06/28/18 02:35:35 /home/ivan/Desktop/Repositorio-Final/mp3_processor.c

```c
#include <stdio.h>
#include <stdlib.h>

#include "mp3_processor.h"

extern status_t (*format_output[MAX_FORMATS])(void *, const void *, FILE *);
extern int (*sort_output[MAX_SORTS]) (void *, void *);

/*Diccionario de formatos*/
char * format_dictionary[MAX_FORMATS] = {
    CSV_FORMAT,
    XML_FORMAT
};

/*Diccionario de ordenamientos*/
char * sort_dictionary[MAX_SORTS] = {
    SORT_BY_NAME,
    SORT_BY_ARTIST,
    SORT_BY_GENRE
};

/*Esta función se ocupa de insertar un track en un vector*/
status_t process_mp3_data(setup_t * setup, FILE * fi, ADT_Vector_t ** vector)
{
    status_t st;
    char header[MAX_HEADER_SIZE];
    ADT_track_t * track;

    if(setup == NULL || fi == NULL || vector == NULL)
        return ERROR_NULL_POINTER;

    if((st = ADT_track_new(&track)) != OK)
        return ERROR_INVALID_TRACK;

    if((st = get_mp3_header(fi, header)) != OK) {
        if((st = ADT_track_delete(&track)) != OK)
            return st;

        return st;
    }

    if((st = ADT_track_set(header, track)) != OK)
        return st;

    if((st = ADT_Vector_append_element(vector, track)) != OK)
        return st;

    return OK;
}

/*Lee el "header" del archivo mp3*/
status_t get_mp3_header(FILE * fi, char header[])
{
    size_t length;

    if(fi == NULL)
        return ERROR_NULL_POINTER;


    if ((fseek(fi, 0, SEEK_END)) != OK)
        return ERROR_INVALID_MP3_FILE;

    length=ftell(fi);

    if((fseek(fi,length - MP3_HEADER_SIZE, SEEK_SET)) != OK)
        return ERROR_INVALID_MP3_FILE;

    if (fread(header, sizeof(char), MP3_HEADER_SIZE, fi) != MP3_HEADER_SIZE)
        return ERROR_INVALID_MP3_FILE;

    return OK;
}

status_t get_tracks_from_mp3_files (int argc, char * argv[], ADT_Vector_t ** vector, setup_t * setup)
{
    FILE * mp3_file;
    size_t i;
    status_t st;

    for(i = 0; i < argc - INDEX_FIRST_MP3_FILE; i++ )
```

```
 81      {
 82
 83          if((mp3_file = fopen(argv[INDEX_FIRST_MP3_FILE + i], "rt")) == NULL)
 84          {
 85              st = ERROR_INVALID_MP3_FILE;
 86              return st;
 87          }
 88
 89          if((st = process_mp3_data(setup, mp3_file, vector)) != OK)
 90          {
 91              fclose(mp3_file);
 92              return st;
 93          }
 94
 95          if((fclose(mp3_file)) == EOF)
 96          {
 97              st = ERROR_CLOSING_FILE;
 98              return st;
 99          }
100      }
101      return OK;
102  }
103
104  status_t set_vector_for_tracks (ADT_Vector_t * vector, setup_t * setup)
105  {
106      status_t st;
107
108      if((st = ADT_Vector_set_printer (vector, format_output[setup->doc_type])) != OK)
109          return st;
110
111      if((st = ADT_Vector_set_comparator (vector, sort_output[setup->sort_by])) != OK)
112          return st;
113
114      if((st = ADT_Vector_set_destructor(vector, ADT_track_delete)) != OK)
115          return st;
116
117      return OK;
118  }
119
120  status_t export_tracks_vector (ADT_Vector_t * vector, char * argv[], void * context, size_t out_index,
     setup_t * setup)
121  {
122      FILE * file_out;
123      status_t st;
124
125      if ((file_out = fopen(argv[out_index], "wt")) == NULL)
126      {
127          st = ERROR_INVALID_OUTPUT_FILE;
128          ADT_Vector_delete(&vector);
129          return st;
130      }
131      if((st = ADT_Vector_export(vector, context, file_out, setup)) != OK)
132      {
133          ADT_Vector_delete(&vector);
134          fclose(file_out);
135          return st;
136      }
137      if((st = ADT_Vector_delete(&vector)) != OK)
138      {
139          fclose(file_out);
140          return st;
141      }
142      if((fclose(file_out)) == EOF)
143      {
144          st = ERROR_CLOSING_FILE;
145          return st;
146      }
147      return OK;
148  }
149
150  status_t process_mp3_files (size_t out_index, void * context, int argc, char * argv[], setup_t * setup)
151  {
152      ADT_Vector_t * vector;
153      status_t st;
154
155      if((st = ADT_Vector_new(&vector)) != OK)
156          return st;
157      if((st = set_vector_for_tracks (vector, setup)) != OK)
158      {
159          ADT_Vector_delete(&vector);
160          return st;
161      }
162      if((st = get_tracks_from_mp3_files(argc, argv, &vector, setup)) != OK)
```

```
163     {
164         ADT_Vector_delete(&vector);
165         return st;
166     }
167     if((st = ADT_Vector_sort_elements(&vector, ADT_Vector_swap_elements)) != OK)
168     {
169         ADT_Vector_delete(&vector);
170         return st;
171     }
172     if((st = export_tracks_vector (vector, argv, context, out_index, setup)) != OK)
173         return st;
174 
175     return OK;
176 }
```