Sudoku Project

Generated by Doxygen 1.8.11

Contents

Chapter 1

File Index

1.1 File List

Here is a list of all files with brief descriptions:

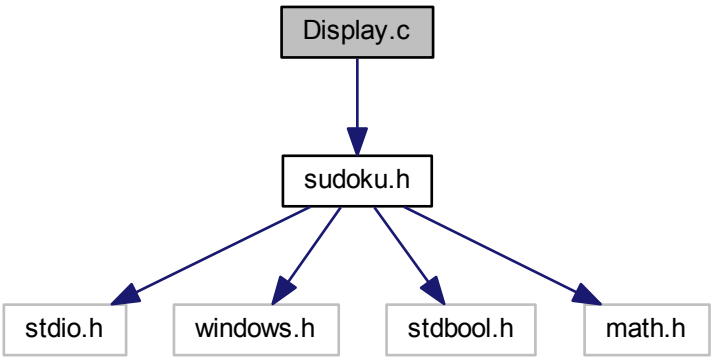
Chapter 2

File Documentation

2.1 Display.c File Reference

```
#include "sudoku.h"
```

Include dependency graph for Display.c:



Functions

- void **Intro** ()

  Generate intro interface.
- void **ChoseOptions** ()

  Generate chose options interface for switch .
- void **Back** (int puzzle, int copy, int **SizeOfPuzzle**)

  Call back-up of last puzzle.
- void **InsertNumberInPozitionChoice** (int puzzle, int copy, int **SizeOfPuzzle**)

  Insert number in pozition at your choice !
- void **GenerateRandomPuzzle** (int puzzle, int copy, FILE ∗nameOfFile, int **SizeOfPuzzle**)

  Generate puzzle.
- void **SolvePuzzle** (int puzzle, FILE ∗fileToPrint, int **SizeOfPuzzle**)

  A function which will call solving parts to solve the puzzle !
- void **AskSize** (int **SizeOfPuzzle**, int puzzle)

  Ask for size of puzzle .

2.1.1 Function Documentation

2.1.1.1 void AskSize (int *SizeOfPuzzle,* int *puzzle*)

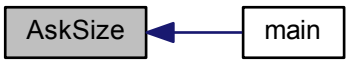
Ask for size of puzzle .

Parameters

puzzle	The respective puzzle
SizeOfPuzzle	Puzzle size.

Here is the call graph for this function:

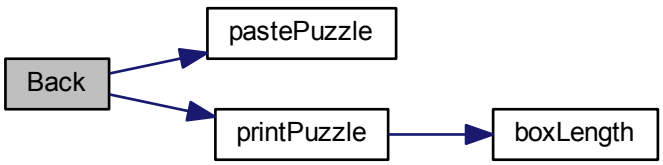
Here is the caller graph for this function:



2.1.1.2 void Back (int *puzzle,* int *copy,* int *SizeOfPuzzle*)

Call back-up of last puzzle.

Here is the call graph for this function:

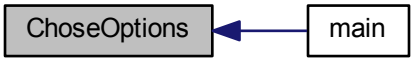


Here is the caller graph for this function:

2.1.1.3 void ChoseOptions ( )

Generate chose options interface for switch .

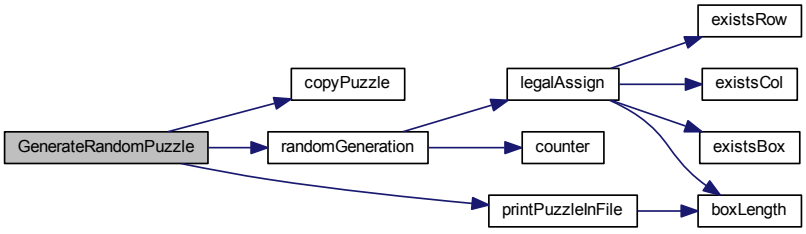
Here is the caller graph for this function:



2.1.1.4 void GenerateRandomPuzzle (int *puzzle,* int *copy,* FILE * *nameOfFile,* int *SizeOfPuzzle*)

Generate puzzle.

