OBJETIVOS PRINCIPALES

square.c: La primera función es **init_square**, que inicializa cada casilla e inicializa cada atributo de la estructura square, que són la posición y el target. **get_position** retorna la posición de la casilla, **get_target_position** retorna si esa casilla es “snake” o “Ladder”. **set_target** position comprueba que la casilla sea valida y coloca en “x” casilla si es “snake”,” ladder” o nada. **is_ladder**, coloca en las casillas que corresponda a “ladders” y devuelve si hay “ladders” o no. **is_snake** hace lo mismo que **is_ladder** pero con “snake”.

board.c: Primeramente, tenemos la función **init_board**; que inicializa el tablero, y lo hemos hecho mediante las funciones: **set_rows, set_columns, get_size** y hemos hecho un for loop con **init_square**. **get_rows**, da la información de las filas, **get_columns** de las columnas y **set_columns** y **set_rows** configuran número de filas y de columnas del tablero. **Get_size**, retorna el tamaño del tablero, que es filas por columnas y **get_square_at** devuelve la casilla del tablero en una posición específica, por lo tanto, retornara un Square* (puntero).

board_utils.c: check_square_data mira si la información de la casilla es correcta, es decir, que este dentro del tamaño del tablero, que sea “snake” o “ladder”. **read_square_line,** lee las líneas del fichero, mediante un fgets y un sscanf, el cual, nos dará el valor de aciertos que haya hecho y si son los esperados, llama la función **check_square_data** y mediante **set_target_position** coloca las “ladder” y “snakes” correspondientes. Esta función nos retornará el status. **Read_board_file**, con el mismo mecanismo de la función anterior, leeremos las dimensiones del tablero del fichero y si están escritas de la forma correcta y en ese caso llamamos a **init_board**. **Load_board**, carga el archivo y llama **read_board_file**.

state.c: init_state inicializa el estado de la partida, inicializa cada atributo de la estructura State que son un tablero, la posición, un valor que corresponde a la finalización de la partida y una secuencia. **set_current_position,** configura la posición actual. **Get_current_position**, da la informaciónde la posición actual. **set_finished**, configura si la partida está acabada, **is_finished**, da la información si la partida ha acabado o no. **move,** configura los movimientos, y mira la casilla actual, y si es “ladder” o “snake” en caso de que sea una de las dos cambia la posición actual, a la posición que corresponde y en caso de que hayamos llegado a la última casilla, mediante **set_finished,** informa si la partida ha acabado. **Add_step**, añade un paso a la secuencia, y **print_state_secuence** imprime la secuencia del estado.

sequence.c: Inicializa la secuencia, inicializando cada atributo de la estructura Sequence que son: el primer paso (usando la estructura step), el último paso y la medida de la secuencia. La estructura Step tiene los atributos: la posición, el valor del dado y establece un next (el siguiente paso) y un prev (el anterior paso). **Add_step_as_first**, crea una linked list con malloc; añade un paso a la secuencia desde el principio. **Add_step_as_last**, hace lo mismo, pero añadiéndolo al final. **Get_sequence_size** da la información del tamaño de la secuencia, **clear_sequence**, libera todos los elementos de la secuencia, con un for loop, donde los va eliminando todos uno a uno, y **print_sequence**, imprime todos los pasos de la secuencia.

game.c: do_recursive_move mira si se ha alcanzado la profundidad máxima, en ese caso retorna nulo, mira si se ha acabado la partida, en ese caso crea la secuencia que le ha llevado hasta ahí, en caso de que no, llama a **try_dice_values**, para que pruebe otras secuencias mejores y añade el paso realizado a la secuencia. **Try_dice_values,** llama a **do_recursive_move** para cada valor posible del dado (1-6) y selecciona la secuencia más corta de las que hayan sido retornadas, por **do_recursive_move** y la retorna. **Solve,** imprime la mejor secuencia y limpia las listas.

En primer lugar, nuestro grupo sabía que la mejor manera para llevar a cabo este proyecto era distribuir las funciones entre los tres del grupo. Por lo tanto, decidimos que cada uno eligiese alguna función de las diferentes que había en los modulos .c. Cada uno eligió las funciones que pensaba que lo haría correctamente e intentamos asignar proporcionadamente el trabajo entre los tres del grupo, es decir, puede que una persona tuviese más funciones, pero eran más fáciles para programar que otras. De hecho, es evidente que, por ejemplo, la función get_rows es menos compleja que la función move.

Una vez repartido el trabajo, y que todos estuviésemos de acuerdo con el trabajo asignado nos pusimos a trabajar y establecimos una fecha de entrega entre nosotros para comprobar lo que había hecho cada compañero y poner en común las dudas, confusiones, … que teníamos con las funciones asignadas cada uno. Para sincronizar nuestro trabajo, los días marcados por la entrega hemos utilizado la plataforma discord para comunicarnos y compartir las tareas establecidas previamente además de clion para programar el código.

Una vez acabadas todas las funciones, comprobamos el resultado final con diferentes escenarios y no nos ha dado ningún error.

Finalmente, para escribir este “report” hemos seguido la misma ideología que hemos utilizado anteriormente para, distribuir el trabajo en partes iguales. Cada uno explicó (en las reuniones) las funciones implementadas y entre todos escribimos estas últimas dos partes.

IMPRESIONES Y OPINIÓN PERSONAL

Este proyecto nos ha constituido un reto, debido a la distancia social que hemos estado la mayoría de este tiempo. Sin embargo, creemos que nuestro éxito y la hermandad ha contribuido a crear un gran boceto y una experiencia enriquecedora y satisfactoria.

Por otra parte, creemos que la clave de este proyecto ha estado en la comunicación entre los diferentes miembros del equipo, desde el primer día nos organizamos para establecer lo mejor posible las tareas y ponernos a trabajar para poder hacer la tarea a tiempo y de la mejor manera posible.

Es decir, dividimos el trabajo, para que cada uno de los miembros que formaban el equipo pudiese llevar a cabo su o sus partes fácilmente, para continuar asistiendo a las clases online y seguir entregando otros trabajos de otras asignaturas. Las revisiones y reuniones nos han servido para mejorar código, detectar errores y compartir nuestro conocimiento con el resto del grupo. Este proceso justo y mecánico que hemos seguido nos ha llevado a ir rápido, ser limpios y funcionales con el código; y creemos que con este método hemos alcanzado el mejor resultado posible que podríamos alcanzar.

Además, estamos seguros que esta experiencia nos va a ser muy útil para nosotros en los siguientes años, así como también en nuestra carrera académica como profesional, donde la capacidad de trabajar en equipo es más importante que cualquier cosa y los espacios “co-working” están generalizándose en el mundo de las compañías TIC.

Resumiendo, estamos muy satisfechos con el trabajo que hemos hecho y con el beneficio personal que cada uno de los miembros del grupo ha adquirido. Es por esto que esperamos que esto sea perceptible en el resultado del proyecto.

Rubén Vera, Jesús Santos y Eneko Treviño.