

UNIVERSIDAD DE SANTIAGO DE CHILE

FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INFORMÁTICA

**PROGRAMACION SUDOKU**

**Utilizando Lenguaje C**

**POR: RODRIGO CERDA R.**

**PROFESOR: ROBERTO GONZALEZ I.**

**Santiago, 24 de Noviembre 2015**

* 1. Antecedentes y motivación

En el presente proyecto se plantea la realización de un programa capaz de entregar al usuario tableros de sudoku de MxN dimensiones con un grado de dificultad ingresado por el usuario, utilizando para ello el lenguaje C, en el entorno de desarrollo Dev-C++.

La motivación de llevar a cabo este proyecto es poder abarcar un problema como este a través del paradigma de programación imperativa, concentrando la parte algorítmica del juego en iteraciones capaces de conformarlo.

* 1. Descripción del problema

Abordar este problema implica entregar un tablero que sea de dimensiones y datos iniciales variables según el usuario lo disponga, siempre y cuando satisfaga las reglas de sudoku, además de tener el desafío de almacenar en ficheros la información del tablero y del juego, para luego ser capaz de invocar dicha información.

* 1. Solución propuesta

Para efectos de este proyecto se sugiere en el enunciado utilizar estructuras para el tablero y para el juego.

En el caso del tablero, la estructura consiste en los datos tamaño, dificultad, y un arreglo bidimensional de int para almacenar de manera dinámica el tablero.

En el caso del juego, su estructura básicamente consiste en los datos jugador(char), un arreglo de dimensión [1] de tableros, el boardId del tablero a almacenar, y un dato para el puntaje

* 1. Objetivos y alcances del proyecto
     1. Objetivo general

Implementar con lenguaje C un sistema de juego de sudoku valido según los datos de entrada del usuario.

Objetivos específicos

Generar tableros de manera dinámica.

Imprimirlos en pantalla.

Guardar y cargar los tableros en un fichero.

Guardar y cargar los juegos en un fichero.

Chequear la validez del tablero.

Obtener el puntaje de un tablero.

Generar una DLL del proyecto.

* 1. Metodologías y herramientas utilizadas

Las herramientas utilizadas básicamente son los conocimientos adquiridos en clases relacionados con el lenguaje C y la programación imperativa procedural.

**CAPITULO I: FUNDAMENTOS**

Un sudoku es un juego matemático que se presenta en forma de cuadricula, comúnmente de 9x9 , en la que hay que llenar los 81 espacios disponibles según ciertas reglas:

1. Los números para llenar deben ser desde 1 a 9
2. El número de una celda solo puede estar una vez en la fila que la contiene.
3. El número de una celda solo puede estar una vez en la columna que la contiene.
4. El número de una celda solo puede estar una vez en la subcuadricula de 3x3 que la contiene.

Si bien, las dimensiones ejemplificadas son las más comunes en la elaboración de tableros, también es posible diseñarlos de distintos tamaños, siempre que:

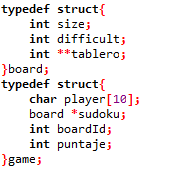
1. Sus lados sean iguales.
2. La raíz de sus lados sea un número natural.

**CAPITULO II: REPRESENTACIÓN**

En la solución propuesta explicada más arriba, se mencionan las estructuras básicas para este proyecto, siendo la estructura board, la que almacenara el tablero.

La manera en la que almacenará el tablero es a través de un arreglo bidimensional de tipo int, en el cual su tamaño, igual en ambas dimensiones, es [0, Dimensión[.

Sin embargo, en el operar del proyecto se emplea un puntero a un arreglo bidimensional de estructuras board, esto para poder manejar dos tableros, ambos iniciados idénticamente, pero solo uno utilizado para la manipulación del usuario. Esto se diseña así, para poder validar una jugada que no sobre escriba un número creado en su origen.



