Algoritmos y programación II

Trabajo practico N°1

"Cuatro en línea"

Alumno: Franco Pomi

Padron:106590

DNI: 42587223

Curso: P. Calvo

Índice

Cuestionario

1) ***¿Que es un Debug?***

• Un Debug es una herramienta que tiene el uso de encontrar y eliminar los errores que pueden cometerse en un programa.

2) ***¿Que es un "Breakpoint"?***

• Un Breakpoint es una sentencia que coloca un punto de quiebre dentro de un programa, permitiendo interrumpir/detener el programa en ese punto para así poder facilitar el hecho de detectar errores.

3) ***¿Que es "Step Into", "Step Over" y "Step Out"?***

• Cuando se está debuggeando/depurando un programa y se necesita saber qué es lo que está pasando, paso a paso, dentro de una función se utiliza la herramienta del **Step Into**, mientras que el **Step Over** se utiliza para saltar este estilo de procedimiento, salteando la función y ahorrando tiempo, dejándome pasar a analizar otra función que todavía no sé si está funcionando correctamente. A diferencia de los anteriores el **Step Out** sirve para encontrar automáticamente la última instrucción de la función en que uno se encuentra, dejándonos situado fuera de ella y así poder continuar con el debugging.

Manual del Programador

Introducción:

En este manual se explicará el funcionamiento del programa diseñado para la simulación de un juego de 4 en línea en el modo de jugador contra jugador.

Archivos:

Para una mejor organización y facilidad de modificación y entendimiento se subdividió este programa dentro de 6 archivos que todos se encuentran interrelacionados entre sí en forma de librerías dentro de un mismo proyecto. Estos archivos son:

• **Principal.cpp** – este archivo es utilizado como el centro del programa ya que es de donde se inicializa debido a que es en donde se encuentra nuestro main ()

• **Interacciones.cpp / Interacciones.h** – en estos archivos se encuentran casi todas las funciones que interaccionan con el/los usuarios y con las decisiones que este/estos tomen.

• **Reglas.cpp / Reglas.h** – dentro de estos podemos encontrar las funciones que están relacionadas con las reglas, es decir, las formas de ganar, perder o empatar este juego.

• **Constantes.h** – un archivo simple donde se encuentran todos los valores fijos (contantes) utilizadas en el programa para una más fácil manipulación de estas mismas.

Estructura de datos:

Después de cada turno, el programa exporta el tablero en un archivo de texto llamado “Archivo_salida.txt” y que dentro de este se encuentra el tablero de juego con la última jugada hecha. Es importante de recordar de modificar la ruta del archivo a una que usted pueda utilizar.

Módulos y funciones:

main [Principal.cpp]

El main es una parte crucial de nuestro programa debido a que es desde este comienza y desde donde se llaman a las diferentes funciones en el orden que son necesarias para el correcto funcionamiento. Además de todo esto éste módulo es aquel que también brinda el cambio de jugador después de cada jugada y donde se indica la ruta del archivo de salida donde se guardara la última jugada. Su estructura es la siguiente:

```
int main(){

    int tablero[MAX_FILAS][MAX_COLUMNAS], columnaElegida, fila, jugador = JUG_1;
    bool ganador = false;

    string rutaSalida = "D:/Franco/Facultad/Algoritmos/Programas_C++_2021/106590_TP1/106590_TP1/Archivo_salida.txt";

    cout << "=========================================" <<endl;
    cout << "************** 4 EN LINEA **************" <<endl;
    cout << "=========================================" <<endl;

    cout<<endl;

    limpiarTablero(tablero);
    mostrarTablero(tablero, rutaSalida);

    do{

        columnaElegida = pedirJugada(tablero, jugador);
        fila = acomodarJugada(tablero, columnaElegida, jugador);

        mostrarTablero(tablero, rutaSalida);

        ganador = busquedaVertical(tablero, columnaElegida, fila, jugador)
                + busquedaHorizontal(tablero, columnaElegida, fila, jugador)
                + busquedaDiagonalDes(tablero, columnaElegida, fila, jugador)
                + busquedaDiagonalAsc(tablero, columnaElegida, fila, jugador);

        if (empate(tablero)){
            break;
        }

        if (jugador == JUG_1){
            jugador = JUG_2;
        }else{
            jugador = JUG_1;
        }

    }while(ganador == false);


    return 0;
}
```

limpiarTablero [Interacciones.cpp]

Como su nombre indica, esta funcion limpia el tablero colocando en todos los casilleros de este, el simbolo determinado como VACIO, en nuestro caso seria “ . ”, y asi poder comenzar una nueva partida.

```
void limpiarTablero(int tablero[MAX_FILAS][MAX_COLUMNAS]){

    for (int a = 0; a < MAX_FILAS; a++){
        for (int i = 0; i < MAX_COLUMNAS; i++){
            tablero[a][i] = VACIO;
        }
    }
}
```

mostrarTablero [Interacciones.cpp]

