

Sequenze: nomi simbolici con indice

Consentiamo anche l'uso di nomi simbolici con **indice**, per rappresentare sequenze come quelle degli array.

$$c[i] \rightsquigarrow 5$$

dove i è un valore naturale. Ad esempio:

$$\{i \rightsquigarrow K, c[0] \rightsquigarrow 0, \dots, c[K-1] \rightsquigarrow 0\} \quad K \geq 0$$

Nello stato sono presenti:

- ▶ un'associazione per il nome simbolico i , al quale è associato un valore naturale K .
- ▶ K associazioni per i nomi simbolici $c[0], c[1], \dots, c[K-1]$, a ciascuno dei quali è associato il valore 0 .

All'interno degli algoritmi, consentiamo di riferire nomi simbolici con indice nella forma $c[exp]$ dove exp deve essere una espressione a valori **naturali**.

Anche in questo caso exp non deve contenere valori generici (nell'esempio, non è lecito un assegnamento del tipo $c[K-1] = 5$, mentre è lecito $c[i-1] = 5$).

Esempio: Calcolare il massimo di una sequenza data di interi.

Stato iniziale: $\{\text{dim} \rightsquigarrow K, c[0] \rightsquigarrow V_0, \dots, c[K-1] \rightsquigarrow V_{K-1}\}$ con $K > 0$

Stato finale: $\{\text{massimo} \rightsquigarrow \max(\{V_0, \dots, V_{K-1}\})\}$

- Dobbiamo calcolare

$$\max\{c[j] \mid j \in [0, \text{dim}-1]\}$$

- Di nuovo, utilizziamo un ciclo controllato da una variabile di controllo i che assume tutti i valori nell'intervallo $[0, \text{dim}-1]$.
- Facciamo in modo che, ad ogni iterazione, valga la seguente proprietà

$$\text{massimo} = \max\{c[j] \mid j \in [0, i-1]\}$$

Per ottenere questo ad ogni iterazione confrontiamo l'elemento corrente con il "massimo in carica".

- Se, alla fine del ciclo, $i = \text{dim}$, abbiamo quanto desiderato e cioè

$$\text{massimo} = \max\{c[j] \mid j \in [0, \text{dim}-1]\}$$

Soluzione:

```
massimo = c[0];
i = 1;
while (i <= dim-1)
{
    qui massimo = max{c[0], ..., c[i-1]}
    if (c[i] > massimo)
        massimo = c[i];
    i = i + 1;  anche qui massimo = max{c[0], ..., c[i-1]}
}
```

- ▶ Inizializzare la variabile `massimo` a 0 sarebbe sbagliato: non è detto che tutti i numeri della sequenza siano negativi.
- ▶ Attenzione: l'istruzione `i = i + 1` viene eseguita dopo l'istruzione `if`, indipendentemente dalla valutazione della condizione. Comunque dobbiamo scorrere tutta la sequenza.
- ▶ Per calcolare il massimo di K elementi devo fare $K-1$ confronti, dato che sono proprio $K-1$ gli elementi che devono uscire perdenti.

Esempio: Calcolare il numero di elementi pari in una sequenza data.

Stato iniziale: $\{\text{dim} \rightsquigarrow K, c[0] \rightsquigarrow V_0, \dots, c[K-1] \rightsquigarrow V_{K-1}\}$ con $K > 0$

Stato finale: $\{\text{count} \rightsquigarrow \#\{j \mid j \in [0, K-1] \wedge V_j \text{ pari}\}\}$

- Dobbiamo calcolare

$$\sum_{\substack{0 \leq j \leq \text{dim}-1 \\ c[j] \text{ pari}}} 1$$

- Di nuovo, utilizziamo un ciclo controllato da una variabile di controllo i che assume tutti i valori nell'intervallo $[0, \text{dim}-1]$.
- Facciamo in modo che, ad ogni iterazione, valga la seguente proprietà

$$\text{count} = \sum_{\substack{0 \leq j \leq i-1 \\ c[j] \text{ pari}}} 1$$

- Se, alla fine del ciclo, $i = \text{dim}$, abbiamo quanto desiderato e cioè

$$\text{count} = \sum_{\substack{0 \leq j \leq \text{dim}-1 \\ c[j] \text{ pari}}} 1$$

Soluzione:

```

count = 0;
i = 0;                                punto 1
while (i <= dim-1)
{
    if (c[i] % 2 == 0)
        count = count + 1;
    i = i + 1;
}

```

- ▶ Attenzione: l'istruzione $i = i + 1$ viene eseguita dopo l'istruzione **if**, indipendentemente dalla valutazione della condizione.
- ▶ Nello stato iniziale del **while**, al punto (1), $\text{count} \rightsquigarrow 0$, $i \rightsquigarrow 0$ e dunque

$$\sum_{\substack{0 \leq j \leq i-1 \\ c[j] \text{ pari}}} 1 = \sum_{\substack{1 \leq j \leq 0 \\ c[j] \text{ pari}}} 1 = 0 = \text{count}$$

- ▶ il corpo del **while** mantiene vera la proprietà

$$\text{count} = \sum_{\substack{0 \leq j \leq i-1 \\ c[j] \text{ pari}}} 1$$

Considerazioni generali

- ▶ In molti problemi è necessario operare allo stesso modo su tutti gli elementi di un intervallo dato

per ogni $j \in [a, b]$ OP_j

esempio: $\sum_{j=a}^b exp_j$

- ▶ in tutti questi casi si ricorre a cicli in cui una variabile di controllo permette di **scandire** tutto l'intervallo di interesse

```
i = a;  
while (i <= b)  
{  
    <operazione in funzione di i>  
    i = i+1;  
}
```

Esercizio Controllare se un valore dato appare in una sequenza data

Stato iniziale: $\{\text{dim} \rightsquigarrow K, \text{val} \rightsquigarrow V, c[0] \rightsquigarrow V_0, \dots, c[K-1] \rightsquigarrow V_{K-1}\}$

Stato finale: $\{\text{trovato} \rightsquigarrow b = 1 \text{ se } \exists j \in [0, K-1] \wedge V_j = V, b = 0 \text{ altr.}\}$