”Proyecto Final ’UEFA Champions League.”’

Sanchez Vilchis Cynthia Valeria

***Abstract*— En este proyecto se realizo un algoritmo, para simular partidos de la ’Champions League’, enfrentandose en distintas etapas, (octavos, cuartos, semifinal, final), para posteriormente obtener un campeon y mostrar estadisticamente el total de juegos, ganados, perdidos, empatados, para posteriormente guardarlo en un archivo.txt.**

I. Creacion de jugadores.

Se crearon dos estructuras una llamada Jugador, y otra estructura llamada Nodo, la estructura Jugador contiene los elementos (strings/enteros) que poseera el Jugador y la estructura Nodo contiene un apuntador a estructura Jugador, la cual definiremos como estructura sobre estructura y un apuntador llamado siguiente, que se define como el enlace de las estructuras ligadas que se formaran.
Se crearon 16 archivos.txt, los cuales incluyen numeros aleatorios, nombre del jugador, posicion.
Estos datos fueron exportados a una listda enlazada simple, mediante funciones, las cuales seran explicadas a continuacion:
”void leerJugaArch();”, los parametros que recibe son de una estructura llamada Jugador la cual tiene un apuntador llamado *p, inicializado en nullptr y otro apuntador a FILE*. p apuntara a el contenido de los elementos que encontramos en la estructura ”Jugador”, y el apuntador FILE*, apuntara a el archivo.txt y poder guardarlo en la estructura ”Jugador”, la cual nos permitira crear las listas enlazadas simples, tambien encontramos las siguientes funciones, **”void leerArch();”**,**”Struct Jugador *crearJU();”**, **”Struct Nodo *crearNod();”**.
Estas funciones antes mencionadas, varian de parametros, **”void leerArch();”**, contiene como parametros un char *apuntador, el cual apuntara, a la direccion absoluta de memoria donde esta guardado el archivo.txt, un doble apuntador a la estructura anterior y otro doble apuntador a la estructura posterior.
La funcion **”struct Jugador *crearJug();”**, alojara memoria en tiempo de ejecucin *nuevoJugador y en **”struct Nodo *crearNod();”**,tendra como parametros un apuntador *p, a estructura Jugador, es aqui donde vizualizaremos la definicion ”estructura sobre estructura”.
Las funciones posteriores son:
”void agregarList();”, **”void imprimirLista();”**, **”void imprimirJugador();”**, **”void libMem();”**, estas ultimas funciones, se relacionan al proceso visto en clase de listas ligadas, la relacion de apuntadores de una estructura a otra estructura, la impresion y la liberacion de memoria que se debe de utilizar cada que utilizamos el proceso de Memoria Dinamica, estas funciones fueron guardadas en una libreria propia llamada **”jugador.hpp”**, las cuales son incluidas en el main (), mediante define ”jugadores.hpp”. Obteniendo como resultado la siguiente imagen:

```
Nombre: Andreas Pereira
Posicion: Medio Centro ofensivo
Tarjetas Amarillas: 5
Faltas: 19
Goles Anotados: 15
Numero Jugador: 2

Nombre: Romelu Lukaku
Posicion: Delantero Centro
Tarjetas Amarillas: 0
Faltas: 28
Goles Anotados: 9
Numero Jugador: 1

Nombre: Marcus Rashford
Posicion: Delantero Centro
Tarjetas Amarillas: 7
Faltas: 19
Goles Anotados: 10
```

II. Creacion Octavos Final

Estas funciones tienen el mismo comportamiento que en la creacion de los jugadores; no se pueden ocupar las anteriores para ejecutar las acciones que requeriamos, por lo tanto creamos una estructura distinta que inclua, el nombre del Estadio y los equipos a disputar, creando una simuacion ”estatica”, por asi decirlo, ya que la informacion fue extraida mediante un archivo.txt.
Algo nuevo que podemos encontrar en la libreria llamada **partidosOctavos.hpp”**, es que incluye una estructura que nos ofrece la libreria **”ctime”** y **”vector”**, las cual nos permite crear un vector de Fechas y Horas.
”Ctime”, nos brinda por defecto una estructura que incluye elementos enteros asignados a dias, mes, ao, hora, minuto y segundo, por ejemplo, el digito (0), nos indica el dia ”domingo” de la semana, al momento de realizar las impresiones de simular la fecha nos mostraba 352019, ya que no habiamos asignado cadena de caracteres a los numeros en estructura y por ende no nos mostraba los dias en nuestro idioma, simulando el proceso que mencionamos, en la parte de la libreria podriamos observar esta forma de registro de los dias.
—

dia	.push$_b ack$	(”Domingo”);
dia	.push$_b ack$	(”Lunes”)
dia	.push$_b ack$	(”Martes”)

III. Creacion Partidos (Arbol Binario)

En esta parte del proyecto, se requiere el uso de Arboles Binarios,para realizar el ”match” de los juegos; me fue consufo e imposible realizar la implementacion de codigo, conforme a mis ideas.
Exponiendo en este apartado las opciones que surgieron entorno a mi imaginacion para elaborar el Match del juego.