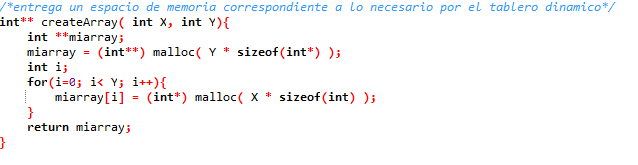
*Imagen 2.1.Estructuras.*

**CAPITULO III: MANEJO DINAMICO DE MEMORIA**

Es memoria que se reserva en tiempo de ejecución. Su principal ventaja frente a la estática, es que su tamaño puede variar durante la ejecución del programa.

Es necesario que sea de esta manera para poder abarcar con la misma estructura, distintos tipos de tableros (de distintas medidas).

La manera en la que se realizó esto fue con una función capaz de generar el espacio en memoria necesario para el puntero contenido en la estructura board, a través de “malloc” y “sizeof”

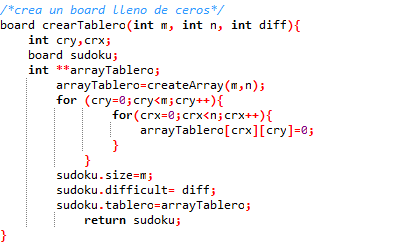


*Imagen 3.1.CreateArray.*

**CAPITULO IV: CREAR TABLERO VALIDO**

Teniendo ya la manera de asignar dinámicamente memoria al puntero dentro de la estructura board, es posible operar sobre una instancia de esta.

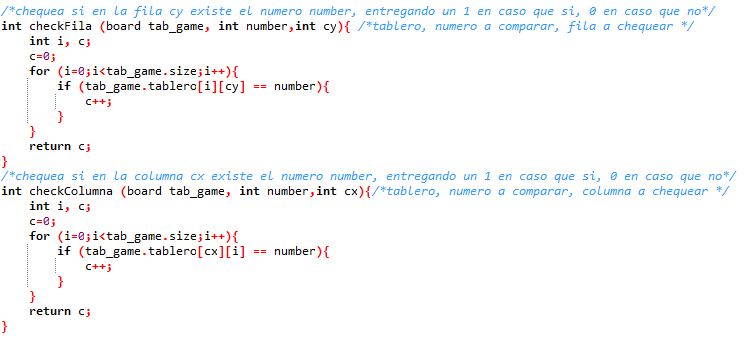
Para el tablero, se genera una función que inicializa a cero cada espacio del array que lo almacena, así como también se llenan los otros datos de la estructura.

