UNIVERZITET U NOVOM SADU
FAKULTET TEHNIČKIH NAUKA

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

PROJEKTNI ZADATAK

Kandidat: **Aleksa Ćup**
Broj indeksa: **SV27/2022**

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

Mentor rada: **dr Miodrag Đukić**

Novi Sad, januar, 2024.

SADRŽAJ

SPISAK SLIKA

1. Uvod

1.1 Sudoku

Sudoku je logička igra koja koristi mrežu od 9x9 kvadrata, podeljenu na manje kvadrate dimenzija 3x3. Cilj igre je popuniti svaku kolonu, red i 3x3 kvadrat brojevima od 1 do 9, pri čemu se svaki broj može pojaviti samo jednom u svakoj koloni, redu i 3x3 kvadratu. Sudoku počinje s nekoliko već unesenih brojeva, a igrač mora koristiti logiku da ispuni preostale praznine. Igra je popularna zbog svoje jednostavnosti i mogućnosti različitih nivoa težine, pružajući izazov i zabavu za sve uzraste.

Na slikama ispod su prikazane postavka Sudoku zagonetke kao i jedno od mogućih rešenja.

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

Slike 1 i 2 - Postavka i jedno od rešenja sudoku zagonetke

1.1.1 Pronalaženje svih rešenja

Osnovna strategija uključuje backtracking algoritam koji prolazi kroz svaku prazninu u matrici, pokušavajući postaviti brojeve od 1 do 9 dok ne naiđe na konflikt. Kad se to dogodi, algoritam se vraća (backtrack) na prethodnu praznu ćeliju i pokušava sledeći mogući broj. Ovaj proces se ponavlja sve dok se ne popune sve ćelije ili dok se ne utvrdi da daljnji pokušaji ne vode ka rešenju. Da bismo pronašli sve moguće rešenja, algoritam mora nastaviti s pretragom čak i nakon pronalaska prvog valjanog rešenja, vraćajući se i isprobavajući alternativne puteve koji bi mogli dovesti do drugih rešenja. Ovo može biti izuzetno vremenski zahtevno, posebno za Sudoku matrice s velikim brojem praznih polja, jer broj potencijalnih kombinacija raste eksponencijalno sa svakim dodatnim praznim poljem.

1.1.2 Pronalaženje jedinstvenih rešenja

Pronalaženje jedinstvenog rešenja u Sudoku zagonetki ne zavisi toliko od broja prethodno popunjenih polja, koliko od njihovog specifičnog rasporeda na igračkoj tabli. Dok intuitivno može delovati da veći broj inicijalnih brojeva olakšava stvaranje jedinstvenog rešenja, u praksi je raspored tih brojeva ključan za definisanje jednoznačnosti rešenja. Postoje primeri u kojima Sudoku zagonetka sa samo 17 unapred popunjenih polja može imati jedinstveno rešenje, što ilustruje ovu poentu. Ovo postizanje jedinstvenosti rešenja sa relativno malim brojem popunjenih polja nije samo impresivan matematički podvig, već i odraz dubokog razumevanja dinamike igre. Svaki broj u zagonetki ne samo da doprinosi eliminisanju mogućnosti u svojoj vrsti, koloni i 3x3 kvadratu, već i utiče na celokupnu mrežu, kreirajući lančanu reakciju ograničenja koja vode ka jednom jedinom rešenju. U dizajniranju ovakvih zagonetki, kreatori moraju pažljivo balansirati između davanja dovoljno informacija za rešavanje i održavanja izazova igre, često se oslanjajući na složene algoritme ili iskusno intuitivno planiranje kako bi osigurali da zagonetka ostane u okvirima jedinstvenog rešenja. Ovaj pristup ne samo da podiže kvalitet i izazovnost igre, već i demonstrira fascinantnu složenost koju Sudoku može ponuditi kao logička igra.

