Dokumentace – Sudoku

Verze projektu

Commit hash: 783c5e36cfd58f7d32ee6345ae5a4a1d6e064827

1. Popis řešených algoritmů

Aplikace implementuje základní logiku hry Sudoku pro velikosti 4×4, 6×6 a 9×9.

Základní algoritmy zahrnují:

1.1 Vytvoření herní desky

Třída **Board** vytváří dynamickou dvourozměrnou matici:

- velikost je dána parametrem boardSize,
- každé pole je inicializováno hodnotou 0 (prázdné pole),
- implementováno metodou createBoard().

Algoritmus je jednoduchý dvojitý cyklus s časovou složitostí: O(n2)

1.2 Validace tahu

Třída **Validator** ověřuje, zda lze vložit číslo na danou pozici.

Kontroluje se:

1. **Pozice je v rozsahu desky**
2. **Pole je prázdné**
3. **Číslo není v řádku**
4. **Číslo není ve sloupci**
5. **Číslo není v příslušném bloku** (2×2, 2×3, 3×3)

Každá kontrola je lineární: O(n)

Celková validace je tedy: O(n)

1.3 Kontrola dokončení hry

Metoda isSudokuSolved() ověřuje, zda:

- žádné pole není 0.

Neověřuje správnost řešení — pouze úplnost.

1.4 Načítání desky ze souboru

Metoda loadBoard():

- otevře soubor,
- načítá trojice row col value,
- volá changeBoard().

Soubor tedy reprezentuje pouze změny, ne celou desku.

2. Popis postupu řešení práce a implementovaných tříd

Aplikace je rozdělena do modulů podle MVC principů.

2.1 Třída Board

Odpovědnost

Reprezentuje stav herní desky.

Atributy

- boardSize – velikost desky (4, 6, 9)
- board – 2D vektor hodnot

Metody

Metoda	Popis
createBoard()	Vytvoří prázdnou desku.
changeBoard(r,c,v)	Zapíše hodnotu na pozici.
loadBoard(path)	Načte změny desky ze souboru.
deleteBoard()	Vymaže celou desku.
getBoard()	Vrací kopii celé desky.
getBoardSize()	Vrací velikost desky.
setBoardSize(size)	Nastaví velikost desky.
getBlockRow()	Vrací počet řádků v bloku.
getBlockCol()	Vrací počet sloupců v bloku.

Poznámky k implementaci

- getBoard() vrací **kopii**, ne referenci → bezpečné, ale pomalejší.

2.2 Třída Renderer

Odpovědnost

Zobrazuje herní desku v terminálu.

Funkce

- používá ANSI escape sekvence pro:
    - vymazání obrazovky,

    - zvýraznění kurzoru (\033[43m),
- vykresluje bloky pomocí oddělovacích čar.

Metody

Metoda	Popis
render(board, cursorRow, cursorCol)	Vykreslí celou desku.
printWelcome()	Vypíše úvodní zprávu.
print(message)	Vypíše libovolný text.

Poznámky

- Vykreslení je kompletní refresh celé obrazovky → jednoduché, stabilní.
- Bloky se oddělují podle blockRows a blockCols.

2.3 Třída Validator

Odpovědnost

Kontroluje validitu tahu a stav hry.

Metody

Metoda	Popis
isCorrectNumber(from,to,number)	Ověří rozsah čísla.
isCorrectPosition(row,col,board,number)	Ověří, zda lze vložit číslo.
isSudokuSolved(board)	Ověří, zda je deska kompletní.

Validace pozice

Probíhá v tomto pořadí:

1. kontrola hranic,
2. kontrola, zda je pole prázdné,
3. kontrola řádku,
4. kontrola sloupce,
5. kontrola bloku.

Poznámky

- Validátor neřeší logické chyby v řešení (např. dvě stejná čísla v jiném bloku).
- Validátor neřeší generování řešitelných desek.

