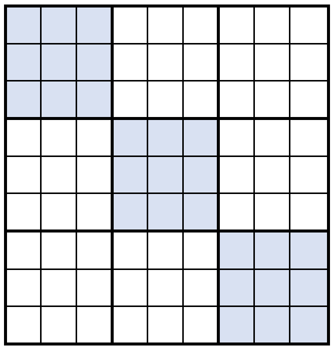
CS510 - Retrospective Project Report

Sudoku - Kamakshi Nagar

Overview -

My original plan was to generate a sudoku by filling three 3*3 diagonal grids first(shown below in Plan1) as they are independent of their neighboring matrices and then use backtracking to fill the remaining cells [1]. Somehow it didn't work out in first trial.

Plan 1



Plan 2

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

So I used another idea (2) where the grid is filled with numbers 1..9 as shown in above figure 2. The first row has numbers 1..9 and then row 1 is left shifted by one cell to get row 4. Similarly row 7 is row 4 left shifted by one and then row 2,5,8,3,6,9 are filled one left shift each. This way all the number lies within the rules of a sudoku.

To shuffle the puzzle, rows 1, 2 and 3 can be swapped with each other and columns 1, 2 and 3 can be swapped with each other. This shuffle preserves the integrity of the sudoku as when you swap entire row or column within the box (3*9 box shaded blue in plan2), there are no row, column or box conflict. Similarly other rows and columns can be swapped in their 3*9 or 9*3 boxes too. Shuffling for about 500 times produces decent sudoku.

To make it a game, remove some numbers randomly by assigning 0s to them. Solve the sudoku using backtracking, the solution need not be unique. The solver was able to solve the sudoku.

Then I applied same solver on my first sudoku generating approach it worked too. So now there are two ways to generate sudokus.

Once the logic part was done, I explored Piston crate's Sudoku Tutorial for my project. Piston has solid Model View Controller system for building games like sudoku, Tetris etc. I was successfully able to use the model I build before and now had an UI for playing the puzzle. Rust crates.io have so many useful crates I used rand for generating random numbers.

Learnings

- Rust is a safe language and memory management is quite efficient. I learned about Struct, Array, Vector (withcapacity, collect), Strings, Char, iterators (Range, Chars, map, filter, fold), match , loops , Enums during this project. I used unsafe for static mutability.
- Type inference- It is nice that I do not have to define every time what type my variable is. Rust Compiler does it best to guess the right type. If it cant then it just asks programmer to do so.
- Compiler messages- when I started learning rust and compiled first program it was full of compile errors. Once I understood how lifetime, ownership and borrowing works the number of errors reduced. Compiler messages and help makes it easy to understand how rust compiler works under the hood. And few things are really magic. Like when we don't have to dereference something and compiler does it for you, it's a relief.
  I have exploited this feature when implementing my project. When I didn't know the return type of some function, I would just say let x = fun(); and will know the return type. When compiler says that something is of type bool and you are giving unit, then its easy to fix that by making it return correct type.
- Garbage collector- There are no dangling pointers or unassigned memory in safe rust. The variable is dropped when it is out of scope or at the end of its lifetime.
- UTF encoding – Strings were quite painful in rust when I started. It requires good understanding of ASCII and UTF8 to handle Strings in rust.

Conclusion

It was my first project where I designed and implemented a Game. The project has ability to generate unique sudoku two ways, can solve the puzzle and UI for user to play the game. While implementing, rust borrow checker did not bother too much. There were no undefined behavior once the program compiled it ran without issue.

Rust documentation and the book(Programming in Rust) is quite helpful to understand concepts. Rust library has lot of useful methods for String, Char, Iterator etc. Crates are easy to import when we need other external features.

Reference:

1. https://www.geeksforgeeks.org/program-sudoku-generator/
2. https://stackoverflow.com/questions/4066075/proper-data-structure-to-represent-a-sudoku-puzzle