

Sequenze: nomi simbolici con indice

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

$$c[i] \rightsquigarrow 5$$

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 costanti generiche (nell'esempio, non è lecito un assegnamento del tipo $c[K-1] = 5$, mentre è lecito $c[i-1] = 5$).

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

Stato iniziale: $\{dim \rightsquigarrow K, c[0] \rightsquigarrow V_1, \dots, c[K-1] \rightsquigarrow V_{K-1}\}$ con $K > 0$

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

- ▶ Dobbiamo calcolare

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

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

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

- ▶ Se, alla fine del ciclo, $i=dim$, abbiamo quanto desiderato e cioè

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

Soluzione:

```

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

```

- Nello stato iniziale del **while**, al punto (1), $\text{count} \rightsquigarrow 0$, $i \rightsquigarrow 0$ e dunque

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

- 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;
}

```

Esercizi proposti

Problema 1: Calcolare il numero di occorrenze di un valore dato in una sequenza data

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

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

Problema 2: Calcolare il massimo e il minimo 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: $\{ \text{massimo} \rightsquigarrow \max(\{V_0, \dots, V_{K-1}\}), \text{minimo} \rightsquigarrow \min(\{V_0, \dots, V_{K-1}\}) \}$

Problema 3: Calcolare il numero di occorrenze del valore massimo in una sequenza di interi

Stato iniziale: ? Stato finale: ?

Problema 4: Calcolare il numero di occorrenze di una cifra nella rappresentazione decimale di un numero naturale

Soluzione problema 1:

Stato iniziale: $\{ \text{dim} \rightsquigarrow K, \text{val} \rightsquigarrow V, c_0 \rightsquigarrow V_0, \dots, c_{K-1} \rightsquigarrow V_{K-1} \}$ con $K > 0$

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

```
occ = 0;
i = 0;
while (i < dim)
{
    if (c[i] == val)
        occ = occ + 1;
    else ;
    i = i + 1;
}
```

Ad ogni iterazione del ciclo vale la proprietà:

$\text{occ} = \#\{ j \mid j \in [0, i) \wedge c[j] = \text{val} \}$

Soluzione problema 2

Stato iniziale: $\{\text{dim} \rightsquigarrow K, c_0 \rightsquigarrow V_0, \dots, c_{K-1} \rightsquigarrow V_{K-1}\}$ con $K > 0$

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

- Preoccupiamoci prima di calcolare il **massimo**
- Scorriamo l'intera sequenza, attraverso una variabile di controllo i , facendo in modo che, ad ogni scansione, valga la proprietà

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

```

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

```

- Il calcolo del minimo è analogo. Otteniamo:

```

massimo = c[0];
minimo = c[0];
i = 1;
while (i < dim)
{
    if (c[i] > massimo)           calcolo del massimo
        massimo = c[i];
    else ;
    if (c[i] < minimo)           calcolo del minimo
        minimo = c[i];
    else ;
    i = i + 1;
}

```

- Usiamo un algoritmo piu' efficiente: diminuiamo il numero di confronti.

```

massimo = c[0];
minimo = c[0];
i = 1;
while (i < dim)
{
    if (c[i] > massimo)           calcolo del massimo
        massimo = c[i];
    else
        if (c[i] < minimo)       calcolo del minimo
            minimo = c[i];
        else ;
    i = i + 1;
}

```

Soluzione problema 3

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

Stato finale: $\{ \text{occ} \rightsquigarrow \#\{j \mid j \in [0, K-1] \wedge V_j = \max\{V_0, \dots, V_{K-1}\}\} \}$

- Un algoritmo ingenuo:
 - Calcoliamo il valore massimo come nel problema 2
 - Scandiamo nuovamente la sequenza per contare il numero di occorrenze del massimo
- Comporta una duplice scansione della sequenza
- Un algoritmo che comporta una singola scansione si ottiene adattando quello visto per il calcolo del massimo

```
massimo = c[0];
i = 1;
occ = 1;
while (i < dim)
{
    if (c[i] > massimo)
    {
        massimo = c[i];
        occ = 1;
    }
    else
        if (c[i] == massimo)
            occ = occ + 1;
        else ;
    i = i + 1;
}
```