1.2 Zadatak

Cilj projekta je razviti objektno orijentisanu, konzolnu aplikaciju za igru Sudoku dimenzija 9x9. Aplikacija treba da omogući korisnicima da učitavaju Sudoku zagonetke iz tekstualnih datoteka ili da koriste generisane zagonetke koje program kreira. Osnovne funkcionalnosti uključuju učitavanje argumenata iz komandne linije, učitavanje i čuvanje zagonetki, automatsko rešavanje zagonetki, proveru ispravnosti rešenja unetih od strane korisnika, kao i generisanje novih zagonetki sa ograničenim brojem unapred popunjenih polja po podmatrici. Dodatno, aplikacija treba da pruža statističke podatke nakon svake partije, uključujući broj tačno i pogrešno postavljenih brojeva, kao i redni broj odigrane igre. Implementacija treba biti modularna, sa jasno definisanim klasama i funkcijama koje adekvatno razdvajaju različite aspekte igre. Aplikacija treba da iskoristi objektno orijentisane koncepte poput nasleđivanja, polimorfizma, kompozicije i inkapsulacije. Ovaj projekat zahteva pažljivo planiranje i implementaciju, vodeći računa o jasnoći koda, efikasnosti i korisničkom iskustvu.

2. Analiza problema

Projekat zahteva duboko razumevanje logike igre Sudoku i objektno orijentisanog programiranja. Glavni izazov je razviti aplikaciju koja je fleksibilna i korisnički prilagođena, pružajući opcije za učitavanje, čuvanje, rešavanje i generisanje Sudoku zagonetki. Svaka od ovih funkcija nosi svoje specifične izazove: od obrade tekstualnih datoteka, preko razvoja algoritama za rešavanje i generisanje zagonetki, do efikasnog rukovanja korisničkim unosima i ispravnosti podataka. Dodatni izazov predstavlja stvaranje jasnog i intuitivnog korisničkog interfejsa u konzoli, koji korisnicima olakšava navigaciju i interakciju sa programom. Pored tehničkih aspekata, potrebno je uzeti u obzir i korisničko iskustvo, osiguravajući da je igra izazovna, ali i pravedna, sa jasno definisanim pravilima i očekivanjima.

3. Koncept rešenja

Projekat se fokusira na razvijanje efikasnog algoritma za rešavanje Sudoku igre. Ključni deo ovog projekta je implementacija backtracking algoritma, koji je izuzetno pogodan za rešavanje problema poput Sudokua, gde je potrebno pronalaženje pravilnog rasporeda brojeva unutar ograničenog prostora.

Inicijalni korak u kreiranju Sudoku igre uključuje učitavanje zagonetke iz fajla ili automatsko popunjavanje cele table, a zatim brisanje određenog broja polja kako bi se formirala zagonetka. Ovaj drugi pristup osigurava da svaka generisana igra ima barem jedno validno rešenje. Generisanje kompletnog Sudokua koristi backtracking da postavi brojeve u svaku ćeliju, uzimajući u obzir pravila Sudokua – svaki broj od 1 do 9 mora biti prisutan u svakom redu, koloni i 3x3 podkvadratu bez ponavljanja. Nakon što je tabela popunjena, slučajnim odabirom brišemo određeni broj ćelija, čime stvaramo Sudoku zagonetku. Broj izbrisanih ćelija određuje težinu igre.

Za rešavanje Sudoku zagonetke, koristi se isti backtracking algoritam. Algoritam prolazi kroz prazne ćelije, postavljajući brojeve od 1 do 9 i proveravajući da li postavljeni broj krši pravila igre. Ako se dođe do situacije gde nijedan broj ne može biti postavljen, algoritam se vraća na prethodnu ćeliju (backtrack) i menja njen broj.

Projekat takođe uključuje opciju automatskog rešavanja Sudoku zagonetke, kao i mogućnost učitavanja rešenja iz datoteke. Automatsko rešavanje koristi pomenuti backtracking algoritam, dok opcija učitavanja rešenja omogućava korisnicima da testiraju vlastita rešenja. Ovo je posebno korisno za edukativne svrhe i unapređenje logičkih veština korisnika.

