UNIVERZITET U NOVOM SADU
FAKULTET TEHNIČKIH NAUKA

UNIVERZITET U NOVOM SADU
FAKULTET TEHNIČKIH NAUKA
NOVI SAD
Softversko inženjerstvo i informacione tehnologije

ISPITNI RAD

Kandidat: **Ana Krunić**
Broj indeksa: **SV62/2022**

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

Mentor rada: **asistent Stefan Marinkov**

Novi Sad, januar, 2024.

SADRŽAJ

1. Uvod

1.1 Sudoku zagonetka

Sudoku je logička igra brojeva koja se sastoji od 9x9 kvadratne mreže podeljene na 3x3 podmreže poznate kao blokovi. Cilj igre je popuniti svako polje mreže brojem od 1 do 9, tako da se svaki red, svaka kolona i svaki blok sastoje od jedinstvenih brojeva. Igra zahteva razmišljanje, strategiju i logičko zaključivanje kako bi se ispunile sve praznine na tabli.

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

1.2 Zadatak

Ovaj projekat implementira Sudoku igru koristeći C++. Cilj je stvoriti efikasno i korisnički prijateljsko rešenje za igranje i rešavanje Sudoku zagonetki. Različite komponente projekta obrađuju različite aspekte igre, od generisanja početnih postavki do rešavanja i provere validnosti unesenih brojeva.

2. Struktura projekta

Projekat se sastoji iz četiri klase: Sudoku, SudokuGame, SudokuSolver i Menu. Svaka od njih ima specifičnu ulogu:

2.1 'Sudoku' klasa

Sudoku klasa je odgovorna za generisanje samih Sudoku igara. Ova klasa ima metode za popunjavanje matrice, proveru ispravnosti, čitanje i pisanje u datoteke, kao i druge funkcionalnosti.

2.1.1 Sudoku(int N, int K)

Konstruktor klase. Kreira objekat sa zadatim dimenzijama (N x N) i težinom (K) za generisanje zagonetki.

2.1.2 ~Sudoku()

Destruktor klase. Oslobađa alociranu memoriju za matricu.

2.1.3 void fillValues()

Metoda za popunjavanje Sudoku matrice vrednostima.

2.1.4 void fillDiagonal()

Popunjava dijagonalne blokove Sudoku matrice.

2.1.5 bool unUsedInBox(int rowStart, int colStart, int num)

Proverava da li je broj n (num) neiskorišćen u određenom bloku.

2.1.6 void fillBox(int row, int col)

Popunjava 3x3 blok u Sudoku matrici, osiguravajući da nema ponovljenih brojeva unutar bloka.

2.1.7 int randomGenerator(int num)

Generiše slučajan broj između 1 i zadatog broja (num). Ova funkcija se koristi pri popunjavanju matrice.

2.1.8 bool CheckIfSafe(int i, int j, int num)

Proverava da li je bezbedno postaviti određeni broj (num) u zadati red (i) i kolonu (j) Sudoku matrice.

2.1.9 bool unUsedInRow(int i, int num)

Proverava da li je broj (num) neiskorišćen u zadatom redu (i) Sudoku matrice.

2.1.10 bool unUsedInCol(int j, int num)

Proverava da li je broj (num) neiskorišćen u zadatoj koloni (j) Sudoku matrice.

2.1.11 bool fillRemaining(int i, int j)

Rekurzivna funkcija koja popunjava preostali deo Sudoku mreže počevši od zadatih indeksa (i, j).

2.1.12 void removeKDigits()

Uklanja K cifara iz Sudoku matrice kako bi se generisala zagonetka. Ovo se postiže postavljanjem određenog broja elemenata na nulu.

2.1.13 void printSudoku()

Ispisuje Sudoku matricu na konzolu. Prazna polja se označavaju razmacima.

2.1.14 vector<vector<int>> toMatrix() const

Konvertuje Sudoku matricu u 2D vektor.

2.1.15 void updateFromMatrix(const vector<vector<int>>& sudokuMatrix)

Ažurira Sudoku matricu na osnovu date 2D matrice.

