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

ISPITNI RAD

Kandidat: **Damjan Vinčić**
Broj indeksa: **SV58/2022**

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

Mentor rada: **dr Miodrag Đukić**

Novi Sad, decembar, 2023.

Sadržaj

• Uvod

Sudoku je popularna logička igra koja se sastoji od mreže od 9x9 polja podeljenih na manje kvadratne oblasti. Cilj je popuniti svako polje brojem od 1 do 9, tako da se u svakom redu, koloni i svakom od devet manjih kvadrati nalazi svaki od brojeva tačno jednom.

Iako je osnovna struktura 9x9 mreže najpoznatija, postoje različite varijante Sudoku-a koje izlaze van tradicionalnog formata. To uključuje veće rešetke (12x12, 16x16), varijacije sa dodatnim pravilima (ne samo brojevi, već i simboli, slova ili kombinacije), kao i specifične forme kao što su dijagonalan sudoku, Killer sudoku i mnoge druge.

Ima mnogo načina za rešavanje sudoku zagonetke, neki od njih su:

- **Backtracking** – Jedna od najčešćih tehnika za rešavanje sudoku-a. Ideja je da se probaju različiti brojevi na praznim mestima i ako dođe do prekršaja pravila, algoritam se vraća unazad i pokušava sa drugim vrednostima. Ovaj pristup može biti spor za neke slučajeve
- **Ograničavanje domena** – Koristi se skup pravila kako bi se suzila mogućnost brojeva koji se mogu smestiti u svako polje. Ovaj pristup koristi informacije o trenutnom stanju tabelek kako bi eliminisao nevalidne brojeve za svako polje, što smanjuje broj opcija koje algoritam mora razmatrati.
- **Optimizovani algoritmi pretrage** – Pored backtracking-a postoje i razni optimizovani algoritmi kako bi efikasnije istraživali moguće kombinacije i brže pronašli rešenje.

• Opis programa

Opis različitih delova programa.

◦ Argumenti komandne linije

Program koristi argumente komande linije kao unos fajlova. Prvi argument je fajl iz kog se čita sudoku zagonetka za rešavanje i upisuje zagonetka generisana od strane programa. Drugi argument je fajl iz kog se čita rešenje zadate sudoku zagonetke, ili upisuje rešenje zagonetke generisane od strane programa.

◦ Tok igre

Nakon pokretanja programa, korisnik bira da li želi da učita sudoku zagonetku iz fajla, koji je prethodno ručno uneo, ili želi da program generiše sudoku zagonetku i upiše je u fajl. Nakon toga ima izbor da li želi da program učita rešenje sudoku zagonetke iz fajla koji je korisnik prethodno uneo ili je već upisano rešenje od strane programa. Drugi izbor je da traži od programa da reši zagonetku učitanu iz fajla i upiše je kao rešenje. Zatim program proverava da li je rešenje validno (da je popunjeno, bez praznih mesta, i da je tabla ista kao tabla zagonetke koja se rešava), ispisuje rešenje na standardni izlaz i ispisuje statistiku, koliko brojeva na

dobrom mestu, koliko na lošem, i koji je trenutni redni broj partije. Nakon završene runde, korisnik bira da li želi da izađe iz programa ili nastavi da igra.

○ Način rešavanja

Za rešavanje sudoku zagonetke koristimo klasičan backtracking algoritam. Dok je backtracking uglavnom spor i ima boljih rešenja, dovoljan je za veliki broj slučajeva table dimenzije 9x9. Može biti veoma spor za određene slučajeve.

Za reprezentaciju sudoku table koristimo vektor. Iako bi u ovom slučaju niz bio dovoljan, ograničen je predefinisanom veličinom. Vektor nam pruža dodatne metode nad strukturom podataka ako nam zatrebaju, možemo ga lakše proširiti na sudoku proizvoljne dužine uz izmenu logike algoritma i indeksiranja, ne moramo da brinemo o oslobađanju memorije u destruktoru jer se automatski oslobađa.

○ Struktura ulazne i izlazne datoteke

Umesto običnog ispisa vrednosti sudoku table, ubacili smo formatiranje za lakši pregled:

```
2 1 3 | 4 5 6 | 8 7 9
4 8 7 | 1 2 9 | 3 5 6
6 9 5 | 3 7 8 | 2 4 1
------+-------+------
1 2 4 | 5 8 3 | 9 6 7
3 7 9 | 6 1 2 | 5 8 4
5 6 8 | 7 9 4 | 1 3 2
------+-------+------
8 4 1 | 2 3 7 | 6 9 5
9 5 6 | 8 4 1 | 7 2 3
7 3 2 | 9 6 5 | 4 1 8
```

○ Pokretanje

Preduslovi:

- **CMake**

U korenu projekta napraviti prazan folder:

$ mkdir build

Prebaciti se u novi folder:

$ cd build

Generisati potrebne fajlove:

$ cmake ..

