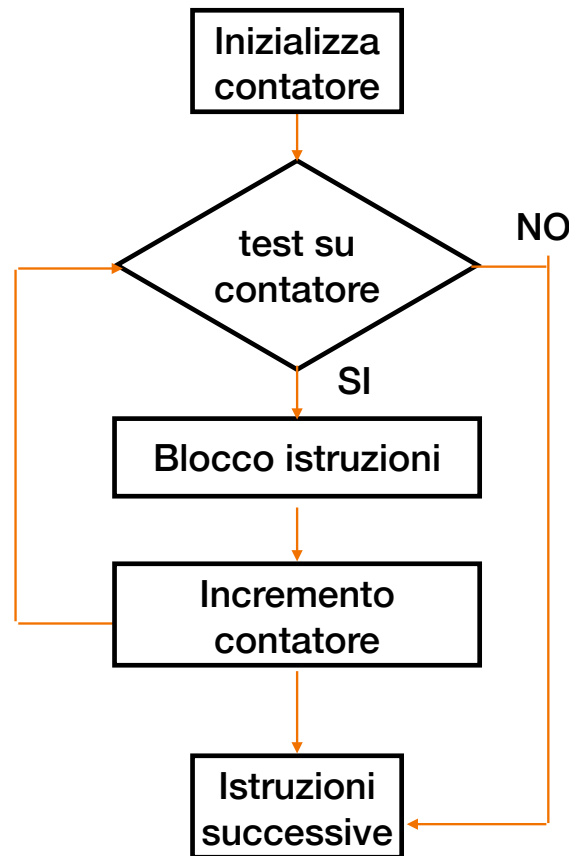
# Iterazione: varianti sul tema

Iterazione precondizionale



```
while(cond){  
    //Blocco  
}
```

Iterazione precondizionale su contatore



```
for(int i=0; i<10; i++){  
    //Blocco  
}
```

```
int i=0;  
while(i<10){  
    //Blocco  
    i++; //i = i+1;  
}
```



# "Reason is but choosing"

- Al programmatore viene fornita la libertà di scegliere quale costrutto di iterazione si adatta meglio al problema che sta affrontando.
- Non esistono regole, ma pratiche. Per esempio:
  - Il for si usa preferibilmente quando il numero di iterazioni è noto a priori.
  - Il while postcondizionale si usa quando il blocco deve essere eseguito almeno una volta.
- ...ma il Teorema di Jacopini-Böhm garantisce che i tre costrutti sono equivalenti, quindi esiste un modo per trasformare uno in un altro.