UNIVERZITET U NOVOM SADU
FAKULTET TEHNIČKIH NAUKA
NOVI SAD
Departman za računarstvo i automatiku
Odsek za računarsku tehniku i računarske komunikacije

ISPITNI RAD

Kandidat: **Tamara Cvjetković**
Broj indeksa: **SV48/2022**

Predmet: **Objektno orijentisano programiranje 2**
Tema rada: **Sudoku Izazov**

Mentor rada: **dr Miodrag Đukić**

Novi Sad, Januar, 2024.

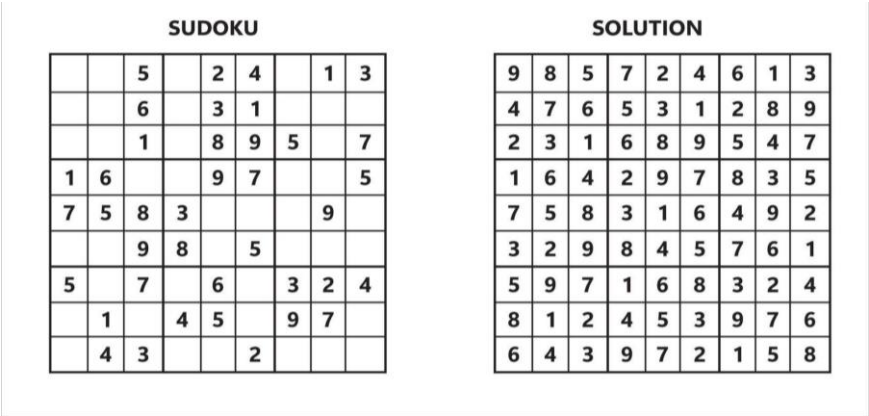
SADRŽAJ

SPISAK SLIKA

1. Uvod

1.1. Sudoku

Sudoku (doslovno 'jednocifren'; originalno nazvan Number Place) je logička, kombinatorna igra sa postavljanjem brojeva. U klasičnom Sudoku-u, cilj je popuniti tablu veličine 9 × 9 cifara tako da svaka kolona, svaki red i svaka od devet podmatrica veličine 3 × 3 sadrži sve cifre od 1 do 9. Postavka zagonetke je djelimično popunjena tabela koja, za dobro postavljenu zagonetku, ima jedinstveno rešenje.



SUDOKU

		5		2	4		1	3
		6		3	1			
		1		8	9	5		7
1	6			9	7			5
7	5	8	3				9	
		9	8		5			
5		7		6		3	2	4
	1		4	5		9	7	
	4	3			2			

SOLUTION

9	8	5	7	2	4	6	1	3
4	7	6	5	3	1	2	8	9
2	3	1	6	8	9	5	4	7
1	6	4	2	9	7	8	3	5
7	5	8	3	1	6	4	9	2
3	2	9	8	4	5	7	6	1
5	9	7	1	6	8	3	2	4
8	1	2	4	5	3	9	7	6
6	4	3	9	7	2	1	5	8

Slika 1: Jedno rješenje Sudoku Zagonetke

Pretpostavlja se da za tablu veličine 9 × 9 postoji 6,670,903,752,021,072,936,960 mogućih rješivih sudoku zagonetki. Za određenu postavku Sudoku table može da postoji više od jednog rješenja, ali se dobrom postavkom Sudoku table smatra ona za koju se može pronaći samo jedno, unikatno rješenje. Da bi Sudoku zagonetka imala unikatno rješenje, potrebno je da je tabela popunjena sa minimalno 17 početnih otkrivenih polja. U svijetu su popularne i table veličine 3 × 3, 4 × 4 ili čak 12 × 12.

Postoje različite težine Sudoku zagonetki. Nivo težine Sudoku zagonetki može se opisati na osnovu broja trikova koje treba otkriti da bi se zagonetka riješila:

- Ako se zagonetka može riješiti samo sa sakrivenim „singlovima“, onda je to Sudoku sa lakim nivoom.
- Ako se zagonetka sastoji samo od parova i trojki, njen izazov se povećava.

