Práctica 1 - *Sudoku*

Estructura de Datos
2009-2010

Objetivos: Aprender a programar una clase sencilla en C++ y a manejar algunos contenedores de la STL.

Tareas a realizar: El alumno deberá entregar los códigos necesarios para jugar al *Sudoku* (`http://es.wikipedia.org/wiki/Sudoku`). El tablero inicial se generará aleatoriamente (ver Apéndice). El tablero de juego debe permitir guardar en cada celda varios números como posibles opciones según indique el jugador.

Para este juego se utilizarán dos TADs:

- `celda`: Clase que implementa una celda de un tablero.
- `sudoku`: Clase que se encarga del juego y conoce sus reglas.

Ficheros a entregar:

- `celda.h`
  Especificación de la clase que corresponde a una celda de un tablero. Una celda contiene un valor y un conjunto de opciones de valores para esta celda. Una celda vacía se representa como una celda cuyo valor es cero. Una celda puede ser fija, es decir, su valor no puede ser modificado una vez que es introducido. El invariante de la clase celda especifica que si una celda no está vacía entonces el conjunto de opciones debe estar vacío.

- `sudoku.h`
  Especificación de la clase encargada del juego. Un sudoku contiene un tablero de celdas de tamaño 9×9 y una pila de jugadas. El tablero puede tener tanto celdas fijas como modificables. El invariante de la clase especifica que cada celda del tablero y cada elemento de la lista de opciones de la celda debe tener valores entre 0 y 9 en donde el cero representa una celda vacía. Además no puede haber en una misma fila, columna o bloque dos valores iguales, ni en la celda ni en la lista de opciones de la misma. Un bloque se define como una matriz de 3×3. Un sudoku contiene 9 bloques, como se puede ver en el siguiente esquema:

```
_ _ _  2 1 _  _ _ _
9 _ _  _ 7 _  5 _ _
_ 2 _  5 3 _  _ 9 4

_ 9 _  _ _ _  _ 4 7
_ _ _  1 _ 4  _ 5 _
_ _ _  _ _ _  _ _ _

_ _ _  _ 5 _  3 1 _
5 _ 7  _ _ _  _ _ _
8 _ 1  _ _ _  _ 6 _
```

- `celda.cpp`
  Implementación de la clase que controla una celda.

- `sudoku.cpp`
  Implementación de la clase encargada del juego.

- `juego.cpp`
  Contiene el main encargado de producir la interfaz entre el juego y el jugador. Este archivo debe contener el menu para jugar conteniendo, al menos, las siguientes opciones:

```
1 - Agregar opcion a una celda.
2 - Borrar opcion de una celda
3 - Fijar valor a una celda.
4 - Borrar valor de una celda.
5 - Verificar factible.
6 - Deshacer.
7 - Salir.
```

Requisitos: El TDA `celda` debe contener, entre otros, *al menos* los siguientes métodos:

- `celda()`
  Constructor por defecto de la clase que inicializa la celda vacía.

- `bool fijar_valor(const unsigned int &dato,`
  `const bool &fija = false)`
  Función que introduce un valor en una celda e indica si el mismo es o no fijo. Si la celda ya es fija entonces no se puede modificar y por tanto la función devuelve `false`. En caso contrario devuelve `true`.

- `void borrar_valor()`
  Procedimiento que borra el contenido de una celda.

- `unsigned int valor() const`
  Función que devuelve el valor que aloja la celda.

- `bool valida() const`
  Función que devuelve `true` si la celda no está vacía y `false` en caso contrario.

- `bool nueva_opcion(const unsigned int &dato)`
  Función que guarda una nueva opción en la celda. Si el dato ya existe en el conjunto de posibles opciones, entonces devuelve `false`, sino devuelve `true`.

- `bool borrar_opcion(const unsigned int &dato)`
  Función que borra una opción de la celda. Si el dato no existe en el conjunto de posibles opciones, entonces devuelve `false`, sino devuelve `true`.

- `set<unsigned int> opciones() const`
  Función que devuelve las opciones de la celda.