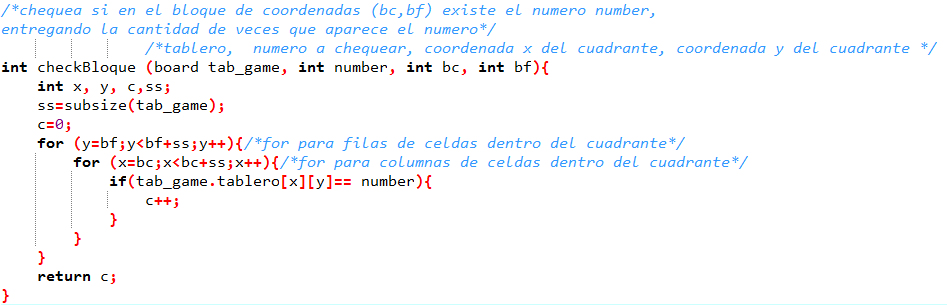


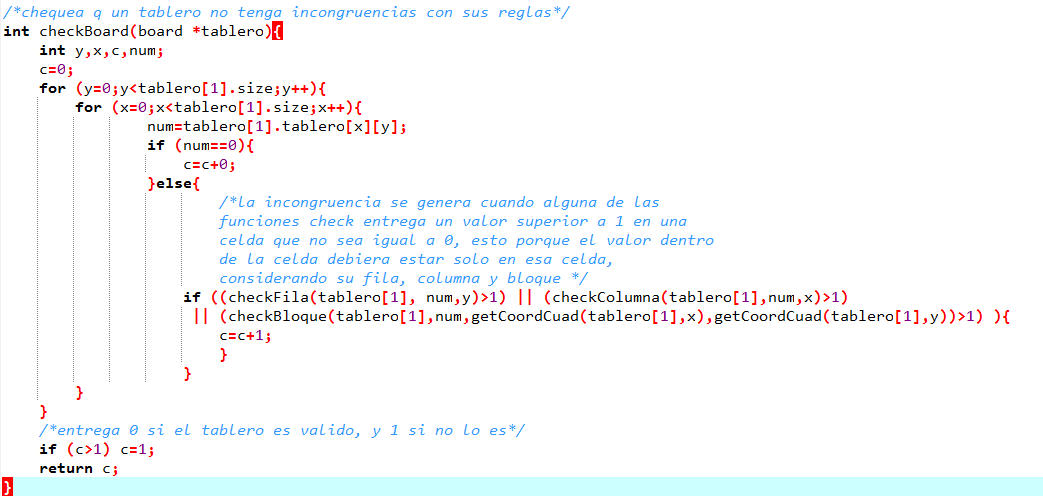
*Imagen 4.1CrearTablero.*

Luego de la representación inicial, se busca generar funciones capaces de determinar el cumplimiento de las reglas del juego en una celda en particular.

De aquí surgen las funciones checkfila, checkcolumna, checkbloque, que luego conformarán, checkboard.

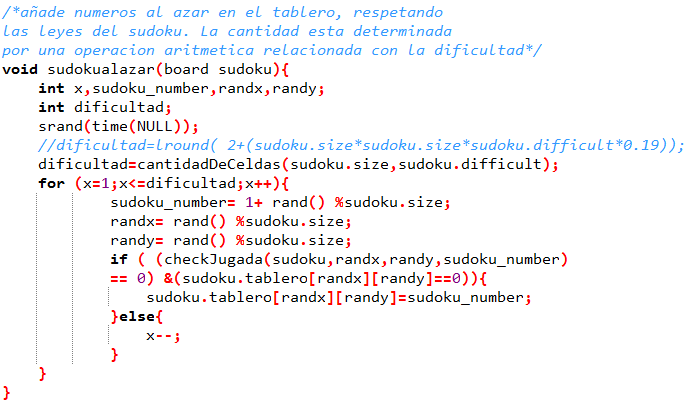


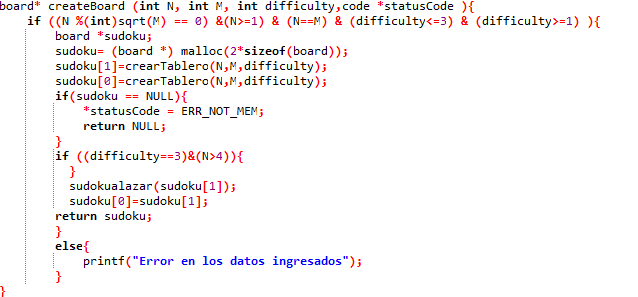
**

**

*Imagen 4.2. Funciones check.*

Despues de tener el tablero lleno de ceros, y las funciones capaces de determinar el cumplimiento de reglas, a traves de una funcion (sudokualazar) que toma en cuenta la dificultad como parametro de cantidad, se llena dicho el tablero con numeros validados en su generación, dicha función se integra a createboard para crear la estructura board.

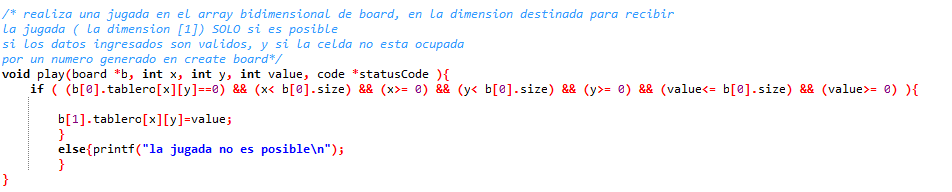




*Imagen 4.3.Funciones sudokualazar y CreateBoard.*

Notar que en esta función, se está generando espacio al puntero sudoku con capacidad para dos estructuras board, las cuales serán idénticas saliendo de esta función.

