**CS 280 - Programming Assignment #3**

Due Friday, October 24, 2014 at 2:30pm

**Information**

* [Handout](Handout.docx)
* The header file: [Text](Sudoku.h)
* A sample driver program: [Text](https://distance.sg.digipen.edu/file.php/1659/assignment3/driver-sample.cpp)
* Here's a [Win32 executable](getopt.zip) for you to experiment with.
* Assuming the name of the executable is **sudoku** (or **sudoku.exe**) Typing **sudoku** at the command line will print this:

Usage: sudoku [options] input\_file

Options:

-a --show\_solution=X show final board (0-OFF, 1-ON, default is ON).

-b --basesize=X the size of the board (e.g. 3, 4, 5, etc. Default is 3)

-c --clear\_screen clear the screen between boards.

-e --show\_boards\_every=X only show boards after every X iterations.

-h --help display this information.

-m --show\_messages show all messages (placing, removing, backtracking,

etc.)

-p --pause pause between moves (press a key to continue).

-s --show\_boards show boards after placing/removing.

-t --symbol\_type=X the type of symbols to use, numbers or letters (0 is

numbers, 1 is letters)

-w --show\_stats=X show statistics after each move.

These options give you control over how much output to show. You can just show the initial and final boards or you can show the board after every move. This can potentially generate lots of output, but is essential for debugging your program. By pausing the program after every board, you can effectively single-step through the boards to see how your algorithm is working.

* [Sample input and output](outputs):

1. A w in the output name means it will show statistics.
2. An s in the name means show all boards.
3. An m means to show all messages (starting, placing, removing, etc.)

Because showing all boards produces a lot of output, it is only in the first example. Please work with the 9x9.txt example first to ensure that you are doing things correctly. The command line to run the program is in parentheses. Check the sample output here.

* + 9x9.txt       [output9x9w.txt](https://distance.sg.digipen.edu/file.php/1659/assignment3/output9x9msw.txt)      [output9x9msw.txt](https://distance.sg.digipen.edu/file.php/1659/assignment3/output9x9msw.txt) (Very verbose, sudoku 9x9.txt -msw)
  + board3-1.txt      output3-1w.txt       (sudoku board3-1.txt -w)
  + board3-2.txt      output3-2w.txt       (sudoku board3-2.txt -w -t1)
  + board3-3.txt      output3-3w.txt       (sudoku board3-3.txt -w)
  + board3-4.txt      output3-4w.txt       (sudoku board3-4.txt -w)
  + board3-5.txt      [output3-5w.txt](https://distance.sg.digipen.edu/file.php/1659/assignment3/output3-5w.txt)       (sudoku board3-5.txt -w)
  + board3-6.txt      [output3-6w.txt](https://distance.sg.digipen.edu/file.php/1659/assignment3/output3-6w.txt)       (sudoku board3-6.txt -w)
  + board4-1.txt      output4-1w.txt       (sudoku board4-1.txt -t1 -b4 -w)
  + board4-2.txt      output4-2w.txt       (sudoku board4-2.txt -t1 -b4 -w)
  + board4-3.txt      [output4-3w.txt](https://distance.sg.digipen.edu/file.php/1659/assignment3/output4-3w.txt)       (sudoku board4-3.txt -t1 -b4 -w)
  + board4-4.txt      [output4-4w.txt](https://distance.sg.digipen.edu/file.php/1659/assignment3/output4-4w.txt)       (sudoku board4-4.txt -t1 -b4 -w)
  + board5-1.txt      output5-1w.txt       (sudoku board5-1.txt -t1 -b5 -w)
  + board5-2.txt      output5-2w.txt       (sudoku board5-2.txt -t1 -b5 -w)
* Sample command lines: (On Windows, you'll need all of the files from this archive, <getopt.zip>, to be present in your directory where you are working. You don't need it with Cygwin (gcc/g++) on Windows, Mac OS X, or Linux as the functionality is already present on those platforms.)

