12BHD INFORMATICA, A.A. 2018/2019

Esercitazione di Laboratorio 6

Obiettivi dell'esercitazione

- Elaborare e manipolare il contenuto di un vettore precedentemente acquisito
- Scrivere programmi che includano semplici funzioni

Contenuti tecnici

- Uso avanzato dei vettori
- Uso dei cicli annidati per l'analisi dei vettori
- Uso preliminare di funzioni di calcolo con parametri passati *by value*

Da risolvere preferibilmente in laboratorio

Esercizio 1. Si scriva un programma C che:

a. legga un vettore di N elementi interi (con N costante predefinita)
b. determini se gli elementi di tale vettore costituiscono una successione palindroma.

Suggerimento: una successione si dice palindroma se e' identica letta da sinistra verso destra o da destra verso sinistra.

Esempio: le seguenti successioni di valori sono palindrome:

```
12  3  12
1  4  5  4  1
10 10 10
```

mentre la seguente non è palindroma:

```
1 3 4 3 2
```

Esercizio 2. Si scriva un programma C che:

a. legga 2 vettori di N elementi interi (con N costante predefinita)
b. stabilisca se i due vettori contengono gli stessi elementi, anche disposti in ordine differente

Esempio: siano dati i due vettori seguenti:

```
v1 → 15  3  12  13  29
v2 → 15  29  13  3  12
```

questi contengono gli stessi valori, anche se in posizioni differenti.

Invece, i due vettori seguenti:

```
v1 → 11  3  12  18  29
v2 → 12  29  13  4  12
```

non contengono gli stessi valori.

Approfondimento: considerare la possibilità che ci siano valori ripetuti tra quelli memorizzati nei vettori. Ad esempio

```
v1 → 12  3  12  13  29
v2 → 12  29  13  3  12
```

contengono gli stessi valori ed il 12 compare 2 volte per vettore.

Invece, i due vettori seguenti:

```
v1 → 12  3  13  13  29
v2 → 12  29  13  3  12
```

non contengono gli stessi valori.

Da risolvere a casa

Esercizio 3. Si scriva un programma C che analizzi il contenuto di un vettore alla ricerca di valori replicati. Il programma dovrà in particolare:

a. Acquisire i valori del vettore da tastiera
b. Scandire il vettore stabilendo se al suo interno esistono valori ripetuti 2 o più volte.
c. Stampi l'elenco dei numeri ripetuti e il numero di occorrenze relative, verificando che ciascun numero compaia una volta sola in tale elenco.

Esercizio 4. Si scriva un programma C che legga da tastiera in un vettore di lunghezza N una sequenza di N numeri, li ordini in senso crescente man mano che vengono introdotti (ordinamento per inserimento, insertion sort) e alla fine stampi il contenuto del vettore.

Suggerimento: a prima vista, sembra che l'algoritmo da seguire sia:
- leggo un valore
- cerco la sua posizione nel vettore, tenendo conto di quanti ho inserito
- sposto tutti i successivi in avanti di una posizione (partendo dal basso)
- inserisco il valore nella sua posizione ordinata

Ad una più attenta analisi, ci si accorge che la parte che cerca la posizione in cui inserire il nuovo valore è ridondante. Si può procedere così;
- leggo un valore
- analizzo il vettore a partire dal basso: se il valore che ho introdotto è minore del dato del vettore che sto considerando, sposto quest'ultimo nella cella successiva
- itero il procedimento tornando all'indietro, finché non trovo un dato minore del valore introdotto (o sono arrivato in cima al vettore): il valore è da inserire nella cella successiva.

Esempio: ho già inserito nel vettore i dati 2, 5, 7, 9. Il valore letto sia 3. Confronto 3 con 9. 3 è minore, sposto in avanti di una posizione 9. Il vettore diventa 2, 5, 7, , 9. Confronto 3 con 7, è ancora minore, sposto il 7 in avanti di un posto. Il vettore diventa 2, 5, , 7, 9. Quando arrivo a confrontare 3 con 2, verifico che è maggiore, lo inserisco nella cella successiva (che era stata liberata al passo precedente). Formalizzare l'algoritmo e tradurlo in C.

