**U.B.A. FACULTAD DE INGENIERÍA**

Algoritmos y Programación II (75.41)

Curso 4 - Wachenchauzer

Trabajo Práctico Nº 2

Curso 2013 - 2do Cuatrimestre

Integrantes:

* Amura Federico, 95202
* Rupcic Florencia, 94525

Fecha de entrega: 11 de noviembre de 2013

Corrector:

Luciano

Análisis y diseño.

Para la realización del presente trabajo práctico fue necesaria la implementación de los siguientes TDA: hash, heap, lista y cola.

En primera instancia se decidio por separar el programa en dos partes, una que seria la entrada ya sea de los archivos y los comandos, el parser, implementado en tp2.c, y otra que seria la encargada de ejecutar los comandos sobre una base de datos con cierta estructura, implementado en mundial.c.

El parser es muy parecido a lo que se realizo en el anterior tp respecto a los comandos, separando en lineas y dentro de cada linea siendo la primer palabra el comando y luego los parametros que se separaron siguiendo cierta estructura para luego usarlos adecuadamente dentro de los comandos.

La base incluia dentro de esta varias de las estructuras que se debian usar, como el hash o el heap según sea la utilidad que fuera necesaria:

Se utilizó un hash para contener a todos los equipos inscriptos al mundial. Se utilizó otro hash para contener a los jugadores de todos los equipos. De esta manera se garantiza un acceso inmediato a los jugadores o a los equipos cuando haga falta.

Los jugadores, además, se guardaron en un heap de goleadores. Esto es simplemente para poder acceder al goleador inmediatamente.

El fixture se realizó en un vector donde se almacenaron los partidos (con su propia estructura) haciendo coincidir la posicion en el vector con el partido en cuestion, es decir, la final seria la posicion 0, las semifinales las 2 siguientes, etc.

De esta manera se puede aprovechar todas las ventajas que nos daba cada estructura de datos, ademas cabe aclarar que para acelerar el acceso, las estructuras de partido tenian punteros a los equipos que lo juegan, y estos equipos tambien tenian accesos a los jugadores del mismo. Todo esto termina resultando en muchos punteros repetidos hacia las mismas estructuras, lo bueno que se consigue asi es agilizar y facilitar el acceso, lo malo, es que hay que trabajarlos de manera acorde, considerando los cambios que puede llegar a hacer cada uno y replicar en las distintas estructuras, ademas de que la destruccion de los datos podria intentar hacerse varias veces y la redimension de cualquier estructura repetida queda terminantemente prohibida ya que si no, habria que ajustar todos los punteros que apuntan a ella.

Comandos de administración.

• agregar\_resultado

Luego de verificar que el partido no este ya cargado o que sea un codigo de partido valido, el comando consigue un puntero hacia la estructura del partido en cuestion, luego carga los goles metidos por ambos equipos (se supone que los equipos ya estan cargados ya sea desde la carga de los equipos al principio o del resultado de un partido anterior). Luego, toma la lista que recibe por parametro, esta incluye los dorsales de los jugadores que metieron los goles, por lo tanto, mediante dos ciclos, uno para el local y otro para el visitante va aumentando los goles que metio en la estructura de cada jugador mediante la funcion jugador\_metio\_gol. Una vez terminado ajusta el heap para que refleje los cambios producidos con los goles. Finalmente debe cargar el equipo ganador en el siguiente partido (a menos que sea la final) por lo que calcula si debe ir como local o visitante y en que posicion esta el siguiente partido para poder hacerlo. Dado que todos los accesos se realizan mediante posiciones o claves, el orden seria constante, sin embargo, dentro de este comando se encuentra el ajuste en el heap, que es de orden O(k), siendo k la cantidad total de jugadores, y por lo tanto, ese tambien es el orden de este comando.

• listar\_jugadores

Al recibir el nombre del país junto a este comando, se buscará en el hash de equipos si el país pertenece a él. En caso afirmativo, se utilizará el filtro deseado para listar a los jugadores, ya sea por dorsal o por nombre. Una función auxiliar se encargará de imprimir los datos contenidos en el vector de jugadoresdorsal o jugadoresnombre. Se almacenaron los jugadores según estos filtros en vectores distintos para hacer de la función listar\_jugadores una cuyo orden fuera de O(k), siendo k la cantidad total de jugadores de un equipo.