Esta función cumple de 2 objetivos. El primero es el de mostrar en la consola el estado actual de nuestro tablero de juego además de las anteriores jugadas de cada jugador con sus signos correspondientes (estos signos se encuentran y pueden ser modificados en el archivo Constantes.h y estos son "." para un casillero vacío, "O" para el jugador 1 y "X" para el jugador 2). Su segundo objetivo consta de sobrescribir en nuestro archivo de texto "Archivo_salida.txt", ya antes mencionado, el estado del tablero con su jugada más reciente.

```
void mostrarTablero(int tablero[MAX_FILAS][MAX_COLUMNAS], string rutaSalida){

    /* abre un archivo de salida */

    ofstream salida;
    salida.open(rutaSalida.c_str());

    cout<<endl;

    for(int i = 0; i < MAX_FILAS; i++){

        cout<<"     ";

        for(int a = 0; a < MAX_COLUMNAS; a++){

            if(tablero[i][a] == JUG_1){
                cout<<SIG_JUG_1;
                salida<<SIG_JUG_1;

            }else{

                if(tablero[i][a] == JUG_2){
                    cout<<SIG_JUG_2;
                    salida<<SIG_JUG_2;

                }else{
                    cout<<SIG_VACIO;
                    salida<<SIG_VACIO;
                }
            }
        }
        cout<<endl;
        salida<<endl;
    }

    /* cierra el archivo, liberando el recurso */
    salida.close();
}
```

pedirJugada / acomodarJugada [Interacciones.cpp]

Interactuando con el/los usuarios a través de la consola, pedirJugada y acomodarJugada a pesar de ser diferentes funciones, ambas trabajan en conjunto para pedirle al usuario que ingrese un valor del 1 al 10 (para nuestro caso particular) para indicar en que columna quiere ingresar su ficha, corroborar si es un valor posible o valido (de no serlo se le pedirá que lo ingrese de nuevo) y por ultimo acomodar la ficha en la columna que el usuario eligió y en la fila que le corresponde.

```
int pedirJugada(int tablero[MAX_FILAS][MAX_COLUMNAS], int jugador){

    int columnaElegida;

    cout<<"Jugador N"<<jugador<<" elija un numero del 1 al "<<MAX_COLUMNAS<<": ";
    cin>>columnaElegida;

    columnaElegida--;

    while(columnaElegida < 0 || columnaElegida > MAX_COLUMNAS - 1 || tablero[0][columnaElegida] != VACIO){
        cout<<"No puede usarse esa columna, intente otra: ";
        cin>>columnaElegida;

        columnaElegida--;
    }

    cout<<endl;

    return columnaElegida;
}

//-----------------------------------------------------------------------------------------------------------

int acomodarJugada(int tablero[MAX_FILAS][MAX_COLUMNAS], int columna, int jugador){

    bool casilleroVacio = false;
    int fila;

    for(int i = MAX_FILAS - 1; casilleroVacio == false; i--){
        if (tablero[i][columna] == VACIO){
                tablero[i][columna] = jugador;
                fila = i;
                casilleroVacio = true;
        }
    }

    return fila;
}
```

Δ Funciones de búsqueda:

busquedaVertical / busquedaHorizontal / busquedaDiagonalAsc / busquedaDiagonalDes [Reglas.cpp]

Estas cuatro funciones cumplen todas con el mismo objetivo de buscar si existe un ganador con el pasar de cada turno. Son cuatro funciones separadas, ya que cada una busca en una dirección diferente respectivamente a sus nombres. Si es de ser que una de estas funciones encuentre un ganador esta devolverá un valor “true” como respuesta, finalizando el programa, además de indicarlo mediante un mensaje en la consola.