Esercizio 5. Si scriva un programma C che legga da tastiera due numeri interi corrispondenti a base ed esponente, ed esegua il calcolo della potenza $base^{esponente}$. Il programma deve invocare una funzione chiamata *power* dal programma main, con il seguente prototipo:

int power(int base, int exponent);

Esempio: siano dati i seguenti valori
base=3
exponent=2
Il risultato di $base^{exponent}$ sarà 9. In un altro caso con
base=2

exponent=3
Il risultato di base^{esponente} sarà 8.
Suggerimento: all'interno della funzione, calcolare la potenza moltiplicando iterativamente la base per se stessa un numero di volte pari all'esponente.

Esercizio 6. (per i più curiosi, per i Talenti e per quelli che aspirano a diventarlo)

Un numero è palindromo se può essere letto da sinistra verso destra o da destra verso sinistra. Realizzare un programma che, per ogni numero intero da 2 fino ad un massimo stabilito come parametro, ad esempio 30, verifichi se esiste una base in cui quel numero è palindromo. Ad esempio, 5 è palindromo in base 2 (101_2), 16 è palindromo in base 3 (121_3), ecc. Per questo esercizio non ci si preoccupi di "rappresentare" le cifre, è sufficiente indicarne il valore (per intenderci, in base 16 la cifra di valore 10 non è richiesto che venga rappresentata come A, è sufficiente il valore 10). Inoltre si escludano i casi di 1 sola cifra (la risposta sarebbe banale): questo implica che per un generico numero N, si deve provare con base che va da 2 a N-1.
Suggerimento: dato il valore di un numero N e data una base b, per verificare se è palindromo (in quella base) bisogna convertire il numero in quella base, memorizzando le cifre che lo compongono, e poi verificare se è indifferente leggere le cifre da sinistra a destra o da destra a sinistra. Ma c'è un metodo più furbo, che non richiede la memorizzazione dell'intero numero.
Infatti: Dato il valore di un numero, l'algoritmo delle divisioni per una base permette di ricavare la sequenza di cifre che lo rappresentano (in quella base): le cifre sono calcolate a partire dalla meno significativa. Esempio: dato 5_{10}, si ricavano per la base 3 le cifre 2, 1 (infatti $5_{10} \rightarrow 12_3$)
Data la sequenza di cifre, per ricavare il valore del numero si parte dalla cifra più significativa e si applica l'algoritmo del prodotto (per la base) e somma (della cifra corrente). Esempio: $12_3 \rightarrow 1*3+2 = 5_{10}$.
Pertanto per verificare se un numero è palindromo, si possono utilizzare i due algoritmi precedenti in cascata: se il valore del numero di partenza è uguale al valore del numero ricavato dalla combinazione dei due algoritmi precedenti, si è in presenza di un palindromo. Si osservi che, sebbene l'algoritmo non richieda la memorizzazione delle cifre in un vettore, può essere conveniente memorizzare se si vuole poi visualizzare.
Il calcolo del valore delle cifre invertite sia realizzato da una funzione.

Esercizio 7. Si realizzi un algoritmo in grado di tabulare il valore della funzione arcsin(x) per x compreso nell'intervallo [a,b] con passo c. I valori di a, b, c siano forniti in input da tastiera. Non si utilizzi a tale scopo la funzione arcsin() di libreria, ma si realizzi una funzione invsin(z1,z2,k,e) in grado di calcolare numericamente col metodo di bisezione la radice dell'equazione sin(z)=k con z (restituito come valore di ritorno della funzione) compreso in [z1,z2] e con precisione pari a e, supponendo che la funzione sin() sia monotona nell'intervallo [z1,z2]. Si descriva l'algoritmo realizzato

utilizzando il linguaggio di programmazione C. L'algoritmo realizzato deve essere testato sul calcolatore in modo da verificarne la correttezza sintattica e semantica.

Esercizio 8. [1]Realizzare un programma che generi e stampi tutte le terne pitagoriche nell'intervallo degli interi (A, B e C formano una terna pitagorica se $A^2 + B^2 = C^2$). E' richiesto che il test venga effettuato da una funzione che restituisca il valore TRUE se la terna passata come parametro e' pitagorica, FALSE altrimenti. Suggerimento: attenzione all'overflow della somma!

[1] Questo esercizio sarà svolto in modo multimediale e inserito sul Portale, tra il materiale comune, nelle settimane successive.