Sudoku Solver

Developed by:

Suyash Phatak (19070122125)

Sayan Sahu (19070122150)

Sharath B Pai (19070122155)

Spandan Pandey (19070122174)

Vanarote Pranhav Nagesh (19070122190)

Yash Verma (19070122202)

Abstract

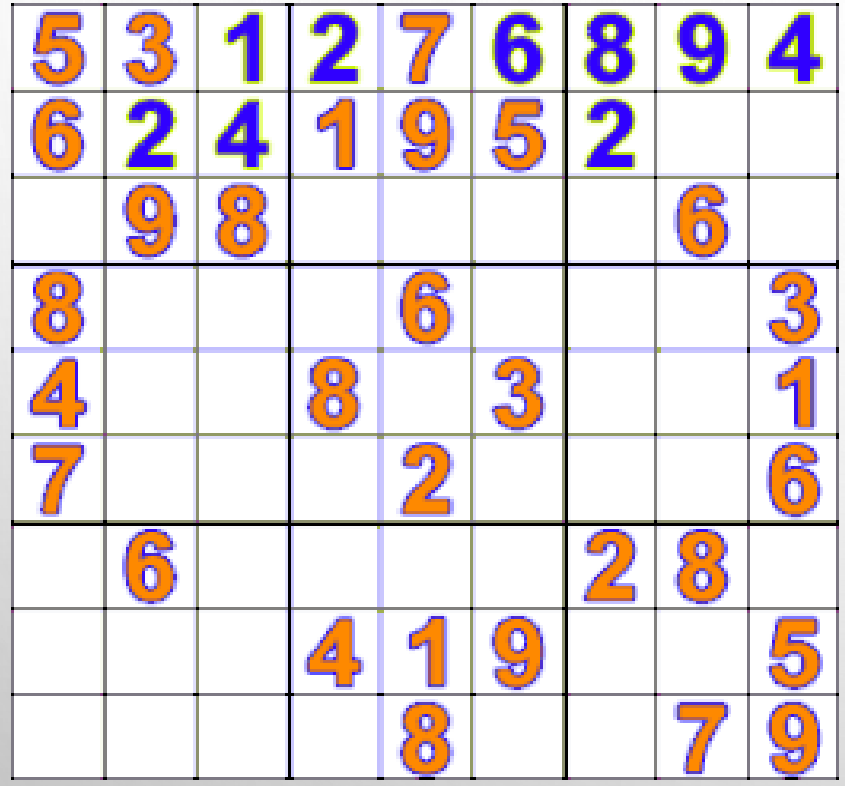
In the last decade, solving the sudoku puzzle has become every one's passion. The simplicity of puzzle's structure and the low requirement of mathematical skills caused people to have enormous interest in accepting challenges to solve the puzzle. The sudoku game consists of a graphical user interface and a sudoku solver; implemented using C, gtk+3 and glade GUI builder. The solver is implemented using the **backtracking algorithm**. Essentially, you keep trying numbers in empty spots until there aren't any that are possible, then you backtrack and try different numbers in the previous slots. The user can try to solve the sudoku or use the solver to find the solution of the puzzle. The program allows the user to choose a set of problems from the file system. This project gives an insight into the different aspects of C programming.

Introduction

3		6	5		8	4		
5	2							
	8	7					3	1
		3		1			8	
9			8	6	3			5
	5			9		6		
1	3					2	5	
							7	4
		5	2		6	3		

- A **sudoku** puzzle is defined as a logic-based, number-placement puzzle. The objective is to fill a 9×9 grid with digits in such a way that each column, each row, and each of the nine 3×3 grids that make up the larger 9×9 grid contains all of the digits from 1 to 9. Each sudoku puzzle begins with some cells filled in. The player uses these seed numbers as a launching point toward finding the unique solution.
- It is important to stress the fact that no number from 1 to 9 can be repeated in any row or column (although, they can be repeated along the diagonals).

Backtracking Algorithm



Like all other **backtracking problems**, we can solve sudoku by one by one assigning numbers to empty cells. Before assigning a number, we check whether it is safe to assign. We basically check that the same number is not present in the current row, current column and current 3X3 sub grid. After checking for safety, we assign the number, and recursively check whether this assignment leads to a solution or not. If the assignment doesn't lead to a solution, then we try the next number for the current empty cell. And if none of the number (1 to 9) leads to a solution, we return false.

