12BHD INFORMATICA, A.A. 2016/2017

Esercitazione di Laboratorio 9

Obiettivi dell'esercitazione

- Realizzare programmi con dati complessi, uso dei file

Contenuti tecnici

- Uso di matrici di interi e caratteri
- Uso di insiemi di vettori "paralleli"
- Lettura di file di tipo testo

Da risolvere preferibilmente in laboratorio

Esercizio 1. Una matrice di caratteri rappresenta in forma schematizzata una palude. La palude è costituita da zone di fango, rappresentate dal carattere '.', e da zone pietrose, indicate dal carattere '*'. Le dimensioni della matrice possono essere fissate a piacere, mediante dei *#define*, comunque non superiori a 25 righe e 80 colonne.

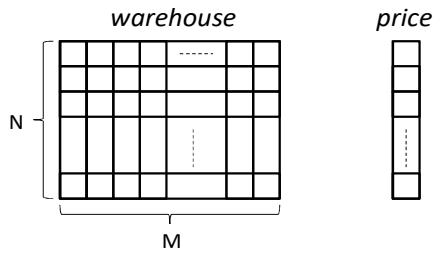
Esempio di palude:

```
**.*.*....*
..*.*...**.
*...*.*....
.*.*.*.*.*.
..*.*...*.*
```

Realizzare un programma che cerchi nella palude un percorso da sinistra a destra, senza salti, costituito tutto da zone pietrose adiacenti. Si ipotizzi che, adiacente a destra di un punto pietroso, ci possa essere al più un altro punto pietroso (non ci sono diramazioni), sulla stessa riga, sulla riga in alto o sulla riga in basso. Il programma deve visualizzare la sequenza righe in cui ci sono le pietre del percorso trovato (le colonne sono implicite, ci deve essere una pietra per ogni colonna), oppure avvertire che non esiste un percorso. Per la prima versione del programma si utilizzi una matrice di stringhe predefinita. Come approfondimento, la palude viene introdotta da tastiera, e nella versione finale, si legge la palude da file.

Esercizio 2. Si scriva un programma in grado di gestire un listino prezzi, ovvero un programma con cui sia possibile gestire un elenco di prodotti e i loro relativi prezzi in €. Il programma utilizza una matrice di caratteri chiamata *warehouse* di dimensione NxM per memorizzare i nomi dei prodotti (massimo N prodotti) e un vettore di numeri decimali chiamato *price* di dimensione N usato per memorizzare i prezzi dei prodotti (il prezzo presente nell'*i*-esima cella di *price* corrisponde al prezzo del prodotto il cui

nome è memorizzato nell'*i*-esima riga di *warehouse*). Di seguito la visualizzazione logica dei dati definiti.



Nel vettore *price*, il valore -2 indica linea libera (ossia nessun prodotto nella riga corrispondente in *warehouse*), mentre un valore positivo indica che la linea corrispondente in *warehouse* contiene un prodotto valido a cui è stato associato un prezzo. All'avvio del programma, il vettore *price* è totalmente inizializzato a -2 (ossia il listino è vuoto/non contiene prodotti).

Il programma propone all'utente un elenco di operazioni possibili sottoforma di menu. L'utente decide quale operazione eseguire selezionando il numero associato all'operazione di suo interesse.
Le operazioni possibili sono:
1) Inserimento di un nuovo prodotto e relativo prezzo
2) Stampa listino attuale (elenco dei prodotti con i relativi prezzi)
3) Uscita dal programma

Le prime due operazioni devono essere realizzate tramite l'invocazione delle seguenti due funzioni di cui si forniscono prototipo e funzionalità:

