Esercizi in preparazione alla I prova parziale di Programmazione I

a cura di Alessandro Menti

31 gennaio 2014

I seguenti esercizi in preparazione alla prima prova parziale sono proposti in ordine approssimativamente crescente di difficoltà. Ulteriori problemi possono essere trovati in H.M. DEITEL E P.J. DEITEL, *C — Corso completo di programmazione*, Apogeo.

1. *(Operazioni sui polinomi)* Si scriva un programma che richieda in input i coefficienti di due polinomi di quinto grado a e b e poi visualizzi il polinomio somma di a e b e il polinomio differenza di a e b. Si utilizzino due array per memorizzare i coefficienti.

2. *(Provvigioni di un venditore)* Un'azienda retribuisce i propri venditori con delle provvigioni come segue: ogni venditore percepisce 200 $ la settimana più il nove per cento delle vendite lorde concluse durante la settimana. Si scriva un programma che utilizzi un array di contatori per determinare quanti venditori abbiano guadagnato una retribuzione compresa in uno dei seguenti intervalli (per semplicità si tronchi la retribuzione a un numero intero):

   a) 200 $ – 299 $
   b) 300 $ – 399 $
   c) 400 $ – 499 $
   d) 500 $ – 599 $
   e) 600 $ – 699 $
   f) 700 $ – 799 $
   g) 800 $ – 899 $
   h) 900 $ – 999 $
   i) 1000 $ e oltre

3. *(Lancio di due dadi)*

   (a) Si scriva un programma che simuli il lancio di due dadi chiamando due volte la funzione `rand()` e visualizzi il valore dei dadi. Il programma dovrà poi calcolare la somma dei due valori.

   (b) Estendere il programma in modo che esso lanci i dadi 36000 volte e che dopo ogni lancio utilizzi un array opportuno per tener traccia del numero di occorrenze di ogni somma possibile. Al termine dei lanci il programma dovrà visualizzare tali occorrenze in forma tabulare. Si controlli anche se i totali sono sensati: ad esempio, dato che il 7 può uscire con probabilità teorica 1/6, circa un sesto dei lanci effettuati dovrà avere somma 7.

4. *(Sistema di prenotazione per linee aeree)* Una piccola compagnia aerea vi ha chiesto di programmare un nuovo sistema per assegnare i posti su ogni volo dell'unico aereo a disposizione (capacità: dieci posti). Il programma dovrà visualizzare il seguente menù:

   ```
   Digita 1 per prenotare un posto in prima classe
   Digita 2 per prenotare un posto nella classe economica
   ```

   Se il cliente digita 1, il programma dovrà prenotare un posto nella prima classe dell'aereo (posti da 1 a 5); analogamente, se il cliente digita 2 il programma dovrà prenotare un posto nella classe economica (posti da 6 a 10). Il programma dovrà quindi stampare una carta d'imbarco che dovrà indicare:

- il numero del posto assegnato;
- la relativa classe (prima o economica).

Si utilizzi un array per rappresentare i posti dell'aereo: esso dovrà essere inizialmente azzerato (per indicare che tutti i posti sono vuoti). A mano a mano che vengono effettuate le prenotazioni dei posti, l'elemento corrispondente dell'array dovrà essere posto a 1 (per indicare che il posto non è più disponibile).

Il programma non dovrà assegnare posti già prenotati. Nel caso in cui un cliente voglia prenotare un posto in una classe che risulta già piena, il programma dovrà richiedere se egli sia disposto ad accettare un posto nell'altra classe; nel caso in cui la risposta sia affermativa si dovrà eseguire la prenotazione, in caso contrario andrà visualizzato il messaggio `Il prossimo volo partirà fra tre ore`. Si gestisca anche il caso in cui tutti i posti di un volo risultino già assegnati.

5. Un'azienda ha quattro venditori (numerati da 1 a 4) che vendono cinque differenti prodotti (numerati da 1 a 5). Una volta al giorno, ogni venditore fornisce un tagliando per ogni tipo di prodotto venduto. Ogni tagliando contiene:

   - il numero del venditore;
   - il numero del prodotto;
   - il valore totale, in dollari, del venduto di tale prodotto.

   Di conseguenza, ogni venditore fornisce da zero a cinque tagliandi al giorno. Supponendo che siano disponibili i dati dei tagliandi dell'ultimo mese, si scriva un programma che legga tali informazioni e sommi le vendite totali per venditore e per prodotto. Le somme dovranno essere memorizzate in un opportuno array. Dopo aver eseguito tali calcoli, il programma dovrà stampare una tabella le cui colonne corrispondano ai venditori e le cui righe corrispondano ai prodotti: esso dovrà anche stampare in fondo a ogni singola riga la somma delle vendite totali del corrispondente prodotto (ossia la somma della riga) e in fondo a ogni singola colonna la somma delle vendite di ogni venditore (ossia la somma della colonna).

6. *(Codice di Vigenère)* Il codice di Vigenère è un codice che si basa sull'utilizzo di un quadrato (*quadrato di Vigenère*) per cifrare un messaggio di testo utilizzando una chiave (parola o frase segreta); se ne propone una versione semplificata che al posto delle frasi utilizza dei numeri.

   L'algoritmo è il seguente.

