 Univerzitet u Zenici 

Politehnički fakultet

Odsjek: softversko inženjerstvo

**ZAVRŠNI PROJEKAT IZ PREDMETA OBJEKTNO ORJENTISANOG PROGRAMIRANJA, AKADEMSKA 2020/2021 GODINA  
NAZIV PROJEKTA: BLAGAJNIČKO POSLOVANJE**

Januar 2021, Zenica

Studenti koji su radili na izradi projekta:

Čančar Ajna 95

Okanović Amina 102

Totić Zlatko

**Objašnjenje projekta:**

Projekat iz predmeta Objektno orjentisanog programiranja, za akademsku 2020/2021 godinu je projekat na temu: *„Blagajničko poslovanje“*. Prije same realizacije projekta, koju ćemo naknadno objasniti u ovoj dokumentaciji, putem Google Meet platforme smo stupili u kontakt te započeli s idejnom realizacijom projekta. Na samom početku smo razvili idejni plan, a zatim i sam realizacijski plan. Dakle, prvo smo razgovarali o tome što naš projekat treba sadržavati i kome je zapravo on namjijenjen. Program „Blagajničko poslovanje“ je namijenjen blagajničkom poslovanju Univerziteta u Zenici (koji će kasnije naknadno biti objašnjen). Ideja samog projekta se bazira na olakšavanju tzv. „administrativnog dijela“ prilikom kreiranja dnevnika i unosa samih uplata, isplata, kao i vođenja samih blagajničkih dnevnika Univerziteta.

Postavlja se pitanje *Šta je to blagajničko poslovanje* ? -Program blagajničkog poslovanja obuhvaća čuvanje i promet novca i evidenciju dnevnika. Promet novca podrazumijeva izravnu predaju gotovine između sudionika platnog prometa te uplatu gotovine na ili isplatu sa transakcijskog računa (kreiran je za Univerzitet u Zenici). Također, program podrazumijeva i izradu blagajničkih dokumenata u koje ubrajamo uplatu, isplatu i dnevnik. Blagajnički dokument predstavlja knjigovodstvenu potvrdu kojim se evidentira promet unutar blagajne.

**Kako zapravo funkcioniše program blagajničkog poslovanja?**

Ideja programa blagajničkog poslovanja se zasniva da se korisniku obezbijedi siguran i pouzdan pristup samom programu te da mu se zapravo olakša administrativni dio posla. Sam program se sastoji od nekoliko podmenija, koji u suštini sadrže registraciju i prijavu na sam sistem. Dakle na samom ulasku se nalazi startni meni putem kojeg se korisnik pruža opcija registracije ili prijave. Bitno je napomenuti da ukoliko se treba registrovati novi korisnik, registraciju isključivo može vršiti administrator kako bi se osigurala sigurnost samog programa, te se time sprječava pristup neovlaštenih osoba u sam program. Kako bi se osigurala veća puzdanost i sigurnost, prilikom registracije vrši se enkripcija podataka i kao takvi podaci se unose u datoteke, iz koje se kasnije čitaju. Nakon što se korisnik uspješno registruje/prijavi na sistem nudi mu se meni s različitim opcijama vezanih za blagajničko poslovanje kao što su uplata, isplata, kreiranje blagajničkih dnevnika, itd. Nakon što korisnik završi s poslom, nude mu se dvije opcije, a to su : napuštanje programa ili vraćanje na početni dio programa, odnosno registraciju/prijavu.

U nastavku slijedi objašnjenje koda.

**Pojašnjenje koda:**

***Klasa Prijava:***

U Prijava.h smo pozvali sve biblioteke koje su nam potrebe da bi nam program radio:

#include <iostream>

#include <string>

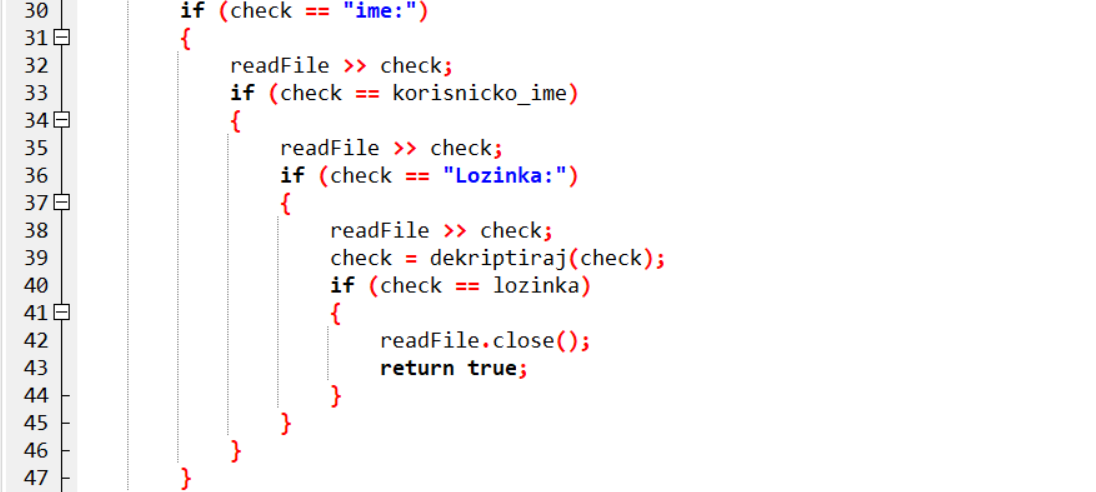
#include "Osoba.h"

U klasi Prijava nalaze se četiri virtualne funkcije koje će kasnije biti implementirane u klasama Blagajnik i Administrator, jer će te funkcije biti drugačije u svakoj od te dvije klase. Kao što im i sami nazivi govore, to su klase koje će se koristiti za ekripciju i dekripciju samih podataka kao i registraciju i/ili prijavu nekog korisnika.

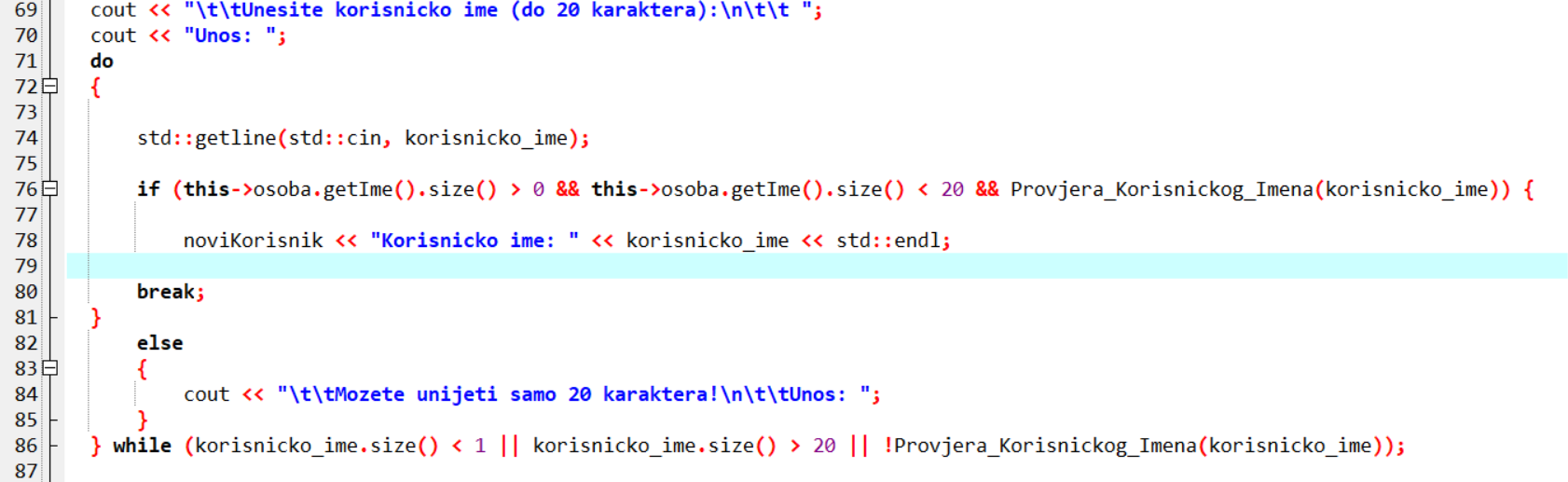
***Klada Blagajnik:***

Klasa Blagajnik sastoji se od peta funkcija, od toga su četiri nasljeđene funkcije (virtaulne funkcije) iz njene bazne klase Prijava i njena nenaslijeđena funkcija koja je zadužena za provjeru da li prilikom registracije postoji u sistemu postoji već dato uneseno korisničko ime.

