Soluzione problema 1:

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

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

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

Ad ogni iterazione del ciclo vale la proprietà:

$occ = \#\{\ j\ \mid\ j \in [0,i\text{-}1] \wedge\ c[j] = val\}$

Variazione problema 1:

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

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

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

Soluzione alternativa alla variazione problema 1:

Dato che la sequenza è memorizzata, posso scorrerla anche a partire dal fondo. Ad ogni iterazione aggiorno la posizione dell' occorrenza corrente (scandendo da destra a sinistra) del valore cercato: alla fine rimarrà la posizione dell'ultima occorrenza (sempre scandendo da destra a sinistra), che corrisponde alla prima (scandendo da sinistra a destra).

```
occ = 0; pos = -1; i = dim-1;
while (i >= 0)
{
    if (c[i] == val)
        {occ = occ + 1;
        pos = i;}
    i = i + 1;
}
```

Se dovessi invece elaborare una sequenza di dati in ingresso senza prima memorizzarli, dovrei usare la prima soluzione.

Soluzione problema 2

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

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

- Come calcolare il massimo è ormai noto, e calcolare il minimo è del tutto analogo (basta invertire il confronto). Per calcolare entrambi possiamo procedere nel modo seguente.

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

- Ottimizzando:

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


Soluzione problema 3

Stato iniziale: $\{\text{dim} \rightsquigarrow \text{K}, \text{c[0]} \rightsquigarrow \text{V}_0, \ldots, \text{c[K-1]} \rightsquigarrow \text{V}_{\text{K-1}}\}$

Stato finale: $\{\text{occ} \rightsquigarrow \#\{\text{j} \mid \text{j} \in [0,\text{K-1}] \wedge \text{V}_\text{j} = \mathit{max}\{\text{V}_0, \ldots, \text{V}_{\text{K-1}}\}\}$

- 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-1)
{
    if (c[i] > massimo)
    {
        massimo = c[i];
        occ = 1;
    }
    else
        if (c[i] == massimo)
            occ = occ + 1;
        else ;
    i = i + 1;
}
```


Soluzione problema 4:

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

Stato finale: $\{somma \rightsquigarrow \sum_{j=0}^{K-1} V_j,\ media \rightsquigarrow (\sum_{j=0}^{K-1} V_j)/K\ \}$

```
media = 0;
somma = c[0];
i = 1;
while (i <= dim -1)
{
    somma = somma + c[i];
    i = i + 1;
}
media = somma/dim;
```

Ad ogni iterazione del ciclo vale la proprietà:

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