Broj otkrivenih polja u postavci zagonetke takođe utiče na težinu Sudoku zagonetke. Smatra se da laki nivo sadrži oko 35-45 otkrivenih polja, a teški nivo oko 25-26.

Najteži Sudoku veličine 9 × 9 je specifična tabla koju je napravio Arto Inkala 2012. godine i ona se smatra najtežom rješivom Sudoku zagonetkom na svijetu.

8								
		3	6					
	7			9		2		
	5				7			
				4	5	7		
			1				3	
		1					6	8
		8	5				1	
	9					4		

Slika 2: Najteža Sudoku 9 × 9 zagonetka na svijetu

1.2. Zadatak

Realizovati program koji će olakšati korisnicima u rješavanju Sudoku zagonetke. Potrebno je da se za program omogući učitavanje i čuvanje Sudoku zagonetke u datotekama, provjera ispravnosti zadane Sudoku zagonetke, automatsko rješavanje zagonetke, ali i generisanje nove validne postavke zagonetke, kao i ispis statističkih informacija u konzolnom meniju.

Učitavanje i čuvanje neke Sudoku zagonetke podrazumijeva čitanje i pisanje u datoteku u tekstualnom formatu. Program treba da rukuje datotekama čiji su nazivi dati kroz *argumente komandne linije*. *Provjeru ispravnosti, automatsko rješavanje i generisanje nove* Sudoku zagonetke realizovati uz pomoć nekog odabranog algoritma, uz strogo poštovanje sva tri pravila igre Sudoku. Razmisliti o dodavanju optimizacije. *Ispis statističkih* informacija predstavlja ispis generisanih informacija nakon svake odigrane partije, kao i ispis formatirane Sudoku tabele.

Izvršiti testiranje pisanjem dodatnih modula za provjeru svih funkcionalnosti programa.

2. Analiza problema

Analiza problema podrazumijeva detaljno razmatranje problema implementacije pri rješavanju problema.

Za *Učitavanje i čuvanje* Sudoku zagonetke, potrebno je korektno učitavanje i čuvanje zagonetke koju korisnik unese u datoteku koja je određena preko argumenata komandne linije, kao i učitavanje i čuvanje Sudoku zagonetke generisane od strane programa u prethodno navedene datoteke. Format datoteke treba da bude jasan i pregledan, a brojevi u tabli vidno razmaknuti i podijeljeni u podmatrice.

Provjera ispravnosti Sudoku zagonetke se odnosi na provjeru broja ponavljanja svakog broja od 1 do 9 u svakom redu, koloni i podmatrici. Da bi Sudoku zagonetka bila validna, potrebno je da se svaki broj od 1 do 9 ponavlja najviše jednom u svakom redu, koloni i podmatrici, u suprotnome, zagonetka neće biti validna i ta greška će se računati.

Za *Automatsko rješavanje* Sudoku zagonetke je potrebno da se implementira algoritam koji će brzo, ali efikasno riješiti datu Sudoku zagonetku. Preporučuje se i implementacija neke vrste optimizacije, jer bi mogla dosta doprinijeti brzini našem automatsko rješavanju zagonetke. Postoji više algoritama koje možemo razmotrit: backtracking sa brojevima od 1 do 9 ili backtracking pomoću bitmaski (praćenje zauzetih brojeva u svim redovima, kolonama i podmatricama), a neka rješenja koriste i grafove. Jedan od načina optimizacije bi uključivalo pamćenje mjesta ili broja pojavljivanja svakog broja u svakom redu, koloni i podmatrici.

Generisanje nove Sudoku zagonetke zahtijeva generisanje isključivo rješivih zagonetki. Postoji više algoritama koji su validni i efikasni: rješavanje zagonetke pa brisanje određenog broja polja, postavljanje brojeva nasumično pa provjeravanje da li je zagonetka validna ili popunjavanje prvo glavnih dijagonala podmatrica, pa onda sporednih polja.