**CAPITULO V: PLAY**

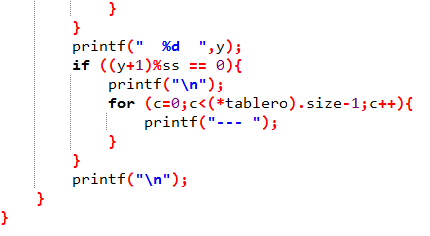
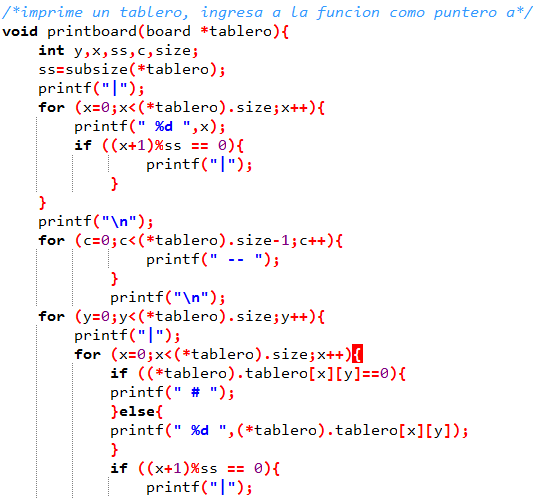


*Imagen 4.1.Función play.*

Esta función recibe un puntero de 2 Boards, de los cuales el de índice 0 se utiliza para validar una jugada en el indice 1.

Después de validar los datos ingresados, se determina que la jugada será posible si y solo si para las mismas coordenadas ingresadas, el board de índice 0 tiene un cero.

**CAPITULO VI: PRINTBOARD**

**

*Imagen 6.1.Función printboard.*

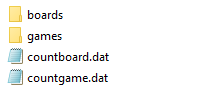
Esta función recibe como parámetro un puntero a board, imprimiendo el tablero de manera de verlo como tal.

Se le añaden caracteres especiales para separar subtableros, así como un índice para las coordenadas X e Y dándole mayor comprensión.

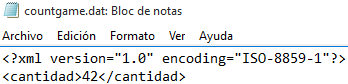
**CAPITULO VII:ALMACENAMIENTO**

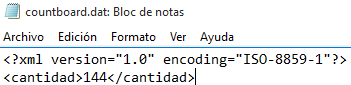
Para almacenar datos, se emplearan etiquetas simulando el formato XML, además, se determina utilizar ficheros de texto extensión .dat, almacenando los board dentro de la carpeta “boards/”, y los games en la carpeta “games/”

Además de la generación de archivos para almacenar dichas estructuras, se determina generar archivos que lleven la cuenta de cuantos boards y cuantos games se han generado. Esto para poder generar una id única respecto a cada estructura en su guardado.



*Imagen 7.1. Archivos de almacenamiento.*

**



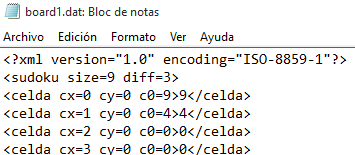
*Imagen 7.2. Estructura de contadores.*

Para el manejo de los ficheros contadores countgame.dat y countboard.dat., se crean funciones capaces de evaluar su existencia y su contenido.

incCountBoard() e incCountGame() son funciones de retorno int. Evalúan la existencia de sus respectivos ficheros, y, si no existe, lo generan e inicializan el contador de cantidad a 1.

En caso de existir, evalúan el valor dentro del nodo cantidad, y lo actualizan incrementándolo en uno, tomando este nuevo valor como retorno de la función.

En definitiva, por medio de la información almacenada en los ficheros contadores, mas estas funciones, se van generando identificadores que se incrementan en una unidad cada vez que se guarda una estructura.



