PONTIFICIA UNIVERSIDAD CATOLICA MADRE Y MAESTRA (PUCMM)

Facultad de Ciencias e Ingeniería

DEPARTAMENTO:

Ingeniería de sistemas y computación.

ASIGNATURA:

Fundamentos de programación (ISC105, Grupo No. 6053)

TEMA:

Proyecto Final

PRESENTADO POR:

Disraely Peralta Uceta (ID: 10140077)

PRESENTADO A:

Carlos Alfredo Camacho Guerrero

FECHA DE ENTREGA:

07/11/2020

SITUACION DE APRENDIZAJE:

Una compañía que es propietaria de un sistema en Linux le ha contratado para resolverle el problema que enfrenta con la edición de texto que es digitado en el editor de texto Vi. Su labor consistirá en crear una aplicación que se encargue de formatear un párrafo de acuerdo con ciertos estándares que han sido suministrados por la compañía contratante. Su programa estará leyendo un archivo tipo texto, indicando la ruta por la entrada estándar y generando el resultado en un nuevo archivo especificado por el usuario.

OBJETIVOS:

- Dividir un problema en bloques, para solucionar con funciones.
- Trabajar con los arreglos y cadenas de caracteres.
- Trabajar con arreglos bidimensionales, matrices.
- Trabajar con estructuras.
- Trabajar con punteros.
- Trabajar con memoria dinámica.
- Trabajar con archivos.
- Separar las funciones en archivos header.

ENLACE DEL PROYECTO: https://github.com/disraelyp/ISC105-ProyectoFinal

REQUERIMIENTOS:

Cada estudiante realizará en el lenguaje C, la solución del problema planteado en este documento, los mismo debe utilizar las librerías estándar del lenguaje. Para la realización del problema planteado NO PUEDEN utilizar la función GOTOXY(). Controlar los errores lógicos del programa. Para dar cierre a nuestro curso y conociendo las reglas del juego de las damas, estamos requiriendo realizar un programa que permita a dos personas jugar el juego de Las Damas. Cuando el programa inicia, se debe indicar el nombre del primer jugador controlando las piezas negras y el 2do jugador controlando las blancas, alternando los turnos en ese mismo orden, dando la opción a cualquier jugador rendirse una vez le toque su turno. Para realizar los movimientos de la partida, se presentará el tablero de las

damas en formato 8 x 8 más la información corresponde al nombre de las filas (8,7,6,5,4,3,2,1) y las columnas (a,b,c,d,e,f,g,h) necesarios para que los jugadores puedan indicar sus movimientos basados en la notación algebraica utilizada en el Ajedrez, método oficial para registrar las partidas que en nuestro caso será simplificada.

Las reglas que estaremos implementando en el sistema están basadas en el sistema americano simplificado las cuales son:

1. La cantidad de peones son 12 para cada jugador.
2. Los peones se colocan en las casillas negras del tablero, utilizando las 3 primeras filas.
3. Los peones se mueven de forma diagonal derecha o izquierda entre las casillas negras, no se permite mover hacia atrás.
4. Para comer un peón enemigo, se debe estar libre la próxima casilla negra, de lo contrario no se permite realizar el movimiento.
5. Es obligatorio comer las fichas enemigas que están en dicha condición.
6. Si la posición de las fichas enemigas lo permiten se debe permitir continuar jugando. si luego de comer una ficha en su nueva casilla de desplazamiento se encuentra otra ficha para comer.

El peón que llega a la última fila del tablero, será coronado, permitiendo desplazamiento en todo el diagonal hacia delante o atrás. Luego de comer una pieza, se comporta igual a un peón. Al momento de coronar un peón, termina el turno de dicho jugador.
Se gana el juego cuando:

- Un jugador coma todas las fichas del contrincante.
- Exista una condición de ahogado, es decir, no puede mover ninguna ficha.
- La rendición de manera explicita por parte de jugador.
- Una partida puede finalizar empate, si ambos jugadores lo acuerdan

DESARROLLO:

El programa esta basado en una matriz 8 x 8, y se van intercambiando el contenido de sus posiciones. Al inicio se despliega un menú con diversas opciones;

1. Iniciar una nueva partida de damas, primero se inicia la función de nueva partida que crea y almacena en estructuras los datos de los jugadores y genera la matriz tablero en base a estas estructuras. El tablero también estos compuestos de estructuras llamadas bloques, que estas pueden o no contener los peones. Los peones están identificados con una posición que varia y un color que es estático. Al momento del jugador iniciar su turno tiene 3 opciones, jugar, rendirse o solicitar empate, la parte de jugar es la más compleja debido a que existen dos tipos de movimientos:
    1. Movimiento normal, este se realiza primero verificando si las posiciones ingresadas por el usuario son correctas y verificado si la posición final esta disponible. Si todas estas condiciones se cumplen se efectúan los movimientos y se almacena en los registros pertinentes.
    2. Movimiento para eliminar, este movimiento es igual que el anterior con la diferencia de que el movimiento. Este elimina un peon del oponente.