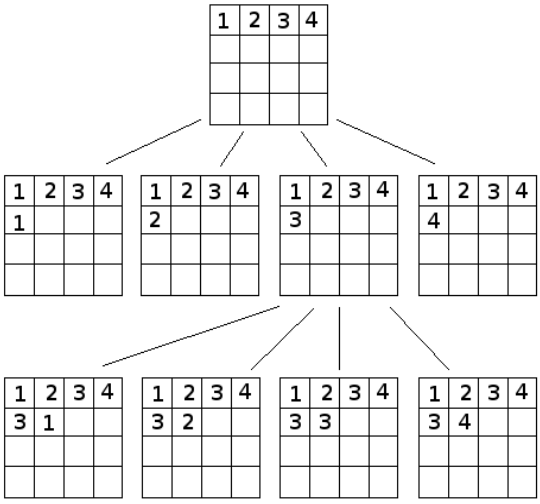
Ispis statističkih informacija predstavlja ispis generisanih informacija nakon svake odigrane partije, kao i ispis formatirane Sudoku tabele.

3. Koncept rešenja

Za *Učitavanje i čuvanje* Sudoku zagonetke koristimo dvije datoteke, jedna iz koje čitamo korisnikovu postavku zagonetke ili u koju upisujemo generisanu postavku Sudoku zagonetke i druga datoteka iz koje čitamo korisnikovo rješenje zagonetke ili u koju upisujemo automatski rješenu Sudoku zagonetku. Nazivi ove dvije datoteke su date kroz argumente komandne linije. Format datoteka sadrži dosta neupotrebljivih karaktera, radi ljepšeg izgleda i lakšeg čitanja zagonetki. Brojevi su karakterima podijeljeni u podmatrice, a svaki broj u jednoj podmatrici je razdvojen razmakom. Prazan karakter označava da ne postoji broj na tom polju. Svi ostali karakteri koji služe samo za vizualizaciju se preskaču tokom iteracije.

Provjera ispravnosti Sudoku zagonetke u ovom rješenju će koristiti tri pomoćne strukturu (mape) gdje će svaka mapa predstavljati brojač ponavljanja brojeva od 1 do 9 redom u svakom redu, koloni i podmatrici. U slučaju da se desi da je negdje nekom broju broj ponavljanja veći od 1, Sudoku zagonetka neće biti validna i greška će se računati, u suprotnom će biti validna i broj grešaka će biti 0.

Automatsko rješavanje Sudoku zagonetke je jedna od najvažnijih ali i najkompleksnijih funkcionalnosti u ovom projektu. Kako bi se postiglo brzo, ali i efikasno rješavanje Sudoku zagonetke, implementiran je rekurzivni algoritam (backtracking), koji za svako početno izabrano prazno polje u tabli provjerava da li se može staviti neki nasumičan broj iz permutacije od 9 brojeva. Mape kao brojači ponavljanja brojeva optimizuju ovaj algoritam i izbjegavaju ponovno računanje tih ponavljanja tokom rekurzije.



Slika 3: Koncept rješenja automatskog popunjavanja zagonetke (backtracking)

Generisanje nove Sudoku zagonetke mora da omogući različitu, ali ispravnu postavku Sudoku zagonetke koja je rješiva. Odabran je algoritam koji će prvo riješiti praznu tablu pomoću automatsog rješavanja Sudoku zagonetke, a potom će petljom proći kroz svih 9 podmatrica i za svaku posebno nasumično izabrati broj „k“ polja koja će se izbrisati (postaviti na prazno), kao i nasumično izabrati permutaciju od prvih 9 brojeva, od kojih ćemo u tabli brisati prvih „k“ brojeva. *Primjer*: trenutno se nalazimo u prvoj podmatrici, izabran je broj k = 3, a permutiacija je p = {2, 5, 1, 6, 4, 7, 8, 9, 3}. Prolazimo kroz sva polja prve podmatrice i brišemo samo prvih k-brojeva iz permutacije u Sudoku tabli, tj. polja koja sadrže brojeve 2, 5 i 1 će biti izbrisana.

Ispis statističkih informacija obavljamo pomoću funkcionalnosti koja provjerava ispravnost Sudoku zagonetke, koja nam govori da li je tabla validna i broj grešaka koje se nalaze u tabli, ali i ažurira broj tačnih polja u Sudoku zagonetci. Broj odigranih Sudoku igri računamo pomoću statičnog brojača u klasi Sudoku, koji se poveća za 1 kada god je napravljena nova instanca klase Sudoku.