Estas son sus respectivas estructuras:

```
bool busquedaVertical(int tablero[MAX_FILAS][MAX_COLUMNAS], int columna, int fila, int jugador){

    //Vertical
    bool encontrado = false, ganador = false;
    int total = 0;

    for(int i = 0; i < MAX_FILAS; i++){

        if(encontrado){

            if(tablero[i][columna] == jugador){
                total++;
            }else{
                encontrado = false;
                total = 0;
            }
        }

        if(tablero[i][columna] == jugador && !encontrado){
            encontrado = true;
            total++;
        }

        if(total == PUNT_GANAR){
            cout<<"El jugador "<<jugador<<" gana!"<<endl;
            ganador = true;
            break;
        }
    }
    return ganador;
}
```

```
bool busquedaHorizontal(int tablero[MAX_FILAS][MAX_COLUMNAS], int columna, int fila, int jugador){

    //Horizontal
    bool encontrado = false, ganador = false;
    int total = 0;

    for(int i = 0; i < MAX_COLUMNAS; i++){

        if(encontrado){

            if(tablero[fila][i] == jugador){
                total++;
            }else{
                encontrado = false;
                total = 0;
            }
        }

        if(tablero[fila][i] == jugador && !encontrado){
            encontrado = true;
            total++;
        }

        if(total == PUNT_GANAR){
            cout<<"El jugador "<<jugador<<" gana!"<<endl;
            ganador = true;
            break;
        }
    }
    return ganador;
}
```

```
bool busquedaDiagonalDes(int tablero[MAX_FILAS][MAX_COLUMNAS], int columna, int fila, int jugador){

    //Diagonal descendiente

    bool encontrado = false, ganador = false;
    int total = 0;

    while((fila != 0 || columna != 0))
    {
        fila--;
        columna--;

        if(fila == 0 || columna == 0)
        break;
    }

    do
    {
        if(fila >= MAX_FILAS){
            break;
        }

        if(encontrado){

            if(tablero[fila][columna] == jugador){
                total++;
            }else{
                encontrado = false;
                total = 0;
            }
        }

        if(tablero[fila][columna] == jugador && !encontrado){
            encontrado = true;
            total++;
        }

        if(total == PUNT_GANAR){
            cout<<"El jugador "<<jugador<<" gana!"<<endl;
            ganador = true;
            break;
        }
        fila++;
        columna++;

    }while(fila < MAX_FILAS);

    return ganador;
}
```

```
bool busquedaDiagonalAsc(int tablero[MAX_FILAS][MAX_COLUMNAS], int columna, int fila, int jugador){

    //Diagonal ascendiente

    bool encontrado = false, ganador = false;
    int total = 0;

    while((fila != 0 || columna != MAX_COLUMNAS)){
        fila--;
        columna++;

        if(fila == 0 || columna == MAX_COLUMNAS){
            break;
        }
    }

     do
    {
        if(fila >= MAX_FILAS){
            break;
        }

        if(encontrado){

            if(tablero[fila][columna] == jugador){
                total++;
            }else{
                encontrado = false;
                total = 0;
            }
        }

        if(tablero[fila][columna] == jugador && !encontrado){
            encontrado = true;
            total++;
        }

        //cout<<"total: "<<total<<endl;
        if(total == PUNT_GANAR){
            cout<<"El jugador "<<jugador<<" gana!"<<endl;
            ganador = true;
            break;
        }

        fila++;
        columna--;

    }while(fila < MAX_FILAS);

    return ganador;
```

empate [Reglas.cpp]

La última función es la función empate que se encarga, valga la redundancia, de declarar un empate y terminar el programa si es que esto llega a suceder. La forma en la que se hace esto, es que cada vez que se coloca una nueva ficha se revisa si todas las columnas de la primera fila (la fila más alta) están completas, de ser así, eso significa que todas las columnas están llenas y por tanto ya no pueden realizarse más jugadas.

```
bool empate(int tablero [MAX_FILAS][MAX_COLUMNAS]){

    int contador = 0;
    bool lleno = false;

    for (int i = 0; i < MAX_COLUMNAS; i++){
        if (tablero[0][i] != VACIO){
            contador ++;
        }
    }

    if(contador == MAX_COLUMNAS){
        lleno = true;
        cout<<"Empate"<<endl;
    }

    return lleno;
}
```

Manual del Usuario

¿Qué es este programa?

Este programa es la forma virtual de jugar al clásico “4 en línea”, en el que vos contra un amigo compiten por ver quién puede formar un conjunto de 4 o más fichas de su mismo símbolo de forma continua, ya sea horizontal, vertical o en diagonal.

¿Cómo funciona?

¡Muy simple!

Una vez vos o tu amigo ya hayan iniciado el programa les aparecerá algo como la imagen a continuación…

• Como se ve en la imagen, aparece el titulo del juego y mas debajo de este se ve un figura con puntos, este sera nuestro tablero de juego, cada punto representa un casillero vacio que se iran llenando a medida que avanzamos en el juego.

Pero seguramente te estarás preguntando ¿Cómo hago para poner mis fichas en el tablero? ¿No? Bueno, ante todo es muy sencillo, únicamente tienes que poner vos o tu amigo/contrincante, con un teclado, lo que indica el texto debajo del tablero. Quien quiera ser el jugador numero 1 debe de comenzar poniendo un número del 1 al 10 (en este caso) **[MUY IMPORTANTE solo colocar números y nada más].**

¿Por qué del 1 al diez? Por lo siguiente ---------->

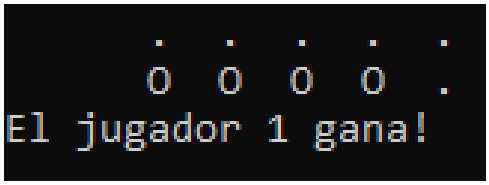
Por ultimo si no saben cuál es la ficha de cada uno sepan que las fichas del Jugador 1 son así “ O ” mientras que la del jugador 2 son así “ X ”.