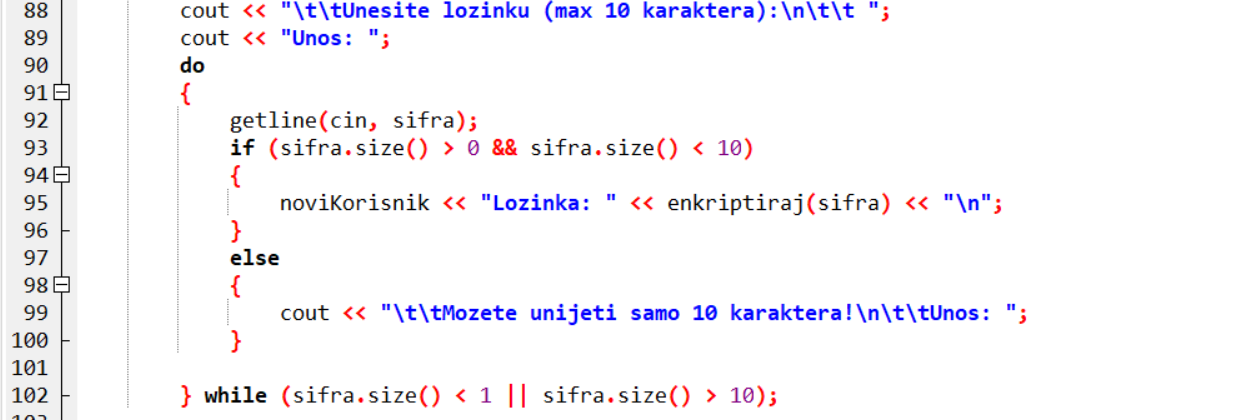
Prva funkcija koju ćemo objasniti jeste funkcija **loginin**. To je funkcija koja provjerava da li korisnik s proslijeđenim korisničkim imenom i njegovom lozinkom se nalazi u datoteci. Data funkcija je bool tipa, odnosno vraća vrijednost bool, to jeste vraća true ukoliko su uneseni podaci tačni, odnosno ukoliko su korisnici upisani na sistem ili vraća false ukoliko uneseni podaci nisu tačni ili korisnik nije upisan na sistem. Shodno navedenom, korisniku se (ne)dupošta pristup dalje programu. Prilikom prijave podaci se čitaju iz datoteke i uspoređuju se s prosljeđenim parametrima.



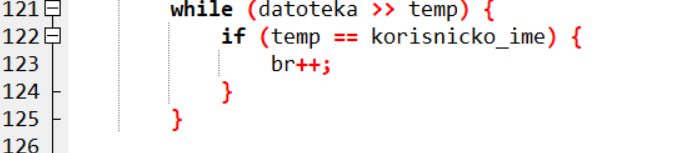
Druga funkcija jeste **signup** pomoću koje se unosi novi blagajnik u sistem. Prvo nakon što je datteka u koju se trebaju unijeti podaci otvorena, unose se opći podaci novog blgajnika (ime i prezime), a to se radi uz pomoću funkcija klase **Osoba** koje će biti naknadno objašnjene. Nakon toga, pristupa se unošenju korisničkog imena za korisnika kojeg želimo registrovati u sistem, s time da postoje ograničenja koja se odnose na to da korisničko ime ne može biti duže od 20 karaktera i da ne se korisničko ime ne može ponavljati, odnosno da dva korisnika ne mogu imati isto korisničko ime.



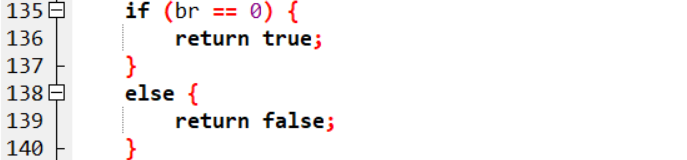
Nakon što smo omogućili unos korisničkog imena, traži se unos lozinke koje također ima ograničenje, koje se odnosi na to lozinka ne može biti duža od deset karaktera. Prilikom zapisivanja lozinke u datoteku dolazi do njene eknripcije.



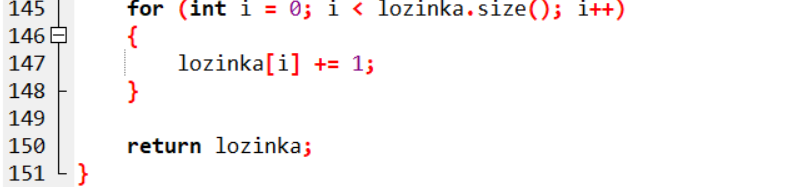
Nakon signup funkcije, slijedi nam funkcija **Provjera\_Korisnickog\_Imena** koja je bool tipa. Ova funkcija provjera da li dolazi do ponovljanja korisničkog imena prilikom registracije, odnosno signup-a. Vraća true ukoliko ne postoji ponavljanje i false ukoliko postoji ponavljanje korisničkog imena.



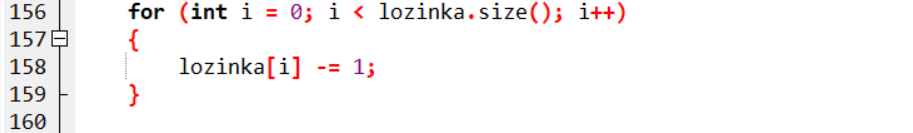
Uz pomoć brojača, prilikom prolaska kroz datoteku, brojač broji da li uneseno korisničko ime postoji već u datoteci.



Funkcija **enkriptiraj** koja kao paramter prima lozinku koja se unosi, vrši kao što i sam naziv kaže enkriptovanje date lozinke, i to na način da se svaki karakter lozinke prema ASCII kodu povećava za jedan. Povratni tip je string, te se vraća enkriptovana lozinka.



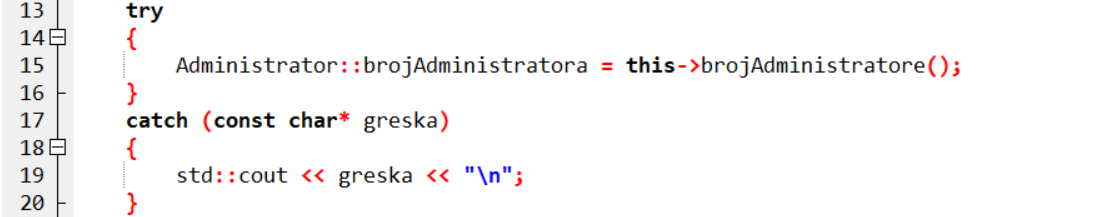
Posljednja funkcija u klasi **Blagajnik** jeste funkcija **dekriptiraj** koja kao parametar prima enkriptovan lozinku koja se čita iz datoteke prilikom prijave na sistem. Ova funkcija radi suprotno od funkcije enkriptiraj. Ona prema ASCII kodu date karaktere lozinke umanjuje za jedan.

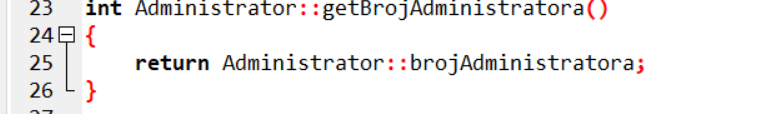


Ova funkcija je tipa bool, koja vraća stvarnu lozinku.

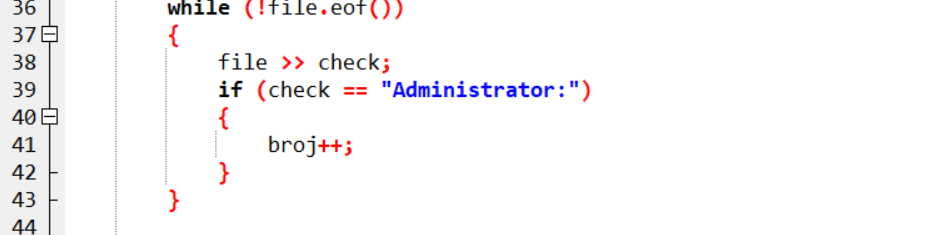
***Klasa Administrator:***

Kako je i klasa **Administrator** naslijeđena iz bazne klasa Prijava, ona kao i klasa Blagajnik sadrži implementaciju virtualnih funkcija login, signup, enkriptiraj i dektriptiraj koje rade na sličnom principu kao i prethodno opisane u klasi Blagajnik. Administrator ima i neke svoje funkcije od kojih su dvije funkcije koje suže za set i get static integer varijable.

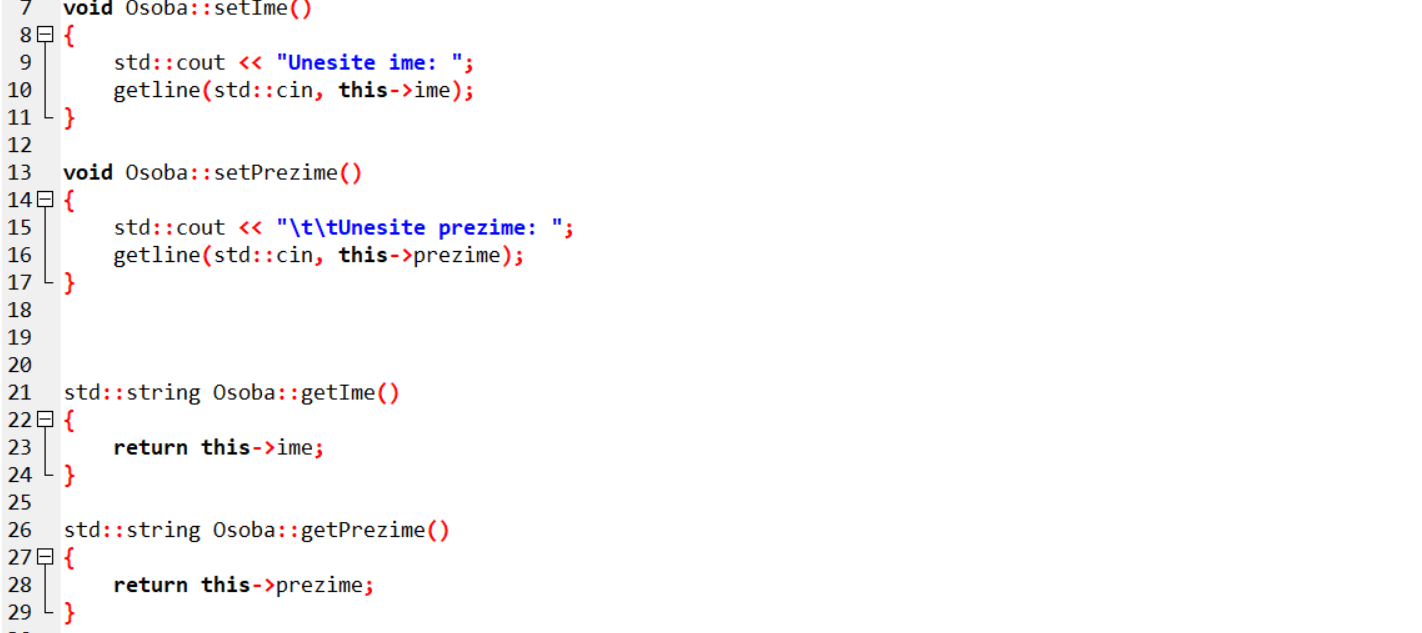
Prva je void **setBrojAdministarora**. Ova funkcija postavlja datu static varijablu na određeni broj to jest govori nam koliko administratora ima u sistemu, a omogućava nam genirasanje korisničkih imena u sistemu. Geter date static varijable vraća vrijednost te varijable.



