|  |  |  |
| --- | --- | --- |
| **CARRERA:**  Ingeniería en Software | **GUÍA**  No. 01 | **TIEMPO ESTIMADO:** |
| **ASIGNATURA:**  Estructura de datos  NRC: 2967 | **FECHA DE ELABORACION:** 27-01-2020  **SEMESTRE**: septiembre 2019 – enero 2020 | |
| **TÍTULO:**  Proyecto Primer Parcial (Sudoku) | **DOCENTE:** Ing. Fernando Solís | |

**OBJETIVO**

Aplicar el conocimiento de la recursividad y back tracking en el desarrollo de un sudoku 9\*9.

**INSTRUCCIONES**

1. Utilice como material principal, las principales herramientas para desarrollo en C++
2. Utilice información consultada en Internet y conocimiento adquirido en clase.

**ACTIVIDADES**

1. **Ubicación de recursos**
2. Formar grupos de máximo 2 personas por computador
3. Realizar el programa en cualquier IDE para C++
4. **Planteamiento del problema**

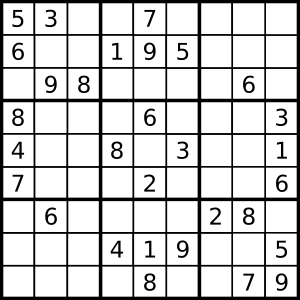
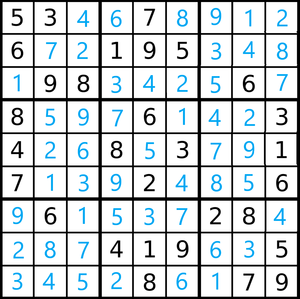
**p.e. Ejercicio No 1**

Desarrollar un programa que aplique los conocimientos aprendidos en el primer parcial, principalmente “back tracking” y recursividad. Utilizar memoria dinámica

1. **Marco Teórico**

El Sudoku es un juego matemático que se inventó a finales de la década de 1970, adquirió popularidad en Japón en la década de 1984 y se dio a conocer en el ámbito internacional en 2005 cuando numerosos periódicos empezaron a publicarlo en su sección de pasatiempos.1​El objetivo del sudoku es rellenar una cuadrícula de 9 × 9 celdas (81 casillas) dividida en

subcuadrículas de 3 × 3 (también llamadas "cajas" o "regiones") con las cifras del 1 al 9 partiendo de algunos números ya dispuestos en algunas de las celdas.



**¿Como resolverlo con back tracking?**

Una llamada inicial: ‘‘‘simplemente se encarga de llamar a la función recursiva. Para darle más utilidad retornará un booleano que indicará si se ha encontrado alguna solución al sudoku.

Una función recursiva: ‘‘‘se encarga del proceso de ir probando números. Incluye un parámetro de entrada/salida que indica si se ha encontrado alguna solución.

El comprobador de validez: ‘‘‘simplemente es una función que retorna un booleano indicando si un valor puede estar en un sitio indicado.

1. **Conclusiones**

La recursividad hace que el resolver estos problemas sean un poco más cortos, pero con más grao de complejidad a la hora de entenderlos como funcionan.