• listar\_goleador

Se necesita que la función encargada de imprimir los jugadores de un determinado equipo tenga un tiempo de O(1). Es por eso que se utilizó un heap de máximos para almacenar a todos los jugadores del mundial según la cantidad de goles realizados. Ver el máximo en un heap de máximos tiene un orden de O(1), cumpliendo con el requisito del enunciado.

• goles\_jugador

Se buscará si el jugador pertenece al hash que contiene a los jugadores del mundial. En caso afirmativo, se imprimirán el nombre del jugador, número de goles, número dorsal y equipo al cual pertenece.

Se necesita que la función encargada de imprimir los goles de un determinado jugador tenga un tiempo de O(1).

• mostrar\_resultado

Si se desea mostrar el resultado de un partido dado, se utilizará el idr para calcular la posición del partido y luego se buscará en el vector que contiene los partidos inscriptos al mundial.

El orden de este comando es O(1), pues accede directamente a la posición del vector gracias a la función posición\_partido, la cual se encarga de calcular la posición exacta en la cual se almacenan los datos del idr.

Mejoras a realizar.

-Podria mejorarse el comando de agregar\_resultado quitando el heapify y realizando un heap\_upheap para cada jugador que haya metido un gol, en escencia seria hacer lo mismo pero solo para los valores que se modificaron, la mejora seria que los que no fueron modificados no es los trata de mover, pues no es necesario.

-Se podrian realizar mejoras en la base de datos, pues se generan muchos punteros a una misma estructura, es decir, se tiene mucha informacion redundante solamente por el hecho de tenerla organizada de alguna manera.

Implementación.

Nota: Para la realización del TP se utilizaron las siguientes estructuras: hash, heap, lista y cola. Esos archivos no se adjuntan en este informe.

**tp2.c**

1 #define \_GNU\_SOURCE

2 #include <stdio.h>

3 #include <stdlib.h>

4 #include <string.h>

5 #include <stdbool.h>

6 #include <stddef.h>

7 #include "lista.h"

8 #include "hash.h"

9 #include "heap.h"

10 #include "mundial.h"

11 #include "cola.h"

12

13

14 **void** procesar\_comando(base\_t\* base, **char**\* comando, **char**\* parametro1, **char**\* parametro2, **char**\* parametro3, cola\_t\* cola) {

15

16 **if** (strcmp(comando,"agregar\_resultado") == 0) {

17 **int** i = agregar\_resultado(base,parametro1,parametro2,parametro3,cola);

18 **if** (i == 0) printf("OK\n");

19 **if** (i == -1) printf("Error: el resultado con id %s ya existe\n",parametro1);

20 **if** ((i == -2) || (i == -3)) printf("Error: el resultado con id %s no existe\n",parametro1);

21 **if** ((i>0) || (i<-3)) printf("Algo pasó%d\n",i);

22 }

23

24 **if** (strcmp(comando,"listar\_jugadores") == 0) {

25 **int** i = listar\_jugadores(base,parametro1,parametro2);

26 **if** (i == -1) printf("Error: el equipo %s no esta inscripto en el fixture\n",parametro2);

27 }

28

29 **if** (strcmp(comando,"listar\_goleador") == 0) {

30 listar\_goleador(base);

31 }

32

33 **if** (strcmp(comando,"goles\_jugador") == 0) {

34 **int** i = goles\_jugador(base,parametro1);

35 **if** (i == -1) printf("Error: el jugador %s no esta inscripto en el fixture\n",parametro1);

36 }

37

38 **if** (strcmp(comando,"mostrar\_resultado") == 0) {

39 **int** i = mostrar\_resultado(base,parametro1);

40 **if** ((i == -1) || (i == -2) || (i == -3)) printf("Error: el resultado con id %s no existe\n",parametro1);

41 }

42 }

43

44

45 **void** destruir\_dato(**void**\* dato) {

46 **if** (dato != NULL)

47 free(dato);

48 }

49

50 **bool** leer\_consola(cola\_t \*cola) {

51 **char** c;

52 **char** lectura[100];

53 **char** limitador = ' ';

54 **bool** condLimitador = **true**;

55 **char** limitador2 = ',';

56 **int** i;

57 i = 0;

58

59 **while** ((c = getchar()) != '\n' && c != EOF) {

60 **if** ((c != limitador || !condLimitador) && c != limitador2) {

61 lectura[i]=c;