Here is the call graph for this function:

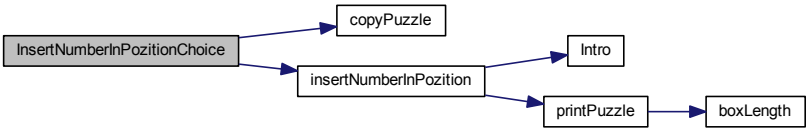


Here is the caller graph for this function:

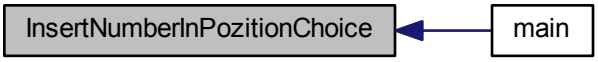
2.1.1.5 void InsertNumberInPozitionChoice (int *puzzle,* int *copy,* int *SizeOfPuzzle*)

Insert number in pozition at your choice !

Here is the call graph for this function:



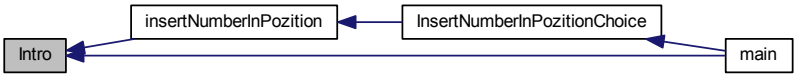
Here is the caller graph for this function:



2.1.1.6 void Intro ( )

Generate intro interface.

Here is the caller graph for this function:



2.1.1.7 void SolvePuzzle (int *puzzle,* FILE * *fileToPrint,* int *SizeOfPuzzle*)

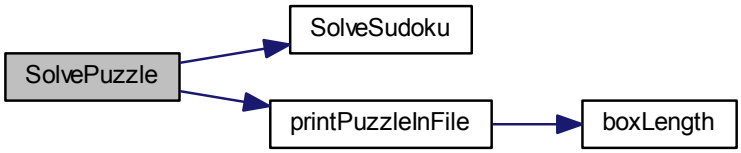
A function which will call solving parts to solve the puzzle !

The solution is printed on console display. /** Generate print interface.

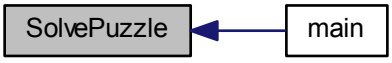
\ printPuzzle(puzzle, SizeOfPuzzle);

\ The solution is printed on Solutions file !

Here is the call graph for this function:



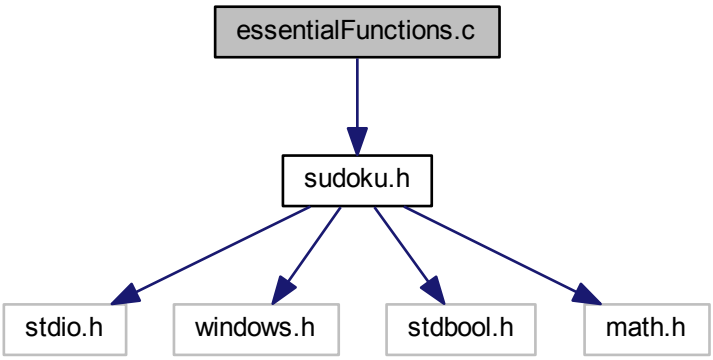
Here is the caller graph for this function:



2.2 essentialFunctions.c File Reference

```
#include "sudoku.h"
```

Include dependency graph for essentialFunctions.c:



Functions

- int **boxLength** (int **SizeOfPuzzle**)

  A function which will determine box size.
- void **copyPuzzle** (int puzzle[**SizeOfPuzzle**][**SizeOfPuzzle**], int copy[**SizeOfPuzzle**][**SizeOfPuzzle**])

  A function to BackUp the puzzle.
- void **pastePuzzle** (int puzzle[**SizeOfPuzzle**][**SizeOfPuzzle**], int copy[**SizeOfPuzzle**][**SizeOfPuzzle**])

  An another function to BackUp the puzzle.
- void **printPuzzle** (int puzzle[**SizeOfPuzzle**][**SizeOfPuzzle**], int **SizeOfPuzzle**)

  Function which will print puzzle.
- void **printPuzzleInFile** (FILE ∗filename, int puzzle[**SizeOfPuzzle**][**SizeOfPuzzle**], int **SizeOfPuzzle**)

  Function which will print puzzle in a file.
- void **insertNumberInPozition** (int puzzle[**SizeOfPuzzle**][**SizeOfPuzzle**], int **SizeOfPuzzle**)

  Function which will insert number in specificated position.
- int **Ask** ()

  Ask function as well as it's called just ask user which will be the size of puzzle.
- int **counter** (int puzzle[**SizeOfPuzzle**][**SizeOfPuzzle**])

  Counter function will return how many numbers shoud be finded in our sudoku puzzle.
- ∗param file Data file ∗bool **OpenFiles** (FILE ∗file)

  A function which will check if file is open.

