Fondamenti di Informatica: esercitazione di laboratorio n. 3
Vettori
Pier Luca Montessoro

NOTA: per tutti gli esercizi proposti si richiede di aggiungere funzioni alle librerie viste a lezione e scrivere opportuni programmi main per provarle, facendo anche uso delle funzioni di lettura e stampa dei vettori già scritte.

1. Istogramma verticale

Si scriva un programma che legga una sequenza (di lunghezza massima 100) di numeri interi compresi tra 0 e 20 e che ne stampi in verticale l'istogramma delle occorrenze.

Eventuali valori della sequenza al di fuori dell'intervallo previsto vanno ignorati ai fini del calcolo delle occorrenze, segnalandolo con un messaggio.

Esempi:

```
#istogramma_verticale
inserisci il numero di valori da scrivere nel vettore: 5
v[0] = 0
v[1] = 20
v[2] = 0
v[3] = 19
v[4] = 5
 *
 *              *                                                 *   *
 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

```
#istogramma_verticale
inserisci il numero di valori da scrivere nel vettore: 10
v[0] = 7
v[1] = 7
v[2] = 7
v[3] = 21
v[4] = -3
v[5] = 14
v[6] = 14
v[7] = 7
v[8] = 14
v[9] = 7
e` stato scartato il voto non valido 21
e` stato scartato il voto non valido -3
                       *
                       *
                       *                    *
                       *                    *
                       *                    *
 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

Si osservi che l'ultima riga contiene i valori corrispondenti agli indici dei contatori stampati con la stringa di formato "%2d ".

2. Istogramma testo normalizzato

Si scriva un programma che legga da tastiera un testo (terminato da EOF, che da tastiera va inserito premendo control-D) e stampi in verticale l'istogramma normalizzato delle occorrenze delle lettere. Prima della stampa dell'istogramma, il programma deve scrivere il fattore di scala che va calcolato secondo questo criterio: - se il massimo valore dei contatori è minore o uguale a 20, il fattore di scala è pari a 1 e ogni asterisco rappresenta una unità; - se il massimo valore dei contatori è maggiore di 20, il fattore di scala deve essere aumentato in proporzione, quindi ogni riga rappresenta un numero di unità pari al valore massimo dei contatori / 20.0.

Esempi:

```
#isto_testo_normalizzato
testo di prova
seconda riga
terza riga

fattore di scala: 1.000000
*
*                 *
*   *   *     *  * *
*  ** * *     *  ***
* *** * *    *** *** *   *
abcdefghijklmnopqrstuvwxyz
```

NOTA: il testo può anche essere contenuto in un file e letto automaticamente dal programma tramite la redirezione dell'input tramite il simbolo "<":

```
#isto_testo_normalizzato < divina_commedia.txt

fattore di scala: 760.400000
    *
    *
*   *
*   *   *
*   *   *     *
*   *   *     *
*   *   *     *
*   *   *     *
*   *   *     *
*   *   *    **  *
*   *   *  * **  * *
* * *   *  * **  ***
* * *   *  * **  ***
* * *   *  * **  ***
* ***   *  * **  ***
* ***   *  ****  ****
* ***   *  ***** ****
* *** ***  ***** *****
* *******  ***** *****
*********  ***********
*********  ********** * *
abcdefghijklmnopqrstuvwxyz
```

3. Ricerca in vettore

Si scriva la funzione di ricerca lineare (cioè scandendo elemento per elemento il sottovettore che inizia dall'indice `inizio` e termina all'indice `fine`)

```
int cerca_in_sottovettore (int v[], int inizio, int fine, int valore_cercato);
```

che restituisce l'indice dell'elemento contenente la prima occorrenza del valore cercato nella porzione di vettore compresa tra gli indici `inizio` e `fine`, estremi inclusi. La funzione deve restituire -1 se il valore cercato non è presente nel sottovettore.

```
inizializza l'indice al primo elemento;
while (ci sono ancora elementi da controllare)
{
   if (l'elemento corrente è quello cercato)
      return indice corrente;
   avanza con l'indice di una posizione;
}
return -1;
```

NOTA: questo algoritmo rappresenta uno dei rari casi in cui un codice non strutturato è più "elegante" e leggibile di un codice strutturato (v. riquadro a lato). Quando si trova l'elemento, infatti, si può uscire immediatamente dalla funzione mediante un'istruzione `return`, restituendo l'indice cercato. Al termine del ciclo, a cui si arriva se non è stata eseguita tale istruzione di `return` all'interno del ciclo stesso, la funzione termina restituendo -1 perché il valore cercato non è stato trovato.