62 i++;

63 }

64 **if** ((c == limitador && condLimitador) || c == limitador2) {

65 lectura[i] = '\0';//LE AGREGO EL CARACTER DE FIN DE STRING PARA ASEGURARME QUE NO META BASURA

66 i = 0;

67 **if** (strcmp("listar\_jugadores", lectura) != 0)

68 condLimitador = **false**;

69 **char**\* param = malloc(strlen(lectura)+1);

70 strcpy(param,lectura);

71 cola\_encolar(cola,param);

72 }

73 }

74 lectura[i] = '\0';

75 **char**\* param = malloc(strlen(lectura)+1);

76 strcpy(param,lectura);

77 cola\_encolar(cola,param);

78 **if** (c == EOF) {

79 **return false**;

80 }

81 **return true**;

82 }

83

84 **int** main(**int** argc, **char** \*argv[]) {

85

86 base\_t\* base\_de\_datos = base\_crear();

87

88 cola\_t\* cola = cola\_crear();

89 **bool** seguir;

90

91 **char**\* param1 = NULL;

92 **char**\* param2 = NULL;

93 **char**\* param3 = NULL;

94 **char**\* param4 = NULL;

95

96 **if** (argc!=2) **return** 0; //no se indico el archivo a abrir o estan mal pasados los argumentos

97 FILE\* file;

98 **char** str[60];

99 **char** str\_jugadores[60];

100 file = fopen(argv[1],"r");

101 **while** ( fgets(str,60,file) != NULL ) {

102 **char**\* equipo = strtok(str,"\r\n");

103 agregar\_equipo(base\_de\_datos,equipo);

104 **for** (**int** i = 1; i < 24; i ++) {

105 **if** ( fgets(str\_jugadores,60,file) != NULL ) {

106 **char**\* jugad = strtok(str\_jugadores,"\r\n");

107 agregar\_jugador(base\_de\_datos,equipo,jugad,i);

108 }

109 }

110 }

111 fclose(file);

112

113 **do** {

114 seguir = leer\_consola(cola);

115

116 param1 = (**char**\*)cola\_desencolar(cola);

117 param2 = (**char**\*)cola\_desencolar(cola);

118 param3 = (**char**\*)cola\_desencolar(cola);

119 param4 = (**char**\*)cola\_desencolar(cola);

120

121 procesar\_comando(base\_de\_datos,param1,param2,param3,param4,cola);

122

123 destruir\_dato(param1);

124 destruir\_dato(param2);

125 destruir\_dato(param3);

126 destruir\_dato(param4);

127 **while** (!cola\_esta\_vacia(cola)) free(cola\_desencolar(cola));

128 fflush(stdin);

129

130 } **while** (seguir);

131

132 cola\_destruir(cola,(**void**(\*)(**void**\*))destruir\_dato);

133 base\_destruir(base\_de\_datos);

134 **return** 0;

135 }

**mundial.h**

1 #ifndef MUNDIAL\_H

2 #define MUNDIAL\_H

3

4 #include <stdio.h>

5 #include <stdlib.h>

6 #include <string.h>

7 #include <stdbool.h>

8 #include <stddef.h>

9

10 #include "cola.h"

11

12 **typedef struct** base base\_t;

13

14 /\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

15 \* PRIMITIVAS DE LA BASE

16 \* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

17

18 /\* Primitivas basicas \*/

19

20 // Crea una base de datos.

21 // Post: devuelve una nueva base vacÃ­a.

22 base\_t \*base\_crear();

23

24 // Destruye una base de datos

25 // Pre: la base fue creada

26 // Post: se destruyo la base

27 **void** base\_destruir(base\_t\* base);

28

29 // Agrega un equipo a la base

30 // Pre: la base fue creada

31 // Post: se agrego el equipo a la base

32 **int** agregar\_equipo(base\_t\* base,**char**\* equipo);

33

34 // Agrega un jugador al equipo

35 // Pre: la base fue creada

36 // Post: se agrego el jugador al equipo

37 **int** agregar\_jugador(base\_t\* base,**char**\* equipo,**char**\* jugador,**int** dorsal);

38

39 // Le agrega un gol al contador de goles del jugador

40 // Pre: la base fue creada

41 // Post: se incremento el contador de goles

42 **int** jugador\_metio\_gol(base\_t\* base,**char**\* equipo,**int** dorsal);