Kroz primenu backtracking algoritma, ovaj projekat predstavlja kako se može efikasno rešiti kompleksan problem kao što je Sudoku. Ovaj pristup nije samo koristan za stvaranje izazovne i zabavne igre, već i demonstrira važnost algoritamskog razmišljanja i programiranja u rešavanju logičkih zagonetki.

4. Opis rešenja

4.1 Program – moduli i osnovne metode

4.1.1 Modul glavnog programa (main)

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

Glavna funkcija programa. Prikazuje osnovne informacije o programu, pokreće i uvezuje sve blokove.

4.1.2 Modul za upravljanje Sudokuom (Sudoku9)

Modul za inicijalizaciju i upravljanje klasom Sudoku9.

4.1.2.1 Članovi klase

- `int** matrix`

Dinamički alocirana matrica koja predstavlja Sudoku tablu.

- `int correctInputs, incorrectInputs, currentGameCounter`

Brojači za praćenje tačnih i netačnih unosa, kao i trenutnog broja igre.

4.1.2.2 Funkcija za proveru popunjenost matrice (isFull)

```
bool isFull();
```

Funkcija provera da li su sva polja na ploči popunjena.

Povratne vrednosti:

- true – matrica je popunjena,
- false – matrica nije popunjena

4.1.2.3 Funkcija za ispis matrice (printBoard)

```
void printBoard();
```

Funkcija ispisuje matricu u konzoli.

Nema povratnih vrednosti. (void)

4.1.2.4 Funkcija za generisanje Sudoku zagonetke (generateSudoku)

```
static void generateSudoku(Sudoku9& sudoku);
```

Funkcija generiše sudoku zagonetku i postavlja je u Sudoku9 objekat koji je prosleđen kao parametar.

Parametri:

- `Sudoku9& sudoku`

Nema povratnih vrednosti. (void)

4.1.3 Modul glavnih funkcija (CoreFunctions)

Funkcije za rešavanje Sudoku zagonetke.

4.1.3.1 Funkcija za proveru validnosti poteza (isValidMove)

```
static bool isValidMove(Sudoku9& sudoku, int row, int col, int value, bool solution);
```

Funkcija proverava da li je zadati potez validan. Razlikuje da li se traži validan potez za postavku ili rešenje zagonetke.

Parametri:

- `Sudoku9& sudoku` – adresa Sudoku objekta nad kojim se proverava potez.
- `int row` – red u koji se upisuje.
- `int col` – kolona u koju se upisuje.
- `int value` – vrednost koja se upisuje.
- `bool solution` – flag koji određuje da li se validan potez traži za postavku ili rešenje.

Povratne vrednosti:

- true – potez je validan,
- false – potez nije validan

4.1.3.2 Funkcija za proveru validnosti matrice (isMatrixValid)

```
static bool isMatrixValid(Sudoku9& sudoku);
```

Proverava da li je matrica validna.

Parametri:

- `Sudoku9& sudoku` – adresa Sudoku objekta.

Povratne vrednosti:

- true – matrica je validna,
- false – matrica nije validna.

4.1.3.3 Funkcija za automatsko rešavanje Sudokua (solve)

```
static bool solve(Sudoku9& sudoku);
```

Rešava zadati Sudoku korišćenjem backtracking algoritma.

Parametri:

- `Sudoku9& sudoku` – adresa Sudoku objekta.

Povratne vrednosti:

- true – matrica je popunjena,
- false – matrica nije popunjena, triger za backtracking.

4.1.3.4 Funkcija za proveru popunjenosti matrice (isSudokuComplete)

```
static bool isSudokuComplete(Sudoku9& sudoku);
```

Proverava da li je Sudoku matrica popunjena.

Parametri:

- `Sudoku9& sudoku` – adresa Sudoku objekta.

Povratne vrednosti:

- true – matrica je popunjena,
- false – matrica nije popunjena.

4.1.3.5 Funkcija za popunjavanje validne Sudoku zagonetke (fillSudoku)

```
static bool fillSudoku(Sudoku9& sudoku);
```

Proverava da li je Sudoku matrica popunjena.