2.2.1 Function Documentation

2.2.1.1 int Ask ( )

Ask function as well as it's called just ask user which will be the size of puzzle.

Here is the caller graph for this function:

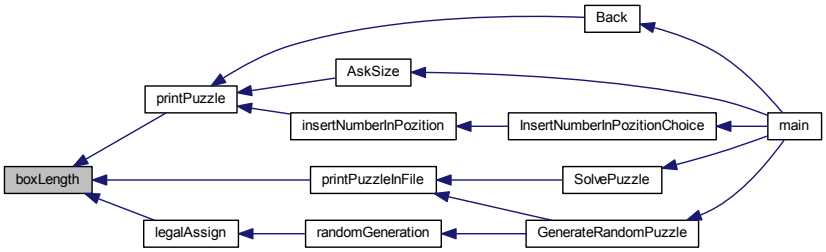
2.2.1.2 int boxLength (int *SizeOfPuzzle*)

A function which will determine box size.

Parameters

SizeOfPuzzle	Puzzle Size

Here is the caller graph for this function:



2.2.1.3 void copyPuzzle (int *puzzle[SizeOfPuzzle][SizeOfPuzzle],* int *copy[SizeOfPuzzle][SizeOfPuzzle]*)

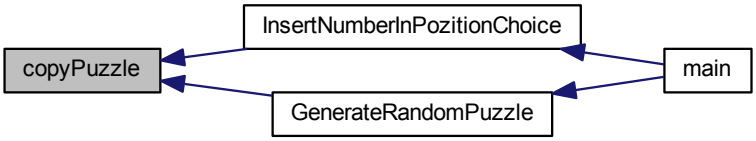
A function to BackUp the puzzle.

Parameters

puzzle	The respective puzzle
copyOfPuzzle	

Store the matrix into other matrix.

Here is the caller graph for this function:



2.2.1.4 int counter (int *puzzle[SizeOfPuzzle][SizeOfPuzzle]*)

Counter function will return how many numbers shoud be finded in our sudoku puzzle.

Parameters

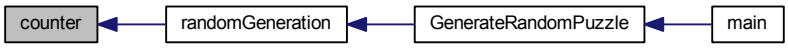
puzzle	The respective puzzle
SizeOfPuzzle	Puzzle Size

Declare a counter.

Increase counter when pozition is not equal to 0.

Return counter value.

Here is the caller graph for this function:



2.2.1.5 void insertNumberInPozition (int *puzzle[SizeOfPuzzle][SizeOfPuzzle],* int *SizeOfPuzzle*)

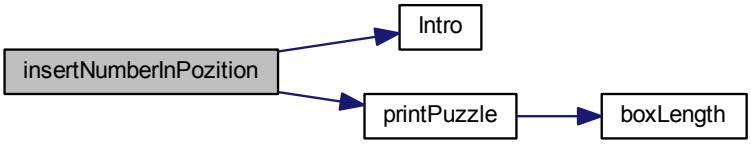
Function which will insert number in specificated position.

Parameters

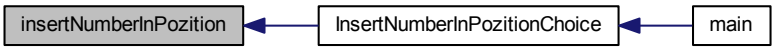
puzzle	The respective puzzle
SizeOfPuzzle	Puzzle Size

Use clear tool to delete the console

Here is the call graph for this function:



Here is the caller graph for this function:



2.2.1.6 ∗ param file Data file∗ bool OpenFiles (FILE ∗ *file*)

A function which will check if file is open.

2.2.1.7 void pastePuzzle (int *puzzle[SizeOfPuzzle][SizeOfPuzzle],* int *copy[SizeOfPuzzle][SizeOfPuzzle]*)

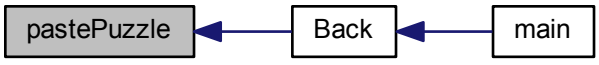
An another function to BackUp the puzzle.

Parameters

puzzle	The respective puzzle
copyOfPuzzle	

Assign the backup values to puzzle.

Here is the caller graph for this function:



2.2.1.8 void printPuzzle (int *puzzle[SizeOfPuzzle][SizeOfPuzzle],* int *SizeOfPuzzle*)

Function which will print puzzle.

