Sudoku

PROJECT 1

Cindy Guijosa, CSC-5, Section 46090

Contents

Introduction

In an attempt to recreate the game Sudoku, I implemented the learning constructs within the requested chapters but also had to include a two-dimensional array that a 9x9 grid calls for. I began by hand drawing a set of numbers where each row, column, and square contained the numbers 1-9, but where no number was repeated in each row column and square. In order to randomize it, I generated two random numbers to represent two columns 1-3 and swap the two. I repeated this process for every columns and also did this for rows. This process will result in a solution grid but to output a grid with only a few showing, I had to create a duplicate array. To output the dashes, I set the whole grid except for a certain number of spots (number of spots varies depending on the level of difficulty the user selects) equal to dashes. Using a grid system the user inputs the position on the grid they wish to input the number, followed by the number itself. By making that position in the grid equal to a number it is no longer equal to a dash, which will then modify the output and display the grid with that number as well. This process of input and output continues until there are no longer any dashes left on the board. At this point, the game is over and the program asks if the user would like to play again.

Game Play and Rules

Sudoku begins with some of the grid cells already filled with numbers. The object is to fill the other empty spots with numbers between 1 through 9.

1. The number can appear only once in each row.

```
   1 2 3 4 5 6 7 8 9
  ==================
1|  - - - - - 5 - - -
2|  - - - 1 - - - 8 -
3|  - - - - 9 8 - - -
4|  - - 1 - 8 - 3 - -
5|  - - - - - - - - -
6|  - - - - 5 - - - 2
7|  - 1 9 5 - - - - -
8|  - - - - 1 - - - -
9|  - - - - 4 - 8 9 -
```

2. The number can appear only once in each column.

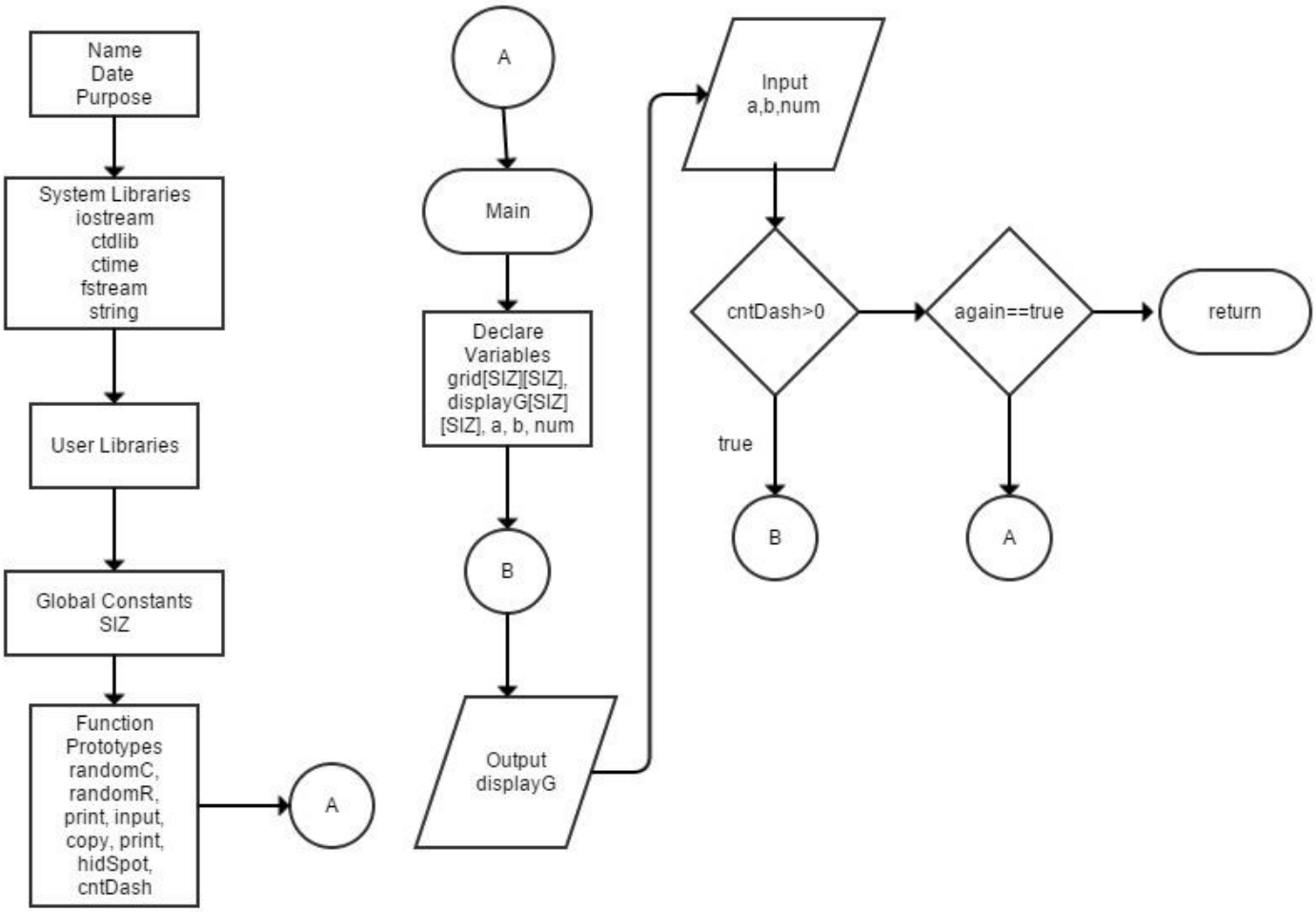
```
   1 2 3 4 5 6 7 8 9
  ===================
1| - - - - - 5 - - -
2| - - - 1 - - - 8 -
3| - - - - 9 8 - - -
4| - - 1 - 8 - 3 - -
5| - - - - - - - - -
6| - - - - 5 - - - 2
7| - 1 9 5 - - - - -
8| - - - - 1 - - - -
9| - - - - 4 - 8 9 -
```