Naredna funkcija iz klase Administrator jeste **brojAdministratore.** Ova funkcija vraća trenutni broj administratora koji se nalazi u sistemu na način da prolazi kroz datoteku u koju su upisani podaci o administratorima i prebrojava koliko ih zapravo ima.



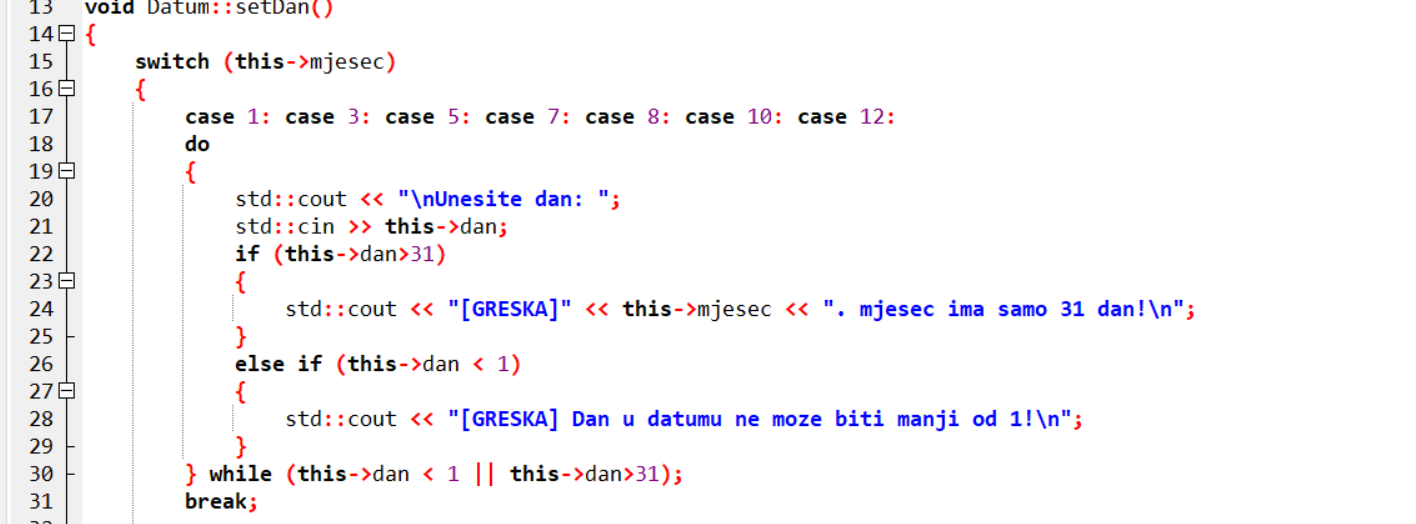
***Klasa Osoba:***

Klasa **Osoba** predstavlja generalnu klasu za unos općih podataka za registraciju korisnika u sistem. Klasa **Osoba** sadrži funkcije za unos imena i prezimena datih osoba. 

***Klasa Datum:***

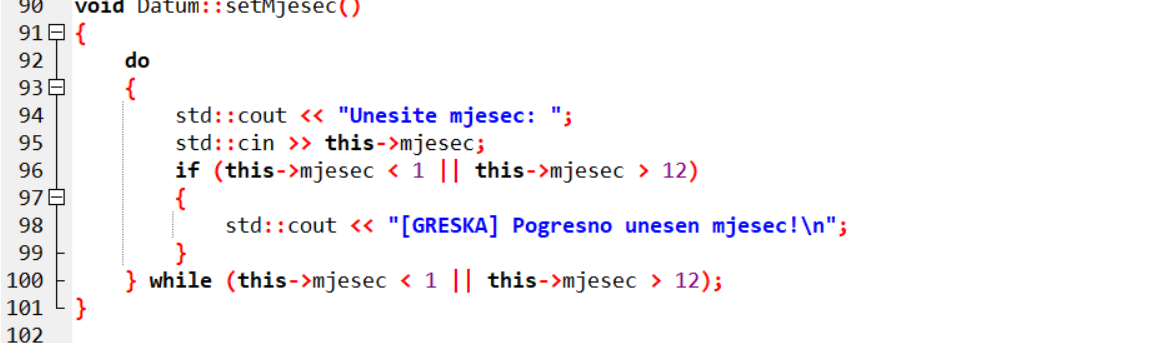
Klasa **Datum** kao što i samo ime govori, odnosi se na mogućnost unosa datuma. Svi članovi privatne klase su integer, jer se radi o unosu brojeva. U konstruktoru se inicijalizira datum na 01.01.1900. godine.

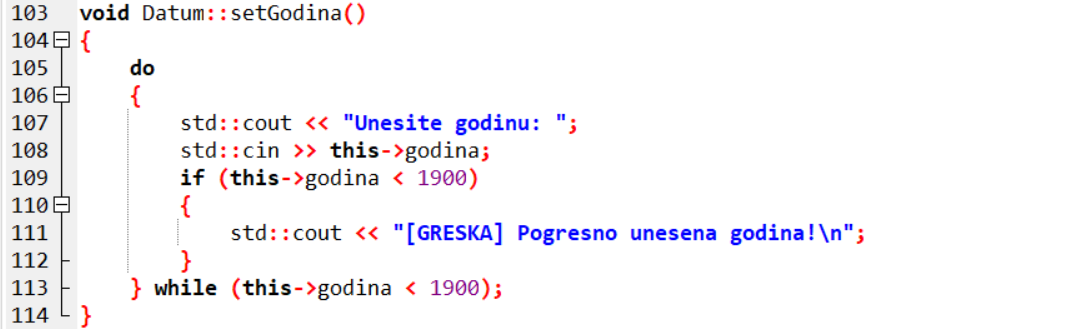


Prve tri funkcije su seteri gdje se na određeni način postavljaju dani, mjeseci i godina. Prva funkcija jeste **setDan ()** koja je izrađena tako da se na osnovu unesenog mjeseca odluči koji dan je zapravo ispravan, odnosno da se onemogući unos pogrešnog dana, npr. 32, -5 itd.

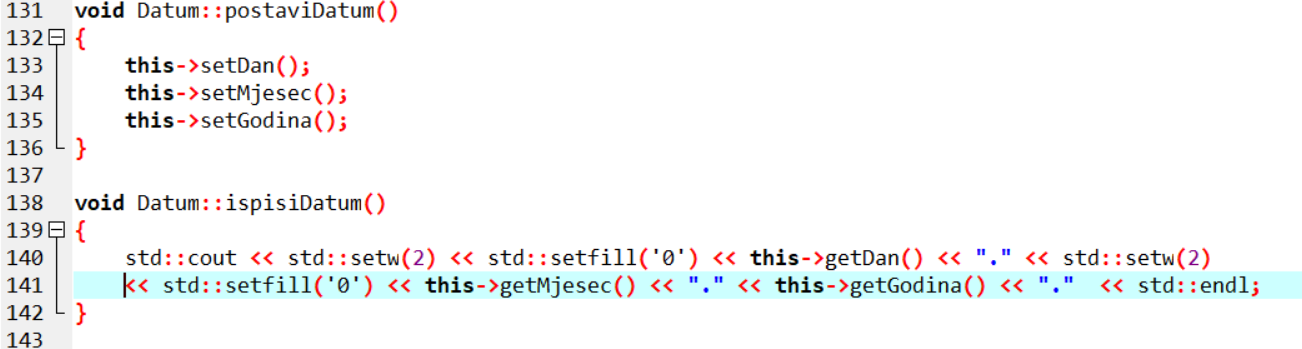
Kao što se vidi na slici, za mjesece koji imaju 31 dan postavila su se ograničenja u while petlji sve dok se ne zadovolji uslov, odnosno dok dati datum nije manji od 1 i veći od 31. Na istom principu je zasnovan i unos za mjesece koje imaju 30 dana, ili 28 odnosno 29 dana za februar (ovisno da li je godina prstupna ili nije).

Sljedeća funkcija jeste **setMjesec()**, koja je vrlo jednostavna funkcija, gdje unos ne smije biti veći od 12 niti manji od 1. Unos se ponavlja dok se uslov ne zadovolji.



Funkcija **setGodina ()** ima ograničenje da se ne može unijeti godina manja od 1900. 

