Forndamenti di programmazione 2
Traccia Laboratorio 2 novembre 2022

Esercizio 1.
Scrivere un programma C++ che gestisca la fila di clienti da un parrucchiere.
Ogni cliente possiede un nome, un trattamento che specifica il trattamento che vuole eseguire, una durataTrattamento che specifica la durata del trattamento in minuti e un prezzo che indica il prezzo del trattamento.
Supponendo che il parrucchiere inizi a lavorare alle 08:00 e che tutte le operazioni avvengano istantaneamente, si implementi un main che consenta le seguenti operazioni:

- Inserimento di un nuovo cliente in fila;
- Estrazione di un cliente dalla fila. Una volta estratto un cliente, si suppone che il suo trattamento venga immediatamente eseguito;
- Richiesta del prossimo orario disponibile per un eventuale nuovo cliente;
- Stampa di tutti i clienti attualmente in fila;
- Stampa dell'orario attuale (partendo dalle 08:00 si aggiungano tutte le durate dei trattamenti gi`a eseguiti);
- Stampa dell'incasso attuale;
- Terminazione del main.

Esercizio 2.
Realizzare la classe Utente che contiene almeno i seguenti campi:

- int numero;
- int tipologiaOperazione;

dove è possibile avere solo tre tipologie di operazioni: 0 per spedizione, 1 per pagamento, 2 per riscossione. La scelta dei metodi da inserire nella classe Utente è libera.

Realizzare la classe Posta che permetta di gestire i clienti in coda in un ufficio postale. La classe deve
contenere al suo interno un campo di tipo list<Utente>.

a) void aggiungiUtente(int tipologiaOperazione); aggiunge un utente in lista di attesa. Il numero dell'utente è progressivo rispetto al numero di utenti in attesa per quella determinata operazione
b) Utente prossimoUtente(); restituisce il prossimo utente e lo rimuove dalla lista di attesa. La precedenza tra gli utenti è stabilita come segue:
    - Prima tutti gli utenti che devono effettuare una spedizione in ordine di arrivo.
    - Se non ci sono utenti che devono effettuare spedizioni, si alternano gli utenti che devono effettuare pagamenti e riscossioni.

- Nel caso in cui tutti gli utenti rimasti in attesa debbano effettuare la stessa operazione, si servono in base all'ordine di arrivo.

c) void stampaUtentiInCoda() const; stampa tutti gli utenti in ordine di priorità.
d) void stampaUtenti(int tipologiaOperazione) const; stampa tutti gli utenti in attesa per una determinata tipologia di operazione.
e) void reset(); rimuove tutti gli utenti.

Esempio. Si immagini il seguente ordine di arrivo:

1. Utente con spedizione
2. Utente con pagamento
3. Utente con pagamento
4. Utente con spedizione
5. Utente con riscossione
6. Utente con riscossione
7. Utente con riscossione

Gli utenti sono i seguenti:

1. Id: 1, tipo 0
2. Id: 1, tipo 1
3. Id: 2, tipo 1
4. Id: 2, tipo 0
5. Id: 1, tipo 2
6. Id: 2, tipo 2
7. Id: 3, tipo 2

Gli utenti devono essere serviti nel seguente ordine:
1, 4, 2, 5, 3, 6, 7

Infine, realizzare un main di prova che permetta l'interazione con un utente utilizzando un'interfaccia testuale. La schermata iniziale deve contenere la seguente interfaccia:

==== Menu ====
Premi 1 per aggiungere un utente specificando la tipologia di operazione
Premi 2 per selezionare il prossimo utente da servire
Premi 3 per stampare gli utenti in coda
Premi 4 per stampare gli utenti in coda per una determinata operazione
Premi 5 per effettuare un reset
Premi 9 per uscire

Esercizio 3.

Realizzare la classe Film che contenga i seguenti campi privati: string titolo; int anno; int incasso; string regista; Genere genere, dove Genere può essere un enumerativo con i seguenti valori: NONDEFINITO, ANIMAZIONE, COMICO, COMMEDIA, DRAMMATICO, FANTASY, HORROR, STORICO.

Realizzare la classe Cinema contenente un campo privato list<Film*> e i seguenti metodi:

- void aggiungi(Film*); aggiunge un elemento all'interno della lista.
- Genere migliorGenere() const; restituisce il genere di film che ha incassato di più tra quelli presenti nella lista (escludendo i film per cui il genere è NONDEFINITO). Se due generi diversi hanno incassato lo stesso importo massimo restituire il primo in ordine alfabetico (esempio, ANIMAZIONE precede COMICO). Se non sono presenti film o se tutti i film hanno il genere NONDEFINITO, restituire NONDEFINITO.
- string registaStanco() const; restituisce il regista meno attivo di recente. In particolare, si consideri A come l'anno più recente di un film nella lista. Restituire il regista che ha diretto meno film nell'anno A. Se due o più registi hanno diretto lo stesso numero minimo di film nell'anno A, restituire il primo in ordine alfabetico. Se non sono presenti film, restituire "-1".
- int registiSettoriali() const; restituisce il numero di registi settoriali. Un regista è detto settoriale se almeno il 70% dei suoi film sono dello stesso genere (includendo il genere NONDEFINITO). Se non sono presenti film, restituire -1. Esempio: Un regista che ha diretto due film di ANIMAZIONE e un film COMICO non è settoriale (67% dei film di ANIMAZIONE e 33% COMICO). Un regista che ha diretto tre film di tipo COMICO e un film COMMEDIA è settoriale (75% dei film di COMICO e 25% COMMEDIA).
- int differenzaIncassoMaggiore() const; restituisce la differenza di incasso più alta tra i film presenti nella lista. Data una qualsiasi coppia di film F1, F2 la differenza di incasso tra i due film è l'incasso di F1 - l'incasso di F2. Se sono presenti meno di 2 film nella lista, restituire -1.