2.1.16 bool readFromFile(const std::string& filename)

Čita Sudoku matricu iz datoteke čije je ime prosleđeno kao argument.

2.1.17 bool writeToFile(const std::string& filename) const

Upisuje Sudoku matricu u datoteku sa zadatim imenom.

2.1.18 int isValid() const

Validira Sudoku matricu i vraća broj grešaka. Proverava popunjenost, redove i kolone na ponavljanje brojeva.

2.1.19 2.1.19. void clear()

Resetuje Sudoku matricu postavljajući sve elemente na nulu.

2.2 'SudokuGame' klasa

SudokuGame klasa predstavlja centralnu komponentu igre, odgovornu za interakciju sa korisnikom, upravljanje tokom igre, i praćenje statistike.

2.2.1 Konstruktor: SudokuGame(int N, int K)

Ovaj konstruktor inicijalizuje objekat SudokuGame sa zadatim dimenzijama (N x N) i težinom (K) za generisanje zagonetki. Prilikom inicijalizacije, postavlja se brojač gameCounter na nulu, kao i brojači valid i mistakes.

2.2.2 void setInputFileName(const string& filename)

Ova funkcija postavlja ime datoteke iz koje će se čitati konfiguracije Sudoku zagonetki. Ime datoteke se prosleđuje kao argument funkcije.

2.2.3 void setOutputFileName(const string& filename)

Ova funkcija postavlja ime datoteke u koju će se upisivati konfiguracije Sudoku zagonetki. Ime datoteke se prosleđuje kao argument funkcije.

2.2.4 void playGame()

Glavna funkcija koja upravlja igrom. Prikazuje glavni meni i na osnovu korisničkog izbora poziva odgovarajuće funkcionalnosti. Ako korisnik odabere opciju za učitavanje postojeće zagonetke, poziva se displayGameMenu() funkcija. Ako odabere opciju za generisanje nove zagonetke, generiše se zagonetka, upisuje u datoteku i poziva se displayGameMenu() funkcija.

2.2.5 void displayGameMenu()

Ova funkcija prikazuje meni za igru, dajući korisniku opcije za samostalno rešavanje zagonetke ili rešavanje zagonetke od strane računara. Na osnovu korisničkog izbora, odgovarajuća funkcionalnost se izvršava. Ako korisnik bira samostalno rešavanje, zagonetka se prikazuje na ekranu, a nakon rešavanja, ispisuje se broj grešaka i igrač ima mogućnost da igra ponovo. Ako korisnik bira da računar reši zagonetku, rešenje se generiše i prikazuje, nakon čega korisnik takođe ima opciju da igra ponovo.

2.2.6 void playAgain()

Ova funkcija omogućava igraču da izabere da li želi da igra ponovo ili da se vrati na glavni meni. Ako igrač bira da igra ponovo, brojač gameCounter se inkrementira, i igra se ponovo pozivanjem funkcije playGame(). Ako igrač bira da se vrati na glavni meni, ispisuje se zahvalnica na kraju igre.

2.3 'SudokuSolver' Klasa

SudokuSolver klasa implementira algoritam rešavanja Sudoku zagonetki koristeći metodu povratnog praćenja (backtracking). Ključna funkcionalnost ove klase je pronalaženje rešenja za zadatu zagonetku.

2.3.1 bool findEmptyLocation(const vector<vector<int>>& sudoku, int& row, int& col) const

Ova funkcija traži prvu praznu lokaciju (vrednost 0) u Sudoku mreži koja je predstavljena 2D vektorom sudoku. Prazna lokacija se postavlja u row i col koje se prosleđuju kao izlazni parametri. Ako prazna lokacija postoji, funkcija vraća true, inače false.

2.3.2 bool isValid(const vector<vector<int>>& sudoku, int row, int col, int num) const

Ova funkcija proverava da li je bezbedno postaviti broj num na određenu lokaciju (row, col) u Sudoku mreži. Proverava se da li se broj num već nalazi u istom redu, koloni ili 3x3 bloku. Ako je postavljanje broja bezbedno, funkcija vraća true, inače false.