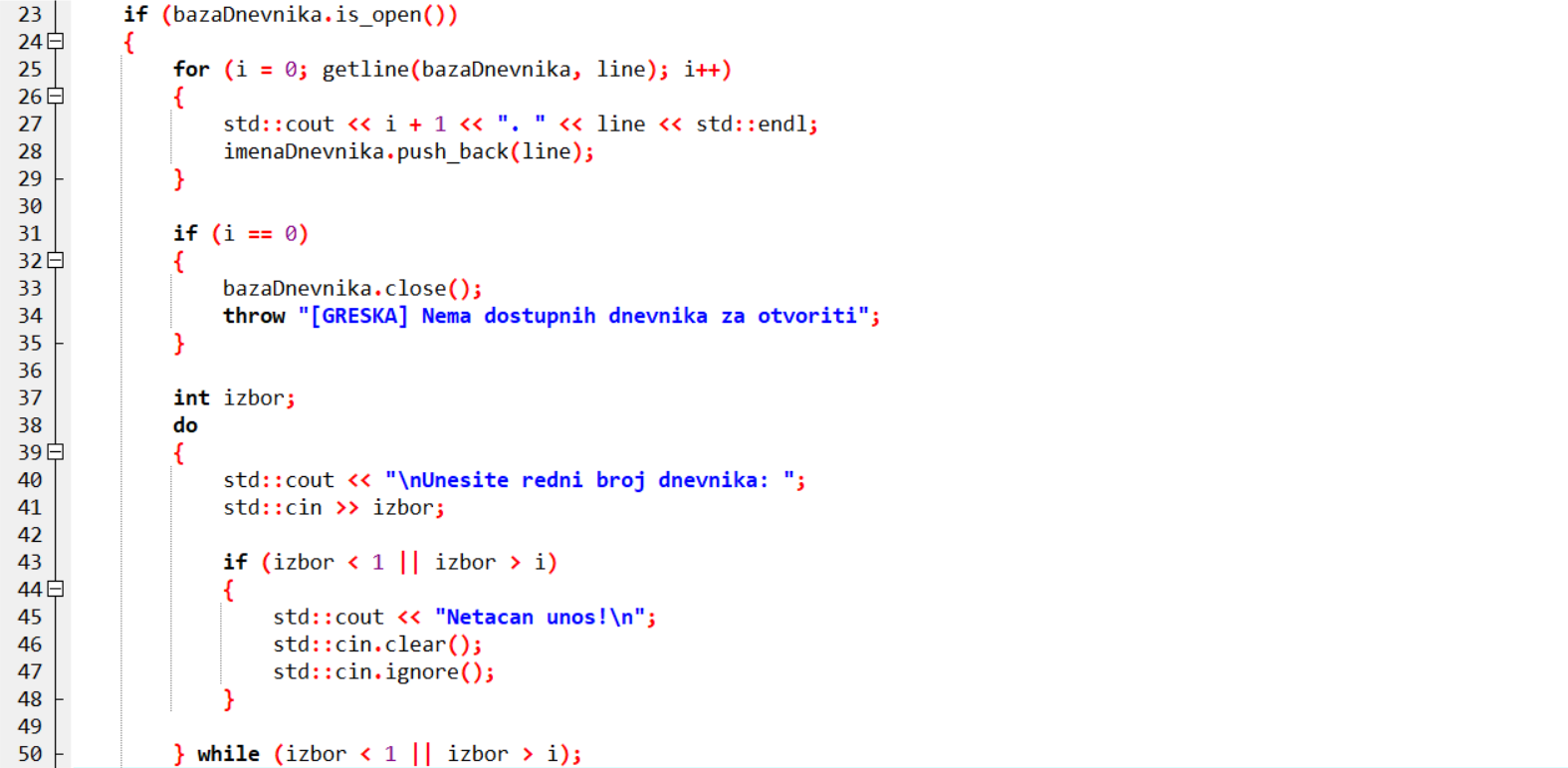
Nakon toga slijede getteri, koji vraćaju tačno unesene vrijednosti dana, mjeseca i godine. Pored toga imamo i generisane funkcije za ispis datuma.



***Klasa Dnevnik:***

Sljedeća kalsa u nizu, jeste jedna od važnijih klasa, a to je klasa **Dnevnik**. Klasa Dnevnik nam postavlja i vraća informaciju o stanju dnevnika, odnosno koji nam je dnevnik trenutno otvoren. Shodno tome, prva funkcija koju ćemo objasniti jeste funkcija **setImetrenutnogDnevnika ()**.

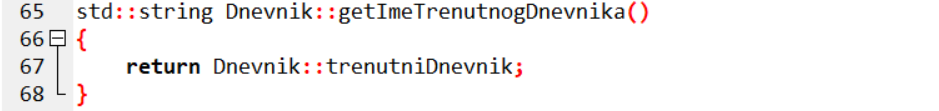
Kao što i samo ime govori, ova funkcija postavlja trenutni dnevnik koji želimo otvoriti. U prvom dijelu funkcije imamo opciju odabira dnevnika, jer u sistemu se nalazi više dnevnika koji imaju različite namjene. Svi dnevnici koji se mogu otvoriti nalaze se u posebnoj datoteci koju je potrebno otvoriti, iščitati i prikazati korisniku postojeće dnevnike, a potom korisnik unos broj dnevnika koji želi otvoriti, s time da uneseni broj ne može biti manji od 1 niti veći od trenutnog broja dnevnika u datoj datoteci.



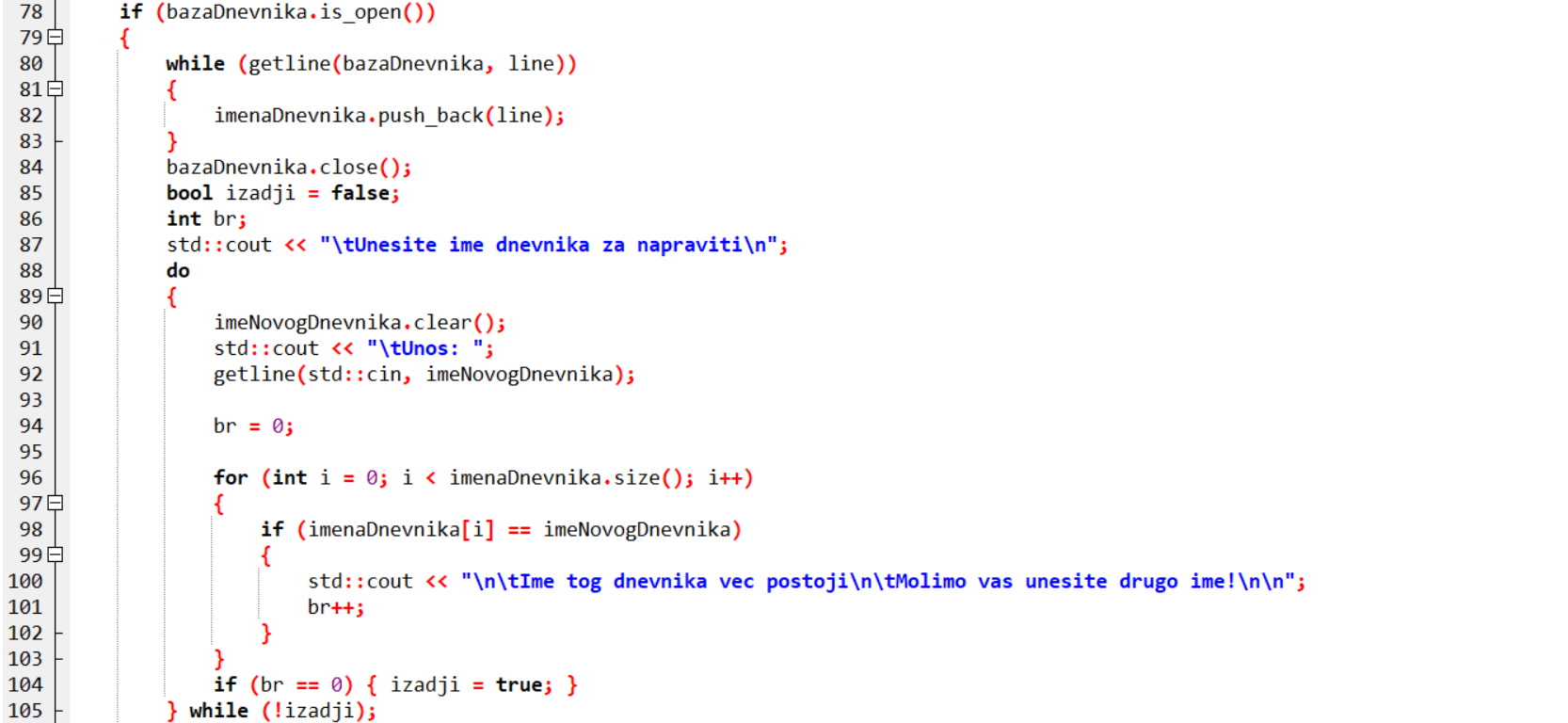
Nakon što korisnik izabere odgovarajući dnevnik, bira se pozicija tog dnevnika u vektoru (u koji su prethodno smješteni nazivi dnevnika) i dodaje mu se ekstenzija .txt. Varijabla u koju se smješta ime dnevnika koju je korisnik odabrao je statička varijabla.