Instrucciones del Juego:

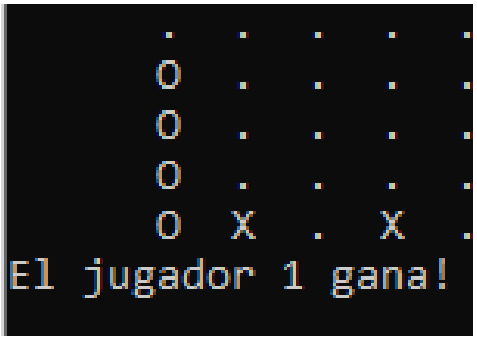
De ser el caso que uno no conozca las reglas las explicaré.

Las reglas básicas del juego son, gana el que pueda poner 4 o más fichas de su mismo símbolo/color de las siguientes formas….

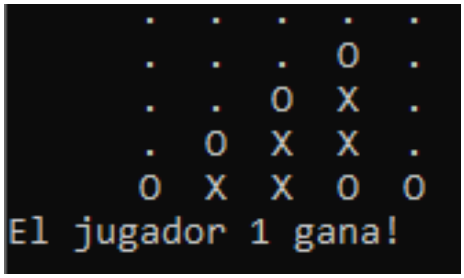
Horizontal

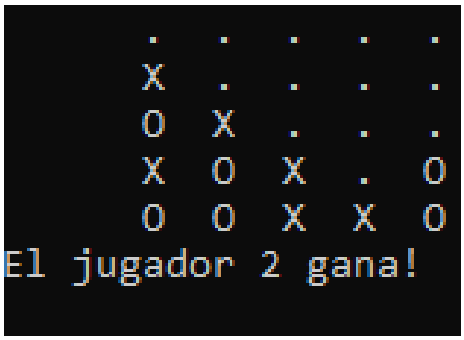


Vertical



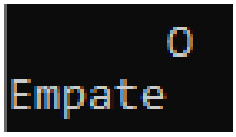
Diagonal





Si cualquiera de los dos llegase a lograr esto debajo del tablero aparecera un cartel indicando quien gano y a su vez terminando con el programa, si es que quieren jugar de nuevo se tendra que volver a iniciar el programa.

Por ultimo de darse el caso que completen el tablero sin haber logrado completar una jugada aparecera un cartel que diga “Empate” y se terminara el juego.



Y eso es todo, espero que disfruten del juego.

Informe

Objetivo:

Generar una pieza de software que simule el funcionamiento del juego 4 en línea en su versión Jugador contra Jugador.

Desarrollo:

Este trabajo, aunque en un principio parecía algo sencillo, fue más largo de lo esperado. Pero a pesar de lo largo del programa no tuve muchas complicaciones a la hora de hacerlo debido a forma de desarrollarlo, comencé el programa probándolo poco en cada progreso que hiciera para no tener una gran pila de problemas que no sabría por dónde comenzar a desarmar o si es que siquiera lo que había escrito hubiese estado bien o mal, así que trate de plantear el problema poco a poco e ir probando a medida que programaba para ver cómo es que este iba saliendo, al hacer esto tuve que hace el programa en un único archivo .cpp y una vez que ya tenía el programa completo, sin problemas y/o buggs de los cuales yo preocuparme, me dedique a buscar cual sería la forma más sencilla de repartir las funciones para que tengan un estilo de similitud pero a su vez no de poner todas en un solo archivo, debido a esa inconveniencia, comencé a buscar ejemplos sobre la repartición de las funciones, y así, termine topándome con el ejemplo del ahorcado, que se encuentra dentro de las pagina del campus, que al tratarse de otro juego me dejo una mejor visión sobre cómo podría hacer la organización de estas funciones.

Dentro de otras de las complicaciones que tuve a lo largo del desarrollo del programa se encontraban las funciones de búsqueda, ósea las funciones utilizadas para buscar a un ganador. Aunque las búsquedas verticales y horizontales eran sencillas, los problemas comenzaron a aparecer al tratarse de las diagonales, ya que de la forma en la que tenía el funcionamiento del programa a veces este detectaba un 4 en línea con las columnas 1, 2 y 10 completas de fichas intercaladas, lo cual no tenía mucho sentido. Pero pasando un tiempo debuggeando, testeando e investigando online sobre una solución llegue a la resolución de que a veces las variables se mezclaban entre filas y columnas para la búsqueda, pero no para resto del programa y una vez ya tenía detectado el problema fue sencillo el hecho de resolverlo. Ante todo, estuvo muy interesante hacer un programa como este y por lo menos yo, estoy satisfecho de cómo termino quedando para el tiempo que me tomo hacerlo y con el tiempo dado.