Programación I (Gestión y Sistemas), curso 2008/09

Práctica de la asignatura – HyperSudoku

Introducción

El HyperSudoku es una variante del sudoku tradicional. Al igual que los sudokus normales, se dispone de un tablero de 9x9 celdas que se debe rellenar con los números 1 al 9, sin que existan valores repetidos en cada fila, en cada columna, y en cada uno de los 9 bloques de 3x3 celdas. Lo que cambia respecto al sudoku normal es que se definen 4 nuevos bloques de 3x3 celdas, que se solapan con los 9 tradicionales, en los que tampoco puede haber valores repetidos.

En la siguiente figura se muestra un HiperSudoku y su solución, los bloques de celdas sombreadas corresponden a los 4 nuevos bloques:

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

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

Objetivos

El objetivo de la práctica es construir un programa que sea capaz de resolver un HyperSudoku de forma eficiente **encontrando todas las soluciones**. La estrategia a seguir consistirá en utilizar la recursividad para generar todas las combinaciones posibles de valores de las celdas vacías, escribiendo las soluciones completas que se vayan obteniendo.

En los HyperSudokus normales sólo existe una solución para el problema planteado. En ésta práctica, sin embargo, se puede proporcionar unos datos de entrada para los cuales existan más de una (o ninguna) solución posible.

Requisitos

El programa que resuelva la práctica debe pedir un nombre de un fichero de texto que contiene el HyperSudoku a resolver, el programa leerá el fichero y si contiene errores mostrará un mensaje y terminará. En caso contrario resolverá el HyperSudoku, mostrando las soluciones encontradas.

El formato del fichero de texto que contiene el sudoku a resolver es el siguiente: Constará de 9 líneas, una por cada fila del HyperSudoku. Cada línea contiene como máximo 9 caracteres, cada carácter indicando un valor de la columna correspondiente: Los caracteres del 1 al 9 indican que la casilla está preasignada con ese valor, y un espacio en blanco que la casilla está vacía. **Nota**: Se permite que las líneas tengan menos de 9 caracteres. En ese caso se supone que las columnas restantes están vacías.

El ejemplo anterior se almacenaría de la siguiente manera en el fichero de texto (El símbolo ␣ representa un espacio en blanco):

```
␣␣␣␣␣␣␣1
␣␣2␣␣␣␣34
␣␣␣␣51
␣␣␣␣␣65
␣7␣3␣␣␣8
␣␣3
58␣␣␣␣9
69
```

El formato de la salida del programa será el siguiente: Cada HyperSudoku se escribirá usando 9 líneas cada una de ellas con los 9 dígitos de esa fila del HyperSudoku. Si existieran varias soluciones, cada una se separa de la siguiente por una línea en blanco. Tras mostrar las soluciones, debe aparecer una línea con el formato "***n*** solucion(es) encontrada(s)", donde *n* es el número de soluciones, que puede ser 0 si el HyperSudoku no tiene solución.

Nota: No importa el orden en que se escriban las distintas soluciones.

La salida del programa en el ejemplo anterior debería ser:

```
946832715
152697834
738451296
819726543
475319682
263548179
327985461
584163927
691274358

1 solucion(es)encontrada(s)
```

Nota: Si existieran más de 1000 soluciones se permite que el programa calcule y presente solamente las 1000 primeras. En ese caso la última línea de la salida sería "Más de 1000 soluciones encontradas".

La práctica se probará de forma automática, por este motivo debe respetarse tanto el formato de entrada como el de salida.

Especificaciones adicionales

El procedimiento recursivo que resuelve el problema es obligatorio que tenga la siguiente cabecera:

```
procedure Resolver(var S: THSudoku; Fil,Col: integer);
```

Donde **THSudoku** es el tipo de datos que almacena la información referente al contenido del HyperSudoku en un momento dado. Cuando se llama a *Resolver*, se cumple que todas las celdas de las filas anteriores a *Fil* están asignadas, y dentro de la fila *Fil*, todas las columnas anteriores a *Col* están asignadas. Además, los valores asignados cumplen las reglas del HyperSudoku.

Si *Fil* es mayor que 9 significa que **todas** las celdas del HyperSudoku están asignadas, y además correctamente. En este caso *Resolver* ha encontrado una solución y debe escribirla por pantalla (éste es el caso base).

Si no es el caso anterior, la tarea de *Resolver* es generar todos los HyperSudokus posibles pero **respetando** la asignación actual. Para ello, si la celda [*Fil*,*Col*] está vacía, la asignará con todos los valores de 1 a 9 posibles (que cumplan las reglas del HyperSudoku) y tras cada asignación se llamará recursivamente a sí mismo con los valores de *Fil* y *Col* actualizados a la celda siguiente, con el objetivo de que se resuelva el (sub)problema de generar todos los HyperSudokus posibles respetando las asignaciones anteriores (pero ahora el problema es más sencillo ya que el número de celdas vacías es uno menos).