Laboratorio di algoritmi e strutture dati

Docente: Violetta Lonati

Dati aggregati - esercizi da svolgere in laboratorio*

Gli ultimi 3 esercizi proposti in questa scheda sono tratti, con qualche variazione, da temi d'esame di appelli passati.

1 Istogramma

Scrivete un programma che legga una sequenza di caratteri terminata da un punto e che visualizzi un istogramma con una barra per ogni carattere dell'alfabeto, lunga quanto il numero delle sue occorrenze. Il programma non deve visualizzare le barre delle lettere che non compaiono e non deve fare distinzione fra maiuscole e minuscole (a tal fine potete usare le funzioni dichiarate nel file `ctype.h`). Ad esempio, se il programma riceve da standard input la frase

```
C'era un RAGAZZO che come ME amava i Beatles e I rolling StoneS.
```

il programma deve stampare

```
A *******
B *
C ***
E ********
G **
H *
I ***
L ***
M ***
N ***
O ****
R ***
S ***
T **
U *
V *
Z **
```

Prima di scrivere il vostro programma, fatevi queste domande:

- E' necessario memorizzare l'intera frase di input?
- Cosa serve memorizzare e quando?

Dopo aver scritto il vostro programma, fatevi queste domande:

- Quante sono le variabili nel mio programma? (Non saranno troppe?)
- Quanto è lo spazio di memoria occupato? Quanto cresce al crescere dell'input?
- Stimate il numero di operazioni che svolge il vostro programma.

Sulla base delle risposte che avete date, valutate se rivedere il vostro programma ...

*Ultima modifica 14 ottobre 2019

2 Anagrammi

Due parole costituiscono un *anagramma* se l'una si ottiene dall'altra permutando le lettere (es: attore, teatro). Scrivete un programma che legga due parole e verifichi se sono anagrammi.

Suggerimento: sfruttate il programma scritto per l'esercizio precedente!

3 Cifre ripetute di un numero

Scrivete un programma che legga in input un numero intero n usando `scanf( "%d", &n )` e stabilisca se n contiene cifre ripetute e in caso affermativo quali.

Esempio di funzionamento:

```
3124241
Cifre ripetute: 1, 2, 4
```

Esempio di funzionamento:

```
312
Non ci sono cifre ripetute
```

Anche in questo caso, fatevi le domande proposte per l'esercizio "Istogramma".

4 Date

Scrivete un programma che legga una sequenza di al massimo 100 date, nella forma `dd/mm/yyyy`, terminata dalla data `00/00/0000`, che non è considerata parte della sequenza, e da un ultima data `DD/MM/YYYY`. Il programma deve stampare solo le date precedenti alla data `DD/MM/YYYY`.

Memorizzate le date in un array di strutture, ciascuna con tre membri chiamati giorno, mese, anno.s

Suggerimento: per stampare un intero con esattamente due cifre (eventualmente preceduto da zeri), dovete usare la specifica di formato `%02d`.

5 Ordinamento per inserimento

Scrivete una programma che legga da standard input una sequenza di interi distinti terminati da 0 (potete assumere che la sequenza sia formata al più 100 numeri), memorizzandoli in un vettore ordinato: ogni volta che viene letto un nuovo intero, il vettore viene scorso fino a trovare l'esatta collocazione del numero, quindi si crea lo spazio per il nuovo numero spostando in avanti i numeri successivi già memorizzati.

6 Passeggiate aleatorie

Scrivete un programma che generi una "passeggiata aleatoria" (in inglese *random walk* in un array bidimensionale di dimensione 10×10. L'array sarà riempito di caratteri (inizialmente da punti). Il programma dovrà muoversi di elemento in elemento spostandosi ogni volta di un passo in direzione su, giù, destra o sinistra. Gli elementi visitati andranno etichettati con le lettere dalla A alla Z, nell'ordine in cui vengono visitati. E' importante controllare ad ogni passo che la passeggiata non esca dall'array e che non ritorni su posizioni già visitate. Quando si verifica una di queste condizioni, provate in altre direzioni. Se tutte e quattro le direzioni sono bloccate, il programma deve uscire.

Esempio di funzionamento completo

```
A . . . . . . . . .
B C D . . . . . . .
. . E . . . . . . .
. . F . . . . . . .
I H G . . . . . . .
J . . . . S T . . .
K N O P Q R U . . .
L M . . Z Y V . . .
. . . . . X W . . .
. . . . . . . . . .
```

Esempio di uscita prematura dal programma

```
A . . . . . . . . .
B C D . . . . . . .
K L E . . . . . . .
J M F . . . . . . .
I H G . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
```

Suggerimenti

- Per generare a caso una direzione potete usare le funzioni`time` (da `time.h`), `rand` e `srand` (da `stdlib.h`). La chiamata funzione `rand()` produce un numero apparentemente casuale, ma generato in realtà a partire da un seme. La funzione `srand(n)` inizializza il seme; se il seme non viene inizializzato, il suo valore di default è 1. La chiamata della funzione `time(NULL)` restituisce data e ora corrente, codificate come un unico intero; con la chiamata `srand(time(NULL))` è possibile differenziare i semi e quindi garantire che la passeggiata sia diversa ad ogni esecuzione del programma.
- Dopo aver generato un numero con `rand()`, considerate il suo resto modulo 4. I quattro possibili valori (0,1,2,3) possono essere usati per indicare le direzioni (rappresentatele con una variabile di tipo enum!).

7 Trova il valore mancante - Appello del 3 luglio 2019

Scrivere un programma che legga un intero n seguito da una sequenza di n interi ordinati (tutti gli interi appartenenti all'insieme $\{0,1,2,\ldots,n\}$, tranne uno), quindi stampi il valore mancante nella sequenza. Ad esempio, su input

```
7
0 1 2 3 4 6 7
```

il programma deve stampare

```
5
```

Nota: Prima di scrivere il programma pensate a 3 esempi significativi (e significativamente diversi) su cui testare il programma.

Oltre alla correttezza della funzione verrà valutata anche l'efficienza dell'algoritmo usato. Una soluzione che impiega tempo lineare è poco interessante.

8 Riordina i bit - Appello del 13 giugno 2019

Si consideri un array `A`, non vuoto, di n interi, in cui ciascun valore può essere esclusivamente 0 oppure 1. I valori sono presenti senza alcun ordine; è anche possibile che tutti i valori siano uguali.

Si scriva un algoritmo lineare che sposti tutti i valori 0 prima di tutti i valori 1. Verrà assegnato punteggio pieno ad algoritmi che scambiano elementi in `A`, che richiedono memoria ulteriore $O(1)$ (quindi che non sfruttano array ausiliari), e che non sono basati sul conteggio del numero di 0 e 1 presenti.

Si scriva un programma che implementa l'algoritmo. Il programma deve leggere da standard input un intero n seguito da una sequenza di n valori 0/1, riordinare l'array e stamparlo. Ad esempio, ricevendo da standard input

```
7
0 1 1 0 0 0 1
```

il programma deve stampare

```
0 0 0 0 1 1 1
```

9 Schiacciatina

Scrivete un programma che legga due interi *r*, *c*, seguiti da una matrice di *r* righe e *c* colonne contenente lettere maiuscole e asterischi, e che stampi in output la matrice che si ottiene da quella in input *schiacciando verso il basso* le lettere e *facendo galleggiare* gli asterischi. Ad esempio, se la matrice è data da

```
V * S
* * B
K * *
* S *
```

il programma dovrà stampare la matrice seguente:

```
* * *
* * *
V * S
K S B
```

Potete assumere che *r* e *c* siano al massimo pari a 100.