2.3.3 bool solveSudoku(vector<vector<int>>& sudoku)

Ova funkcija implementira algoritam rešavanja Sudoku zagonetke pomoću povratnog praćenja. Koristi se rekurzivno pozivanje funkcije kako bi se probale sve mogućnosti. Ako se pronađe rešenje, vraća true, inače false. Samo prvo pronađeno rešenje biće zadržano.

2.3.4 bool solveSudoku(Sudoku& sudoku)

Ova funkcija omogućava rešavanje Sudoku zagonetke predstavljene objektom klase Sudoku. Prvo se konvertuje objekat Sudoku u 2D vektor pomoću toMatrix() metode. Nakon toga, poziva se funkcija solveSudoku(vector<vector<int>>& sudoku) da bi se rešila zagonetka. Ako je zagonetka rešiva, ažuriraju se podaci u originalnom objektu Sudoku koristeći updateFromMatrix() metodu.

2.4 'Menu' Klasa

Menu klasa predstavlja implementaciju korisničkog menija za interakciju sa korisnikom tokom izvršavanja Sudoku igre. Ključne funkcije ove klase su vezane za prikaz različitih menija i prikupljanje korisničkog izbora.

2.4.1 int showMainMenu()

Ova funkcija prikazuje glavni meni programa koji korisnicima omogućava odabir između tri opcije:

1. Učitaj postojecu zagonetku iz datoteke: Omogućava korisniku da učita postojeću Sudoku zagonetku iz datoteke.
2. Generisi novu zagonetku: Generiše novu Sudoku zagonetku koja će se koristiti u igri.
0. Izlaz iz programa: Završava izvršavanje programa.

Funkcija čeka unos korisnika i vraća odabrani broj.

2.4.2 int showGameMenu()

Ova funkcija prikazuje meni za igru, omogućavajući korisniku da bira između tri opcije:

1. Samostalno resi zagonetku: Omogućava korisniku da samostalno rešava generisanu Sudoku zagonetku.
2. Kompjuter resava zagonetku: Omogućava računaru da reši zagonetku i prikaže rešenje.
0. Nazad na glavni meni: Korisnik se vraća na glavni meni.

Funkcija čeka unos korisnika i vraća odabrani broj.

2.4.3 int playAgainMenu()

Ova funkcija prikazuje meni koji se pojavljuje nakon završetka igre, dajući korisniku opciju da igra ponovo ili se vrati na glavni meni. Ponovljeno igranje omogućava korisnicima da nastave sa sledećom igrom, povećavajući ukupan broj odigranih igara.

Funkcija čeka unos korisnika i vraća odabrani broj.

3. Main Funkcija

Main funkcija je centralni deo programa koja inicijalizuje igru Sudoku, obrađuje komandne linije, postavlja nasumično seme generatora brojeva, i pokreće glavnu petlju igre. Evo detaljnog pregleda ključnih elemenata main funkcije:

3.1 Inicijalizacija Generatora Nasumičnih Brojeva

```
srand(static_cast<unsigned>(time(0)));
```

Postavljanje nasumičnog semena pomoću trenutnog vremena omogućava generisanje različitih nizova nasumičnih brojeva pri svakom pokretanju programa.

3.2 Provera Komandnih Linija

```
if (argc < 3) {
    std::cerr << "Nije uneto dovoljno argumenata" << std::endl;
    return 1;
}
```

Proverava se da li su uneti dovoljni argumenti iz komandne linije. Ako ne, ispisuje se poruka o grešci i program se završava sa povratnom vrednošću različitom od nule.

3.3 Kreiranje Instanci Klase SudokuGame

```
SudokuGame game(9, 35);
```

Kreira se objekat game klase SudokuGame sa zadatim parametrima (veličina Sudoku mreže: 9x9, broj uklonjenih cifara pri generisanju zagonetke: 35).

3.4 Postavljanje Imena Ulaznih i Izlaznih Datoteka

game.setInputFileName(argv[2]);

game.setOutputFileName(argv[1]);

Postavljaju se imena ulazne i izlazne datoteke na osnovu argumenata komandne linije.

