Programmazione ad Oggetti - parte A: Esercitazione di laboratorio 2

Esercizio 1:

Scrivere un programma che definisca una classe Libro. La classe deve contenere:

- 2 dati membro privati: titolo (stringa), pagine (numero int)
- Un costruttore che consenta di inizializzare i dati membro, e che contenga anche dei valori di inizializzazione di default (stringa vuota per titolo, 0 per numero di pagine)
- Metodi "set" e "get" per leggere e modificare ciascun dato membro

Scrivere nel "main" un codice di test che crei alcuni oggetti di tipo Libro, modifichi alcuni dati membro e stampi i dati membro di ciascun oggetto.

Esercizio 2:

Scrivere un programma che definisca una classe Persona. La classe deve contenere:

- 3 dati membro privati: nome (stringa), cognome (stringa), anni (numero intero)
- Un costruttore che consenta di inizializzare i dati membro
- Un metodo pubblico "`string getInformation();`" per stampare le informazioni di una persona nel formato: <prima lettera del nome>. <cognome>, <anni>
  (esempio: "M. Rossi, 37"). Per estrarre il primo carattere di una variabile string s, in formato string, è possibile utilizzare il metodo s.substr(0, 1)
- Un metodo pubblico "`void setcognome(string c)`" per modificare il cognome di una persona con un valore passato come argomento

Creare nel "main" tre oggetti Persona con valori a scelta e stampare le informazioni di ciascuna persona utilizzando il metodo "`getInformation()`". Creare un array di 3 persone e salvare nell'array i tre oggetti creati precedentemente. Scrivere un algoritmo (utilizzando un ciclo "for") per modificare il cognome di tutte le persone contenute nell'array con il valore "Bianchi" e stampare le informazioni di ciascuna persona dopo la modifica del cognome.

Esercizio 3:

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 (oggetto della classe 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 4:

Modificare la classe Persona dell'esercizio 2 aggiungendo un operatore binario "!="
`bool operator != (Persona p2)`
come funzione membro, per determinare se due oggetti Persona sono diversi. Due oggetti Persona sono diversi se almeno uno dei tre dati membro è diverso (nome, cognome, anni). Creare due oggetti Persona nel "main" e creare un file "output.txt" (aperto in scrittura). Se i due oggetti Persona sono diversi scrivere sul file la stringa "DIVERSI", altrimenti scrivere nel file la stringa "UGUALI".