Tema1 - Music Playlist

Responsabili:

- Alexandru Mihai Stroie [mailto:mailto:andistroie@gmail.com]
- Horia Nedelcu [mailto:mailto:nedelcu.horia.alexandru@gmail.com]
- Data publicării: 08.03.2020
- Deadline: 22.03.2020 23:55:00

Modificări și actualizări

- publicare temă: 08.03.2020
- **modificare cerinta comanda ADD: 11.03.2020**

Obiective

- Aprofundarea cunoștințelor în utilizarea limbajului C
- Implementarea și utilizarea structurii de date listă dublu înlănțuită

Introducere

Dobre, băiat excentric și student la medicină, dorește ca petrecerea de ziua lui din acest an să fie una memorabilă. Pentru a avea o petrecere flamboaiantă, el își roagă fratele, Gogo, proaspăt inginer in calculatoare să îi creeze un player muzical special pentru această ocazie. Deoarece Gogo este ocupat cu munca, dar este și un tip responsabil, nu vrea să îl dezamăgească pe Dobre, se apucă să caute informații despre fișierele MP3:

Metadate MP3 – ID3 TAG

ID3 este un container de metadate cel mai des utilizat în combinație cu formatul de fișiere audio MP3. Acesta permite ca informații (titlul, artistul, albumul, numărul piesei, genul, etc.) despre fișier să fie stocate chiar în interiorul fișierului. Containerul ID3v1 ocupă 128 de bytes, începând cu șirul TAG și este situat la șfârșitul fișierului pentru a menține compatibilitatea cu media playere mai vechi. Câmpurile containerului ID3v1 sunt:

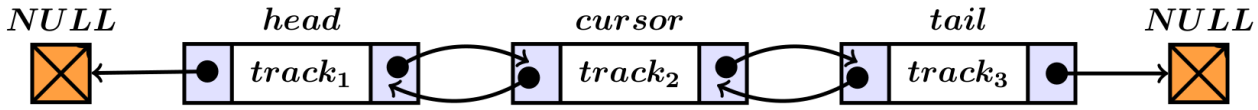
Câmp	Lungime	Descriere
antet	3	“TAG”
titlu	30	30 de caractere pentru titlu
artist	30	30 de caractere pentru numele artistului
album	30	30 de caractere pentru numele albumului
an	4	4 cifre pentru anul apariției
comentariu	28[3] 30	maxim 30 de caractere pentru comentariu
zero-byte[3]	1	dacă numărul de pistă e stocat acest octet va conține un 0 binar
piesă[3]	1	numărul piesei de pe album sau 0; invalid dacă octetul anterior nu este un 0 binar
gen	1	indice într-o listă de genuri sau 255

[3]. Numărul piesei este stocat în ultimii doi octeți din câmpul de comentariu. Dacă comentariul este de 29 sau 30 de caractere, numarul de pistă nu poate fi stocat.

Metadatele nu sunt neapărat şiruri de caractere terminate cu caracterul nul. Trebuie să ne asigurăm că punem caracterul terminal sau să fim atenți când folosim funcții de prelucrare a șirurilor de caractere.

Cerință

Ajută-l pe Gogo să se asigure că termină playerul muzical la timp prin implementarea unei structuri de date care să gestioneze lista de melodii a playerului. Media playerul poate fi implementat folosind o listă dublu înlănțuită, căreia îi atașăm un nod auxiliar **cursor** care pointează către melodia curentă în redare, conform figurii următoare:



Input

Pe prima linie se va afla un număr **q**, reprezentând numărul de comenzi care se vor efectua asupra playerului muzical. Pe următoarele **q** linii se pot afla următoarele tipuri de comenzi:

- **De adăugare**
  1. **ADD_FIRST <nume melodie>** → adaugă o melodie la începutul listei
  2. **ADD_LAST <nume melodie>** → adaugă o melodie la finalul listei
  3. **ADD_AFTER <nume melodie>** → adaugă o melodie după cursor

