esther.silva@uca.es

PRÁCTICAS DE DISEÑO MODULAR

Documentación interna del proyecto

Grupo 4	Jorge Rodríguez De La Sierra José Tur Román José Joaquín Carral Fernández

A continuación, se procederá al comentario del código realizado por el grupo 4.
Se clasificará por archivos C debido a la forma en la que nos dividimos el trabajo entre nosotros. Son un total de 11 archivos.

Nombre	acceso
Precondición	En la función “menus” el usuario elija acceder al sistema.
Descripción	Realiza las comprobaciones necesarias para permitir que el usuario pueda acceder al sistema.
Postcondición	La comprobación de los datos del usuarios se validen dependiendo de si el usuario está o no registrado previamente.

Comentarios del Código

```
#include <stdio.h>
#include <string.h>
#include "tipos.h"
#include "cargar.h"
#include "acceso.h"
#include "menus.h"

void acceso(usuario *vUsuarios,int n){
   int i, existe=0;
   char login[6]; //login almacenará el nombre de usuario //
   char pass[9]; //pass almacenará la contraseña del usuario//

   printf("\nIntroduzca los datos: ");
   printf("\nLogin: ");
   fflush(stdin); //limpieza del stdin, es decir del buffer de entrada//
```

```
    scanf("%5s",login);

    printf("\nContrasena: ");
    fflush(stdin);
    scanf("%8s",pass);

    for(i=0;i<n;i++){
    //En este bucle se comprueban los datos para el acceso en base a los datos//
    introducidos por el usuario anteriormente//
        if(strcmp(vUsuarios[i].login,login)==0) {
   //Si vUsuarios[i].login (mediante i se va moviendo el sistema por los datos
registrados en el fichero usuarios.txt) es idéntico a algún login registrado, el strcmp
devuelve 0, y por lo contrario devolverá 1//
            existe=1; //"existe" le dice al sistema si el usuario existe (1) o no (0)//

            if(strcmp(vUsuarios[i].pass,pass)==0){ //Si la contraseña introducida
coincide con la del usuario en cuestión se comprueba el tipo de usuario//
                if(strcmp(vUsuarios[i].perfil,"participante")==0)
                    menuParticipante(vUsuarios[i].id);
                if(strcmp(vUsuarios[i].perfil,"cronista")==0)
                    menuCronista();
                if(strcmp(vUsuarios[i].perfil,"administrador")==0)
                    menuAdministrador();
            }
                else{ //Si la contraseña no coincide le imprime un mensaje de error//
                printf("\nLa contrasena introducida es incorrecta.");

                acceso(vUsuarios,nUsuarios); //Devuelve al usuario al inicio de la función//
            }
        }
    }

    if(existe==0){ //Si el usuario no existe imprime un mensaje de error//
        printf("\nEl usuario no esta registrado en el sistema, para continuar
registrese.");
    }

    system("pause");
}
```

Nombre	admin
Precondición	El usuario en “acceso” sea identificado como perfil administrador.
Descripción	Realizará todas las funciones encargadas al administrador predefinidas por la empresa.
Postcondición	El programa finaliza cuando el usuario elige salir del sistema.