Parametri:

- `Sudoku9& sudoku` – adresa Sudoku objekta.

Povratne vrednosti:

- true – matrica je popunjena,
- false – matrica nije popunjena, triger za backtracking.

4.1.3.6 Funkcija za uklanjanje nasumičnih elemenata matrice (removeRandomElements)

```
static void removeRandomElements(Sudoku9& sudoku);
```

Funkcija uklanja nasumične elemente iz popunjene matrice.

Parametri:

- `Sudoku9& sudoku` – adresa Sudoku objekta.

Nema povratnih vrednosti. (void)

4.1.3.7 Funkcija za proveru rešenja iz fajla (checkSolutionFromFile)

```
static bool checkSolutionFromFile(Sudoku9& startingSudoku, Sudoku9& solutionSudoku, int& mistakes, int& validMoves);
```

Proverava rešenje iz fajla u slučaju da igrač rešenje učitava iz fajla.

Parametri:

- `Sudoku9& startingSudoku` – Sudoku zagonetka.
- `Sudoku9& solutionSudoku` – potencijalno rešenje zagonetke.
- `int& mistakes` – pogrešni potezi.
- `int& validMoves` – ispravni potezi.

Povratne vrednosti:

- true – polja iz postavke se poklapaju sa onima u rešenju,
- false – polja iz postavke se ne poklapaju sa onima u rešenju.

4.1.3.8 Funkcija za čišćenje konzole (clearConsole)

```
static void clearConsole();
```

Funkcija čisti konzolu.

Nema povratnih vrednosti. (void)

4.1.3.9 Funkcija za ispis glavnog menija (menu)

```
static void menu();
```

Funkcija prikazuje glavni meni.

Nema povratnih vrednosti. (void)

4.1.3.10 Funkcija za ispis menija rešenja (solutionMenu)

```
static void solutionMenu();
```

Funkcija prikazuje meni sa načinima rešavanja Sudoku zagonetke.

Nema povratnih vrednosti. (void)

4.1.4 Modul za rukovanje fajlovima (FileHandler)

Modul sa funkcijama zaduženim za rad sa fajlovima.

4.1.4.1 Metoda za čitanje iz fajla (loadSudoku)

```
static bool loadSudoku(const string& filename, Sudoku9& sudoku);
```

Učitava sudoku iz fajla.

Parametri:

- `const string& filename` – putanja fajla.
- `Sudoku9& sudoku` – Sudoku objekat.

Povratne vrednosti:

- true – uspešno upisivanje,
- false – neuspešno upisivanje.

4.1.4.2 Metoda za upis u fajl (saveSudoku)

```
static void saveSudoku(const string& filename, Sudoku9& sudoku);
```

Upisuje sudoku u fajl.

Parametri:

- `const string& filename` – putanja fajla.
- `Sudoku9& sudoku` – Sudoku objekat.

Nema povratnih vrednosti. (void)

5. Testiranje

Implementacija se testira Google testovima, a korišćen je Google Test framework za testiranje. Napravljeni su posebni testni skupovi (skupovi testnih slučajeva) za svaki modul, a svaki testni skup sadrži testove (testne slučajeve, *test cases*) raznih segmenata tog modula – funkcija i struktura, kao i testiranje njihovih međusobnih odnosa.

5.1 Testni skupovi

5.1.1 Sudoku9Test

Skup testova za verifikaciju klase *Sudoku9*.

5.1.1.1 MatrixInit

Test proverava da li je inicijalno stanje Sudoku matrice pravilno postavljeno, sa svim elementima matrice postavljenim na 0. Proverava se svako polje u 9x9 matrici da se uveri da su svi elementi nula, što ukazuje na ispravnu inicijalizaciju. Takođe se proveravaju ostala polja objekta da li su pravilno inicijalizovana.

5.1.1.2 SetAndGetMatrixElement

Testira funkcionalnosti postavljanja i dobijanja elementa matrice. Postavlja broj 5 u prvo polje matrice (0,0) i zatim proverava da li je taj element zaista postavljen na 5, čime se potvrđuje ispravnost ovih metoda.

