Ricorsione e iterazione

1 (9 ottobre)

Si vuole calcolare il fattoriale di un qualsiasi numero intero (positivo) sfruttando una funzione ricorsiva.
Riscrivere poi la sua versione iterativa (quale delle due è più efficiente?).

Bonus:
Usare la libreria `chrono` (`#include <chrono>`) per misurare quanto ogni funzione impiega. Per ottenere un tempo attuale è possibile usare:

```
chrono::time_point<chrono::steady_clock> start, end;
chrono::duration<double> time_span;

start = chrono::steady_clock::now(); // inizia a contare
// chiamata alla funzione
end = chrono::steady_clock::now();
time_span = chrono::duration_cast<chrono::duration<double>>(end - start); // durat
```

Prendendo il tempo prima e dopo la funzione che si vuole misurare (`start` e `end`) è possibile sottrarli e ottenere la durata.

2 (9 ottobre)

Scrivere una funzione ricorsiva che calcoli la somma tra due numeri interi **positivi**.

Bonus: calcolare anche prodotto e potenza (ad esempio con `int potenzaRicorsiva(int base, int esponente)`).

3

Creare una funzione che riceve un numero e restituisce la somma delle cifre del numero solo quando questa è minore di 10. Deve altrimenti restituire il risultato della funzione applicata alla somma delle cifre del numero.
Ad esempio:

1. riceve 15, esegue `f(15) = 1 + 5 = 6`, restituisce 6
2. riceve 392, esegue `f(392) = 3 + 9 + 2 = 14 > 10`, `f(14) = 1 + 4 = 5`, restituisce 5

(*suggerimento: usare una funzione d'appoggio ricorsiva che controlli la somma*)

4

Data una media iniziale e un numero di crediti totali acquisiti, far inserire all'utente il voto e i crediti di un nuovo esame e scrivere una funzione per ricalcolare la nuova media pesata.

Bonus: Scrivere una funzione con passaggio per valore, una con passaggio per riferimento e una con passaggio per puntatore. Provare ad eseguire ognuna delle funzioni e capire cosa succede.

5

Dati in input tre interi positivi in tre variabili (n1, n2, n3), scrivere una funzione che con una procedura "ri-ordini" i numeri in ordine crescente.

Bonus: Riscrivere la funzione usando il passaggio di parametri per riferimento.

6

Dato in input un numero `n`, in maniera ricorsiva chiedere all'utente di inserire una lettera per `n` volte e, sapendo che le lettere in maiuscolo valgono 10 mentre quelle in minuscolo valgono 5, calcolare e stampare a video la somma totale. È possibile usare la funzione `islower(char)` contenuta nella libreria `cctype` per controllare che il carattere in input sia minuscolo.

7

Dato un array inizializzato in modo casuale, ordinarlo usando *StalinSort*, descritto nella seguente procedura:

- Confrontare i due valori iniziali nell'array;
- Se il primo è maggiore del secondo, eliminalo (settando a 0).
- Ripeti per tutti i valori

8

FoldLeft è un'operazione molto usata nella programmazione funzionale. Solitamente prende come parametro un array, un valore iniziale e lo step a cui siamo nell'array. La funzione viene applicata a tutti gli elementi dell'array, partendo dal primo, e il risultato viene passato come parametro alla funzione per il prossimo elemento.

Ad esempio, se abbiamo un array di interi e vogliamo calcolare la somma di tutti gli elementi, possiamo usare la funzione `foldLeft` in questo modo:

```
int foldLeft(array, init) {
    val = get_valore_from_array(array);
    return foldLeft(array, init + val);
};
```

Scrivere una funzione `foldLeft` che prenda in input un array, un valore iniziale e l'indice corrente, e restituisca il massimo valore dell'array.

NB: alcuni step sono mancanti nella funzione precedente, è necessario aggiungerli per farla funzionare.

Hint: è necessario aggiungere un caso base per la ricorsione.