Figura 5.21. Cálculo del índice de la lista para acceder a la celda (sCell_x, sCell_y).

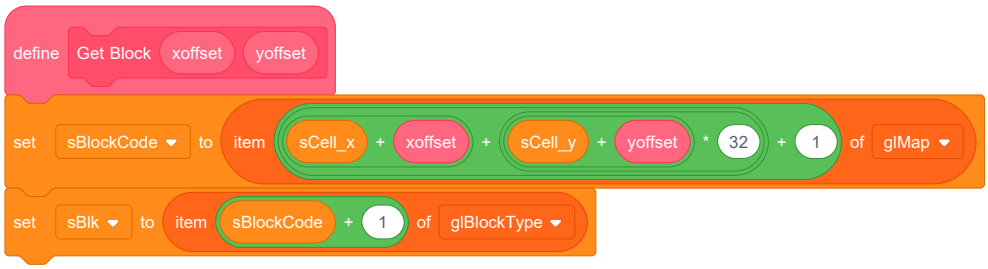

Figura 5.22. Script Get Block de Map.

al elemento del mapa. Luego guarda en la variable auxiliar sBlockCode el código del bloque presente en la celda. Por último, almacena en sBlk el tipo de bloque correspondiente, tomado de la lista glBlockType.

La llamada a Get Block permite tomar decisiones en función del tipo de bloque almacenado en sBlk, en lugar de usar directamente el código numérico en sBlockCode, lo que hace que la implementación más legible.

Complementariamente, para inicializar una celda del mapa utilizaremos el script Set Block de la Figura 5.24. Este script recibe los desplazamientos (xoffset, yoffset) respecto de la celda actual (sCell_x, sCell_y), junto con el tipo de bloque en blocktype.

Código	Tipo de bloque
0	empty
1	stone1
2	stone2
3	stone3
4	rope start
5	rope main
6	rope end
7	sky
8	sand
9	trapdoor

Figura 5.23. Izquierda, correspondencia entre el código de bloque y el tipo de bloque. Derecha, elementos de la lista glBlockType.

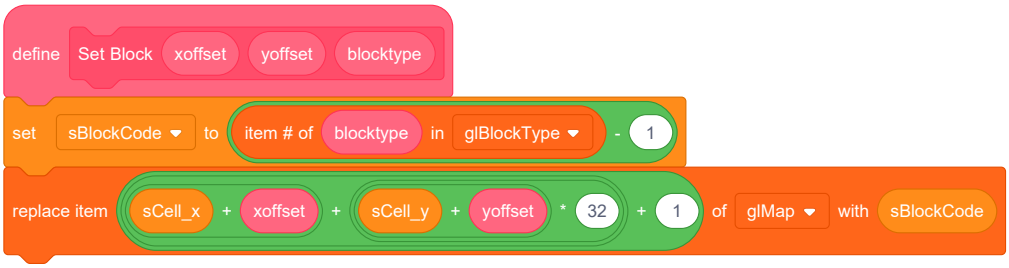

Figura 5.24. Script Set Block de Map.

5.2.4 Inicialización y generación del laberinto

Para crear el mapa utilizaremos tres listas auxiliares. Dos de ellas se emplean para implementar la tabla en las que se guardan las coordenadas X e Y de las celdas con bifurcaciones durante la construcción del laberinto, se denominan slBranch_x y slBranch_y.

Por otro lado, necesitaremos una lista para guardar las direcciones posibles al avanzar un paso, lo que permitirá escoger una de ellas al azar. Para ello, utilizaremos la lista auxiliar slDirections. Cada dirección almacenada es un número entre 1 y 4, que corresponde a uno de los nombres de dirección almacenados en la lista slDirectionName, según la siguiente correspondencia:

1. Arriba –up–.
2. Derecha –right–.
3. Abajo –down–.
4. Izquierda –left–.

Por ejemplo, si la lista slDirections contiene los números 1, 3 y 4, desde la celda actual es posible el movimiento hacia arriba, hacia abajo y hacia la izquierda.

La Figura 5.25 muestra el script que recibe el mensaje Create Map. Si la variable gDebugMap toma el valor true, carga el mapa de depuración en la lista glMap mediante el script Load Debug Map, como veremos a continuación. Además, el script Set Trapdoor Position inicializa la variable gTrapdoor_x, que indica la coordenada X en la que se sitúa la trampilla de salida. Cabe señalar que, en el mapa de depuración, este elemento se encuentra en la columna 25.

Si gDebugMap es false, genera un nuevo mapa, llamando a los scripts Initalize Map, Create Maze, Set Up The Ropes y Set Hatch Position. Los nombres de estos scripts corresponden a las fases explicadas en los apartados anteriores.

El mapa de depuración, almacenado en la lista slDebugMap, corresponde a la Figura 5.1, utilizado durante el desarrollo del programa. Durante las pruebas, resulta conveniente trabajar con un mapa fijo y conocido en lugar de uno generado al azar.

La Figura 5.26 muestra el script Load Debug Map, encargado de copiar la lista que contiene el mapa de prueba, slDebugMap, en la lista del mapa, glMap. La carga se realiza elemento a elemento, ya que Scratch carece de un bloque para duplicar una lista. Observe que no es necesario utilizar una variable como índice: la longitud de glMap en cada iteración actúa como índice en curso.