PROJEKTNI ZADATAK

Studenti:

1. Boris Divković
2. Sanela Fazlić
3. Almedina Žunić
4. Ahmed Smajlović

Sadržaj

[1 Tema 1](#_Toc122537262)

[2 Izrada i predavanje projekta 1](#_Toc122537263)

[3 Tehnički zahtjevi 1](#_Toc122537264)

[4 Formalni zahtjevi 2](#_Toc122537265)

[5 Izvještaj o projektu 3](#_Toc122537266)

[5.1 Uvod 3](#_Toc122537267)

[5.2 Main funkcija 3](#_Toc122537268)

[5.3 Header file “libIncludes” 4](#_Toc122537269)

[5.4 Zaglavlja i funkcije unutar main-a 4](#_Toc122537270)

[5.4.1 Registracija 4](#_Toc122537271)

[5.4.2 Login 5](#_Toc122537272)

[5.4.3 Administratorske funkcionalnosti 6](#_Toc122537273)

[5.4.4 Korisničke funkcionalnosti 7](#_Toc122537274)

[5.5 Zaključak 8](#_Toc122537275)

# Tema

Napisati program za upravljanje radom biblioteke.

# Izrada i predavanje projekta

1. Projekat rade grupe po najviše 4 studenta.
2. Rok za predaju projekta je 20.12.2022. godine.
3. Završen CodeBlocks C++17 projekat i Word dokument arhivirati kao ZIP datoteku i poslati na e-mail adresu [mirza.hadzic@ipi-akademija.ba](mailto:mirza.hadzic@ipi-akademija.ba) sa subject-om “PJIP Projekat”. Obavezno obrisati izvršne/objektne datoteke (.exe i .o) iz projekta prije arhiviranja!
4. Projekat se predaje i ocjenjuje prije izlaska na završni ispit.
5. Ukoliko student odluči ne raditi projekat, može izaći na završni ispit.
6. Ukoliko student pošalje projekat nakon termina završnog ispita, mora ponovo izaći na završni ispit kako bi se ostvarili bodovi na projekat.
7. Ispunjenjem tehničkih i formalnih zahtjeva grupa studenata ostvaruje po 20 bodova.
8. Djelimičnim ispunjenjem tehničkih i formalnih zahtjeva asistent odlučuje o broju bodova koji će se dodjeliti grupi za projekat.

# Tehnički zahtjevi

1. Omogućiti operacije unosa, izmjene, pregleda i brisanja: članova, radnika, knjiga, književnog žanra, polica za knjige, kategorija članarina, te posuđenih knjiga.
2. Unos novih stavki, te izmjenu postojećih vršiti sa tastature.
3. Željenu radnju (unos, izmjena, pregled, brisanje) korisnik unosi preko tastature na osnovu ponuđenih opcija u terminalu.
4. Za novi unos definirati potrebne kolone i ograničiti unos ukoliko se ne unesu potrebni podaci.
5. Za povezivanje tabela koristit će se ključevi, koji predstavljaju jedinstveni identifikator određene stavke, kao npr:

* radnik/član: tekstualni zapis JMBG od 13 cifara,
* knjiga: tekstualni zapis od 5 znakova, gdje 1. znak označava žanr (B1010, K5425, itd.),
* polica: tekstualni zapis sa oznakom žanra i numeracijom police;

1. Napraviti dokumentaciju projekta u obliku Word dokumenta. Koristiti ovaj dokument kao šablon, te početi sa 5. poglavljem svoj izvještaj o projektu.

# Formalni zahtjevi

1. Programski kôd (*backend*) pisati u C++ programskom jeziku.
2. Koristiti što više elemenata C++17 standarda.
3. Programsko sučelje (*frontend*) bit će terminal (*Command Prompt* ili UNIX terminal).
4. Strogo se pridržavati principa objektno – orijentiranog programiranja.
5. Jasno definirati potrebne klase.
6. Koristiti naslijeđivanje gdje je potrebno.
7. Maksimalno ograničiti vidljivost i pristup članovima klase.
8. Implementirati sve konstruktore i destruktore.
9. Napraviti odgovarajuće verzije konstruktora ovisno o očekivanom unosu.
10. Koristiti STL kontejnere, a gdje to nije moguće koristiti dinamički alociranu memoriju.
11. Smanjiti ponavljanje kôda kroz implementaciju funkcija.
12. Gdje je moguće koristiti metode umjesto funkcija.
13. Deklaracije izdvojiti u .h datoteke, definicije u .cpp datoteke, a napraviti i main.cpp datoteku koja predstavlja glavni program.
14. Paziti na zauzimanje resursa radne memorije.
15. Objekte, klase, članove klase i njene metode imenovati razumljivo.
16. Nastojati na jedinstven način imenovati metode koji obavljaju iste funkcije u više klasa.