43

44 // Imprime una lista de los jugadores de un equipo de la base

45 // Pre: la base fue creada

46 // Post: se imprime la lista

47

48 **int** listar\_jugadores(base\_t\* base,**char**\* filtro,**char**\* equipo);

49

50 // Imprime la informacion de un jugador

51 // Pre: la base fue creada

52 // Post: se imprime la informacion

53 **int** goles\_jugador(base\_t\* base,**char**\* jugador);

54

55 /\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

56 \* PRIMITIVAS DEL FIXTURE

57 \* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

58

59 // Agrega un resultado a la base de datos de los partidos

60 // Pre: la informacion recibida es valida y con el formato correspondiente

61 // Post: se agrego el resultado

62

63 **int** agregar\_resultado(base\_t\* base, **char**\* idr, **char**\* gloc, **char**\* gvis, cola\_t\* dorsales);

64

65 // Agrega un resultado a la base de datos de los partidos

66 // Pre: la informacion recibida es valida y con el formato correspondiente

67 // Post: se agrego el resultado

68 **int** listar\_goleador(base\_t\* base);

69

70 // Agrega un resultado a la base de datos de los partidos

71 // Pre: la informacion recibida es valida y con el formato correspondiente

72 // Post: se agrego el resultado

73 **int** mostrar\_resultado(base\_t\* base,**char**\* informacion);

74

75 #endif // MUNDIAL\_H

**mundial.c**

1 #ifndef MUNDIAL\_C

2 #define MUNDIAL\_C

3

4 #include <stdio.h>

5 #include <stdlib.h>

6 #include <string.h>

7 #include <stdbool.h>

8 #include <stddef.h>

9 #include "mundial.h"

10 #include "lista.h"

11 #include "hash.h"

12 #include "heap.h"

13 #include "cola.h"

14 #define LNOMBRE 150

15 #define LEQUIPO 100

16 #define LFILTRO 10

17

18 **typedef struct** jugador{

19 **char**\* nombre;

20 **int** dorsal;

21 **int** goles;

22 **char**\* equipo;

23 }jugador\_t;

24

25 **typedef struct** equipo{

26 **char**\* nombre;

27 **size\_t** jugadoresanotados;

28 jugador\_t\*\* jugadoresdorsal;

29 jugador\_t\*\* jugadoresnombre;

30 }equipo\_t;

31

32 **typedef struct** partido{

33 equipo\_t\* local;

34 equipo\_t\* visitante;

35 **int** goleslocal;

36 **int** golesvisitante;

37 }partido\_t;

38

39 **struct** base{

40 hash\_t\* equipos;

41 hash\_t\* jugadores;

42 heap\_t\* goleadores;

43 partido\_t\*\* partidos;

44 **int** equiposanotados;

45 };

46

47 /\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

48 \* PRIMITIVAS DE LA BASE

49 \* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

50

51 // Funcion auxiliar para ordenar los jugadores por dorsal

52 **int** comparar\_dorsales(**const** jugador\_t\* jugador1, **const** jugador\_t\* jugador2){

53 **return** (jugador1->dorsal-jugador2->dorsal);

54 }

55

56 // Funcion auxiliar para ordenar los jugadores por nombre

57 **int** comparar\_nombres(**const** jugador\_t\* jugador1, **const** jugador\_t\* jugador2){

58 **return** strcmp(jugador1->nombre,jugador2->nombre);

59 }

60

61 // Funcion auxiliar que imprime el vector de jugadores

62 **void** listar\_vector\_jugadores(jugador\_t\*\* vectorjugadores){

63 **for**(**int** i = 0;i<23;i++) printf("%s,%d: Goles: %d\n",vectorjugadores[i]->nombre,vectorjugadores[i]->dorsal,vectorjugadores[i]->goles);

64 }

65

66 // Funcion auxiliar que crea un nuevo jugador

67 jugador\_t\* crear\_jugador(**char**\* nombre, **int** dorsal, **char**\* equipo){

68 jugador\_t\* jugador = malloc(**sizeof**(jugador\_t));

69 **if** (jugador == NULL) **return** NULL;

70 jugador->nombre = malloc(LNOMBRE\***sizeof**(**char**));

71 strcpy(jugador->nombre,nombre);

