

PROYECTO GLUD: SUDOKU

CRISTHIAN M. YARA P. - 20181020071

**PROGRAMA DE INGENIERÍA DE SISTEMAS
GRUPO DE TRABAJO GLUD
FACULTAD DE INGENIERIA**

**UNIVERSIDAD DISTRITAL FRANCISO JOSÉ DE CALDAS
BOGOTÁ D.C.
2018**

TABLA DE CONTENIDO

1. INTRODUCCIÓN.....	3
2. OBJETIVOS	4
2.1 Objetivo generales	4
2.3 Objetivo específico.....	4
3. LICENCIA.....	5
3.1 GPLv2	5
3. Marco teórico	7
3.1 Que es el Sudoku.....	7
3.2 Como se juega	8
4. Desarrollo	11
5.2 Análisis del problema	11
4.3 Planteamiento de la solución	11
4.4 Conclusiones.....	19
5. BIBLIOGRAFÍA.....	20

INTRODUCCIÓN

Este informe presenta la documentación con respecto al proyecto planteado para el grupo de trabajo de la Universidad Distrital GLUD, el cual es un SUDOKU, que contará con la licencia GPLv2, contribuyendo de ese modo al pensamiento de Richard Stallman y respetando las 4 reglas del software libre, el documento también contará la información necesaria sobre su desarrollo y el código fuente, en el cual cualquiera podrá modificar para solucionar deficiencias y mejorar la experiencia de cualquier usuario.

Para ello se ha planteado, la incorporación de un marco teórico que explique la teoría, y un análisis detallado del planteamiento para su solución.

OBJETIVO GENERAL

El objetivo general de este laboratorio es el de contribuir al desarrollo de las diferentes capacidades para resolver de manera eficaz los diferentes problemas que se han presentado en la solución de un SUDOKU, como los diferentes niveles de dificultad, el desarrollo de una matriz donde las filas y columnas no repitan el mismo número y en cada bloque de 3x3 de la matriz de 9X9, con el uso del lenguaje de programación C/C++. Este objetivo requiere la adquisición del conocimiento teórico y práctico sobre las técnicas básicas de programación estructurada consiguiendo la obtención de ideas intuitivas y claras de los conceptos y técnicas estudiados, y permitirá entender fácilmente nuevos modelos facilitando la aplicación práctica de los algoritmos.

OBJETIVOS ESPECIFICOS

- Desarrollar un juego matemático (sudoku), desarrollando la lógica en C/C++.
- Proporcionar a los interesados del código fuente del programa.
- Presentar la documentación necesaria, para generar una mayor comprensión del programa.
- Permitir la selección del tipo de dificultad.
- La implementación del pensamiento del software libre.
- Permitir al usuario generar un tablero con las respuestas.

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

La licencia de este proyecto es la GPL v2, Según Richard Stallman, el mayor cambio en GPLv2 fue la cláusula “Liberty or Death” («libertad o muerte»), como la llama en la sección 7 de ese documento.²¹ Esta sección dice que si alguien impone restricciones que le prohíben distribuir código GPL de tal forma que influya en las libertades de los usuarios (por ejemplo, si una ley impone que esa persona únicamente pueda distribuir el software en binario), esa persona no puede distribuir software GPL. La esperanza es que esto hará que sea menos tentador para las empresas el recurrir a las amenazas de patentes para exigir una remuneración de los desarrolladores de software libre.

En 1990 se hizo evidente que una licencia menos restrictiva sería estratégicamente útil para la librería C y para las librerías de software que esencialmente hacían el trabajo que llevaban a cabo otras librerías comerciales ya existentes.²² Cuando la versión 2 de GPL fue liberada en junio de 1991, una segunda licencia Library General Public License fue introducida al mismo tiempo y numerada con la versión 2 para denotar que ambas son complementarias. Los números de versiones divergieron en 1999 cuando la versión 2.1 de LGPL fue liberada, esta fue renombrada como GNU Lesser General Public License para reflejar su lugar en esta filosofía.

JUSTIFICACIÓN

Teniendo en cuenta que el grupo de trabajo GLUD es un grupo enfocado a las distribuciones del sistema operativo Linux, también se enfoca en la práctica y el pensamiento del Free Software y Open Source, con lo cual partiremos con el objetivo de desarrollar propuesta de software libre, en el que cualquier usuario interesado, pueda acceder al código fuente y plantear modificaciones, que permitan mejorar la eficiencia del código, el manejo de recursos y simplificación del mismo.