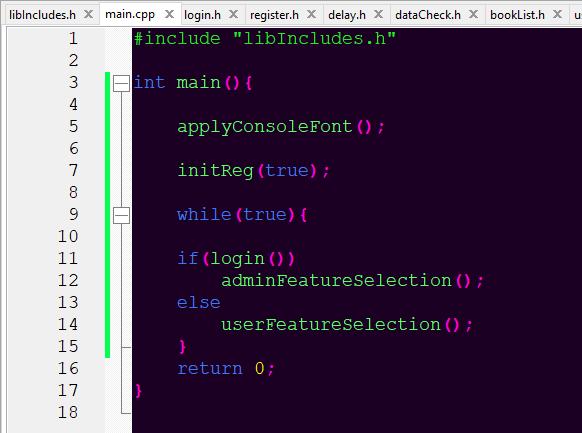
# Izvještaj o projektu

## **Uvod**

Projekat aplikacije za administraciju biblioteke nalazi se u folderu “libProj” pod nazivom “main.cpp”. Aplikacija je napisana u C++ programskom jeziku (podržava izdanja do C++17 pod GNU g++ kompajlerom). Svi dodatni file-ovi pored “main.cpp” sadrže dodatni kod potreban za izvršavanje funkcija i klasnih metoda koje rade sa unesenim i postojećim podacima. Zbog nedostatka baze podataka, svi uneseni podaci su privremeni, te žive za vrijeme izvršavanja procesa aplikacije. Aplikacija kao i dodatni opis trenutnih i budućih mogućnosti unutar aplikacije se nalaze online unutar GitHub repozitorija organizovanog uputrebom git bash-a. Svi dosadašnji radovi na projektu su zapisani u vidu commit-ova unutar repozitorija. Repozitorij je trenutno privatan iz sigurnosnih razloga, ali se njegova vidljivost može promijeniti. Kod trenutno ne sadrži komentare, te će u narednih nekoliko poglavlja biti detaljno objašnjeni ključni dijelovi koda za izvršavanje ove aplikacije kao i njena struktura.

## **Main funkcija**

Main() funckija se nalazi unutar “main.cpp” datoteke i predstavlja glavnu izvršnu funkciju ovog programa. Njena struktura predstavljena je slikom 5.2.1.

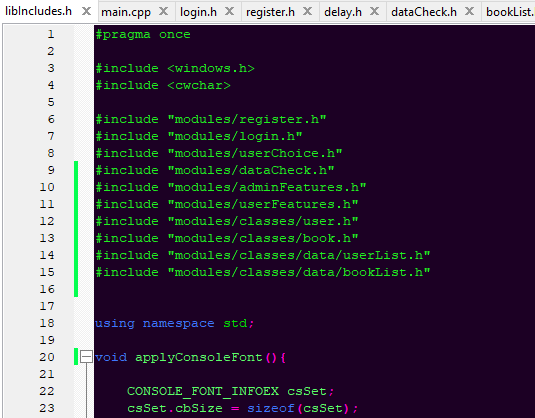


*Slika 5.2.1. – Izvorni kod i struktura main() funkcije*

Iz ove funkcije se poziva funkcija applyConsoleFont() koja konzoli daje font i drugu veličinu prilikom izvršavanja programa, te se učitava prije ostatka programa. Zatim, korisnik se loguje prvi put kao administrator sistema pomoću initReg() funkcije. Beskonačna petlja nam ne dozvoljava da se program prekine, te se korisnici prijavljuju i odjavljuju po potrebi. U zavisnosti od povratne vrijednosti login() funkcije (true ili false), program prepoznaje da li je trenutni instanca administrator ili obični korisnik. To je nužno jer se razlikuju po mogućnostima rada sa podacima unutar programa.

## **Header file “libIncludes”**

Unutar main funkcije, nalazi se komanda #include “libIncludes.h” koja dozvoljava kompajleru da prepozna sve funkcije i vrste podataka neophodne za izvršavanje programa. Na slici 5.3.1. nalazi se dio koda iz “libIncludes.h” zaglavlja.