Buildovati projekat:

$ cmake –build .

Pokretanje projekta:

$ ./sudoku input.txt output.txt

Pokretanje testova:

$ ./sudoku_test

Imena kreiranog foldera i fajlova su proizvoljna.

Primeri komandi su dati na Linuxu, za Windows se verovatno malo razlikuju.

• Klase

Opis klasa sa njihovim atributima i metodama.

• Sudoku9

Klasa **Sudoku9** predstavlja sudoku igru sa funkcionalnostima vezanim za manipulisanje table, rešavanje i statistiku.

• Atributi

- **originalBoard** – Reprezentuje originalnu sudoku tablu.
- **board** – Reprezentuje trenutno stanje table.
- **numberOfValidCells** – Prati broj dobro unetih brojeva na sudoku tabli.
- **numberOfInvalidCells** – Prati broj loše unteih brojeva na sudoku tabli.
- **numberOfGames** – Statički atribut koji prati broj odigranih partija.

- ## Konstruktori
  - **Sudoku9()** - Prazan konstruktor koji inicijalizuje atribute na default vrednosti.
  - **Sudoku9(const Sudoku9& other)** – Konstruktor kopije, prima referencu na Sudoku9 objekat za kopiranje
  - **Sudoku9(const std::vector<std::vector<unsigned short>>& board)** – Konstruktor koji prima dvodimenzionalni vektor za inicijalizaciju tabli.

- ## Metode

  Metode **Sudoku9** klase.

- ## Operator dodele

  ```
  Sudoku9& operator=(const Sudoku9& other);
  ```

  Operator dodele Sudoku9 klase.

  Parametri:

  - **other** – referenca na Sudoku9 objekat za dodelu

  Povratne vrednosti:

  - Referenca na dodeljeni Sudoku9 objekat radi ulančavanja operatora dodele.

- ## Dobavljanje originalne table

  ```
  Board getOriginalboard() const;
  ```

  Vraća originalnu tablu.

  Povratne vrednosti:

  - Originalna tabla – objekat klase **Board**

- **Dobavljanje trenutnog stanja table**

```
Board getBoard() const;
```

Vraća trenutno stanje table.

Povratne vrednosti:

- Trenutno stanje table – objekat klase **Board**

- **Rešavanje sudoku table**

```
bool solveSudoku();
```

Funkcija rešava sudoku tablu korišćenjem backtracking algoritma. Prolazi kroz sva prazna polja table i proverava da li se broj može staviti da zadovoljava uslove pozivajući odvojene funkcije, ako ni jedan broj ne zadovoljava uslove, vraća se u nazad i pokušava dalje.

Povratne vrednosti:

- **true** – uspešno rešen sudoku.
- **false** – nemoguće rešiti sudoku tablu.

- **Generisanje sudoku table**

```
void generateSudoku();
```

Funkcija generiše novi sudoku u skladu sa pravilima, ažurira originalnu i trenutnu tablu. Tabla se generiše tako što se prvo popune 3 dijagonalna kvadrata dimenzije 3x3 jer ne moramo da obraćamo pažnju na redove i kolone već samo da se brojevi ne ponavljaju više puta u podtablama. Zatim rešimo tu tablu i izbacimo nasumičan broj brojeva između 40 i 64 jer mora biti dostupno bar 17 brojeva da bi tabla bila validna.

- **Provera ispravnosti rešenja**

```
bool checkSolution();
```

Funkcija proverava da li je uneto rešenje ispravno. Koristi orginalnu tablu da vidi koje pozicije su unete, za svaku poziciju proverava da li uneti broj poštuje pravila igre.

Povećava broj dobro i pogrešno unetih polja u odnostu na proveru ispravnosti.

Povratne vrednosti:

- **true** – Ako je tabla rešenja ista kao originalna i popunjena.
- **false** – Ako tabla rešenja nije ista kao originalna ili nije popunjena.

- **Dobavljanje broja dobro unetih polja**

```
int getNumberOfValidCells() const;
```

Funkcija vraća broj dobro unetih polja u trenutnoj tabli.

Povratne vrednosti:

- **int** – Broj dobro unetih polja u trenutoj tabli.

- **Dobavljanje loše unetih polja**

```
int getNumberOfInvalidCells() const;
```

Funkcija vraća broj loše unetih polja u trenutoj tabli.

Povratne vrednosti;

- **int** – Broj loše unetih polja.

- **Dobavljanje broja odigranih rundi**

```
static int& numberOfGamesPlayed();
```

Funkcija vraća broj odigranih rundi, istovremeno se koristi i za izmenu broja rundi na kraju svake partije.

Povratne vrednosti:

- **int** – Broj odigranih rundi.

