- Resolución del tablero:

He hecho un cambio en la forma en que se comprueba la 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.

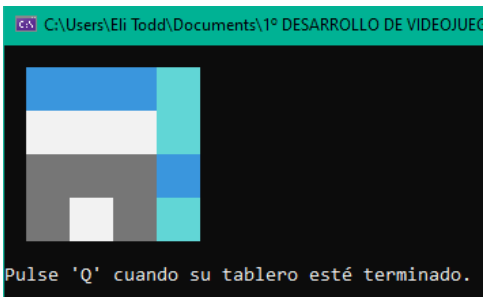
En cada iteración se llamará al método TableroCompletado(), que devolverá true si, al leer el tablero e insertarlo en un nuevo vector de listas: *vectorActual* , cada nodo de cada elemento de este vector, coincide con cada nodo de cada elemento de *vectorDatos*. Es decir: va recorriendo *vectorDatos*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

Anteriormente no se utilizaba sólo una lista, pero de esta manera es mucho más claro.

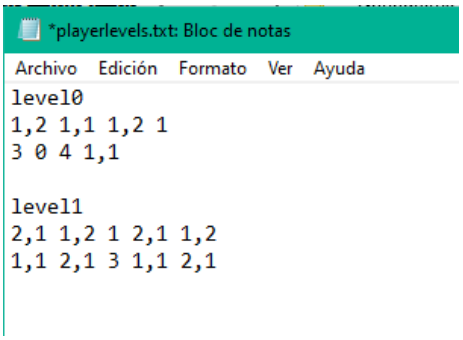
- El jugador puede guardar sus propios tableros

Para la ampliación del juego, he implementado que el jugador pueda crear sus propios tableros y jugarlos más tarde, creando su propio bucle de juego con niveles.

Al seleccionar la opción de crear un nonograma, primero introducirá el número de columnas y filas que quiere que tenga. Tras esto aparecerá un tablero de la dimensión adecuada completamente en blanco, por el que se puede mover el jugador y colorear o tachar las casillas que quiera. Cuando el tablero esté a su gusto, pulsará 'Q' para volver al menú.

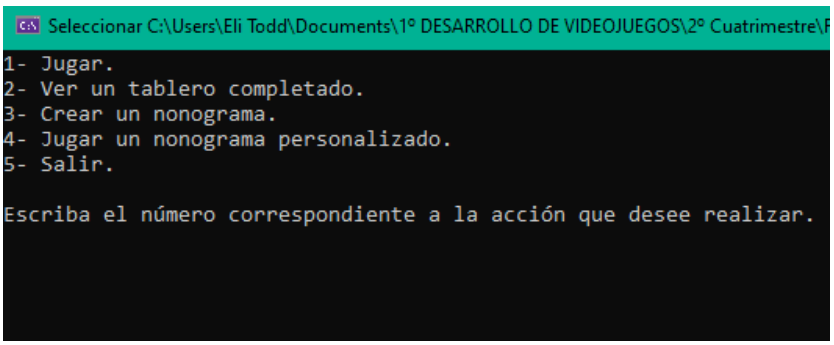


Una vez terminado, el tablero se **lee** (de la misma forma que se comprueba cuando está completo), y se guarda en el archivo *playerlevels.txt*. Queda codificado de la misma forma que en archivo *levels.txt*.



Si el jugador quiere añadir otro tablero, este se añadirá debajo del anterior, de modo que cuando quiera jugarlos irán uno tras otro, al igual que en el modo de juego normal.

- Bucle de juego:



Este es el aspecto que tiene el juego en su menú principal:

1- Jugar reproducirá cada tablero detrás del anterior, hasta que el jugador quiera volver al menú. Si vuelve al menú y luego quiere seguir jugando, vuelve al último tablero sin resolver.
2- Ver un nonograma muestra la imagen de un nonograma ya completado, sin los elementos del jugador.
3- Crear un nonograma es la funcionalidad explicada anteriormente.
4- Jugar un nonograma personalizado comienza un bucle de juego igual al 1 pero con los niveles creados por el jugador.
5- Salir termina la ejecución.

- Extra: 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.