- **<nume melodie>** reprezintă **doar** numele fișierului din care trebuie citite metadatele melodiei, nu și calea către aceasta.
- Dacă o melodie deja există în playlist, ~~aceasta este mutată în noua poziție~~ aceasta este mai întâi ștearsă și apoi readăugată în noua poziție.
- La adăugarea unei melodii într-un playlist gol, poziția cursorului va fi inițializată la începutul listei.
- Se poate adăuga o melodie (modifica poziția unei melodii) dupa cursor doar dacă melodia diferă de cea către care cursorul pointează, altfel se ignoră comanda.
- Se poate adăuga după cursor o melodie, doar dacă acesta există, altfel se ignoră comanda.

- **De ștergere**
  1. **DEL_FIRST** → șterge melodia de la începutul listei
  2. **DEL_LAST** → șterge melodia de la finalul listei
  3. **DEL_CURR** → șterge melodia curentă (melodia către care pointează cursorul)
  4. **DEL_SONG <nume melodie>** → șterge melodia, dacă aceasta există în listă

- La fiecare ștergere se verifică dacă este necesară mutarea cursorului. Cursorul se va muta la melodia următoare dacă este posibil, dacă nu, se va muta la melodia anterioară.
- Dacă nu este posibil să ștergem de la început sau de la finalul listei, afișăm mesajul **“Error: delete from empty playlist\n”**.
- Dacă nu este posibil să ștergem melodia curentă, afișăm mesajul **“Error: no track playing\n”**.
- Dacă nu există melodie de șters, afișăm mesajul **“Error: no song found to delete\n”**.

- **De deplasare cursor**
  1. **MOVE_NEXT** → mută cursorul la următoarea melodie (dacă s-a ajuns la finalul listei, comanda este ignorată)
  2. **MOVE_PREV** → mută cursorul la melodia anterioară (dacă s-a ajuns la începutul listei, comanda este ignorată)

- Dacă nu există cursor, afișăm mesajul **“Error: no track playing\n”**.

- **De afișare**
  1. **SHOW_FIRST** → afișează informații despre melodia de la începutul listei
  2. **SHOW_LAST** → afișează informații despre melodia de la finalul listei
  3. **SHOW_CURR** → afișează informații despre melodia curentă (cea pe care este setat cursorul)
  4. **SHOW_PLAYLIST** → afișează pentru toate melodiile din listă titlul

- Dacă nu există informații de afișat în cadrul primelor 3 comenzi, afișăm mesajul **“Error: show empty playlist\n”**.

Formatul de afișare pentru informațiile melodiilor este:

```
"Title: <titlu melodie>\n"
"Artist: <artist melodie>\n"
"Album: <album melodie>\n"
"Year: <an melodie>\n"
```

Formatul de afișare pentru playlist este:

```
"[<titlu melodie_1>; <titlu melodie_2>; … <titlu melodie_n>]\n"
```

Comenzile se vor citi din fișierul **<media.in>** și outputul se va afișa în fișierul **<media.out>**. Numele fișierelor de in/out se vor da ca parametri în linie de comandă. Pentru simplitate, am ales ca în această temă, containerul cu metadate să conțină doar: TAG, titlu, artist, albumul și anul apariției (97 bytes în loc de 128 bytes cum sunt în standardul ID3). Tot pentru a simplifica lucrul cu fișierele .mp3 **“melodiile”** folosite în tema vor fi fișiere generate care se vor găsi în directorul **“songs”** din arhiva de testare. O **“melodie”** din directorul **“songs”** va avea următoarea structură:

Garbage	TAG	Title	Artist	Album	Year

* secțiunea de **garbage** reprezintă bytes care ar fi folosiți pentru a stoca efectiv melodia

Output

Fișierul **<media.out>** va conține atât informațiile care au fost cerute, cât și erorile comenzilor când acestea nu au putut fi executate, în ordinea primirii lor din fișierul de intrare.

