UNIVERSITÀDELLACALABRIA

Facoltà di Scienze Matematiche Fisiche e Naturali
Corso di Laurea in Informatica
Prova di Programmazione ad Oggetti
Data: 18 luglio 2007

Esercizio n. 1

In una applicazione in cui si richiede di organizzare le informazioni relative ad un centro di controllo dei taxi in una grande città si fa uso delle classi: Taxi e VettoreTaxi. La prima classe consente di memorizzare le informazioni relative ad un taxi, mentre la seconda implementa un array dinamico di oggetti di tipo Taxi. Seguono le intestazioni delle classi:

```
class Taxi
{
friend ostreamOperator<<(ostream&, const Taxi&);
private:
    int codice;
    MyString nomeGuidatore;
    bool occupato;
    MyString posizioneCorrente;
public:
    Taxi (int c, MyString n, bool o, MyString pos);
    Taxi (const Taxi&);
    bool operator==(const Taxi&);
    int getCodice();
    MyString getNomeGuidatore();
    bool occupato();
    MyString getPosizione ();
    void setCodice(int);
    void setNomeGuidatore(MyString);
    void setOccupato(bool);
    void setPosizioneCorrente (MyString);
};
class VettoreTaxi
{
friend ostreamOperator<<(ostream&, const VettoreTaxi&);
private:
    Taxi* elenco;
    int lunghezza;
public:
    VettoreTaxi(int = 0);
    VettoreTaxi(const VettoreTaxi&);
    Taxi & operator[](int);
    VettoreTaxi & operator=(const VettoreTaxi&);
    int getLunghezza();
    ~VettoreTaxi();
};
```

Si implementino **solo** i metodi di seguito elencati, supponendo già implementati i rimanenti.

- Costruttore di copia classe Taxi
- Metodo occupato() della classe Taxi
- Operatore << della classe Taxi
- Operatore= della classe VettoreTaxi
- Costruttore della classe VettoreTaxi
- Distruttore classe Taxi

Nota: E' possibile aggiungere, qualora lo studente lo ritenga necessario, ulteriori membri pubblici o privati alle classi. Nel caso in cui venga aggiunta una nuova funzione deve esserne fornita l'implementazione.

Esercizio 2

Siano date le classi Cliente e Biglietto, che rappresentano i dati dei visitatori di un Teatro, le cui intestazioni sono di seguito riportate:

class Cliente
{
public:
  int codiceCliente;
  MyString Nome;
};
	class Biglietto
	{
	public:
	  MyString nomeEvento;
	  int codiceCliente;
	  int prezzo;
	};

class ListaCliente; class IteratoreCliente;	class ListaBiglietto; class IteratoreBiglietto;

Per ogni cliente si memorizzano il nome e un codice (quest'ultimo identifica univocamente un cliente all'interno del sistema), mentre per ogni biglietto acquistato da un cliente si memorizza il prezzo pagato ed il nome dell'evento cui assiste il cliente.
Si suppone, inoltre, che esistano le classi ListaCliente, IteratoreCliente, ListaBiglietto e IteratoreBiglietto, definite come di consueto, che consentono di rappresentare e manipolare liste di oggetti di tipo Cliente e Biglietto. **Utilizzando gli iteratori** scrivere la funzione:

- ```
  ListaCliente clientiAbitualiGold(const ListaCliente& lclienti,
                                   const ListaBiglietto& lbiglietti);
  ```

che, data una lista di clienti (*lclienti*) ed una lista di bliglietti da essi acquistati (*lbiglietti*), restituisce la lista dei *clienti abituali gold*. Un cliente è da considerarsi abituale se ha acquistato almeno 100 biglietti e se spende in media almeno 100 euro.
Nota: La media della spesa di un cliente si calcola sommando i prezzi dei biglietti acquistati da un cliente diviso il numero di biglietti acquistati; ad esempio, se Gianni ha comprato tre biglietti che costano 20, 21 e 22 euro, allora la sua media di spesa è (20+21+22)/3 = 21 euro.

Esercizio 3

Data la classe Cliente descritta nell'esercizio 1, si utilizzi l'ereditarietà per progettarne una estensione dal nome ClienteAbbonato. La nuova classe deve essere in grado di rappresentare/manipolare, oltre alle informazioni già presenti nella classe Cliente, ulteriori informazioni specifiche degli abbonati al teatro, quali il codice dell'abbonamento (da rappresentare con un intero), l'elenco degli eventi coperti dall'abbonamento (da rappresentare con una lista di MyString) e l'anno di validità dell'abbonamento (da rappresentare con un intero).
È richiesta l'implementazione dell'interfaccia della classe ClienteAbbonato e di un costruttore.
  ```