Se opta inicialmente por un proyecto que permita, mediante un análisis, establecer e identificar comportamientos frecuentes, niveles de dificultad, entre otros factores.

BENEFICIARIOS DEL PROYECTO

Los beneficiarios serán todas aquellas personas que quieran profundizar en el tema, entender la lógica, estudiar el código o incluso pasar un tiempo de recreación:

- ***Estudiantes.***
- ***Profesores.***
- ***Programadores.***
- ***Cualquier tipo de usuarios.***

3.1. MARCO TEÓRICO

¿Qué es el Sudoku?

Le llaman el Cubo Rubik del siglo XXI. Es el crucigrama lógico por excelencia: engañoso, juguetón y que engancha totalmente, sin que te haga falta ser un genio de las matemáticas para resolverlo.

Se hizo popular en Japón durante los años ochenta, una nación que adora los pasatiempos, pero cuya lengua no ofrece ninguna posibilidad de jugar con palabras, por lo que la mayoría de los crucigramas que inventan se basan en números o dibujos. No obstante, el sudoku empezó con los cuadrados latinos que inventó el matemático suizo Leonard Euler en el siglo XVIII. Se trataba de rellenar una tabla o cuadrícula de manera que cada columna, cada fila y cada uno de los nueve bloques cuadrados de la cuadrícula incluyeran un número de 1 al 9 sin que se repitieran ni se excluyera ninguno.

Parece fácil, y lo es, mientras tu mente esté alerta y tengas a goma a mano. Cualquiera puede hacerlo, y cuantos más descifres, más fácil te resultará. Así que prepárate para batallar ^[1].

[1]. Tomado de SuDoku 2, editorial Biblioteca El Tiempo 2006

COMO SE JUEGA

Los Sudokus se suelen estructurar en cuadrículas divididas en cajas de 3x3 celdas en las que hay algunos números escritos de antemano. Para jugar, simplemente debes rellenar las celdas en blanco de tal forma que cada fila, columna y caja de 3x3 no tenga números repetidos.

Así explicado parece sencillo, pero conforme uno se inicia en el rompecabezas, descubre que las cosas no son tan simples. Es más complicado de lo que parecía en un principio.

Es un rompecabezas que necesita de paciencia, agudeza visual y razonamiento.

Dependiendo de la dificultad del Sudoku se tarda más o menos tiempo en resolverlo. Los más fáciles se pueden resolver en unos pocos minutos y para los más difíciles se pueden emplear varias horas. Algunos ejemplos de Sudokus de diferentes niveles son:

FÁCIL

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

MEDIO

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

DIFÍCIL

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

MUY DIFÍCIL

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

Reglas del Sudoku

Las reglas del Sudoku son muy simples. En este rompecabezas no se trata de sumar nada con los números, ni que éstos tengan un orden lógico, sino que jugamos con los números como si fueran piezas de un puzzle, sin repetir ninguna ni en horizontal (filas), ni vertical (columnas), ni en las cajas de 3x3.

Cada una de las filas en Sudoku está compuesta por 9 celdas en las que debes poner la serie de números del 1 al 9 en el orden que creas oportuno, pero sin repetirlo y, obviamente, sin dejar ninguno por poner.

A su vez, las columnas también tienen la misma estructura, sólo que en vertical, que las filas y también sus condiciones de juego, es decir, al colocar un número en una fila tienes que tener en cuenta que no se repita en la columna en la que está incluido.

No conformes con esto, el juego se complica un poco más con las cajas de 3x3. Todas ellas deben contener en su interior la serie completa del 1 al 9.

Este es un ejemplo de Sudoku sin resolver y ya resuelto:

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

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

Métodos y consejos para resolver Sudokus

Algunos consejos prácticos para empezar son:

- Si estás empezando a hacer SuDoKus, lo más recomendable es que comiences por los niveles más fáciles y posteriormente, cuando tengas más práctica, aumentes la dificultad.
- utilizar lápiz y goma de borrar (a menos, claro, que lo estés haciendo en un ordenador)
- Comenzar por las cajas de 3x3 que contengan más números.

- Una buena ayuda puede ser escribir los números posibles de cada celda en pequeñito dentro de la misma. De esa manera, te será más fácil recordar todas las posibilidades.
- Recuerda que no hay que olvidarse de las cajas de 3x3 al descartar los números de las posiciones.