Comentarios del Código

```
#include "admin.h"

void editConfig(config **vConfigs,int nConfigs){
   int i;
   int val;

   printf("\n<_______CONFIGURACION_______>\n");

   for(i=0;i<nConfigs;i++){ //Aquí se imprimen las configuraciones almacenadas en configuración.txt//
         printf("%d. %s: %d\n", i+1, (*vConfigs)[i].config, (*vConfigs)[i].valor);
   }

   printf("\n_____\n");

   i=-1;

   while(i<0 || i>nConfigs){//Mientras que i no tenga un valor que oscile entre las distintas opciones de edición de configuración y el 0(para salir) seguirá repitiendo//
      printf("Introduzca el numero de la opcion de la configuracion que desea modificar");
         printf("\nSi desea volver al menu administrador introduzca 0\n");
      scanf("%d",&i);

      if(i!=0){ //Si el valor introducido no está entre loslimites puestos antes se muestra un mensaje de error//
         if(i<1 || i>nConfigs){
            printf("\nPor favor, introduzca un valor entre 1 y %d\n",nConfigs);
         }
                  else{ //Si el valor está entre los valores se le muestra la configuración actual de la opción elegida//
                  printf("\nEl valor actual de %s es de %d\n",(*vConfigs)[i-1].config,(*vConfigs)[i-1].valor);
```

```
                printf("\nIntroduzca un nuevo valor para %s: ",(*vConfigs)[i-
1].config);

                        scanf("%d",&val);

                (*vConfigs)[i-1].valor=val; //Le asignamos el nuevo valor a la
configuración editada por el usuario //

                printf("\nEl nuevo valor de %s es %d\n",(*vConfigs)[i-
1].config,(*vConfigs)[i-1].valor);

            }
        }
    }
        //system pause;// //Es opcional ponerlo o no, en ambos casos no da fallo
alguno//
}

void listarUsuarios(usuario *vUsuarios,int nUsuarios){
    int i;

    printf("\n<_______USUARIOS_______>\n");
    printf("Introduzca los datos en el orden de: ID-NOMBRE-PERFIL-LOGIN-
CONTRASENA\n");

    for(i=0;i<nUsuarios;i++){ //El bucle se encarga de leer la información de los
usuarios almacenada en usuarios.txt e imprimirla posteriormente//
        printf("%s-%s-%s-%s-
%s\n",vUsuarios[i].id,vUsuarios[i].nombre,vUsuarios[i].perfil,vUsuarios[i].login,vU
suarios[i].pass);
    }

    printf("_____\n");

}

void modificarUsuario(usuario **vUsuarios,int nUsuarios){

    int i;
    int u=-1;
    char resp=' ';
    char id[3];
    int nModUsuario=0;
    usuario *modUsuario;

    modUsuario=malloc(sizeof(usuario)); //Reservamos un espacio que luego
devolveremos, y es con un tamaño que especifica el sizeof(tamaño de) de usuario.//
```

```
    printf("\nMODIFICAR USUARIO\n");
    printf("##################\n");

    while(u==-1){
        printf("\nIntroduzca el ID del usuario a modificar: \n");
        scanf("%2s",id);
        printf("\n");

        for(i=0;i<nUsuarios;i++){ //Aquí se recorre el nUsuarios para saber cuántos usuarios hay//
            if(strcmp((*vUsuarios)[i].id,id)==0){ //Y comparamos el usuario correspondiente a la vuelta i para ver si es el que el usuario pide es igual, a lo que devolverá strcmp un 0. Cuando se cumpla le damos un valor a u distinto de -1 para que salga del bucle//
                u=i;
            }
        }

        if(u==-1){ //Sin embargo, si no se ha modificado mostramos un mensaje de error, ya que no existe el usuario//
            printf("\nSu usuario no existe, introduzca un usuario existente.\n");

        }
            else{ //Por lo contrario, imprimimos la info del usuario en cuestión//
            printf("\nInfo del usuario: %s-%s-%s-%s-%s\n", (*vUsuarios)[u].id, (*vUsuarios)[u].nombre, (*vUsuarios)[u].perfil, (*vUsuarios)[u].login, (*vUsuarios)[u].pass);

            printf("Introduzca la nueva informacion siguiendo el modelo: \n(ID-NOMBRE-PERFIL-LOGIN-CONTRASENA)\n");

            anadirUsuario(&modUsuario,&nModUsuario,1);
//Llamamos a la función añadir un nuevo usuario para recoger los datos nuevos//
            while(resp!='s' && resp!='n'){ //Le preguntamos al usuario si quiere guardar los cambios mediante la variable resp que almacenará s o n dependiendo de la respuesta del usuario//
                printf("¿Modificar el usuario? Responda (s/n) \n");
                fflush(stdin);
                scanf("%c",&resp);
            }

            if(resp=='s'){ //Se confirman los datos//
                (*vUsuarios)[u]=modUsuario[0];
            }
                    else{ //Se cancela la acción de modificación//
                printf("\nNo se sobrescribiran los datos del usuario.\n");
            }
```

```
      }
   }
}

void anadirUsuario(usuario **vUsuarios,int *nUsuarios,int admin){
   int n;

   *vUsuarios=realloc((usuario *)(*vUsuarios),((*nUsuarios)+1)*sizeof(usuario));
//Aquí aumentamos el tamaño de almacenamiento reservado para el nuevo
usuario//
   printf("\nIntroduzca el ID del usuario (ID --> 2 Cifras maximo): ");
   fflush(stdin);
   scanf("%2s",(*vUsuarios)[*nUsuarios].id);

   printf("\nAhora, introduzca el nombre completo del usuario (20 letras como
maximo): ");
   fflush(stdin);
   scanf("%20s",(*vUsuarios)[*nUsuarios].nombre);

   while(n<1 || n>3){ //Mientras que no se salga de los límites de opciones (3)//
      if(admin==1){
         printf("\nSeleccione el perfil del usuario:\n");
         printf("1) Administrador\n");
         printf("2) Cronista\n");
         printf("3) Participante\n");
         fflush(stdin);
         scanf("%d",&n);
      }
         else{ //Si no elige ningún perfil se asignará por defecto el participante (3)//
         n=3;
      }

      switch(n){
         case 1:
            strcpy((*vUsuarios)[*nUsuarios].perfil,"Administrador");
            break;
//Con la instrucción strcpy coipamos lo que hay en el segundo parámetro en el
primer parámetro que señalemos, en este caso copiamos “Administrador” en
(*vUsuarios)[*nUsuarios].perfil //
         case 2:
            strcpy((*vUsuarios)[*nUsuarios].perfil,"Cronista");
            break;

         case 3:
            strcpy((*vUsuarios)[*nUsuarios].perfil,"Participante");
            break;

         default:
```

```
            printf("\nLa opcion que ha introducido no es correcta.\n");
        }
    }

    printf("\nIntroduzca el login del usuario (5 letras como maximo): ");
    fflush(stdin);
    scanf("%5s",(*vUsuarios)[*nUsuarios].login);

    printf("\nAhora, introduzca la contrasena del usuario (8 letras como maximo): ");
    fflush(stdin);
    scanf("%8s",(*vUsuarios)[*nUsuarios].pass);

    (*nUsuarios)++; //Aumentamos en uno el número de usuarios//
}

void eliminarUsuario(usuario **vUsuarios,int *nUsuarios){
    int i;
    int u;
    char resp=' ';
    char id[3];

    printf("\n<-------ELIMINAR USUARIO------->");

    printf("\n\nIntroduzca el ID del usuario que desea eliminar: (2 cifras maximo)
");
    scanf("%2s",id);

    for(i=0;i<(*nUsuarios);i++){  //Aquí se recorre el nUsuarios para saber cuántos usuarios hay//
        if(strcmp((*vUsuarios)[i].id,id)==0){ //Y comparamos el usuario correspondiente a la vuelta i para ver si es el que el usuario pide es igual, a lo que devolverá strcmp un 0. Cuando se cumpla le damos un valor a u distinto de -1 para que salga del bucle//
            u=i;
        }
    }

    while(resp!='s' && resp!='n'){ //Al igual que en casos anteriores, preguntamos al usuario si desea guardar los cambios realizados, y según su respuesta (s o n) se guardaran o se cancelarán//
        printf("\nSe va a eliminar al usuario %s-%s-%s-%s-
%s\n",(*vUsuarios)[u].id,(*vUsuarios)[u].nombre,(*vUsuarios)[u].perfil,(*vUsuario
s)[u].login,(*vUsuarios)[u].pass);
        printf(" ¿Desea continuar? Responda con (s/n)\n");
        fflush(stdin);
        scanf("%c",&resp);
    }
```

```
    if(resp=='s'){ //Si la respuesta es s, copiamos en las direcciones donde están los datos del usuario el contenido de una posición vacía, y de esa manera se borra//
        for(i=u;i+1<(*nUsuarios);i++){
            strcpy((*vUsuarios)[i].id,(*vUsuarios)[i+1].id);
            strcpy((*vUsuarios)[i].nombre,(*vUsuarios)[i+1].nombre);
            strcpy((*vUsuarios)[i].perfil,(*vUsuarios)[i+1].perfil);
            strcpy((*vUsuarios)[i].login,(*vUsuarios)[i+1].login);
            strcpy((*vUsuarios)[i].pass,(*vUsuarios)[i+1].pass);
        }

        (*nUsuarios)--; //Se resta en uno el número de usuarios//

        *vUsuarios=realloc((usuario *)(*vUsuarios),(*nUsuarios)*sizeof(usuario)); //Y se disminuye la reserva de espacio para los usuarios//
    }
    else{
        printf("\nLa eliminacion del usuario ha sido cancelada.\n");
    }
}

void listarEquipos(equipo *vEquipos,int nEquipos){
    int i;

    printf("\n<------------EQUIPOS------------>\n");
    printf("ID-NOMBRE\n");

    for(i=0;i<nEquipos;i++){ //El bucle recorre el fichero que contiene los equipos y los imprime por pantalla//
            printf("%s %s\n",vEquipos[i].id,vEquipos[i].nombre);
    }
}

void modificarEquipo(equipo **vEquipos,int nEquipos){
    int i;
    int e=-1;
