Liga Fantástica MP

Pablo Cumbreras Hernández
Alejandro Díaz Sadoc
Patricio Santiago Fernández Flórez
José Javier Gómez Rosado

Descripción funcional

Principal: Permite registrarse como usuario participante y acceder al sistema en la sección correspondiente con su rol.

Administrador: Permite gestionar los datos de la configuración, los equipos y los usuario(crear, modificar, listar y eliminar). Falta la gestión de futbolistas.

Cronista:Permite listar los equipos y modificar la valoración de los jugadores.

Participante: Permite crear plantillas y eliminarlas

Planteamiento

Puesto que la aplicación tiene tres usos distintos dependiendo del perfil del usuario, se han creado tres módulos distintos: “administrador”, “participante” y “cronista”. Cada uno de ellos se encarga de las funciones necesarias para cada rol. Teniendo en cuenta que todos los módulos requerían de unas operaciones de escritura y lectura similares, también se ha creado un módulo “fichero” que encapsula dichas operaciones. Por otra parte, también tenemos el módulo principal en el que están la función “main”.

Documentación

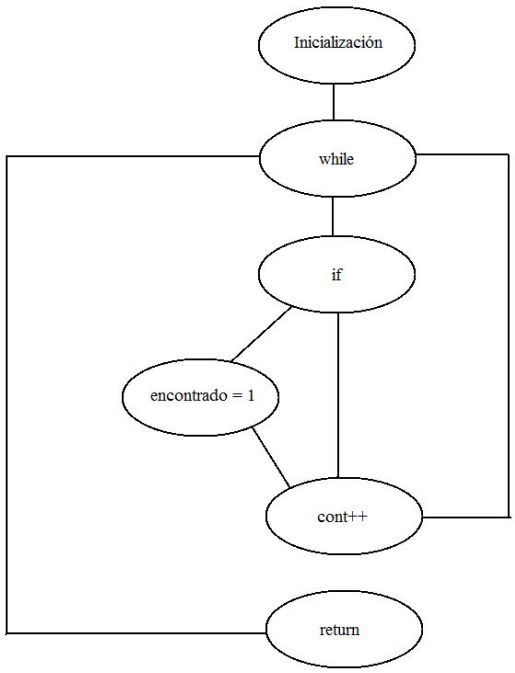
Administrador.c
Contiene todas las funciones necesarias para el perfil administrador. Fundamentalmente, son operaciones para mostrar, eliminar, agregar y modificar datos referentes a los equipos, usuarios del sistema y parámetros de configuración.
Fichero.c
Contiene todas las operaciones de lectura y escritura de datos que son necesarias para el funcionamiento de los otros módulos. Aquí se definen las estructuras utilizadas para facilitar el manejo de datos.
Cronista.c
Contiene todas las funciones del perfil cronista, es decir, listar equipos y realizar valoraciones de los futbolistas y equipos.

Participante.c
Contiene las funciones del perfil participante. Esencialmente son funciones de manejo de plantillas.

Casos de prueba

Módulo fichero.c, función loguear

```
Usuario loguear(char* logUsuario, char* passUsuario){
    Usuario *u = obtenerUsuarios();
    Usuario usuario;
    strcpy(usuario.codigo, "00");
    int dimension = nUsuarios();
    int encontrado = 0;
    int cont = 0;
    while(cont < dimension && encontrado == 0){
        if(strcmp(u[cont].login, logUsuario) == 0 &&
                strcmp(u[cont].pass, passUsuario) == 0){
            encontrado = 1;
            usuario = u[cont];
        }
        cont++;
    }
    return usuario;
}
```

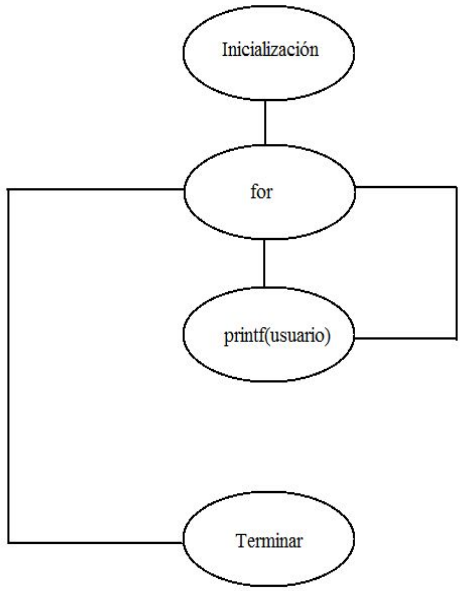


Complejidad ciclomática: 7- 6 = 1

Caso nº1	Usuario existente, contraseña correcta	No devuelve “00”	Resutltado OK
Caso nº2	Usuario existente, contraseña incorrecta	Devuelve “00”	Resultado OK
Caso nº3	Usuario existente, contraseña de otro usuario	Devuelve “00”	Resultado OK
Caso nº4	Usuario inexistente, contraseña inexistente	Devuelve “00”	Resultado OK
Caso nº5	Usuario vacío, contraseña vacía	Devuelve “00”	Resultado OK

Módulo administrador.c, función listarUsuarios

```
static void listarUsuarios(Usuario *u, int* numUs) {
    int i;
    puts("\n <### LISTANDO USUARIOS ###>");
    for (i = 0; i < *numUs; i++) {
        printf("%s %s %s %s %s\n", u[i].codigo, u[i].nombre, u[i].tipo,
               u[i].login, u[i].pass);
    }
}
```

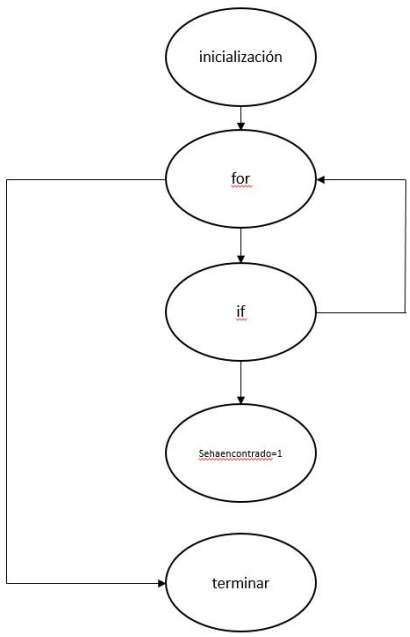


Complejidad ciclomática: 4-4 = 0

Módulo usuario.c, función comprobarQueNoEsta

```
// Cabecera: void comprobarQueNoEsta(int codigo,int *vector,int tam)
// Precondicion:
// Comentarios: Comprobar que un int no esta en un vector de int
int comprobarQueNoEsta(int codigo,int *vector,int tam)
{
    int i,sehaencontrado=0;
    for(i=0;i<tam;i++)
    {
        if(codigo==vector[i])
            sehaencontrado=1;
    }

    return sehaencontrado;
```



Complejidad ciclomática: 5-5 = 0