**Project description:**

This project is a Sudoku solver which can solve any Sudoku problem in a very short time.

**File description:**

There are three c++ files in the folder for better readability. The main.cpp is the main function, includes the timing operations, eliminator.cpp contains high leveled functions, to be specific, eliminateCandidate(), eliminateGroup() and makeAssumption() represent the three steps descripted in the PPT. And, the HelpFunctions.cpp is some low leveled functions invoked by other function from the other two cpp files.

The main.exe is the file to run, and you can recompile main.cpp to generate it.

The input.txt is the place where you can put your puzzle in. One puzzle per time.

There should be an output.txt once you run the program.

**Time efficiency and strategies description:**

All the tests I have ran can be done within 0.01 second as you can find when testing.

In my program, each step contains the previous step(s). In detail, the group elimination step invokes the candidate elimination for relevant cells, and brute force assumption step contains the group elimination operation as a forward checking process. And the cell to make assumption is chosen by both low possibility and high number of fixed relevant cells.

This program used a depth-first search tree with forward checking. Each time the assumption process reached a cell (node), it makes assumptions based on the possibilities it has, and eliminate possibilities of relevant cells, then perform a forward checking in the respect to terminate fail assumptions as soon as possible.

**Algorithms in my program:**

Data structure:

I used a 9\*9\*10 array to represent the whole board, each cell has 10 integer elements for better understanding. The first element is the possibility of current cell, and number “1” in the following cells means the cell could be assigned the position number, (for example, if array[7] is 1, this means the cell could be a 7), the possibility should equal to the number of “1” appeared in the following 9 slots.

For candidate elimination:

Start

For each cell within the board:

If possibility greater than 1

Determine possibility by relevant cells

If possibility equal to 1

Eliminate this fixed number from relevant cells

End

For group elimination:

Start

For each cell within the board:

Find groups by row, column and square

Eliminate these numbers

End

For make assumption:

Start

Backup current board

Find the cell with lowest possibility and has the biggest number of fixed relevant cells

For possible assumptions

Make assumption

Eliminate this fixed number from relevant cells

If succeed

Check if the board were finished

If finished

Return true

Do a forward checking by group elimination

If no violation

Do make assumption recursively

If succeed return true

If failed restore the board

If reached here then return false

End