    char resp=' ';
    char id[3];
    int nModEquipo=0;

    equipo *modEquipo;

    modEquipo=malloc(sizeof(equipo)); //Reservamos espacio para modificar un equipo, con el tamaño de equipo, que es el tipo de dato que definimos para estos//

    printf("\n<------------MODIFICAR EQUIPO------------>\n");

    while(e==-1){
```

```
        printf("\nIntroduzca el ID del equipo que desea modificar: (2 cifras como
maximo");
        scanf("%2s",id);

        for(i=0;i<nEquipos;i++){ //Recorremos el fichero equipos hasta encontrar el
equipo del usuario, y si no se encuentra se mostrará un mensaje de error//
            if(strcmp((*vEquipos)[i].id,id)==0){ //Compara todos los valores del fichero
con el introducido por el usuario//
                e=i;
            }
        }

        if(e==-1){
            printf("\nEquipo no existente.\n");
        }
            else{
            printf("\nInformacion actual del equipo seleccionado --> %s-
%s\n\n",(*vEquipos)[e].id,(*vEquipos)[e].nombre);
            printf("\nPor favor, introduzca la nueva informacion del equipo (siguiendo
ID-NOMBRE)\n\n");

            anadirEquipo(&modEquipo,&nModEquipo);
//Llamo a la función añadir equipo para que recoja los datos para el nuevo equipo//
            while(resp!='s' && resp!='n'){ //Preguntamos al usuario si quiere validar la
modificación//
                printf("¿Seguro que quiere modificar el equipo? Responda con (s/n)\n");
                fflush(stdin);
                scanf("%c",&resp);
            }

            if(resp=='s'){
                (*vEquipos)[e]=modEquipo[0];
            }
                else{
                printf("\nLa modificacion no se guardara, ha sido cancelada.\n");
            }
        }
    }
}



void anadirEquipo(equipo **vEquipos,int *nEquipos){
    *vEquipos=realloc((equipo *)(*vEquipos),((*nEquipos)+1)*sizeof(equipo));
//Aumentamos en uno el tamaño reservado para los equipos para el nuevo//
    printf("\nIntroduzca el ID del equipo (2 cifras como maximo): ");
    scanf("%2s",(*vEquipos)[*nEquipos].id);

    printf("\nAhora, introduzca el nombre del equipo (20 letras como maximo): ");
```

```
    scanf("%20s",(*vEquipos)[*nEquipos].nombre);

    (*nEquipos)++; //Aumentamos en uno el número de equipos//

}

void eliminarEquipo(equipo **vEquipos,int *nEquipos){
    int i;
    int e;
    char resp=' ';
    char id[3];

    printf("\n<------------ELIMINAR EQUIPO------------>\n\n");

    printf("Introduzca el ID del equipo a eliminar (2 cifras como maximo):  ");
    scanf("%2s",id);

    for(i=0;i<(*nEquipos);i++){ //Recorre el fichero equipos y compara cada valor con
el introducido por el usuario//
        if(strcmp((*vEquipos)[i].id,id)==0){
            e=i;
        }
    }

    while(resp!='s' && resp!='n'){ //Preguntamos al usuario si quiere validar la
eliminación//
        printf("\nSe va a eliminar el equipo %s-
%s\n",(*vEquipos)[e].id,(*vEquipos)[e].nombre);
        printf("¿Esta seguro de eliminar el equipo? Responda con (s/n)\n");
        fflush(stdin);
        scanf("%c",&resp);
    }

    if(resp=='s'){ //Si valida la acción, copiamos un espacio vacío en los datos del
equipo en cuestión//
        for(i=e;i+1<(*nEquipos);i++){
            strcpy((*vEquipos)[i].id,(*vEquipos)[i+1].id);
            strcpy((*vEquipos)[i].nombre,(*vEquipos)[i+1].nombre);
        }

        (*nEquipos)--; //Y restamos en uno el número de equipos//

        *vEquipos=realloc((equipo *)(*vEquipos),(*nEquipos)*sizeof(equipo));
    } //Además descartamos la reserva de espacio que realizamos par aun equipo//
    else{
        printf("\nEl equipo no sera eliminado, se ha cancelado el proceso.\n");
    }
}
```

```
void listarFutbolistas(futbolista *vFutbolistas,int nFutbolistas){
   int i;

   printf("\n<------------FUTBOLISTAS------------>\n");
   printf("FORMATO --> ID-ID_EQUIPO-NOMBRE-PRECIO(Millones)-
VALORACION\n");

   for(i=0;i<nFutbolistas;i++){ //Recorremos el fichero futbolistas e imprimimos por
pantalla//
         printf("%s-%s-%s-%d-
%d\n",vFutbolistas[i].id,vFutbolistas[i].ideq,vFutbolistas[i].nombre,vFutbolistas[i].
precio,vFutbolistas[i].valor);
   }
}

void modificarFutbolista(futbolista **vFutbolistas,int nFutbolistas){
   int i;
   int f=-1;
   char resp=' ';
   char id[3];
   int nModFutbolista=0;

   futbolista *modFutbolista;

   modFutbolista=malloc(sizeof(futbolista)); //Reservamos espacio para los nuevos
datos del futbolista//

   printf("\n<------------MODIFICAR FUTBOLISTA------------>\n");

   while(f==-1){
      printf("\nIntroduzca el ID del futbolista que desea modificar (2 cifras como
maximo): ");
      scanf("%2s",id);
      printf("\n");

      for(i=0;i<nFutbolistas;i++){ //Recorremos el fichero futbolistas y lo
comparamos con el vlaor que el usuario a dado//
         if(strcmp((*vFutbolistas)[i].id,id)==0){
            f=i;
         }
      }

      if(f==-1){ //Si no existe ninguna coincidencia se muestra un mensaje de error//
         printf("\nNo existe ningun futbolista asociado a esa ID.");
      }
         else{
```

```
            printf("\nInformacion actual de: %s-%s-%s-%d-
%d\n\n",(*vFutbolistas)[f].id,(*vFutbolistas)[f].ideq,(*vFutbolistas)[f].nombre,(*vF
utbolistas)[f].precio,(*vFutbolistas)[f].valor);
            printf("\nAhora, introduzca la nueva informacion del futbolista segun (ID-
NOMBRE)\n");

            anadirFutbolista(&modFutbolista,&nModFutbolista);
//Llamamos a la función añadir futbolista para que recoja los datos nuevos del
futbolista//
            while(resp!='s' && resp!='n'){ //Preguntamos al usuario si quiere validar la
modificación//
                printf("¿Seguro que quiere guardar los cambios del futbolista? Responda
con (s/n)\n");
                fflush(stdin);
                scanf("%c",&resp);
            }

            if(resp=='s'){ //Si el usuario confirma, se realiza la asignación de los nuevos
datos al futbolista//
                (*vFutbolistas)[f]=modFutbolista[0];
            }
               else{
                printf("\nNo se guardaran los cambios, la operacion ha sido
cancelada.\n");
            }
        }
    }
}

void anadirFutbolista(futbolista **vFutbolistas, int *nFutbolistas){
    *vFutbolistas=realloc((futbolista
*)(*vFutbolistas),((*nFutbolistas)+1)*sizeof(futbolista));
//Aumentamos la reserva de espacio para un nuevo futbolista y recogemos la info de
este//
    printf("\nIntroduzca el ID del futbolista (2 cifras como maximo): ");
    scanf("%2s",(*vFutbolistas)[*nFutbolistas].id);

    printf("\nAhora, introduzca el ID del equipo del futbolista (2 cifras como maximo
ambos): ");
    scanf("%2s",(*vFutbolistas)[*nFutbolistas].ideq);

    printf("\nA continuacion, introduzca el nombre del futbolista (20 caracteres
como maximo): ");
    scanf("%20s",(*vFutbolistas)[*nFutbolistas].nombre);

    printf("\nAhora, marque el precio del futbolista en millones de euros (2 cifras
maximo):");
    scanf("%d",&(*vFutbolistas)[*nFutbolistas].precio);
```

```
    printf("Finalmente, introduzca la valoracion del futbolista en una escala de 0 a
10: ");
    scanf("%d",&(*vFutbolistas)[*nFutbolistas].valor);

    (*nFutbolistas)++; //Aumentamos en uno el número de futbolistas//
}

void eliminarFutbolista(futbolista **vFutbolistas,int *nFutbolistas){
    int i;
    int f;
    char resp=' ';
    char id[3];

    printf("\n<----------ELIMINAR FUTBOLISTA--------->");

    printf("\nIntroduzca el ID del futbolista a eliminar: ");
    scanf("%2s",id);

    for(i=0;i<(*nFutbolistas);i++){  //Recorre el fichero de futbolistas y compara los id
con el introducido por el usuario//
        if(strcmp((*vFutbolistas)[i].id,id)==0){
            f=i;
        }
    }

    while(resp!='s' && resp!='n'){ //Preguntamos al usuario si desea validar la
eliminación//
        printf("\nSe va a eliminar el siguiente usuario: %s-%s-%s-%d-
%d\n",(*vFutbolistas)[f].id,(*vFutbolistas)[f].nombre,(*vFutbolistas)[f].nombre,(*v
Futbolistas)[f].precio,(*vFutbolistas)[f].valor);
        printf("\n¿Esta seguro que desea eliminar? Responda con (s/n)\n");
        fflush(stdin);
        scanf("%c",&resp);
    }

    if(resp=='s'){
        for(i=f;i+1<(*nFutbolistas);i++){ //Copiamos un espacio vacío en donde estaban
los datos del futbolista a eliminar//
            strcpy((*vFutbolistas)[i].id,(*vFutbolistas)[i+1].id);
            strcpy((*vFutbolistas)[i].ideq,(*vFutbolistas)[i+1].ideq);
            strcpy((*vFutbolistas)[i].nombre,(*vFutbolistas)[i+1].nombre);
            (*vFutbolistas)[i].precio=(*vFutbolistas)[i+1].precio;
            (*vFutbolistas)[i].valor=(*vFutbolistas)[i+1].valor;
        }

        (*nFutbolistas)--; //Restamos en uno el número de futbolistas//
```

```
        *vFutbolistas=realloc((futbolista
*)(*vFutbolistas),(*nFutbolistas)*sizeof(futbolista));
    } //Disminuimos el espacio reservado de un futbolista//
    else{
        printf("\nLa eliminacion no sera efectuada, se ha cancelado.\n");
    }
}
```

