fecha: 22 de marzo del 2021

Proyecto Programado

Programación Imperativa en C: 4 en línea.

Preparado por: Cristopher Zúñiga jiménez, Jorge Víquez y alonso Obando Carmona

Tecnológico de Costa Rica

Requerimientos de software

Contenido

[Diseño de la solución 2](#_Toc67317674)

[Datos 2](#_Toc67317675)

[Interacción con la persona usuaria 3](#_Toc67317676)

[Descripción de las estructuras de datos 3](#_Toc67317677)

[Estructuras de datos 3](#_Toc67317678)

[Funciones 4](#_Toc67317679)

[Lecciones aprendidas 6](#_Toc67317680)

# Diseño de la solución

## Datos

Matriz [7][7]: es donde vamos a almacenar el tablero y modificar sus elementos utilizando la representación de caracteres “ ”, “O”, “X” para mostrar las fichas de los jugadores.

Árbol: Conjunto de nodos que permitirán la solución al problema mediante el valor de máximos y mínimos para implementar dentro de la solución

Nodo: Elemento base del árbol que adquiere un valor para clasificar según su nivel dentro del árbol además de cada uno tener una copia del tablero. (sea Max o mini).

Jugador: La única información relevante para el juego del jugador es el nombre, por lo tanto, es la única información solicitada.

## Interacción con la persona usuaria

Una vez iniciado el programa se pedirá al usuario el nombre con el que desea que se le llame durante todo el tiempo de vida del programa, ya que no posee modo multijugador se irá directamente al menú principal el cual mostrará únicamente 2 opciones: Jugar y salir.

Si el usuario decide jugar entonces el juego comenzará una vez el primero turno sea decido mediante un sistema aleatorio, durante cada turno del jugador será mostrado el estado actual del tablero de 7x7, el movimiento realizado por la IA en el turno anterior (Si posible) y un menú con 2 opciones: Colocar ficha y rendirse.

Si el usuario decide colocar ficha entonces se le solicitará un valor de la columna donde desea colocar la ficha y por otro lado si decide rendirse será enviado al menú principal.

Este ciclo de cambio de turnos entre computadora seguirá hasta que el estado del juego no pueda cambiar más, se encuentre un ganador o el jugador se rinda.

Una vez devuelta en el menú principal la única opción restante del jugador es salir del juego lo cual hace exactamente lo que su nombre indica, despedir al usuario y cerrar la aplicación.

## Descripción de las estructuras de datos

### Estructuras de datos

Array Multidimensional: Este será un array de 7x7 utilizado para representar el tablero del juego y guardar las jugadas, además este tablero será la base lógica del juego y las funciones partirán del análisis de este.

ÁrbolMinMax: Árbol creado para guardar todos los posibles movimientos de los jugadores el cual va a ser analizado para calcular la mejor jugada posible. El algoritmo MinMax cuenta de un nivel inicial que es cuando el juego inicia y luego pasa a una función sucesora que es usada para definir todos los posibles movimientos del jugador en ese momento y finaliza con un estado terminal que decide cual es el movimiento ganador determinado por un valor de tipo euristico donde a mayor valor mayor es la posibilidad de ganar del cpu, dicho algoritmo esta implementado para simular una inteligencia artificial que jugara contra la persona a la hora de ejecutar el programa.

### Funciones

Solicitar nombre: Una consulta al jugador que se ejecuta inmediatamente después del inicio del sistema la cual únicamente pedirá el nombre con el cual el sistema llamará al jugador.

Menú principal: Se mostrará un menú principal del juego que tendrá 2 opciones “Iniciar juego” y “Salir”. Tendrá un switch que permitirá al usuario decidir que desea realizar.

Juego: Es llamada desde la opción “jugar” del menú principal, durante esta función varias va a ser llamadas, “Imprimir tablero”, “TurnoJugador”,” Rendirse” serán o pueden ser llamadas durante el turno del jugador si es el turno de la IA será llamada la función “TurnoCPU”. Al inicio de la función se mostrará el estado actual del tablero y luego un menú con un switch para que el usuario tome la decisión que desea las opciones serían: “Colocar Ficha” y “Rendirse”.

Elegir tirada: Función principal donde realizaremos las llamadas respectivas para ejecutar las funcionalidades del algoritmo MINIMAX, con el objetivo de devolver una lista de jugadas buenas y escogiendo al azar de entre la categoría de mejores jugadas para realizar un movimiento por parte de la computadora.

