# Projekat Sudoku 9x9 - izveštaj

*Vuk Vićentić, SV45-2022*

# Rad U/I podsistema

1. **Početak igre**
    a. Učitaj sudoku zagonetku iz datoteke
    b. Generiši novu sudoku zagonetku
2. **Učitavanje zagonetke iz datoteke**: Ako korisnik izabere prvu opciju, program traži od korisnika da unese naziv datoteke. Zagonetka se zatim učitava iz te datoteke i prikazuje se korisniku.
3. **Generisanje nove zagonetke**: Ako korisnik izabere drugu opciju, program generiše novu zagonetku, upisuje je u datoteku ‘sudoku.txt’, a zatim je prikazuje korisniku.
4. **Rešavanje zagonetke**: Nakon što je zagonetka učitana ili generisana, korisniku se prikazuje novi meni sa četiri opcije:
    a. Unesi rešenje
    b. Reši zagonetku
    c. Ispiši statistiku
    d. Izađi iz programa
5. **Unos rešenja**: Ako korisnik izabere prvu opciju, program traži od korisnika da unese red, kolonu i broj. Ako je unos validan, broj se unosi u zagonetku i prikazuje se korisniku.
6. **Automatsko rešavanje zagonetke**: Ako korisnik izabere drugu opciju, program pokušava da reši zagonetku. Ako je zagonetka uspešno rešena, rešenje se prikazuje korisniku.
7. **Prikaz statistike**: Ako korisnik izabere treću opciju, program prikazuje statistiku igre.
8. **Kraj igre**: Ako korisnik izabere četvrtu opciju, program se završava.
9. **Nova igra**: Ako je zagonetka rešena, korisniku se prikazuje meni sa dve opcije:
    a. Započni novu igru
    b. Izađi iz programa

# Klasa Sudoku9

Program ima samo jednu klasu po imenu Sudoku9.

## Atributi klase

- **std::vector<std::vector<int>> grid;**
  - Ovaj atribut predstavlja dvodimenzionalni vektor koji čuva trenutno stanje Sudoku zagonetke. Svaki element vektora predstavlja jedno polje na Sudoku tabli. Vrednost 0 obično označava prazno polje, dok brojevi od 1 do 9 predstavljaju unesene brojeve.
- **int correctEntries;**
  - Ovaj atribut predstavlja broj tačno unesenih brojeva u Sudoku zagonetku. Svaki put kada korisnik unese broj koji je validan za dato polje (tj. ne krši pravila Sudokua), ovaj brojač se povećava za jedan.
- **int falseEntries;**
  - Ovaj atribut predstavlja broj netačno unesenih brojeva u Sudoku zagonetku. Svaki put kada korisnik pokuša da unese broj koji nije validan za dato polje (tj. krši pravila Sudokua), ovaj brojač se povećava za jedan.
- **int gamesPlayed;**
  - Ovaj atribut predstavlja broj odigranih igara. Svaki put kada se završi igra (bilo da je zagonetka rešena ili korisnik izabere da započne novu igru), ovaj brojač se povećava za jedan.

## Funkcije klase

- **void print();**
  - **Ispisivanje brojeva kolona**: Prva dva for ciklusa ispisuju brojeve kolona od 1 do 9. Ovi brojevi služe kao oznake kolona prilikom igranja igre.
  - **Ispisivanje linije razdvajanja**: Sledeći for ciklus ispisuje liniju razdvajanja. Ova linija služi za vizualno razdvajanje oznaka kolona od samih polja Sudoku zagonetke.
  - **Ispisivanje polja zagonetke:** Poslednji for ciklus prolazi kroz svako polje u Sudoku zagonetki. Za svaki red, prvo se ispisuje broj reda (od 1 do 9), a zatim se ispisuju vrednosti svih polja u tom redu. Vrednost 0 označava prazno polje, dok brojevi od 1 do 9 predstavljaju unesene brojeve.
- **bool isSafe(int row, int col, int num);**