3.5 Pokretanje Glavne Petlje Igre

game.playGame();

Pokreće se glavna petlja igre pozivom playGame metode objekta game. Ova metoda upravlja tokom igre, uključujući odabir opcija, generisanje zagonetki, rešavanje, i interakciju sa korisnikom.

3.6 Povratna Vrednost Main Funkcije

return 0;

main funkcija se završava sa povratnom vrednošću 0, što označava uspešno izvršenje programa.

Ova struktura main funkcije obezbeđuje osnovnu infrastrukturu za pokretanje Sudoku igre, a dalje ponašanje i interakcija su detaljno implementirane u okviru klase SudokuGame i povezanih klasa.

4. Testiranje

4.1 Testovi nad klasom Sudoku

U navedenom testnom skupu, izvršena su četiri testa nad funkcionalnostima klase Sudoku koja se koristi za generisanje, čitanje i proveru ispravnosti Sudoku tablica.

4.1.1 TestFillValues

Ovaj test proverava da li funkcija fillValues ispravno popunjava sva polja Sudoku tablice. Nakon popunjavanja, vrši se provera ispravnosti tablice korišćenjem funkcije isValid koja bi trebala vratiti 0 (nula) jer su sva polja popunjena.

4.1.2 TestReadFromFile

Ovaj test proverava funkcionalnost čitanja Sudoku tablice iz datoteke. Prvo se kreira objekat klase Sudoku i zatim pokušava učitavanje tablice iz datoteke "proba_sudoku.txt". Proverava se uspešnost čitanja pomoću Assert::IsTrue(success).

4.1.3 TestWriteToFile

Ovaj test proverava funkcionalnost čuvanja Sudoku tablice u datoteku. Prvo se kreira objekat klase Sudoku, zatim se tablica popunjava vrednostima, a potom se pokušava čuvanje u datoteku "proba_sudoku.txt". Proverava se uspešnost čuvanja pomoću Assert::IsTrue(success).

4.1.4 TestIsValid

Ovaj test proverava funkciju isValid koja treba da vrati nulu ako je Sudoku tablica ispravno popunjena. Prvo se kreira objekat klase Sudoku, tablica popunjava vrednostima, a zatim se proverava ispravnost. Nakon toga, nekoliko vrednosti se ručno postavlja kako bi se simulirala greška, a ponovno se proverava ispravnost.

4.1.5 TestClear

Ovaj test proverava funkciju clear koja treba da postavi sva polja Sudoku tablice na nulu. Nakon popunjavanja tablice, poziva se clear i proverava se da li su sva polja vraćena na početne vrednosti.

Ovi testovi pružaju sveobuhvatnu proveru ključnih funkcionalnosti klase Sudoku i obezbeđuju pouzdanost i ispravnost njenog ponašanja. Ukoliko testovi prolaze, možemo biti sigurni da osnovne funkcionalnosti implementacije rade ispravno.

4.2 Testovi nad klasom SudokuSolver

U ovom testnom skupu, vršena su tri ključna testiranja funkcionalnosti klase SudokuSolver, koja je odgovorna za rešavanje Sudoku zagonetki.

4.2.1 TestFindEmptyLocation

Ovaj test proverava funkciju findEmptyLocation, koja treba da pronađe prvo prazno polje u zadatoj Sudoku tablici. Kreiramo objekat klase SudokuSolver i dve matrice – jedna u u kojoj se može dodati jedinica (nema greške) i jednu kojoj se broj 1 pojavljuje dva puta u koloni (doći će do greške)

Prva matrica:

1 2 3 4 5 6 7 8 9

4 5 6 7 8 9 1 2 3

7 8 9 1 2 3 4 5 6

2 3 4 5 6 7 8 9 1

5 6 7 8 9 1 2 3 4

8 9 1 2 3 4 5 6 7

3 4 5 6 7 8 9 1 2

6 7 8 9 1 2 3 4 5

9 1 2 3 4 5 6 7 8

Druga matrica:

1 2 3 4 5 6 7 8 9

4 5 6 7 8 9 1 2 3

7 8 9 1 2 3 4 5 6

2 3 4 5 6 7 8 9 1

5 6 7 8 9 1 2 3 4

1 9 1 2 3 4 5 6 7

3 4 5 6 7 8 9 1 2

6 7 8 9 1 2 3 4 5

9 1 2 3 4 5 6 7 8

Pozivamo funkciju findEmptyLocation za obe matrice i proveravamo rezultate. Očekujemo da prva matrica ne sadrži prazno polje (Assert::AreEqual(false, result)) dok druga matrica sadrži prazno polje (Assert::AreEqual(true, result)) i dodatno proveravamo vrednost praznog polja.

