DEPARTAMENTO DE
INGENIERÍA
INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

CONTROL 1 ESTRUCTURAS DE DATOS 2021-1

ALTERNATIVA DE SOLUCIÓN CONTROL 1 – FORMA B

A continuación, se presenta una de las muchas alternativas de solución que existen para resolver la segunda forma del control 1.

El archivo datosB.txt nos permite definir el struct siguiente para almacenar los datos de los partidos:

```c
//Estructura para almacenar la informacion de un partido
struct partido{
    char local[4];  //Uno de los equipos que juega el partido
    char visita[4]; //El otro equipo que juega el partido
    char dia[3];    //Dia en que se juega el partido
    char mes[3];    //Mes en que se juega el partido
    int hora;       //Hora a la que se juega el partido
    char lugar[3];  //Lugar en el que se juega el partido
};
typedef struct partido Partido;
```

Las funciones que se crean para resolver el problema planteado son:

```c
/*Encabezados (prototipos) de las funciones*/
Partido *leerEntrada(char const *nombreArchivo, int *n);
void partidosPorPais(Partido *p, int n);
void partidosPorFecha(Partido *p, int n);
void partidosPorLugar(Partido *p, int n);
```

Función principal:

```c
/*FUNCIÓN PRINCIPAL*/
int main(int argc, char const *argv[]){
    int n; //cantidad de partidos del archivo
    //Lee archivo de entrada y se actualiza la cantidad de partidos
    Partido *listaPartidos = leerEntrada(argv[1], &n);
    if(n == -1){ //Se comprueba si hubo un error al leer el archivo
        printf("No se pudo encontrar el archivo de entrada indicado.");
        return 0;
    }
    else if(n <= 0){
        printf("El archivo de entrada no posee partidos o hay un error en el formato.");
        return 0;
    }
    int opcion=0;  //Variable para almacenar la opcion ingresada por el usuario
    while(opcion != 4){
        printf("Menu:\n 1) Partidos por fecha\n 2) Partidos por pais\n 3) Partidos por lugar\n 4) Salir\n \nIngrese opcion: ");
        if(scanf("%d", &opcion) == 0 || opcion > 4 || opcion < 1){
            printf("\nOpcion invalida\n");
        }else{
            if(opcion == 1) //Partidos por fecha
                partidosPorFecha(listaPartidos, n);
            if(opcion == 2) //Partidos por pais
                partidosPorPais(listaPartidos, n);
            if(opcion == 3) //Partidos por lugar
                partidosPorLugar(listaPartidos, n);
            if(opcion == 4) //Terminar el programa
                printf("Terminando la ejecucion del programa\n");
        }
    }
    return 0;
}
```

Función que lee los datos del archivo:

```
//Entrada: Nombre archivo de entrada y direccion de memoria de n donde se almacena la cantidad de elementos
//Salida: Lista de los partidos presentes en el archivo de entrada
//Objetivo: Leer el archivo de entrada y almacenar la informacion contenida en el donde corresponda
Partido *leerEntrada(char const *nombreArchivo, int *n){
    FILE *pf;
    pf = fopen(nombreArchivo, "r");
    if (pf == NULL) { //En caso de que el archivo de entrada no exista
        *n = -1;
        Partido *listaPartidos = (Partido*)malloc(sizeof(Partido)*(0));
        return listaPartidos;
    }
    //Se lee el primer elemento que corresponde a la cantidad de partidos y se actualiza su valor
    fscanf(pf, "%d", n);
    //Se asigna memoria para la cantida de partidos
    Partido *listaPartidos = (Partido*)malloc(sizeof(Partido)*(*n));

    for(int i=0; i < *n; i++){//Ciclo para leer el archivo
        //Se crea un nuevo partido
        Partido nuevoPartido;

        //Se leen y almacenan los datos del partido
        fscanf(pf, "%3s %3s", nuevoPartido.local, nuevoPartido.visita);
        fscanf(pf, "%s", nuevoPartido.dia);
        fscanf(pf, "%s", nuevoPartido.mes);
        fscanf(pf, "%d", &nuevoPartido.hora);
        fscanf(pf, "%s", nuevoPartido.lugar);
        //Se agrega el partido a la lista de partidos
        listaPartidos[i] = nuevoPartido;
    }
    fclose(pf);//Se cierra el archivo de entrada
    return listaPartidos;
}
```

Función que permite resolver el primer requerimiento: Dada una fecha (día, mes), indicar cuáles y cuántos partidos se realizarán.

```
//Entrada: Lista de Partido que contiene todos los partidos y la cantidad de partidos en la lista
//Salida: Nada
//Objetivo: Buscar la cantidad de partidos que se jugaran en cierta fecha y mostrar la informacion de estos partidos
void partidosPorFecha(Partido *p, int n){
    char dia[3]; //Variables para almacenar los datos ingresados por el usuario
    char mes[3];
    int diaInt, mesInt; //Variables para almacenar una version int de estos datos
    printf("Ingrese fecha partido (dd mm): "); //Se pide la informacion al usuario
    scanf("%s %s", &dia, &mes);
    diaInt = atoi(dia); //Se transforma esta informacion a entero
    mesInt = atoi(mes);
    //Se comprueba que la informacion entregada cumpla el formato
    if(diaInt > 0 && diaInt <= 31 && mesInt > 0 && mesInt <= 12 && strlen(dia) == 2 && strlen(mes) == 2){
        //Se muestra la cantidad de partidos encontrados
        printf("Hay %d partidos para el %s de %s: \n", cuenta_partidos(p,n,dia,mes,"",1), dia, transformarMes(mes));
        for(int i=0; i < n; i++){//Se recorre la lista mostrando la informacion de dichos partidos
            if(strcmp(p[i].dia, dia) == 0 && strcmp(p[i].mes, mes) == 0){
                printf("  %s vs %s a las %d en %s\n", p[i].local, p[i].visita , p[i].hora, p[i].lugar);
            }
        }
        printf("\n");
    }else{
        printf("\nFormato ingresado es invalido\n\n");
    }
}
```