*Slika 5.3.1. – Izvorni kod zaglavlja libIncludes.h*

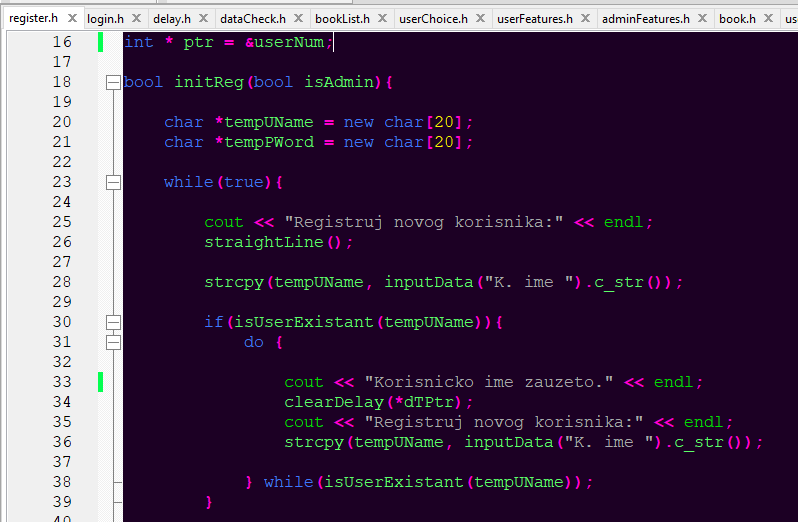
Zaglavlje sadrži sve neophodne #include komande poput #include “modules/register.h” koja dozvoljava poziv initReg() funkcije unutar main-a, jer se nalazi u navedenom zaglavlju.

## **Zaglavlja i funkcije unutar main-a**

Potrebno je spomenuti da se u “libIncludes.h” nalazi većina zaglavlja koja su potrebna za izvršavanje programa, dok su neka zaglavlja uključena samo u zaglavlja koja koriste njihove funkcionalnosti. Tako sam strukturirao kod i datoteke jer nije bilo potrebe da se main() zatrpava prekomjerno. U narednih nekoliko poglavlja preći ću ključna zaglavlja, počevši od register zaglavlja.

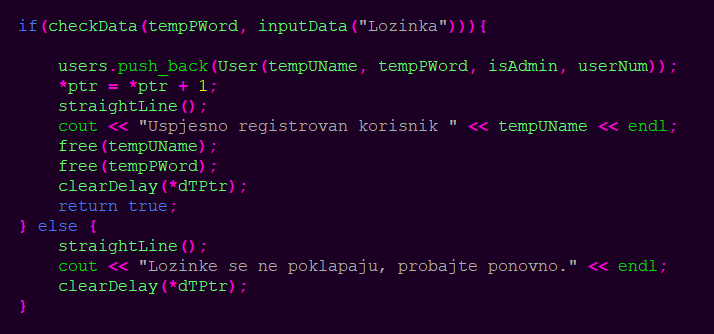
### **Registracija**

Unutar register zaglavlja, predstavljena je jedna od najvećih funkcija unutar ovog koda i zove se initReg(). Prima jedan parametar – bool podatak koji govori da li će korisnik koji se registruje biti admin (true) ili neće (false). Ovo je neophodno jer želimo da se korisnik može samo prijaviti u naš sistem kada ispuni uslove za to. Želimo da naši administrator registruje nove korisnike po potrebi, te nam prvi korisnik mora biti administrator. Postupak bi bio mnogo lakši sa nekom bazom podataka, te nam initReg() ne bi bio potreban na početku main funkcije. Na narednoj slici prikazano je zaglavlje i initReg() unutar njega.



*Slika 5.4.1.1. – Dio tijela initReg() funkcije*

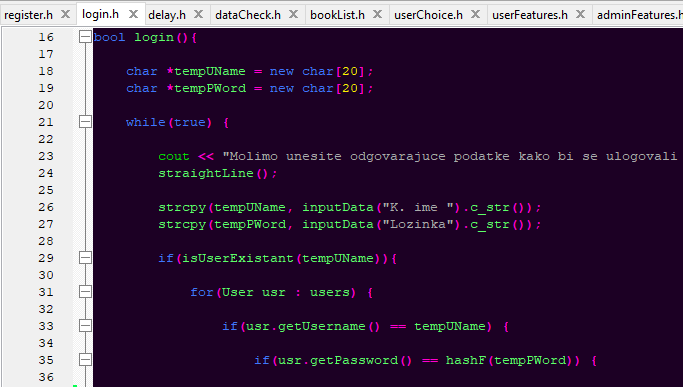
Pomoću ove funkcije, korisnik se registruje – unosi svoje podatke u lokalnu memoriju programa, te se kasnije može prijaviti u svoj nalog sa registrovanim podacima. Provjerava se naravno, da li je uneseno korisničko ime već u upotrebi, te da li je lozinka koju je unio prvi put identična lozinci koju unosi za potvrdu. Može se primjetiti upotreba dinamičke alokacije memorije za vrijeme registracije, kako podaci o lozinkama korisnika ne bi ostali vidljivi dugo, prije nego što se hashiraju (enkriptuju). Postoji funkcija u programu pod nazivom “hashF” koja obavlja zadatak hashiranja.



