**Programación**

****

**Trabajo Final**

**Integrantes:** Tomás Bulgarelli (2404340) y Joaquin Bonino (2406840).

**Profesor:** Cayuela Pablo.

**Fecha de emisión:** 10/6/2024

**La Batalla Naval**

**Introducción:**

La Batalla Naval es un clásico juego de estrategia en el que dos jugadores disponen sus flotas de barcos en una cuadrícula e intentan hundir los barcos del oponente adivinando las coordenadas correctas. Cada jugador coloca sus barcos en posiciones estratégicas y luego, en turnos alternados, intenta "disparar" a las coordenadas donde cree que se encuentran los barcos enemigos. El objetivo es hundir todos los barcos del adversario antes de que él haga lo mismo.

Elegimos desarrollar la Batalla Naval como nuestro proyecto final porque nos pareció una opción entretenida y desafiante que nos permitiría aplicar de manera práctica los conceptos y habilidades adquiridos a lo largo de la materia.

**Desarrollo:**

Primero, identificamos las funciones principales que debía tener el código para asegurar un funcionamiento eficiente y fluido del juego. Estas funciones clave son: mostrar el tablero en pantalla para que los jugadores puedan visualizar el estado actual del juego; verificar si una posición es válida para colocar un barco, asegurando que no haya solapamientos ni posiciones fuera del tablero; colocar un barco en el tablero de manera correcta; manejar el disparo del usuario, registrando los intentos y actualizando el tablero en consecuencia; manejar el disparo de la computadora, simulando la inteligencia del oponente; y, finalmente, verificar si quedan barcos en el tablero para determinar si el juego ha terminado.

Para llevar a cabo el proyecto, utilizamos varias herramientas fundamentales de programación. Usamos condicionales para múltiples propósitos: verificar si hubo impacto en un disparo, corroborar si el input del usuario es correcto y determinar qué información mostrar al usuario en cada momento del juego. Estas estructuras de control nos permitieron manejar la lógica del juego de manera clara y precisa, asegurando una interacción adecuada y continua entre el usuario y el programa.

Además, los ciclos fueron esenciales para el funcionamiento del juego. Utilizamos ciclos para asegurarnos de que el juego se ejecutara mientras hubiera barcos en el tablero, para volver a pedir un dato en caso de un input incorrecto y para recorrer las matrices que representan los tableros de juego. Las matrices fueron cruciales, ya que todo lo relacionado con el tablero y sus datos, incluyendo la ubicación de disparos y barcos, se manejó a través de ellas. También empleamos librerías como `iostream` para la entrada y salida de datos, `ctime` para la generación de números aleatorios y `windows.h` para la función `Sleep()`, permitiendo pausas temporales en el juego. Los arreglos nos ayudaron a imprimir el tablero y convertir letras a números, mientras que las funciones, en su mayoría de tipo void, nos permitieron separar el código y mantenerlo bien estructurado y organizado.

**Funciones:**

La función encargada del disparo del bot: fue diseñada para simular un adversario automatizado que realiza disparos en el tablero del jugador. Para lograr esto, generamos coordenadas aleatorias para la fila y la columna donde el bot dispararía, utilizando la función `rand()` junto con el tamaño del tablero para asegurar que las coordenadas generadas estuvieran dentro de los límites válidos.

Una vez obtenidas las coordenadas, convertimos estos valores numéricos en un formato más legible para el usuario, utilizando arrays que mapean números a letras y viceversa. Esto nos permitió mostrar al usuario en qué posición exacta disparaba el bot, por ejemplo, en la coordenada "B5".

Después de determinar la posición del disparo, verificamos el contenido del tablero del usuario en esa ubicación. Si el disparo del bot impactaba en un barco, marcábamos esa posición con una 'H' para indicar un impacto y mostrábamos un mensaje informando al usuario sobre el acierto del bot. Si el disparo caía en el agua, se marcaba con una 'A' y se notificaba al usuario que el bot había fallado.

La función encargada de generar la matriz de la computadora: fue diseñada para posicionar los barcos del bot de manera automática y aleatoria en su propio tablero. Este proceso es esencial para que el bot tenga una configuración inicial de barcos contra la cual el jugador humano pueda jugar.

Primero, determinamos la longitud de los barcos que se colocarán. Utilizamos barcos de diferentes longitudes, definidos previamente en el código. Luego, para cada barco, generamos coordenadas aleatorias para la fila y la columna iniciales donde se intentará colocar el barco, utilizando la función `rand()` para obtener estos valores dentro del rango del tamaño del tablero.