4. Opis rešenja

4.1. Sudoku program – moduli i osnovne metode

4.1.1. Modul glavnog programa (main)

4.1.1.1. Funkcija za pokretanje programa (main)

```
int main(int argc, char** argv);
```

Glavna funkcija programa. Predstavlja početak programa u kojem korisnik interaguje sa programom. Podrazumijeva prikaz konzolnog menija, korišćenje svih potrebnih funkcionalnosti, odabir korisničkih opcija, kao i rukovanje datotekama koje su proslijeđene kroz „argv“. Kroz argumente komandne linije treba da budu prosleđeni nazivi dva fajla, od kojih je jedan za čitanje postavke korisnikove zagonetke ili za čuvanje generisane Sudoku zagonetke, a drugi za čitanje korisnikovog rješenja zagonetke ili za čuvanje automatski riješene Sudoku zagonetke.

Primjer: u argumente komandne linije proslijedimo fajlove koji već postoje u Resources Filex: sudokuLoadFile.txt sudokuSolutionFile.txt

- postavka Sudoku zagonetke će se čuvati u sudokuLoadFile.txt

- rješenje Sudoku zagonetke će se čuvati u sudokuSolutionFile.txt

Parametri:

- arcg – broj proslijeđenih argumenata (prvi je uvijek adresa našeg programa),
- argv - argv[1] je za naziv datoteke za postavku, a argv[2] je naziv datoteke za rješenje zagonetke.

Povratne vrednosti:

- 0 – uspješno izvršena,
- ostalo – neuspješno izvršena.

4.1.2. Modul za rukovanje Sudoku igrom (App)

Modul za rukovanje Sudoku igrom se sastoji iz klase App koja apstraktno predstavlja Sudoku igru.

4.1.2.1. Klasa za rukovanje Sudoku igrom (App)

App je klasa koja predstavlja Sudoku igru.

Atributi:

- string loadFile - predstavlja ime fajla iz kog se čitaju Sudoku postavke
- string solutionFile - predstavlja ime fajla u koji se čuvaju rješenja

Metode:

- App();
- App(string lf, string sf);
- string getLoadFile();
- string getSolutionFile();
- void initialMessage();
- int showFirstMenu();
- int showSecondMenu();
- void start();

4.1.2.2. Podrazumijevani konstruktor za klasu App

App();

Podrazumijevani konstruktor za klasu App, postavlja atribute loadFile i solutionFile na prazne stringove.

4.1.2.3. Konstruktor sa dva parametra klase App

App(string lf, string sf);

Konstruktor za klasu App sa dva proslijeđena atributa.

Parametri:

- string lf - vrijednost atributa loadFile
- string sf - vrijednost atributa solutionFile

4.1.2.4. Getter metoda za atribut loadFile klase App

string getLoadFile();

Getter funckija za loadFile.

Povratna vrijednost:

- string - vrijednost atributa loadFile

4.1.2.5. Getter metoda za atribut solutionFile klase App

string getSolutionFile();

Getter funckija za solutionFile.

Povratna vrijednost:

- string - vrijednost atributa solutionFile

4.1.2.6. Metoda za prikaz početka igre

void initialMessage();

Funkcija koja prikazuje početnu, uvodnu poruku na početku Sudoku igre.

4.1.2.7. Metoda za prikaz prvog menija igre

int showFirstMenu();

Funkcija koja prikazuje prvi meni Sudoku igre, u kome je moguće izabrati koji način postavke Sudoku zagonetke želite (čitanje iz datoteke ili generisanje nove od strane programa).

Povratna vrijednost:

- int - broj izabrane opcije (1 ili 2)

4.1.2.8. Metoda za prikaz drugog menija igre

int showSecondMenu();

Funkcija koja prikazuje drugi meni Sudoku igre, u kome je moguće izabrati koji način rješenja Sudoku zagonetke želite (čitanje iz datoteke ili automatsko rješavanje od strane programa).