3. The number can appear only once in each 3x3 regions.

```
   1 2 3 4 5 6 7 8 9
  ===================
1| - - - - - 5 - - -
2| - - - 1 - - - 8 -
3| - - - - 9 8 - - -
4| - - 1 - 8 - 3 - -
5| - - - - - - - - -
6| - - - - 5 - - - 2
7| - 1 9 5 - - - - -
8| - - - - 1 - - - -
9| - - - - 4 - 8 9 -
```

Flowchart

This is a flowchart that should give you a general idea of what the game will be like.



Variables

Type	Variable Name	Description	Location
bool	again	To repeat game after finished if it is true	main
char	displayG[][]	Array that is displayed	main
	a,b,num	To input row, column, and number for grid	Main and input()
	level	User inputs what level(a, b, or c) they want to play game which controls how many numbers initially output	hidSot()
	x,y	To randomize spots not empty	hidSpot()
	temp,tempo,tempT	To randomize gride	randomR and randomC
Unsigned short	grid[][]	Solution grid	main
	cnt	To count how many dashes are left and know when to end game	cntDash()
int	i,j	To form and edit arrays	cntDash(), hidSpot(), print(), copy()
string	rules	To input from file	main

Major Constructs

Construct	Location
Infile/outfile	main
cin/cout	Main, print(), input(), hidSpot()
Global constant	Global constants
Unsigned variable	main
Static casting	copy()
If	hidSpot(), cntDash()
If else	Main, input(), print()
while	Main, hidSpot()
do while	main
switch	hidSpot()
For loop	randomC(), randomR()
Nested loop	cntDash(), print(), copy()
counter	cntDash()
Increment/decrement	cntDash(), print()
Pass by reference	Input()
Void	randomC(), randomR(), print(), input(), print(), copy(), hidSpot()
Return	cntDash()
overloading	Print(), print()

```
/*
 * File:   main.cpp
 * Author: Cindy Guijosa
 * Purpose: Project Soduko
 * Created on July 18, 2015, 9:29 PM
 */

//System Libraries
#include <iostream>
#include <cstdlib>
#include <ctime>//for random number generator
#include <fstream>
#include <string>
using namespace std;

//User Libraries

//Global Constants
const unsigned short SIZ=9;
//Function Prototypes
void randomC(unsigned short [][SIZ]);
void randomR(unsigned short [][SIZ]);
void print(unsigned short [][SIZ]);
void input(char &,char &,char &,unsigned short [][SIZ], char [][SIZ]);
void copy(unsigned short [][SIZ], char [][SIZ]);
void print(char [][SIZ],unsigned short [][SIZ]);
void hidSpot(char [][SIZ]);
unsigned short cntDash(char [][SIZ]);
//Execution Begins Here!
int main(int argc, char** argv) {
    //Plant random number seed
    srand(static_cast<unsigned int>(time(0)));
```

```
    //Declare variables
    unsigned short grid[SIZ][SIZ]={{1,2,3,7,8,9,4,5,6},
                                   {4,5,6,1,2,3,7,8,9},
                                   {7,8,9,4,5,6,1,2,3},
                                   {9,1,2,6,7,8,3,4,5},
                                   {3,4,5,9,1,2,6,7,8},
                                   {6,7,8,3,4,5,9,1,2},
                                   {8,9,1,5,6,7,2,3,4},
                                   {2,3,4,8,9,1,5,6,7},
                                   {5,6,7,2,3,4,8,9,1}};

    char displayG[SIZ][SIZ];
    char a,b,num;
    bool again;
    do{
        randomC(grid);
        randomR(grid);
        copy(grid,displayG);

        //to output rules from a file
        ifstream infile("rules.txt");
        string rules;
        while(!infile.eof()){
            infile>>rules;
            cout<<rules<<" ";
        }
        cout<<endl;

        cout<<"The object is to fill all empty spots so that the numbers 1 to 9 appear exactly once in each row,\n";
        cout<<"column and 3x3 box (ex:starting at row 1 column 1 and ending at row 3 column 3)."<<endl<<endl;
        hidSpot(displayG);
```

```
65          //Loop to repeat until there are no more dashes on grid
66
67          do
68          {
69              print(displayG,grid);
70              input(a,b,num,grid,displayG);
71
72          }while(cntDash(displayG)>0);
73
74          char redo;
75          cout<<endl;
76          cout<<"Congratulations, would you like to play again?(y for yes or n for no)"<<endl;
77          cin>>redo;
78          if(redo=='y' || redo=='Y'){
79              again=true;
80              cout<<endl;
81          }else{
82              again=false;
83              cout<<endl;
84          }
85      }while(again==true);
86
87      return 0;
88  }
89
90  /*************************************************************
91   * ************************randomC*****************************
92   * ************************************************************
93   * Purpose: To randomize the columns
94   * Input:
95   *         firstGN & secGN->swap these two random columns
96   *         firstG & secG->swap these two columns
```

```
96   *         firstG & secG->swap these two columns
97   *         first & sec->swap these two columns
98   * Output:
99   *         grid->array with swapped random columns
100  * ************************************************************/
101
102 void randomC(unsigned short grid[][SIZ]){
103     //Randomize columns
104
105     //first 3 columns
106     char firstGN=rand()%3;
107     char secGN=rand()%3;
108     //columns 4-6
109     char firstG=rand()%2+4;
110     char secG=rand()%2+4;
111     //columns 5-9
112     char first=rand()%3+6;
113     char sec=rand()%3+6;
114
115     //for loop to swapping columns
116     for(int i=0;i<SIZ;i++){
117
118         char temp;
119         temp=grid[i][firstGN];
120         grid[i][firstGN]=grid[i][secGN];
121         grid[i][secGN]=temp;
122
123         char tempO;
124         tempO=grid[i][firstG];
125         grid[i][firstG]=grid[i][secG];
126         grid[i][secG]=tempO;
127
```

```
            char tempT;
            tempT=grid[i][first];
            grid[i][first]=grid[i][sec];
            grid[i][sec]=tempT;
        }
    }

/*************************************************************
 * ************************randomR*****************************
 * ************************************************************
 * Purpose: To randomize the rows
 * Input:
 *         firstGN & secGN->swap these two random rows
 *         firstG & secG->swap these two rows
 *         first & sec->swap these two rows
 * Output:
 *         grid->array with swapped random rows
 * ************************************************************/

void randomR(unsigned short grid[][SIZ]){
    //Randomize Rows
    char firstGN=rand()%3;
    char secGN=rand()%3;
    char firstG=rand()%2+4;
    char secG=rand()%2+4;
    char first=rand()%3+6;
    char sec=rand()%3+6;

    //for loop to swapping rows
    for(int i=0;i<SIZ;i++){

            char temp;
            temp=grid[firstGN][i];
            grid[firstGN][i]=grid[secGN][i];
            grid[secGN][i]=temp;

            char tempO;
            tempO=grid[firstG][i];
            grid[firstG][i]=grid[secG][i];
            grid[secG][i]=tempO;

            char tempT;
            tempT=grid[first][i];
            grid[first][i]=grid[sec][i];
            grid[sec][i]=tempT;
        }
    }

/*************************************************************
 * ************************print*******************************
 * ************************************************************
 * Purpose: To print the solution grid (only if wanted)
 * Output:
 *         grid->array with swapped random columns
 * ************************************************************/

void print(unsigned short grid[][SIZ]){
    cout<<"    1 2 3 4 5 6 7 8 9"<<endl;
    cout<<"   ==================="<<endl;
    for(int i=0;i<SIZ;i++){
        cout<<i+1<<"|  ";
        for(int b=0;b<SIZ;b++){
```

```
190             cout<<grid[i][b]<<" ";
191           }
192           cout<<endl;
193         }
194         cout<<endl;
195     }
196
197 /*****************************************************************
198  * ************************input*********************************
199  * ****************************************************************
200  * Purpose: To prompt/input and take input if right modify display
201  *          and choose level
202  * Input:
203  *         a->row user input
204  *         b->column user input
205  *         num->number user input for answer
206  *         grid->to identify if input matches value in array
207  * Output:
208  *         displayG->modify value if answer is correct
209  * ****************************************************************/
210
211 void input(char &a,char &b,char &num,unsigned short grid[][SIZ],char displayG[][SIZ]){
212     cout<<"To select a coordinate enter number of row, number of column,"<<endl;
213     cout<<"and number you want to enter.(ex: 1(row),4(column),9)"<<endl;
214     cin>>a;
215     cin.ignore();
216     cin>>b;
217     cin.ignore();
218     cin>>num;
219
220     int x=a-49;
221     int y=b-49;
222     int c=num-48;
223     if(c==grid[x][y]){
224         displayG[x][y]='a';
225     }else{
226         cout<<"Thats wrong! Try again!"<<endl;
227     }
228
229 }
230
231 /*****************************************************************
232  * ************************copy**********************************
233  * ****************************************************************