La metodología para resolver un Sudoku es la siguiente:

- Lo primero que se debe hacer es una visualización general de los números y sus posiciones con el fin de eliminar posibilidades, como por ejemplo, eliminar números por regiones (siempre que se pueda). Esto consiste en eliminar los números de una fila que falten pero ya estén incorporados dentro de una caja, veámoslo con este caso concreto:

			1	7	9	4	2	
3								

Sabemos que el 3 va en la última posición de la fila, ya que al estar dentro de la primera caja, le impide formar parte de las 3 primeras posiciones de la fila:

			1	7	9	4	2	3
3								

Esta regla se puede extender en el Sudoku de la siguiente manera, llamada escaneo, donde las líneas rojas eliminan las posiciones donde podría ir el 8 para las imágenes 1 y 2 y el número 2 para la imagen 3:

TRUCO 1

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

TRUCO 2

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

TRUCO 3

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

- Cuando es imposible que con el paso anterior se puedan descubrir números nuevos, es bueno recurrir al consejo anteriormente citado de marcar en cada celda los números candidatos a ocuparla.
- A este paso anterior sigue el de eliminación, en el que se escogen sucesivamente posibles soluciones hasta que se llega a la solución final. Esto se lleva a cabo mediante la elección de una de las posibles opciones de una celda y se realiza a partir de ella un nuevo escaneo. Sucesivamente, se eliminan las posibilidades que no nos llevan a la resolución del Sudoku. Un consejo: empezar por aquellas celdas que tengan menor cantidad de números candidatos.

3.2. ANÁLISIS DEL PROBLEMA

3.2.1 Problema

- Generación de matrices aleatorias que cumplan con las características propias del sudoku.
- Establecer el proceso de verificación del cumplimiento de las reglas en cada caja de texto.
- Establecer niveles de dificultad (próximas versiones).
- Establecer casillas que casilla serán visibles de manera aleatoria dependiendo de la dificultad (próximas versiones).

3.3. PLANTEAMIENTO DE LA SOLUCIÓN

Para la solución de matrices de 9X9 que cumplieran con el requisito que exige el juego se planteó una solución alternativa en C/C++, que aunque no está del todo implementada en el proyecto de JAVA, nos permite tener una noción de a donde se quiere llegar, debido a que la implementación en JAVA contempla el uso de una matriz fija.

3.3.1. Definición de funciones Generales:

Para el desarrollo del problema, establecemos una matriz de 9X9 el cual estará inicializado en ceros (0) mediante la función ***llenado_martiz()***.

```
int llenado_martiz(int _sudoku[][9]){
    for(int filas=0;filas<9;filas++)
        for(int columnas=0;columnas<9;columnas++)
            _sudoku[filas][columnas]=0;
}
```

La función ***numero_aleatorio()***, nos generará números aleatorios comprendidos entre el 1 y el 9, que albergará la variable ***NumeroEspecial*** en la función ***main()***.

```
int numero_aleatorio(){
    int variable=1+rand()%9;
    return variable;
}
```

Las funciones ***validando_bloque()***, ***validando_fila()*** y ***validando_columna()***, son las funciones que nos permitirán establecer si el número se repite en cada fila de la matriz, en cada columna o en cada bloque de 3X3.

validando_bloque():

```
bool validando_bloque(int _sudoku[][9],int valor, int fila, int columna) {
    //Variables para dererminar el rango

    int minimo_fila;
    int maximo_fila;
    int minimo_columna;
    int maximo_columna;
    boolean resultado = false;
    //Determinamos las filas del bloque
    if (fila >= 1 && fila < 4) {
        minimo_fila = 1;
        maximo_fila = 3;
    } else if (fila >= 4 && fila < 7) {
        minimo_fila = 4;
        maximo_fila = 6;
    } else {
        minimo_fila = 7;
        maximo_fila = 9;
    }
    //Deterinamos las columnas del bloque
    if (columna>=1 && columna <=3) {
        minimo_columna = 1;
        maximo_columna = 3;
    } else if (columna >= 4 && columna < 7) {
```

```

        minimo_columna = 4;
        maximo_columna = 6;
    } else {
        minimo_columna = 7;
        maximo_columna = 9;
    }
    //Recorremos la matriz en busqueda de un numero igual al ingresado.
    for (int f = minimo_fila; f <= maximo_fila; f++) {
        for (int c = minimo_columna; c <= maximo_columna; c++) {
            if (valor == 0) {
            } else {
                if (matriz[f][c] == valor) {
                    resultado = true;
                    break;
                }
            }
        }
    }
    return resultado;
}

