Evan Chen

CIS15BG

Professor Lee-Klawender

Programming HW#3 Design

Program Name: SimpleSudoku.c

**Step 1: Analyze the Problem**

**Output**: Allow user to play the Sudoku puzzle

**Input**: File containing the Sudoku puzzle

**Step 2: Design (Structure Chart, Pseudo code, Algorithmic pattern, Modular Specs)**

**Step 3: Main Pseudocode:**

CALL openFile(prompt,mode)

WHILE(openFile does not equal NULL)

CALL assignArray(sudokuArray)

CALL readArray(sudokufile, sudokuArray)

CALL playPuzzle(sudokuArray)

CALL openFile(prompt,mode)

END main (when WHILE loop is exited)

**Step 4: Logic of program**: The program will first read an incomplete Sudoku puzzle from a file that will be used for reference so that a user can play the game. It will then create a Sudoku grid on the screen with the numbers inserted into cells according to the file that was read in. The user will then be prompted to insert numbers into certain cells in hopes of solving the puzzle. The program will check and make sure that there is no duplicate number in the respective rows and columns of the number the user inserted. If there is none, the game will continue to ask the user for a number and check for duplications until the Sudoku puzzle is filled out.

**Step 5: Module Specs**

**1. openFile**

Return Value: FILE\*

Receives: prompt(string), mode(string)

Reference Parameters: prompt(string), mode(string)

Preconditions: None

Logic: Prompts user to provide the file’s name. The function will then read the file name into a local string. If the name of the file is QUIT, then the function returns NULL and the program quits. Otherwise, the function will open this file with the received by the mode(string) and return the file as a FILE\*

**2. assignArray**

Return Value: void

Receives: 2-dim array

Reference Parameters: 2-dim array

Preconditions: 2-dim array has been initialized

Logic: Using nested for loops and a pre-declared #define of the max dimensions, this function will assign -1 to each element in the 2-dim array.

**3. readArray**

Return Value: size(int)

Receives: file, 2-dim array

Reference Parameters: file, 2-dim array

Preconditions: File must be opened, 2-dim array has been initialized

Logic: The function obtains the size of the first line in an input file and then uses the size-1 to read the file into a 2-dim array. The size is then returned.

**4. playPuzzle**

Return Value: void

Receives: 2-dim array

Reference Parameters: 2-dim array

Preconditions: 2-dim array initialized

Logic: Using a while loop, display the 2-dim array in a table-format with column numbers and row letters but calling another function to print blank spaces if the element is 0. Then call another function that will allow the user to place numbers into the 2-dim array (play the game). If it returns true, call another function to check if the puzzle is solved. If not, continue the while loop.

**4a. elemZero**

Return Value: void

Receives: 2-dim array, index

Reference Parameters: 2-dim array

Preconditions: 2-dim array has been initialized and index has been declared

Logic: This function will be called if an element in the 2-dim array was 0. The function will use the index and print a blank space in the cell that the user sees.

**5. userNum**

Return Value: true or false(bool)

Receives: 2-dim array

Reference Parameters: 2-dim array

Preconditions: 2-dim array must be initialized

Logic: Prompts user if they want to place a number into the puzzle. While the user does not put ‘n’ or ‘N’, prompt user and read a row letter, column number, and value to place into the puzzle.

This function then calls another function to check if the selected cell is valid to place a number in.

If the other function returns 0, then error message is printed or else the value is assigned to the user’s puzzle coordinates and return true. If not, read the user’s answer again and repeat this function over.

**5a. checkSolved**

Return Value: true or false(bool)

Receives: 2-dim array

Reference Parameters: 2-dim array

Preconditions: 2-dim array must be initialized

Logic: Checks if the puzzle is solved. If it is, display a success message and display the solved 2-dim array. Return true if solved, false if not solved.

**6. validUsInput**

Return Value: int

Receives: char, int, int, 2-dim array

Reference Parameters: 2-dim array

Preconditions: char, int, int declared, 2-dim array initialized

Logic: This function checks if the row letter and column number entered by the user are valid in the dimensions of the current puzzle square. It also checks if the number is already in the same row or column or not. If it is not, then return true. Invalid row is return -1, invalid column is return -2, number is already in row or column then return -3.