3. Popis kompilace a spuštění

3.1 Kompilace pomocí CMake

Doporučený způsob:

1. mkdir build
2. cd build
3. cmake ..
4. cmake --build .

Spuštění: ./mini-sudoku

4. Přepínače programu

Program podporuje: --help

Vypíše:

- popis hry,
- ovládání,
- způsob spuštění.

Ukázka: ./mini-sudoku --help

5. Ovládání aplikace

Menu:

1. výběr velikosti desky,
2. potvrzení Enterem.

Hra:

- šipky → pohyb kurzoru,
- čísla → vložení hodnoty,
- q → ukončení hry.

6. Testování programu

6.1 Testovací scénáře

Test	Popis	Očekávaný výsledek
1	Vložení čísla do prázdného pole	Tah povolen
2	Vložení čísla mimo rozsah	Tah odmítnut
3	Vložení čísla do obsazeného pole	Tah odmítnut
4	Duplicitní číslo v řádku	Tah odmítnut
5	Duplicitní číslo ve sloupci	Tah odmítnut
6	Duplicitní číslo v bloku	Tah odmítnut
7	Kompletní vyplnění desky	isSudokuSolved() vrací true

6. Testování programu

Testování probíhalo kombinací **automatických jednotkových testů** (Catch2) a **manuálního ověřování chování aplikace** v terminálu. Cílem bylo ověřit správnost implementovaných algoritmů, stabilitu aplikace a odhalení případných chyb v logice hry.

6.1 Automatické testy (Catch2)

Pro projekt byla vytvořena sada jednotkových testů pokrývající klíčové části aplikace:

- práci s herní deskou (Board),
- validaci tahů (Validator),
- načítání desky ze souboru,

- kontrolu dokončení hry,
- základní chování Rendereru.

Testy jsou umístěny ve složce tests/ a jsou kompilovány jako samostatný spustitelný soubor sudoku_tests.

Spuštění testů

./sudoku_tests

Testovací scénáře

Test	Popis	Očekávaný výsledek
1	Vložení čísla do prázdného pole	Tah povolen
2	Vložení čísla mimo rozsah	Tah odmítnut
3	Vložení čísla do obsazeného pole	Tah odmítnut
4	Duplicitní číslo v řádku	Tah odmítnut
5	Duplicitní číslo ve sloupci	Tah odmítnut
6	Duplicitní číslo v bloku	Tah odmítnut

7	Kompletní vyplnění desky	isSudokuSolved() vrací true

Testy pokrývají jak běžné situace, tak i chybové stavy, aby byla ověřena robustnost aplikace.

6.2 Analýza paměti (Valgrind)

Aplikace byla otestována pomocí nástroje **Valgrind**, který slouží k detekci úniků paměti a neplatných přístupů.

Testováno bylo:

- spuštění hlavní aplikace (mini_sudoku),
- spuštění testů (sudoku_tests).

Výsledek:

Nebyly detekovány žádné úniky paměti ani neplatné přístupy.

> Aplikace korektně uvolňuje všechny dynamicky alokované struktury.

6.3 Manuální testování

Kromě automatických testů proběhlo i manuální ověřování:

- výběr velikosti desky v menu,
- pohyb kurzoru pomocí šipek,
- zadávání hodnot,
- mazání hodnot,
- ukončení hry klávesou q,
- vykreslování desky v různých stavech,
- validace tahů v reálném čase.

Aplikace se ve všech testovaných scénářích chovala stabilně a bez chyb.

7. Srovnání algoritmů

V projektu jsou implementovány dvě hlavní strategie validace:

A) Lineární kontrola (aktuální implementace)

- jednoduchá,
- přehledná,
- časová složitost O(n).

B) Hash-set kontrola (alternativní možnost)

- rychlejší detekce duplicit,
- složitost O(1) na kontrolu prvku,
- vhodné pro větší desky.

Výsledek srovnání

Na testovaných vstupech:

- u 4×4 a 6×6 rozdíl zanedbatelný,
- u 9×9 je hash varianta rychlejší při opakovaných validacích.