Nombre	cargar
Precondición	Se llama a la función cargar() en main, que siempre se dará por lo tanto siempre se cumple la precondición.
Descripción	Realizará la función de cargar todos los datos para el sistema al completo.
Postcondición	Se finalice el sistema.

Comentarios del Código

```
#include "cargar.h"

int cargar(){ //Declaramos las funciones que se pueden dar en él//
    cargarUsuarios(&vUsuarios,&nUsuarios);
    cargarEquipos(&vEquipos,&nEquipos);
    cargarFutbolistas(&vFutbolistas,&nFutbolistas);
    cargarPlantillas(&vPlantillas,&nPlantillas);
    cargarJugplants(&vJugplants,&nJugplants);
    cargarConfigs(&vConfigs,&nConfigs);

    return (0);
}

int cargarUsuarios(usuario **vUsuarios,int *n){
    FILE *fichero;

    char linea[57]; //Aquí, cada línea que leemos del fichero lo guardaremos.//
    char *token; //token será un puntero que usaremos para la tokenización//

    *vUsuarios=NULL;
    *n=0;

    fichero=fopen("DATA/usuarios.txt","r"); //DATA/usuarios.txt = Carpeta DATA,
archivo usuarios.txt//

    if (fichero==NULL){
```

```
        printf("\nHa ocurrido un error al abrir el fichero DATA/usuarios.txt, compruebe que los datos son correctos."); //Si no podemos abrir el fichero mostramos un error por pantalla//
    }
    else{
        do{ //Aquí recogemos todos los datos referentes a los usuarios almacenados en el fichero usuarios.txt//
            fgets(linea,57,fichero);

            if (strcmp(linea,"\0")){ //Comparamos el contenido de linea con \0, si es verdad que son iguales significa que el campo está vacío, por el contrario habrá un valor en dicho campo. Además si se cumple que está vacío aumentamos la reserva de espacio para el dato del usuario que el principio asignamos como NULL//
                *vUsuarios=realloc((usuario *)(*vUsuarios),((*n)+1)*sizeof(usuario));

                token=strtok(linea,"-"); //Con strtok separamos la cadena con todos los datos de un usuario en los distintos datos que necesitemos mediante la “señal” asignada “-“ (guion), en este caso es el id. Y lo asignamos a token//
                strcpy((*vUsuarios)[*n].id,token); //Luego de haber separado el dato que necesitamos y haberlo almacenado en token, lo copiamos con strcpy en su posición asignada//

                token=strtok(NULL,"-"); //Nombre de usuario//
                strcpy((*vUsuarios)[*n].nombre,token);

                token=strtok(NULL,"-"); //Perfil de usuario, es decir si es administrador, participante o cronista//
                strcpy((*vUsuarios)[*n].perfil,token);

                token=strtok(NULL,"-"); //Login de usuario//
                strcpy((*vUsuarios)[*n].login,token);

                token=strtok(NULL,"\n"); //Contraseña de usuario//
                strcpy((*vUsuarios)[*n].pass,token);

                (*n)++; //Aumentamos en uno *n cada vez que da una vuelta para ir cambiando entre las líneas que marcan a un nuevo usuario en el fichero//
            }
        }while(!feof(fichero));
    }

    fclose(fichero); //Que no se olvide cerrar el fichero//

    return 0;
}

int cargarEquipos(equipo **vEquipos,int *n){
    FILE *fichero;
```

```
    char linea[25];
    char *token;

    *vEquipos=NULL;
    *n=0;

    fichero=fopen("DATA/equipos.txt","r");

    if (fichero==NULL){ //Si al abrir el fichero da error, se muestra un mensaje por pantalla de error//
        printf("\nHa ocurrido un error al abrir el fichero DATA/equipos.txt, compruebe que los datos son correctos.");
    }
    else{
        do{ //Sino, haremos similar al anterior caso ya que la forma de cargar los distintos datos es la misma, pero cambian los datos. Esta acción no finalizará hasta fin de fichero (FEOF)//
            fgets(linea,25,fichero); //Con fgets almacenamos en linea un máximo de 25 caracteres leídos de fichero, que se corresponde a un puntero que apunta al fichero correspondiente, en este caso equipos.txt//

            if (strcmp(linea,"\0")){ //Comparamos linea con el valor que marca el fin de fichero, "\0", para saber si está vacío o no//
                *vEquipos=realloc((equipo *)(*vEquipos),((*n)+1)*sizeof(equipo)); //Aumentamos en uno la reserva de espacio que realizamos para los equipos//
                token=strtok(linea,"-"); //Al igual que en el anterior, el carácter que nos marcará los distintos datos en la cadena de caracteres es el guion "-"  y lo hacemos con strtok asignándolo a token//
                strcpy((*vEquipos)[*n].id,token); //Luego copiamos el contenido de token en su lugar correspondiente, en este caso el id//
                token=strtok(NULL,"\n"); //Nombre del equipo//
                strcpy((*vEquipos)[*n].nombre,token);

                (*n)++; //Aumentamos en uno el número de equipos por cada vuelta//
            }
        }while(!feof(fichero));
    }

    fclose(fichero);

    return 0;
}

int cargarFutbolistas(futbolista **vFutbolistas,int *n){
    FILE *fichero;
```

```
   char linea[37];
   char *token;

   *vFutbolistas=NULL;
   *n=0;

   fichero=fopen("DATA/futbolistas.txt","r");

   if (fichero==NULL){ //Mostramos este mensaje en caso de error con el fichero//
      printf("\nHa ocurrido un error al abrir el fichero DATA/futbolistas.txt,
compruebe que los datos son correctos.");
   }
   else{
      do{ //Ahora recogemos los datos//
         fgets(linea,37,fichero); //Almacenamos en linea un máximo de 37 caracteres
leídos de fichero//

         if (strcmp(linea,"\0")){ //Comprobamos si el fichero está vacío comparándolo
con "\0"//
            *vFutbolistas=realloc((futbolista
*)(*vFutbolistas),((*n)+1)*sizeof(futbolista));
//Aumentamos en uno la reserva de espacio que realizamos anteriormente para los
futbolistas//
            token=strtok(linea,"-"); //Almacenamos en token el resultado de separar el
primer fragmento de la cadena de caracteres y que corresponde con el id//
            strcpy((*vFutbolistas)[*n].id,token);

            token=strtok(NULL,"-"); //ID equipo del futbolista//
            strcpy((*vFutbolistas)[*n].ideq,token);

            token=strtok(NULL,"-"); //Nombre del jugador//
            strcpy((*vFutbolistas)[*n].nombre,token);

            token=strtok(NULL,"-"); //Precio del futbolista//
            (*vFutbolistas)[*n].precio=atoi(token);

            token=strtok(NULL,"\n"); //Valoración del cronista del futbolista//
            (*vFutbolistas)[*n].valor=atoi(token); //Con la función atoi convertimos la
valoración del cronista en un valor numérico//

            (*n)++; //Aumentamos en uno el número de futbolistas//
         }
      }while(!feof(fichero));
   }

   fclose(fichero);

   return 0;
```

```
}

int cargarPlantillas(plantilla **vPlantillas,int *n){
   FILE *fichero;

   char linea[44];
   char *token;

   *vPlantillas=NULL;
   *n=0;

   fichero=fopen("DATA/plantillas.txt","r");

   if (fichero==NULL){ //Imprimimos este mensaje en caso de error con el fichero//
      printf("\nHa ocurrido un error al abrir el fichero DATA/plantillas.txt,
compruebe que los datos son correctos.");
   }
   else{
      do{ //Recogemos los distintos datos de las plantillas almacenando como
máximo 44 caracteres de la cadena en linea leyéndolo de fichero//
         fgets(linea,44,fichero);

         if (strcmp(linea,"\0")){ //Comprobamos que el fichero no está vacío
comparándolo con el carácter de fin de fichero \0//
            *vPlantillas=realloc((plantilla *)(*vPlantillas),((*n)+1)*sizeof(plantilla));
//Aumentamos en uno el espacio que reservamos para las plantillas//
            token=strtok(linea,"-"); //Almacenamos en token el id del propietario de la
plantilla separado de la cadena mediante strtok con el guion "-" y copiado en su
respectivo espacio con strcpy//
            strcpy((*vPlantillas)[*n].idprop,token);

            token=strtok(NULL,"-"); //ID de plantilla//
            strcpy((*vPlantillas)[*n].idplant,token);

            token=strtok(NULL,"-"); //Nombre de la plantilla//