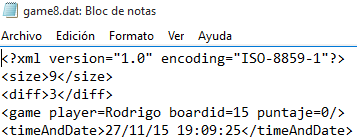
*Imagen 7.3. Estructura de ficheros de almacenamiento de boards.*

*En los ficheros que almacenan boards, la primera línea es de declaración del formato XML, luego, la segunda línea, inicializa el nodo sudoku con atributos correspondientes al tamaño y dificultad del sudoku.*

*Dentro del nodo sudoku se generan nodos correspondientes a las celdas, cuyos atributos son: coordenada X, coordenada Y, valor de la celda en la dimensión cero.*

*Dentro de los nodos celda, se almacena un elemento correspondiente al valor de la celda dentro de la dimensión uno.*

Al final de cada fichero se almacena la hora y fecha de guardado.



*Imagen 7.3. Estructura de ficheros de almacenamiento de games.*

En los ficheros de almacenamiento de games, la primera línea declara el formato XML, luego, la segunda y tercera fila almacenan el tamaño y la dificultad del tablero relacionado con el juego. Esto para utilizarlo en loadboard.

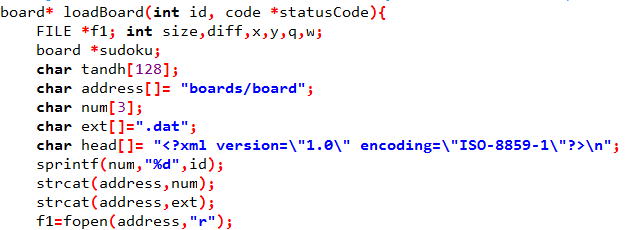
Luego en la cuarta línea, se genera el nodo game, en el cual se almacenan los atributos player, relacionado con el nombre del jugador. Boardid, registra el id del board relacionado con el juego. Puntaje, porcentaje del tablero correctamente solucionado.

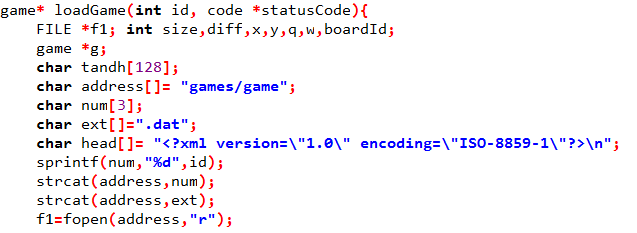
Al final de cada fichero se almacena la hora y fecha de guardado.

**CAPITULO VIII: LLAMADOS.**

LoadBoard y loadgame, son funciones que retornan un puntero a board y a game respectivamente. En su llamado generan el espacio de memoria necesario para almacenar los datos cargados desde los ficheros.

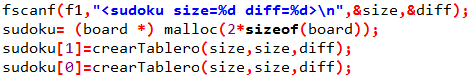
En sus parámetros de entrada, se solicita el id de la estructura a cargar, número que se transforma a char, y se concatena con una serie de char que en su conjunto generan el nombre del archivo a llamar.



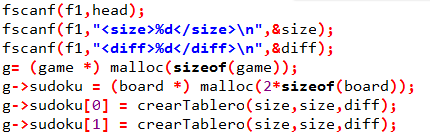


*Imagen 8.1.generación de nombre del archivo.*

Luego de llamar al archivo, se escanean las líneas de datos, capturando la información de los nodos declarados en las funciones save. Fue necesario en el guardado de los ficheros, guardar primero el tamaño, para que en el momento de que la funcion load obtiene dicho tamaño, genere el espacio en memoria para la estructura a cargar.

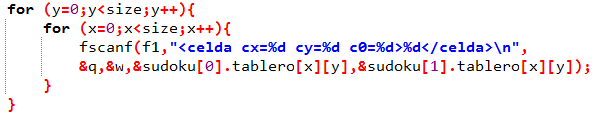


*Imagen 8.2.generación de memoria de loadboard.*



*Imagen 8.3. Generación de memoria de loadgame.*

Luego de esto, en el caso de loadboard, genera un ciclo para capturar todas las celdas necesarias.



