Fundamentos de la programación

Práctica 2. Simon Says

Indicaciones generales:

- **Lee atentamente** el enunciado e implementa el programa tal como se pi de, con los métodos y requisitos que se especifican.
- El programa, además de ser correcto, debe estar bien comentado. Se valorar án la claridad, la concisión y la eficiencia.
- La entrega se realizará a través del campus virtual, subiendo únicamente el archivo `Program.cs` (generado por MonoDevelop), con el programa completo.
- Los nombres de los participantes se pondrán como comentario al principio de dicho archivo (antes de `using System; namespace ...`, etc).
- El **plazo de entrega** finaliza el día 1 de febrero de 2016.

Simon Says es un juego electrónico creado por Ralph Baer y Howard J. Morrison en 1978. Es un disco con cuatro cuadrantes de colores verde, rojo, azul y amarillo tal como se muestra en la figura:

La mecánica del juego es muy sencilla: el juego de forma aleatoria va iluminando los cuadrantes de colores y generando un sonido (distinto para cada color). Después el usuario debe introducir la secuencia mostrada en el orden correcto, i.e., debe memorizar la secuencia de colores-sonidos. Si lo consigue, la máquina mostrará una secuencia más larga (se incrementa en 1 cada vez), y así sucesivamente. Si falla, el usuario ha perdido la partida.

En esta práctica implementaremos una versión sencilla de este juego. Se solicitará al usuario la longitud máxima de la secuencia a generar y comienza el juego: la máquina genera una secuencia de un solo color que el usuario debe repetir. Si acierta, ampliará esa secuencia con otro color y así sucesivamente hasta alcanzar la longitud máxima (en cuyo caso el jugador gana) o cometer un error (el jugador pierde).

Para almacenar la secuencia utilizaremos un `array` y un contador de jugadas que corresponde a la longitud de la secuencia en cada momento del juego (empieza en 1 y va incrementándose en 1 en cada jugada).

El programa mostrará la secuencia de colores escribiendo en pantalla cada uno de ellos (`RED`, `GREEN`, `BLUE`, `YELLOW`) durante un breve lapso de tiempo (`DELTA`) y dejando entre ellos otro lapso (`LAP`) sin nada en pantalla. Después quedará a la espera para que el usuario repita la secuencia. Para abreviar, el usuario podrá teclear solo iniciales de los colores (`r`, `g`, `b`, `y`). Ambos valores `DELTA` y `LAP` pueden declararse como constantes de la clase (fuera del método `main`), así como el generador de números aleatorios (de esta forma serán *visibles* en cualquier parte del programa):

```
class MainClass
{
  static Random rnd = new Random();
  const int DELTA = 700;
  const int LAP = 400;
```

Deben implementarse **al menos** los siguientes métodos (decidiendo el tipo de los parámetros y la forma de paso de los mismos):

- `pulsa(boton)`: muestra en pantalla el color correspondiente a `boton` (`RED`, `GREEN`, `BLUE`, `YELLOW`) durante un tiempo `DELTA` y después lo borra de pantalla (simula la pulsación de un botón en el juego).

  Para sobreescribir en pantalla (en la esquina superior izquierda, por ejemplo) puede situarse el cursor en una posición dada:

  ```
  Console.SetCursorPosition (0, 0);
  ```

  Además pueden cambiarse los colores del texto y del fondo para hacer el juego más visual:

  ```
  Console.ForegroundColor = ConsoleColor.Black;
  Console.BackgroundColor = ConsoleColor.Red;
  ```

- `muestraSecuencia(secuencia, jugada)`: muestra la secuencia de colores de longitud *jugada*, almacenada en el *array* `secuencia`. Para ello invocará repetidamente al método `pulsa` con el color correspondiente. Tras mostrar cada color dejara un lapso de tiempo *LAP* para facilitar al usuario la memorización de la secuencia.

- `extiendeSecuencia(secuencia,jugada)`: amplía la secuencia de colores almacenada en `secuencia` e incrementa el contador de jugadas `jugada` (que coincide con la longitud de la secuencia).

- `bool recogeSecuencia(secuencia,jugada)`: lee de teclado la secuencia del usuario comprobando que coincide con la secuencia generada por la máquina que viene dada en el parámetro `secuencia`. **En cuanto** el usuario cometa algún error en la secuencia el método devolverá `false`. Si completa correctamente la secuencia (cuya longitud viene en el parámetro `jugada`), devolverá `true`.

El método `main` hará uso de estos métodos para implementar el juego tal cual se ha descrito.

1. Mejoras y extensiones

Opcionalmente se pueden añadir algunas extensiones al juego:

- puede hacerse multijugador. El programa solicitará el número de jugadores que jugarán por turnos, cada uno con una secuencia propia, i.e., se gestionará un *array* por cada uno de los jugadores (un array bidimensional).

- es fácil añadir sonidos al juego utilizando el método `Console.Beep(frequency,duration)`. En la página:

  `http://www.waitingforfriday.com/index.php/Reverse_engineering_an_MB_Electronic_Simon_game#Sound_frequencies_and_timing`

  pueden encontrarse datos precisos sobre los tiempos de retardo `DELTA` y `LAP`, frecuencia de los sonidos, duración de la muestra de colores-sonidos y retardos, etc.