    - **Provera argumenata**: Prvo se proverava da li su argumenti validni. Ako su row, col ili num van dozvoljenih granica (0-8 za row i col, 1-9 za num), funkcija odmah vraća false.
    - **Provera u trenutnom redu**: Zatim se proverava da li se broj num već nalazi u trenutnom redu. Ako se nalazi, funkcija vraća false.
    - **Provera u trenutnoj koloni**: Nakon toga se proverava da li se broj num već nalazi u trenutnoj koloni. Ako se nalazi, funkcija vraća false.
    - **Provera u trenutnom 3x3 kvadratu**: Na kraju se proverava da li se broj num već nalazi u trenutnom 3x3 kvadratu. Ako se nalazi, funkcija vraća false.
    - **Povratna vrednost**: Ako broj num ne krši nijedno pravilo Sudokua, funkcija vraća true, što znači da je bezbedno postaviti broj num na poziciju (row, col).
- **bool solve();**
    - **Pretraga praznih polja**: Funkcija prolazi kroz svako polje na Sudoku tabli (koristeći dva ugnježđena for ciklusa). Ako nađe prazno polje (vrednost 0), prelazi na sledeći korak.
    - **Pokušaj postavljanja brojeva**: Za prazno polje, funkcija pokušava da postavi brojeve od 1 do 9 (koristeći for ciklus).
    - **Provera bezbednosti**: Za svaki broj, funkcija proverava da li je bezbedno postaviti taj broj na trenutno polje pozivom funkcije isSafe(row, col, num). Ako je bezbedno, broj se postavlja na to polje.
    - **Rekurzivni poziv**: Nakon postavljanja broja, funkcija rekurzivno poziva sebe da reši preostalu zagonetku. Ako rekurzivni poziv vrati true, to znači da je zagonetka uspešno rešena, pa funkcija vraća true.
    - **Povratak na prethodno stanje**: Ako rekurzivni poziv vrati false, to znači da trenutno postavljeni broj ne vodi do rešenja, pa se polje vraća na 0 (prazno polje), a funkcija nastavlja sa sledećim brojem.
    - **Nemogućnost postavljanja broja**: Ako nijedan broj ne može biti postavljen na trenutno prazno polje, funkcija vraća false. Ovo uzrokuje povratak na prethodno stanje u prethodnim rekurzivnim pozivima.
    - **Provera validnosti rešenja**: Nakon što su sva polja popunjena, funkcija proverava da li je trenutno stanje table validno pozivom funkcije isSolutionValid(). Ako nije validno, funkcija vraća false.
    - **Uspešno rešenje**: Ako su sva polja popunjena i trenutno stanje table je validno, funkcija vraća true, što znači da je zagonetka uspešno rešena.
- **void loadFromFile(std::string filename);**
    - **Otvaranje datoteke**: Funkcija pokušava da otvori datoteku sa imenom filename. Ako datoteka ne može biti otvorena, funkcija ispisuje poruku o grešci i završava se.

- **Čitanje linija**: Ako je datoteka uspešno otvorena, funkcija prolazi kroz svaku liniju datoteke (svaki red Sudoku table). Za svaku liniju, funkcija proverava da li linija sadrži tačno 9 karaktera. Ako ne, funkcija ispisuje poruku o grešci i završava se.
- **Čitanje karaktera**: Za svaki karakter u liniji, funkcija proverava da li je karakter cifra. Ako nije, funkcija ispisuje poruku o grešci i završava se.
- **Konverzija karaktera u broj**: Ako je karakter cifra, funkcija ga konvertuje u broj oduzimanjem vrednosti karaktera '0' od vrednosti karaktera.
- **Postavljanje broja**: Ako je broj različit od nule, funkcija proverava da li je bezbedno postaviti taj broj na trenutnu poziciju u tabeli pozivom funkcije isSafe(i, j, num). Ako nije, funkcija ispisuje poruku o grešci i završava se. Ako jeste, broj se postavlja na tu poziciju pozivom funkcije set(i, j, num).
- **Zatvaranje datoteke**: Nakon što su sve linije pročitane, funkcija zatvara datoteku.
- **Ažuriranje stanja table**: Na kraju, ako je sve prošlo bez grešaka, trenutno stanje Sudoku table se ažurira sa stanjem iz učitane datoteke.

