Batalla Naval

Programación I

Alumnos: Ignacio Rodríguez, Joaquín Scarone

Institución: UCU

Curso: Desarrollador en Software

Profesor: Rodrigo Dearmas

Maldonado, Uruguay

Fecha: 22/06/2023

Índice

Análisis del problema y posibles soluciones alternativas 3
Diseño, descripción y justificación de la solución propuesta 3
Manual de usuario 4
Manual 5
Conclusiones del trabajo 7

Análisis del problema y posibles soluciones alternativas

En este trabajo grupal se pide crear una simplificación del juego tradicional de estrategia “Batalla Naval”, en el juego original las partidas se llevan a cabo con dos jugadores. Pero en este caso, se simplificará el juego para un jugador, donde se tiene que tratar de adivinar donde están los barcos enemigos y el programa responderá si hubo impacto o no. Para esto, hay que tener en cuenta que en esta versión simplificada los barcos solo ocupan una casilla del tablero.

Básicamente el programa tiene que solucionar la necesidad de jugar una partida de Batalla Naval, pero en solitario, donde el programa responde a las configuraciones ingresadas por el jugador.

La posible solución alternativa que también podría haber sido efectiva para realizar este trabajo sería utilizando la estructura “for” y prints() para crear el tablero, después verificar las coordenadas de los barcos aleatorios con las coordenadas de los disparos ingresados por el jugador. Luego, utilizar la función replace() para indicar si fue impacto o no. Si bien es parecida a la que nosotros realizamos, tiene algunas diferencias en los tipos de datos que utilizamos para guardar la información y funciones utilizadas.

Diseño, descripción y justificación de la solución propuesta

La solución elegida consta de 3 funciones principales, la primera función fue realizada para crear el tablero, este tablero se crea mediante una lista de listas según las dimensiones ingresadas por el jugador. Este tablero utiliza una función de la librería *String*, la cual, según el tamaño ingresado, pega las letras del abecedario en mayúscula en las columnas del tablero. Por ejemplo, si la dimensión ingresada fue 3, la función pegaría ”A B C “ arriba del tablero creado por las matrices. También utilizamos la clase *enumerate ()*, esto sirve para enumerar las filas del tablero desde 1 hasta el tamaño del mismo.

La segunda función principal es la de ubicar los barcos aleatoriamente en el tablero, para lograrlo, creamos una lista vacía donde se guardarán las coordenadas aleatorias en forma de tupla. Para generar las coordenadas, utilizamos la estructura *while* y la función *random.randint* de la librería *Random,* esta función la hemos utilizado anteriormente en el curso por lo que fue fácil de reconocer que podía ser una posibilidad utilizarla para ubicar los barcos. En el *while*, hasta que el largo de la lista no sea igual a la cantidad de barcos ingresados por parámetro, no termina. Un error que surgió en el proceso de realización del programa, era que las coordenadas se repetían, por lo que podía haber dos barcos en una sola coordenada. Para solucionar esto, agregamos un *if* adentro del *while* donde si las tuplas generadas no estaban en la lista, que se agregaran a la misma.

Posteriormente, la ultima función principal creada es la de los disparos, en esta función la columna (letra) que nosotros ingresamos en la configuración del juego para realizar un disparo, es transformada al número de índice del abecedario, Por ejemplo, si nosotros ingresamos la letra “A”, el índice de esta letra va a ser 0. Luego a la fila ingresada, la cual es el número, le restamos 1, esto es para que vaya acorde con el índice. Después, utilizamos un *if* para verificar si la tupla del índice de la columna y la fila por ej.: (0,0) se encuentra en la lista de tuplas de la ubicación de los barcos generados aleatoriamente.

Si se encuentra en la lista de los barcos generados, se cambiará el "-" del tablero a una "X" e imprimirá True, si en la posición de las coordenadas se encuentra un guion, es decir, no esta en la lista de los barcos generados, se imprimirá False y se cambiará el "-" por un "o". Como *else* del *if* indicamos al usuario que ya ha disparado en ese lugar, aunque se pierde un turno de igual manera. En la misma función también imprimimos en pantalla el tablero ya modificado, esto lo hacemos de igual forma que en la primera función.

Esta solución fue elegida ya que era la más sencilla y lógica dentro de lo complejo que se podía ver realizar el trabajo, sobre todo los barcos generados aleatoriamente fueron fáciles de relacionar con las coordenadas a ingresar por el usuario, además que como sugerencia se encontraba esta forma de implementar el tablero. Si bien tuvimos algunas complicaciones con los parámetros que necesitábamos poner en el programa y como llamar a las funciones de una manera ordenada, logramos encontrar soluciones, además que pudimos informarnos sobre ciertas funciones que podrían facilitarnos el trabajo y que nos pueden servir para un futuro.