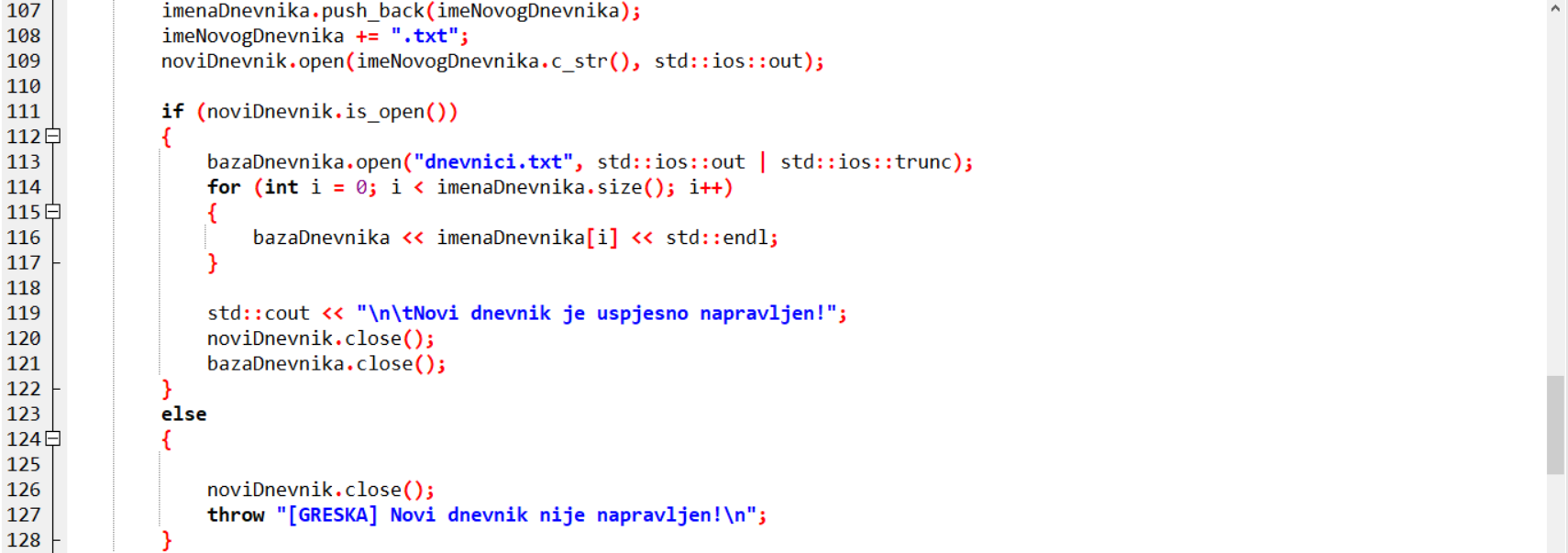
Sljedeća funkcija je statičkog tipa koja vraća ime trenutnog dnevnika, koji smo postavili u prethodnoj funkciji. Ta funckija se naziva **getImeTrenutnogDnevnika ()**.

****

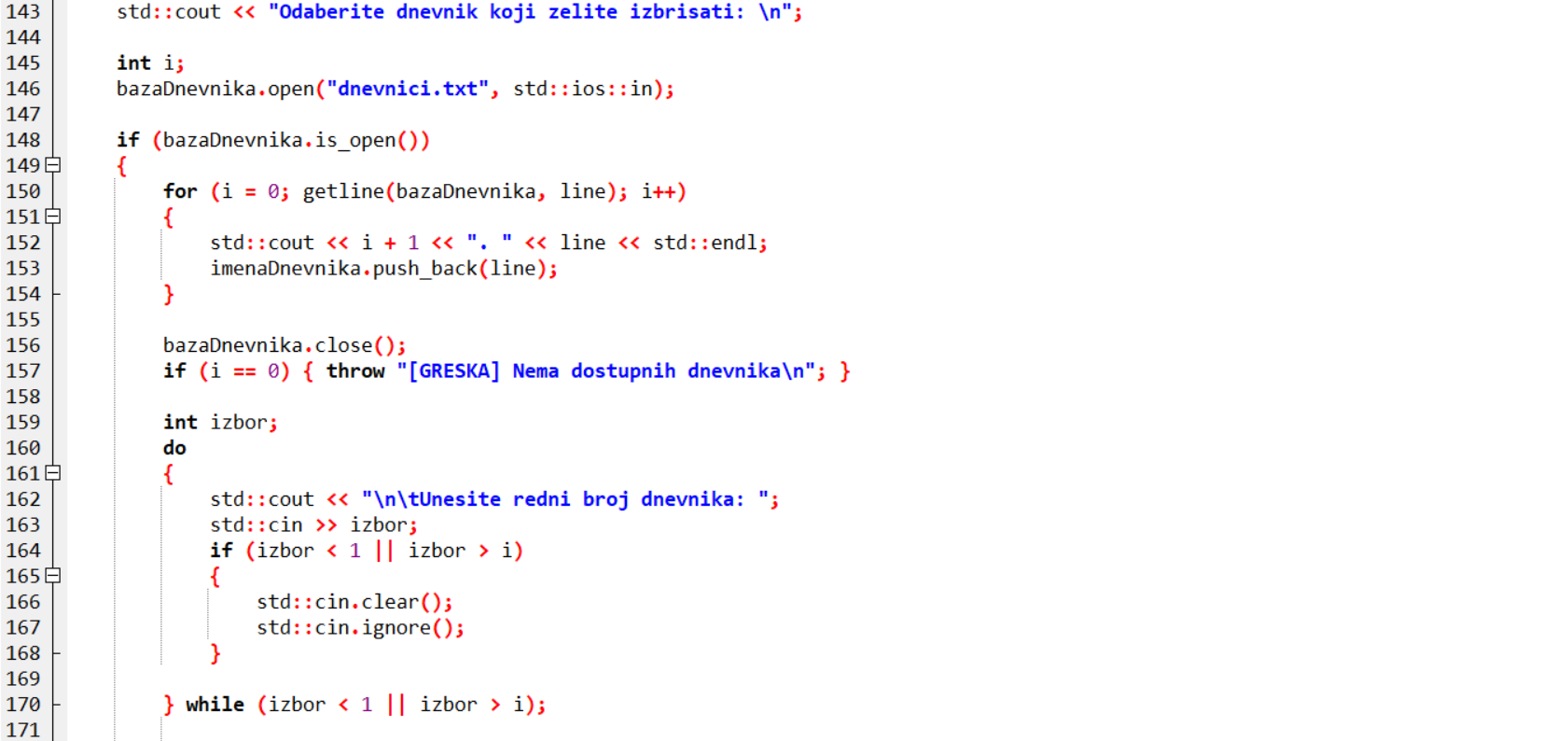
Naravno, u jednom ovakvom sistemu za vođenje blagajničkih dnevnika, moramo imati opciju kreiranja novih dnevnika. Shodno tome, imamo i funkciju **napraviDnevnik ()**. Da bi se kreirao dnevnik, prvo moramo itvoriti datoteku u koju su smještana imena drugh dnevnika, zatim imena tih već postojećih dnevnika smještamo u zaseban vektor kako bi korisniku koji želi kreirati novi dnevnik, da li to ime za novi dnevnik već postoji. Ukoliko ime za kreiranje novog dnevnika već postoji, onemogućava se korisniku kreiranje novog dnevnika i traži novi unos.



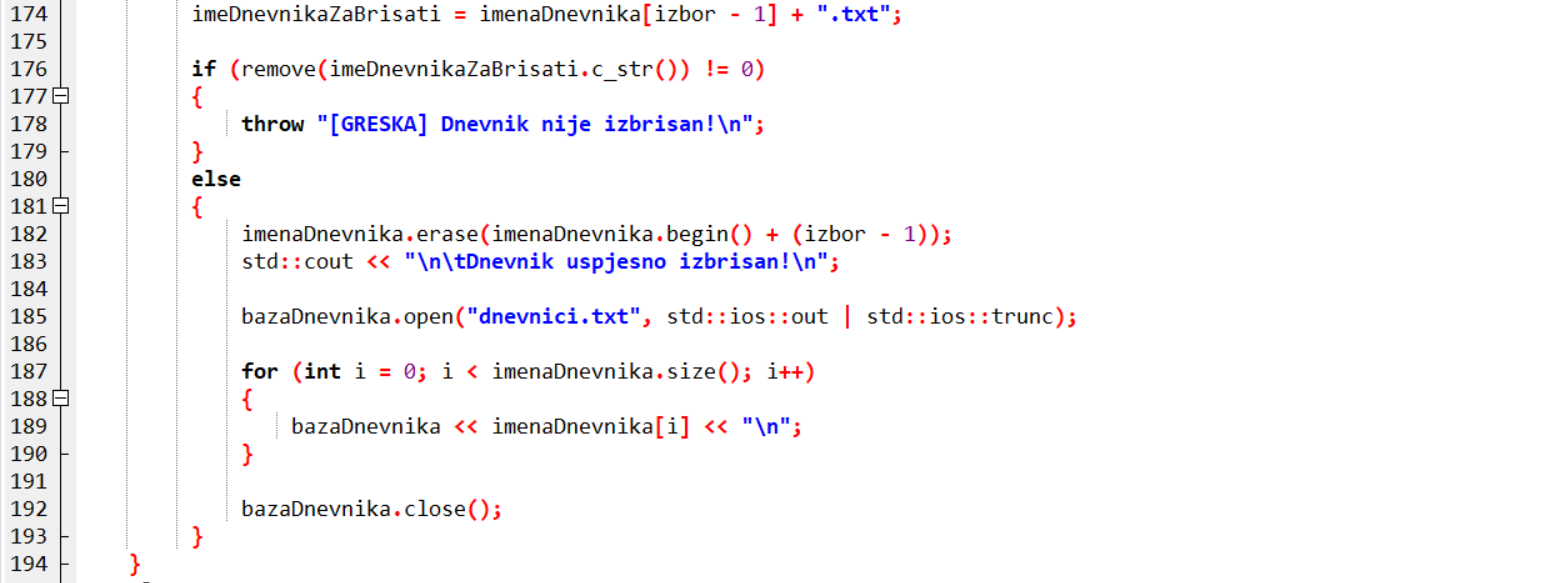
Nakon što je korisnik unio zadovoljavajuće ime, to ime dnevnika se smješta u rethodno kreirani vektor s ostalim imenima dnevnika. Nakon toga, na ime novog dnevnika dodaje se esktenzija .txt, novi dnevnik se otvara (kreira se datoteka), te se u prethodnu datoteku u kojoj su se nalazila imena prethodno upisanih dnevnika, dodaje i ime novog kreiranog dnevnika.