5.1.1.3 SetAndGetCorrectInputs

Proverava da li se tačno prati broj ispravnih unosa u igri. Postavlja broj tačnih unosa na 23 i zatim proverava da li je dobijena vrednost zaista 23, što potvrđuje funkcionalnost ovih metoda.

5.1.1.4 SetAndGetIncorrectInputs

Slično kao i prethodni test, ovaj proverava funkcionalnost praćenja broja neispravnih unosa. Postavlja broj neispravnih unosa na 52 i proverava da li se ta vrednost tačno reflektuje kada se dohvati.

5.1.1.5 SetAndGetCurrentGameCounter

Testira funkcije za postavljanje i dobijanje trenutnog brojača igre. Postavlja brojač igre na 2 i proverava da li je dobijena vrednost tačno 2.

5.1.1.6 IsFull

Proverava da li funkcija isFull pravilno detektuje da li je Sudoku matrica potpuno popunjena. Prvo proverava da li je prazna matrica prepoznata kao nepotpuna, a zatim popunjava celu matricu brojevima 1 i proverava da li je matrica prepoznata kao potpuna.

5.1.1.7 GetMatrix

Testira da li se ispravno dobija referenca na matricu klase Sudoku9. Proverava da li je svaki element u dobijenoj matrici inicijalno postavljen na 0.

5.1.1.8 SetMatrix

Ovaj test proverava funkcionalnost postavljanja nove matrice u Sudoku9 objekat. Kreira novu matricu sa specifičnim vrijednostima, postavlja tu matricu u objekat i zatim proverava da li se sve vrednosti tačno prenele u objekat.

5.1.1.9 AssignmentOperator

Testira da li operator dodele (=) pravilno kopira stanje jednog Sudoku9 objekta u drugi. Postavlja nekoliko vrednosti u prvi objekat, kopira ga u drugi koristeći operator dodele i zatim proverava da li su sve vrednosti tačno kopirane u drugi objekat.

5.1.2 CoreFunctionsTest

Skup testova za verifikovanje klase *CoreFunctions*.

5.1.2.1 IsValidMove_Valid

Test proverava da li je moguće validno postaviti broj 1 u prvo polje Sudokua (0, 0), očekujući da je potez ispravan jer nema konflikta.

5.1.2.2 IsValidMove_InvalidRow

Testira potez postavljanjem broja 1 u prvi red i prvu kolonu, a zatim pokušava postaviti isti broj u istom redu, očekujući da potez bude nevažeći zbog konflikta u redu.

5.1.2.3 IsValidMove_ InvalidColumn

Test postavlja broj 1 u prvo polje, a zatim testira da li postavljanje istog broja u istu kolonu rezultira nevažećim potezom zbog konflikta u koloni.

5.1.2.4 IsValidMove_ InvalidSubgrid

Test proverava neispravnost poteza postavljanjem broja 1 u gornji levi ugao, a zatim pokušava postaviti isti broj unutar istog 3x3 kvadrata, što bi trebalo rezultovati konfliktom.

5.1.2.5 IsValidMove_ OutOfRange

Testira neispravnost poteza kada su parametri van opsega, kao što su negativni indeksi ili nula, što nije dozvoljeno u Sudokuu.

5.1.2.6 IsMatrixValid_Valid

Učitava validnu Sudoku matricu iz datoteke i proverava da li je celokupna matrica ispravna.

5.1.2.7 IsMatrixValid_Invalid

Postavlja dva ista broja u različite, ali konfliktne pozicije u Sudoku mreži i proverava da li je matrica neispravna.

5.1.2.8 IsSudokuComplete_Complete

Popunjava celi Sudoku i proverava da li je matrica potpuno popunjena i ispravna.

5.1.2.9 IsSudokuComplete_Incomplete

Popunjava Sudoku i zatim iz jednog polja uklanja broj, proveravajući da li je Sudoku nepotpun.