72 jugador->dorsal = dorsal;

73 jugador->goles = 0;

74 jugador->equipo = malloc(LEQUIPO\***sizeof**(**char**));

75 strcpy(jugador->equipo,equipo);

76 **return** jugador;

77 }

78

79 // Funcion auxiliar que crea un nuevo equipo

80 equipo\_t\* crear\_equipo(**char**\* nombre){

81 equipo\_t\* equipo = malloc(**sizeof**(equipo\_t));

82 **if** (equipo == NULL) **return** NULL;

83 equipo->nombre = malloc(LEQUIPO\***sizeof**(**char**));

84 strcpy(equipo->nombre,nombre);

85 equipo->jugadoresanotados = 0;

86 equipo->jugadoresdorsal = malloc(23\***sizeof**(jugador\_t\*));

87 **if** (equipo->jugadoresdorsal == NULL){

88 free(equipo->nombre);

89 free(equipo);

90 **return** NULL;

91 }

92 equipo->jugadoresnombre = malloc(23\***sizeof**(jugador\_t\*));

93 **if** (equipo->jugadoresnombre == NULL){

94 free(equipo->nombre);

95 free(equipo->jugadoresdorsal);

96 free(equipo);

97 **return** NULL;

98 }

99 **return** equipo;

100 }

101

102 // Funcion auxiliar que destruye un jugador

103 **void** destruir\_jugador(jugador\_t\* jugador){

104 free(jugador->nombre);

105 free(jugador->equipo);

106 free(jugador);

107 }

108

109 // Funcion auxiliar que destruye un equipo

110 **void** destruir\_equipo(equipo\_t\* equipo){

111 free(equipo->nombre);

112 free(equipo->jugadoresdorsal);

113 free(equipo->jugadoresnombre);

114 free(equipo);

115 }

116

117 // Funcion auxiliar que comparar dos jugadores segun sus goles

118 **int** comparar\_goleador(**const** jugador\_t\* jugador1, **const** jugador\_t\* jugador2){

119 **return** ((jugador1->goles)-(jugador2->goles));

120 }

121

122 /\* Implementaciones de primitivas \*/

123

124 base\_t \*base\_crear(){

125 base\_t\* base = malloc(**sizeof**(base\_t));

126 **if** (base==NULL) **return** NULL;

127 base->equiposanotados=0;

128 base->jugadores = hash\_crear((hash\_destruir\_dato\_t)&destruir\_jugador);

129 base->equipos = hash\_crear((hash\_destruir\_dato\_t)&destruir\_equipo);

130 base->goleadores = heap\_crear((cmp\_func\_t)&comparar\_goleador);

131 base->partidos = malloc(15\***sizeof**(partido\_t\*));

132 //verifico que todo se haya creado correctamente

133 **if** ((base->jugadores==NULL) || (base->equipos==NULL) || (base->goleadores==NULL) || (base->partidos==NULL)){

134 //alguna cosa se creo mal

135 **if** (base->jugadores!=NULL) hash\_destruir(base->jugadores);

136 **if** (base->equipos!=NULL) hash\_destruir(base->equipos);

137 **if** (base->goleadores!=NULL) heap\_destruir(base->goleadores,NULL);

138 **if** (base->partidos!=NULL) free(base->partidos);

139 free(base);

140 **return** NULL;

141 }

142 **for**(**int** i=0;i<15;i++) base->partidos[i] = calloc(1,**sizeof**(partido\_t));

143 **return** base;

144 }

145

146 **void** base\_destruir(base\_t\* base){

147 heap\_destruir(base->goleadores,NULL);

148 hash\_destruir(base->jugadores);

149 hash\_destruir(base->equipos);

150 **for**(**int** i=0;i<15;i++) free(base->partidos[i]);

151 free(base->partidos);

152 free(base);

153 **return**;

154 }

155

156 **int** agregar\_equipo(base\_t\* base,**char**\* pais){

157 equipo\_t\* equipo = crear\_equipo(pais);

158 **if** (equipo==NULL) **return** -1; //no se pudo crear el equipo

159 **if** (!hash\_guardar(base->equipos,pais,equipo)){

160 destruir\_equipo(equipo);

161 **return** -2; //no se pudo guardar el equipo

162 }

163 **int** resto = (base->equiposanotados%2);

