Fondamenti di Programmazione - parte A: Esercitazione di laboratorio 2

Esercizio 1:

Scrivere un programma che definisca due classi: Money e CreditCard. La classe Money deve contenere due dati membri privati interi (euros e cents). La classe Money deve anche contenere:

- Costruttore che inizializza a 0 entrambi i dati membri.
- metodi pubblici per impostare e leggere il valori dei dati membri: get_euros(), set_euros(int e), get_cents(), set_cents(int c)
- Un operatore '+' definito come funzione non-membro: `Money operator+(Money m1, Money m2)` che esegue la somma di due oggetti Money sommando euro e centesimi (i centesimi se eccedono il valore di 100 vanno convertiti in euro: es 10.50 euro+ 5.70 euro= 16.20 euro)
- Un operatore '<<' definito come funzione non-membro: `ostream& operator<<(ostream& os, Money m)` per stampare a video i dati membri euros e cents

La classe CreditCard contiene tre dati membri privati: il nome del proprietario (string), il numero della carta di credito (string), il totale dei pagamenti effettuati (Money). La classe CreditCard contiene anche:

- Costruttore per creare una carta dato il nome del proprietario e il numero della carta.
- Una funzione membro: print() per stampare il nome del proprietario e il numero
- Una funzione membro: `Money get_totalcharges()` che restituisce il totale dei pagamenti
- Una funzione membro: `charge(string item, int e, int c)` che aggiorna il totale dei pagamenti a seguito di una singola spesa (con causale "item") di euro 'e' e di centesimi 'c'

Il programma principale crea un oggetto di tipo CreditCard, legge da un file di testo un elenco di spese e aggiorna il totale dei pagamenti. Il file di testo delle spese contiene per ogni riga le informazioni di una singola spesa su tre colonne: <causale della spesa> <euro> <centesimi>
es:
book 90 60
pizza 20 50

Esercizio 2:

Scrivere una funzione template: `template <typename T> T most_common(T *A, int size)`

che accetti in ingresso un array di elementi di tipo generico T e la sua dimensione "size" e fornisca in uscita l'elemento dell'array ripetuto più volte. Se vi sono due o più elementi con lo stesso numero di ripetizioni massimo la funzione restituisce il primo elemento trovato.

Esercizio 3:

Scrivere una classe template `template<class T> class Pair` i cui oggetti rappresentino coppie di elementi ("first" e "second") dello stesso tipo generico T. La classe oltre ai costruttori deve contenere i seguenti metodi pubblici:

- `void set_element(int position, T value); // imposta value in posizione 1 o 2`
- `T get_element(int position) const; // restituisce l'elemento in posizione 1 o 2`
- `void add_up(const Pair<T>& the_pair); // somma gli elementi corrispondenti di un oggetto Pair con quelli contenuti nell'argomento the_pair`

Esercizio 4:

Scrivere una classe template `template <class T> class Matrix2x2` per rappresentare array a due dimensioni generici di dimensione 2x2 (matrici). La classe oltre ai costruttori deve contenere un metodo pubblico per sommare ad un oggetto `Matrix2x2` una matrice passata come argomento

```
Matrix2x2<T> Add(Matrix2x2 x);
```

(la somma di due matrici è la matrice i cui elementi sono la somma di elementi in posizione corrispondente)