            strcpy((*vPlantillas)[*n].nombre,token);

            token=strtok(NULL,"-"); //Presupuesto del propietario de la plantilla//
            (*vPlantillas)[*n].presup=atoi(token);

            token=strtok(NULL,"\n");
            (*vPlantillas)[*n].punt=atoi(token); //Puntuación de la plantilla//

            (*n)++; //Aumentamos en uno el número de plantilla//
         }
      }while(!feof(fichero));
   }
```

```
    fclose(fichero);

    return 0;
}

int cargarJugplants(jugplant **vJugplants,int *n){
    FILE *fichero;

    char linea[12];
    char *token;

    *vJugplants=NULL;
    *n=0;

    fichero=fopen("DATA/jugadores_plantillas.txt","r");

    if (fichero==NULL){ //Mensaje de error con el fichero a usuario//
        printf("\nHa ocurrido un error al abrir el fichero
DATA/jugadores_plantillas.txt, compruebe que los datos son correctos.");
    }
    else{ //Recogemos los datos de los jugadores de la plantilla//
        do{
            fgets(linea,12,fichero); //Almacenamos en linea un máximo de 12 caracteres
leídos de fichero//

            if (strcmp(linea,"\0")){ //Comprobamos si el fichero está vacío comparándolo
con el carácter de fin de fichero \0//
                *vJugplants=realloc((jugplant *)(*vJugplants),((*n)+1)*sizeof(jugplant));
//Aumentamos en uno la reserva de espacio para los jugadores de una plantilla//
                token=strtok(linea,"-"); //Carácter de distinción entre datos es el guion//
                strcpy((*vJugplants)[*n].idjug,token); //Copiamos el dato almacenado en
token en su lugar correspondiente, en este caso el id del jugador//

                token=strtok(NULL,"-"); //ID del equipo//
                strcpy((*vJugplants)[*n].ideq,token);

                token=strtok(NULL,"\n"); //ID de la plantilla//
                strcpy((*vJugplants)[*n].idplant,token);

                (*n)++; //Aumentamos en uno el número de jugadores de una plantilla//
            }
        }while(!feof(fichero));
    }

    fclose(fichero);

    return 0;
}
```

```
int cargarConfigs(config **vConfigs,int *n){
   FILE *fichero;

   char linea[35];
   char *token;

   *vConfigs=NULL;
   *n=0;

   fichero=fopen("DATA/configuracion.txt","r");

   if (fichero==NULL){ //Mensaje de error al abrir el fichero//
      printf("\nHa ocurrido un error al abrir el fichero DATA/configuracion.txt,
compruebe que los datos son correctos.");
   }
   else{ //Recogemos los datos necesarios//
      do{
         fgets(linea,35,fichero); //Almacenamos en linea un máximo de 35 caracteres
leídos de fichero//

         if (strcmp(linea,"\0")){ //Comprobamos si el fichero está vacío comparándolo
con el carácter de fin de fichero \0//
            *vConfigs=realloc((config *)(*vConfigs),((*n)+1)*sizeof(config));
//Aumentamos en uno la reserva de espacio que realizamos para la configuración//
            token=strtok(linea,"-"); //Almacenamos en token la configuración//
            strcpy((*vConfigs)[*n].config,token);

            token=strtok(NULL,"\n"); //Valor de la configuración asignada por el
administrador//
            (*vConfigs)[*n].valor=atoi(token);

            (*n)++;
         }
      }while(!feof(fichero));
   }

   fclose(fichero);

   return 0;
}
```

Nombre	confPlant
Precondición	Se elija el menú de configuración de plantilla en la función menus.
Descripción	Realiza las funciones de configuración de una plantilla, principalmente realizada por el participante.
Postcondición	El usuario salga del menú configuración de la plantilla.

Comentarios del Código

```
#include "confPlant.h"


void listarFutbPlant(char *idPlant, jugplant *vJugplants, int nJugplants){
    for(int i=0;i<nJugplants;i++){ //Con este for vamos pasando por los distintos datos de los id de plantilla//
    if(strcmp(vJugplants[i].idplant,idPlant) == 0){ //Comparamos el id introducido por el usuario con el id correspondiente a la vuelta del for//
        printf("\n%s-%s-%s\n", vJugplants[i].idjug, vJugplants[i].ideq, vJugplants[i].idplant);
    }
    }
}


int esFutbTitular(char * idFutb, jugplant *vJugplants, int nJugplants ){
    for(int i=0;i<nJugplants;i++){
//Con este for vamos pasando el valor de i para pasar los id de los jugadores hasta encontrar mediante la comparación de abajo el coincidente con el id introducido por el usuario//
    if(strcmp(vJugplants[i].idjug, idFutb) == 0){
        return 1;
    }
    }
    return 0;
```

```
}


void listarFutbDisp(char *idPlant, jugplant *vJugplants, int nJugplants, futbolista
*vFutbolistas , int nFutbolistas){

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

//Listamos a los futbolista restantes mediante su id//

   if(!esFutbTitular(vFutbolistas[i].id, vJugplants, nJugplants) )

      printf("%s-%s-%s-%d-%d\n", vFutbolistas[i].id, vFutbolistas[i].ideq,
vFutbolistas[i].nombre, vFutbolistas[i].precio, vFutbolistas[i].valor);

   }

}

void ficharFutb(char *idPlant, jugplant **vJugplants, int *nJugplants, futbolista
*vFutbolistas , int nFutbolistas){

   char ideq[3], idFutb[3];


   printf("\nIntroduzca el ID del jugador a fichar(3 cifras como maximo): ");

   fflush(stdin);

   scanf("%2s",idFutb);


   (*nJugplants)++; //Aumentamos en uno el número de jugadores de plantilla//

   *vJugplants=realloc((jugplant *)(*vJugplants),(*nJugplants)*sizeof(jugplant));

//Aumentamos el espacio reservado para los jugadores de una plantilla//

   for(int i=0;i<nFutbolistas;i++)


   if( strcmp(vFutbolistas[i].id, idFutb) == 0 ){ //Comparamos el id introducido por
el usuario con el id de la vuelta marcado por i en el for//

      strcpy(ideq, vFutbolistas[i].ideq); //Cuando lo encuentra copia el id del equipo
en su lugar correspondiente//

   }

//Copiamos los datos del jugador la plantilla y restamos el número de los jugadores
disponibles//

   strcpy( (*vJugplants)[(*nJugplants) - 1].idjug, idFutb);

   strcpy( (*vJugplants)[(*nJugplants) - 1].ideq, ideq);

   strcpy( (*vJugplants)[(*nJugplants) - 1].idplant, idPlant);
```

```
}

void despedirFutb(char *idPlant, jugplant **vJugplants, int *nJugplants,
futbolista *vFutbolistas , int nFutbolistas){

   int i = 0;

   char idFutb[3], resp=' ';


   printf("\nIntroduzca el ID del jugador que desea despedir:(3 cifras como
maximo) ");

   fflush(stdin);

   scanf("%2s",idFutb);

//Comparamos el id del jugador o el id de plantilla con todos los jugadores de la
plantilla mediante una variable i que va aumentando en uno por cada vuelta que
haga el bucle hasta encontrarlo//

   while((strcmp((*vJugplants)[i].idjug, idFutb)!= 0 ||
strcmp((*vJugplants)[i].idplant, idPlant)!= 0) && i <(*nJugplants)){

      i++;

   }


   while(resp!='s' && resp!='n'){ //Validamos que el usuario quiera despedir al
jugador de su plantilla permanentemente//

         printf("\nSe va a proceder a eliminar al jugador con ID %s de la plantilla
%s ", idFutb, idPlant);

         printf("\n¿Esta seguro que desea eliminar al jugador de la plantilla?
Responda con (s/n)\n");

         fflush(stdin);

         scanf("%c",&resp);

   }


   if(resp=='s'){ //Si el usuario valida el despido, eliminamos los datos del jugador de
la plantilla y copiamos los datos a los jugadores disponibles, también aumentamos
en uno el número de jugadores disponibles//

         for(i;i+1<(*nJugplants);i++){

            strcpy( (*vJugplants)[i].idjug,(*vJugplants)[i+1].idjug );

            strcpy((*vJugplants)[i].ideq,(*vJugplants)[i+1].ideq);
```

```
            strcpy((*vJugplants)[i].idplant,(*vJugplants)[i+1].idplant);
        }


        (*nJugplants)--; //Restamos en uno el número de jugadores de la plantilla//
        *vJugplants=realloc((jugplant *)(*vJugplants),(*nJugplants)*sizeof(jugplant)); //Devolvemos el espacio reservado para ese jugador de la plantilla//
    }
    else{
        printf("\nLa eliminacion del jugador de su plantilla no sera guardada, se ha cancelado el proceso.\n");
    }
}
```

