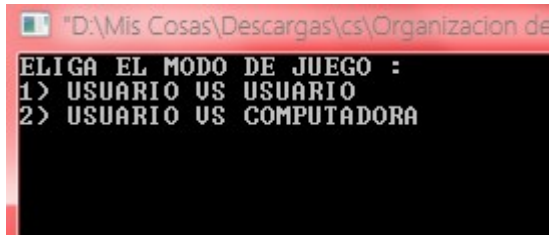


DOCUMENTACION

ESTRUCTURAS Y CONTENIDO

En este proyecto enfoco en la creacion de un juego en lenguaje “C” conocido por muchos el “TA-TE-TI” basado ,con el fin de entrettemiento o para pasar el rato ,y como se desarrollo para que sea facil de jugar , puede ser jugado tanto por niños ,jovenes o adultos .

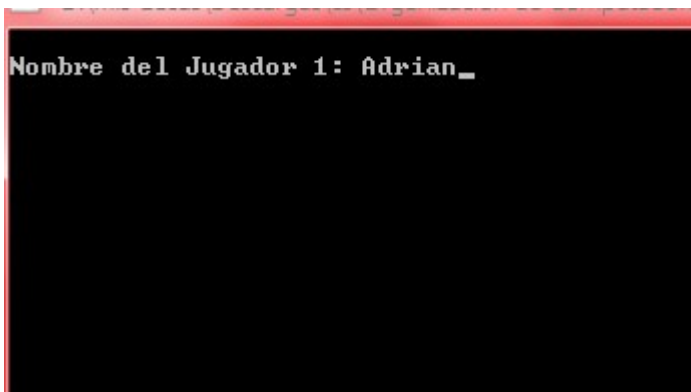
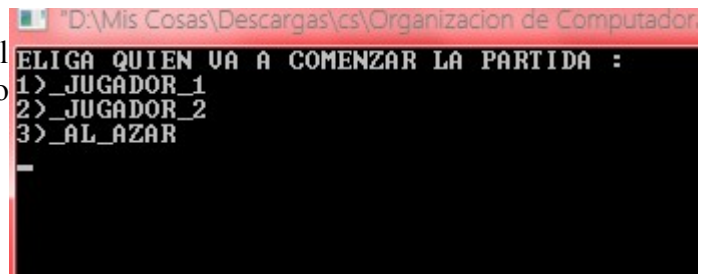
Pautas que se siguieron para el proyecto :



USUARIO_VS_COMPUTADORA.

- 1) Brinda 2 modos de juego
USUARIO_VS_ USUARIO o

- 2) El jugador elige quien comenzara el juego JUGADOR_1 ,JUGADOR_2 o AL_AZAR .

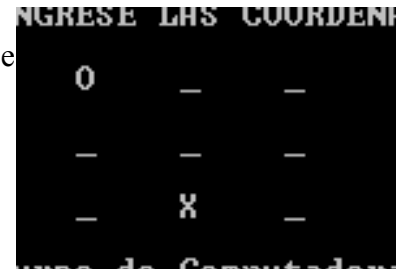


- 3) El jugador puede ingresar su nombre tanto el nombre del jugador como el de su openente .

- 4) El sistema le debe pedir las coordenadas al jugador de turno ;las cuales deben pertenecer al intervalo [1,3] ej : 1 3 ,en este caso el usuario pide colocar su simbolo (X o O) en la fila 1 ,columna 3.



- 5) Mostrar la tabla “TA-TE-TI ” cada vez que cambie de turno



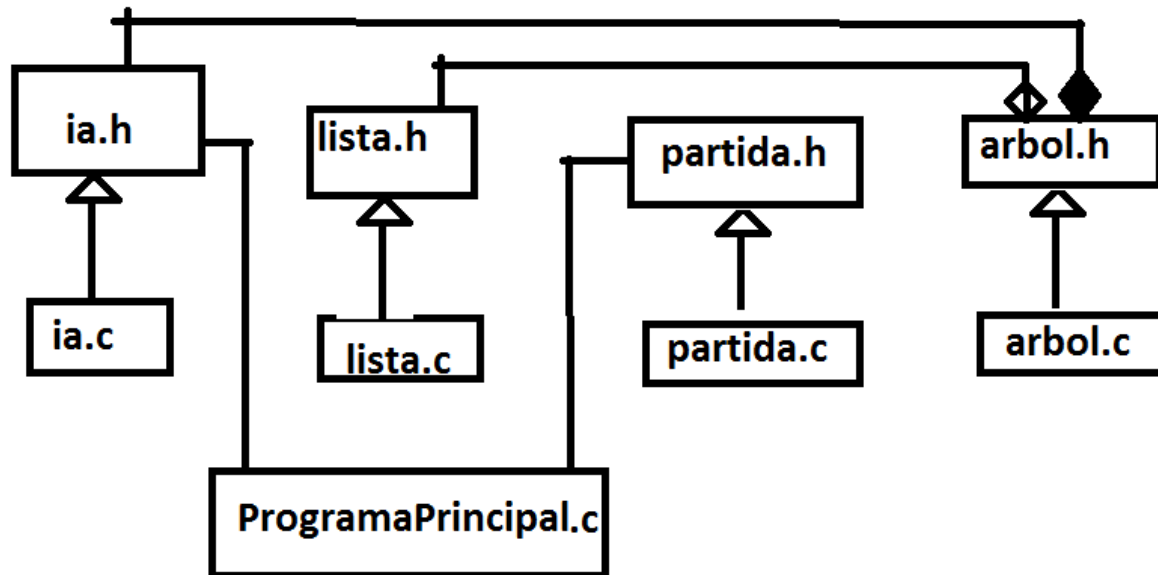
LIMITACIONES :