A. Opcion I

Crear un estructura Partidos con elemento string nombreE; elemento int puntos;, crear una estructura Nodo, que apunte a la estructura Partidos y que tenga como elemento int aux;, la estructura Nodo tendria dos apuntadores *izq; *der;,(nodo.izquierdo, nodo.derecho), los cuales tendrian Nombre, Pts y Aux.
Implemenetar 6 funciones, las cuales serian: **”struct Nodo crearPartidos();”**, **”void insertarElementos();”**,recorrido por profundidad **”void PostOrden();”**, **”void eliminarEquipo();”**, **”struct Nodo unirArbol();”** y **”void imprimirArbol”**;, guardar estas funciones en una libreria llamada ”ABB.hpp”, pedir al usuario insertar el nombre del equipo y los puntos de tal, para posteriormente incluir una validacion si los puntos ¿= puntuajeMaximo (asignado), continuar en el partido, sino entraria la funcion **”void eliminarEquipo();”**, asi posteriormente, se eliminarian los nodos del arbol que tienen menor puntuacion.

B. Opcion II

Otra manera en que pense que seria viable, es llenar el arbol con diguitos random, que seria como nuestro arbolAuxiliar, en el arbolJuego pedir al usuario el nombre y el marcador del equipo, de igual manera poner una validacion de si el marcador es igual a un marcadorMax, clonar el valor al arbol auxiliar y se colocarian los que mayor marcador, para posteriormente hacer un ordenamiento directo y poder observar al equipo con mayor puntuacion en base al marcador y seria el ”Campeon”.

C. Opcion III

Crear un arbol de desiciones, cuestionando preguntas como Gano Barcelona?, si tu respuesta es ’no’, dirigirse al nodo derecho y cuestionar Gano Bayern?, si tu respuesta es ’no’, podriamos definir un cout ¡¡ ”Empate”; y crear un sub arbol para que exista un ganador, cuando el resultado sea ’si’, preguntar Cuantos goles? dependiendo del valor de goles, guardarlo a la izquierda o derecha de un arbolAuxiliar, asi generando el ”Match” del partido y guardando los datos en dos arboles uno estatico y el otro auxiliar.

IV. Creacion Estadisticas

La creacion de estadisticas fue creada estaticamente, creando una estructura que posee elementos de tipo entero y un elemento de tipo string , para total de partidos, partidos ganados, partidos empatados, partidos perdidos, creando un arreglo estatico, pidiendo al usuario la cantidad de partidos antes mencionados, anidando dos for uno se enfoca en cada equipo (en este caso los 16 equipos), y el for dentro del for nos indica que nos mostrara una sola vez el pedir datos al usuario.
Cree tres funciones de tipo void, las cuales fueron llamadas **”void ingresarD()”**, **”void Promedio();”** e **”void impresionD();”**, este proceso fue totalmente estatico, para posteriormente guadar esos datos en un archivo.txt.

V. Creacion main

Esta es la parte principal de nuestro proyecto, en la cual incluimos siete librerias de C++, las cuales son: **”include iostream”**, **”include stdio.h”**, **”include stdlib.h”**, **”include ctime”**, **”include vector”**, **”include stream”**, **”include strings.h”**, las librerias propias fueron: **”include ”jugadores.hpp””**, **”include ”Estadisticas.hpp””**, **”include ”partidosOctavos.hpp””**, **”include ”ABB.hpp””**, despues de ingresar las librerias que requerimos, se genero el uso de un switch para poder ordenar visualmente las opciones que desee ejecutar el usuario.
Creando un switch con opciones del [1]Equipos, [2]Fechas Octavos, [3] Partidos, [4] Estadisticas, [0] Salir.
Dentro del case [1], encontraremos un switch anidado de consonantes con las letras [A] Grupo, [B] Grupo, [C] Grupo, [D] Grupo, dentro de estos casos, veremos 4 equipos en cada uno, y es donde llamamos a la funcion imprimirLista, para generar la impresion de jugadores con las caracteristicas antes mencionadas.
Nuestros cases[], los colocamos dentro de un do-while el cual te indica si quieres regresar con un cout ¡¡ ”Atras: (Presione ’x’);, al menu principal y ejecutar las opciones que posee el porgrama.