- *insert_product*: è una funzione che permette di inserire un nuovo prodotto e il suo prezzo nel listino (il nome del prodotto deve essere di max M-1 caratteri). La funzione restituisce il valore 1 se il prodotto non era ancora presente nel listino e il suo inserimento nel listino è avvenuto con successo, 0 se già presente (ossia prodotto già inserito in precedenza), 2 se il listino è pieno (e quindi non risulta possibile inserire il nuovo prodotto). Se il prodotto non era ancora presente nel listino la funzione deve inserire il suo nome nella prima cella libera presente in *warehouse* e il prezzo a lui associato nella cella corrispondente di *price.*

int insert_product(char warehouse[][M], float price[], int n, char new_product[], float price_new_product);

Il parametro *n* corrisponde al numero di righe della matrice *warehouse*, che corrisponde anche al numero di celle del vettore *price* (in fase di invocazione nel nostro caso passeremo il valore N).

- *print_all*: è una funzione che permette di visualizzare a video il contenuto del listino (elenco prodotti e relativi prezzi). Inoltre, la funzione restituisce due valori (tramite due parametri passati per indirizzo): il prezzo medio ed il prezzo massimo dei prodotti presenti nel listino. Visualizzare a video i due valori restituiti.

*void print_all(char warehouse[][M], float price[], int n, float *avg, float *max);*

Avvertenza: l'esercizio suggerisce di utilizzare il vettore *price* contemporaneamente in due modi: per contenere il prezzo della merce, come flag di posizione vuota. Tenerne conto quando si cerca il nome di un prodotto, quando si effettua la stampa, etc.

Da risolvere a casa

Esercizio 3. Si scriva un programma che legga da un file, il cui nome è introdotto da tastiera, alcune informazioni ferroviarie. Per ciascuna linea, il file contiene le seguenti informazioni (ciascuno dei campi non superi i 20 caratteri di lunghezza e sia privo di spazi)

<stazione_partenza> <ora_partenza> <stazione_arrivo> <ora_arrivo>

Il programma riceve poi da tastiera il nome di una città: il programma calcoli e stampi il numero di treni in arrivo ed il numero di treni in partenza da tale città (se inclusa nell'elenco).

Esercizio 4. Estendere il programma realizzato come Esercizio 2 aggiungendo due ulteriori funzioni:
- 4) Aggiornamento prezzo prodotto (*update_product*)
- 5) Rimozione prodotto (*remove_product*)

Le due operazioni sono realizzate tramite l'invocazione delle seguenti funzioni:

a. *update_product*: è una funzione che permette di aggiornare il prezzo di uno specifico prodotto. La funzione riceve il nome del prodotto da aggiornare e il suo nuovo prezzo. La funzione restituisce 1 se l'aggiornamento è avvenuto con successo, 0 se il prodotto non esiste nel listino.

int update_product(char warehouse[][M], float price[], int n, char product[], int new_price);

b. *remove_product*: è una funzione che permette di rimuovere un prodotto dal listino; restituisce 1 se la rimozione è avvenuta con

successo, 0 se il prodotto non esiste. La funzione deve impostare il valore -2 nel vettore *price* in corrispondenza dell'elemento rimosso.

int remove_product(char warehouse[][M], float price[], int n, char old_product[]);

Esercizio 5. Si realizzi un programma in grado di simulare il movimento di un topolino.
Si dispone di una matrice Q, di dimensione NxN, che descrive le caratteristiche di una certa area: in ciascuna locazione Q(x,y) della matrice vi e' la quota del quadratino di superficie posto alle coordinate x,y. A partire da una posizione iniziale del topolino, x0,y0, si stampino le coordinate di tutti i quadratini toccati dal topolino se esso segue le seguenti regole di movimento:

- Il topolino si sposta ad ogni passo di un solo quadratino, nelle 8 direzioni possibili
- Il topolino sceglie il quadratino su cui muoversi determinando il quadratino di quota massima tra gli 8 adiacenti
- Se tutti i quadratini adiacenti sono ad una quota inferiore o uguale rispetto al quadratino attuale, il topolino si ferma.