Nombre	cronista
Precondición	El usuario sea identificado como cronista y entre en el sistema como tal.
Descripción	Realiza las funciones designadas al cronista.
Postcondición	El usuario salga del menú cronista.

Comentarios del Código

```
#include "tipos.h"

#include "cargar.h"

#include "cronista.h"

#include "menus.h"

#include "admin.h"


void valorarEquipos(futbolista **vFutbolistas,int nFutbolistas,equipo
*vEquipos,int nEquipos){

    int i, e, f;

        //e=indice para identificar el equipo seleccionado


    char ideq[3]="";

    char idfut[3]="";

    int val=0;


    printf("\nPara valorar un equipo seleccione su ID acontinuacion:(0=volver a
menu) ");

    scanf("%2s",ideq);


    if(strcmp(ideq,"0")==0){

        menuCronista();

    }

    else{
```

```
        for(i=0;i<nEquipos;i++){ //Busca el indice del equipo y lo guarda en 'e'.
            if(strcmp(vEquipos[i].id,ideq)==0){
                e=i;
               }
        }

        while(strcmp(idfut,"0")!=0){
            printf("\n<__________%s__________>\n",vEquipos[e].nombre);

            for(i=0;i<nFutbolistas;i++){
            // A continuación se mostraran por pantalla todos los futbolistas que pertenecen al equipo seleccionado antes
                if(strcmp((*vFutbolistas)[i].ideq,vEquipos[e].id)==0){
                    printf("%s %s\n",(*vFutbolistas)[i].id,(*vFutbolistas)[i].nombre);
                      }
            }

            // Ahora se realizará la valoración de los futbolistas

            printf("\nA continuacion introduzca el ID del futbolista que quiere valorar:(0=volver a menu)  ");
            scanf("%2s",idfut);

            if(strcmp(idfut,"0")!=0){
                for(i=0;i<nFutbolistas;i++){
            //Realizamos este bucle para ver que id de futbolista es el introducido por el usuario
                    if(strcmp((*vFutbolistas)[i].id,idfut)==0){
                        f=i;
                          }
                }

                do{
```

```
                if(val<0 || val>10){
                   printf("\nPor favor, introduzca una valoracion comprendida entre 0 y 10, sino se repetira esta accion.\n");
                    }

                printf("\nLa valoracion de %s es %d\n",(*vFutbolistas)[f].nombre,(*vFutbolistas)[f].valor);

                printf("\nIntroduzca la nueva valoracion de %s (numero de 0 a 10): ",(*vFutbolistas)[f].nombre);
                scanf("%d",&val);
            }while(val<0 || val>10);

actualizarPlantillas(vJugplants,nJugplants,&vPlantillas,nPlantillas,idfut,val);
        // Necesitamos una función para guardar en ella las nuevas valoraciones
            (*vFutbolistas)[f].valor=val;

            printf("\nLa valoracion actualizada de %s es %d\n",(*vFutbolistas)[f].nombre,(*vFutbolistas)[f].valor);
        }

        system("pause");

        system("cls");
      }
   }
   system("pause");
}

void actualizarPlantillas(jugplant *vJugplants,int nJugplants,plantilla **vPlantillas,int nPlantillas,char *idfut,int newVal){
   char idplant[4]="";
   int i;
```

```
    for(i=0;i<nJugplants;i++){
        if(strcmp(vJugplants[i].idjug,idfut)==0){       //Comparamos
            strcpy(idplant,vJugplants[i].idplant);       //Copiamos el contenido de
vJugplants[i].idplant en idplant
          }
    }


    if(strcmp(idplant,"")!=0){
        for(i=0;i<nPlantillas;i++){
            if(strcmp((*vPlantillas)[i].idplant,idplant)==0){
                (*vPlantillas)[i].punt=(*vPlantillas)[i].punt+newVal;
              }
        }
    }
}
```

Nombre	guardar
Precondición	Se llame a la función en main, que es estrictamente obligatorio, por lo que siempre se llamará a esta.
Descripción	Realiza la función de guardar los cambios realizados por los distintos usuarios de los datos que estén a su disposición dependiendo del perfil de usuario que sea.
Postcondición	Finalice el programa main.

Comentarios del Código

```
#include "guardar.h"


void guardarUsuario(usuario *u, size_t tam){
    FILE *fich;
    int i;


    fich=fopen("DATA/usuarios.txt","w+");


    i = 0;


    if(fich==NULL){ //Mensaje de error al abrir el fichero mostrado al usuario//
        printf("\nNo se ha podido editar el fichero debido a un error. Intentelo mas adelante.\n");
    }
    else{
        while(i<tam-1){ //tam es el tamaño de la cadena del dato en cuestión, en este caso usuario, y usamos tam-1 ya que contamos con que al final de la cadena esté el carácter de fin de cadena, \0//
            fprintf(fich, "%s-%s-%s-%s-%s\n", u[i].id, u[i].nombre, u[i].perfil, u[i].login, u[i].pass);
            i++; //Aumentamos en uno el valor de i para que pase de vuelta//
        }


        fprintf(fich, "%s-%s-%s-%s-%s", u[i].id, u[i].nombre, u[i].perfil, u[i].login, u[i].pass );
```

```
    }

    printf("\nLos usuarios han sido guardados correctamente.\n");
}

void guardarEquipo(equipo *e, size_t tam){
    FILE *fich;
    int i;

    fich=fopen("DATA/equipos.txt","w+");

    i = 0;

    if(fich==NULL){ //Mensaje de error al abrir el fichero mostrado al usuario//
        printf("\nNo se ha podido editar el fichero debido a un error. Intentelo mas
adelante.\n");
    }
    else{ //Al igual que antes, usamos tam para sobrescribir los datos usando su
tamaño específico de dato, que en este caso es nombre de equipo//
        while(i<tam-1){
            fprintf(fich, "%s-%s\n", e[i].id, e[i].nombre );
            i++;
        }

            fprintf(fich, "%s-%s", e[i].id, e[i].nombre );
    }

    printf("\nLos equipos han sido guardados correctamente.\n");
}

void guardarFutbolistas(futbolista *f, size_t tam){
    FILE *fich;
    int i;
```

```
    fich=fopen("DATA/futbolistas.txt","w+");


    i = 0;


    if(fich==NULL){ //Mensaje de error al abrir el fichero mostrado al usuario//
        printf("\nNo se ha podido editar el fichero debido a un error. Intentelo mas
tarde.\n");
    }
    else{ //Al igual que antes, usamos tam para sobrescribir los datos usando su
tamaño específico de dato, que en este caso es futblista//
        while(i<tam-1){
            fprintf(fich, "%s-%s-%s-%d-%d\n", f[i].id, f[i].ideq, f[i].nombre, f[i].precio,
f[i].valor );
            i++;
        }


            fprintf(fich, "%s-%s-%s-%d-%d", f[i].id, f[i].ideq, f[i].nombre, f[i].precio,
f[i].valor );
    }


    printf("\nLos futbolistas han sido guardados correctamente.\n");
}


void guardarPlantillas(plantilla *p, size_t tam){
    FILE *fich;
    int i;


    fich=fopen("DATA/plantillas.txt","w+");


    i = 0;


    if(fich==NULL){ //Mensaje de error al barir el fichero mostrado al usuario//
        printf("\nNo se ha podido editar el fichero debido a un error. Intentelo mas
tarde.\n");
    }
```

```
    else{ //Al igual que anteriormente, usamos tam para sobrescribir los datos usando su tamaño específico de dato, que en este caso es plantilla//
        while(i<tam-1){
            fprintf(fich, "%s-%s-%s-%d-%d\n", p[i].idprop, p[i].idplant, p[i].nombre, p[i].presup, p[i].punt );
            i++;
        }
        fprintf(fich, "%s-%s-%s-%d-%d", p[i].idprop, p[i].idplant, p[i].nombre, p[i].presup, p[i].punt );
    }
    printf("\nLa plantilla ha sido guardada correctamente.\n");
}

void guardarJugPlant(jugplant *jp, size_t tam){
    FILE *fich;
    int i;

    fich=fopen("DATA/jugadores_plantillas.txt","w+");

    i = 0;

    if(fich==NULL){ //Mensaje de error al abrir el fichero mostrado al usuario//
        printf("\nNo se ha podido editar el fichero debido a un error. Intentelo mas tarde.\n");
    }
    else{ //Usamos tam para sobrescribir los datos usando su tamaño específico de dato, que en este caso es jugadores de plantilla//
        while(i<tam-1){
            fprintf(fich, "%s-%s-%s\n", jp[i].idjug, jp[i].ideq ,jp[i].idplant );
            i++;
        }

        fprintf(fich, "%s-%s-%s", jp[i].idjug, jp[i].ideq ,jp[i].idplant );
    }
```

```
    printf("\nLos datos de jugadoresplantilla se han guardado correctamente.\n");
}

void guardarConf(config *c, size_t tam){
    FILE *fich;
    int i;

    fich=fopen("DATA/configuracion.txt","w+");

    i = 0;

    if(fich==NULL){ //Mensaje de error al abrir el fichero mostrado al usuario//
        printf("\nNo se ha podido editar el fichero debido a un error. Intentelo mas tarde.\n");
    }
    else{ //Usamos tam para sobrescribir los datos usando su tamaño específico de dato, que en este caso es configuración//
        while(i<tam-1){
            fprintf(fich, "%s-%d\n", c[i].config, c[i].valor );
            i++;
        }

        fprintf(fich, "%s-%d", c[i].config, c[i].valor );
    }

    printf("\nLa configuración ha sido guardada correctamente.\n");
}

void guardar(){
    char resp=' ';

    while(resp!='s' && resp!='n'){ //Validamos que el usuario desea guardar los datos al salir del sistema//
      printf("\nHa seleccionado salir. ¿Antes de salir desea guardar los cambios realizados? Responda con (s/n)\n");
```

```
        fflush(stdin);
        scanf("%c",&resp);
    }

    if(resp=='s'){ //Si introduce una s realizamos las operaciones pertinentes para guardar todos los datos que hayan podido ser cambiados//
        guardarUsuario(vUsuarios, nUsuarios);
        guardarEquipo(vEquipos, nEquipos );
        guardarFutbolistas(vFutbolistas, nFutbolistas);
        guardarPlantillas(vPlantillas, nPlantillas);
        guardarJugPlant(vJugplants, nJugplants);
        guardarConf(vConfigs, nConfigs);

        exit(0); //Salimos del sistema para finalizar el programa en el main//
    }
    else{ //Mensaje informativo de que los datos no serán guardados al introducir una n para cancelar que se guarden los datos al salir//
        printf("\nLos cambios realizados no han sido guardados.\n");
    }
}
```