Povratna vrijednost:

- int - broj izabrane opcije (1 ili 2)

4.1.2.9. Metoda za pokretanje igre

void start();

Funkcija koja pokreće program (Sudoku igra). Pokreće beskonačnu petlju i prikazuje oba menija sve dok korisnik ne odluči da završi igru i izađe iz programa. Prikazani meniji nude mogućnost izbora dvije opcije, a u zavisnosti koja je opcija odabrana, program ili učitava korisnikovu postavku ili rješenje, ili korisnik pušta program da generiše postavku ili automatski riješi postavljenu Sudoku zagonetku.

4.1.3. Modul za rukovanje Sudoku zagonetkom (Sudoku)

Modul za rukovanje Sudoku zagonetkom se sastoji iz klase Sudoku koja predstavlja jednu Sudoku zagonetku ili Sudoku tablu.

4.1.3.1. Klasa za rukovanje Sudoku zagonetkom (Sudoku)

Sudoku je klasa koja predstavlja Sudoku zagonetku ili Sudoku tablu.

Atributi:

- int table[9][9] - predstavlja tablu Sudoku zagonetke
- int correctFieldCnt - brojač tačnih polja u Sudoku tabli
- static int gamesPlayedCnt - brojač odigranih Sudoku igara do određenog trenutka
- map <pair <int, int>, int> rowCnt - mapa koja predstavlja brojač ponavljanja svakog broja od 1 do 9 u svim redovima Sudoku table
- map <pair <int, int>, int> colCnt - mapa koja predstavlja brojač ponavljanja svakog broja od 1 do 9 u svim kolonama Sudoku table
- map <pair <int, int>, int> submatrCnt - mapa koja predstavlja brojač ponavljanja svakog broja od 1 do 9 u svim podmatricama Sudoku table

Metode:

- Sudoku();
- Sudoku(int table[9][9]);
- Sudoku(const Sudoku& s);
- ~Sudoku();
- int(*getTable())[9];
- void setTable(int t[9][9]);
- int getCorrectFieldsCnt();
- void setCorrectFieldsCnt(int c);
- static int getGamesPlayedCnt();
- int getTableField(int i, int j);
- void setTableField(int i, int j, int value);
- void fillMaps();
- void loadSudoku(string filename);
- void saveSudoku(string filename);
- pair<bool, int> checkSudoku();

- void printSudoku();
- pair<int, int> findEmptyField();
- bool isSafe(int i, int j, int num);
- void solveSudoku();
- bool solveSudoku(bool ok);
- void generateSudoku();

4.1.3.2. Podrazumijevani konstruktor za klasu Sudoku

Sudoku();

Podrazumijevani konstruktor za klasu Sudoku, postavlja sva polja Sudoku table i sve ostale atribute na 0. Poziva pomoćnu metodu fillMaps().

4.1.3.3. Konstruktor sa jednim parametrom klase Sudoku

Sudoku(int t[9][9]);

Konstruktor za klasu Sudoku sa jednim proslijeđenim atributa. Ostala polja postavlja na 0. Poziva pomoćnu metodu fillMaps().

Parametri:

- int t[9][9] - predstavlja novu tablu Sudoku zagonetke

4.1.3.4. Konstruktor kopije klase Sudoku

Sudoku(const Sudoku& s);

Konstruktor kopije za klasu Sudoku sa jednim proslijeđenim atributom. Poziva pomoćnu metodu fillMaps().

Parametri:

- const Sudoku& s - referenca na neku drugu instancu klase Sudoku

4.1.3.5. Getter metoda za atribut table klase Sudoku

int(*getTable())[9];

Getter funckija za table.

Povratna vrijednost:

- int(*)[9] - pokazivac na prvi element Sudoku table

4.1.3.6. Setter metoda za atribut table klase Sudoku

void setTable(int t[9][9]);

Setter funckija za table.

Parametri:

- int t[9][9] - predstavlja novu tablu Sudoku zagonetke

4.1.3.7. Getter metoda za atribut correctFieldsCnt klase Sudoku

int getCorrectFieldsCnt();

Getter funckija za correctFieldsCnt.

Povratna vrijednost:

- int - vrijednost atributa correctFieldsCnt

4.1.3.8. Setter metoda za atribut correctFieldsCnt klase Sudoku

void setCorrectFieldsCnt(int c);

Setter funckija za correctFieldsCnt.

