Proiecte Modern C++ 2022-2023

Reguli generale

1. Cerințele proiectelor sunt distribuite în funcție de specializarea pe care o urmati astfel:
   1.1. Informatica aplicata: proiect Platforma recomandare filme
   1.2. Informatica teoretica: proiect Triviador
2. Fiecare echipa este coordonată de mentorul care trebuie adăugat pe Github ca membru de echipa utilizând următoarele nume de utilizator:
   - Vlad Vrabie: vladvrabie
   - Iulian Popa: iulian-unitbv
3. **Durata proiectului**:**7 noiembrie -> 15 ianuarie** (perioada 19 decembrie - 30 decembrie formează o săptămână de commituri, astfel că aveți timp 2 săptămâni pentru realizarea numărului de commituri).
4. **Durata totală a proiectului: 8 săptămâni.**
5. **Commituri:**
   - **minim 12 commituri cu SENS/săptămână/echipă de 4 persoane SAU minim 9 commituri cu SENS/săptămână/echipă de mai puțin de 4 persoane SAU minim 15 commituri cu SENS/săptămână/echipă de mai mult de 4 persoane**
   - **Minim 24 commituri cu SENS/persoană la deadline la prezentare proiect**
6. **Este necesară 1 "prezentare" intermediară (întrebări punctuale, erori, prezentare componentă, componenta preferată etc.)**
   6.1. **Deadline prima prezentare: 24 decembrie**
7. **A se ține cont de fișierul de CLEAN CODE TIPS atașat pe platformă**
8. **Proiect parte basic**: pentru rezolvarea cerințelor din baremul minimal se poate obține maximum nota 8
9. **Proiect parte advanced**: basic + **1 parte advanced**(care valorează 2 puncte)
10. Fiecare echipă o să pregătească pentru prezentarea finală proiectului un **video**. Aceasta conține explicații pe cod și funcționalitatea aplicației, punand accent pe elementele de barem – vezi demo pe platformă, **Video Demo Proiect. Acest video va fi commis pe github pana la data de 15 ianuarie, ora 23:59.**
11. **Biblioteci/framework-uri suplimentare acceptate**: SDL, SFML, Qt(pentru oricare altă bibliotecă trebuie să se obțină acceptul coordonatorului de proiect)
12. La predarea proiectului soluția VS completă va trebui să fie **clonabilă** de pe repo și **compilabilă** pe toate configurațiile existente (se vor șterge configurațiile care nu sunt suportate). A se încerca pe un calculator pe care nu s-a făcut dezvoltare.
13. **Echivalarea examenului scris se poate realiza dacă este îndeplinită cel puțin una dintre următoarele cerințe:**
    13.1. Implementarea a 2 componente avansate
    13.2. Tratarea excepțiilor și implementarea unit testelor(se vor utiliza framework-uri specifice alese doar cu acordul mentorilor), code coverage 50%

Barem corectare

Prezentarea elementelor de Modern C++ se va realiza la evaluarea proiectului. Trebuie justificată prezența (sau absența) următoarelor elemente:

- const ref
- move semantics
- structuri de date moderne - Containers library
- algoritmi moderni - Algorithms library
- smart pointers - Dynamic memory management
- template
- Regex
- DLL - implementarea unui DLL, altul decât cel prezentat la laborator
- Studierea profilului și performanței aplicației

Triviador

Jocurile de inteligență și strategie sunt populare în rândul copiilor, dar și al adulților care se mandresc cu o vastă cultură generală. Un astfel de tip de joc este Triviador, iar scopul acestui proiect este de a implementa propria noastra versiune a acestui joc. Pentru implementarea aplicației trebuie să respectăm regulile jocului:

1. Acțiunea jocului se desfășoară în cadrul unei hărți și este asemenea unei bătălii pentru teritoriu din cadrul unui război, obiectivul fiind cel de cucerire a unei suprafețe cat mai mari prin luptă
2. “Lupta” între 2 jucători se realizeaza prin intermediul întrebărilor de cultura generala, iar întrebările pot fi 2 tipuri:
    a. întrebări de tip grila cu un singur răspuns corect: acest tip de întrebări decide castigatorul doar dacă unul dintre jucătorii răspunde corect
    b. întrebări cu răspuns numeric: aceste întrebări decid castigatorul, acesta fiind cel care a oferit cel mai apropiat răspuns de cel corect(în cazul în care toți jucătorii au oferit același răspuns castigator va fi cel care a răspuns primul)
3. Fiecare jucător activ al jocului deține:
    a. o regiune numită “bază” - cucerirea bazei unei jucator reprezinta pierderea tuturor regiunilor detinute de acesta
    b. mai multe(sau niciuna) regiuni teritoriale

