Proyecto final FP2

NONOGRAMA – HITO 1

* Codificación de los tableros:

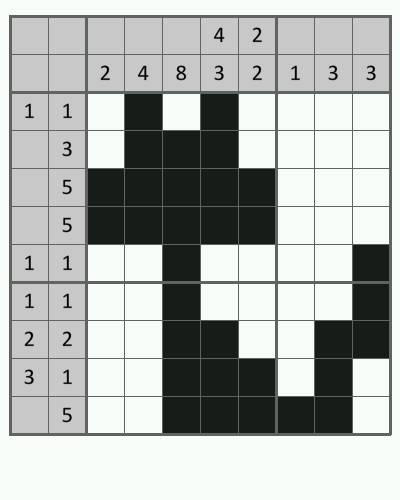
Los tableros irán ordenados uno tras otro en un fichero ‘levels’. Éste estará compuesto por los niveles de la siguiente forma:

*levelX*

*[números de las columnas, separados por comas los de la misma, y por espacios los de diferentes]*

*[números de las filas, separados por comas los de la misma, y por espacios los de diferentes]*

*[espacio en blanco]*

Ejemplo:

level0

2 4 8 4,3 2,2 1 3 3

1,1 3 5 5 1,1 1,1 2,2 3,1 5

* Clase tablero:

Esta clase contendrá todos los métodos relacionados con la información del tablero de juego: su creación, pintado de una casilla, borrado, si está completo, etc.

Tiene los siguientes datos:

// Los cuatro posibles valores de una casilla

enum Casillas { Libre, Coloreada, Tachada};

Casillas[,] tab;

string[] datosFilas, datosColumnas;

int maxDatosFil, maxDatosCol;

Coor jugador;

*Casillas* servirá para dar un valor a cada posición del tablero, *tab*, una matriz de estas casillas, dependiendo del input del usuario. Inicialmente, tab tendrá únicamente el valor *Libre*.

*datosFilas* y *datosColumnas* son dos arrays que contienen los datos de las filas y columnas. Cada posición del array contiene la información de una fila o una columna. Por ejemplo, en el level0 mencionado anteriormente:

datosFilas = [2**|**4**|**8**|**4,3**|**2,2**|**1**|**3**|**3]; ( Cada **|** indica diferentes posiciones del array.)

datosColumnas = [1,1**|**3**|**5|5**|**1,1**|**1,1**|**2,2**|**3,1**|**5];

Los enteros *maxDatosFil* y *maxDatosCol* son valores que se obtienen a partir de los datos de datosFilas y de datosColumnas. Sirven para la representación por pantalla del tablero. Su valor indica el valor máximo de datos que hay en filas y en columnas. Es decir, sirve para saber cuánto espacio libre se debe dejar para poder representar los datos por pantalla.

Por último, la coordenada *jugador,* indica la posición del mismo dentro del tablero.

Hasta el momento, esta clase tiene los métodos necesarios para la representación por pantalla: la constructora Tablero() y Dibuja(), para moverse por el tablero: Mueve() y SiguientePosicion(), y para rellenar las casillas con información: Colorea(), Tacha() y Borra().

* Clase Coors:

Se reutilizará la clase Coors que se aportó en clase para este proyecto.

* Controles del juego:

Moverse: flechas del panel de dirección.

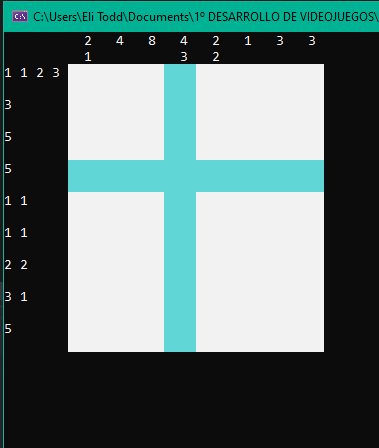
Colorear una casilla: Enter.