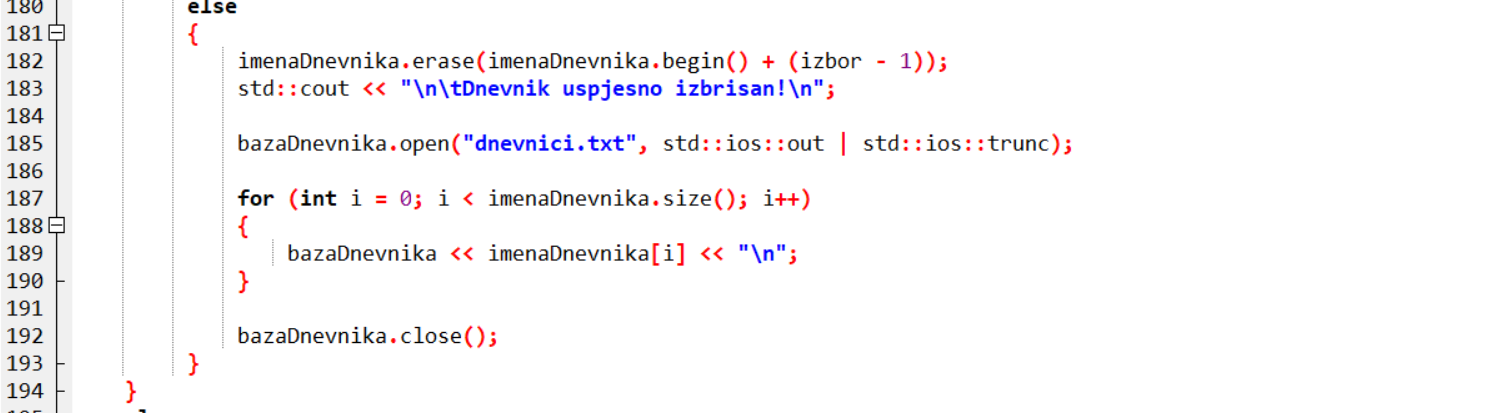
Ponekad postoji i potreba da se neki dnevnik izbrise iz nekog sistema, pa shodno tome postoji funkcija pod nazivom **izbrišiDnevnik ()**. U ovoj funkciji korisniku se izlistaju svi dnevnici koji se nalaze u sistemu, pa se kao i u prethodnoj funkciji imena već postojećih dnevnika smještaju u vektor, nakon čega se korisniku omogućava unos broja dnevnika koji želi izbirsati.



Nakon toga, datom imenu dnevnika koju je korisnik izabrao, pridružuje se .txt ekstenzija, gdje se uz pomoć remove funkcije u potpunosti briše izabrana dnevnika (ne briše se samo sadržaj datoteke, nego se briše i sama datoteka iz foldera).

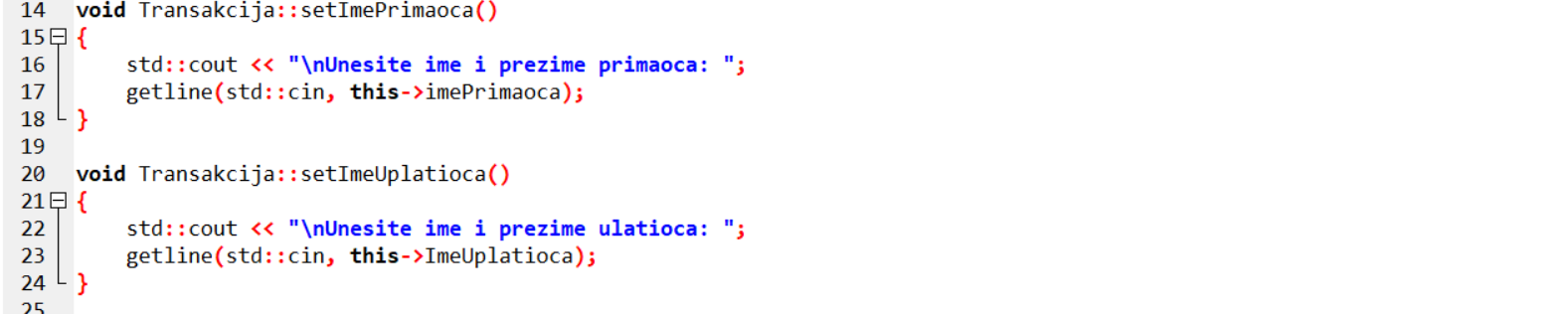


Nakon toga, ime dnevnika koje je korisnik odabrao briše se iz vektora i briše se i iz datoteke u kojoj su bili smješteni nazivi dnevnika.

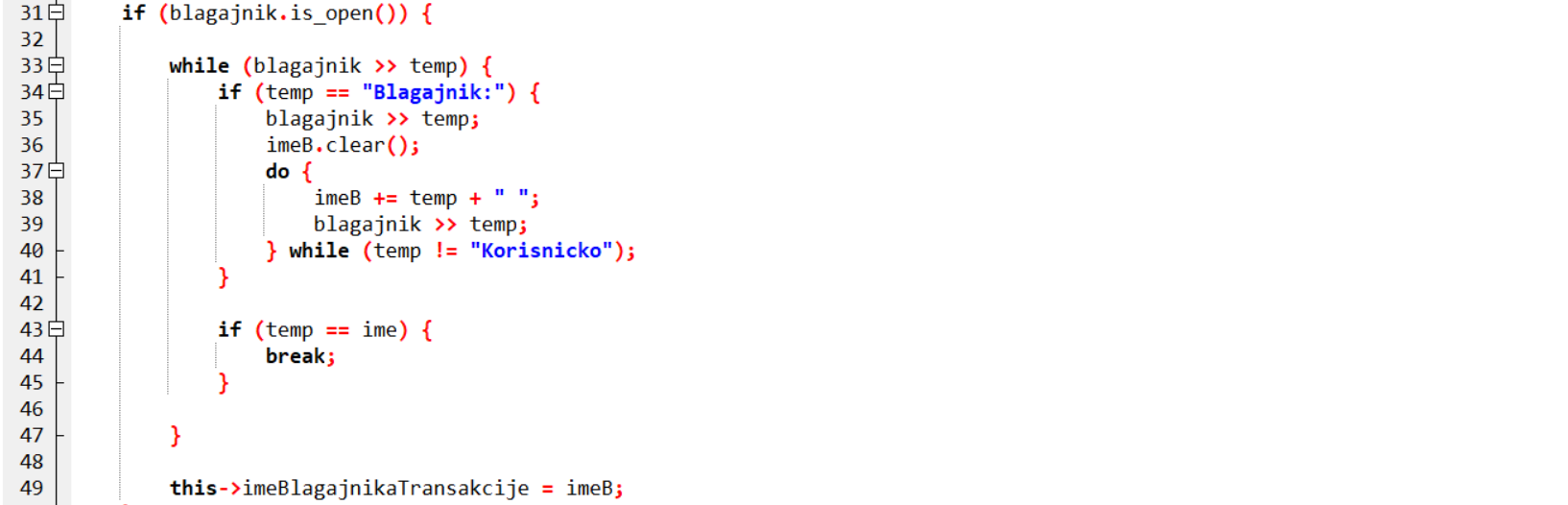


***Klasa Transakcija:***

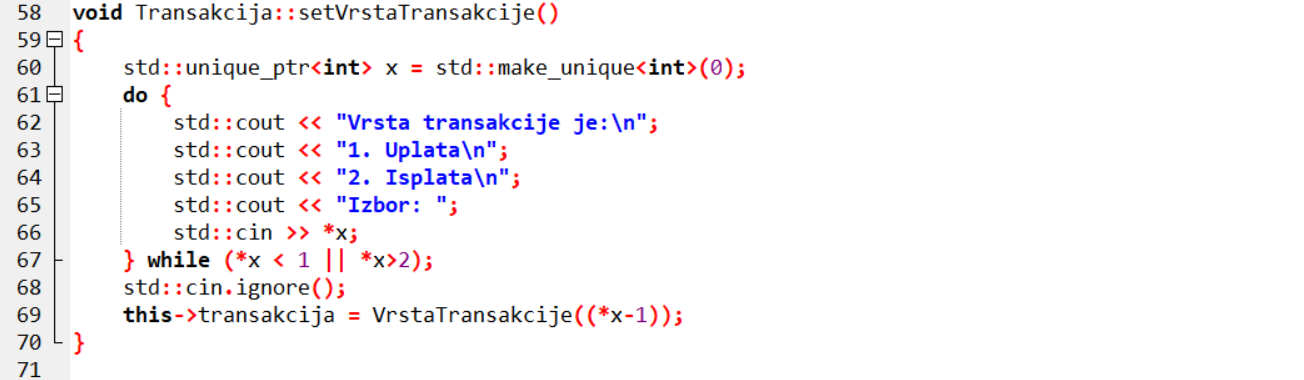
U klasi **Transakcija** obavljajuje se sve one najbitnije i najvažnije stvari vezane za blagajničko poslovanje. Najednostavnije funkcije su setImePrimaoca i setImeUplatioca koje kao što im i samo ime kaže postavlja se ime primaoca i ime uplatioca.



