Laboratorio di Fondamenti di Informatica Lab05_2015-10-29

Anno accademico 2015/2016

1. La matrice impazzita

Si inizializzi una matrice quadrata m[N][N] con gli interi da 1 a N^2 in modo che la prima riga contenga i numeri da 1 a N, la seconda da N a 2N, etc. Si definisca N tramite #define

Si stampi a video la matrice e si chieda all'utente di scegliere una cella della matrice inserendo i numeri si riga e colonna r e c. Sia K il numero contenuto in m[r][c]. Si modifichi quindi la matrice come descritto in seguito, e la si stampi nuovamente:

- La cella selezionata resti invariata;
- Tutte le celle della colonna c diverse da quella selezionata (in verde in figura) assumano valore 2K
- A tutte le celle "precedenti" (cioè su righe precedenti a r, oppure sulla stessa riga ma in colonne precedenti a c) che non siano nella colonna c venga aggiunto K (beige in figura);
- A tutte le celle "successive" (righe successive, oppure stessa riga ma colonne successive) che non siano sulla colonna c venga sottratto K (rosa in figura).

Ad esempio, per N = 5:

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

14	15	26	17	18
19	20	26	22	23
24	25	13	1	2
3	4	26	6	7
8	9	26	11	12

2. Anagrammi

Si leggano da stdin **due parole**, e si verifichi se sono una l'anagramma dell'altra.

Si badi a verificare bene le occorrenze delle lettere:

PENTOLA e POLENTA sono anagrammi, ma TONNO e TONTO non lo sono, anche se sono formate dalle stesse lettere (T, O, N), hanno pari lunghezza, e… in un certo senso sono anche sinonimi :)

3. Anagrammi a frase

Una nuova amica: la funzione **gets(…)** legge da stdin e copia nella stringa passatale come parametro una sequenza di caratteri (spazi inclusi) fino a quando trova il '\n', che nella stringa è sostituito con il terminatore '\0'. Risulta molto più comoda della scanf() per "leggere" intere frasi che possono includere spazi. Esempio di uso:

```
char stringa[50];
gets( stringa );
```

Si scriva un programma che legge **due frasi** da stdin e decide se sono anagrammi, considerando solo i caratteri alfanumerici e ignorando tutto il resto (punteggiatura, spazi, altri segni diacritici, e il case delle lettere). Esempi:

```
Halloween?
hello, Wane!
Le frasi sono anagrammi

Come si studia l'informatica?
Combinando creativita' e rigore
Le frasi NON sono anagrammi

E come si codificano gli "algoritmi"?
Cacciando fogli, e stimoli... o emigri!
Le frasi sono anagrammi
```

```
L'asma bronchiale
Ballo in maschera
Le frasi sono anagrammi

Una pentola di polenta
Tante padelle, non una!
Le frasi NON sono anagrammi
```

4. Cambi di base

- Scrivere un programma che legge da stdin un numero espresso (in una forma **n1**) in una base di partenza **b1** compresa tra 2 e 10, e lo converte in una forma **n2** che esprime il numero in una base di destinazione **b2**, pure compresa tra 2 e 10.
- L'utente dovrà
  - specificare le due basi (b1 e b2)
  - immettere l'espressione del numero da convertire (cioè la forma n1 del numero in base b1)
- Il programma dovrà
  - Verificare che la forma n1 sia valida (cioè contenga solo cifre tra 0 e b1-1)
  - Visualizzare il numero convertito (cioè espresso nella forma n2, in base b2, con cifre tra 0 e b2-1)

Suggerimenti *(rispecchiano **un** modo di procedere, non l'unico):*

- Utilizzare un primo array (n1) per memorizzare le **cifre** della forma n1 e un altro array (n2) per le **cifre** via via calcolate per la forma n2, prima di visualizzarle
- Calcolare il **valore** del numero moltiplicando le potenze di b1 per le cifre di n1 (da prendere nell'ordine giusto!), e usare poi su tale valore il metodo dei resti per generare le cifre di n2 (da visualizzare nell'ordine giusto!)

Estensione: *si considerino sistemi di numerazione in base b1 e b2 compresi tra 2 e un generico N, e si utilizzi per entrambe le codifiche un alfabeto arbitrario, da rappresentarsi a sua volta come un vettore di caratteri (volendo essere "tradizionalisti", le prime 36 cifre del vettore possono essere proprio 1,2,3…9,A,B,…,Z, ma si permetta di specificare I due alfabeti come stringhe arbitrarie. Con N>36, peraltro, occorre utilizzare comunque anche altri simboli).*

5. Vettore Ciclico

Si consideri un array v di **N** interi (come ad esempio v1 e v2 in calce). Si vuole "esplorare" il vettore, iniziando dal primo elemento v[0] e muovendosi con la regola descritta in seguito, per stabilire se l'esplorazione porta ad un percorso "infinito", perché il vettore è *ciclico*, oppure ad un certo punto l'esplorazione termina.

Come ci si muove nel vettore: il valore contenuto in ogni elemento, se compreso tra 0 e N-1, rappresenta l'indice del prossimo elemento da visitare; se invece è esterno a tale intervallo rappresenta un riferimento "errato", che indica una posizione esterna al vettore, e comporta la fine del processo di esplorazione.

Nell'esempio $\mathbf{v_1}$ è ciclico, $\mathbf{v_2}$ non lo è.