```

validando_fila():

```

bool validando_fila(int _sudoku[][9],int _numero, int _fila) {
    bool resultado = false;
    /*Se realiza el recorrido a lo largo del arreglo (9 filas)
    *para comprobar si el número se repite
    */
    for (int i = 1; i <=9; i++) {
        /*Si el numero en la posición matriz[fila][i] existe
        entonces el resultado booleano será 'true'*/
        if (_sudoku[_fila][i] == _numero) {
            resultado = true;
            break;
        }
    }
    return resultado;
}

```

validando_columna():

```

bool validando_columna(int _sudoku[][9],int _numero, int _columna) {
    bool resultado = false;
    /*
    *Se realiza el recorrido a lo largo del arreglo (9 columnas)
    *para comprobar si el numero se repite
    */
    for (int i = 1; i <=9; i++) {
        /*Si el numero en la posición matriz[i][columna] existe
        entonces el resultado booleano será 'true'*/
    }
}

```

```

        if (_sudoku[i][_columna] == _numero) {
            resultado = true;
            break;
        }
    }
    return resultado;
}

```

3.3.2. Entrada – Proceso – Salida:

La columna vertebral para el llenado exitoso mediante el uso de las funciones anteriormente mencionadas se encuentra en el siguiente proceso:

```

for(filas=0;filas<9;filas++){
    for(columnas=0;columnas<9;columnas++){
        do{
            if (validando_fila(sudoku,NumeroEspecial,filas)== true) {
                contador++;
            } else {
                //Evaluamos si el numero se repite en la columna
                if (validando_columna(sudoku,NumeroEspecial,columnas)== true) {
                    contador++;
                } else {
                    //Evaluamos si el número se repite en el bloque correspondiente de
                    3X3
                    if (validando_bloque(sudoku,NumeroEspecial,filas,columnas)==true)
                    {
                        contador++;
                    } else {
                        *En caso de que no esté repetido añadimos el numero en la
                        casilla*/
                        sudoku[filas][columnas] = NumeroEspecial;
                        contador=0;
                    }
                }
            }
            NumeroEspecial=numero_aleatorio();
        }while((contador>0 && contador<4) && (sudoku[filas][columnas]==0));
    }
}

```

El anterior proceso se encarga de validar función por función y descarta los valores que encuentre repetidos. Repitiendo el proceso hasta llenar la matriz con números diferentes.

Para la visualización de la matriz usamos el siguiente proceso:

```

//Generamos la salida por pantalla del arreglo
cout<<"\n\n\n\t***-----*** SUDOKU ***-----***\n\n\n"<<endl;;
//Generamos la salida por pantalla del arreglo
for(int i=0;i<9;i++){

```

```

        for(int j=0;j<9;j++){
            cout<<"\t"<<sudoku[i][j];
            if(((j+1)%3)==0)
                cout<<"\t";
        }
        cout<<"\n\n";
        if(((i+1)%3)==0)
            cout<<"\n\n";
    }
    cout<<"\n\n\n";

```

Y al final el resultado se vera del siguiente modo:

```

***-----*** SUDOKU ***-----***

7      4      8          2      5      9          1      3      0
0      0      2          1      6      4          9      8      0
0      6      3          8      7      0          0      4      0

5      9      1          4      0      8          3      2      7
4      3      7          9      2      5          0      0      0
8      2      0          3      0      0          4      9      1

9      8      5          0      0      3          6      1      2
6      0      0          7      1      0          0      0      3
3      1      0          0      0      0          0      0      0

```

La anterior imagen ilustra la salida por pantalla de la ejecución del programa escrito en C/C++.

3.3.3. Desarrollo en Java:

Para la implementación en Java, se utilizaron en esencia las mismas funciones que en C/C++, con la diferencia de que el programa en Java incluye una interfaz gráfica, y algunas casillas quedan vacías las cuales el usuario determinará que valores incluye en cada caja de texto.

Para la ejecución del programa se hizo uso de los eventos del teclado en Java, los cuales son:

- **keyPressed(KeyEvent e):** Se ejecuta cuando el usuario presiona una tecla.
- **keyReleased(KeyEvent e):** Se ejecuta cuando el usuario libera una tecla
- **keyTyped(KeyEvent e):** Se ejecuta cuando el usuario presiona una tecla, pero solo cuando la tecla corresponde a caracteres, teclas especiales como F1, F2 entre otras no son identificadas.