Parameters

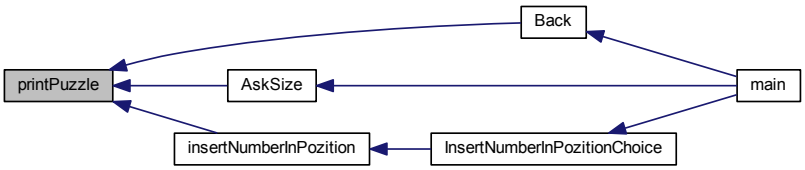
puzzle	The respective puzzle
SizeOfPuzzle	Puzzle Size

Get length of box.

Generate print interface.

Here is the call graph for this function:

Here is the caller graph for this function:



2.2.1.9 void printPuzzleInFile (FILE * *filename,* int *puzzle[SizeOfPuzzle][SizeOfPuzzle],* int *SizeOfPuzzle*)

Function which will print puzzle in a file.

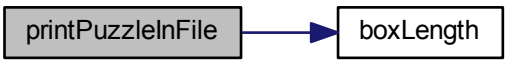
Parameters

filename	Data file
puzzle	The respective puzzle
SizeOfPuzzle	Puzzle Size

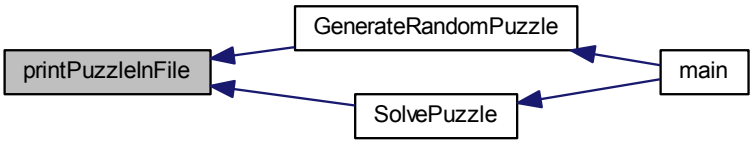
Get length of box.

Generate print interface.

Here is the call graph for this function:



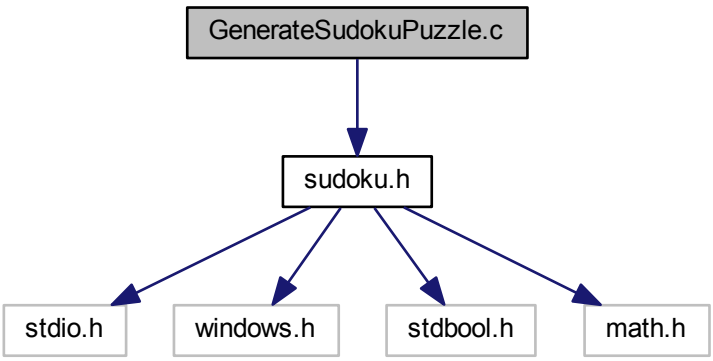
Here is the caller graph for this function:



2.3 GenerateSudokuPuzzle.c File Reference

```
#include "sudoku.h"
```

Include dependency graph for GenerateSudokuPuzzle.c:



Functions

- bool **existsRow** (int puzzle[**SizeOfPuzzle**][**SizeOfPuzzle**], int row, int value)

  There is Generation algorithm of our Sudoku puzzle, it is a simple generation using backtracking method !
- bool **existsCol** (int puzzle[**SizeOfPuzzle**][**SizeOfPuzzle**], int col, int value)

  Function which will check if the value exist on actual Column .
- bool **existsBox** (int puzzle[**SizeOfPuzzle**][**SizeOfPuzzle**], int boxRow, int boxCol, int value)

  Function which will check if the value exist on actual box .
- bool **legalAssign** (int puzzle[**SizeOfPuzzle**][**SizeOfPuzzle**], int row, int col, int value)

  Give permission to insert value in primary matrix.
- void **randomGeneration** (int puzzle[**SizeOfPuzzle**][**SizeOfPuzzle**])

  There is Generation algorithm of our Sudoku puzzle, it is a simple generation using backtracking method !

2.3.1 Function Documentation

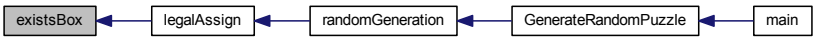
2.3.1.1 bool existsBox (int *puzzle[SizeOfPuzzle][SizeOfPuzzle],* int *boxRow,* int *boxCol,* int *value*)

Function which will check if the value exist on actual box .

