Esercizi Blocco 3

Luca Oliveri
luca.olivieri-1@unitn.it

Università di Trento — November 19, 2020

Introduzione

Focus principale su manipolazione di stringhe. Usare quando possibile le funzioni della `cstring` e `ctype`. Essendo che alcuni di questi esercizi richiedono di essere risolti dividendo il problema principale in più problemi semplici e puntuali, dedicare tempo ad organizzare il codice in modo adeguato.

3.1

SUP che prende una stringa del tipo `"923D"` ed estrae il numero, convertendolo in `int`. La lettera D è sempre alla fine della stringa. Verificare il funzionamento anche con numeri negativi. Esistono diversi modi per fare questa conversione, ma probabilmente la soluzione più semplice è usare il metodo nativo C++, che è la prima soluzione proposta dal sito.

3.2

SUP che prende una stringa dall'utente che può essere anche una frase, comprensiva quindi di spazi e punteggiatura. Il programma ristampa la stessa stringa al contrario.

3.3

Localizzare in una stringa tutte le occorrenze di un determinato carattere. Le stringhe in ingresso sono composte di sole lettere minuscole e spazi. Il programma stampa la stessa stringa in ingresso con le occorrenze trovate convertite a lettere maiuscole. Stampa inoltre un conteggio delle occorrenze.

Info: Ci sono diverse funzioni di libreria che possono essere utili in questo esercizio. Consultare l'elenco completo e scegliere quella ritenuta più adatta.

3.4

SUP esegue *parsing* di stringa e svolge una funzionalità programmabile. Nello specifico il programma prende in input una stringa di valori di temperatura del tipo `"20C@34F@12F@23C"` e converte questi in gradi Kelvin. Stampare due tabelle distinte per conversioni Celsius e Fahrenheit.

Info: Come nell'esercizio precedente, diverse funzioni di libreria possono aiutare a dividere la stringa in più parti. In questo la scelta più adatta è `strtok`.

3.5

SUP che data una stringa in input calcola le occorrenze di ogni carattere. Risolvere il problema usando un unico array per contare le occorrenze. Stampare il risultato in tabella, con possibili interessanti statistiche a vostra discrezione. Considerare solo lettere dell'alfabeto, maiuscole e minuscole fanno parte dello stesso conteggio. Scartare tutto ciò che non è una lettera, per semplicità si scartano anche i caratteri della tabella ASCII estesa come le lettere accentate.

Info: Tenere a mente come sono rappresentate le lettere nella tabella ASCII. È possibile indicizzare l'array dove incrementiamo mano a mano il conteggio usando direttamente la lettera sotto esame e applicando un offset. Ad esempio la lettera D, quarta nell'alfabeto quindi avente indice 3, ha valore ASCII 44, oppure anche 64 nel nostro caso. Sottraendo un offset di 41 o di 61 rispettivamente, troviamo il nostro indice di valore 3.

Estensione avanzata: Invece che prendere in input una stringa dell'utente, prendere in input il nome di un file di testo *plain-text*, in altre parole un file `.txt`. È possibile trovare online interi testi in formato `.txt`, come ad esempio I Promessi Sposi, oppure anche La Divina Commedia. La manipolazione di file in C++ usa lo stesso paradigma di uso della console. Trovate qui quello che vi serve specificatamente per risolvere questo esercizio, in particolare basatevi sull'esempio.

3.6

SUP che simula il comportamento di due dadi. I dadi possono avere numero di facce variabile, ma per verificare la correttezza del programma cominciare con dadi a sei facce. Il programma 'tira' i due dadi usando la funzione `rand`, opportunamente inizializzata. La somma dei valori di ogni lancio è salvata in array. Il programma deve lanciare i dadi decine di migliaia di volte, numero specificato dall'utente, e stampare la probabilità di ogni valore sotto forma di tabella. Verificare che somma 7 è la più probabile con probabilità di un sesto, essendo che ci sono 6 lanci diversi che restituiscono somma 7. Indagare poi con dadi aventi numero di facce diverse da sei.

Info: Il programma può simulare qualsiasi tipo di dado, ma in realtà solo alcuni poligono possono richiudersi in un solido. Qua una lista dei possibili dadi, dai più classici ai più esoterici.

Esercizi `CodeStepByStep`

- isPermutation
- longestUniqueString
- compressString