Fundamentos de la programación 1

Grado en Desarrollo de Videojuegos

Examen de convocatoria ordinaria. Curso 25/26

Indicaciones generales:

- Se entregará únicamente el archivo `Program.cs` con el programa pedido. Se proporciona una **plantilla** con el esquema del programa y algunos métodos ya implementados.
- Las líneas 1 y 2 serán comentarios de la forma:
  `// Nombre Apellido1 Apellido2`
  `// Laboratorio, puesto`
- **Lee atentamente el enunciado** e implementa el programa tal como se pide, con los métodos, parámetros y requisitos que se especifican. No puede modificarse la representación propuesta, ni alterar los parámetros de los métodos especificados. No se especifica el modo de paso de los parámetros en los métodos (`out`, `ref`, ...), que debe determinar el alumno.

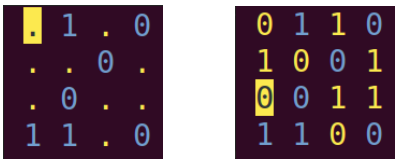
  Pueden implementarse todos los métodos adicionales que se considere oportuno, especificando claramente su cometido, parámetros, etc.
- **El programa debe compilar y funcionar correctamente, y estar bien estructurado y comentado**. Se valorarán la claridad, la concisión y la eficiencia.
- La **entrega:** se realizará a través del servidor FTP de los laboratorios.

  Importante: comprobar el archivo entregado en el puesto del profesor.

Vamos a implementar un puzzle conocido como Takuzu o Binairo, análogo al Sudoku. Se desarrolla sobre una cuadrícula de dimensión $N \times N$ (con N par), que hay que rellenar con 0s y 1s cumpliendo dos reglas:

1. Cada fila y cada columna tienen que contener el mismo número de 0s y 1s.
2. No pueden colocarse más de dos 0s ó 1s consecutivos en una fila ni en una columna.

Inicialmente la cuadrícula está parcialmente ocupada con 0s y 1s fijos, que **no** podrán cambiarse durante el juego. El jugador deberá rellenar las casillas vacías de acuerdo a las reglas anteriores. A continuación se muestra un tablero de 4×4 que utilizaremos como ejemplo más adelante:



A la izquierda se muestra el estado inicial, sin resolver. Los números fijos se dibujan en azul y las casillas vacías con el carácter `'.'` en amarillo; el cursor, que corresponde a la *casilla activa*, está en la esquina superior izquierda. A la derecha se muestra la solución con los números que ha rellenado el jugador en amarillo (es fácil comprobar que cada fila y columna cumplen las dos reglas de arriba).

En nuestra implementación el jugador podrá moverse por la cuadrícula con las flechas de cursor y escribir `'0'` ó `'1'` en las casillas vacías. Podrá también sobreescribir números y borrarlos con la tecla *Espacio*, siempre que no sean casillas fijas (azules). También podrá terminar el juego con la tecla de *Escape*.

Para representar el juego utilizaremos las siguientes variables:

- `tab`: matriz de caracteres que representa el tablero; cada casilla contendrá uno de los valores `'0'`, `'1'` ó `'.'` (vacía).
- `fijas`: matriz de booleanos, del mismo tamaño que `tab`, que determina los números fijos. Las casillas con `true` corresponden a números fijos (azules) que el jugador no puede cambiar; las que tienen `false` (amarillas) sí pueden cambiarse. Esta matriz, una vez rellena, no cambiará durante el juego.
- `fil,col`: posición (fila y columna) de la casilla activa.

En el archivo `Program.cs` se proporciona una plantilla con el tablero de nuestro ejemplo:

```
int N = 4;      // tamaño del tablero de ejemplo
char [,] tab;
tab = new char[N,N] {    // tablero del ejemplo
     {'.','1','.','0'},  // fila 0
     {'.','.','0','.'},  // fila 1
     {'.','0','.','.'},  // etc
     {'1','1','.','0'} };

bool [,] fijas = new bool[N,N]; // matriz de posiciones fijas
int fil, col; // fila y columna de la casilla activa
```

Para desarrollar el juego utilizaremos este ejemplo, pero **el programa deberá funcionar para cualquier otro tablero válido de cualquier tamaño** `N`×`N`. Implementaremos los siguientes métodos (se recomienda seguir el orden establecido e ir comprobando el funcionamiento de los mismos):

- [**1**] `void Inicializa(char [,] tab, bool [,] fijas, int fil, int col)`: rellena la matriz `fijas` con el valor `false` para las casillas que tienen hueco en `tab` y `true` para el resto (las que tienen número). Además sitúa la casilla activa (`fil,col`) en `(0,0)` (esquina superior izquierda).

  Recordemos que el número de filas de `tab` puede obtenerse con `tab.GetLength(0)` y coincide con el número de columnas.