Al final de la partida, se mostrará un apartado de Resultados el cual te indica los disparos efectuados, los enemigos acertados y las coordenadas de los disparos efectuados.

Manual de usuario

Es fundamental aclarar que las configuraciones de la partida serán ingresadas por el jugador, pero con ciertos límites descritos en la consigna del trabajo. Estas configuraciones a ingresar son: Dimensiones del tablero (3 – 10), Cantidad de barcos, Nivel de dificultad (Fácil, Intermedio, Difícil) y las coordenadas de disparo las cuales quieres disparar. El numero de disparos es asignado según el nivel de dificultad que el usuario elige, si eliges el nivel Fácil, tendrás un 70% del total del tablero, si eliges Intermedio, tendrás un 50%, y si eliges Difícil tendrás un 30%.

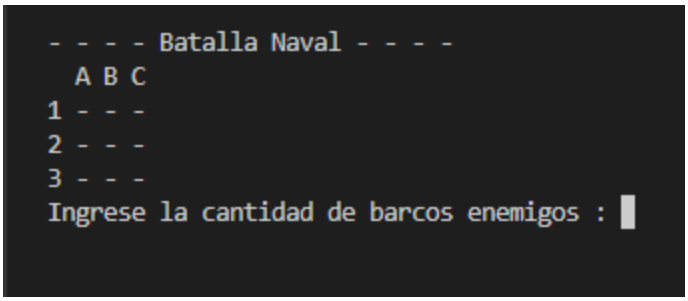
Cabe indicar que, si alguna de estas configuraciones es ingresada incorrectamente por el jugador, surgirá un error en el programa el cual te indicará de nuevo lo que tenías que haber ingresado o simplemente se corte el programa con la función *raise ().*

El código del programa está comentado por si el usuario necesita comprender lo que hacen algunas líneas de código o que fue utilizado en las mismas.

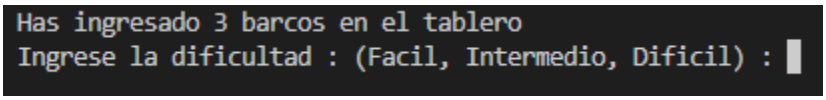
Manual

1- Ingresamos las dimensiones del tablero, las dimensiones tienen que ser de 3 a 10 inclusive.

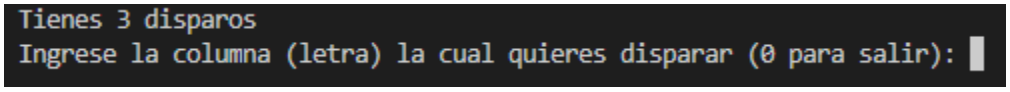
2- Se crea el tablero con las dimensiones ingresadas, ahora el usuario tiene que ingresar la cantidad de barcos que quiere. La cantidad debe ser menor o igual a la raíz cuadrada de total de casillas del tablero. Por ej.: Si elegimos de dimensiones 3, no puede haber más de 3 barcos.



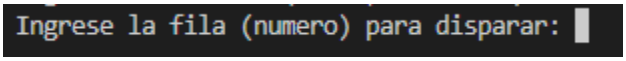
3- Se indica la cantidad de barcos ingresados, ahora se tiene que ingresar la dificultad del juego. Recuerda que tiene que ser Facil, Intermedio o Dificil.



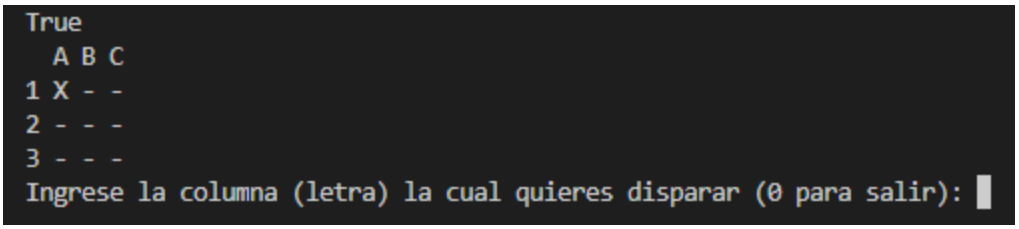
4- Se indicará la cantidad de disparos que tienes, además te pedirá que ingreses una letra la cual quieres disparar. En este caso, como el tablero es de 3, el usuario tiene como opciones A, B o C.