Además de las coordenadas iniciales, generamos una dirección aleatoria para cada barco. Las direcciones posibles eran "arriba", "abajo", "izquierda" y "derecha", seleccionadas usando un número aleatorio que se mapeaba a una de estas direcciones.

Una vez obtenidas las coordenadas y la dirección, verificábamos si era posible colocar el barco en esa posición sin que se saliera del tablero ni se solapara con otro barco ya colocado. Esta verificación se realizaba mediante la función `esPosicionValida`, que revisaba si todas las casillas necesarias para el barco estaban disponibles.

Si la posición y dirección eran válidas, colocábamos el barco en la matriz del bot utilizando la función `colocarBarco`, que marcaba las casillas correspondientes con 'B' para indicar la presencia de un barco. Este proceso se repetía hasta que todos los barcos estuvieran colocados correctamente en la matriz.

La función encargada de colocar los barcos en el tablero: tanto del usuario como del bot, es fundamental para la preparación inicial. La función `colocarBarco` se diseñó para tomar las coordenadas de inicio, la longitud del barco y la dirección en la que se desea colocar el barco, y luego actualizar el tablero con la ubicación del barco.

Primero, la función recibe las coordenadas iniciales (fila y columna), la longitud del barco y la dirección . Dependiendo de la dirección especificada, la función itera a través del número de casillas que ocupa el barco, actualizando las posiciones correspondientes en la matriz del tablero para marcar la presencia del barco.

Por ejemplo, si la dirección es "arriba", la función decrementa la fila en cada iteración, colocando una parte del barco en cada posición hasta completar su longitud. Si la dirección es "abajo", incrementa la fila. Para las direcciones "izquierda" y "derecha", la función modifica la columna de manera similar, decrementándola o incrementándola respectivamente.

Durante este proceso, cada casilla del tablero donde se coloca una parte del barco se marca con un 'B', indicando la presencia de un barco en esa posición. Este método asegura que el barco se coloque de manera continua y recta en la dirección especificada, sin interrupciones ni solapamientos con otras partes del tablero.

La implementación de esta función es crucial porque garantiza que los barcos se coloquen correctamente en el tablero, respetando las reglas del juego y evitando errores. Al manejar tanto las coordenadas como la dirección de colocación, la función `colocarBarco` facilita una preparación precisa y ordenada del tablero, tanto para el usuario como para el bot, asegurando una base sólida para el desarrollo del juego.

La función `esPosicionValida`: juega un papel crucial en la colocación de barcos. Su propósito es verificar que una posición y dirección especificadas para un barco sean válidas, asegurándose de que el barco no se salga del tablero ni se solape con otros barcos ya colocados. Esta verificación es esencial para mantener la integridad del juego y evitar configuraciones inválidas.

Primero, la función recibe la matriz del tablero, las coordenadas iniciales (fila y columna), la longitud del barco y la dirección en la que se desea colocar el barco. Dependiendo de la dirección proporcionada ("arriba", "abajo", "izquierda" o "derecha"), la función realiza diferentes verificaciones.

Para la dirección "arriba", la función verifica si el barco cabe en el tablero sin salirse por la parte superior. Esto se realiza comprobando si la fila inicial menos la longitud del barco más uno es mayor o igual a cero. Si esta condición no se cumple, la posición es inválida. Además, la función itera a través de las casillas necesarias en esa dirección para asegurarse de que todas estén vacías (marcadas con 'X'). Si alguna casilla ya contiene un barco, la posición es inválida.

El mismo tipo de verificación se aplica para las otras direcciones:

* Para "abajo", se verifica que la fila inicial más la longitud del barco no exceda el tamaño del tablero.
* Para "izquierda", se asegura que la columna inicial menos la longitud del barco más uno sea mayor o igual a cero.
* Para "derecha", se comprueba que la columna inicial más la longitud del barco no exceda el tamaño del tablero.

En cada caso, además de verificar los límites del tablero, la función también recorre las casillas necesarias para asegurar que todas estén disponibles para colocar el barco.

Si todas las condiciones se cumplen, la función retorna `true`, indicando que la posición es válida para colocar el barco. De lo contrario, retorna `false`, indicando que la posición es inválida.

La función `mostrarMatriz`: se diseñó para presentar el estado actual del tablero a los jugadores de manera clara y legible. Esta función es esencial para que los jugadores puedan visualizar la disposición de sus barcos y los disparos realizados, facilitando la toma de decisiones durante el juego.