- 1) Solo puede jugar una vez ,luego de acabar el juego ya sea luego de PERDER , GANAR o EMPATE ,si quiere jugar otra partida debe ejecutar el programa de nuevo.
- 2) Si no elige correctamente entre las opciones que se le da,se le pedira que ingrese correctamente ,hasta que cumpla lo pedido.

HERRAMIENTAS USADAS :

- 1) archivos (.h) que nos brindo la catedra .Las cuales se implemnetaron
- 2) como plataforma se uso CodeBlock para la creacion del proyecto.

ARQUITECTURA DEL SISTEMA



MODULOS UTILIZADOS : A continuacion realizaremos una descripcion resumida de cada uno de los modulos utilizados en este proyecto ,de acuerdo en orden en el que fueron implementados

LISTA :

`crear_lista(tLista * l)` : Inicializa una lista vacía.

`l_insertar(tLista l, tPosicion p, tElemento e)` : Inserta el elemento E, en la posición P, en L.

`l_eliminar(tLista l,tPosicion p,void (*fEliminar)(tElemento))` :Elimina la celda P de L. El elemento almacenado en la posición P es eliminado mediante la función fEliminar parametrizada.

`l_destruir (tLista * l, void (*fEliminar)(tElemento))` :Destruye la lista L, eliminando cada una de sus celdas. Los elementos almacenados en las celdas son eliminados mediante la función fEliminar parametrizada.

`tElemento l_recuperar(tLista l, tPosicion p)`; Recupera y retorna el elemento en la posición P.

`tPosicion l_primera(tLista l)` Recupera y retorna la primera posición de L.

`tPosicion l_siguiente(tLista l, tPosicion p)` Recupera y retorna la posición siguiente a P en L

`tPosicion l_anterior(tLista l, tPosicion p)` Recupera y retorna la posición anterior a P en L

`tPosicion l_ultima(tLista l)` Recupera y retorna la última posición de L

`tPosicion l_fin(tLista l)` Recupera y retorna la posición fin de L

ARBOL:

`crear_raiz(tArbol a, tElemento e)` Crea la raíz de A

`tNodo a_insertar(tArbol a, tNodo np, tNodo nh, tElemento e)` : Inserta y retorna un nuevo nodo en A

`a_eliminar(tArbol a, tNodo n, void (*fEliminar)(tElemento))` : Elimina el nodo N de A

`a_destruir(tArbol * a, void (*fEliminar)(tElemento))` Destruye el árbol A, eliminando cada uno de sus nodos

`tElemento a_recuperar(tArbol a, tNodo n)` Recupera y retorna el elemento del nodo N

tNodo **a_raiz**(tArbol a) : Recupera y retorna el nodo correspondiente a la raíz de A
tLista **a_hijos**(tArbol a, tNodo n) : Obtiene y retorna una lista con los nodos hijos de N en A
a_sub_arbol(tArbol a, tNodo n, tArbol * sa) : Inicializa un nuevo árbol en *SA ,

PARTIDA :

nueva_partida(tPartida * p, **int** modo_partida, **int** comienza, **char** * j1_nombre, **char** * j2_nombre) :
Inicializa una nueva partida, indicando:

int nuevo_movimiento(tPartida p, **int** mov_x, **int** mov_y) : Actualiza, si corresponde, el estado de la partida considerando que el jugador al que le corresponde jugar, decide hacerlo en la posición indicada (X,Y)

finalizar_partida(tPartida * p) Finaliza la partida referenciada por P, liberando toda la memoria utilizada

IA :cabe aclarar que este archivo es donde la computadora “planea” su proximo movimiento .

crear_busqueda_adversaria(tBusquedaAdversaria * b, tPartida p) Inicializa la estructura correspondiente a una búsqueda adversaria,

proximo_movimiento(tBusquedaAdversaria b, **int** * x, **int** * y) Computa y retorna el próximo movimiento a realizar

destruir_busqueda_adversaria(tBusquedaAdversaria * b) :Libera el espacio asociado a la estructura correspondiente para la búsqueda adversaria.