Parametri:

- int c - predstavlja novu vrijednost atributa correctFieldsCnt

4.1.3.9. Getter metoda za atribut gamesPlayedCnt klase Sudoku

static int getGamesPlayedCnt();

Getter funckija za gamesPlayedCnt.

Povratna vrijednost:

- int - vrijednost atributa gamesPlayedCnt

4.1.3.10. Getter metoda za polje table

int getTableField(int i, int j);

Getter funckija za tableField.

Parametri:

- int i - predstavlja broj reda u tabli
- int j - predstavlja broj kolone u tabli

Povratna vrijednost:

- int - vrijednost polja u tabli

4.1.3.11. Setter metoda za polje table

void setTableField(int i, int j, int value);

Setter funckija za tableField.

Parametri:

- int i - predstavlja broj reda u tabli
- int j - predstavlja broj kolone u tabli
- int value - nova vrijednost polja u tabli

4.1.3.12. Pomoćna metoda za popunjavanje atributa klase Sudoku

void fillMaps();

Pomoćna funkcija koja optimizuje provjeru Sudoku zagonetke i pronalaženje grešaka u njoj. Tri mape u klasi Sudoku predstavljaju brojače ponavljanja brojeva od 1 do 9 redom u svakom redu, koloni i podmatrici. Na početku se sve vrijednosti postavljaju na 0, a nakon toga se iterira kroz Sudoku tablu i ažuriraju se mape na osnovu rednog broja reda, kolone i podmatrice. Za svako pojavljivanje broja, brojač tog broja na tom mjestu se povećava za 1.

4.1.3.13. Metoda za učitavanje Sudoku zagonetke

void loadSudoku(string filename);

Metoda koja učitava Sudoku iz date datoteke. Automatski ažurira Sudoku tablu i poziva pomoćnu metodu fillMaps(). Čita jedan po jedan red u datoteci i preskače nebitne karaktere. Ažurira Sudoku tablu samo ako naiđe na broj ili na prazno mjesto (koje predstavlja broj 0 u Sudoku tabli).

Parametri:

- string filename - predstavlja naziv datoteke iz koje se čita.

4.1.3.14. Metoda za čuvanje Sudoku zagonetke

void saveSudoku(string filename);

Metoda koja čuva Sudoku u datu datoteku. Upisuje sve nebitne karaktere (radi vizualizacije) i brojeve koji predstavljaju vrijednosti u Sudoku tabli. Ako naiđe na broj 0 u tabli, sačuva ga u obliku praznog mjesta.

Parametri:

- string filename - predstavlja naziv datoteke u koju se čuva.

4.1.3.15. Metoda za provjeru validnosti Sudoku zagonetke

pair<bool, int> checkSudoku();

Metoda koja provjerava da li je Sudoku tabla validna. Ova metoda koristi mape koje su prethodno ažurirane sa pomoćnom metodom fillMaps(). Ako bilo koji broj u tabli u bilo kom redu, koloni ili podmatrici se ponavlja više od jednom (što možemo lako provjeriti preko mapa), Sudoku tabla neće biti validna, a greška će biti uračunata.

Povratna vrijednost:

- pair<bool, int> - par vrijednosti bool i int, gdje prva vrijednost (bool) predstavlja da li je validna Sudoku tabla (true/false), a druga vrijednost (int) predstavlja broj grešaka.

4.1.3.16. Metoda za ispis Sudoku zagonetke

void printSudoku();

Metoda koja ispisuje dobro formatiranu Sudoku tablu.

4.1.3.17. Metoda za pronalaženje praznog polja u Sudoku tabli

pair<int, int> findEmptyField();

Metoda koja iterira kroz cijelu Sudoku tablu i vraća prvo slobodno polje (prvo polje koje ima vrijednost 0).

Povratna vrijednost:

- pair<int, int> - par vrijednosti int i int, gdje prva vrijednost predstavlja red a druga vrijednost predstavlja kolonu u tabli gdje se nalazi prazno polje.