Primero, la función recibe dos parámetros; la matriz del tablero que se desea mostrar y un booleano `ocultar` que indica si los barcos deben ser ocultados (por ejemplo, para el tablero de la computadora).

La función comienza imprimiendo un encabezado de números que representan las columnas del tablero. Esto se logra mediante un bucle que recorre el rango del tamaño del tablero y muestra los números del 1 al 10, seguidos por un salto de línea.

Luego, la función recorre cada fila del tablero utilizando un bucle for. Para cada fila, primero imprime la letra correspondiente (de 'A' a 'J') para etiquetar las filas. Después, recorre cada columna de la fila actual y decide qué mostrar en función del valor de `ocultar` y el contenido de la matriz:

- Si `ocultar` es verdadero y la casilla actual contiene un barco ('B'), se imprime una 'X' en lugar de la 'B'. Esto asegura que los barcos del oponente no sean visibles para el jugador.

- Si `ocultar` es falso o la casilla no contiene un barco, se imprime el contenido real de la casilla (ya sea 'X' para vacío, 'B' para barco, 'H' para impacto o 'A' para agua).

Después de procesar todas las casillas de una fila, se imprime un salto de línea para pasar a la siguiente fila. Este proceso se repite para todas las filas del tablero, asegurando que el tablero completo se muestre en una cuadrícula organizada y fácil de entender.

La función `tiro\_errado\_frase`: fue creada con el propósito de añadir dinamismo y humor al juego de Batalla Naval, brindando una experiencia más entretenida para los jugadores cuando realizan un disparo fallido. Esta función no solo informa al jugador que su disparo no impactó en ningún barco, sino que también lo hace de una manera variada y divertida.

Para lograr esto, la función utiliza la generación de números aleatorios mediante `rand()` para seleccionar una de varias frases humorísticas predeterminadas. Esto asegura que cada vez que el jugador falle un disparo, reciba un mensaje diferente, manteniendo el juego interesante y menos repetitivo.

La función se implementa de la siguiente manera:

1. Inicialización Aleatoria: Se llama a `srand(time(NULL))` para inicializar el generador de números aleatorios con la hora actual, garantizando que las frases seleccionadas sean diferentes en cada ejecución del juego.

2. Selección de Frases: Se genera un número aleatorio entre 1 y 7, que se utiliza para seleccionar una de las siete frases humorísticas disponibles.

3. Impresión de la Frase: Utilizando una estructura `switch`, la función imprime una frase específica según el número aleatorio generado. Estas frases están diseñadas para ser humorísticas y agregar un toque de diversión al informar al jugador sobre el fallo de su disparo.

Ejemplo de frases utilizadas:

- "¡Agua! Pero tranquilo, los peces te lo agradecen."

- "¡Casi! Al menos sabes dónde no está el barco."

- "Ese tiro fue directamente al... ¡agua!"

Esta función es parte de un conjunto de funciones similares que añadimos al juego para manejar diferentes eventos con un toque de humor y variedad. Por ejemplo, `tiro\_acertado\_frase` proporciona mensajes cuando el jugador acierta un disparo, y `usuario\_gana\_juego` ofrece frases festivas cuando el jugador gana. Cada una de estas funciones utiliza un enfoque similar de generación aleatoria de frases para mantener la dinámica del juego.

La función `disparoUsuario` : es una de las piezas centrales, responsable de manejar la interacción del usuario cuando realiza un disparo en el tablero de la computadora. Esta función se asegura de que el jugador pueda ingresar sus coordenadas de disparo, valida esas coordenadas y luego actualiza el estado del tablero en consecuencia.

1. Entrada de Coordenadas: La función comienza pidiendo al usuario que ingrese la columna en la que desea disparar. Esto se hace mediante un bucle que sigue solicitando una columna hasta que el usuario ingrese un valor válido (entre 1 y 10). Una vez validada la columna, el valor ingresado se ajusta para que sea compatible con el índice de la matriz (restando uno para convertirlo a un índice basado en cero).

2. Validación de la Fila: Similar al paso anterior, la función pide al usuario que ingrese una fila, representada por una letra (de 'A' a 'J'). La entrada se valida comparándola con un array de letras. Si la letra ingresada es válida, se convierte a su índice correspondiente en la matriz utilizando la función `letrasAnumeros`.