Per il collaudo del programma si scriva un main che faccia uso della medesima funzione di lettura del vettore dell'esercizio "1. Istogramma verticale". La dimensione massima del vettore sia pari a 100.

Esempi:

```
#./ricerca_lineare
inserisci il numero di valori da scrivere nel vettore: 5
v[0] = 1
v[1] = 2
v[2] = 3
v[3] = 4
v[4] = 5
inserisci il valore da cercare: 4
inserisci l'indice del primo elemento del sottovettore in cui cercare: 2
inserisci l'indice dell'ultimo elemento del sottovettore in cui cercare: 4
ricerca lineare...
v[3]=4

#./ricerca_lineare
inserisci il numero di valori da scrivere nel vettore: 8
v[0] = 90
v[1] = 34
v[2] = 8
v[3] = 22
v[4] = 34
v[5] = 5
v[6] = 66
v[7] = 3
inserisci il valore da cercare: 34
inserisci l'indice del primo elemento del sottovettore in cui cercare: 0
inserisci l'indice dell'ultimo elemento del sottovettore in cui cercare: 5
ricerca lineare...
v[1]=34

#./ricerca_lineare
inserisci il numero di valori da scrivere nel vettore: 8
v[0] = 90
v[1] = 34
v[2] = 8
v[3] = 22
v[4] = 34
v[5] = 5
v[6] = 66
v[7] = 3
inserisci il valore da cercare: 34
inserisci l'indice del primo elemento del sottovettore in cui cercare: 2
inserisci l'indice dell'ultimo elemento del sottovettore in cui cercare: 7
ricerca lineare...
v[4]=34

#./ricerca_lineare
inserisci il numero di valori da scrivere nel vettore: 8
v[0] = 90
v[1] = 34
v[2] = 8
v[3] = 22
v[4] = 34
v[5] = 5
v[6] = 66
v[7] = 3
inserisci il valore da cercare: 34
inserisci l'indice del primo elemento del sottovettore in cui cercare: 5
inserisci l'indice dell'ultimo elemento del sottovettore in cui cercare: 7
ricerca lineare...
valore non trovato
```

4. Scorrimento circolare

Si scriva la funzione `void scorrimento_circolare (int v[], int dim)` che riceve come argomenti un vettore di interi e la sua dimensione e fa scorrere in avanti tutti gli elementi del vettore di una posizione, spostando l'ultimo al posto del primo. Per esempio, se il vettore di ingresso ha 5 elementi e contiene i valori { 10, 13, 9, -4, 5 }, al termine dell'esecuzione della funzione esso dovrà contenere { 5, 10, 13, 9, -4 }.

NOTA: si sviluppi un algoritmo che non richieda l'impiego di un vettore di appoggio.

Suggerimento: si copi in una variabile temporanea l'ultimo elemento del vettore, si copi poi ogni elemento del vettore nella posizione immediatamente seguente, si copi infine il valore salvato nella variabile temporanea nella prima posizione.

Per il collaudo del programma si scriva un main che faccia uso della medesima funzione di lettura del vettore dell'esercizio "1. Istogramma verticale". La dimensione massima del vettore sia pari a 100.

Esempi:

```
#./scorrimento circolare
inserisci il numero di valori da scrivere nel vettore: 5
v[0] = 0
v[1] = 18
v[2] = 0
v[3] = 19
v[4] = 5
applico lo scorrimento...
v[0] = 5
v[1] = 0
v[2] = 18
v[3] = 0
v[4] = 19
```

```
#./scorrimento circolare
inserisci il numero di valori da scrivere nel vettore: 5
v[0] = 10
v[1] = 20
v[2] = 30
v[3] = 40
v[4] = 50
v[5] = 60
v[6] = 70
v[7] = 80
v[8] = 90
v[9] = 100
v[10] = 110
v[11] = 120
v[12] = 130
v[13] = 140
v[14] = 150
v[15] = 160
v[16] = 170
v[17] = 180
v[18] = 190
v[19] = 200
applico lo scorrimento...
v[0] = 200
v[1] = 10
v[2] = 20
v[3] = 30
v[4] = 40
v[5] = 50
v[6] = 60
v[7] = 70
v[8] = 80
v[9] = 90
v[10] = 100
v[11] = 110
v[12] = 120
v[13] = 130
v[14] = 140
```

```
v[15] = 150
v[16] = 160
v[17] = 170
v[18] = 180
v[19] = 190
```