Tachar una casilla: X.

Borrar el contenido de una casilla: Retroceso.

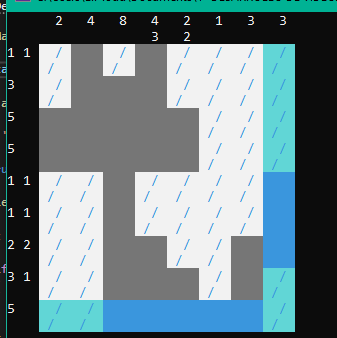
Guardar: G.

Cargar: H.

* La representación en pantalla, se vería así:

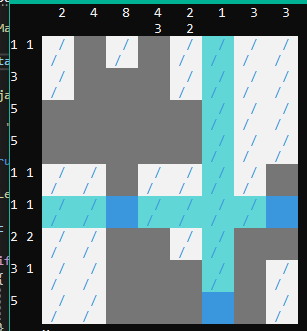
Como se puede observar, las filas y columnas tienen un ancho de 2x2 para que en el caso de que un dato tuviese dos cifras, su representación se realizase con claridad, y sin mezclarse con el resto de datos.

Además de esta manera se ve un tablero más grande que hace más agradable el juego.



Con los 3 métodos en la clase Tablero que se ocupan de cambiar el contenido de la casilla sobre la que se encuentra el jugador, se puede conseguir un tablero completo, que se vería de la siguiente forma:

El tablero se ha completando usando las teclas de ‘X’, y Enter, colocando la información necesaria en cada casilla, proporcionada por los datos.



Además, cuando el jugador está en una fila o columna que contiene un tachado, este se representa con un color azul oscuro, para que se siga viendo la información necesaria del tablero.

* Resolución del tablero:

Para comprobar si un tablero está completo o no, se utilizará un vector de listas.

Inicialmente se crea *vectorDatos*, con los datos de las columnas de la siguiente manera:

2 | 4 | 8 | 4,3| 2,2 | 1 | 3 | 3

Cada posición equivale a una columna del tablero, formada por una lista. Si la lista sólo tiene un elemento, quiere decir que en esa columna solo se requiere un dato, si hay dos, se requieren dos datos, y así sucesivamente hasta el número máximo de posibles datos.

En cada iteración se llamará al método TableroCompletado(), que devolverá true si, al leer el tablero e insertarlo en un nuevo vector de listas: *vectorActual* *,* cada nodo de cada elemento de este vector, coincide con cada nodo de cada elemento de *vectorDatos*. Es decir: va recorriendo *vectorDatos* y comparándolo con *vectorActual,* hasta llegar al último elemento o se encuentre alguna diferencia.

* Mejora del rendimiento

Hasta ahora, el juego funciona correctamente, pero con tableros de mayor tamaño, el tiempo transcurrido entre paso y paso es mayor al deseado, ya que el programa tarda un tiempo en comprobar si está completo o no: aunque el tablero esté vacío o tenga un par de casillas rellenas, tiene que recolectar toda la información para compararla.

Para evitar esto y hacer un programa más fluido, se añade:

int casillasColoreadas = 0, casillasNecesarias = 0;

Vamos a hacer que solo se lleve a cabo la comprobación cuando haya las suficientes casillas rellenas: si hay más o menos de las necesarias, es absurdo comprobar.

casillasNecesarias irá acumulando todos los valores leídos de las columnas, para cuando acabe la lectura saber cuál es el total exacto de casillas que se necesitan para que el tablero sea correcto.

casillasColoreadas se irá modificando con la entrada del usuario, cuando vaya tachando, coloreando o borrando casillas.

De este modo, cuando se llama al método TableroCompletado(), comprobará primero si las casillas coloreadas son (en cantidad) las mismas que las necesarias. En caso afirmativo, seguirá con el resto de comprobaciones. De otro modo, devolverá falso y terminará el método.