Tekst projektnog zadatka

U programskom jeziku C++ realizovati simulaciju platforme za učenje u vidu konzolne aplikacije. Korisnik sistema ima pristupne podatke za sistem (korisničko ime i lozinka). Korisnik može da bude upisan na više kurseva. Na svakom kursu svaki upisani korisnik može biti ili student ili predavač na tom kursu. Student može da se prijavi na kurs i da ga pohađa. Prijava mora biti odobrena od strane predavača na kursu. Svaki korisnik može da bude sprijateljen sa drugim korisnicima. Korisnik može da drugom korisniku pošalje zahtjev za prijateljstvo, koji drugi korisnik može da prihvati ili da odbije. Ako je zahtjev prihvaćen, smatra se da se ta dva korisnika poznaju. Korisnik može da razmjenjuje poruke sa korisnicima sa kojima je sprijateljen u sistemu. Takođe, svaki student može da razmjenjuje poruke sa predavačima koji su zaduženi na kursevima koje taj student pohađa. Poruke se čuvaju u inboksu korisnika. Na primjer, moguće je da se funkcionalnost za razmjenu poruka realizuje tako da se može demonstrirati sljedećom sekvencom akcija: 1) aplikacija se pokreće, 2) korisnik A se prijavljuje, 3) korisnik A šalje poruku korisniku B i odjavljuje se sa sistema i 4) korisnik B se prijavljuje i u inboksu vidi poruku od korisnika A.Administrator sistema može dodati, modifikovati ili ukloniti kurseve i korisnike. Omogućena je i perzistencija podataka u sistemu, tako da je stanje sistema očuvano ako se aplikacija zatvori. Nakon otvaranja aplikacije, serijalizovani podaci se učitaju i podaci u sistemu budu isti kao i prije zatvaranja. Kroz aplikaciju je moguće porediti dva kursa: pronalaženje unije, presjeka i razlike između skupova studenata koji pohađaju kurs, odnosno studenata koji su položili ispit. Takođe, može se vršiti filtriranje poruka u inboksu prema zadatom stringu za pretragu.Da bi se student mogao prijaviti na kurs potrebno je da je ispunio preduslove tog kursa koji zavise od odslušanih i od položenih kurseva od strane tog studenta. Rješenje treba biti konstruisano tako da programer može da s lakoćom proširi sistem novim tipovima preduslova. Neki primjeri tipova preduslova su: da je student odslušao neki kurs, da je student položio specificiran broj kurseva sa nekim zajedničkim svojstvom, da je student položio određene kurseve s određenom minimalnom ocjenom za svaki od tih kurseva i da je student položio bar po jedan kurs iz nekoliko različitih oblasti. Moguće je ispitati graf poznanstva između svih osoba u sistemu i utvrditi najkraću putanju u grafu poznanstva između proizvoljne dvije osobe, kao i prosječnu najkraću putanju (realan broj) između osoba koje zadovoljavaju neki uslov odabran od strane korisnika (npr. prva osoba pohađa kurs A, a druga osoba pohađa kurs B). Program se mora moći kompajlirati, izvršiti i testirati. Pravilno izvršiti modularizaciju. Dozvoljeno je korištenje samo standardne biblioteke i standardne biblioteke šablona (STL). Pridržavati se principa objektno-orijentisanog programiranja, principa pisanja čistog programskog koda, principa ponovne upotrebe programskog koda, SOLID principa i konvencija za programski jezik C++. Izdvojiti u statičku biblioteku korištene komponente koje su zasebno izvan domena problema (npr. osnovna klasa za graf). Formirati testni skup podataka kojim se može inicijalizovati sistem u svrhu demonstracije funkcionalnosti.

Text of the project task

In the programming language C++, realize the simulation of the learning platform in the form of a console application. The system user has access data for the system (user name and password). The user can be enrolled in several courses. In each course, each registered user can be either a student or a lecturer in that course. A student can register for a course and attend it. The application must be approved by the course instructor. Any user can be friends with other users. A user can send a friend request to another user, which the other user can accept or decline. If the request is accepted, the two users are considered to know each other. The user can exchange messages with users he is friends with in the system. Also, each student can exchange messages with the lecturers who are in charge of the courses that the student attends. Messages are saved in the user's inbox. For example, it is possible for the messaging functionality to be implemented so that it can be demonstrated with the following sequence of actions: 1) the application starts, 2) user A logs in, 3) user A sends a message to user B and logs out of the system, and 4) user B logs in and sees a message from user A in his inbox. The system administrator can add, modify or remove courses and users. Persistence of data in the system is also enabled, so that the state of the system is preserved if the application is closed. After opening the application, the serialized data is loaded and the data in the system is the same as before closing. Through the application, it is possible to compare two courses: finding the union, intersection and difference between the sets of students attending the course and students who passed the exam. In addition, messages in the inbox can be filtered according to the specified search string. In order for a student to be able to register for a course, it is necessary to have fulfilled the prerequisites of that course, which depend on the courses taken and passed by that student. The solution should be constructed so that the developer can easily extend the system with new types of prerequisites. Some examples of prerequisite types are: that the student has taken a certain course, that the student has passed a specified number of courses with some common property, that the student has passed certain courses with a certain minimum grade for each of those courses, and that the student has passed at least one course from several different areas. It is possible to examine the acquaintance graph between all persons in the system and determine the shortest path in the acquaintance graph between any two persons, as well as the average shortest path (a real number) between persons who meet a condition selected by the user (e.g. the first person attends course A, and the other person attends course B). The program must be able to be compiled, executed and tested. Perform modularization correctly. Only the standard library and the standard template library (STL) are allowed to be used. Adhere to the principles of object-oriented programming, the principles of writing clean programming code, the principles of reuse of programming code, the SOLID principles and conventions for the C++ programming language. Separate into a static library used components that are separately outside the domain of the problem (eg the base class for the graph). Create a test set of data that can be used to initialize the system for the purpose of demonstrating functionality.

