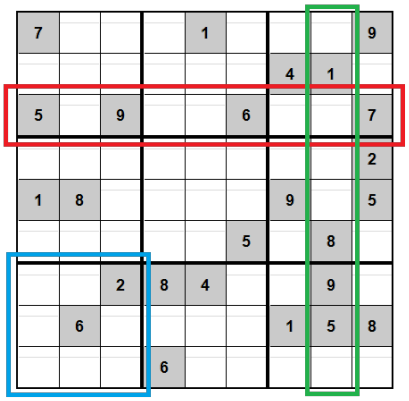
Seliverstov Mark, 1 ročník
Zimní semestr 2021/22
Programování NPRG030

---

# *Dokumentace (sudoku)*

- Program řeší Sudoku *4x4, 9x9 a 16x16*. Používá se rekurzivní algoritmus, který prochází všemi hodnotami.

**Sudoku** je logická hra s číslicemi. Cílem hry v je doplnit chybějící cifry v zadané, zčásti vyplněné čtvercové tabulce. K předem vyplněným číslicím je třeba doplnit další číslice tak, aby platilo, že v každém řádku, v každém sloupci a v každém dílčích čtverců jsou použity vždy všechny číslice, každá právě jednou.




Na vstup programu je zadána složitost (4, 9 nebo 16) a pak zadán řádek po řádku Sudoku přes mezeru (prázdná buňka = 0). Hodnoty jsou zapsány do seznamu.
Poté dojde k ověření zadaných hodnot pro správnost:
- Pokud je zadání *správné*, program spustí funkci, která rekurzivně přebírá čísla, dokud nenajde řešení.
- Pokud je zadání *nesprávné*, pak program požádá o zadání hodnoty znovu

Jakmile program našel správné řešení, zobrazí správné řešení a zeptá se, zda uživatel chce vyřešit nový Sudoku.

## *Algoritmus řešení(rekurzivní)*

1. Je prázdná hodnota («0»)
2. Hodnota je kladena na prázdné místo (od 1 do "složitosti")
3. Ověřuje správnost Sudoku (aby platilo, že v každém řádku, v každém sloupci a v každém dílčích čtverců jsou použity vždy všechny číslice, každá právě jednou
4. Pokud kontrola proběhne, vrátí se k prvnímu bodu. Pokud ne, zkontroluje se následující číslo (návrat k bodu 2)

*Datové struktury použité v programu:*

• Podprogram «`NapisVelikost`»:
*funkce, která žádá o zadání složitosti (Velikost) sudoku. Pokud není zadána 4, 9 nebo 16, program požádá o opakování zadání.*

• Podprogram «`NapisData`» :
*funkce, která požádá o zadání známých hodnot do pole sudoku na samostatném řádku přes mezeru a namísto prázdných hodnot je nastavena «0». Pokud není v řádku dostatek hodnot, program bude vyžadovat opětovné zadání tohoto řádku. Pro ukládání sudoku se používá seznam.*

• Podprogram «`check`» :
Funkce kontroluje v každém řádku, v každém sloupci a v každém dílčích čtverců jsou použity vždy všechny číslice, každá právě jednou.

• Podprogram «`reseni`» :
hlavní funkce, která řeší Sudoku.

• Podprogram «`vystup`» :
funkce která přináší řešení v čitelné podobě

hlavní proměnné:
`grid(list)` – seznam který obsahuje pole sudoku
`end(boolean)` – Pokud je konec - True; pokud ne - False
`jet(boolean)` – když program najde řešení, rekurze se zastaví a program přestane měnit hodnoty v poli grid. Tato proměnná je zodpovědná za ukončení změny pole
`velikost(int)` – složitost algoritmu (4, 9 nebo 16)
`correctly(boolean)` – pokud zadaný Sudoku prošel kontrolou - True; pokud ne - False

Možné zlepšení:

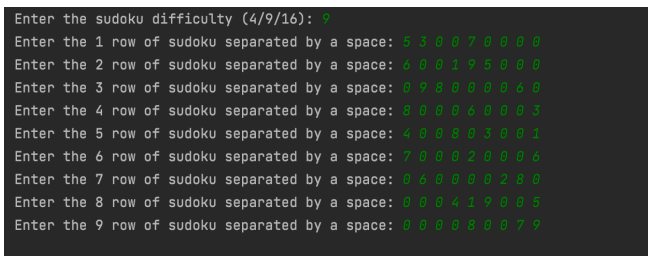
• Použití algoritmus bez rekurze, aby program běžel rychleji.
• Použití grafického rozhraní, aby bylo pro uživatele pohodlnější používat program.

Ukázka fungování programu:

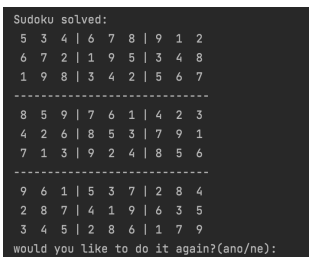
Jakékoliv Sudoku

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

Vstup dat




Výstup dat



*Jak probíhala práce na projektu*

Nejprve jsem se začal dozvědět více o Sudoku: příběh tvorby, jaké jsou jeho verze a tak dále, bylo to velmi zajímavé a poučné;
Pak se uvažovalo o algoritmu řešení problému a nebylo vymyšleno nic lepšího než rekurze;
Pak bylo třeba udělat, aby se při návratu rekurze nezměnily hodnoty v poli. To bylo provedeno pomocí proměnné `«jet»;`
Dále bylo nutné provést ochranu před nesprávným zadáním;
Na konci bylo třeba udělat čitelný závěr výsledku.

Chtěl bych ještě udělat grafické rozhraní, bylo by to zajímavé, ale nepodařilo se to udělat. nicméně, jak si myslím, že jsem zvládl projekt, bylo to docela zajímavě.