Pero en el que únicamente usamos **keyReleased(KeyEvent e)**, el código implementado en cada acción de estas se explica en el siguiente código:

```
private void j12KeyReleased(java.awt.event.KeyEvent evt)

    int con = 0;
    //Evaluamos que el valor ingresado sea un numero entero valido
    if (sudoku_metodos.comprobar_valor((j12.getText()))) {
        //Evaluamos si el numero se repite en la fila
        if (sudoku_metodos.existe_fila(Integer.valueOf(j12.getText()), 1)) {
            //En caso de repetirse en la fila se envia el siguiente mensaje
            JL_informacion.setText("el numero " + j12.getText() + " ya esta en la fila");
            //se desocupa la caja
            j12.setText("");
        } else {
            //Evaluamos si el numero se repite en la columna
            if (sudoku_metodos.existe_columna(Integer.valueOf(j12.getText()), 2)) {
                //En caso de repetirse en la columna se envia el siguiente mensaje
                JL_informacion.setText("el numero " + j12.getText() + " ya esta en la
columna");
                //se desocupa la caja
                j12.setText("");
            } else {
                //Evaluamos si el numero se repite en el bloque correspondiente de 3X3
                if (sudoku_metodos.existe_caja(Integer.valueOf(j12.getText()), 1, 2)) {
                    //En caso de repetirse en el bloque de 3X3, se envia el siguiente mensaje
                    JL_informacion.setText("el numero " + j12.getText() + " ya esta en la
caja");
                    //se desocupa la caja
                    j12.setText("");
                } else {
                    /*
                    *En caso de que no esté repetido en ningún caso añadimos
                    *el numero en la casilla y mantenemos la casilla de
                    *información desocupada
                    */
                    sudoku_metodos.matriz[1][2] = Integer.valueOf(j12.getText());
                }
            }
        }
    }
}
```



```

        JL_informacion.setText("");
        con++;
    }
}
}
}
//Si el numero ingresado no es un numero la casilla queda vacía
else {
    j12.setText("");
}
/*
 *Si contador 'con' es mayor a uno es porque el número ingresado
 *es válido, en caso contrario la ubicación de la matriz se mantendrá en cero
 */
if (con == 0) {
    sudoku_metodos.matriz[1][2] = 0;
}
}
}

```

Teniendo en cuenta que **j12** corresponde a cada caja de texto y su respectiva ubicación.

J12 →

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

Detalles del juego:

Terminar

La ubicación de las variables está definida por la siguiente matriz:

$$\begin{bmatrix} J11 & J12 & J13 & J14 & J15 & J16 & J17 & J18 & J19 \\ J21 & J22 & J23 & J24 & J25 & J26 & J27 & J28 & J29 \\ J31 & J32 & J33 & J34 & J35 & J36 & J37 & J38 & J39 \\ J41 & J42 & J43 & J44 & J45 & J46 & J47 & J48 & J49 \\ J51 & J52 & J53 & J54 & J55 & J56 & J57 & J58 & J59 \\ J61 & J62 & J63 & J64 & J65 & J66 & J67 & J68 & J69 \\ J71 & J72 & J73 & J74 & J75 & J76 & J77 & J78 & J79 \\ J81 & J82 & J83 & J84 & J85 & J86 & J87 & J88 & J89 \\ J91 & J92 & J93 & J94 & J95 & J96 & J97 & J98 & J99 \end{bmatrix}$$

3.4. CONCLUSIONES

El proyecto del sudoku se encuentra en fase de desarrollo, ya que la primera versión contempla un único nivel de dificultad y una sola plantilla de solución.

El desarrollo en C/C++ busca establecer matrices aleatorias que contribuyan al desarrollo en la versión de Java, el cual también incluirá, la visualización de algunas casillas y su ubicación según se determine en el nivel de dificultad.

BIBLIOGRAFÍA

- [1] Grossman, S., (2012), *Algebra Lineal, Séptima edición*, México DF, México: MC Graw Hill.
- [2] <https://www.matesfacil.com/matrices/resueltos-matrices-suma.html>
- [3] Huergo, Luis A.,(20xx), *Aritmética Binaria*, Departamento de Telecomunicaciones.
- [4] https://es.wikipedia.org/wiki/Cuadrado_mágico
- [5] rincondelvago.com/informacion/sudoku/tecnicas-de-resolucion-de-un-sudoku
-