   1. Si crea una matrice **V** come quella qui di seguito riportata:

$$\mathbf{V} = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 0 \\ 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 0 & 1 \\ 3 & 4 & 5 & 6 & 7 & 8 & 9 & 0 & 1 & 2 \\ 4 & 5 & 6 & 7 & 8 & 9 & 0 & 1 & 2 & 3 \\ 5 & 6 & 7 & 8 & 9 & 0 & 1 & 2 & 3 & 4 \\ 6 & 7 & 8 & 9 & 0 & 1 & 2 & 3 & 4 & 5 \\ 7 & 8 & 9 & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 8 & 9 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 9 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{bmatrix}$$

   2. Si chiedono all'utente un messaggio da cifrare e una chiave.
   3. La chiave viene allineata al messaggio da cifrare e ripetuta tante volte quante sono necessarie per coprire tutto il testo. Ad esempio, se si vuole cifrare il messaggio `123456789012345` utilizzando la chiave `583051`, si opera come segue:

Messaggio da cifrare	`123456789012345`
Chiave ripetuta	`583051583051583`

4. Per cifrare la k-esima cifra del messaggio ci si posiziona sulla colonna che presenta tale cifra nella prima riga del quadrato di Vigenère e si scende fino alla riga che inizia con la k-esima cifra della chiave ripetuta. La cifra all'incrocio di tali riga e colonna è la k-esima cifra del messaggio cifrato. Nell'esempio, il messaggio cifrato è `606407262063828`.
5. Per decifrare il messaggio si applica l'algoritmo di cifratura al contrario: si scende fino alla riga che inizia con la k-esima cifra della chiave ripetuta, si scorre tale riga fino alla colonna che presenta la k-esima cifra del messaggio cifrato e si legge la cifra che compare in tale colonna e nella prima riga del quadrato.

(a) Si scriva un programma che cifri e decifri messaggi con il codice di Vigenère. Il messaggio deve essere un numero di quindici cifre, mentre la chiave deve essere un numero di sei cifre. Sia il messaggio sia la chiave dovranno essere salvati in opportuni array. Si noti inoltre che entrambi i numeri possono avere degli zeri come cifre iniziali (ad esempio, una possibile chiave di sei cifre potrebbe essere `008472`).

(b) Modificare il programma scritto in modo che esso non usi la matrice **V** per la crittazione e la decrittazione.
Suggerimento: se m_k, k_k e e_k sono la k-esima cifra, rispettivamente, del messaggio da cifrare, della chiave ripetuta e del messaggio cifrato, vale che $e_k \equiv m_k + k_k \bmod 10$.

Nota storico-crittografica: il codice venne creato da Blaise de Vigenère nel XVI secolo basandosi su idee analoghe di Leon Battista Alberti. Rispetto al codice di Cesare (sostituzione in un messaggio di ogni lettera con quella successiva di k posti nell'alfabeto) esso presentava il vantaggio di poter codificare una stessa lettera nel messaggio originale con lettere anche diverse nel messaggio cifrato, rendendo più difficile la decrittazione senza essere in possesso della chiave. L'algoritmo, divenuto noto con l'appellativo di *chiffre indéchiffrable*, fu tuttavia violato nel 1854 circa da Babbage e, indipendentemente, nel 1863 da Kasiski.

7. *(Selection sort ricorsivo decrescente)* Si scriva una funzione *ricorsiva* `selectionSort(int a[], int size)` che ordini l'array `a` usando l'ordinamento per selezione in versione decrescente, vale a dire:

   - trovando l'elemento più grande di `a`;
   - scambiando tale elemento con quello in ultima posizione all'interno di `a`;
   - ripetendo il procedimento per il sottoarray che va dall'inizio alla penultima posizione (compresa) di `a`.

   Si completi poi il programma con una funzione `main` che richieda all'utente dieci numeri, li salvi in un array, lo ordini utilizzando la funzione `selectionSort` prima definita e stampi l'array ordinato.

8. *(Le torri di Hanoi)* Vi sono tre paletti 1, 2 e 3; sul primo di questi sono impilati alcuni dischi, ordinati in modo crescente dal basso verso l'alto. Vi è stato richiesto di spostare tutti i dischi dal paletto 1 al paletto 3 secondo le seguenti regole:

   1. si può muovere esattamente un disco per mossa;
   2. un disco più grande non può mai essere impilato sopra un disco più piccolo.

   Il problema può essere risolto in modo ricorsivo come segue: per spostare n dischi dal paletto a al paletto c possiamo:

   1. muovere $n-1$ dischi dal paletto a al paletto b, utilizzando il paletto c come area di deposito temporanea;
   2. muovere l'ultimo disco dal paletto a al paletto c;
   3. muovere gli $n-1$ dischi dal paletto b al paletto c, utilizzando il paletto a come area di deposito temporanea.

   Chiaramente, se $n = 1$ è sufficiente muovere direttamente il disco.

   Si scriva un programma che risolva il problema delle torri di Hanoi utilizzando una funzione ricorsiva con quattro parametri:

1. il numero dei dischi da muovere;
2. il paletto su cui tali dischi saranno inizialmente infilati;
3. il paletto su cui tale pila di dischi dovrà essere spostata;
4. il paletto da usare come area di deposito temporanea.

Il programma dovrà chiedere all'utente il numero dei dischi inizialmente impilati sul paletto 1 e stampare a schermo tutti i movimenti dei dischi. Ad esempio, per muovere una pila di tre dischi dal paletto 1 al paletto 3 il programma dovrà stampare:

```
1 -> 3
1 -> 2
3 -> 2
1 -> 3
2 -> 1
2 -> 3
1 -> 3
```