    Jucătorul devine inactiv în momentul în care pierde regiunea “bază”
4. Fiecărei regiuni îi este atribuit un anumit scor, reprezentand importanța acesteia
5. Jocul este alcatuit din 4 etape:
    a. Etapa alegerii bazei: prin intermediul unei intrebari cu raspuns numeric va fi stabilită ordinea în care jucătorii își vor alege locul în care va fi construită bază următorului joc. Scorul atribuit acestei regiuni este 300
    b. Etapa împărțirii regiunilor: prin intermediul unei intrebari cu raspuns numeric va fi stabilită ordinea și numărul de regiunii pe care le va putea selecta fiecare jucător(jucătorul de poziția n în clasament va putea selecta n-1 regiuni care îi vor intra în posesie, n-fiind numărul de jucători). Această etapă se încheie în momentul în care toate regiunile din hartă sunt atribuite unui jucător, scorul acestora fiind 100 de unități.
    c. Duelul: se va desfășura pe mai mai multe runde(numărul acestora este egal cu dublul numărului de jucători). În cadrul rundelor fiecare jucător(în ordine aleatoare) va încerca să cucerească o regiune vecină(2 regiuni sunt vecine doar dacă au cel puțin o granița comuna). Cei doi jucători se vor lupta iar dacă:
        i. posesorul regiunii castiga lupta, atunci scorul regiunii va fi incrementat cu 100 de unități
        ii. cel care a pornit duelul castiga, atunci regiunea intra în posesia acestuia cu scorul 100 doar dacă scopul acesteia era 100 inainte de duel, astfel regiunea va ramane posesia adversarului, dar scorul va fi decrementat cu 100 de unități

    d. Stabilirea câștigătorilor: însumând scorurile tuturor regiunilor deținute de jucători după încheierea rundelor de joc
6. Avantaje: în cadrul luptelor, jucătorii pot aplica cel mult o data pe durata unui joc fiecare dintre următoarele avantaje:
    a. 50-50: avantaj care va elimina 2 răspunsuri dintre cele 4 ale întrebărilor de tip grila
    b. alegere răspuns: avantaj care oferă utilizatorului 4 răspunsuri din care poate alege în cazul întrebărilor cu răspuns numeric
    c. sugerare răspuns: avantaj care sugerează utilizatorului răspunsul corect(sau o valoare apropiata cu acesta) în cazul întrebărilor cu răspuns numeric

    Costul fiecărui avantaj este egal cu 100 de unități. Acest cost va fi plătit din scorul unei regiuni a cărei valoare este de minimum 200 de unități, regiunea care va plăti prețul va fi aleasă de jucător după încheierea luptei în care a folosit avantajul.

Cerințe de bază

➡ Retelistica: implementarea aplicației respectand arhitectura client-server(aplicația trebuie sa asigure posibilitatea creării a minim 2 instanțe de client + 1 aplicație server care vor comunica prin rețea).
➡ Pagină de Login/Register: la pornire, unui utilizator i se oferă posibilitatea de a se loga în contul său sau își poate crea un cont. Logarea/înregistrarea presupune introducerea numelui de utilizator. Atenție: numele de utilizator trebuie să fie unic!
➡ Pagină/Fereastra jocului: este fereastra în care va fi desenată harta(în varianta basic va fi afișată în consolă orice reprezentare a acesteia) și desfășurarea jocului. Dimensiunea hartii este dependentă de numărul de jucători:

- 2 jucatori → harta de 3x3, 5 runde
- 3 jucatori → harta de 5x3, 4 runde
- 4 jucatori → harta de 6x4, 4 runde

Dimensiunea tablei de joc și numărul de runde pot fi modificate! A se urmări o implementare cat mai generică. Deasemenea, dimensiunea tablei de joc poate avea și altă formă(diferită de cea dreptunghiulară)
➡ Pagina de profil a utilizatorului: Aici un utilizator poate vizualiza un istoric al meciurilor jucate

Componente avansate ale proiectului

➡ GUI - Să se implementeze o interfață grafică pentru cerințele de bază folosind framework-ul Qt.
➡ Bază de date: Pentru a vă organiza datele, puteți să le stocați într-o bază de date. Se va folosi biblioteca de SQLite SQLite ORM (NU ALTA). Se poate instala într-un proiect de Visual Studio folosind Microsoft vcpkg (vedeți aici și la curs).