Text der Projektaufgabe

Realisieren Sie in der Programmiersprache C++ die Simulation der Lernplattform in Form einer Konsolenanwendung. Der Systembenutzer hat Zugangsdaten zum System (Benutzername und Passwort). Der Benutzer kann in mehreren Kursen eingeschrieben sein. In jedem Kurs kann jeder registrierte Benutzer entweder Student oder Dozent in diesem Kurs sein. Ein Student kann sich für einen Kurs anmelden und daran teilnehmen. Der Antrag muss von der Lehrgangsleitung genehmigt werden. Jeder Benutzer kann mit anderen Benutzern befreundet sein. Ein Benutzer kann einem anderen Benutzer eine Freundschaftsanfrage senden, die der andere Benutzer annehmen oder ablehnen kann. Wird die Anfrage angenommen, gelten die beiden Nutzer als bekannt. Der Benutzer kann Nachrichten mit Benutzern austauschen, mit denen er im System befreundet ist. Außerdem kann jeder Student Nachrichten mit den Dozenten austauschen, die für die Kurse verantwortlich sind, die der Student besucht. Nachrichten werden im Posteingang des Benutzers gespeichert. Beispielsweise ist es möglich, die Messaging-Funktionalität so zu implementieren, dass sie mit folgender Aktionsfolge demonstriert werden kann: 1) die Anwendung startet, 2) Benutzer A meldet sich an, 3) Benutzer A sendet eine Nachricht an Benutzer B und sich vom System abmeldet und 4) Benutzer B sich anmeldet und eine Nachricht von Benutzer A in seinem Posteingang sieht. Der Systemadministrator kann Kurse und Benutzer hinzufügen, ändern oder entfernen. Die Persistenz von Daten im System wird ebenfalls aktiviert, sodass der Zustand des Systems erhalten bleibt, wenn die Anwendung geschlossen wird. Nach dem Öffnen der Anwendung werden die serialisierten Daten geladen und die Daten im System sind die gleichen wie vor dem Schließen. Durch die Anwendung ist es möglich, zwei Kurse zu vergleichen: Finden der Vereinigung, Schnittmenge und Differenz zwischen den Gruppen von Studenten, die den Kurs besuchen, und Studenten, die die Prüfung bestanden haben. Zusätzlich können Nachrichten im Posteingang nach dem angegebenen Suchbegriff gefiltert werden.Damit sich ein Student für eine Lehrveranstaltung anmelden kann, ist es erforderlich, dass die Voraussetzungen dieser Lehrveranstaltung erfüllt sind, die von den belegten Lehrveranstaltungen abhängen und von diesem Schüler bestanden. Die Lösung sollte so aufgebaut sein, dass der Entwickler das System einfach um neue Arten von Voraussetzungen erweitern kann. Einige Beispiele für Voraussetzungstypen sind: dass der Student einen bestimmten Kurs belegt hat, dass der Student eine bestimmte Anzahl von Kursen mit einigen Gemeinsamkeiten bestanden hat, dass der Student bestimmte Kurse mit einer bestimmten Mindestnote für jeden dieser Kurse bestanden hat und dass der Studierende mindestens eine Lehrveranstaltung aus mehreren unterschiedlichen Bereichen bestanden hat. Es ist möglich, den Bekanntschaftsgraph zwischen allen Personen im System zu untersuchen und den kürzesten Weg im Bekanntschaftsgraph zwischen zwei beliebigen Personen sowie den durchschnittlich kürzesten Weg (eine reelle Zahl) zwischen Personen zu bestimmen, die eine vom Benutzer ausgewählte Bedingung erfüllen (z. B. die erste Person besucht Kurs A und die andere Person besucht Kurs B). Das Programm muss kompiliert, ausgeführt und getestet werden können. Führen Sie die Modularisierung korrekt durch. Es dürfen nur die Standardbibliothek und die Standardvorlagenbibliothek (STL) verwendet werden. Halten Sie sich an die Prinzipien der objektorientierten Programmierung, die Prinzipien des Schreibens von sauberem Programmiercode, die Prinzipien der Wiederverwendung von Programmiercode, die SOLID-Prinzipien und Konventionen für die Programmiersprache

C++. Trennen Sie verwendete Komponenten, die separat außerhalb des Problembereichs liegen (z. B. die Basisklasse für den Graphen), in eine statische Bibliothek. Erstellen Sie einen Testdatensatz, der zum Initialisieren des Systems verwendet werden kann, um die Funktionalität zu demonstrieren.