4.1.3.18. Metoda za provjeravanje validnosti postavljanja vrijednosti u polje Sudoku table

bool isSafe(int i, int j, int num);

Metoda koja provjerava da li je validno staviti broj u polje sa određenim koordinatama. Metoda koristi mape koje su prethodno ažurirane sa pomoćnom funkcijom fillMaps(). Nije validno postaviti broj u zadano polje u Sudoku tabli ako je vrijednost tog broja u bilo kojoj od tri mape veća nego 0.

Parametri:

- int i - broj reda u Sudoku tabli
- int j - broj kolone u Sudoku tabli
- int num - vrijednost koja treba da se postavi u dato polje

Povratna vrijednost:

- bool - predstavlja vrijednost koja pokazuje da li je validno staviti dati broj u dato polje (true/false)

4.1.3.19. Pomoćna metoda koja poziva solveSudoku()

void solveSudoku();

Pomoćna metoda koja poziva metodu koja rješava Sudoku zagonetku. Generiše permutaciju od 9 brojeva, koja se čuva u globalnu promjenljivu i koristi u glavnoj solveSudoku() funkciji.

4.1.3.20. Metoda za za rješavanje Sudoku table

bool solveSudoku(bool ok);

Metoda koja rješava Sudoku zagonetku. Implementiran je rekurzivni algoritam (backtracking), koji za svako početno izabrano prazno polje u tabli provjerava da li se može staviti

neki nasumičan broj iz permutacije od 9 brojeva. Mape kao brojači ponavljanja brojeva optimizuju ovaj algoritam i izbjegavaju ponovno računanje tih ponavljanja tokom rekurzije. Ako je prazno polje validno, mape se ažuriraju i tabla se mijenja, a ako prazno polje ipak nije validno, mape se vraćaju na prethodno stanje i oslobađa se polje u tabli. Algoritam prestaje kada više ne postoji praznih polja u tabli.

Parametri:

- bool ok - boolean vrijednost koja pomaže u razlikovanju dvije funckije istog imena

Povratna vrijednost:

- bool - vrijednost koja se koristi tokom izvršavanja algoritma, true predstavlja validnost Sudoku table od prvog postavljenog polja, false pokreće backtracking.

4.1.3.21. Metoda za za rješavanje Sudoku table

void generateSudoku();

Metoda koja generiše Sudoku tablu. Algoritam prvo riješi praznu Sudoku tablu (sve vrijednosti su 0), a a potom petljom prolazi kroz svih 9 podmatrica i za svaku posebno nasumično izabere broj „k“ polja koja će se izbrisati (postaviti na prazno) i nasumično izabere permutaciju od prvih 9 brojeva, od kojih će u tabli brisati prvih „k“ brojeva. *Primjer*: trenutno se nalazimo u prvoj podmatrici, izabran je broj k = 3, a permutiacija je p = {2, 5, 1, 6, 4, 7, 8, 9, 3}. Prolazimo kroz sva polja prve podmatrice i brišemo samo prvih k-brojeva iz permutacije u Sudoku tabli, tj. polja koja sadrže brojeve 2, 5 i 1 će biti izbrisana. Automatski ažurira Sudoku tablu i poziva pomoćnu funckiju fillMaps().

5. Testiranje

Implementacija se testira nad klasom TestSudoku koji sadrži testove za svaku implementiraju funkcionalnost u klasi Sudoku. Testovi sadrže više primjera, neki od njih su očigledni, dok su neki namješteni kako bi se testirali svi edge case-ovi. Testiranje započinje pravljenjem instance klase TestSudoku, jer ona sama u konstruktoru poziva runTests().

5.1. Testovi klase TestSudoku

5.1.1. Test za provjeru pomoćne funkcije za popunjavanje mapa fillMaps()

Ovaj test provjerava da li pomoćna funckija fillMaps() radi kako treba. Testira se na dva primjera tabela, od kojih je jedna popunjena sa nulama i sa dvije jedinice, a druga je riješena tabela. Ručno se provjerava da li mape od odgovarajućih brojeva imaju odgovarajuću vrijednost.