    Las únicas maneras de finalizar una partida es cuando alguno de los dos jugadores se queda sin ficha, si no hay mas movimientos disponibles, si uno se rinde o si llegan a un empate acordado;
2. Ver récord de jugadores y Ver notación algebraica de partidas guardadas. Este apartado se basa en que a medida que se va jugando partidas de damas se van llenando dos archivos, uno de notaciones que almacena todas las jugadas que se hace en cada partida y otro archivo que va almacenando un registro de todas las partidas que se realizan. El de las notaciones se llena con cada cambio de posición hasta que finaliza el juego, a diferencia del de registro que se llena cada vez que termina un partido de manera que existe un jugador y un ganador, esto significa que si se declara empate no se almacena registro, pero si sus notaciones.
3. Salir y guardar todo.

ENLACE DEL PROYECTO: https://github.com/disraelyp/ISC105-ProyectoFinal

CONCLUSIONES:

Esta práctica fue todo un éxito debido a que se cumplieron todos los objeticos, y además con todos los requerimientos de estas. Los conocimientos utilizados para la realización de esta asignatura fueron:

- Punteros.
- Arreglos bidimensionales de punteros.
- Estructuras.
- Enum.
- GIT.
- Manejo de archivos.
- Creación y uso de archivos .C
- Manejo de memoria dinámica.
- Uso del código ASCII.
- Funciones de la librería string.h
- Manejo de variables tipo Char y Char*
- Funciones de ordenamiento como QSOR
- Estrategias como dividir un problema en bloques.
- Manejo de errores.

ANEXO DE FUNCIONES Y COMENTARIOS ACERCA DE SU FUNCIONAMIENTO.

- void agregar_notacion(const char*, int const id, jugador*, jugador*, posiciones, posiciones); // CREA UNA ESTRUCTURA DE TIPO NOTACION Y LA AGREGA
- void lista_notaciones(const char*); // PRESENTA UNA LISTA DE NOTACIONES
- int nuevo_id(const char*); // CREA UN NUEVO ID
- int cantidad_id(const char*); // CUENTA LA CANTIDAD DE ID EN UN ARCHIVO
- void leer_archivo(const char*, int id); // PRESENTA UNA LISTA DE RECORD DE JUGADORES
- void escribir_archivo(const char*, notacion_algebraica); // AGREGA UN REGISTRO EN EL ARCHIVO DE RECORDS
- int cantidad_notaciones(const char*); // CANTIDAD DE NOTACIONES
- FILE* abrir_archivo(const char*, char *funcion); // ABRIR UN ARCHIVO
- void cerrar_archivo(FILE*); // CERRAR UN ARCHIVO
- int verificar_archivo(const char*); // VERIFICAR SI UN ARCHIVO EXISTE
- int cantidad_registros(const char*); // CANTIDAD DE RECORDS
- void modificar_registros(const char*, record_partidas, int posicion); // ACTUALIZA UN RECORD
- void escribir_registro(const char*, record_partidas); // CREA UNA ESTRUCTURA DE TIPO RECORDS
- void recorrer_registro(const char*, record_partidas); // RECORRE EL ARCHIVO DE RECORDS
- void agregar_registro(const char*, jugador*, int, int); // AGREGA UN REGISTRO EN EL ARCHIVO DE NOTACIONES
- int funcion_ordenamiento(const void*, const void*); // FUNCION DE ORDENAMIENTO PARA EL QSORT
- void leer_registro(const char*); // IMPRIME LA LISTA DE RECORD
- jugador captura_jugador(const int); // CREA UNA ESTRUCTURA TIPO JUGADOR
- void creacion_jugadores (jugador*, jugador *); // CREA DOS ESTRUCTURAS TIPO JUGADOR CON DISTINTO COLOR
- int verificar_char_int (const char*); // VERIFICA SI UN CHAR PUEDE CONVERTIRSE A UN INT
- int captura_int(const int, const int); // CAPTURA UN VALOR TIPO INT
- int captura_charSN(); // CAPTURA UNA RESPUESTA TIPO CHAR
- int char_int (const char*); // CONVIERTE UN ELEMENTO TIPO CHAR A TIPO INT