Backtracking Algorithm

- FIND ROW, COL OF AN UNASSIGNED CELL
- IF THERE IS NONE, RETURN TRUE
- FOR DIGITS FROM 1 TO 9

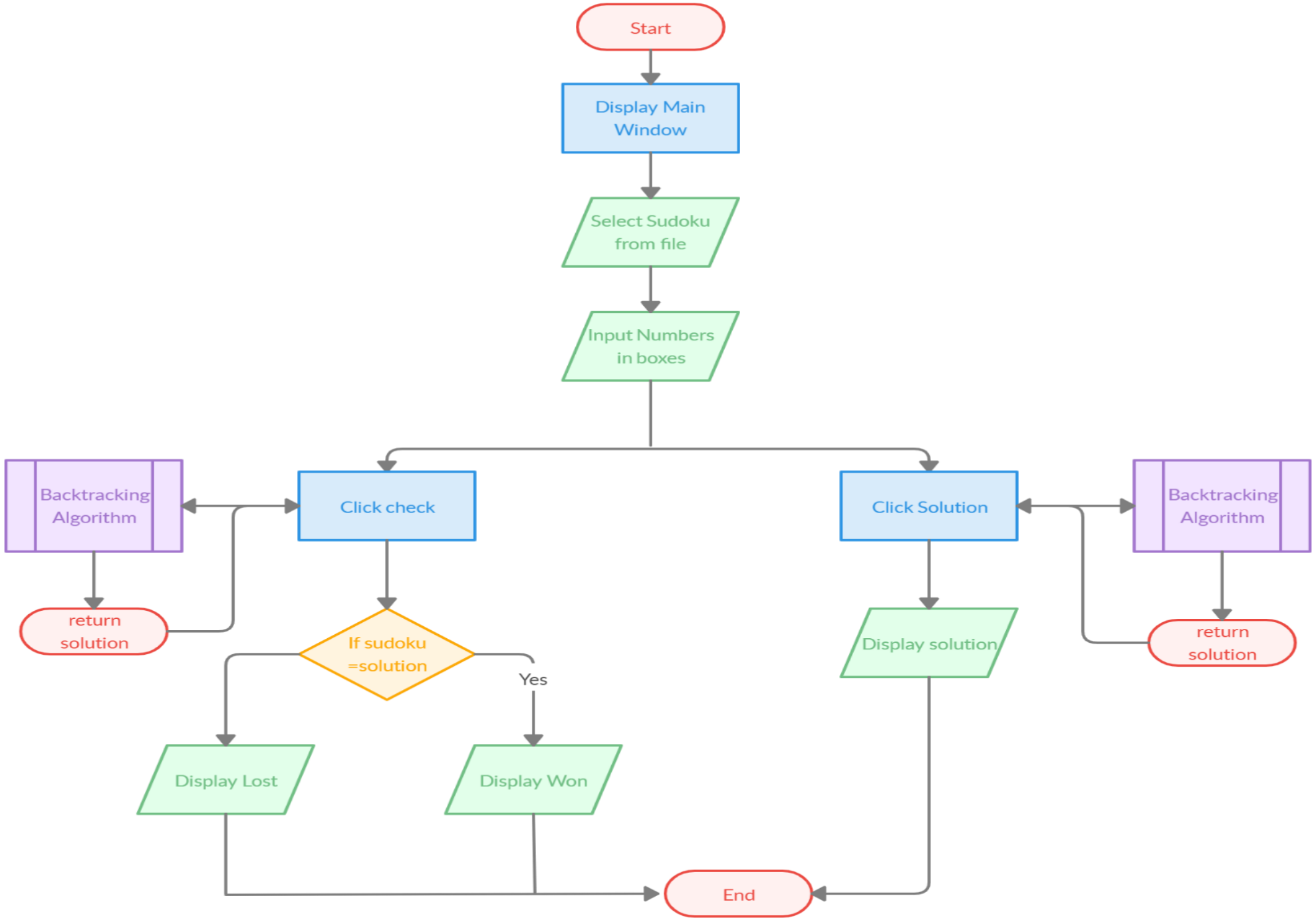
  A) IF THERE IS NO CONFLICT FOR DIGIT AT ROW, COL

  ASSIGN DIGIT TO ROW, COL AND RECURSIVELY TRY FILL IN REST OF GRID

  B) IF RECURSION SUCCESSFUL, RETURN TRUE

  C) ELSE, REMOVE DIGIT AND TRY ANOTHER
- IF ALL DIGITS HAVE BEEN TRIED AND NOTHING WORKED, RETURN FALSE

Flowchart


Start
Display Main Window
Select Sudoku from file
Input Numbers in boxes
Backtracking Algorithm
Click check
Click Solution
Backtracking Algorithm
return solution
If sudoku =solution
Yes
Display solution
return solution
Display Lost
Display Won
End

Requirements

Linux

Sr. No	Software	Name	Version
1	Operating System	Ubuntu	18.2
2	C Compiler	gcc	8.3.0
3	Editor	Geany	1.34.1
4	Rapid GUI Builder	Glade	3.22.1
5	GUI Library	GTK + 3	3.24

Windows

Sr. No	Software	Name	Version
1	Operating System	Windows	10
2	Cross-compiler Tool for generating Windows Exe	MinGW-w64	i686-w64-mingw32-gcc

Module Description

gtk/gtk.h	stdlib.h	stdio.h
GTK is a widget toolkit. Each user interface created by GTK consists of UI elements known as widgets. Each user interface is built by adding buttons, text labels, input fields, drop down menus, and other widgets to a window.	This header defines several general-purpose functions, including dynamic memory management, random number generation, communication with the environment, integer arithmetic, searching, sorting and converting.	The stdio.h header defines three variable types, several macros, and various functions for performing input and output in the console and files.

