UNIVERZITET U TUZLI
FAKULTET ELEKTROTEHNIKE

Objektno-orijentirano programiranje

Zadaća 4

Tuzla, decembar/prosinac 2021.

Sadržaj

Zadatak 1

Kreirati program koji od korisnika trazi unos proizvoljnog broja cijelih brojeva. Unijeti sve brojeve u listu. Sortirati listu te ispisati najveci i najmanji broj. Kao pomoc, kreirati funkciju koja prima listu i cijeli broj. Funckija treba da simulira operator [ ], te vraca referencu na broj u listi na poziciji koju diktira cijeli broj. Uz pomoc te funkcije vrsiti sortiranje. Tipove parametara funckije, kao i povratne tipove zakljuciti na osnovu postavke.

Ulaz	Izlaz
1 2 10 2 -1 235	-1 235

Zadatak 2

Kreirati program koji od korisnika trazi unos prozivoljnog broja rijeci. Nakon sto korisnik zavrsi sa unosom (CTRL + D), ispisati broj pojavljivanja svake od rijeci. Ukoliko korisnik unese rijec sa svim velikim slovima potrebno je umanjiti broj pojavljivanja za jedan. Ukoliko rijec dostigne broj pojavljivanja nula, potrebno je obrisati rijec iz mape i ne prikazati je. Program uraditi uz pomoc map kontejnera, iteriranje izvrsiti uz pomoc iteratora, te se sluziti metodama find, i erase.

Ulaz	Izlaz
emir emir test jabuka kruska jabuka EMIR KRUSKA	emir: 1 jabuka: 2 test: 1

Zadatak 3

Napisati **samo jednu** funkciju koja kao argument prima bilo koji kontejner (list ili vektor) sa elementima bilo kojeg tipa, te ispisuje sve elemente.

Kod	Izlaz

`std::vector<int> niz1{1,2,3};`
`std::list<float> niz2{1.1,2.2,3.3};`
`ispisi(niz1);`
`ispisi(niz2);`
	1 2 3
	1.1 2.2 3.3

Zadatak 4

Napisati funkciju koja kao argument prima mapu, lambdu, i varijablu generickog tipa. Mapa kao kljuc ima genericki tip a kao vrijednost list kontejner koji pohranjuje elemente bilo kojeg tipa. Lambda predstavlja kriterij za sortiranje elemenata liste. Genericki tip kao treci argument sluzi za identifikaciju liste u mapi, te ukoliko ne identificira ni jednu listu, potrebno je u mapu dodati novu, te okoncati izvrsenje funkcije, u suprotnom sortirati identificiranu listu. Funkcija ne vraca nista, sprijeciti bilo kakvo kopiranje.

Zadatak 5

Kreirati program koji ce simulirati bankomat. Stukturom Bankomat sa adekvatnim atributima i metodama modelirati bankomat. Omoguciti funkcionalnost dizanja novca, uplacivanja novca, te provjeru stanja na racunu. Zabraniti bilo koju funkcionalnost dok god se korisnik ne autentifikuje brojem kartice i pinom. Dizanje novca kosta 1KM, dok provjera stanja 0.5KM, dok je uplata novca bez kamate. Ukoliko korisnik pokusa upotrijebiti bilo kakvu funkcionalnost bez adekvatnog stanja na racunu, zabraniti mu isto. Ukoliko korisnik tri puta unese pogresan pin, zakljucati bankomat i sve njegove funkcionalnosti na 5 sekundi. Potrebno je da bankomat pamti sve operacije izvrsene nad bankomatom, te ukoliko se pozove metod *historija,* ispisati ih na ekran. Metod *historija* trazi unos password-a (proizvoljno odabrati). Prilikom ispisa operacija, operacija autentifikacije mora biti cenzuirisana na nacin da pin bude cenzurisan sa zvjezdicama, a broj kartice cenzurisan sem prva cetri broja. Otkljucavanje bankomata cekati uz pomoc funkcije na linku. Bankomat treba da u svojoj bazi ima minimalno tri korisnika, koje je potrebno unijeti prilikom konstrukcije. Primjer ispisa operacija:

Autentifikacija: kartica: 1234 **** **** ****, PIN: ****
Dizanje novca: 10.00KM
Provjera stanja racuna
Odjava
Ispis historije
Neuspjesna autentifikacija
Autentifikacija: kartica: 1111 **** **** ****, PIN: ****
Neuspjesno dizanje novca
Odjava

Zadatak 6

Potrebno je implementirati program koji ima mogućnost unosa i evidencije studenata, predmeta i ocjena. Program treba da podržava operacije dodavanja novih predmeta, dodavanje studenata, editovanje polja jednog studenta te ispis svih studenata zajedno sa svim ocjenama iz pojedinih predmeta i prosjekom. Sve ove operacije implementirati korisničkim menijem. Uzeti u obzir da naziv predmeta i odsjeka na kojem se predmet nalazi može imati više riječi.

Strukture `Predmet` i `Student` modelirati kao što je to ispod prikazano:

```
struct Predmet {
     std::string naziv;
     std::string odsjek;
};

struct OcjenaIzPredmeta {
     int ocjena;
     std::list<Predmet>::const_iterator predmet;
};

struct Student {
     std::string brojIndeksa;
     std::string ime;
     std::string prezime;
     std::string grad;
     std::vector<OcjenaIzPredmeta> ocjene;
};
```

Iz modela strukture `OcjenaIzPredmeta` jasno je da je za predmete potrebno koristiti kontejner tipa `std::list`. Zašto ne vektor? Šta se sa iteratorima vektora dešava kada se vektor kontejner proširuje?

Unos novih predmeta treba da bude vrlo jednostavan, od korisnika se traži unos naziva predmeta i odsjeka. Novi predmet je potrebno dodati na kraj kontejnera tipa `std::list<Predmet>` osim ako predmet sa unešenim nazivom već ne postoji. U tom slučaju ispisati grešku korisniku te se vratiti na početni meni.

Unos studenta treba da ide u redoslijedu:

1. Indeks (samo jedna riječ). Ukoliko student sa unesenim brojem indeksa već postoji, ispisati grešku na ekran te se vratiti nazad na početni meni.
2. Ime (može sadržavati više riječi)
3. Prezime (može sadržavati više riječi)
4. Grad (može sadržavati više riječi)
5. Unos ocjena
    a. Unos predmeta za koju je ocjena vezana. Ukoliko predmet ne postoji ispisati grešku na ekran, vratiti se na početni meni i ne dodavati ništa u kolekciju studenata. U suprotnom kreirati instancu strukture `Ocjena` iz predmeta čiji iterator pokazuje na konkretan predmet iz kolekcije.
    b. Unos ocjene (moguće vrijednosti od 5 do 10). Ukoliko korisnik unese vrijednost manju od 5 ili veću od 10 zatražiti ponovni unos ocjene.

Opcija editovanja studenta treba od korisnika da traži unos broja indeksa. Ukoliko student sa tim brojem indeksa ne postoji, ispisati grešku i vratiti se na početni meni, u suprotnom potrebno je ponuditi novi meni gdje korisnik može odabrati kakvu operaciju editovanja želi odraditi. Moguće operacije su:

1. Mijenjanje broja indeksa studenta
2. Mijenjanje imena studenta
3. Mijenjanje prezimena studenta
4. Mijenjanje grada studenta
5. Brisanje svih ocjena
6. Dodavanje novih ocjena

Struktura `Student` i sve potencijalne funkcije vezane za nju je potrebno izdvojiti u poseban hpp file. Svu implementaciju ovih funkcija vezanih za studente je potrebno prebaciti u poseban cpp file.