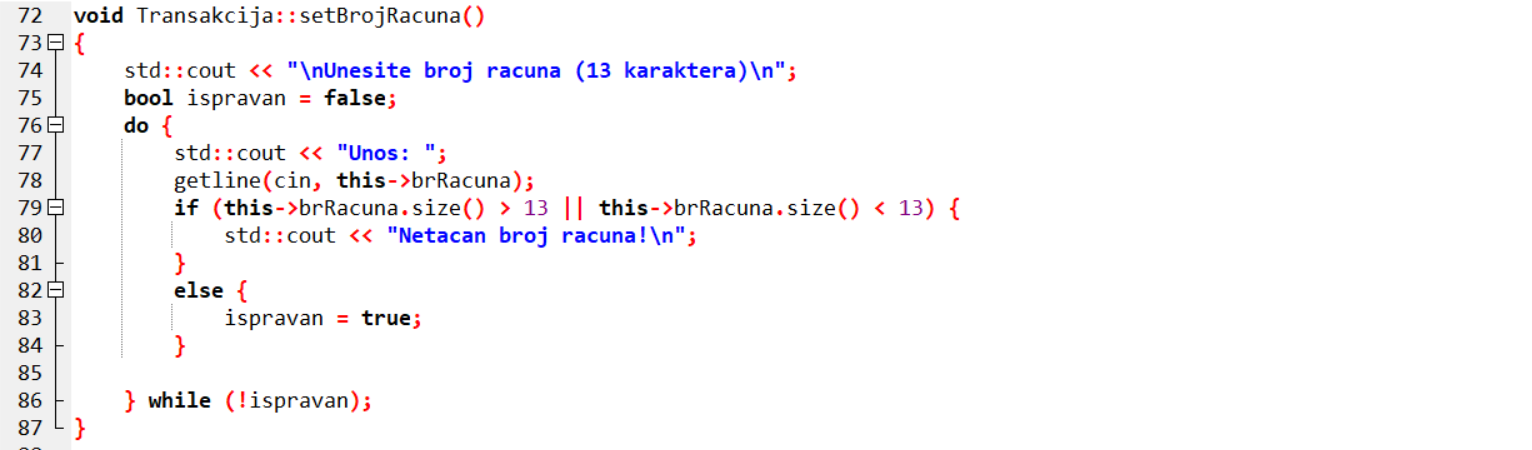
Sljedeća funkcija jeste funkcija **setImeBlagajnikaTransakcije** koja postavlja ime blagajnika koji se prijavio u sistem, i on prima parametar koji je tipa string, koji će se proslijediti iz main-a. Na osnovu tog stringa pronalazimo ime i prezime blagajnika koji se prijavio u sistem. Na osnovu proslijeđenog korisničkog imena, mi iz datoteke u koju su smješteni podaci o blagajnicima pronalazimo ime i prezime blagajnika koji se prijavio u sistem i smještamo ga u posebnu varijablu koja je deklarisana u datoj funkciji.



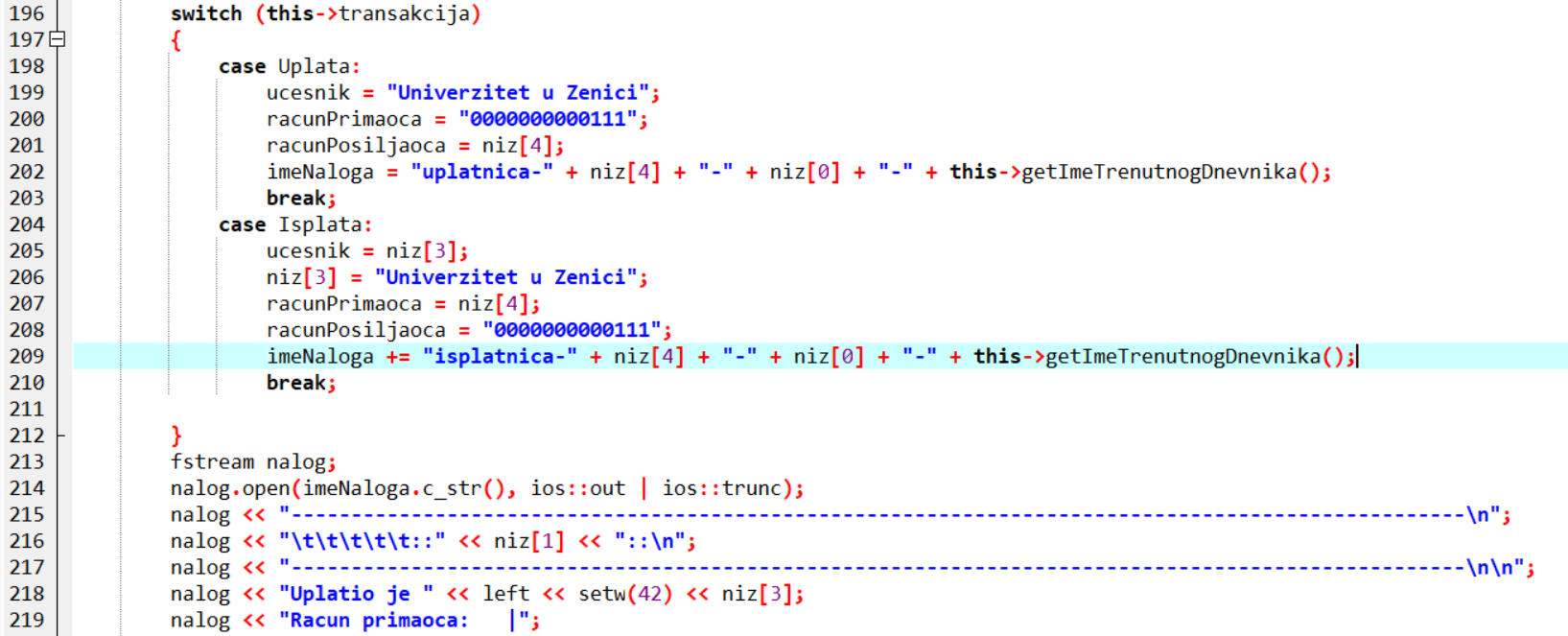
Sljedeći setter je funkcija u kojj se postavlja vrtsa transakcija (uplata ili isplata), gdje smo koristli pametne pokazivače u koje smo unosili redni broj transakcije koju želimo da izvršimo. Za to smo koristili enumeraciju.

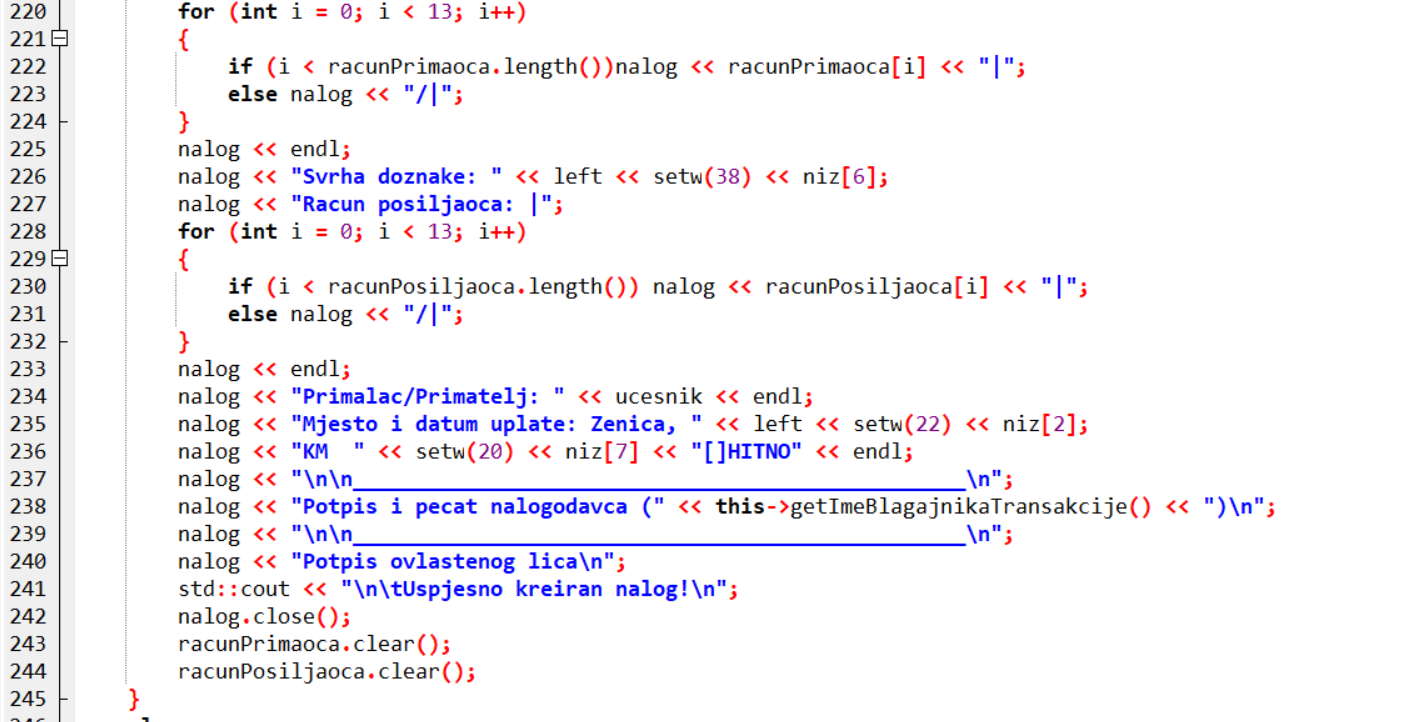


Ono najvažnije prilikom blagajničkog poslovanja jesu zapravo brojevi računa na koji će se vršiti uplata, odnosno isplata. U našoj funkciji **setBrojRačuna** u kojem se unosi korisnikov broj računa, s ograničenjem da broj računa ne može imati manje ili više od 13 brojeva, odnosno broj računa mora imati 13 brojeva.



Sljedeća funkcija jeste **ispisNaloga** koja ispisuje uplatnicu, odnosno isplatnicu u obliku datoteteke s esktenzijom .txt koja se može printati. Da bi korisnik znao koji nalog treba da isprinta, prvo mu se mora ispisati sadržaj renutnog dnevnika po redoslijednim brojevima. Nakon toga, pnovo uz pomoć pametnog pokazivaća, tražimo od korisnika da unese redni broj naloga koji želi da isprinta. Nakon što korisnik unese redni broj, na osnovu podataka unesenih u dnevniku za transakciju, sistem kreira uplatnicu odnosno isplatnicu ovisno o kojoj vrsti transakcije je riječ. Kada se odluči koja vrsta transakcije, onda se podaci vezani za taj redni broj u dnevniku povlače i upisuju se u novu datoteku koja se kreira, koja zapravo predstvalja tu uplatnicu odnosno isplatnicu.