164 **int** posicion = 7 + (base->equiposanotados)/2;

165 **if** (resto==0){

166 base->partidos[posicion]->local = equipo;

167 }**else**{

168 base->partidos[posicion]->visitante = equipo;

169 }

170 base->equiposanotados++;

171 **return** 0; //todo bien

172 }

173

174 **int** agregar\_jugador(base\_t\* base,**char**\* equipo,**char**\* nombre,**int** dorsal){

175 jugador\_t\* jugador = crear\_jugador(nombre,dorsal,equipo);

176 **if** (jugador==NULL) **return** -1; //no se pudo crear el jugador

177 //veo si esta anotado el pais

178 **if** (!hash\_pertenece(base->equipos,equipo)){

179 destruir\_jugador(jugador);

180 **return** -2; //el pais no esta anotado

181 }

182 equipo\_t\* pais = hash\_obtener(base->equipos,equipo);

183 //agrego el jugador al hash de jugadores y verifico

184 **if** (!hash\_guardar(base->jugadores,nombre,jugador)){

185 destruir\_jugador(jugador);

186 **return** -3; //no se pudo guardar el jugador en el hash de jugadores

187 }

188 //agrego el jugador al pais en los vectores y si esta completo ordeno

189 pais->jugadoresdorsal[pais->jugadoresanotados] = jugador;

190 pais->jugadoresnombre[pais->jugadoresanotados] = jugador;

191 pais->jugadoresanotados++;

192 **if** (pais->jugadoresanotados==23){

193 heap\_sort((**void**\*)pais->jugadoresdorsal,23,(cmp\_func\_t)comparar\_dorsales);

194 heap\_sort((**void**\*)pais->jugadoresnombre,23,(cmp\_func\_t)comparar\_nombres);

195 }

196 //agrego el jugador al heap de goleadores

197 **if** (!heap\_encolar(base->goleadores,jugador)){

198 hash\_borrar(base->jugadores,nombre);

199 destruir\_jugador(jugador);

200 **return** -5; //no se pudo agregar el jugador al heap de goleadores

201 }

202 **return** 0;

203 }

204

205 **int** jugador\_metio\_gol(base\_t\* base,**char**\* pais,**int** dorsal){

206 **if** (dorsal<1 || dorsal>23) **return** -2; //el jugador tiene un dorsal invalido

207 **if** (!hash\_pertenece(base->equipos,pais)) **return** -1; //el pais no esta en el hash

208 equipo\_t\* equipo = hash\_obtener(base->equipos,pais);

209 **if** (equipo->jugadoresanotados<dorsal) **return** -2; //el jugador tiene un dorsal invalido

210 equipo->jugadoresdorsal[dorsal-1]->goles++;

211 **return** 0;

212 }

213

214

215 **int** listar\_jugadores(base\_t\* base,**char**\* filtro,**char**\* pais){

216 **if** (!hash\_pertenece(base->equipos,pais)) **return** -1;

217 equipo\_t\* equipo = hash\_obtener(base->equipos,pais);

218 **if** (strcmp(filtro,"dorsal")==0) listar\_vector\_jugadores(equipo->jugadoresdorsal);

219 **if** (strcmp(filtro,"nombre")==0) listar\_vector\_jugadores(equipo->jugadoresnombre);

220

221

222 **return** 0;

223 }

224

225 **int** goles\_jugador(base\_t\* base,**char**\* nombre){

226 **if** (!hash\_pertenece(base->jugadores,nombre)) **return** -1; //el jugador no esta en el hash

227 jugador\_t\* jugador = hash\_obtener(base->jugadores,nombre);

228 fprintf(stdout,"%s,%d: %s Goles: %d\n",jugador->nombre,jugador->dorsal,jugador->equipo,jugador->goles);

229 **return** 0;

230 }

231

232 /\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

233 \* PRIMITIVAS DEL FIXTURE

234 \* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

235

236 // Funcion auxiliar que devuelve la posicion del partido segun su clave

237 **int** posicion\_partido(**char**\* idr){

238 **if** (strlen(idr)>2) **return** -1; // caso en el que idr tenga mas de dos caracteres

239 // notar que en las pruebas no hay partidos como 16a, por ejemplo

240 **return** ((idr[0]-'1')+(idr[1]-'a'));