*Slika 5.4.1.2. – Tražimo od korisnika da potvrdi unesenu lozinku*

### **Login**

Nakon što se podaci o registrovanom korisniku unesu u lokalnu memoriju, nastupa glavni dio main funkcije koji odlučuje koje će od postojećih funkcionalnosti biti prikazane u zavisnosti od administratorskog statusa naloga. Login funkcija se nalazi unutar “login.h” zaglavlja pod nazivom login(). Ova funkcija ne prima nikakve parametre jer se radi o validaciji unesenih podataka u odnosu na postojeće podatke u memoriji.

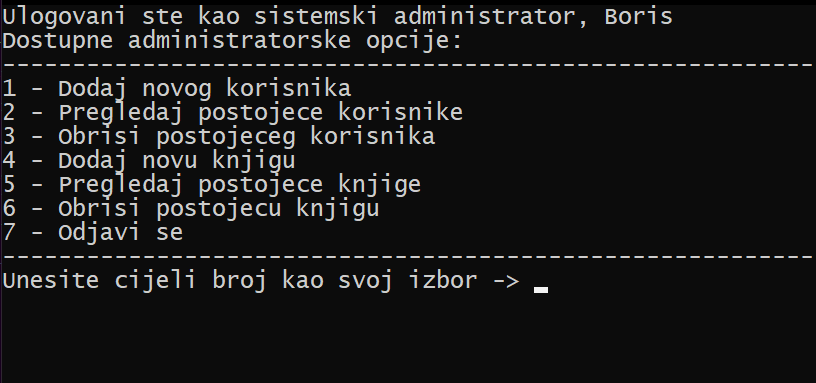


*Slika 5.4.2.1. – Tijelo login() funkcije*

Zahtjevamo od korisnika da unese podatke o svom nalogu, te će ako nalog postoji, korisniku biti dozvoljen pristup aplikaciji. Za vrijeme autorizacije provjeravaju se podaci koje korisnik unosi, kako po formatu, tako i po validnosti podataka na osnovu poređenja sa zapisima u lokalnoj memoriji.

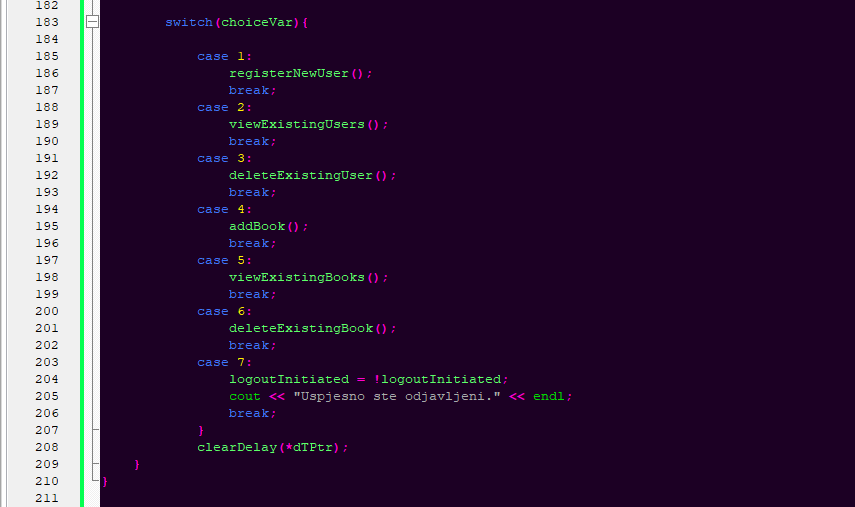
### **Administratorske funkcionalnosti**

Nakon što smo se prijavili sa nekim nalogom, ukoliko je administratorski status naloga tačan, prijavljujemo se kao administrator te smo dobili pristup administratorskim funkcionalnostima sistema unutar aplikacije. Meni je prikazan na narednoj slici (5.4.3.1.).