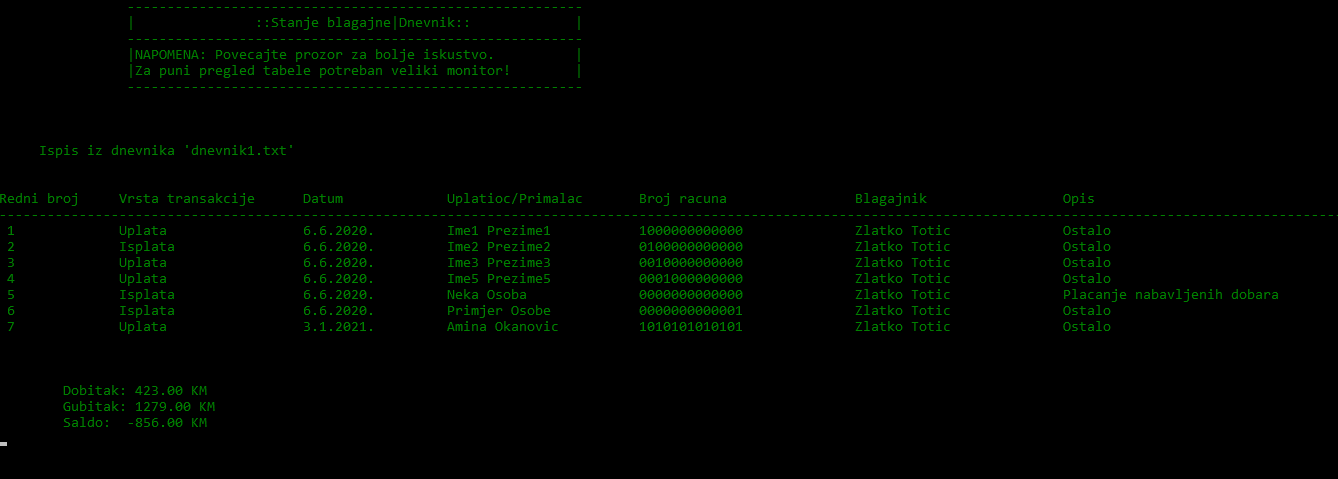




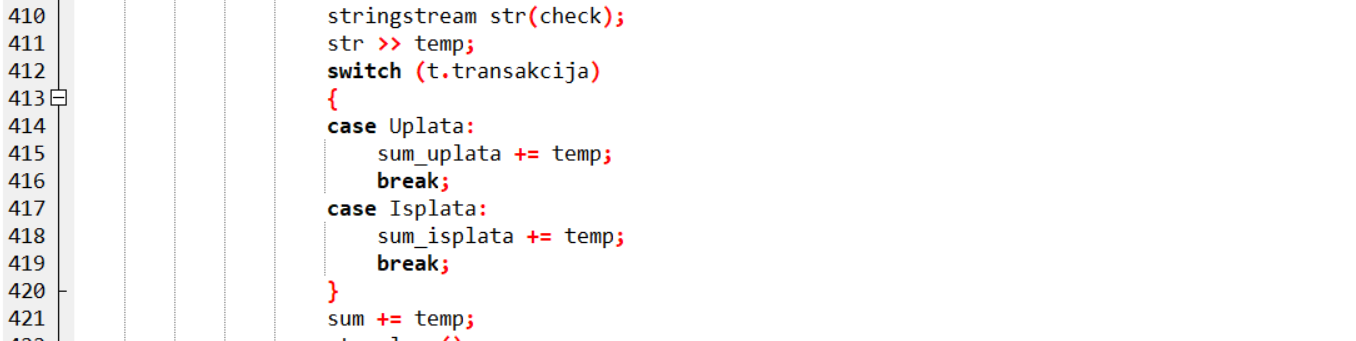
Nakon ispisa naloga, slijede getteri za sve prethodno unesene settere.



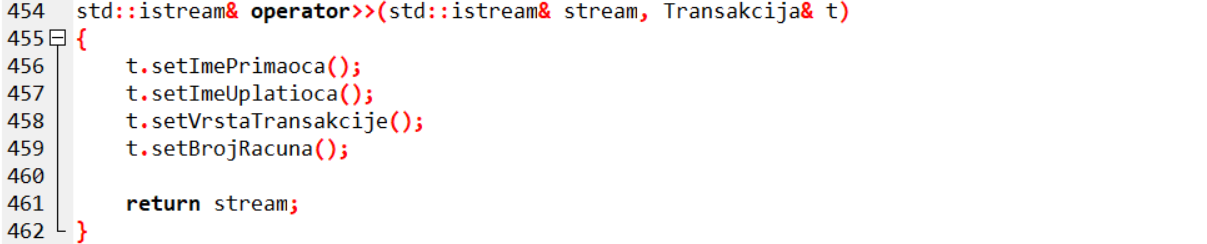
Sljedeći jeste operator za ispis transakcija. Dati operator ispisuje sve transakacije koje su se prethodno upisali u trenutno otvoreni dnevnik. Pored toga, ispisuje se dobitak, gubitak i saldo za sve obavljenje transkacije. Gubitak predstavljaju sumu isplata, dobitak predstavlja sumu uplata, a saldo predstavlja razliku između navedene dvije. Kada ispisujemo iz dnevnika, postoji šablon koji je identičan onome na upisu podataka.



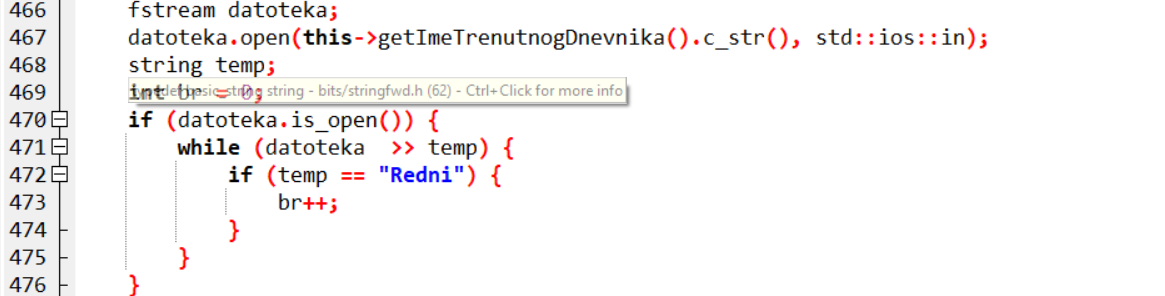
Da bismo izračunali dobitak, gubitak i saldo, potrebno je string pretvoriti u float. Da bismo pretvorili u string u float koristit ćemo stringstream funkciju.



Sljedeći operator jeste operator za unos podataka. U ovom operatoru se pozivaju prethodno kreirani setteri i vraćaju se ti podaci koji su zapravo upisani.



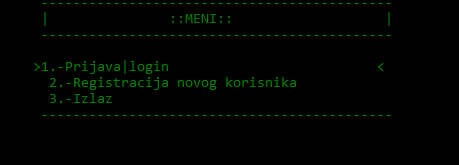
Zadnja funkcija u klasi **Transakcija** jeste funkcija koja se odnosi na unos isplata i uplata u trenutni dnevnik. Prvo u ovoj funkciji, otvara se dnevnik koji je korisnik odabrao. Nakon toga prebrojavaju se transakcije u datom dnevniku, kako bi se znao redni broj transakcije koja će sljedeća biti upisana u dnevnik.



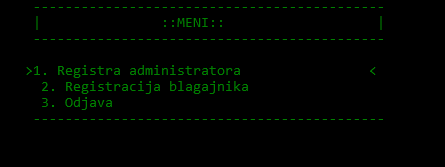
Nakon toga, korisniku se daje opcija odabira transakcije, isplate ili uplate, zatim se unosi trenutni datum, ime uplatioca ili ime primaoca (pošto je sistem rađen za Univerzitet u Zenici, unosi se samo ime računa uplatioca/primaoca dok je fiksan broj računa Univerziteta u Zenici), broj računa, iznos i svrha same transakcije. Nakon što je korisnik sve to unio, svi ti podaci zapisuju se u dnevnik.

***Main.cpp***

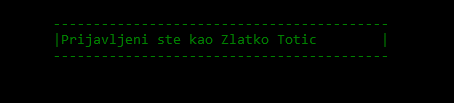
U main.cpp nalaze se meniji. Sva tri glavna menija rade na principu menija koji se lista uz pomoć strelica ili W i S slova za kretanje po meniju. To je zapravo implementirano u funkciji menu koja prima kao paramtar vektor tipa string. Također na ulazi i izlazu programa imamo loading bar. Prvi meni jeste ulazni meni koji se pojavljuje prilikom pokretanja programa i u njemu su sadržani registracija, login i izlaz.



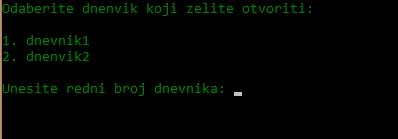
Drugi meni je meni koji se pojavljuje kada želimo registrovati novog korisnika koji nam nudi opciju registracije administratora ili blagajnika.



Treći meni je meni koji se tiče svih ostalih opcija koji se vežu za blagajničko poslovanje. Prije nego što se korisniku učita program za blagajnik, prikaže se ime korisnika koji je izvršio prijavu u sistem.



Nakon toga, korisnik ima opciju odabira dnevika.



Na kraju, učitava se meni s opcijama vezanim za blagajničko poslovanje.