241 }

242

243

244 **int** agregar\_resultado(base\_t\* base, **char**\* idr, **char**\* gloc, **char**\* gvis, cola\_t\* cola) {

245 **int** goleslocal;

246 **int** golesvisitante;

247 **int** posicion;

248 //consigo la posicion del partido

249 posicion = posicion\_partido(idr);

250

251 **if** (posicion == -1) **return** -3; // si el idr tiene mas de dos caracteres

252

253 //verifico si no esta cargado ya (no puede haber partidos que hayan salido 0-0 y esa es la condicion inicial)

254 **if** (((base->partidos[posicion]->goleslocal)!=0) || ((base->partidos[posicion]->golesvisitante)!=0)){

255 **return** -1;

256 }// ya esta cargado el resultado de ese partido

257

258 //verifico que esten los equipos cargados correctamente

259 **if** ((base->partidos[posicion]->local==NULL) || (base->partidos[posicion]->visitante==NULL)){

260 **return** -2;

261 }// faltan resultados previos, no se sabe que equipos corresponden

262

263 //consigo el puntero hacia el partido para modificarlo

264 partido\_t\* partido = base->partidos[posicion];

265 goleslocal = atoi(gloc);

266 partido->goleslocal = goleslocal;

267 golesvisitante = atoi(gvis);

268 partido->golesvisitante = golesvisitante;

269

270 //actualizo los goles del local en la base de datos

271 **char**\* dorsal;

272 **int** numdorsal;

273 **int** m = 0;

274 **while** (!cola\_esta\_vacia(cola) && (m<goleslocal)) {

275 dorsal = (**char**\*)cola\_desencolar(cola);

276 numdorsal = atoi(dorsal);

277 jugador\_metio\_gol(base,partido->local->nombre,numdorsal);

278 free(dorsal);

279 m++;

280 }

281 **int** n = 0;

282 **while** (!cola\_esta\_vacia(cola) && (n<golesvisitante)) {

283 dorsal = cola\_desencolar(cola);

284 numdorsal = atoi(dorsal);

285 jugador\_metio\_gol(base,partido->visitante->nombre,numdorsal);

286 free(dorsal);

287 n++;

288 }

289 heap\_heapify(base->goleadores);

290 **int** k,siguientepartido;

291 **if** (posicion==0) **return** 0; // si estoy en la final no hay siguiente partido

292 //como hay siguiente partido cargo el ganador segun corresponda

293 equipo\_t\* ganador;

294 **if** (goleslocal>golesvisitante) ganador=partido->local; **else** ganador=partido->visitante; //guardo el equipo ganador

295 siguientepartido = (posicion-1)/2; //calculo la posicion del siguiente partido en la base

296

297 k = posicion%2; //veo si viene de una posicion par o impar para saber si lo asigno como local o visitante

298 partido = base->partidos[siguientepartido]; //muevo el puntero hacia el siguiente partido

299 **if** (k==0) partido->visitante=ganador; **else** partido->local=ganador; //asigno el ganador segun corresponda en el siguiente partido

300

301 **return** 0;

302 }

303

304 **int** listar\_goleador(base\_t\* base){

305 jugador\_t\* goleador = heap\_ver\_max(base->goleadores);

306 printf("%s: %s Goles: %d\n",goleador->nombre,goleador->equipo,goleador->goles);

307 **return** 0;

308 }

309

310 **int** mostrar\_resultado(base\_t\* base,**char**\* idr){

311 **int** posicion = posicion\_partido(idr);

312 **if** (posicion == -1) **return** -3;

313 partido\_t\* partido = base->partidos[posicion];

314 //verifico que esten los equipos cargados correctamente

315 **if** ((base->partidos[posicion]->local==NULL) || (base->partidos[posicion]->visitante==NULL))**return** -2;//faltan resultados previos, no se sabe que equipos corresponden

316 //verifico si esta cargado el resultado (no puede haber partidos que hayan salido 0-0 y esa es la condicion inicial)

317 **if** (((base->partidos[posicion]->goleslocal)==0) && ((base->partidos[posicion]->golesvisitante)==0)) **return** -1; //no esta cargado el resultado

318 printf("resultado: %s:%d vs %s:%d\n",partido->local->nombre,partido->goleslocal,partido->visitante->nombre,partido->golesvisitante);

319 **return** 0;

320 }

321

322 #endif // MUNDIAL\_C