Fondamenti di Programmazione 2

Laboratorio

November 11, 2022

Esercizio 1

Scrivere un programma che presa in input una sequenza di n interi $x_1, ..., x_n$ e un intero m individui, qualora esistessero, due elementi x_i, x_j tali che $x_i + x_j = m$. I due elementi devono essere distinti, cioè $i \neq j$, ma è possibile che $x_i = x_j$.

Suggerimento: La soluzione naive ha complessità $O(n^2)$, ma utilizzando un'hashmap esiste una soluzione in $O(n)$.

Esercizio 2

Implementare una classe `Doublemap` che permetta di gestire coppie (x, y) in modo "bidirezionale" - accedendo a y dato x (come in una hashmap tradizionale) ma anche ad x dato y. Si può assumere, per semplicità, che le coppie gestite dalla classe non andranno mai modificate o rimosse, ma solamente aggiunte. **L'implementazione dovrà fare uso dei template per gestire in modo generico i tipi delle chiavi e dei valori, supportando (almeno) i tipi primitivi e `std::string`.**

Per utilizzare una classe arbitraria `K` come chiave in una `unordered_map<K,V>` richiede di definire una funzione di hash per la classe `K` - per semplicità si può assumere `Doublemap` utilizzi solo tipi primitivi e classi per cui la funzione di hash è già definita.

Esercizio 3

Definire una classe `Persona` che abbia i seguenti campi privati:

- `string nome`
- `string cognome`

Implementare un `main` che implementi le funzionalità di una rubrica telefonica. Per semplicità, si assuma ogni persona possa avere un unico numero di telefono

e che la rubrica non possa contenere più contatti con lo stesso nome e cognome. **Ciò significa che la concatenazione di nome e cognome di una persona è univoca.**

In particolare, dovrà essere possibile:

- Inserire un nuovo contatto
- Cercare un contatto specificando nome e cognome della persona
- Eliminare un contatto specificando nome e cognome della persona
- Modificare il numero di un contatto
- Modificare nome e cognome di un contatto
- Stampare la rubrica telefonica
- Svuotare la rubrica telefonica
- Terminare l'esecuzione del programma

Per implementare le funzionalità della rubrica telefonica, implementare tutti i metodi di `Persona` necessari e/o ritenuti opportuni.

Suggerimento: Utilizzare un'hashmap che mappi la concatenazione di nome e cognome a una `std::string` per rappresentare il numero di telefono.

Esercizio 4

Un *tweet* è una stringa lunga al più 280 caratteri - per semplicità, supponiamo questi caratteri possano essere solo lettere latine minuscole, spazi e il carattere `#`. In particolare, una sequenza di lettere minuscoli priva di spazi che inizia con un unico `#` è detta *hashtag*.

Scrivere un programma che presa in input una sequenza di n tweet:

- Stampi la frequenza di ogni *hashtag* presente nei tweet in input, in ordine decrescente di frequenza
- Preso in input un hashtag, restituisca la lista di tweet che lo contengono

Suggerimento:

- Per prendere in input stringhe contenenti spazi, è utile la funzione `getline`:

```
#include<string>
#include<iostream>
using namespace std;
int main() {
    string x;
    getline(cin, x);
    cout << x << endl;
    return 0;
}
```

- Per identificare gli hashtag in un tweet, possono tornare utili i metodi `substr` e `find` di `std::string`