- **void printToFile(std::string filename);**
  - Otvaranje datoteke: Funkcija pokušava da otvori datoteku sa imenom filename za pisanje. Ako datoteka ne može biti otvorena, funkcija ispisuje poruku o grešci i završava se.
  - Ispisivanje polja zagonetke: Ako je datoteka uspešno otvorena, funkcija prolazi kroz svako polje na Sudoku tabli (koristeći dva ugnježđena for ciklusa) i ispisuje vrednost svakog polja u datoteku. Nakon svakog reda, funkcija ispisuje novi red.
  - Zatvaranje datoteke: Nakon što su sva polja ispisana, funkcija zatvara datoteku.
- **bool isSolutionValid();**
  - **Provera svakog reda**: Prva petlja prolazi kroz svaki red na Sudoku tabli. Za svaki red, funkcija proverava da li svaki broj od 1 do 9 postoji tačno jednom. Ako neki broj postoji više od jednom, funkcija vraća false.
  - **Provera svake kolone**: Druga petlja prolazi kroz svaku kolonu na Sudoku tabli. Slično kao i za redove, funkcija proverava da li svaki broj od 1 do 9 postoji tačno jednom u svakoj koloni. Ako neki broj postoji više od jednom, funkcija vraća false.
  - **Provera svakog 3x3 kvadrata**: Treća petlja prolazi kroz svaki 3x3 kvadrat na Sudoku tabli. Ponovo, funkcija proverava da li svaki broj od 1 do 9 postoji tačno jednom u svakom kvadratu. Ako neki broj postoji više od jednom, funkcija vraća false.

    - **Povratna vrednost**: Ako su sve provere prošle (tj. svaki broj od 1 do 9 postoji tačno jednom u svakom redu, svakoj koloni i svakom 3x3 kvadratu), funkcija vraća true, što znači da je trenutno stanje Sudoku table validno rešenje.
- **void generatePuzzle();**
    - **Inicijalizacija generatora slučajnih brojeva**: Funkcija prvo inicijalizuje generator slučajnih brojeva koristeći trenutno vreme kao seme.
    - **Generisanje potpuno popunjene table**: Zatim, funkcija generiše potpuno popunjenu Sudoku tablu pozivom funkcije solve(). Ako solve() vrati false, funkcija se završava.
    - **Uklanjanje brojeva sa table**: Nakon što je tabela potpuno popunjena, funkcija prolazi kroz svako polje na Sudoku tabli i sa 50% šanse uklanja broj sa tog polja.
    - **Provera broja popunjenih polja po podmatrici**: Funkcija zatim prolazi kroz svaki 3x3 kvadrat na Sudoku tabli i broji koliko polja u tom kvadratu je popunjeno. Ako je broj popunjenih polja veći od 6, funkcija uklanja neka polja dok broj popunjenih polja ne postane 6.
- **void printStats() const;**
    - **Ispisivanje broja tačno unesenih brojeva**: Funkcija prvo ispisuje broj tačno unesenih brojeva u Sudoku zagonetku. Ovaj broj se čuva u atributu correctEntries.
    - **Ispisivanje broja netačno unesenih brojeva**: Zatim, funkcija ispisuje broj netačno unesenih brojeva u Sudoku zagonetku. Ovaj broj se čuva u atributu falseEntries.
    - **Ispisivanje rednog broja partije koja se igra**: Na kraju, funkcija ispisuje redni broj partije koja se trenutno igra. Ovaj broj se čuva u atributu gamesPlayed i povećava se za jedan pre ispisa, jer brojanje partija počinje od 1, a ne od 0.

# Strukture ulazne i izlazne datoteke

## Struktura ulazne datoteke

Ulazna datoteka predstavlja početno stanje Sudoku zagonetke. Ova datoteka se sastoji od 9 redova teksta, pri čemu svaki red predstavlja jedan red na Sudoku tabli. Svaki red sadrži 9 cifara, koje predstavljaju vrednosti polja u tom redu, od leve ka desno. Broj 0 označava prazno polje.

Na primer:

023406089

050700123

089123006

200005097

365890014

007214305

531642908

640970501

070500640

Ovaj primer predstavlja početno stanje Sudoku zagonetke gde su neka polja već popunjena, dok su ostala polja prazna (0).

## Struktura izlazne datoteke

Izlazna datoteka je identična strukturi ulazne datoteke. Ona predstavlja trenutno stanje Sudoku zagonetke nakon što je program generisao novu zagonetku. Svaki red u datoteci predstavlja jedan red na Sudoku tabli, a svaka cifra u redu predstavlja vrednost odgovarajućeg polja na tabli. Broj 0 označava prazno polje.

Nakon što je program generisao novu zagonetku, stanje zagonetke se upisuje u izlaznu datoteku koristeći isti format kao za ulaznu datoteku. Ovo omogućava da se izlazna datoteka može koristiti kao ulazna datoteka za sledeću igru.