List of Functions with Description

S. No	Function Name	Return Type	Description
1	**Print_sudoku**	Void	This function is invoked to display sudoku solution when valid sudoku solution is available
2	**Number_unassigned**	Int	It checks whether all cells are assigned or not.
3	**Is_safe**	Int	This is used to check if we can put a value in a cell or not. Also validates that other cell in same row or column doesn't hold same value.
4	**Solve_sudoku**	Int	Invoked when solve sudoku button is pressed to get the solution of the problem.
5	**Get_elements**	Void	This function is called when sudoku solution button clicked to show the solution. Internally is calls print_sudoku function for displaying a valid solution.
6	**Check_elements**	void	Click event of Check button calls this function to check and validate the sudoku matching done by user.
7	**On_clear_menu_item_activate**	Void	This function flushes the screen.
8	**On_open_menu_item_activate**	Void	This is invoked when File → Open menu option is selected to open file chooser dialog box for selecting sudoku file.
9	**On_quit_menu_item_activate**	Void	When File → Quit menu option is selected, this function is called to close the main window.
10	**On_menuitm_about_activate**	Void	This function is used to open the about dialog box when Help → About menu option is selected.
11	**On_dlg_about_response**	Void	When About dialog box Close button is clicked, this function is invoked to close the About dialog box.
12	**On_quit_activate**	Void	Invoked when close(X) button is clicked to close the main window.
13	**On_window0_destroy**	Void	Internally called by On_quit_activate function to close the main window.
14	**Main**	Void	Main sudoku program body.

Code Snapshot

sudoku_main.c backupgrid.c makefile ss1.c

```c
/* ******************************************************************************************
 * Sudoku solver application                                                                *
 *                                                                                          *
 * Created on: 22-March-2020                                                                *
 *                                                                                          *
 * Developed by : Suyash Phatak ,Sayan Sahu, Sharath B Pai, Spandan Pandey, Pranhav Vanarote, Yash Verma  *
 *                                                                                          *
 * ******************************************************************************************/

#include <gtk/gtk.h>
#include <stdio.h>
#include <stdlib.h>


#define SIZE 9

static GtkWidget *window,*wid[9][9], *hbox, *vbox, *sudoku_lbl, *gtkbox;
static GtkBuilder *builder ;
static int sol = 0;

// Pointer to about dialog box
typedef struct {

    GtkWidget *w_dlg_about;
} app_widgets;

// Pointer to about dialog box

int a[9][9], b[9][9], check[9][9];

//function to print sudoku
void print_sudoku()
{
    for(int i = 0; i<9; ++i){
      for(int j = 0; j<9; ++j){
            gtk_entry_set_alignment (GTK_ENTRY(wid[i][j]), 0.5);
            if(a[i][j] != 0){
                char c[2];
                sprintf(c,"%d",a[i][j]);
                gtk_entry_set_text(GTK_ENTRY(wid[i][j]),c);
                gtk_editable_set_editable(GTK_EDITABLE(wid[i][j]), FALSE);
                }
        }
```

*sudoku_main.c - /home/lenovo/cproject/sudoku_project/src - Geany

File Edit Search View Document Project Build Tools Help

Symbols

Functions
check_elements
get_elements [1
is_safe [77]
main [284]
number_unassig
on_clear_menu_
on_dlg_about_re
on_menuitm_ab
on_open_menu_
on_quit_activate
on_quit_menu_it
on_window0_de
print_sudoku [3]
solve_sudoku [1

Structs
anon_struct_0 [
w_dlg_about

Typedefs / Enums
app_widgets [25

Macros
SIZE [15]

Variables
a [29]
b [29]
builder [18]
check [29]
gtkbox [17]
hbox [17]
sol [19]
sudoku_lbl [17]
vbox [17]
wid [17]
window [17]