- **Postavljanje rešenja sudoku table**

  void setSolution(const std::vector<std::vector<unsigned short>>& board);

  Funkcija postavlja rešenje sudoku table.

  Parametri;

  - **board** – Dvodimenzionalni vektor koji predstavlja rešenje sudoku table.

- **Board**

Klasa Board predstavlja sudoku tablu i sadrži metode za manipulaciju tablom i validaciju.

- **Atributi**
  - **board** – Dvodimenzionalni vektor koji predstavlja sudoku tablu.

- **Konstruktori**
  - **Board()** - Prazan konstruktor koji inicijalizuje dvodimenzioni vektor dimenzija 9x9 i postavlja inicijalne vrednosti na 0.
  - **Board(const Board& other)** – Konstruktor kopije, prima referencu na Board objekat za kopiranje
  - **Board(const std::vector<std::vector<unsigned short>>& board)** – Konstruktor koji prima dvodimenzionalni vektor za inicijalizaciju table.

- ## Metode

  Metode **Board** klase.

- ## Operator dodele

  ```
  Board& operator=(const Board& other);
  ```

  Operator dodele Board klase.

  Parametri:

  - **other** – referenca objekat klase Board za dodelu

  Povratne vrednosti:

  - Referenca na dodeljeni Board objekat radi ulančavanja operatora dodele.

- ## Provera da li se broj može upisati u podtablu

  ```
  bool isSubgridSafe(int row, int col, unsigned short num);
  ```

  Funkcija proverava da li je broj jedinstven u podtabli u kojoj se polje nalazi.

  Parametri:

  - **row** – Red koji se proverava
  - **col** – Kolona koja se proverava
  - **num** – Broj koji se proverava da li može biti upisan

  Povratne vrednosti:

  - **true** – U polje se može upisati broj po pravilima igre.
  - **false** – U polje se ne može upisati broj.

- **Provera da li se broj može upisati u tablu**

```
bool isSafe(int row, int col, unsigned short num);
```

Funkcija proverava da li se broj može upisati u dato polje. Da li u datom redu, koloni i podtabli postoji broj sa istom vrednošću. Poziva isSubgridSafe() funkciju.

Paramteri:

- **row** – Red koji se proverava
- **col** – Kolona koja se proverava
- **num** – Broj koji se proverava da li može biti upisan

Povratne vrednosti:

- **true** – Broj se može upisati u polje
- **false** – Broj se ne može upisati u polje.

- **Popunjavanje dijagonalnih podtabli**

```
void fillDiagonalSubgrids():
```

Funkcija nasumično popunjava dijagonalne podtable vrednostima od 1 do 9, koristi se za generisanje table jer ne moramo da proveravamo da li broj može da se smesti u red ili kolonu, već samo da li može da se smesti u podtablu.

- **Ispis table**

```
friend std::ostream& operator<<(std::ostream& os, const Board& board);
```

Funkcija ispisuje tablu na prosleđeni output stream.

Parametri:

- **os** – Output stream
- **board** – Tabla za ispis

Povratne vrednosti:

- Referenca na prosleđeni output stream

- **Dobavljanje table**

  ```
  std::vector<std::vector<unsigned short>> getBoard() const;
  ```

  Funkcija vraća dvodimenzioni vektor koji predstavlja sudoku tablu.

  Povratne vrednosti:

  - **Dvodimenzioni vektor** – predstavlja sudoku tablu

• SudokuFileIO

Klasa za funkcionalnosti čitanja iz fajla i pisanja u fajl.

• Metode

Metode **SudokuFileIO** klase.

• Čitanje iz fajla

```
static std::vector<std::vector<unsigned short>> read(const std::string& filename);
```

Funkcija čita sudoku tablu iz fajla i vraća dvodimenzioni vektor.

Parametri:

- **filename** – Fajl iz koga se čita sudoku tabla.

Povratne vrednosti:

- Dvodimenzioni vektor koji reprezentuje učitanu tablu.

Baca izuzetak:

- **std::runtime_error** – Kada ne može da se otvori prosleđeni fajl.

• Upis u fajl

```
static void write(const std::string& filename, const Sudoku9& sudoku);
```

Funkcija za upis sudoku table u fajl.

Parametri:

- **filename** – Fajl u koji se upisuje sudoku tabla.
- **sudoku** – Objekat klase Sudoku9 čija tabla se upisuje u fajl.

Baca izuzetak:

- **std::runtime_error** – Kada ne može da se otvori posleđeni fajl.

- **Testiranje**

Za testiranje funkcija smo koristili unit testove, upotrebili smo Google Tests biblioteku da nam olakša pravljenje testova. Za svaku klasu smo napravili posebnu grupu testova odvojenih u poseban fajl. Većina funkcija je testirana sa više slučaja, očiglednim slučajem da treba da prođe, i nekim slučajevima koji bi trebalo da je obore.