- [**1**] `void Renderiza(char [,] tab, bool [,] fijas, int fil, int col)`: escribe el tablero en pantalla tal como se muestra en los ejemplos. Las casillas fijas se escriben en azul y las otras en amarillo (con `Console.ForegroundColor = ConsoleColor.Blue`, análogo con `...Yellow`). **Se dejará un espacio blanco delante de cada carácter** para proporcionar el aspecto. La casilla activa determinada por `(fil,col)` se mostrará con fondo amarillo.

  Puede utilizarse el método `Console.SetCursorPosition(left,top)` para situar el cursor de escritura en unas coordenadas determinadas de la pantalla antes de escribir (la posición `(0,0)` corresponde a la esquina superior izquierda).

- [**1**] `void ProcesaInput(char c, char[,] tab, bool[,] fijas, int fil, int col)`: procesa el input codificado en `c` dependiendo de su valor:

  - `'u'`, `'d'`, `'l'`, `'r'` (up, down, left, right): actualiza la posición de la casilla activa (`fil,col`) controlando que no se salga de los límites de la cuadrícula (no hace nada si se sale).

  - `'0', '1', '.'`: coloca en `tab` el valor correspondiente, en la posición del cursor, si dicha posición no es fija; no hace nada en otro caso.

- [**1**] `bool TabLleno(char [,] tab)`: comprueba si el tablero `tab` está completo, es decir, si no contiene casillas vacias.

Nótese que el método `leeInput` proporcionado en la plantilla transforma las pulsaciones de flechas de cursor en los caracteres `'l','u','d','r'`; los dígitos `'0','1'` en ellos mismos, el espacio en `'s'` y la tecla de escape en `'q'`.

Con estos métodos ya puede implementarse una primera versión del método `Main` con el bucle principal, que debe, **por este orden**: leer input, procesarlo y mostrar en pantalla el tablero resultante. El bucle terminará cuando todas las casillas estén rellenas (aunque las filas y/o columnas no verifiquen las reglas del juego) o bien cuando el usuario pulse *Escape*.

A continuación vamos a mejorar el juego para verificar si el tablero rellenado por el usuario es solución (si cumple las reglas). Para ello implementaremos los siguientes métodos:

- [**1**] `void SacaFilCol(int k, char [,] tab, char [] filk, char [] colk)`: dado el tablero `tab` y un índice `k` devuelve en `filk` la fila `k`-ésima y en `colk` la columna `k`-ésima.

  Por ejemplo, para el tablero resuelto del ejemplo inicial (el de la derecha), para `k=2` en `fil` obtendríamos el array `['0','0','1','1']` y en col `['1','0','1','0']`.

- [**1**] `bool TresSeguidos(char [] v)`: comprueba si el array `v` tiene tres valores consecutivos iguales.

- [**1**] `bool IgCerosUnos(char [] v)`: comprueba si `v` tiene el mismo número de 0s y 1s.

- [**0.5**] `void MuestraResultado(char [,] tab)`: utilizando los métodos anteriores comprueba si el tablero es correcto. Para cada fila y columna escribe en pantalla si se incumple alguna de las reglas del juego. Por ejemplo, para un tablero incorrecto:

```
0 1 1 0
1 1 0 0
1 0 1 1
1 1 0 0


Tres iguales seguidos en columna 0
No mismo núm de ceros y unos en columna 0
No mismo núm de ceros y unos en columna 1
No mismo núm de ceros y unos en fila 2
No mismo núm de ceros y unos en columna 3
```

  En el caso de que el tablero sea correcto, indicará que efectivamente es solución. Este método se llamará tras el bucle principal para informar al usuario del resultado del juego.

Para poder leer los tableros desde archivo, implementaremos el siguiente método:

- [**1**] `void LeeArchivo(string file, char [,] tab)`: lee del archivo `file` una cuadrícula y crea el archivo `tab` con el contenido. El archivo contendrá el tamaño `N` del tablero en la primera línea, y a continuación una fila del tablero en cada línea, tal como se muestra en el ejemplo `takuzu6` que se proporciona.

Para procesar este archivo se leerá cada una de las líneas en un string y se pasará el contenido a la fila correspondiente. Recordemos que cada caracter individual de un string `s` puede accederse como si fuese un array: `s[i]` denota el i-ésimo carácter de `s`.

El archivo debe contener un tablero en formato de texto correcto (matriz cuadrada, con caracteres `'0'`, `'1'`, `'.'`), con una o más líneas vacías al final. Se proporciona un tablero de ejemplo de 6×6 en el archivo `takuzu6`.

[**1.5**] Por último, completar el método `Main` para que el usuario pueda arrancar el juego con tablero de archivo o bien con el del ejemplo inicial.