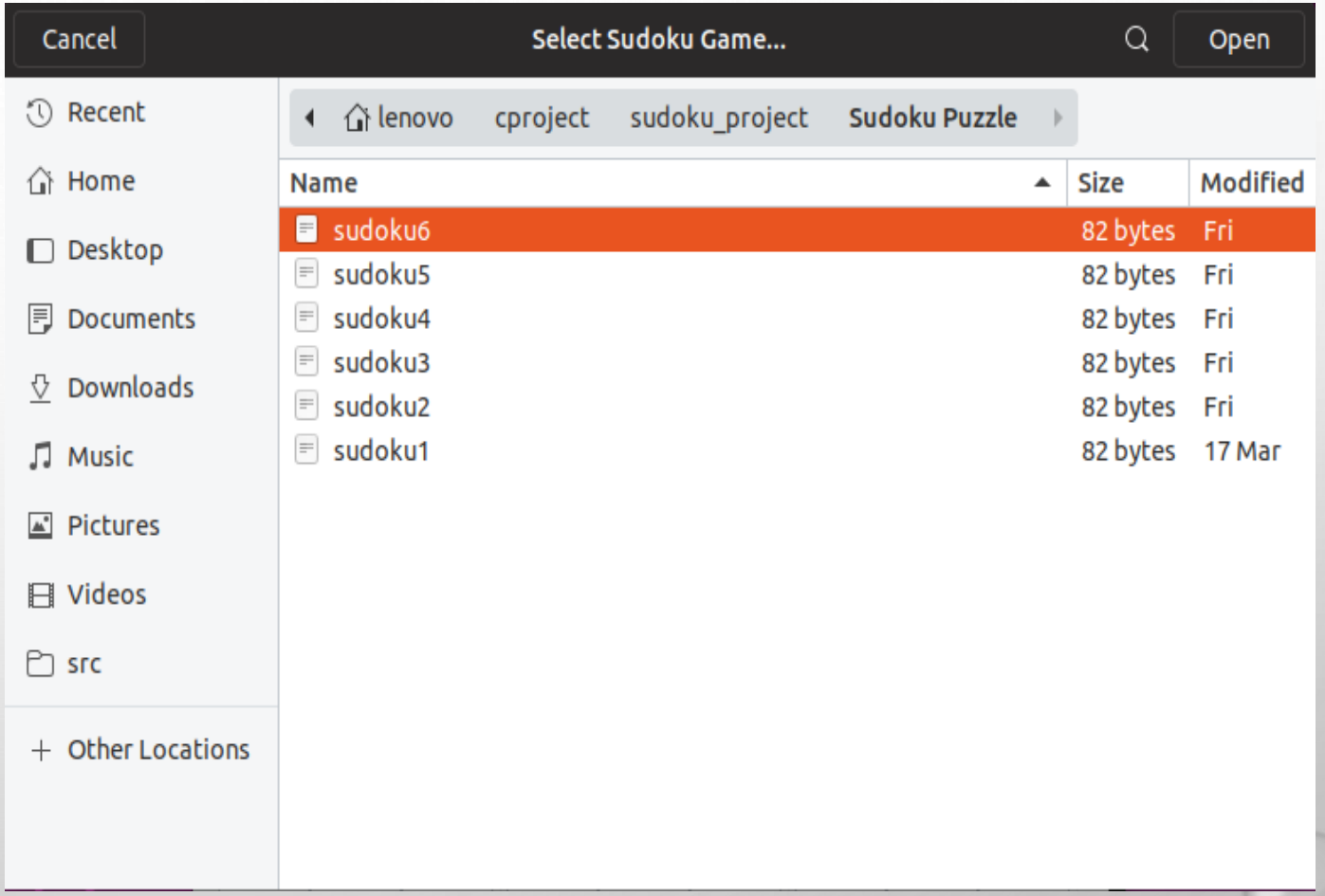
sudoku_main.c

```c
int main(int argc, char *argv[]) {

  gtk_init(&argc, &argv);
// Pointer to button, button box
  GtkWidget  *button1, *separator, *button2, *buttonbox;

  builder = gtk_builder_new();

//Getting the glade file
  gtk_builder_add_from_file (builder, "/home/lenovo/cproject/sudoku_project/glade/sudoku_main.glade", NULL);

  window = GTK_WIDGET(gtk_builder_get_object(builder, "window0"));
  gtkbox = GTK_WIDGET(gtk_builder_get_object(builder, "gtkbox0"));
  buttonbox = gtk_box_new(GTK_ORIENTATION_HORIZONTAL, 0);
  vbox=gtk_box_new(GTK_ORIENTATION_VERTICAL, 0);
  gtk_window_set_title(GTK_WINDOW(window), "Sudoku Solver");
  app_widgets *widgets = g_slice_new(app_widgets);

// Get pointers to widgets
  widgets->w_dlg_about = GTK_WIDGET(gtk_builder_get_object(builder, "dlg_about"));

  gtk_builder_connect_signals(builder, widgets);
  sudoku_lbl = gtk_label_new("***  Sudoku Puzzle  *** ");
  gtk_box_pack_start(GTK_BOX(vbox),sudoku_lbl,TRUE,TRUE,10);

// Building the sudoku grid
  for(int i=0;i<9;i++)
        {
                // Generating s horizontal box
                hbox=gtk_box_new(GTK_ORIENTATION_HORIZONTAL, 0);
                for(int j=0;j<9;j++)
                {
                        wid[i][j]=gtk_entry_new();
                        gtk_entry_set_width_chars(GTK_ENTRY(wid[i][j]),5);
                        gtk_entry_set_max_length(GTK_ENTRY(wid[i][j]),1);
                        gtk_widget_set_size_request(wid[i][j],50,50);
                        gtk_box_pack_start(GTK_BOX(hbox),wid[i][j],0,0,0);

                        // Adding a vertical sepearator after every 3 boxes
                        if(((j+1)%3)==0)
                        {
                                separator = gtk_separator_new(GTK_ORIENTATION_VERTICAL);
                                gtk_box_pack_start(GTK_BOX(hbox),separator,TRUE,TRUE,4);
                        }

                }
                gtk_box_pack_start(GTK_BOX(vbox),hbox,0,0,0);

                // Adding a horizontal sepearator after every 3 boxes
                 if(((i+1))%3==0)
```