3. Verificación del Disparo: Con las coordenadas de fila y columna validadas, la función verifica el contenido de la matriz de la computadora en la posición indicada. Si la posición contiene un barco ('B'), se marca como hundido ('H') y se llama a la función `tiro\_acertado\_frase` para informar al jugador sobre su éxito con un mensaje variado y humorístico. Si la posición no contiene un barco, se marca como agua ('A') y se llama a la función `tiro\_errado\_frase` para proporcionar un mensaje sobre el fallo del disparo.

La función `posicionarBarcosUsuario` : permite al jugador colocar sus barcos en el tablero al inicio del juego. Este proceso interactivo asegura que los barcos se posicionen según las reglas del juego, validando las entradas del usuario para evitar errores.

La función comienza solicitando al jugador que ingrese las coordenadas de columna y fila, así como la dirección del barco. Primero, pide un número de columna entre 1 y 10, validando que la entrada esté dentro de este rango y ajustándola para que sea compatible con los índices de la matriz. Luego, solicita una letra para la fila (A-J), convirtiéndola a un índice numérico mediante la función `letrasAnumeros`. Por último, el jugador ingresa la dirección del barco, que debe ser "arriba", "abajo", "izquierda" o "derecha".

Con las coordenadas y la dirección validadas, la función verifica si el barco se puede colocar en la posición deseada sin salirse del tablero ni solaparse con otros barcos. Esta verificación se realiza llamando a la función `esPosicionValida`. Si la posición es válida, se llama a `colocarBarco` para actualizar la matriz del tablero con la posición del barco.

Después de colocar cada barco, la función `mostrarMatriz` actualiza y muestra el estado del tablero al usuario, proporcionando una retroalimentación visual inmediata. Este proceso se repite para cada barco, asegurando que todos los barcos se coloquen correctamente antes de que comience la partida.

La función `hayBarcosRestantes`: es fundamental para determinar si el juego debe continuar o si uno de los jugadores ha ganado. Su propósito es verificar si todavía quedan barcos sin hundir en el tablero de un jugador.

La función recorre toda la matriz del tablero, buscando cualquier casilla que contenga un barco ('B'). Si encuentra al menos una casilla con un barco, retorna `true`, indicando que aún quedan barcos por hundir y, por lo tanto, el juego debe continuar. Si no encuentra ninguna casilla con un barco, retorna `false`, señalando que todos los barcos han sido hundidos y el juego ha terminado.

Este proceso se realiza utilizando dos bucles anidados que recorren cada fila y cada columna de la matriz. La función compara el contenido de cada casilla con el valor 'B'. Si encuentra un 'B', retorna `true` de inmediato. Si recorre toda la matriz sin encontrar ningún 'B', retorna `false`.

La función `hayBarcosRestantes` es esencial porque permite al juego verificar de manera eficiente y precisa si algún jugador ha ganado. Sin esta verificación, no sería posible determinar el estado del juego y gestionar adecuadamente el flujo de la partida.

Por último, la elección de la dificultad en el juego se implementa modificando la longitud de los barcos. Al ajustar la longitud de los barcos, se altera la probabilidad de que el número aleatorio generado para el disparo del bot impacte en un barco del usuario. Con barcos más largos, el bot tiene más posibilidades de acertar un disparo, lo que aumenta la dificultad del juego. Esta modificación no solo hace que el juego sea más desafiante para el jugador, sino que también reduce el tiempo total de la partida, ya que los barcos más largos son más fáciles de localizar y hundir, acelerando el proceso de juego. La implementación de esta opción de dificultad permite a los jugadores personalizar su experiencia y enfrentarse a desafíos ajustados a su nivel de habilidad.

**Flujo de la Función Main:**

A continuación, se detallan los pasos y procesos llevados a cabo en esta función, que garantizan la correcta ejecución del juego:

Inicialización y Configuración del Juego

En primer lugar, se definen las constantes y variables globales necesarias para el tamaño del tablero y la longitud de los barcos. Esto incluye la constante `tamano` que define el tamaño del tablero como una matriz de 10x10 y las variables `longitudBarco1` y `longitudBarco2` que definen la longitud de los barcos a utilizar. Luego, se inicializan las matrices `matrizUsuario` y `matrizComputadora` que representan los tableros del usuario y de la computadora respectivamente. Estas matrices son fundamentales para almacenar la ubicación de los barcos y el estado de los disparos a lo largo del juego.