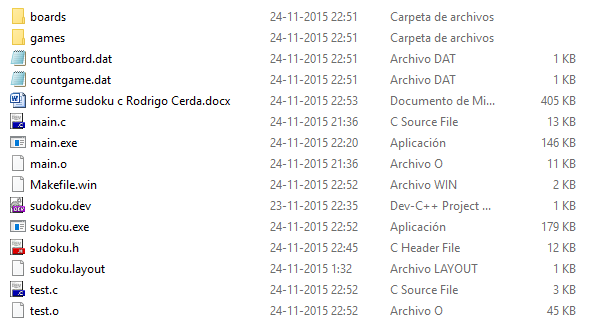
*Imagen 8.4 . Carga de tablero en loadboard.*

En el caso de loadgame, luego de la generación de memoria necesaria, se llama a la función loadboard, alojando el puntero que entrega dicha función, en el parámetro sudoku de la estructura game.

****

*Imagen 8.5 . Carga de sudoku en loadgame.*

**CAPITULO IX: CARPETA DE PROYECTO**

****

*Imagen 9.1 Carpeta de proyecto*

Dentro de la carpeta del proyecto se pueden encontrar las carpetas destinadas a almacenar los ficheros.

Señalar que el proyecto opera asumiendo la existencia de estas.

Countboard y countgame almacenan la cantidad de ficheros que se han guardado para cada estructura. En caso de no existir estos ficheros, el programa los genera automáticamente.

Sudoku.H es el archivo de cabecera en el que se especifican todas las funciones desarrolladas en el proyecto. Debe cargarse en el archivo .c en el que se prueben.

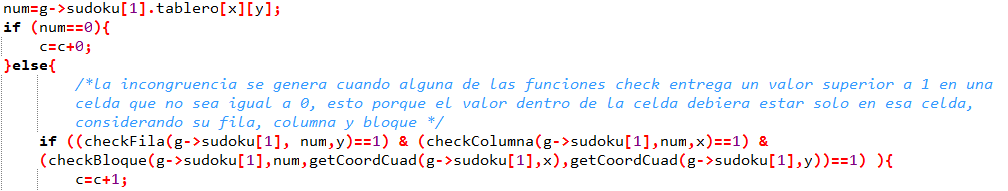
Test.c es un archivo diseñado para el análisis de las funciones desarrolladas, con llamados ya definidos, cubiertos por símbolos de comentario, para poder utilizarlos cuando se estime necesario.

**CAPITULO IX: EXTRAS**

Dentro de los extras del proyecto, se plantea generar una función capaz de entregar el puntaje del juego en su desarrollo.

Esto se logra, generando una función similar a checkboard, en la cual se recorren todas las celdas, incrementando un contador en caso de que la celda evaluada no sea cero y cumpla con todas las reglas del juego.

Luego de la evaluación, se obtiene la cantidad total de celdas correctas, cantidad que, al restarle la cantidad de celdas generadas en un principio, determinada por el tamaño y la dificultad, entrega la cantidad de celdas ingresadas por el usuario con correcta evaluación.



Para calcular el puntaje del tablero desarrollado, se realiza una regla de tres, donde el 100% es el total de celdas del tablero menos las celdas generadas al comienzo.

Y las celdas correctamente evaluadas menos las celdas generadas al comienzo determinan el puntaje.



Notar que al ingresar celdas que generen inconsistencias en las reglas de las celdas generadas al comienzo, el puntaje será negativo, por lo que se determina asignar cualquier puntaje negativo a cero.

**CAPITULO X: CONCLUSIONES**

Se logran los objetivos estipulados en un principio. Generando tableros dinámicamente, capaces de ser modificados, chequeados, y almacenados en ficheros a los que se puede invocar para continuar con el juego.

Se logra también la generación de una función getscore, utilizada para evaluar el tablero y almacenar en game el puntaje obtenido.