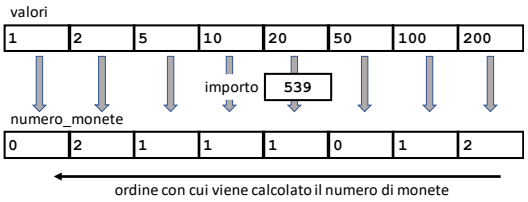
5. Calcolo del resto in monete

Un vettore contiene i valori delle monete utilizzate in una data valuta.

Per esempio, per l'euro, con i valori espressi in centesimi:

```
int valori[8] = { 1, 2, 5, 10, 20, 50, 100, 200 };
```

Si assuma che i valori in tale vettore siano già ordinati in ordine crescente.



Si deve scrivere una funzione di gestione del resto per un distributore automatico. È quindi necessario calcolare quante monete servono per ogni valore. Si assuma che il numero di monete disponibili sia sempre sufficiente per comporre l'importo richiesto.

La funzione deve avere il seguente prototipo:

```
int resto_in_monete (int resto, int n, int valori[], int numeromonete[]);
```

Essa riceve come argomenti il resto da comporre, il numero di tagli delle monete nella valuta considerata, il vettore dei valori (come prima descritto) e un vettore in cui dovrà essere scritto il numero di monete necessarie per ciascun taglio. La funzione restituisce l'eventuale resto residuo (possibile nel caso in cui il taglio minimo della valuta sia maggiore di uno).

Suggerimento: un algoritmo semplice ed efficace consiste nel ridurre l'importo residuo da comporre utilizzando i valori delle monete disponibili a partire dal massimo e passando alle monete di valore inferiore quando l'importo residuo risulta minore della moneta attualmente considerata (si veda l'esempio nel riquadro in alto a sinistra).

È interessante notare che questo algoritmo talvolta non fornisce il numero minimo di monete. Per esempio, disponendo dei valori 1, 2, 4 e 5, per comporre il valore 8 verrebbero utilizzate tre monete (5+2+1) mentre il numero minimo sarebbe due (4+4).

Per il collaudo del programma si scriva un main che chiami le funzioni di lettura del vettore dei valori delle monete e di stampa del vettore del numero di monete di resto, indicando anche i valori delle monete restituite.

Si assuma che il taglio della moneta di valore minimo sia sempre uno (quindi non ci sia mai del resto residuo) e che il numero massimo di tagli di monete sia pari a 20.

Esempi:

```
#./resto_in_monete
inserisci il numero di valori da scrivere nel vettore: 8
v[0] = 1
v[1] = 2
v[2] = 5
v[3] = 10
v[4] = 20
v[5] = 50
v[6] = 100
v[7] = 200
inserisci il valore totale del resto: 436
calcolo il resto...
1: 1
2: 0
5: 1
10: 1
20: 1
50: 0
100: 0
200: 2

#./resto_in_monete
inserisci il numero di valori da scrivere nel vettore: 5
v[0] = 1
v[1] = 5
v[2] = 9
v[3] = 13
```

```
v[4] = 47
inserisci il valore totale del resto: 567
calcolo il resto...
1: 3
5: 0
9: 0
13: 0
47: 12

#./resto_in_monete
inserisci il numero di valori da scrivere nel vettore: 8
v[0] = 1
v[1] = 2
v[2] = 4
v[3] = 8
v[4] = 16
v[5] = 32
v[6] = 64
v[7] = 128
inserisci il valore totale del resto: 1000
calcolo il resto...
1: 0
2: 0
4: 0
8: 1
16: 0
32: 1
64: 1
128: 7
```