

06/27/18 07:37:20 /home/ivan/Desktop/Repositorio-Final/track.c

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  #include "types.h"
6  #include "track.h"
7  #include "errors.h"
8  #include "main.h"
9
10 extern char * errors_dictionary[MAX_ERRORS];
11
12 /*Diccionarios de formato, tipos de ordenes y generos*/
13
14 status_t (*format_output[MAX_FORMATS]) (void *, const void *, FILE *) =
15 {
16     ADT_track_export_to_csv,
17     ADT_track_export_to_xml
18 };
19
20 int (*sort_output[MAX_SORTS]) (void *, void *) =
21 {
22     ADT_track_compare_by_title,
23     ADT_track_compare_by_artist,
24     ADT_track_compare_by_genre
25 };
26
27 char * genres_dictionary[MAX_GENRES] =
28 {
29     "Blues", "Classic Rock", "Country", "Dance", "Disco", "Funk", "Grunge",
30     "Hip-Hop", "Jazz", "Metal", "New Age", "Oldies", "Other", "Pop",
31     "R&B", "Rap", "Reggae", "Rock", "Techno", "Industrial", "Alternative",
32     "Ska", "Death Metal", "Pranks", "Soundtrack", "Euro-Techno", "Ambient", "Trip-Hop",
33     "Vocal", "Jazz+Funk", "Fusion", "Trance", "Classical", "Instrumental", "Acid",
34     "House", "Game", "Sound Clip", "Gospel", "Noise", "AlternRock", "Bass",
35     "Soul", "Punk", "Space", "Meditative", "Instrumental Pop", "Instrumental Rock", "Ethnic",
36     "Gothic", "Darkwave", "Techno-Industrial", "Electronic", "Pop-Folk", "Eurodance", "Dream",
37     "Southern Rock", "Comedy", "Cult", "Gangsta", "Top 40", "Christian Rap", "Pop/Funk",
38     "Jungle", "Native American", "Cabaret", "New Wave", "Psychadelic", "Rave", "Showtunes",
39     "Trailer", "Lo-Fi", "Tribal", "Acid Punk", "Acid Jazz", "Polka", "Retro",
40     "Musical", "Rock & Roll", "Hard Rock", "Folk", "Folk-Rock", "National Folk", "Swing",
41     "Fast Fusion", "Bebob", "Latin", "Revival", "Celtic", "Bluegrass", "Avantgarde",
42     "Gothic Rock", "Progressive Rock", "Psychedelic Rock", "Symphonic Rock", "Slow Rock", "Big Band", "Chorus",
43     "Easy Listening", "Acoustic", "Humour", "Speech", "Chanson", "Opera", "Chamber Music",
44     "Sonata", "Symphony", "Booty Brass", "Primus", "Porn Groove", "Satire", "Slow Jam",
45     "Club", "Tango", "Samba", "Folklore", "Ballad", "Power Ballad", "Rhythmic Soul",
46     "Freestyle", "Duet", "Punk Rock", "Drum Solo", "A Capela", "Euro-House", "Dance Hall"
47 };
48
49 /*Esta función crea una nueva pista*/
50 status_t ADT_track_new (ADT_track_t ** track)
51 {
52     if (track == NULL)
53         return ERROR_NULL_POINTER;
54
55     if ((*track = (ADT_track_t*)malloc(sizeof(ADT_track_t))) == NULL)
56         return ERROR_OUT_OF_MEMORY;
57
58     (*track)->tag[0] = '\0';
59     (*track)->title[0] = '\0';
60     (*track)->artist[0] = '\0';
61     (*track)->album[0] = '\0';
62     (*track)->year[0] = '\0';
63     (*track)->comment[0] = '\0';
64     (*track)->genre = 0;
65
66     return OK;
67 }
68
69 /*Esta funcion destruye una pista*/
70 status_t ADT_track_delete (void * t)
71 {
72     ADT_track_t * track;
73
74     track = (ADT_track_t *)t;
75
76     if (track == NULL)
77         return ERROR_NULL_POINTER;
78
79     track->tag[0] = '\0';
80     track->title[0] = '\0';
81     track->artist[0] = '\0';
82     track->album[0] = '\0';
83     track->year[0] = '\0';
84     track->comment[0] = '\0';
85     track->genre = 0;
86
87     free(track);
88 }

```



```

180     if(fprintf(file_out, "%s%s\n", xml_contexts[2], xml_contexts[7], xml_contexts[3]) < 0)
181         return ERROR_WRITING_TO_FILE;
182
183     if(fprintf(file_out, "\t\t%s\n", xml_contexts[1], xml_contexts[8], xml_contexts[3]) < 0)
184         return ERROR_WRITING_TO_FILE;
185
186     if(fprintf(file_out, "%s", genres_dictionary[track->genre]) < 0)
187         return ERROR_WRITING_TO_FILE;
188
189     if(fprintf(file_out, "%s%s\n", xml_contexts[2], xml_contexts[8], xml_contexts[3]) < 0)
190         return ERROR_WRITING_TO_FILE;
191
192     if(fprintf(file_out, "\t\t%s\n", xml_contexts[2], xml_contexts[5], xml_contexts[3]) < 0)
193         return ERROR_WRITING_TO_FILE;
194
195     return OK;
196 }
197
198 /*Esta función compara dos pistas según el artista*/
199 int ADT_track_compare_by_artist (void * t1, void * t2)
200 {
201     size_t i;
202     ADT_track_t *track1, *track2;
203
204     track1 = (ADT_track_t *)t1;
205     track2 = (ADT_track_t *)t2;
206
207     if (track1 == NULL || track2 == NULL)
208         return 0;
209
210     for(i=0; track1->artist[i] && track2->artist[i]; i++)
211     {
212         if (track1->artist[i] != track2->artist[i])
213         {
214             return (track1->artist[i] - track2->artist[i]);
215         }
216     }
217     if (!track1->artist[i] && track2->artist[i])
218     {
219         return 1;
220     }
221     if (track1->artist[i] && !track2->artist[i])
222     {
223         return -1;
224     }
225     return 0;
226 }
227
228 /*Esta función compara dos pistas segun el nombre*/
229 int ADT_track_compare_by_title (void * t1, void * t2)
230 {
231     size_t i;
232     ADT_track_t *track1, *track2;
233
234     if (t1 == NULL || t2 == NULL)
235         return 0;
236
237     track1 = (ADT_track_t *)t1;
238     track2 = (ADT_track_t *)t2;
239
240
241     for(i=0; track1->title[i] && track2->title[i]; i++)
242     {
243         if (track1->title[i] != track2->title[i])
244         {
245             return (track1->title[i] - track2->title[i]);
246         }
247     }
248     if (!track1->title[i] && track2->title[i])
249     {
250         return 1;
251     }
252     if (track1->title[i] && !track2->title[i])
253     {
254         return -1;
255     }
256     return 0;
257 }
258
259 /*Esta función compara dos pistas segun el género*/
260 int ADT_track_compare_by_genre (void * t1, void * t2)
261 {
262     ADT_track_t *track1, *track2;
263
264     if (t1 == NULL || t2 == NULL)
265         return 0;
266
267     track1 = (ADT_track_t *)t1;
268     track2 = (ADT_track_t *)t2;
269
270     return track1->genre - track2->genre;

```

271 }