# Backtracking algoritam

Program koristi *backtracking* algoritam za rešavanje zagonetke. Ovaj algoritam se sastoji iz sledećih koraka:

- **Pretraga praznih polja**
  - Algoritam prolazi kroz svako polje na Sudoku tabli. Ako nađe prazno polje, prelazi na sledeći korak.
- **Pokušaj postavljanja brojeva**
  - Za prazno polje, algoritam pokušava da postavi brojeve od 1 do 9.
- **Provera bezbednosti**
  - Za svaki broj, algoritam proverava da li je bezbedno postaviti taj broj na trenutno polje pozivom funkcije isSafe(). Ako je bezbedno, broj se postavlja na to polje.
- **Rekurzivni poziv**
  - Nakon postavljanja broja, algoritam rekurzivno poziva sebe da reši preostalu zagonetku. Ako rekurzivni poziv vrati true, to znači da je zagonetka uspešno rešena, pa algoritam vraća true.
- **Povratak na prethodno stanje**
  - Ako rekurzivni poziv vrati false, to znači da trenutno postavljeni broj ne vodi do rešenja, pa se polje vraća na 0 (prazno polje), a algoritam nastavlja sa sledećim brojem.
- **Nemogućnost postavljanja brojeva**
  - Ako nijedan broj ne može biti postavljen na trenutno prazno polje, algoritam vraća false. Ovo uzrokuje povratak na prethodno stanje u prethodnim rekurzivnim pozivima.
- **Uspešno traženje**
  - Ako su sva polja popunjena i trenutno stanje table je validno, algoritam vraća true, što znači da je zagonetka uspešno rešena.

# Testiranje

Testiranje funkcionalnosti je izvršeno u posebnom projektu pomoću modula *assert,* proverom svih graničnih slučajeva.

- **testIsSafe();**
  - **Testiranje sa praznom tablom**: Prvo se proverava da li metoda isSafe() vraća true za svako polje na praznoj Sudoku tabli kada se postavlja broj 1. Ovo je očekivano ponašanje jer je na praznoj tabli bezbedno postaviti bilo koji broj na bilo koje polje.
  - **Testiranje sa popunjenom tablom**: Zatim se popunjava cela tabla brojem 1 i proverava se da li metoda isSafe() vraća false za svako polje na tabli kada se postavlja broj 1. Ovo je očekivano ponašanje jer nije bezbedno postaviti broj 1 na polje koje već sadrži broj 1.
  - **Testiranje sa nevažećim indeksima**: Nakon toga se proverava da li metoda isSafe() vraća false kada se koriste nevažeći indeksi (tj. indeksi koji su van granica 0-8).
  - **Testiranje sa nevažećim brojevima**: Zatim se proverava da li metoda isSafe() vraća false kada se koriste nevažeći brojevi (tj. brojevi koji su van granica 1-9).
  - **Testiranje sa različitim brojevima na istom mestu**: Na kraju, za svaki broj od 1 do 9, broj se postavlja na prvo polje na tabli, a zatim se proverava da li metoda isSafe() vraća false za to polje i taj broj. Ovo je očekivano ponašanje jer nije bezbedno postaviti broj na polje koje već sadrži taj broj.
- **testSolve();**
  - **Testiranje sa praznom tablom**: Prvo se proverava da li metoda solve() vraća true za praznu Sudoku tablu. Ovo je očekivano ponašanje jer je prazna tabla početna tačka za rešavanje Sudoku zagonetke.
  - **Testiranje sa tablom koja može da se reši**: Zatim se učitava stanje Sudoku table iz datoteke “solve_valid.txt” i proverava se da li metoda solve() vraća true. Ovo je očekivano ponašanje jer bi tabela koja može da se reši trebalo da bude rešena metodom solve().
  - **Testiranje sa tablom koja ne može biti rešena**: Nakon toga se postavljaju sve vrednosti na Sudoku tabli na 1 i proverava se da li metoda solve() vraća false. Ovo je očekivano ponašanje jer tabela koja ne može biti rešena ne bi trebalo da bude rešena metodom solve().
- **testLoadFromFile();**