Además se debe realizar la siguiente sobrecarga:

- `ostream& operator<<(ostream& os, const celda& c)`
  Sobrecarga del operador `<<` para la clase `celda`. Este operador mostrará solamente el valor definitivo si es valido, sino deberá mostrar un "`_`". No se mostrarrán las opciones de la celda.

El TDA `sudoku` debe contener, entre otros, *al menos* los siguientes métodos:

- `sudoku(const string &file)`
  Constructor de la clase que inicializa el juego con un tablero dado por archivo (ver Apéndice).

- ```
  bool fijar_valor(const unsigned int &fila,
                   const unsigned int &columna,
                   const unsigned int &valor)
  ```
  Esta función fija el valor dado en la celda ubicada en la posición (fila,columna) del tablero en caso de que la jugada sea valida, en cuyo caso se devolverá `true`, o `false` en caso contrario.

- ```
  void borrar_valor(const unsigned int &fila,
                    const unsigned int &columna)
  ```
  Procedimiento que elimina el valor de la celda ubicada en la posición (fila,columna) del tablero.

- `bool finalizado() const`
  Función que devuelve `true` si el juego ha finalizado, es decir, todas las celdas contienen valores y éstos son válidos según las reglas del juego.

- ```
  list<unsigned int> opciones(const unsigned int &fila,
                              const unsigned int &columna) const
  ```
  Función que devuelve las opciones que ha dejado el jugador guardadas la celda ubicada en la posición (fila,columna) del tablero.

- ```
  bool nueva_opcion(const unsigned int &fila,
                    const unsigned int &columna,
                    const unsigned int &valor)
  ```

Función que guarda un nuevo valor en las opciones de la celda ubicada en la posición (fila,columna) del tablero. Si el dato ya existe en las opciones de la celda, entonces devuelve `false`, sino devuelve `true`.

- `bool borrar_opcion(const unsigned int &fila, const unsigned int &columna, const unsigned int &valor)`
  Función que borra el valor indicado de las opciones de la celda ubicada en la posición (fila,columna) del tablero. Si el dato no existe en las opciones de la celda, entonces devuelve `false`, sino devuelve `true`.

- `const celda valor(const unsigned int &fila, const unsigned int &columna) const`
  Función que devuelve la celda ubicada en la posición (fila,columna) del tablero.

- `bool es_factible(const unsigned int &fila, const unsigned int &columna, const unsigned int &valor) const`
  Función que devuelve `true` si el valor dado puede ubicarse en la celda ubicada en la posición (fila,columna) del tablero según las reglas del juego.

- `bool undo()`
  Función que deshace la ultima jugada. En el caso en que no haya más jugadas para deshacer la función devuelve `false` sino devuelve `true`.

Además se debe realizar la siguiente sobrecarga:

- `ostream& operator<<(ostream& os, const sudoku& c)`
  Sobrecarga del operador `<<` para la clase `sudoku`. Se muestran tanto el tablero como las opciones alojadaas en cada una de las celdas.

Apéndice

Generación de tableros aleatorios (`http://davidbau.com/downloads/sudoku.py`):

Para generar tableros aleatorios se puede utilizar el programa `sudoku.py`. Tan solo ejecutando `python sudoku.py` genera un puzzle nuevo. El formato de los archivos que se pueden utilizar es como el siguiente:

```
PUZZLE:
_ _ _  2 1 _  _ _ _
9 _ _  _ 7 _  5 _ _
_ 2 _  5 3 _  _ 9 4

_ 9 _  _ _ _  _ 4 7
_ _ _  1 _ 4  _ 5 _
- - -  - - -  - - -

_ _ _  _ 5 _  3 1 _
5 _ 7  _ _ _  _ _ _
8 _ 1  _ _ _  _ 6 _
```

```
RATING: 1.5
```

Sudoku resuelto:

```
PUZZLE:
4 5 8  2 1 9  6 7 3
9 1 3  4 7 6  5 8 2
7 2 6  5 3 8  1 9 4

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

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

RATING: 1.5
```