Platformă de recomandat filme

În perioada recentă s-a raportat o creștere a utilizatorilor abonați la platformele de seriale și filme online. Mulți dintre ei doresc să țină o evidență a filmelor urmărite și doresc să primească recomandări de filme în baza istoricului și a preferințelor lor. Să se implementeze o aplicație în care utilizatorii să își organizeze filmele văzute/nevăzute și să primească recomandări, conform cerințelor de mai jos.

Cerințe de bază

➡ Bază de date: Pentru a vă organiza datele, trebuie să le stocați într-o bază de date. Se va folosi biblioteca de SQLite SQLite ORM (NU ALTA). Se poate instala într-un proiect de Visual Studio folosind Microsoft vcpkg (vedeți aici și la curs).
➡ Pagină de Login/Register: la pornirea aplicației, un utilizator se va putea loga în aplicație folosind username-ul; parola este opțională; În cazul în care utilizatorul nu are cont, își poate crea unul prin introducerea unui username.

Hint: Puteți să faceți procesul de creare de cont mai folositor pentru recomandările voastre viitoare. Puteți să îl întrebați pe utilizator ce genuri de filme urmărește sau să aleagă câteva filme care îi plac sau **oricare alte informații utile unui sistem de recomandare complet.**
➡ Pagină/Funcționalitate de search: utilizatorul poate căuta un film după titlul său; Se va afișa o pagină de rezultate (0/1/N) și utilizatorul poate să meargă pe pagina unui film.
➡ Pagina unui film: Se vor afișa detaliile pe care le aveți la dispoziție legate de acel film (titlu, actori, anul apariției, genul etc.), încărcarea acestor date se va realiza doar la cererea utilizatorului pe baza unor filtre. Deasemenea vor fi posibile și următoarele funcționalități:

- Marcarea filmului ca văzut
- Adăugarea filmului în wishlist
- Notarea filmului (like/dislike, 5 stele, de la 1 la 10 etc.)

Sub detaliile filmului se va afișa o secțiune de “Recomandări de filme similare” prin care se vor putea accesa paginile altor filme.
➡ Pagina de profil a utilizatorului: Aici un utilizator își poate vizualiza lista de filme văzute, wishlist-ul etc.
➡ Pagina de recomandări: O pagină cu recomandări de filme pentru utilizator în baza tuturor preferințelor sale.

Pentru implementarea acestor funcționalități, o să aveți nevoie de un set de date de filme. Folosiți unul dintre următoarele seturi de date: Netflix TV Shows and Movies sau MovieLens 20M Dataset. Preprocesați datele într-un format ușor și **eficient** de folosit pentru aplicația voastră.

Funcționalitatea de search: O variantă simplă de implementare este ca un search care conține N cuvinte să returneze toate titlurile de filme care conțin toate cuvintele din querry (apoi cele care conțin n-1 cuvinte etc.). Desigur, puteți să adăugați și alte criterii sau să folosiți alți algoritmi consacrați :)

Cum se fac recomandări de filme?

Pentru cerințele de bază, puteți să vă gândiți la câțiva algoritmi simpli de selecție a filmelor în baza datelor oferite de seturile de date. De exemplu: filme din același gen; filme cu actori comuni, sequeluri, prequeluri etc. filme de la genul care are cele mai mari note de la utilizator etc.

Componente avansate ale proiectului

➡ GUI - Să se implementeze o interfață grafică pentru cerințele de bază folosind unul dintre framework-urile enumerate la inceputul documentului.
➡ Recomandări de filme folosind Machine Learning: Integrați algoritmii de ML prezentați la curs în funcționalitatea voastră de a genera recomandări