Parameters

puzzle	The respective puzzle
BoxRow	
BoxCol	
Value	The value of number

Here is the caller graph for this function:



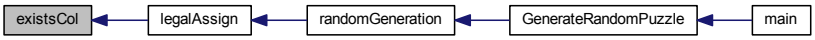
2.3.1.2 bool existsCol (int *puzzle[SizeOfPuzzle][SizeOfPuzzle],* int *col,* int *value*)

Function which will check if the value exist on actual Column .

Parameters

puzzle	The respective puzzle
Column	Respective Column
Value	The value of number

Here is the caller graph for this function:



2.3.1.3 bool existsRow (int *puzzle[SizeOfPuzzle][SizeOfPuzzle],* int *row,* int *value*)

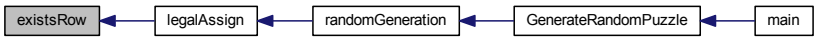
There is Generation algorithm of our Sudoku puzzle, it is a simple generation using backtracking method !

Function which will check if the value exist on actual Row .

Parameters

puzzle	The respective puzzle
Row	Respective Row
Value	The value of number

Here is the caller graph for this function:



2.3.1.4 bool legalAssign (int *puzzle[SizeOfPuzzle][SizeOfPuzzle],* int *row,* int *col,* int *value*)

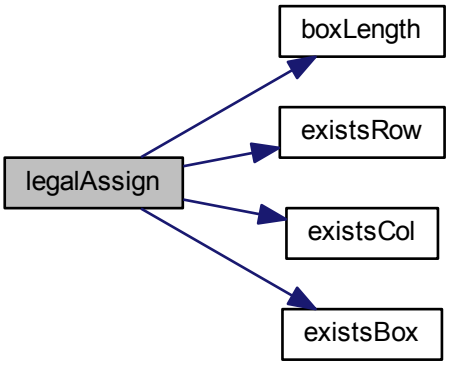
Give permission to insert value in primary matrix.

Parameters

puzzle	The respective puzzle
Row	Respective Row
Value	The value of number
Value	The value of number

Get size of box

Here is the call graph for this function:



Here is the caller graph for this function:

2.3.1.5 void randomGeneration (int *puzzle[SizeOfPuzzle][SizeOfPuzzle]*)

There is Generation algorithm of our Sudoku puzzle, it is a simple generation using backtracking method !

Parameters

puzzle	The respective puzzle

Random seed by clock.

Generate random numbers on rows !

Generate random numbers on columns !

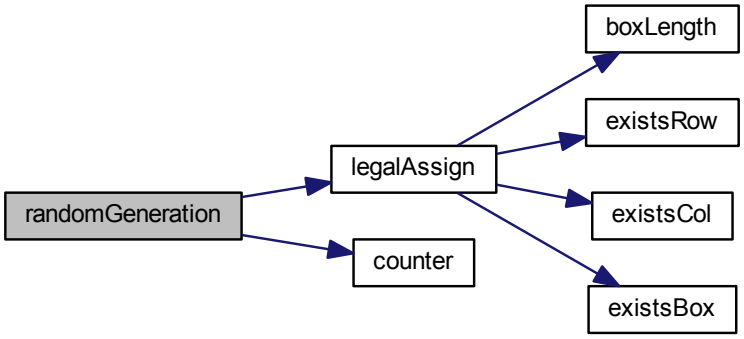
Generate a value which will fill matrix !

Get rights to assign value

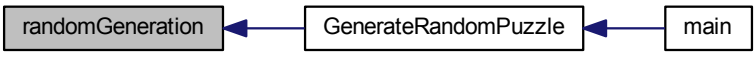
Assign the value

Declare a variable which will store how much numbers are avaiable on matrix.

Here is the call graph for this function:

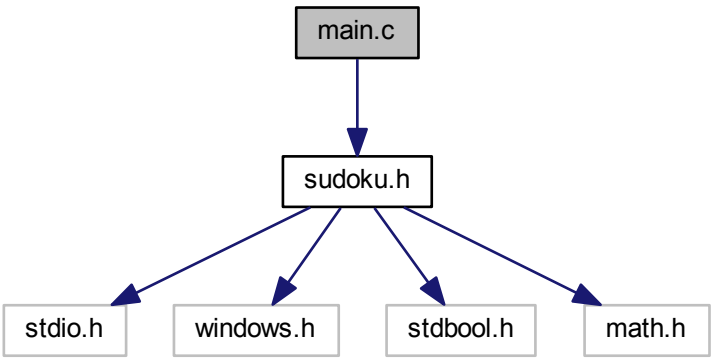


Here is the caller graph for this function:



2.4 main.c File Reference

`#include "sudoku.h"`
Include dependency graph for main.c:



Functions

- int **main** ()

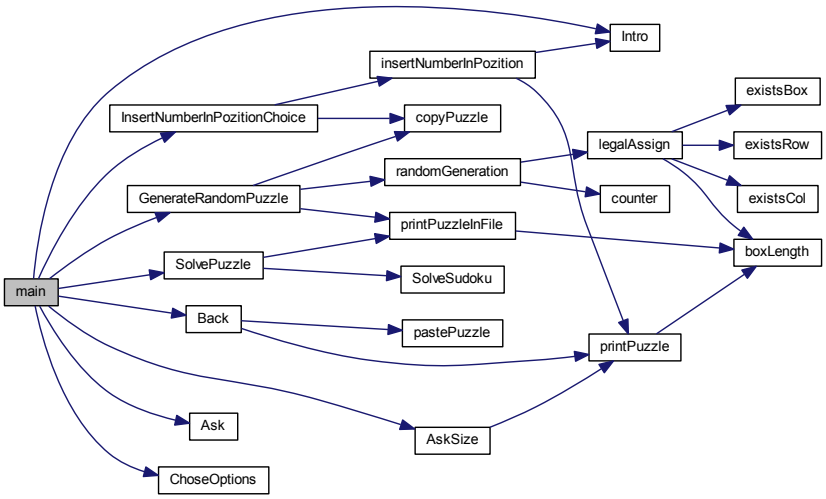
2.4.1 Function Documentation

2.4.1.1 int main ( )

Create Interface of Program

Here is generated the puzzle full with 0

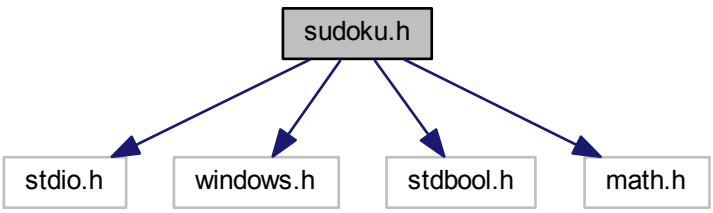
Here is the call graph for this function:



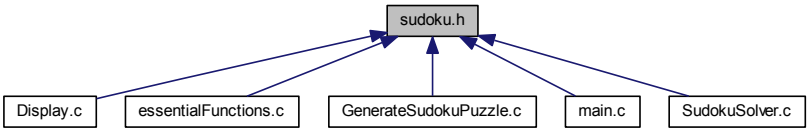
2.5 sudoku.h File Reference

```
#include <stdio.h>
#include <windows.h>
#include <stdbool.h>
#include <math.h>
```

Include dependency graph for sudoku.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define **UNASSIGNED** 0

Variables

- int **SizeOfPuzzle**
- bool **GameOver**
- bool **Start**
- int **choice**
- int **numberOfSolutions**
- int **solution**
- int **iterations**

2.5.1 Macro Definition Documentation

2.5.1.1 #define UNASSIGNED 0

2.5.2 Variable Documentation

2.5.2.1 int choice

Interface status

2.5.2.2 bool GameOver

Size of puzzle

2.5.2.3 int iterations

2.5.2.4 int numberOfSolutions

Choice status for switch statements

2.5.2.5 int SizeOfPuzzle

2.5.2.6 int solution

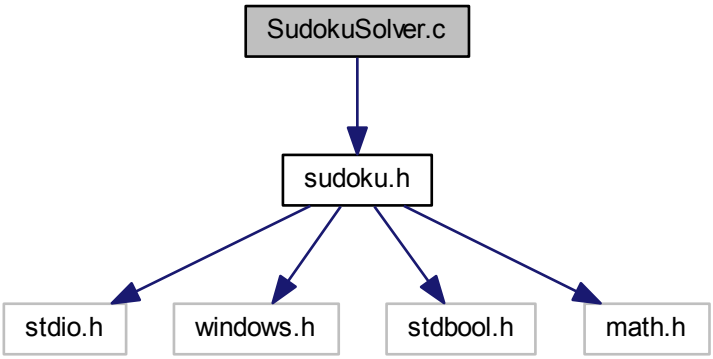
2.5.2.7 bool Start

Game status

2.6 SudokuSolver.c File Reference

```
#include "sudoku.h"
```

Include dependency graph for SudokuSolver.c:



Functions

- int **FindUnassigned** (int puzzle[**SizeOfPuzzle**][**SizeOfPuzzle**], int ∗row, int ∗col)

  Starting counter of solution.
- int **isSafe** (int puzzle[**SizeOfPuzzle**][**SizeOfPuzzle**], int row, int col, int num)

  Returns a boolean which indicates whether it will be legal to assign num to the given row,col location.
- int **UsedInRow** (int puzzle[**SizeOfPuzzle**][**SizeOfPuzzle**], int row, int num)

  Returns a boolean which indicates whether any assigned entry in the specified row matches the given number.
- int **UsedInCol** (int puzzle[**SizeOfPuzzle**][**SizeOfPuzzle**], int col, int num)

  Returns a boolean which indicates whether any assigned entry in the specified column matches the given number.
- int **UsedInBox** (int puzzle[**SizeOfPuzzle**][**SizeOfPuzzle**], int BoxStartRow, int BoxStartCol, int num)

  Returns a boolean which indicates whether any assigned entry within the specified box matches the given number.
- int **SolveSudoku** (int puzzle[**SizeOfPuzzle**][**SizeOfPuzzle**], FILE ∗fileName)

  Takes a partially filled-in grid and attempts to assign values to all unassigned locations in such a way to meet the requirements for Sudoku solution (non-duplication across rows, columns, and boxes)

Variables

- **solution** = 0

2.6.1 Function Documentation

2.6.1.1 int FindUnassigned (int *puzzle[SizeOfPuzzle][SizeOfPuzzle],* int * *row,* int * *col*)

Starting counter of solution.

Searches the puzzle to find an entry that is still unassigned. If found, the reference parameters row, col will be set the location that is unassigned, and true is returned. If no unassigned entries remain, false is returned.

Parameters

puzzle	The respective puzzle
Row	
Column	

2.6.1.2 int isSafe (int *puzzle[SizeOfPuzzle][SizeOfPuzzle],* int *row,* int *col,* int *num*)

Returns a boolean which indicates whether it will be legal to assign num to the given row,col location.

Parameters

puzzle	The respective puzzle
Row	
Column	
Number	

Check if 'num' is not already placed in current row, current column and current box

2.6.1.3 int SolveSudoku (int *puzzle[SizeOfPuzzle][SizeOfPuzzle],* FILE * *fileName*)

Takes a partially filled-in grid and attempts to assign values to all unassigned locations in such a way to meet the requirements for Sudoku solution (non-duplication across rows, columns, and boxes)

Parameters

puzzle	The respective puzzle
file	Data file

If there is no unassigned location, we are done

Starting counter of iteraions

Consider digits 1 to SizeOfPuzzle

If looks promising

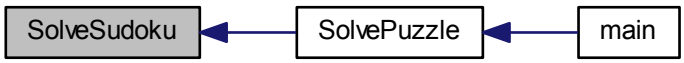
Make tentative assignment

Return, if success

Failure, unmake & try again

This triggers backtracking

Here is the caller graph for this function:



2.6.1.4 int UsedInBox (int *puzzle[SizeOfPuzzle][SizeOfPuzzle],* int *BoxStartRow,* int *BoxStartCol,* int *num*)

Returns a boolean which indicates whether any assigned entry within the specified box matches the given number.

Parameters

puzzle	The respective puzzle
BoxStartRow	
BoxStartCol	
Number	

2.6.1.5 int UsedInCol (int *puzzle[SizeOfPuzzle][SizeOfPuzzle],* int *col,* int *num*)

Returns a boolean which indicates whether any assigned entry in the specified column matches the given number.

Parameters

puzzle	The respective puzzle
Column	
Number	

2.6.1.6 int UsedInRow (int *puzzle[SizeOfPuzzle][SizeOfPuzzle],* int *row,* int *num*)

Returns a boolean which indicates whether any assigned entry in the specified row matches the given number.

Parameters

puzzle	The respective puzzle
Row	
Number	

2.6.2 Variable Documentation

2.6.2.1 solution = 0

Index