Exemplu

```
media.in:
10
DEL_FIRST
ADD_LAST Metallica - Battery.mp3
ADD_FIRST Gorillaz - Feel Good Inc.mp3
ADD_AFTER NOMA - Brain power.mp3
SHOW_CURR
MOVE_NEXT
DEL_LAST
MOVE_PREV
SHOW_CURR
SHOW_PLAYLIST

media.out:
Error: delete from empty playlist
Title: Battery
Artist: Metallica
Album: Master Of Puppets
Year: 1985
Title: Feel Good Inc
Artist: Gorillaz
Album: Feel Good Inc
Year: 2005
[Feel Good Inc; Battery]
```

Precizări

Structura playlistului va trebui să implementeze o listă dublu înlănțuită. Nerespectarea acestui lucru va aduce depunctări.

Având în vedere structura metadatelor, recomandăm să folosiți pentru printare formatul **"%.*precision*s"**, unde ***precision*** specifică numărul maxim de caractere care trebuie tipărite până la întâlnirea caracterului terminal nul.

Checker

Teste locale: checker-tema1-2020.zip

Temele vor fi trimise pe vmchecker [https://elf.cs.pub.ro/vmchecker/ui/#SD]. **Atenție!** Temele trebuie trimise în secțiunea **Structuri de Date (CA)**.

Arhiva trebuie să conțină:

- surse
- fișier Makefile cu două reguli:
  - regula **build**: în urma căreia se generează un executabil numit **tema1**
  - regula **clean** care şterge executabilul si fișierele obiect
- fișier **README** care să conțină detalii despre implementarea temei

Punctaj

Atenite! O temă care nu compilează va primi 0 puncte.

1. 80p teste
2. **Fiecare** test este verificat cu valgrind. Dacă un test are memory leaks, nu va fi punctat.
3. 20p README + comentarii/claritate cod (ATENȚIE! Fișierul README trebuie făcut explicit, cât să se înțeleagă ce ați făcut în sursă, dar fără comentarii inutile și detalii inutile).
4. Se acordă 20% din punctajul obținut pe teste, ca bonus pentru coding style. De exemplu, pentru o temă care obține maxim pe teste, se pot obține 20p bonus dacă nu aveți erori de coding style. Pentru o temă ce trece 18 teste din 20, se pot obține 18p dacă nu aveți erori de coding style.
5. O temă care obține 0p pe vmchecker este punctată cu 0.
6. Temele au deadline hard. Prin urmare, o temă trimisă dupa deadline este punctată cu 0.

Nu copiați! Toate soluțiile vor fi verificate folosind o unealtă de detectare a plagiatului. În cazul detectării unui astfel de caz, atât plagiatorul cât și autorul original (nu contează cine e) vor primi punctaj 0 pe **toate temele**!

De aceea, vă sfătuim să nu vă lăsați rezolvări ale temelor pe calculatoare partajate (la laborator etc), pe mail/liste de discuții/grupuri etc.

FAQ

Q: Se pot folosi flag-uri de optimizare?

A: Nu aveți voie să folosiți flag-uri de optimizare în Makefile (-O3, -O2, etc.).

Link-uri utile

http://www.cplusplus.com/reference/cstdio/printf/ [http://www.cplusplus.com/reference/cstdio/printf/]
https://ieeexplore.ieee.org/abstract/document/1524886 [https://ieeexplore.ieee.org/abstract/document/1524886]
http://www.loc.gov/preservation/digital/formats/fdd/fdd000106.shtml [http://www.loc.gov/preservation/digital/formats/fdd/fdd000106.shtml] https://prezi.com/zwbjyii_fsih/creation-of-a-music-playlist/ [https://prezi.com/zwbjyii_fsih/creation-of-a-music-playlist/] https://www.geeksforgeeks.org/applications-of-linked-list-data-structure/ [https://www.geeksforgeeks.org/applications-of-linked-list-data-structure/]