Nombre	main
Precondición	Se inicie el programa.
Descripción	Realiza las llamadas a funciones vitales para el sistema.
Postcondición	Se finalice el programa.

Comentarios del Código

```
#include <stdio.h>
#include <stdlib.h>
#include "menus.h"
#include "cargar.h"
#include "guardar.h"
#include "participante.h"
//Incluímos todos los ficheros de cabecera que nos hace falta para el main (ojo wue no hace falta meter admin ni cronista//


int main(){
    cargar(); //Llamamos a la función cargar para que cargue los datos//


    menuInicio(); //Llamamos a la función menuInicio para que despliegue el menú inicial independientemente de qué perfil de usuario sea//


    guardar(); //Llamamos a la función guardar para que guarde los datos que hayan podido ser sobrescritos por el usuario//


    return (0);
}
```

Nombre	menus
Precondición	Se llame a la función menu en alguno de los archivos dependiendo del perfil de usuario y del menú que se llame.
Descripción	Realiza la función de desplegar los distintos menús con las opciones correspondientes en cada caso.
Postcondición	Salir del menú inicial, ya que es donde se encuentra la opción para salir del sistema.

Comentarios del Código

```
#include "tipos.h"
#include "cargar.h"
#include "registro.h"
#include "acceso.h"
#include "menus.h"
#include "participante.h"
#include "cronista.h"
#include "admin.h"
#include "guardar.h"
#include "confPlant.h"


void menuInicio(){
   int n;

   while(n!=3){ //Cuando el usuario elija el 3, se saldrá del menú//
      system("cls"); //Limpiamos la pantalla de las instrucciones anteriores//

      printf("\nBienvenido a la Liga Fantastica MP\n");
      printf("<*-*-*-*-*-*-*-*-*-*-*-*-*-*-*->\n");
      printf("\nAcciones:(Responda con el numero)\n");
      printf("1. Registrarse");
      printf("\n2. Acceder al sistema");
      printf("\n3. Salir\n");
      scanf("%d",&n);

      system("cls");

      switch(n){ //Según la opción elegida por el usuario se llamará a la función encargada de realizar la tarea correspondiente//
         case 1:
            anadirUsuario(&vUsuarios,&nUsuarios,0); //Añadir usuario nuevo//
            break;

         case 2:
            acceso(vUsuarios,nUsuarios); //Acceder al sistema//
            break;
```

```
            case 3: //Salir//
                break;
        }
    }
}

void menuParticipante(char *id){
    int n=0;

    contarPlantillas(id,vPlantillas,nPlantillas,&nPlantillasUsuario);
//Llamamos a la función contarPlantilla para que pueda servir a las funciones venideras//
    system("pause");

    while(n!=6){ //Se saldrá del menú cuando el usuario introduzca 6//
        system("cls");

        printf("\n|Usuario: PARTICIPANTE|");
        printf("\n<------------------------------>\n");
        printf("Selecciona una accion:(numero) \n");
        printf("\n1. Crear plantilla");
        printf("\n2. Configurar plantilla");
        printf("\n3. Lista de plantillas");
        printf("\n4. Eliminar Plantilla");
        printf("\n5. Ranking");
        printf("\n6. Salir\n");
        scanf("%d",&n);

        system("cls");

        switch(n){ //Se realizan las llamadas a las funciones encargadas para las tareas que se ofertan en el menú del usuario en cuestión//
            case 1: //Crear una plantilla nueva//
                printf("\n|CREAR PLANTILLA NUEVA|\n\n");

crearPlantilla(id,&vPlantillas,&nPlantillas,&vJugplants,&nJugplants,&nPlantillasUsuario);
                    system("pause");
                break;

            case 2: //Modificar una plantilla//
                printf("\n|MODIFICAR PLANTILLA|\n\n");
                menuParticipantePlantilla(id);
                system("pause");
                break;

            case 3: //Mostrar un listado de las plantillas del usuario//
                printf("\n|<--LISTADO DE PLANTILLAS-->|\n");
                printf("\nID_PROPIETARIO - ID_PLANTILLA - NOMBRE - PRESUPUESTO - PUNTUACION\n");
                listarplantillas(id, vPlantillas, nPlantillas, nPlantillasUsuario);
                system("pause");
                break;

            case 4: //Eliminar una plantilla//
                printf("\n|X ELIMINAR PLANTILLA X|\n");
```

```
            eliminarPlantilla(id,&vPlantillas,&nPlantillas,&nPlantillasUsuario);
            system("pause");
            break;

        case 5: //Mostrar un listado con el top 3 mejores plantillas//
            printf("\n[:::::::TOP 3:::::::]\n");
            ranking(vPlantillas,nPlantillas);
            system("pause");
            break;

        case 6: //Salir//
            break;
    }
  }

  guardar(); //Guardamos los datos//
}

void menuParticipantePlantilla(char *id){
  int n=0;
  char idPlant[4];

  printf("\nIntroduce el ID de su plantilla: ");
  listarplantillas(id, vPlantillas, nPlantillas, nPlantillasUsuario);

  fflush(stdin);
  scanf("%3s",idPlant);

  system("pause");

  while(n!=5){ //Se saldrá del menú cuando el usuario introduzca 5//
    system("cls");

    printf("\n# CONFIGURACION DE PLANTILLA #");
    printf("\nIntroduzca una opcion:(numero) ");
    printf("\n1. Lista de futbolistas de la plantilla actual");
    printf("\n2. Lista de futbolistas disponibles");
    printf("\n3. Anadir futbolista a la plantilla");
    printf("\n4. Eliminar jugador de plantilla");
    printf("\n5. Salir\n");

    fflush(stdin);
    scanf("%d",&n);

    system("cls");

    switch(n){ //Se realizan las llamadas a las funciones encargadas de las tareas
correspondientes al menú//
        case 1: //Mostrar un listado de los futbolistas de la plantilla actual//
            printf("\nLista de futbolistas de la plantilla actual:\n");
            listarFutbPlant(idPlant, vJugplants, nJugplants);
            system("pause");
            break;

        case 2: //Mostrar un listado de los futbolistas disponibles//
            printf("\nLista de futbolistas disponibles:\n");
```

```
            listarFutbDisp(idPlant, vJugplants, nJugplants, vFutbolistas,
nFutbolistas);
            system("pause");
            break;

         case 3: //Añadir futbolistas a la plantilla//
            printf("\nAnadir futbolista a la plantilla: ");
            ficharFutb(idPlant,&vJugplants, &nJugplants, vFutbolistas,
nFutbolistas);
            system("pause");
            break;

         case 4: //Eliminar un jugador de la plantilla//
            printf("\nEliminar jugador de la plantilla:");
            despedirFutb(idPlant, &vJugplants, &nJugplants, vFutbolistas
,nFutbolistas);
            system("pause");
            break;

           case 5: //Salir//
                 break;

      }
   }
}

void menuCronista(){
   int n=0;

   system("cls");

   while(n!=3){ //Se saldrá del menú cuando el usuario introduzca 3//
      printf("\n|MENU CRONISTA|\n");;
      printf("\nSeleccione una accion:(numero) ");
      printf("\n1. Listar equipos");
      printf("\n2. Valorar Equipos");
      printf("\n3. Salir del sistema.\n");
      scanf("%d",&n);

      system("cls");

      switch(n){ //Se realizan las llamadas a las funciones encargadas de las tareas
correspondientes a las opciones indicadas en el menú//
         case 1: //Mostrar un listado de los equipos//
            listarEquipos(vEquipos,nEquipos);
            system("pause");
            system("cls");
            break;

         case 2: //Valorar equipos//
            valorarEquipos(&vFutbolistas,nFutbolistas,vEquipos,nEquipos);
            system("cls");
            break;

         case 3: //Salir//
            break;
```

```
        }
    }

    guardar(); //Guardar los cambios que haya podido realizar el usuario//
}

void menuAdministrador(){
    int n=0;

    while(n!=4){ //Se saldrá del menú cuando el usuario introduzca 4//
        system("cls");

        printf("\n|+ MENU ADMINISTRADOR +|\n");
        printf("\nSeleccione una opcion:(numero) ");
        printf("\n1. Equipos");
        printf("\n2. Usuarios");
        printf("\n3. Configuracion");
        printf("\n4. Salir del programa\n");

        scanf("%d",&n);

        system("cls");

        switch(n){ //Llamamos a las funciones que realizaran las tareas pertinentes
según las opciones mostradas al usuario//
            case 1: //Mostrar el menú administrador de equipos//
                menuAdminEquipos();
                break;

            case 2: //Mostrar menú administrador de usuarios//
                menuAdminUsuarios();
                break;

            case 3: //Editar la configuración//
                editConfig(&vConfigs,nConfigs);
                break;

            case 4: //Salir//
                break;
        }
    }

    guardar(); //Guardar los datos que hayan podido ser editados por el usuario//
}

void menuAdminEquipos(){
    int n=0;

    while(n!=9){ //Se saldrá del menú cuando el usuario introduzca 9//
        system("cls");

        printf("\n#-------ADMINISTRAR EQUIPOS-------#\n");
        printf("\nSeleccione una opcion:(numero) ");
        printf("\n1. Listar equipos");
        printf("\n2. Modificar equipo");
        printf("\n3. Anadir equipo");
```

```
        printf("\n4. Eliminar equipo");
        printf("\n5. Listar futbolistas");
        printf("\n6. Modificar futbolista");
        printf("\n7. Anadir futbolista");
        printf("\n8. Eliminar futbolista");
        printf("\n9. Volver al menu administrador\n");

        scanf("%d",&n);

        system("cls");

        switch(n){ //Llamamos a las funciones que realizan las tareas ofertadas en el menú al usuario//

            case 1: //Mostrar listado de los equipos//
                listarEquipos(vEquipos,nEquipos);
                system("pause");
                break;

            case 2: //Modificar un equipo//
                modificarEquipo(&vEquipos,nEquipos);
                system("pause");
                break;

            case 3: //Añadir nuevo equipo//
                printf("\nANADIR NUEVO EQUIPO\n");
                anadirEquipo(&vEquipos,&nEquipos);
                system("pause");
                break;

            case 4: //Eliminar un equipo//
                eliminarEquipo(&vEquipos,&nEquipos);
                system("pause");
                break;

            case 5: //Mostrar listado de futbolistas//
                listarFutbolistas(vFutbolistas,nFutbolistas);
                system("pause");
                break;

            case 6: //Modificar un futbolista//
                modificarFutbolista(&vFutbolistas,nFutbolistas);
                system("pause");
                break;

            case 7: //Añadir nuevo futbolista//
                printf("\nANADIR NUEVO FUTBOLISTA\n");
                anadirFutbolista(&vFutbolistas,&nFutbolistas);
                system("pause");
                break;

            case 8: //Eliminar un futbolista//
                eliminarFutbolista(&vFutbolistas,&nFutbolistas);
                system("pause");
                break;
```

```
            case 9: //Salir//
                break;
        }
    }
}

void menuAdminUsuarios(){
    int n=0;

    while(n!=5){ //Se saldrá del menú cuando el usuario introduzca 5//
        system("cls");

        printf("\n#-------ADMINISTRAR USUARIOS-------#\n");
        printf("\nSeleccione una opcion:(numero) ");
        printf("\n1. Listar usuarios");
        printf("\n2. Modificar usuario");
        printf("\n3. Anadir usuario");
        printf("\n4. Eliminar usuario");
        printf("\n5. Volver al menu administrador\n");

        scanf("%d",&n);

        system("cls");

        switch(n){ //Se llaman a las funciones encargadas de las tareas
correspondientes a las mostradas al usuario//
            case 1: //Mostrar listado de usuarios//
                listarUsuarios(vUsuarios,nUsuarios);
                system("pause");
                break;

            case 2: //Modificar un usuario//
                modificarUsuario(&vUsuarios,nUsuarios);
                break;

            case 3: //Registrar un nuevo usuario//
                printf("\nREGISTRAR NUEVO USUARIO\n");
                anadirUsuario(&vUsuarios,&nUsuarios,1);
                system("pause");
                break;

            case 4: //Eliminar un usuario//
                eliminarUsuario(&vUsuarios,&nUsuarios);
                system("pause");
                break;

            case 5: //Salir//
                break;
        }
    }
}
```

Nombre	participante
Precondición	El usuario accede al sistema como participante.
Descripción	Realiza las funciones correspondientes al perfil de usuario participante.
Postcondición	El usuario salga del menú participante.