    - **Testiranje sa validnom datotekom**: Prvo se proverava da li metoda loadFromFile() pravilno učitava Sudoku tablu iz validne datoteke. Očekuje se da prva linija učitane table izgleda ovako: 003000000. Ako to nije slučaj, test ne prolazi.
    - **Testiranje sa nevažećom datotekom**: Zatim se proverava kako metoda loadFromFile() reaguje na nevažeću datoteku. Očekuje se da će metoda ispisati poruku o grešci i da neće promeniti trenutno stanje table. Ako to nije slučaj, test ne prolazi.
    - **Testiranje sa nepostojećom datotekom**: Na kraju, proverava se kako metoda loadFromFile() reaguje na nepostojeću datoteku. Očekuje se da će metoda ispisati poruku o grešci i da neće promeniti trenutno stanje table. Ako to nije slučaj, test ne prolazi.
- **testPrintToFile();**
    - **Testiranje sa validnom datotekom**: Prvo se proverava da li metoda printToFile() pravilno upisuje trenutno stanje Sudoku table u datoteku. Nakon poziva metode printToFile(), test funkcija proverava da li je datoteka uspešno kreirana proverom da li se datoteka može otvoriti za čitanje. Ako to nije slučaj, test ne prolazi.
    - **Testiranje sa nevažećom putanjom do datoteke**: Zatim se proverava kako metoda printToFile() reaguje na nevažeću putanju do datoteke. Očekuje se da će metoda ispisati poruku o grešci ako ne može da kreira datoteku na navedenoj putanji.
- **testIsSolutionValid();**
    - **Testiranje sa validnom rešenom tablom**: Prvo se učitava stanje Sudoku table iz datoteke “valid_resenje.txt” koja predstavlja validno rešenje Sudoku zagonetke. Zatim se proverava da li metoda isSolutionValid() vraća true. Ovo je očekivano ponašanje jer bi metoda isSolutionValid() trebalo da vrati true za validno rešenje Sudoku zagonetke.
    - **Testiranje sa nevalidnom rešenom tablom**: Zatim se učitava stanje Sudoku table iz datoteke “invalid_resenje.txt” koja predstavlja nevalidno rešenje Sudoku zagonetke. Zatim se proverava da li metoda isSolutionValid() vraća false. Ovo je očekivano ponašanje jer bi metoda isSolutionValid() trebalo da vrati false za nevalidno rešenje Sudoku zagonetke.
- **testGeneratePuzzle();**
    - **Generisanje zagonetke**: Prvo se poziva metoda generatePuzzle() da generiše novu Sudoku zagonetku.
    - **Provera broja popunjenih polja po podmatrici**: Zatim se proverava da li je broj popunjenih polja u svakoj 3x3 podmatrici manji ili jednak 6. Ovo je

očekivano ponašanje jer metoda generatePuzzle() treba da generiše zagonetku sa najviše 6 popunjenih polja u svakoj 3x3 podmatrici.

- **Provera validnosti svakog polja**: Na kraju, za svako polje na Sudoku tabli koje nije prazno, proverava se da li je broj na tom polju bezbedan za to polje. Ovo se postiže tako što se prvo postavlja polje na 0, zatim se proverava da li je originalni broj bezbedan za to polje pozivom metode isSafe(i, j, num), a zatim se polje vraća na originalnu vrednost. Ako metoda isSafe(i, j, num) vrati false za bilo koje polje, test ne prolazi.

# Problemi i ograničenja

- **Performanse algoritma**: Algoritam “Backtracking” koji koristite za rešavanje Sudoku zagonetke može biti neefikasan za određene vrste zagonetki, posebno za one koje imaju mnogo praznih polja. Ovo može dovesti do dugih vremena izvršavanja.
- **Generisanje zagonetki**: Metoda generatePuzzle() generiše zagonetke tako što prvo generiše potpuno popunjenu tablu, a zatim uklanja brojeve sa table. Međutim, ovo ne garantuje da će generisana zagonetka imati jedinstveno rešenje, što je poželjno za Sudoku zagonetke.
- **Obrada grešaka pri učitavanju datoteka**: Metode loadFromFile() i printToFile() trenutno ispisuju poruke o grešci na konzolu kada naiđu na problem sa datotekom. Međutim, ovo može biti neadekvatno za korisnički interfejs koji ne koristi konzolu. Za tako nešto bi bilo bolje da ove metode vraćaju kodove grešaka ili bacaju izuzetke koje bi pozivajući kod mogao da obradi.
- **Ograničenja u dizajnu**: Klasa Sudoku9 trenutno podržava samo standardne Sudoku zagonetke veličine 9x9. Za druge veličine zagonetki (kao što su 4x4, 6x6, 12x12, itd.), bile bi potrebne značajne izmene u dizajnu klase.