Esempio:
Supponendo di avere la seguente matrice Q di dimensione 10x10, e di aver scelto come punto iniziale il punto (3,7), allora il topolino effettuerà i seguenti spostamenti: (3,7) (4,8) (5,8) (6,8) (7,9). Il punto un cui il topolino si ferma è quindi il punto (7,9).

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
2	0.1	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3	0.1
3	0.1	0.2	0.3	0.3	0.4	0.5	0.5	0.5	0.5	0.1
4	0.1	0.2	0.3	0.3	0.5	0.5	0.5	0.7	0.7	0.1
5	0.1	0.2	0.4	0.4	0.5	0.7	0.7	0.8	0.7	0.1
6	0.1	0.2	0.4	0.4	0.5	0.7	0.8	0.9	0.8	0.1
7	0.1	0.2	0.4	0.4	0.5	0.7	0.8	0.9	1.4	0.1
8	0.1	0.2	0.4	0.4	0.5	0.7	0.8	0.9	1.2	0.1
9	0.1	0.2	0.4	0.4	0.5	0.7	0.8	0.9	1.1	0.1
10	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1

In fase di test dell'algoritmo risolutivo, i su indicati valori siano assegnati come valori iniziali della matrice mappa, nella definizione.
Successivamente i valori della matrice siano letti da un file, il cui nome sia introdotto da tastiera.
Nota: la prima riga e la prima colonna non fanno parte della matrice, sono riportati per comodità come numerazione delle colonne e delle righe.

Esercizio 6. Si realizzi un programma che permetta di introdurre da tastiera una matrice di interi. Le dimensioni della matrice, *M x N*, possono essere fissate a priori con dei *#define*, ma sarebbe preferibile che fosse il programma a determinare automaticamente il numero di righe e il numero di colonne della matrice introdotta, entro i limiti massimi fissati dal programma (vedi nota). Il programma deve azzerare la sottomatrice di dimensioni m x n, con m<*M*, n<*N*, a partire dall'elemento r, c (riga, colonna). I valori di m, n, c ed r siano richiesti da tastiera. Il programma visualizzi la matrice prima e dopo l'azzeramento. Il programma deve controllare di non andare oltre i limiti della matrice.

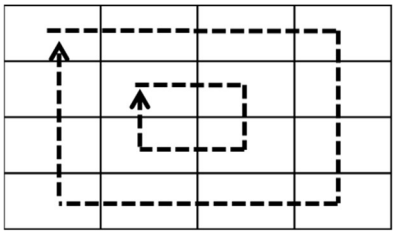
Approfondimento: generalizzare il programma realizzando l'operazione mediante una funzione che effettui la sostituzione di valore, il cui prototipo sia:

```
void Set (int m[][N], int m, int n, int i, int j, int val)
```

dove *val* è il valore da sostituire.

Nota: un modo semplice per rilevare le dimensioni di una matrice introdotta da tastiera è il seguente: si pone riga e colonna a zero. Si legge fino ad EOF (<CONTROL>+Z) un intero e il carattere successivo. Si memorizza il valore letto nella posizione (riga, colonna) e si incrementa colonna. Si testa poi il carattere letto: se è new-line (è finita la riga), si azzera colonna e si incrementa riga. Alla fine del ciclo riga dice quante righe si sono lette, invece colonna deve essere memorizzata prima dell'azzeramento. Servono ovviamente i controlli per non superare il numero di righe e colonne effettive della matrice.

Esercizio 7. Si scriva un programma in grado di riempire una matrice quadrata di dimensioni NxN di interi (con N pari e maggiore o uguale a 4 definito come costante tramite la direttiva *define*) secondo lo schema delle cornici concentriche; per ogni cornice si parta dalla cella in alto a sinistra e si riempia la cornice progressivamente con numeri crescenti a partire da 1.



1	2	3	4
12	1	2	5
11	4	3	6
10	9	8	7

Si proceda infine con la stampa della matrice assicurandosi che colonne e righe siano correttamente allineate.