fantasy19 / Derp

Code Issues Pull requests Actions Projects

e681c03cbb

Derp / 280 / assignment03-Sudoku / Sudoku.h

fantasy19 No commit message

1 contributor

Raw Blame

95 lines (77 sloc) 2.95 KB

```
//---------------------------------------------------------------------------
#ifndef SUDOKUH
#define SUDOKUH
//---------------------------------------------------------------------------

/****************************************************************************/
/*!
\file   Sudoku.h
\author Ang Cheng Yong
\par    email: a.chengyong\@digipen.edu
\par    DigiPen login: a.chengyong
\par    Course: CS280
\par    Programming Assignment #3
\date   25/10/2016
\brief
This file contains the driver functions needed for Sudoku.
*/
/****************************************************************************/

#include <stddef.h> // size_t
#include<iostream>
#include <vector>

class Sudoku
{
  public:
      // Used by the callback function
    enum MessageType
    {
      MSG_STARTING,      // the board is setup, ready to go
```

```
      MSG_FINISHED_OK,   // finished and found a solution
      MSG_FINISHED_FAIL, // finished but no solution found
      MSG_ABORT_CHECK,   // checking to see if algorithm should continue
      MSG_PLACING,       // placing a symbol on the board
      MSG_REMOVING       // removing a symbol (back-tracking)
    };

      // 1-9 for 9x9, A-P for 16x16, A-Y for 25x25
    enum SymbolType {SYM_NUMBER, SYM_LETTER};

    const static char EMPTY_CHAR = ' ';

      // Implemented in the client and called during the search for a solution
    typedef bool (*CALLBACK)
      (const Sudoku& sudoku, // the gameboard object itself
       const char *board,    // one-dimensional array of symbols
       MessageType message,  // type of message
       size_t move,          // the move number
       unsigned basesize,    // 3, 4, 5, etc. (for 9x9, 16x16, 25x25, etc.)
       unsigned index,       // index of current cell
       char value            // symbol (value) in current cell
      );

    struct SudokuStats
    {
      unsigned basesize; // 3, 4, 5, etc.
      unsigned placed;   // the number of values the algorithm has placed
      size_t moves;      // total number of values that have been tried
      size_t backtracks; // total number of times the algorithm backtracked
      SudokuStats() : basesize(0), placed(0), moves(0), backtracks(0) {}
    };

      // Constructor
    Sudoku(int basesize, SymbolType stype = SYM_NUMBER, CALLBACK callback = 0);

      // Destructor
    ~Sudoku();

      // The client (driver) passed the board in the values parameter
    void SetupBoard(const char *values, size_t size);

      // Once the board is setup, this will start the search for the solution
    bool Solve();

      // For debugging with the driver
    const char *GetBoard() const;
    SudokuStats GetStats() const;

  private:
          // Other private fields and methods...
        size_t moves_;
        SudokuStats sStats;
```

```

        char * board;
        CALLBACK cb;
    size_t width;
        char first, last;
        bool place_value(size_t);
    bool ConflictCheck(size_t,char);



};

#endif  // SUDOKUH
```