The Ticketing App

Creați o aplicație ce permite **gestionarea biletelor emise** pentru diferite evenimente (film, **piesă de teatru**, meci de fotbal, etc.)

Aplicația va avea următoarele funcționalități:

* **va permite citirea (prin constructor)** caracteristicilor locației (ex: **număr maxim de locuri**, **număr de rânduri**, **zone**, **număr**/codificare **scaune pe rând**, etc.)
* **va permite citirea (prin constructor)** caracteristicilor evenimentului (ex: **dată**, **oră**, **denumire**, etc.)
* permite generarea de **bilete nominale conform caracteristicilor dorite** (ex: VIP, peluză, tribună, loje, etc.)
* **biletele generate vor avea un id unic ce va putea fi verificat ulterior** (id-ul va fi **generat aleator** astfel încât să nu existe id-uri duplicate pentru același eveniment și nici posibilitatea ca participanții să ghicească id-urile biletelor)
* **aplicația va pune la dispoziția utilizatorului un meniu unde va putea introduce caracteristicile locației și ale evenimentului, va putea genera și ulterior valida biletele emise**
* aplicația trebuie să trateze **orice fel de input** și **va afișa mesaje de eroare** dacă acesta este **invalid**
* **aplicația nu va genera erori de execuție pentru niciun fel de input**
* **NU pot fi folosite biblioteci externe (third party) pentru citirea sau validarea datelor**
* **soluția trebuie implementată utilizând doar biblioteci standard din C++ (precum iostream, string, fstream, etc) utilizate în cadrul cursurilor și laboratoarelor. Clasele STL nu sunt acceptate decât atunci când sunt cerute în mod explicit (doar în cadrul fazei 2)**

Exemple de scenarii de luat în considerare:

* Emitere de bilete de fotbal într-un stadion dotat cu o peluză și două tribune (tribuna 1 și tribuna 2). Fiecare zonă are rând și loc. Biletele pentru un anumit meci se vor genera în limita locurilor disponibile. Atunci când un utilizator prezintă biletul se va verifica id-ul unic al acestuia pentru a vedea dacă este valid.
* Emitere de bilete la film din categoriile Normal sau VIP. Fiecare zonă are rând și loc. Biletele se vor genera în limita locurilor disponibile. Atunci când un utilizator prezintă biletul se va verifica id-ul unic al acestuia pentru a vedea dacă este valid.
* Emitere de bilete la teatru din categoriile **Categoria 1**, **Categoria 2** și **Lojă**. Fiecare zonă are rând și loc. **Biletele se vor genera în limita locurilor disponibile**. Atunci când un utilizator prezintă biletul se va verifica id-ul unic al acestuia pentru a vedea dacă este valid.

Faza 1:

* Deadline: 18.12.2022 23:59
* **Minim 3 clase** ce au legătură cu proiectul
* Clasele trebuie să conțină (per total, nu fiecare) **cel puțin un vector alocat dinamic de caractere**, **un vector numeric alocat dinamic**, **un câmp constant**, **un câmp static** și **o metodă statică**. Membrii trebuie să aibă legătură cu clasa.
* **Toate atributele vor fi definite în zona privată a clasei**
* **Toate atributele vor avea metode de acces**; **setterii** vor conține validări
* Fiecare clasa va conține cel puțin 2 metode generice (altele decât constructorii sau metodele de acces) ce vor fi utilizate pentru a realiza diverse prelucrări
* **Fiecare clasă va conține cel puțin un constructor implicit și unul cu parametri (biletCategoria1, biletCategoria2 și biletLoja au, dar poate fi îmbunătățit).** Clasele cu membri pointeri vor respecta **„regula celor 3”.**
* Fiecare clasă va conține supraîncărcări pentru **operatorii << și >>**
* Fiecare clasă va avea **cel puțin 2 operatori** supraîncărcați **dintre următorii** (același operator nu poate fi supraîncărcat în 2 clase diferite):
* **operator de indexare []**
* **operatori aritmetici (+,-,\* or /)**
* **++ or -- (ambele forme)**
* **cast implicit sau explicit**
* **operator de negație !**
* **operatori relaționali (<.>,=<,>=,==)**
* **Clasele vor fi implementate în propriul fișier header și/sau cpp**
* **Funcția main va fi localizată într-un alt fișier cpp**
* Această fază este considerată implementată dacă cel puțin 75% dintre cerințe sunt implementate