Función que permite resolver el segundo requerimiento: Dado un país (tres primeras letras versus tres últimas letras del partido), indicar cuáles y cuántos partidos se realizarán.

```
//Entrada: Lista de Partido que contiene todos los partidos y la cantidad de partidos en la lista
//Salida: Nada
//Objetivo: Buscar la cantidad de partidos que jugara cierto pais y mostrar la informacion de estos partidos
void partidosPorPais(Partido *p, int n){
    char paisBuscar[3];//Variable para almacenar el nombre del pais
    printf("\nIngrese codigo pais: "); //Se pide el pais al usuario
    scanf("%3s", paisBuscar);
    if(strlen(paisBuscar) != 3){//Se comprueba que el dato ingresado cumpla el formato
        printf("Formato de pais invalido.\n\n");
    }else{
        //Se muestra la cantidad de partidos encontrados
        printf("Hay %d partidos para %s:\n", cuenta_partidos(p,n,"","",paisBuscar,2), paisBuscar);
        //Se recorre la lista mostrando la informacion de dichos partidos
        for(int i=0; i < n; i++){
            if(strncmp(p[i].local, paisBuscar,3) == 0 || strncmp(p[i].visita, paisBuscar,3) == 0)
                printf("  %s vs %s: %s de %s a las %d en %s\n", p[i].local,p[i].visita,p[i].dia,transformarMes(p[i].mes),p[i].hora,p[i].lugar);
        }
    }
    printf("\n");
}
```

Función que permite resolver el tercer requerimiento: Para un lugar dado, que entregue la información de los partidos que allí se realizarán.

```
//Entrada: Lista de Partido que contiene todos los partidos y la cantidad de partidos en la lista
//Salida: Nada
//Objetivo: Buscar la cantidad de partidos que se realizaran en el lugar indicado y mostrar la informacion de estos
void partidosPorLugar(Partido *p, int n){
    char lugarBuscar[3]; //Variable para almacenar el lugar del partido
    printf("\nIngrese lugar de encuentro: "); //Se pide el lugar
    scanf("%3s", lugarBuscar);
    if(strlen(lugarBuscar) != 3){//Se comprueba que tenga el formado requerido
        printf("Formato de lugar invalido.\n\n");
    }else{
        //Se muestra la cantidad de partidos a realizarce en el lugar indicado
        printf("Hay %d partidos programados en %s:\n", cuenta_partidos(p,n,"","",lugarBuscar,3), lugarBuscar);
        //Se recorre la lista mostrando la informacion de los partidos que se realizaran en el lugar indicado
        for(int i=0; i < n; i++){
            if(strncmp(p[i].lugar, lugarBuscar,3) == 0 )
                printf("  %s vs %s: %s de %s a las %d en %s\n",p[i].local,p[i].visita,p[i].dia,transformarMes(p[i].mes),p[i].hora,p[i].lugar);
        }
    }
    printf("\n");
}
```

Función que retorna el nombre del mes dado un número:

```c
//Entrada: String con el mes a transformar
//Salida: String con el nombre del mes
//Objetivo: Obtener el nombre de un mes a partir de su numero
char const *transformarMes(char *mesString){
    if (strcmp(mesString, "01") == 0)
        return "Enero";
    else if (strcmp(mesString, "02") == 0)
        return "Febrero";
    else if (strcmp(mesString, "03") == 0)
        return "Marzo";
    else if (strcmp(mesString, "04") == 0)
        return "Abril";
    else if (strcmp(mesString, "05") == 0)
        return "Mayo";
    else if (strcmp(mesString, "06") == 0)
        return "Junio";
    else if (strcmp(mesString, "07") == 0)
        return "Julio";
    else if (strcmp(mesString, "08") == 0)
        return "Agosto";
    else if (strcmp(mesString, "09") == 0)
        return "Septiembre";
    else if (strcmp(mesString, "10") == 0)
        return "Octubre";
    else if (strcmp(mesString, "11") == 0)
        return "Noviembre";
    else if (strcmp(mesString, "12") == 0)
        return "Diciembre";
}
```

Función que obtiene el total de partidos para los filtros de búsqueda de los requerimientos 1, 2 y 3.

```c
/*
Entrada: lista con partidos, tamaño de ésta, dia, mes, pais (o lugar) partido según corresponda y opción de búsqueda
Salida: número de partidos que cumple el filtro según la opción de búsqueda
Descripcion: SEgún la opción de búsqueda, determina el total de partidos que cumple con el filtro de búsqueda:
             por fecha (op==1), por país (op==2) o por lugar (op==3)
*/
int cuenta_partidos(Partido *p, int n, char dia[], char mes[], char aBuscar[], int op){
    int numPartidos=0;
    int i=0;
    for(i=0;i<n;i++){
        if (op==1){
            if(strcmp(p[i].dia, dia) == 0 && strcmp(p[i].mes, mes) == 0)
                numPartidos++;
        }
        if (op==2){
            if(strcmp(p[i].local, aBuscar) == 0 || strcmp(p[i].visita, aBuscar) == 0)
                numPartidos++;
        }
        if (op==3){
            if(strcmp(p[i].lugar, aBuscar) == 0 )
                numPartidos++;
        }
    }
    return numPartidos;
}
```