5.1.2.10 SolveEmptyMatrix

Testira funkciju za rešavanje Sudokua počevši od prazne matrice i očekuje da će pronaći kompletno i ispravno rešenje.

5.1.2.11 SolveRandomGeneratedMatrix

Generiše slučajnu, ali validnu Sudoku zagonetku, zatim testira funkciju Solve očekujući da pronađe ispravno rešenje.

5.1.2.12 SolveFullMatrix

Učitava već rešen i validan Sudoku i proverava da li će funkcija Solve prepoznati da je zadatak već rešen.

5.1.2.13 SolveInvalidMatrix

Postavlja neispravne vrednosti u Sudoku mrežu i proverava da li će funkcija Solve prepoznati da problem nije rešiv.

5.1.2.14 FillSudokuEmpty

Proverava funkciju za popunjavanje Sudokua od prazne matrice, očekujući da će mreža biti potpuno i ispravno popunjena.

5.1.2.15 FillSudokuWithStartFields

Učitava Sudoku sa početnim vrednostima i proverava da li funkcija FillSudoku pravilno popunjava ostatak matrice.

5.1.2.16 FillSudokuFull

Proverava da li funkcija FillSudoku prepoznaje već popunjenu matricu i pravilno javlja da je matrica potpuna.

5.1.3 FileHandlerTest

Skup testova za verifikovanje klase *FileHandler*.

5.1.3.1 SaveSudokuEmpty

Test proverava funkcionalnost čuvanja prazne Sudoku matrice u datoteku. Kreira se prazna Sudoku mreža, koja se zatim čuva u datoteku "test_out.txt". Test potom otvara ovu datoteku i čita sadržaj, proveravajući da li se vrednosti u datoteci podudaraju sa vrednostima u Sudoku mreži, što bi trebalo da budu sve nule.

5.1.3.2 SaveSudokuFull

Slično prethodnom, ali ovde se testira čuvanje potpuno popunjene Sudoku matrice. Najpre se Sudoku mreža popunjava rešenjem pomoću funkcije Solve, a zatim se čuva u datoteku. Test proverava da li su vrednosti u datoteci tačno zabeležene i da li odgovaraju popunjenim poljima u Sudoku matrici.

5.1.3.3 LoadSudoku

Test proverava sposobnost učitavanja Sudoku mreže iz datoteke. Učitava se Sudoku mreža iz datoteke "test_out.txt" i proverava se da li je učitavanje bilo uspešno. Očekuje se da se vrednosti u objektu **valid_sudoku** poklapaju sa onima koje su prethodno sačuvane u datoteci.

5.1.3.4 LoadSudoku_NonExistentFile

Testira ponašanje funkcije za učitavanje kada datoteka ne postoji. Pokreće se pokušaj učitavanja iz nepostojeće datoteke "non_existent_file.txt" i očekuje se da će funkcija vratiti neuspeh, što bi značilo da funkcija pravilno rukuje scenarijem kada datoteka ne može biti pronađena ili otvorena.

5.2 Rezultati izvršenja

Svi skupovi testova i testni slučajevi unutar njih su uspešno izvršeni, što ukazuje na ispravnost verifikovanog programa nad definisanim skupom testova. Pod pretpostavkom da definisani skup testova obuhvata sve kritične slučajeve, rešenje projekta možemo smatrati ispravnim i uzeti za referentni.

6. Zaključak

Na kraju razmatranja projekta Sudoku, možemo konstatovati da predstavljena implementacija predstavlja samo jedan od mnogih mogućih načina realizacije ove popularne logičke igre. Ispravnost svakog segmenta programa pažljivo je proverena koristeći Google test biblioteku, što osigurava pouzdanost i korektnost rada svih funkcija unutar aplikacije. Međutim, važno je napomenuti da postoji prostor za dodatnu optimizaciju, koji bi mogao doprineti boljim performansama i korisničkom iskustvu. Dalji razvoj i unapređenje algoritama za rešavanje Sudoku zagonetki će omogućiti kreiranje još efikasnijeg i bržeg softverskog rešenja.