5.6 Esercizi

* 1. Scrivere un programma di ordinamento in senso decrescente .

* 2. Scrivere un programma che carichi una matrice bidimensionale di caratteri e successivamente ricerchi al suo interno un valore passato in ingresso dall'utente. Il programma restituisce quindi il numero di linea e di colonna relativo all'elemento cercato se questo è presente nella matrice, il messaggio `Elemento non presente` altrimenti.

3. Modificare il programma per la ricerca binaria in modo che visualizzi i singoli passi effettuati (cioè mostri i dati di Figura 5.3). Sperimentare il comportamento del programma con la ricerca dell'elemento 45 nel seguente vettore:

vet	21	33	40	41	45	50	60	66	72	81	88	89	91	93	99

4. Verificare, analogamente a quanto fatto in Figura 5.1, il comportamento della prima versione di bubblesort applicata al seguente vettore:

vet	3	31	1	23	41	5	0	66	2	8	88	9	91	19	99

5. Verificare il comportamento della versione ottimizzata di bubblesort applicata al vettore del precedente esercizio. Quanti cicli interni si sono risparmiati rispetto alla prima versione?

6. Calcolare il numero di confronti effettuati dall'algoritmo di ordinamento ingenuo applicato al vettore dell'Esercizio 4 e confrontarlo con quello di bubblesort.

7. Scrivere un programma che, richiesti i valori di un vettore ordinato in modo crescente, li inverta ottenendo un vettore decrescente. Si chiede di risolvere il problema utilizzando un solo ciclo.

8. Verificare il comportamento del programma di fusione applicato ai seguenti vettori:

vet1	3	31	41	43	44	45	80

vet2	5	8	21	23	46	51	60	66

9. Modificare l'algoritmo di ricerca binaria nel caso il vettore sia ordinato in modo decrescente invece che crescente.

10. Se il vettore è ordinato la ricerca completa può essere migliorata in modo da diminuire in media il numero di confronti da effettuare: come? Modificare in questo senso il programma esaminato nel presente capitolo.

11. Scrivere un programma che, richiesti all'utente i primi $n-1$ elementi già ordinati di un vettore di dimensione n e un ulteriore elemento finale, inserisca quest'ultimo nella posizione corretta facendo *scivolare* verso il basso tutti gli elementi più grandi.

12. [*Insertion-sort*] Utilizzare l'algoritmo del precedente esercizio per scrivere un programma che ordini il vettore contemporaneamente all'inserimento dei dati da parte dell'utente.

13. Scrivere un programma che, richiesti all'utente i valori di una matrice, ne ordini tutte le colonne in senso crescente.

15. Scrivere un programma che, richiesti all'utente i valori di una matrice, ne ordini le righe in modo che il vettore i cui elementi corrispondono alla somma delle righe risulti ordinato in senso crescente.