- int verifica_entrada(const char*); // VERIFICIA LA ENTRADA DE NOTACIONES ALGEBRAICA
- int contar_fichas(const tablero, const jugador*); // CUENTA LAS FICHAS DE UN JUGADOR
- int verificar_propietario(const tablero, const jugador*, const posiciones); // VERIFICA EL PROPETARIO DE UNA FICHA
- int turno(tablero*, jugador*, jugador*, const int, char*, const int, const char*); // TURNO DE UN JUGADOR
- void turno_movimiento(tablero*, jugador*, jugador*, char*, const int, const char*); // MOVIMIENTO DE UNA FICHA
- void nueva_partida(const char*, const char*); // NUEVA PARTIDA
- tablero generar_tablero(const jugador*, const jugador*); // GENERA UNA ESTRUCTURA DE TIPO TABLERO
- peon creacion_peon(const jugador*, const int, const int); // GENERA UNA ESTRUCTURA DE TIPO PEON
- bloque creacion_bloque(const jugador*, const int, const int); // GENERA UNA ESTRUCTURA DE TIPO BLOQUE
- void cambio_posicion(tablero*, const jugador*, const jugador*, const posiciones, const posiciones, const char*, int const, const char*); // CAMBIA DE POSICION DOS FICHAS
- void eliminar_posicion(tablero*, const jugador*, const jugador*, const posiciones, const posiciones, const char*, int const, const char*); // CAMBIA DE POSCION DOS FICHAS Y ELIMINA UNA
- int verificar_posiciones(const tablero, const posiciones, const posiciones); // VERIFICA SI DOS POSICIONES SON ACCESIBLES POR UNA FICHA
- int verificar_movimiento(tablero*, jugador*, jugador*, const char*, const char*, const int, const char*); // VERIFICA SI EL MOVIMIENTO ES VALIDO
- void reajustar_tablero(tablero*); // REAJUSTA EL TABLERO Y LAS ESTADISTICAS
- int recorrer_tablero(const tablero, const jugador*, const jugador*); // RECORRE TODAS LAS FICHAS DEL TABLERO
- int verificar_eliminar(const bloque**, const jugador*, const posiciones); // VERIFICA SI SE PUEDE ELIMINAR UNA FICHA
- int parametros_eliminar(const tablero, const posiciones, const posiciones); // VERIFICA SI UNA ELIMINACION CUMPLE TODOS LOS PARAMETROS ANTES DE REALIZARLA
- int verificar_ahogado(const tablero); // VERIFICA SI HAY MOVIMIENTOS DISPONIBLES PARA LS FICHAS

- posiciones extraer_posiciones(const char*); // EXTRAE LAS POSICIONES UNA CADENA DE TEXTO
- void imprimir_posiciones(const posiciones); // IMPRIME POSCICIONES
- int posiciones_jugables(const posiciones); // CALCULA SI UNA POSICION ES JUGABLE
- int posiciones_diagonales(const posiciones, const posiciones); // CALCULA SI DOS POSICIONES ESTAN EN UNA DIAGONAL
- posiciones posicion_intermedia(const posiciones, const posiciones); // CALCULA LA POSICION INTERMEDIA DE DOS FICHAS
- posiciones calcular_posicion(const int); // CALCULA LA POSCION EXACTA DE UNA UBICACION
- int ubicaciones_externas(const posiciones); // CALCULA SI UNA POSCICION ES UNA UBICACION EXTERNA
- int ubicaciones_iniciales(const int); // CALCULA SI UNA POSCICION ES UNA UBICACION INICIAL
- int ubicaciones_jugables(const int); // CALCULA SI UNA POSCICION ES UNA UBICACION JUGABLE
- int calcular_ubicacion(const posiciones); // CALCULA UNA UBICACION
- int calcular_cordenadaX(const posiciones); //CALCULA LAS CORDENADAS DE UNA POSICION
- int calcular_cordenadaY(const posiciones);//CALCULA LAS CORDENADAS DE UNA POSICION
- void imprimir_bloque(const bloque); // IMPRIME UN BLOQUE
- void imprimir_tablero(const tablero); // IMPRIME EL TABLERO
- void imprimir_color(const color); // IMPRIME EL COLOR
- void imprimir_representacion(const color); // IMPRIME LA PRIMERA LETRA DE UNA REPRESENTACION