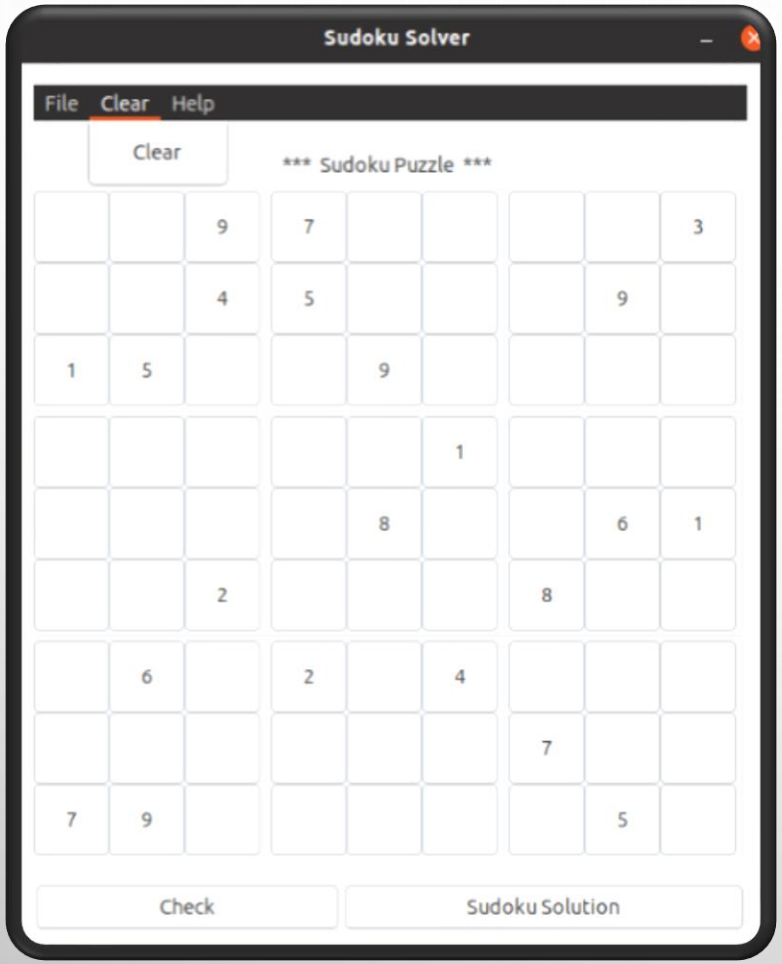
line: 305 / 385 col: 48 sel: 0 INS TAB MOD mode: LF encoding: UTF-8 filetype: C scope: main