5.1.2. Test za provjeru postavljanja vrijednosti u tabli setTableField()

Ovaj test provjerava da li pomoćna funckija setTableField() radi kako treba. Test postoji zbog toga što ova funkcija automatski ažurira i mape, pa se provjeravaju te vrijednosti. Testira se na jednoj tabli i ručno se provjerava da li mape od odgovarajućih brojeva imaju odgovarajuću vrijednost.

5.1.3. Test za provjeru učitavanja Sudoku zagonetke iz datoteke loadSudoku()

Testira se na dva primjera Sudoku tabli, koje se nalaze u datotekama loadTest1.txt i loadTest2.txt. Ručno se provjerava da li prethodno postavljenje tabele odgovaraju učitanim tabelana.

5.1.4. Test za provjeru čuvanja Sudoku zagonetke u datoteku saveSudoku()

Testira se na dva primjera Sudoku tabli, koje se nalaze u datotekama saveTest1.txt i saveTest2.txt. Prvo se sačuavaju Sudoku zagonetke u navedene fajlove, pa se opet učitaju u drugu promjenljivu, nakon čega se ručno provjerava da li prethodno postavljenje table odgovaraju učitanim tablama.

5.1.5. Test za provjeru provjeravanja validnosti Sudoku zagonetke checkSudoku()

Testira se na pet primjera, od kojih su prvi nekoliko jednostavne, validne Sudoku zagonetke, a preostali primjeri su napravljeni kao nevalidne Sudoku zagonetke. Provjerava se koju vrijednost vraća (true/false) i koliko grešaka broji.

5.1.6. Test za provjeru pronalaženja praznog polja u Sudoku tabli findEmptyField()

Testira se na tri primjera. Ručno se provjerava koju boolean vrijednost vraća metoda findEmptyField() i koje koordinate i na osnovu toga se tumače rezultati testa.

5.1.7. Test za provjeru provjeravanja validnosti postavljena nekog polja u Sudoku tabli isSafe()

Testira se na četiri primjera. Ručno se provjerava koju boolean vrijednost vraća funckija isSafe() i na osnovu toga se tumače rezultati testa.

5.1.8. Test za provjeru automatskog rješavanja Sudoku zagonetke solveSudoku()

Testira se na pet lakših i težih primjera. Nakon pozivanja metode koja se testira - solveSudoku(), riješena Sudoku zagonetka se provjerava sa checkSudoku() metodom, gdje se ručno gleda koju vrijednost vraća metoda checkSudoku() i na osnovu toga se tumače rezultati testa.

5.1.9. Test za provjeru generisanja postavke Sudoku zagonetke generateSudoku()

Testira se na pet primjera. Nakon pozivanja metode koja se testira - generateSudoku(), generisana Sudoku zagonetka se provjerava sa checkSudoku() metodom, gdje se ručno gleda koju vrijednost vraća metoda checkSudoku(). Nakon toga se poziva metoda solveSudoku(), pa checkSudoku() ponovo, kako bi se provjerilo da li je generisana Sudoku zagonetka rješiva. Na osnovu povratnih vrijednosti ovih provjera, tumači se rezultat testa.

5.2. Rezultati i analiza vremena izvršenja

Svi skupovi testova i testni slučajevi unutar njih su uspješno izvršeni, što ukazuje na ispravnost Sudoku programa nad definisanim skupom testova. Pod pretpostavkom da definisani skup testova obuhvata sve kritične slučajeve, naš program možemo smatrati ispravnim.

6. Zaključak

Napravljen je jedan koncept za realizaciju datog problema - “Sudoku Izazov”, koji je testiran i provjeren kroz pravljenje klase za testiranje TestSudoku i test metoda unutar nje, koje su napravljene tako da testiraju obične, ali i edge case slučajeve. Time je potvrđena funkcionalnost i ispravnost metoda svih modula.

Tokom implementiranja funkcionalnosti vezanih za Sudoku zagonetku, iskorišten je algoritam koji brzo i efikasno rješava problem, uz dodanu optimizaciju, koja dodatno poboljšava vrijeme izvršavanja. Glavni fokus implementacije je bio u realizovanju brzog i tačnog Sudoku-a, sa jasnim i čitljivim prikazom, omogućavajući modularnost funkcionalnosti, što je i zadovoljeno.