4.2.2 TestIsValid

Ovaj test proverava funkciju isValid, koja treba da utvrdi da li je moguće postaviti određeni broj na zadato prazno polje u Sudoku tablici. Kreiramo objekat klase SudokuSolver i matricu koja predstavlja Sudoku tablicu. Prvo proveravamo validnost postavljanja broja 1 na prvo prazno polje u matrici, a zatim menjamo vrednost tog polja kako bi postavili nevalidan broj (1) i ponovno proveravamo. Očekujemo da prva provera bude uspešna (Assert::AreEqual(true, result)) dok druga neuspešna (Assert::AreEqual(false, result)).

4.2.3 TestSolveSudoku

Ovaj test proverava funkciju solveSudoku, koja treba da reši zadatu Sudoku tablicu. Kreiramo objekat klase SudokuSolver i objekat klase Sudoku koji se popunjava vrednostima. Zatim konvertujemo Sudoku objekat u matricu, pozivamo solveSudoku i ažuriramo originalni Sudoku objekat sa rešenjem. Na kraju proveravamo da li je rešenje ispravno, odnosno da li je rezultat funkcije isValid jednak 0 (Assert::AreEqual(0, sudoku.isValid())).

Ova testiranja pružaju sveobuhvatnu proveru ključnih funkcionalnosti klase SudokuSolver i obezbeđuju pouzdanost i ispravnost njenog ponašanja. Ukoliko testovi prolaze, možemo biti sigurni da osnovne funkcionalnosti implementacije rada ispravno

5. Zaključak

Projekat Sudoku implementira kompletno iskustvo igranja Sudoku igre, pružajući korisnicima mogućnost generisanja novih zagonetki, samostalnog rešavanja ili rešavanja od strane računara. Kroz korišćenje C++ programskog jezika i objektno-orijentisanog dizajna, projekat ostvaruje čistu i modularnu strukturu.

5.1 Postignuti Ciljevi

Generisanje Sudoku Zagonetki: Implementirana je funkcionalnost za generisanje novih Sudoku zagonetki sa zadatim parametrima (veličina tablice i broj uklonjenih cifara). Ovo pruža beskrajne mogućnosti za igranje.

Rešavanje Sudoku Zagonetki: Backtracking algoritam koristi se za rešavanje zagonetki, kako bi korisnici imali opciju da ih reše samostalno ili da puste računar da pronađe rešenje.

Interaktivna Igra: Kroz SudokuGame klasu, implementirana je interaktivnost, omogućavajući korisnicima da biraju opcije, prate rezultate, i nastave sa igranjem ili se vrate na glavni meni.

Testiranje i Sigurnost Koda: Razvijeni su testovi koristeći Microsoftov C++ Unit Test Framework kako bi se garantovala funkcionalnost i ispravnost implementacije. Ovo doprinosi sigurnosti koda i lakšem održavanju.

5.2 Dalji Razvoj

Mogući koraci za dalji razvoj projekta uključuju dodavanje naprednih algoritama za generisanje zagonetki, poboljšanje korisničkog interfejsa, i eventualno dodavanje više nivoa težine za Sudoku igru. Takođe, mogu se implementirati dodatne statistike o igrama, što bi dodatno obogatilo korisničko iskustvo.

5.3 Zaključak

Projekat Sudoku uspešno implementira klasičnu igru Sudoku, nudeći korisnicima zabavno iskustvo rešavanja zagonetki. Modularna arhitektura koda, testiranje i jasan interfejs doprinose održivosti i proširivosti projekta. Sudoku ostaje popularna igra koja podstiče logičko razmišljanje, a implementacija kroz ovaj projekat omogućava pristupačnost igre svim korisnicima.