Output Screenshots

File Menu

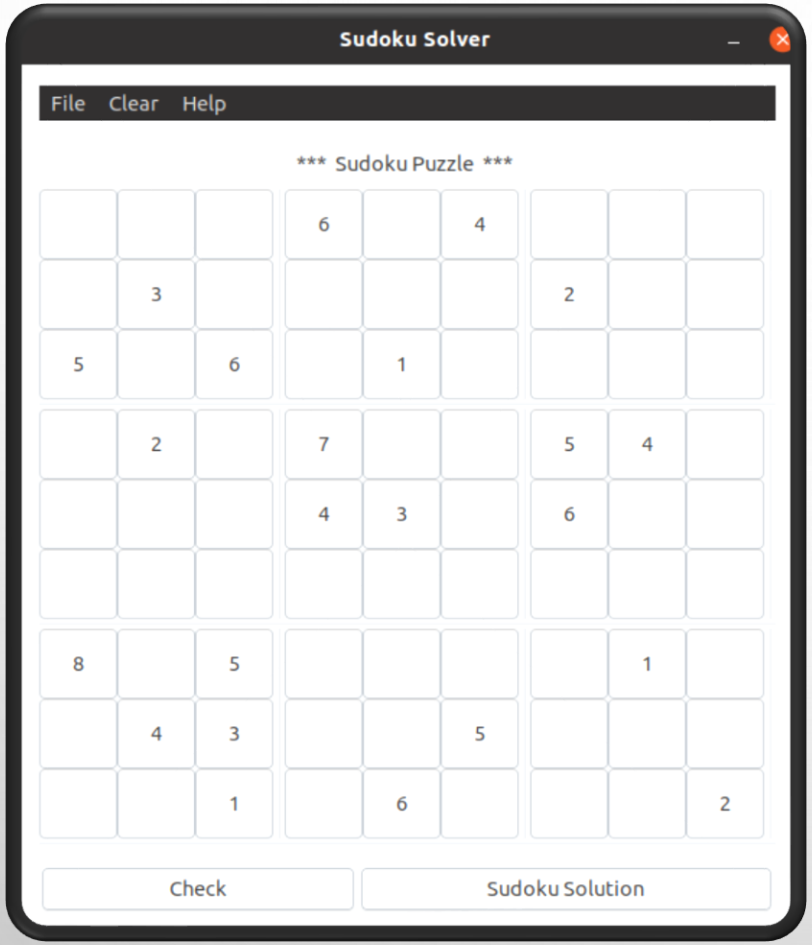


File Selector

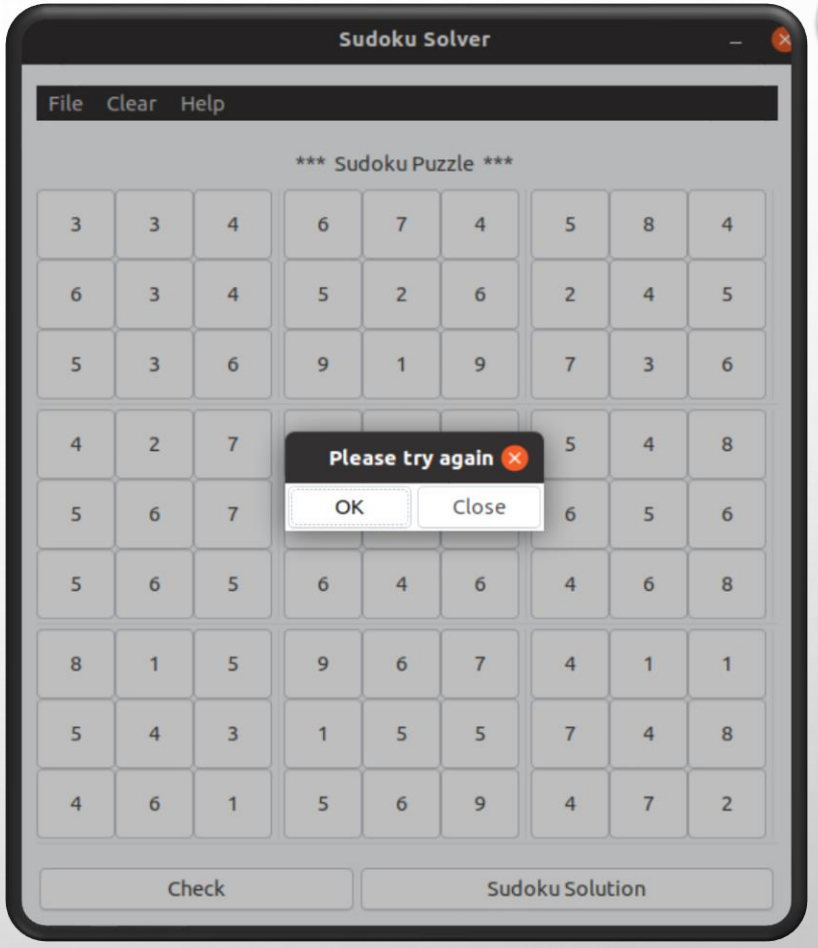


Clear Menu

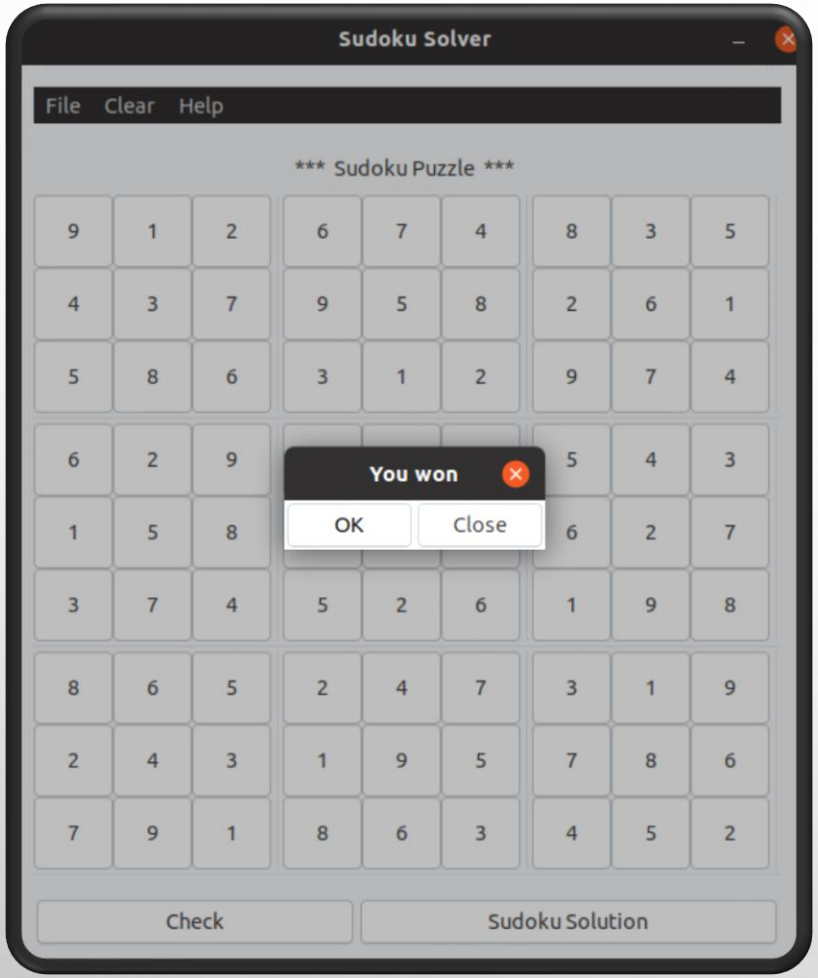
Help Menu



Starting Screen

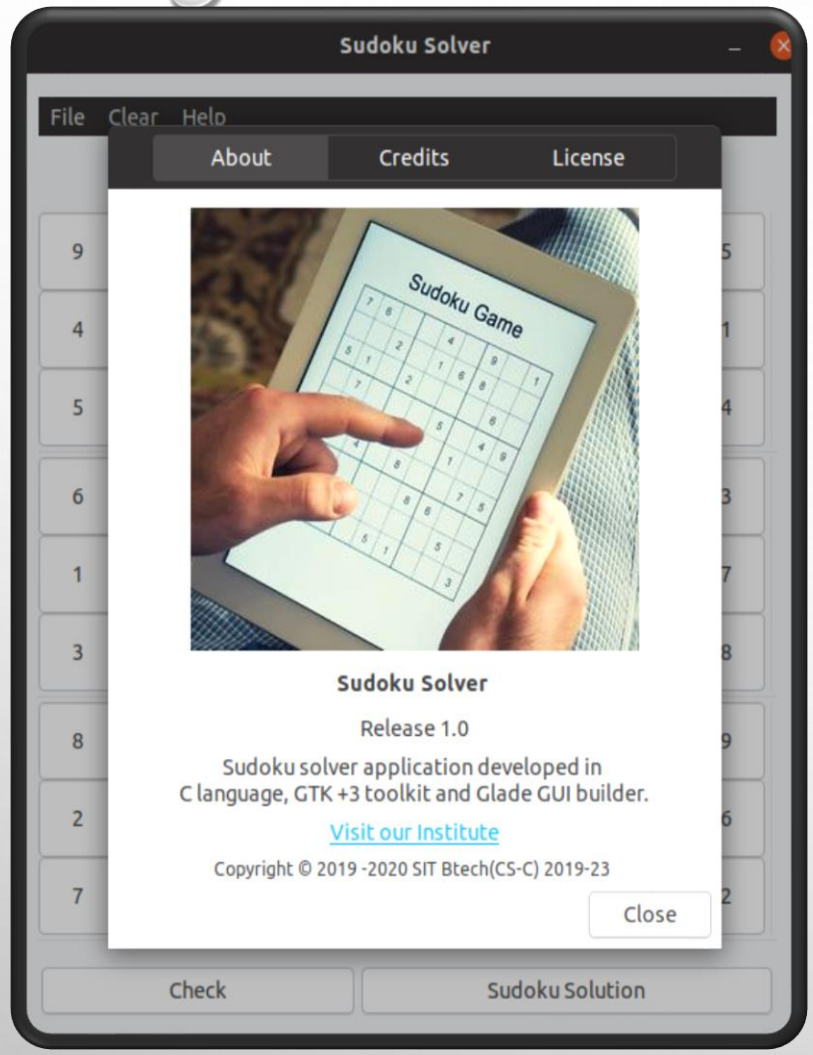


Game Over

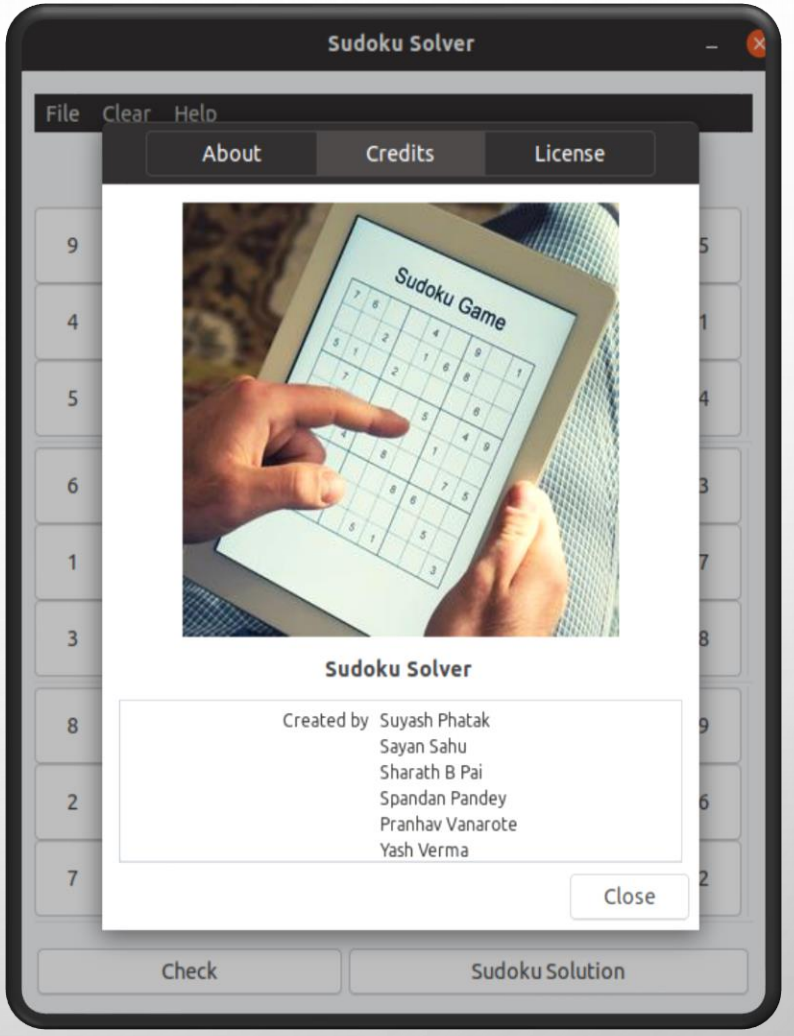


Won

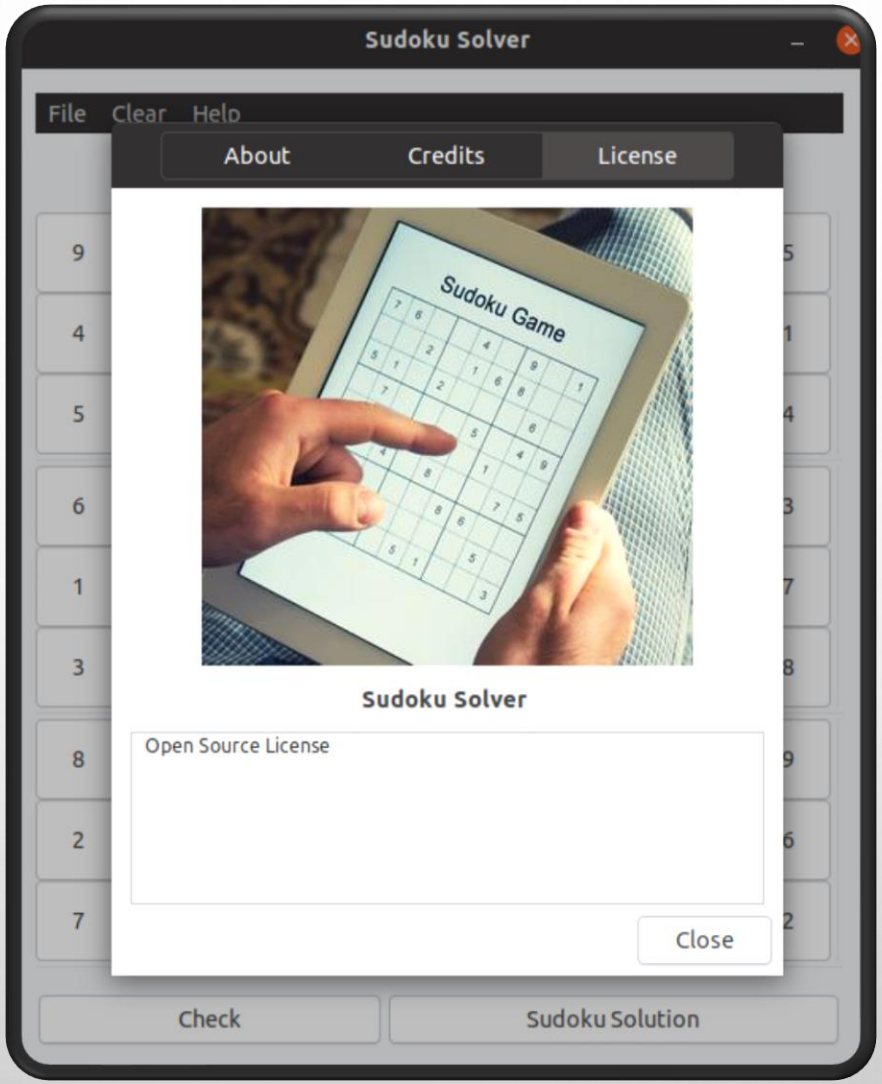
Sudoku Solution



About Screen



Credits


Sudoku Solver
File
Clear
Help
About
Credits
License
Sudoku Game
Sudoku Solver
Open Source License
Close
Check
Sudoku Solution

License

Conclusion

Through programming this sudoku game and solver, we have improved our programming skills. It was quite challenging to build this GUI software in C as compared to other OOPS language. We learned some GUI concepts, how to develop graphical user interfaces using the gtk+ toolkit, how to work with pointers and manipulating files in C. Writing the solver showed us how efficient smart algorithms are as compared to naïve algorithms. The backtracking algorithm seems to be a useful method to solve any sudoku puzzles and it can guarantee to find at least one solution.