Presentación Inicial

El juego comienza con la presentación de un mensaje inicial que explica las reglas básicas y los símbolos utilizados en el tablero. Se indica que `A` representa el agua, `B` un barco, `H` una parte de un barco hundido, y `X` una incógnita. Además, se menciona la longitud de los barcos disponibles para que el usuario tenga claridad sobre las reglas del juego. A continuación, se solicita al usuario que ingrese la palabra "YA" para confirmar que está listo para comenzar el juego. Este paso asegura que el usuario entiende las instrucciones antes de proceder.

Selección de Dificultad

Una vez que el usuario confirma que está listo para comenzar, se le solicita que elija el nivel de dificultad del juego. Puede seleccionar entre dos niveles: `1` para fácil y `2` para difícil. Dependiendo de la elección, se ajustan las longitudes de los barcos para aumentar o disminuir la dificultad del juego. En el modo fácil, ambos barcos tienen una longitud de 8, mientras que en el modo difícil, los barcos tienen longitudes de 5 y 3. Esta variabilidad en la longitud de los barcos permite que el usuario experimente diferentes niveles de desafío.

Posicionamiento de los Barcos

Con la dificultad seleccionada, el siguiente paso es mostrar el tablero del usuario en su estado inicial y permitir que el usuario posicione sus barcos. La función `posicionarBarcosUsuario` es llamada para manejar este proceso interactivo. El usuario ingresa las coordenadas y la dirección para cada barco, y la función verifica que las posiciones sean válidas antes de actualizar la matriz del tablero con la ubicación de los barcos. Una vez que los barcos del usuario están posicionados, se genera automáticamente la matriz del tablero de la computadora con sus barcos colocados aleatoriamente usando la función `generarMatrizComputadora`.

Inicio del Juego

Con ambos tableros listos, el juego comienza oficialmente. Se muestra el tablero de la computadora al usuario, pero con los barcos ocultos (`X` en lugar de `B`). El juego se desarrolla en turnos alternados entre el usuario y la computadora. En cada turno, el usuario realiza un disparo ingresando las coordenadas donde cree que se encuentra un barco enemigo, y la función `disparoUsuario` se encarga de validar el disparo y actualizar el tablero en consecuencia. Después del disparo del usuario, se verifica si quedan barcos en el tablero de la computadora. Si no quedan barcos, el usuario gana y se muestra un mensaje de victoria.

Turno de la Computadora

Si el usuario no ha ganado, es el turno de la computadora. La computadora realiza un disparo en el tablero del usuario utilizando la función `disparobot`, que genera coordenadas aleatorias y actualiza el tablero del usuario en consecuencia. Se muestra el estado actualizado del tablero del usuario y se verifica si quedan barcos en su tablero. Si no quedan barcos, la computadora gana y se muestra un mensaje de derrota.

Finalización del Juego

El ciclo del juego continúa alternando entre los turnos del usuario y de la computadora hasta que uno de los dos hunde todos los barcos del oponente. El juego termina cuando se cumple esta condición, mostrando el mensaje correspondiente de victoria o derrota.

**Para Cerrar:**

Uno de los desafíos principales al desarrollar este código fue la necesidad de actualizar continuamente la estructura de datos de la matriz

con cada entrada de usuario o proceso del juego. Cada disparo realizado, cada barco colocado y cada cambio en el tablero requería que la matriz se actualizara para reflejar el estado actual del juego. Esto implicaba no solo almacenar correctamente los datos ingresados por el usuario, sino también asegurarse de que las verificaciones y actualizaciones fueran precisas y eficientes para mantener la integridad del juego. Manejar estas constantes actualizaciones de la matriz fue crucial para garantizar que el juego funcionara correctamente y proporcionara una experiencia fluida para el jugador.

En cuanto a posibles mejoras, una de las principales sería hacer que las elecciones de disparo del bot fueran más lógicas y estratégicas. Por ejemplo, si el bot acierta un disparo, los siguientes disparos podrían enfocarse en las casillas adyacentes, aumentando la probabilidad de encontrar y hundir el resto del barco. Esto se podría lograr generando números aleatorios dentro de una zona reducida alrededor del impacto, en lugar de en todo el tablero, lo que haría al bot un oponente más desafiante y realista. Otra mejora significativa sería desarrollar una interfaz gráfica para el juego, lo que no solo haría que la experiencia de usuario fuera más atractiva visualmente, sino que también podría facilitar la interacción del jugador con el juego, haciendo que sea más intuitivo y accesible.

Bulga 1-2-3-4-9-10-13-15-17-19

Joaco 5-6-7-8-11-12-14-16-18-20