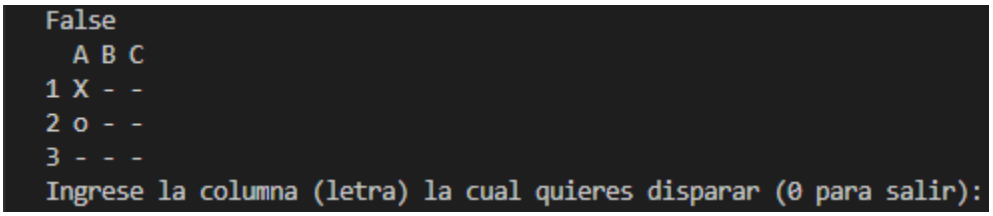
5- Una vez ingresada la letra, se le pedirá la fila al usuario, es decir, el número. En este caso tienes como opción 1,2 y 3.



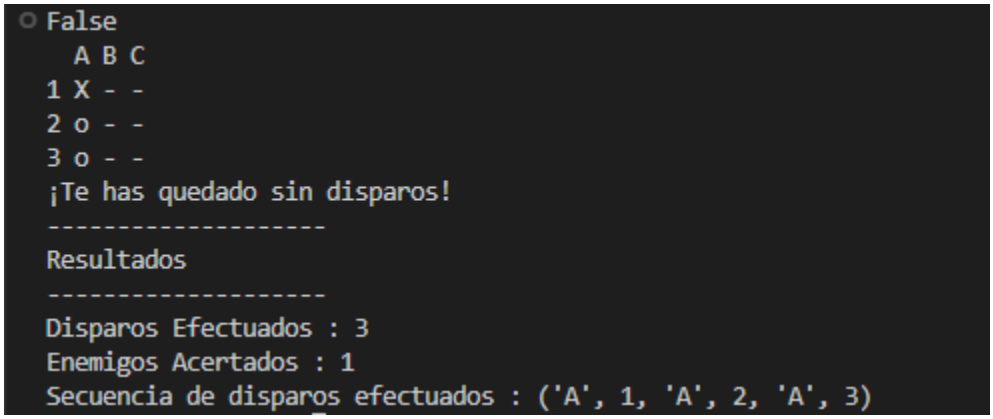
6- Una vez ingresado el número, se imprimirá el tablero indicando con una “X” e imprimiendo True si hubo impacto o con una “o” imprimiendo False si no, en este caso se ha destruido un barco. También se pedirá que repitas el proceso hasta que te quedes sin disparos.



7- Aquí en el siguiente disparo efectuado se puede ver una demostración de como sería cuando el usuario falla.



8- Aquí se utiliza el siguiente disparo, como en este caso el usuario solo tenía 3 disparos ya que eligió la dificultad difícil, se termina la partida, con una sección de Resultados donde indica lo siguiente :



Conclusiones del trabajo

Realizar este trabajo fue una muy buena experiencia ya que nos aportó bastante conocimiento nuevo y como implementar en un programa tal vez un poco mas complejo, otras funciones y formas de hacer algo que tal vez previamente no habíamos visto. Es por esto que también nos incentivó a buscar información nueva, tratar de comprenderla y poder utilizarla a nuestro favor, además de que nos puso en un ambiente de lo que hace un desarrollador para algún trabajo por ejemplo, o simplemente estar en un entorno más serio, haciendo un programa que tiene un proceso más largo y una cantidad más amplia de líneas de código que lo que veníamos haciendo en otros programas o ejercicios.

Tal vez, implementaríamos una forma de poder jugar contra la máquina, aunque esto ya sería mas complejo y llevaría mas tiempo, pero creemos que es una buena forma de seguir mejorando la tarea propuesta, y que sea un poco más entretenido el juego en sí.

En cuanto a lo realizado, tal vez cambiaríamos algunas cosas para que sea un poco más familiar con el usuario, ya que en la mayoría de configuraciones a ingresar por el usuario, si el usuario ingresa incorrectamente algún dato, el programa se termina. Es por esto, que el manual al usuario ayuda, ya que está previsto para que los datos sean colocados correctamente y que no falle el juego.

La construcción de esta Batalla Naval nos brindó diversos conocimientos acerca de Python y nos dio espacio a aplicar todo lo que aprendimos durante los últimos meses. Aprendimos sobre como trabajar en equipo, organizarnos y aprender mutuamente uno del otro.