Crear Nodo: Es donde realizamos la inicialización de los valores del Struct NODO, acá vamos a asignar los respectivos elementos a tomar en cuenta y el estado actual del tablero para próximamente ser analizado.

Crear árbol: Función llamada por “Elegir tirada”, una vez se crear el nodo raíz este en enviado a la función de crear árbol, donde se convertirá en un árbol n-ario donde n es la cantidad de columnas no vacías. El árbol se crea en niveles de 2 en 2, donde el primero nivel es la jugada de la CPU y el segundo es una jugada del jugar.

Valorar hojas: Las ultimas hojas del árbol se valoran con 1000 si gana la CPU o -1000 si gana el jugador y este número decrecerá o aumentara con respecto al nivel del árbol para así ayudar a determinar la mejor jugada posible.

Mini max: Función que interpreta el árbol de forma que este empeza desde la asignación de valor del padre con respecto a sus arboles hijos y mediante recursividad se recorre hasta llegar a la parte de las ultimas hojas, donde usamos un identificador par e impar para determinar los niveles MIN y MAX tomando así el menor o mayor de los nodos hijos respectivamente y asignándolo al padre para seguir la ruta mas adecuada a la hora de recorrer el árbol y seleccionar el movimiento del jugador

Colocar Ficha: Función que es llamada desde el menú del juego con la opción de “Colocar ficha”, esta pedirá el jugador que decida en que columna del tablero se introducirá la ficha mediante un número. Una vez el sistema valido que la colocación de ficha es válida y nadie ha ganado el juego, el turno acaba.

Verificar Victoria (CPU y jugador): Función que será llamada después de cada colocación exitosa, como el nombre lo indica para ganar en un juego de 4 en línea debes ser el primero en lograr conectar 4 fichas del mismo color desde cualquier dirección por lo tanto cada vez que se coloque una ficha nueva se debe revisar en 7 de las 8 direcciones (omitiendo arriba) que posea al menos 3 posiciones entre ella y el fin del tablero.

Validar colocación: Función que se ejecutará automáticamente siempre que el usuario quiera colocar una ficha en el tablero, esta revisará únicamente que en la columna seleccionada aún haya campo para colocar la ficha, el programa no ayudará para nada el usuario y no avisará de malas dediciones al jugador.

Imprimir tablero: Función que se activa automáticamente al inicio de cada turno durante el juego, se utilizará una combinación de prints acomodados de manera específica para formar el tablero de la manera más visual posible, variando los colores dependiendo del jugador que coloco la ficha.

Rendirse: Función que será llamada desde el menú del juego, después de la confirmación de que el jugador si desea rendirse, será enviado al menú principal.

Salir: Función llamada desde el menú principal del juego, únicamente despedirá al jugador dándole las gracias por jugar y luego la aplicación se cerrará.

## Lecciones aprendidas

Durante la elaboración del proyecto decidimos como equipo la implementación de técnicas de desarrollo ágil para optar por la búsqueda de una solución y la creación de código de un entregable a corto plazo, ante la dificultad que planteaba el diseño de la interfaz y la adicción de un árbol MINIMAX al juego como simulador de un contrincante trabajamos bajo la metodología de Pair Programming donde destinamos a dos de tres desarrolladores a la implementación y entendimiento del árbol y un recurso para el desarrollo de la estructura del juego y su interfaz, gracias a esto se nos permitió trabajar en paralelo ambos requerimientos y acoplarlos al finalizar el proyecto. A lo largo del proyecto se nos presentaron dificultades tales como a la hora de tomar decisiones y mostrar los elementos gráficamente donde optamos por falta de tiempo dejar que el encargado del despliegue de la información de la pantalla tomara la decisión final después de una lluvia de ideas. Logramos comprender que muchos de los problemas se pueden abstraer y formar diferentes estructuras, como lo es como el caso del árbol Minimax que simula un jugador y escoge la mejor opción a jugar a nivel matemático. Pese a que no es un jugador natural quien realiza el movimiento logramos implementar una estructura para automatizar una tarea, aunque bien descubrimos que este tipo de soluciones se pueden implementar para Estados finitos o que no superen la capacidad computacional de nuestros dispositivos.