234  * Purpose: To copy value from one array to another
235  * Input:
236  *         grid->array being copied
237  * Output:
238  *         displayG->new array
239  * ****************************************************************/
240
241 void copy(unsigned short grid[][SIZ], char displayG[][SIZ]){
242     for(int i=0;i<SIZ;i++){
243         for(int j=0; j<SIZ;j++){
244             displayG[i][j]=static_cast<char>(grid[i][j]);
245             //if(grid[i][j] != '-')
246                // ++cnt
247         }
248     }
249 }
250
251 /*****************************************************************
```

```
252  * ************************print**********************************
253  * ***************************************************************
254  * Purpose: To print grid with dashes where answer is hidden
255  * Output:
256  *         displayG->print dashes where numbers are not displayed
257  * ************************************************************/
258
259  void print(char displayG[][SIZ], unsigned short grid[][SIZ]){
260      cout<<"    1 2 3 4 5 6 7 8 9"<<endl;
261      cout<<"   ==================="<<endl;
262      for(int i=0;i<SIZ;i++){
263          cout<<i+1<<"|  ";
264          for(int j=0;j<SIZ;j++){
265              if(displayG[i][j]=='-'){
266                  cout<<displayG[i][j]<<" ";
267              }
268              else{
269                  cout<<grid[i][j]<<" ";
270              }
271          }
272          cout<<endl;
273      }
274  }
275
276  /***************************************************************
277  * ************************hidSpot*******************************
278  * ***************************************************************
279  * Purpose: To choose random spots to display number and how many
280  * Input:
281  *         level->to choose amount of spots
282  * Output:
283  *         grid->array with swapped random rows
284  * ************************************************************/
285
286  void hidSpot(char displayG[][SIZ]){
287      char level;
288      cout<<"Choose level of difficulty: a for easy, b for medium, and c for hard.\n
289      cin>>level;
290      switch(level){
291              case 'a':
292                  for(int i=0;i<(SIZ*SIZ)-21;i++){
293                      while(true){
294                          char x=rand()%SIZ; //8
295                          char y=rand()%SIZ; //8
296                          if(displayG[x][y]!='-')
297                          {
298                           displayG[x][y]='-';
299                           break;
300                          }
301                          cout<<displayG[x][y]<<" ";
302                      }
303                  }
304                  cout<<endl;
305                  break;
306              case'b':
307                  for(int i=0;i<(SIZ*SIZ)-19;i++){
308                      while(true){
309                          char x=rand()%SIZ; //8
310                          char y=rand()%SIZ; //8
311                          if(displayG[x][y]!='-')
312                          {
313                           displayG[x][y]='-';
314                           break;
```

```
                        }
                        cout<<displayG[x][y]<<" ";
                    }
                }
                cout<<endl;
                break;
            case 'c':
               for(int i=0;i<(SIZ*SIZ)-17;i++){
                    while(true){
                        char x=rand()%SIZ; //8
                        char y=rand()%SIZ; //8
                        if(displayG[x][y]!='-')
                        {
                         displayG[x][y]='-';
                         break;
                        }
                        cout<<displayG[x][y]<<" ";
                    }
                }
                cout<<endl;
                break;
        }

}

/*****************************************************************
 * ************************cntDash*******************************
 * ****************************************************************
 * Purpose: To count number of dashes in order to end game
 * Input:
 *         displayG: grid with dashes
 * Output:
 *         cnt->count dashes
 * ****************************************************************/

unsigned short cntDash(char displayG [][SIZ]){
    unsigned short cnt=0;
    for(int i=0;i<SIZ;i++){
        for(int j=0; j<SIZ;j++){
            if(displayG[i][j] == '-'){
                cnt++;


            }
        }
    }
    return cnt;
}
```