Comentarios del Código

```
#include "participante.h"

#include "admin.h"


void contarPlantillas(char *id,plantilla *vPlantillas,int nPlantillas,int
*nPlantillasUsuario){

   *nPlantillasUsuario=0;


   for(int i=0;i<nPlantillas;i++){ //Comparamos el id introducido por el usuario con
los distintos id que el for va pasando del fichero plantillas, cuando se encuentra
una coincidencia se suma 1 al número de plantillas//

      if(strcmp(vPlantillas[i].idprop,id)==0){

         (*nPlantillasUsuario)++;

      }

   }

}


void crearPlantilla(char *id,plantilla **vPlantillas,int *nPlantillas,jugplant
**vJugplants,int *nJugplants,int *nPlantillasUsuario){

   if(*nPlantillasUsuario<vConfigs[0].valor){ //Si el número de plantillas del
usuario es menor al número máximo de plantillas permitidas mostrado en la
configuración significa que el usuario puede crear otra plantilla//

      *vPlantillas=realloc((plantilla
*)(*vPlantillas),((*nPlantillas)+1)*sizeof(plantilla));

//Aumentamos en uno el espacio reservado para las plantillas, y seguidamente
recogemos los datos necesarios para la nueva plantilla//

      printf("\nIntroduzca el ID de la plantilla (Max. 3 cifras): ");
```

```
        fflush(stdin);
        scanf("%3s",(*vPlantillas)[*nPlantillas].idplant);

        printf("\nIntroduzca el nombre completo de la plantilla (Max. 30 letras): ");
        fflush(stdin);
        //scanf("%30s",(*vPlantillas)[*nPlantillas].nombre); es otra opción//
        fgets((*vPlantillas)[*nPlantillas].nombre,31,stdin);

        strcpy((*vPlantillas)[*nPlantillas].idprop,id); //Copiamos el id en el id del propietario de la plantilla//

        (*vPlantillas)[*nPlantillas].presup=vConfigs[1].valor; //Se asigna el presupuesto por defecto a la nueva plantilla//

        (*vPlantillas)[*nPlantillas].punt=0; //Asignamos una puntuación inicial de 0//

        (*nPlantillas)++; //Aumentamos en uno el número de plantillas//
        (*nPlantillasUsuario)++; //Aumentamos en uno el número de plantillas del usuario//

    }
    else{ //Si el cupo de plantillas está completo, se le muestra un mensaje de error//
        printf("\nNo puede crear nuevas plantillas. Elimine alguna para poder continuar.");
    }
}

void listarplantillas(char *id, plantilla*vPlantillas, int nPlantillas, int nPlantillasUsuario){
    if( nPlantillasUsuario == 0) //Si el número de plantillas del usuario es 0, mostramos un mensaje de error//
        printf("\nNo tienes plantillas registradas.\n");
    else
        for(int i=0;i<nPlantillas;i++){ //Si tiene plantilla, comparamos el id del usuario con el id del propietario de cada plantilla para hallar la correspondiente y la imprimimos//
```

```
        if(strcmp(vPlantillas[i].idprop,id) == 0)
            printf("%s-%s-%s-%d-
%d\n",vPlantillas[i].idprop,vPlantillas[i].idplant,vPlantillas[i].nombre,vPlantillas[i
].presup,vPlantillas[i].punt);
    }
}

void eliminarPlantilla(char *id, plantilla **vPlantillas, int *nPlantillas, int
*nPlantillasUsuario){
   int p, i;
   char resp=' ';
   char idplant[4];

   printf("\nIntroduzca el ID de la plantilla que desea eliminar: ");
   scanf("%3s",idplant);

   if( nPlantillasUsuario == 0){ //Si el número de plantillas es 0 mostramos error//
      printf("\nNo existen plantillas.\n");
   }
   else{
      for(i=0;i<(*nPlantillas);i++){ //Comparamos el id de la plantilla introducido por
el usuario con los distintos id de plantilla del fichero plantillas//
         if(strcmp((*vPlantillas)[i].idplant,idplant)==0){
            p=i;
         }
      }

      while(resp!='s' && resp!='n'){ //Validamos que el usuario quiere eliminar la
plantilla//
         printf("\nSe eliminara la plantilla %s-%s-%s-%d-
%d\n",(*vPlantillas)[p].idprop,(*vPlantillas)[p].idplant,(*vPlantillas)[p].nombre,(*v
Plantillas)[p].presup,(*vPlantillas)[p].punt);
         printf("¿Esta seguro de eliminar la plantilla? Responda con (s/n)\n");
         fflush(stdin);
         scanf("%c",&resp);
      }
```

```
        if(resp=='s'){ //Si valida se copia un espacio vacío sobre los datos de la plantilla a eliminar//
            for(i=p;i+1<(*nPlantillas);i++){
                strcpy((*vPlantillas)[i].idprop,(*vPlantillas)[i+1].idprop);
                strcpy((*vPlantillas)[i].idplant,(*vPlantillas)[i+1].idplant);
                strcpy((*vPlantillas)[i].nombre,(*vPlantillas)[i+1].nombre);
                (*vPlantillas)[i].presup=(*vPlantillas)[i+1].presup;
                (*vPlantillas)[i].punt=(*vPlantillas)[i+1].punt;
            }

            (*nPlantillas)--;
            (*nPlantillasUsuario)--;
//Se resta en uno el número de plantillas y el número de plantillas del usuario//
            *vPlantillas=realloc((plantilla *)(*vPlantillas),(*nPlantillas)*sizeof(plantilla));
        } //Devolvemos el espacio reservado para la plantilla//
        else{
            printf("\nLa eliminacion de la plantilla ha sido cancelada\n");
        }
    }
}

void ranking(plantilla *vPlantillas, int nPlantillas){
    int top[3]={0,0,0}, ind[3]={0,0,0};
//Creamos dos vectores que almacenarán el top 3, y otro ayudará como auxiliar//
    for(int i=0;i<nPlantillas;i++){
        if(vPlantillas[i].punt>=top[0]){ //Si la puntuación de la plantilla de la vuelta es mayor al de la plantilla más alta por el momento almacenada en la primera posición, se asigna la nueva plantilla el top 1, y a la segunda la antigua top 1, y a la tercera la antigua top 2, ….//
            top[2]=top[1];
            top[1]=top[0];
            top[0]=vPlantillas[i].punt;
```

```
            ind[2]=ind[1];
            ind[1]=ind[0];
            ind[0]=i;
        }
        else{ //Si no es mayor al top 1, comprobamos si lo es al top 2//
            if(vPlantillas[i].punt>=top[1]){
                top[2]=top[1];
                top[1]=vPlantillas[i].punt;

                ind[2]=ind[1];
                ind[1]=i;
            }
            else{ //Y si no es mayor al top 2, comprobamos si lo es al top 3//
                if(vPlantillas[i].punt>=top[2]){
                    top[2]=vPlantillas[i].punt;

                    ind[2]=i;
                }
            }
        }
    }


    printf("1-> %s-%s-%s-%d-
%d\n",vPlantillas[ind[0]].idprop,vPlantillas[ind[0]].idplant,vPlantillas[ind[0]].nom
bre,vPlantillas[ind[0]].presup,vPlantillas[ind[0]].punt);
    printf("2-> %s-%s-%s-%d-
%d\n",vPlantillas[ind[1]].idprop,vPlantillas[ind[1]].idplant,vPlantillas[ind[1]].nom
bre,vPlantillas[ind[1]].presup,vPlantillas[ind[1]].punt);
    printf("3-> %s-%s-%s-%d-
%d\n",vPlantillas[ind[2]].idprop,vPlantillas[ind[2]].idplant,vPlantillas[ind[2]].nom
bre,vPlantillas[ind[2]].presup,vPlantillas[ind[2]].punt);
//Finalmente, imprimimos los resultados del top//
}
```

Nombre	printCargar
Precondición	Se llame a la función imprimir en alguna función, o se llame a la función printCargar de los datos como usuarios, equipos, …
Descripción	Realiza la función de imprimir los ficheros de datos necesarios, y cargar los datos por pantalla con un formato específico para cada uno de las clases de datos.
Postcondición	Finalice la tarea especificada de impresión.

Comentarios del Código

```
#include "tipos.h"

#include "cargar.h"


void imprimir(){ //Declaramos las funciones que se pueden realizar dentro de esta opción//

    printCargarUsuarios(vUsuarios,nUsuarios);

    printCargarEquipos(vEquipos,nEquipos);

    printCargarFutbolistas(vFutbolistas,nFutbolistas);

    printCargarPlantillas(vPlantillas,nPlantillas);

    printCargarJugplants(vJugplants,nJugplants);

    printCargarConfigs(vConfigs,nConfigs);

}


void printCargarUsuarios(usuario *vUsuarios,int n){

    int i;


    printf("\n#_____USUARIOS_____#\n");


    for(i=0;i<n;i++){ //Imprime los datos específicos de los usuarios según el formato//

        printf("\nID USUARIO:%s\n",vUsuarios[i].id);

        printf("\nNOMBRE:%s\n",vUsuarios[i].nombre);

        printf("\nPERFIL:%s\n",vUsuarios[i].perfil);

        printf("\nLOGIN:%s\n",vUsuarios[i].login);

        printf("\nPASS:%s\n",vUsuarios[i].pass);
```

```
    }
}

void printCargarEquipos(equipo *vEquipos,int n){
    printf("\n#_____EQUIPOS_____#\n");

    for(int i=0;i<n;i++){ //Se imprimen los datos de los equipos según el formato//
        printf("\nID EQUIPO: %s\n",vEquipos[i].id);
        printf("\nNOMBRE: %s\n",vEquipos[i].nombre);
    }
}

void printCargarFutbolistas(futbolista *vFutbolistas,int n){
    printf("\n#_____FUTBOLISTAS_____#\n");

    for(int i=0;i<n;i++){ //Se imprimen los datos de los futbolistas según el formato//
        printf("\nID FUTBOLISTA: %s\n",vFutbolistas[i].id);
        printf("\nID EQUIPO: %s\n",vFutbolistas[i].ideq);
        printf("\nNOMBRE: %s\n",vFutbolistas[i].nombre);
        printf("\nPRECIO: %d\n",vFutbolistas[i].precio);
        printf("\nVALORACION: %d\n",vFutbolistas[i].valor);
    }
}

void printCargarPlantillas(plantilla *vPlantillas,int n){
    printf("\n#_____PLANTILLAS_____#\n");

    for(int i=0;i<n;i++){ //Se imprimen los datos de las plantillas según el formato//
        printf("\nID PROPIETARIO: %s\n",vPlantillas[i].idprop);
        printf("\nID PLANTILLA: %s\n",vPlantillas[i].idplant);
        printf("\nNOMBRE: %s\n",vPlantillas[i].nombre);
        printf("\nPRESUPUESTO: %d\n",vPlantillas[i].presup);
        printf("\nPUNTUACION: %d\n",vPlantillas[i].punt);
```

```
    }
}

void printCargarJugplants(jugplant *vJugplants,int n){
    printf("\n#_____JUGADORES DE LA PLANTILLA_____#\n");

    for(int i=0;i<n;i++){ //Se imprimen los datos de jugadores de plantilla según el formato//
        printf("\nID USUARIO: %s\n",vJugplants[i].idjug);
        printf("\nID PLANTILLA: %s\n",vJugplants[i].idplant);
    }
}

void printCargarConfigs(config *vConfigs,int n){
    printf("\n#_____CONFIGURACION_____#\n");

    for(int i=0;i<n;i++){ //Se imprimen los datos de configuración según el formato//
        printf("\nCONFIGURACION: %s\n",vConfigs[i].config);
        printf("VALOR: %d\n",vConfigs[i].valor);
    }
}
```

Nombre	registro
Precondición	El usuario selecciona la función registrarse en el menú de inicio.
Descripción	Realiza la función de registrar a un nuevo usuario como participante, ya que los administradores y cronistas son introducidos directamente por el administrador.
Postcondición	El usuario se registre correctamente.

Comentarios del Código

```
#include <stdio.h>
#include <string.h>
#include "tipos.h"
#include "cargar.h"
#include "registro.h"

void registro(usuario **vUsuarios,int *n){ //Recogemos los datos necesarios para registrar a un nuevo usuario//
    printf("<----------REGISTRO---------->");
//El ID//
    printf("\nIntroduzca el ID de usuario que desee (2 caracteres como maximo): ");
    fflush(stdin);
    scanf("%2s",(*vUsuarios)[*n].id);
//Nombre completo del usuario//
    printf("\nAhora, introduzca su nombre completo (20 caracteres como maximo): ");
    fflush(stdin);
    scanf("%20s",(*vUsuarios)[*n].nombre);
//Login o nombre de usuario para iniciar sesión//
    printf("\nA continuacion, introduzca el login del usuario (5 caracteres como maximo): ");
    fflush(stdin);
    scanf("%5s",(*vUsuarios)[*n].login);
//Contraseña del usuario//
    printf("\nFinalmente, introduzca una contrasena de 8 caracteres como maximo: ");
    fflush(stdin);
    scanf("%8s",(*vUsuarios)[*n].pass);

    strcpy((*vUsuarios)[*n].perfil,"participante"); //Recordemos que sólo se pueden registrar participantes, y por lo tanto copiamos “participante” en el dato perfil del nuevo usuario//

    (*n)++; //Sumamos uno al número de usuarios para el siguiente registro//
}
```