Faza 2:

* Deadline: 15.01.2023 23:59
* Programul poate primi ca parametru în linia de comandă modalitate de funcționare: dacă utilizează un meniu în consolă sau prelucrează un fișier text cu toate datele de intrare (exemplu: dacă aplicația se numește oop.exe, atunci poate fi lansată în execuție în felul următor: *oop.exe date.txt*, situație în care va prelucra datele din fișierul primit ca parametru)
* Aplicația poate procesa date dintr-un fișier text și va afișa rezultatele la consolă conform celor mai sus menționate
* Aplicația va utiliza fișiere binare pentru a salva biletele emise până în prezent; la repornirea aplicației datele vor fi restaurate
* Toate entitățile necesare pentru a implementa funcționalitățile precedente vor fi adăugate în clasele existente (sau se pot definit clase noi ce au legătură cu proiectul)
* Un meniu va fi implementat pentru a ajuta utilizatorul să navigheze printre diferitele funcționalități
* Cel puțin o nouă clasă va fi creată prin derivarea unei clase existente
* Aplicația va conține cel puțin o clasă abstractă (poate fi și interfață) ce conține cel puțin o metodă virtuală pură. Metoda pură va fi supradefinită într-o clasă derivată.
* Clasele existente vor avea cel puțin 2 metode virtuale, altele decât cele pur virtuale. Metodele vor fi supradefinite de clasele derivate.
* Cel puțin o clasă STL va fi folosită într-un din clasele existente
* Aplicația trebuie să funcționeze la cheie (odată pornită nu va necesita alte modificări la nivelul codului sursă pentru a expune toate funcționalitățile)
* Această fază este considerată implementată dacă cel puțin 75% dintre cerințe sunt implementate

BONUS:

* Biletele vor fi generate sub forma de PDF ce poate fi trimis pe email participantului
* În acest caz se permite utilizarea de biblioteci third party pentru a crea fișierul PDF

Barem Faza 1

* **1 punct** - respectarea cerințelor de încărcare și de structurare ale proiectului
* **3 puncte** - fiecare clasă corect implementată astfel:
* **1 punct** - implementarea corectă a clasei (conform cerințelor - atribute, constructori, getteri, setteri, regula celor 3)
* **1 punct** - metodele generice ce au legătură cu clasa
* **1 punct** - operatorii (de afișare și de calcul)
* **-5 puncte** - clasele nu sunt funcționale și nu pot rezolva nici măcar unul dintre scenariile prezentate

\* clasele ce nu au legătură cu proiectul nu vor fi luate în considerare

Barem Faza 2

* **1 punct** - Respectarea cerințelor de încărcare și structurare ale proiectului
* **0.5 puncte** - Programul poate primi fișierul ca parametru în linia de comandă
* **1 punct**- Aplicația poate procesa datele dintr-un fișier text
* **1 punct**- Utilizarea de fișiere binare pentru a salva și restaura datele
* **1 punct**- Existența unui meniu funcțional conform cerințelor
* **1 punct**- Existența metodelor virtuale și pur virtuale
* **0.5 puncte** - Utilizarea unui container STL
* **4 puncte** - Tratarea corectă a celor 3 scenarii propuse (1 punct pentru fiecare) + 1 scenariu primit la prezentarea proiectului

\* Clasele și/sau metodele ce nu au legătură cu proiectul nu vor fi luate în considerare