**cl** /MT /W4 /O2 /EHa /Za /WX driver.cpp Sudoku.cpp /D\_CRT\_SECURE\_NO\_DEPRECATE **getopt.lib**

**bcc32** -v -vG -w driver.cpp Sudoku.cpp **getoptb.lib**

**g++** driver.cpp Sudoku.cpp -Wall -Wextra -ansi -pedantic -Wconversion –Werror

You need to make sure that you are sending the appropriate message at the correct time:

* + After you place a value on the board, you will send MSG\_PLACING.
  + Before removing a value from the board, you will send MSG\_REMOVING.
  + You will send MSG\_ABORT\_CHECK immediately before you place a value or remove a value. If this call returns true, you will terminate the search.
  + If, after placing a value you have found a tour, you will send MSG\_FINISHED\_OK and terminate the search.
  + If you do not find a solution after exhaustively checking, you will send MSG\_FINISHED\_FAIL.
* Sample pseudocode for the public interface method **Solve**:

**bool** Sudoku::Solve()

{

*// Initialize some private data.*

*// Call place\_value to place 1 at the starting point (row 1, column 1).*

*// Return true if a solution was found, otherwise, return false.*

}

**More Details**

Example boards: (From [KSudoku](http://www.kde.org/applications/games/ksudoku/))

|  |  |
| --- | --- |
| **Starting board (9x9)** | **Ending board (9x9)** |
| [https://faculty.digipen.edu/~mmead/www/Courses/2013/fall/cs280/project3/ksudoku-easy-begin-9x9.png](https://faculty.digipen.edu/~mmead/www/Courses/2013/fall/cs280/project3/ksudoku-easy-begin-9x9-full.png) | [https://faculty.digipen.edu/~mmead/www/Courses/2013/fall/cs280/project3/ksudoku-easy-end-9x9.png](https://faculty.digipen.edu/~mmead/www/Courses/2013/fall/cs280/project3/ksudoku-easy-begin-9x9-full.png) |
| **Starting board (16x16)** | **Ending board (16x16)** |
| [https://faculty.digipen.edu/~mmead/www/Courses/2013/fall/cs280/project3/ksudoku-easy-begin-16x16.png](https://faculty.digipen.edu/~mmead/www/Courses/2013/fall/cs280/project3/ksudoku-easy-begin-16x16-full.png) | [https://faculty.digipen.edu/~mmead/www/Courses/2013/fall/cs280/project3/ksudoku-easy-end-16x16.png](https://faculty.digipen.edu/~mmead/www/Courses/2013/fall/cs280/project3/ksudoku-easy-begin-16x16-full.png) |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Starting board (25x25)** | **Ending board (25x25)** | | **[https://faculty.digipen.edu/~mmead/www/Courses/2013/fall/cs280/project3/ksudoku-easy-begin-25x25.png](https://faculty.digipen.edu/~mmead/www/Courses/2013/fall/cs280/project3/ksudoku-easy-begin-25x25-full.png)** | **[https://faculty.digipen.edu/~mmead/www/Courses/2013/fall/cs280/project3/ksudoku-easy-end-25x25.png](https://faculty.digipen.edu/~mmead/www/Courses/2013/fall/cs280/project3/ksudoku-easy-begin-25x25-full.png)** | |  |  |  |
|  |  |  |  |

Searching reminders:

* You will always start in the upper-left corner and work towards the bottom-right corner.
* Be careful not to remove or change any of the original values that the board was started with.

**Pseudocode for a place\_value Function**

* This function takes a position (row/column or an index) and is called every time you want to place a value on the board.
* The function is initially called from the public Solve method with the position as the first cell (upper-left) on the board.
* You'll need to add the appropriate calls to the callback function in various places in your code.

Pseudocode: (assume 9x9)

1. Place a value at the specified location. (The first time you will be placing the value 1 in the first position,)
2. Increment the number of moves taken.
3. Increment the current move number.
4. If there is no conflict with any neighboring values then
   * If this was the last position left on the board:
     + The algorithm has finished, nothing left to do but to perform the proper callback.
   * If there are more empty cells:
     + call place\_move recursively with the location of the next empty cell. (This will put you at step #1 again.)
5. If there is a conflict:
   * If the value is less than 9:
     + Remove that value, increment it, and place it in the cell and check for conflicts.
   * If the value is 9:
     + Backtrack by removing the 9
     + Decrement the current move number.
     + Return from the function (It will return to the previous call and will likely be at step #7 ready to try the next position)

The details are left for you to figure out. Of course, this is not the only way, or necessarily the "right" way to approach the algorithm. However, it is straight-forward and relatively simple. Feel free to modify or tweak the algorithm to suit your needs. Make sure to send the appropriate messages via the callback function so the driver can update the display properly.