*Slika 5.4.3.1. – Funkcionalnosti dozvoljene administratorima unutar aplikacije*

Administrator odavdje može izabrati koja od navednih funkcionalnosti sistema mu je potrebna da izvrši neku postojeću funkciju u sklopu programa. Unutar koda organizovano je da korisnik odabere opciju po želji tako što će unijeti cijeli broj od 1 do 7 (granice uključene). Time se otvara navedena funkcionalnost koju korisnik želi upotrijebiti. Kako je ranije spomenuto, unos je ograničen, te vrijednosti osim cijelih brojeva ne utiču na odabir funkcionalnosti već izazivaju ponovni unos dok se kriterij ne zadovolji.

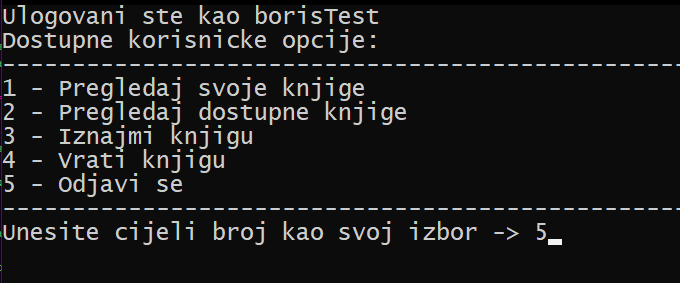


*Slika 5.4.3.2. – Struktura odabira administratorske funkcionalnosti u kodu*

Odabir je realizovan pomoću switch case kontrolnog bloka koji na osnovu unosa od strane korisnika (choiceVar), nakon validacije, otvara funkciju za izvršavanje relevantne funkcionalnosti.

### **Korisničke funkcionalnosti**

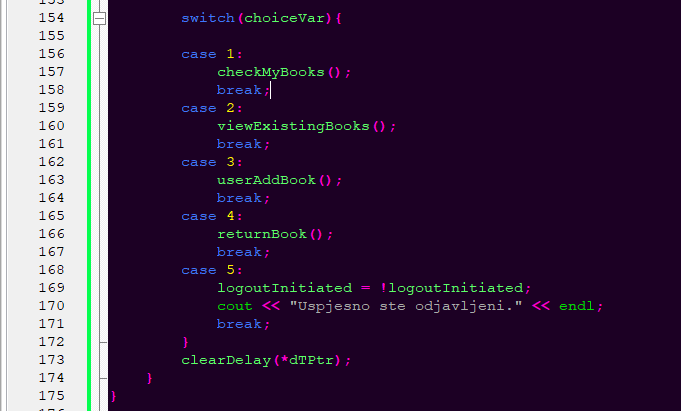
Kada se prijavimo kao korisnik (bez administratorskih privilegija), preko naloga koji je za nas kreirao administrator, otvara nam se meni za odabir funkcionalnosti koje su trenutno integrisane u program.



*Slika 5.4.4.1. – Funckionalnosti dostupne korisnicima*

Odavdje korisnik može iznajmiti knjige koje su dostupne u sistemu (koje je ubacio administrator), te ih vratiti po potrebi. Korisnik također može pregledati listu knjiga koje je iznajmio do sada.

Slično strukturi koda za odabir administratorskih funkcionalnosti, i odabir korisničkih funckionalnosti uključuje switch case kontrolni blok koji je vidljiv na narednoj (5.4.4.2.) slici.



*Slika 5.4.4.1. – Struktura odabira funkcionalnosti za korisnika unutar koda*

Kada korisnik unese svoj odabir, pomoću switch case bloka mu se otvara meni za datu funckionalnost, gdje korisnik može dalje upotrijebiti relevantnu funkcionalost za svoje ciljeve.

## **Zaključak**

Unutar ove aplikacije vidljiva je potreba za bazom podataka. Svi podaci koji prođu kroz ovu aplikaciju bivaju izgubljeni momenta kada se proces zaustavi. Što se tiče dosadašnjih funkcionalnosti i logičke postavke koda, ima mnogo prostora za poboljšanja. Neki dijelovi postojećeg koda se mogu presložiti logički i semantički kako bi kod imao više smisla ili na efikasniji način obavljao svoju funkciju. Korisničko sučelje je ograničeno što se tiče formatiranja jer je sučelje tekstualnog oblika. Postoje načini da se C++ kod iskoristi za građenje korisničkog sučelja poput Qt framework-a, te se u budućnosti može napraviti funckionalno grafičko korisničko sučelje na bazi dosadašnjeg backend koda. Eventualne sitne korekcije su neizbježne.

Funkcionalnosti rade kako bi trebale, te za vrijeme odbrane ovog projekta postoji mogućnost isprobavanja svih funkcionalnosti i praćenja uticaja njihovog izvršavanja unutar aplikacije.