UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

MATEUS TICIANELI SARTORIO
YURI AIKAU DE CASTRO REIS SANCHEZ

RELATÓRIO DO TRABALHO I DE ESTRUTURAS DE DADOS I

Vitória

2021

MATEUS TICIANELI SARTORIO
YURI AIKAU DE CASTRO REIS SANCHEZ

RELATÓRIO DO TRABALHO I DE ESTRUTURAS DE DADOS I

Relatório de Trabalho apresentado à disciplina de Estrutura de Dados I do curso de graduação em Engenharia de Computação da Universidade Federal do Espírito Santo, como requisito parcial para avaliação. Profª. Patrícia Dockhorn Costa.

Vitória
2021

SUMÁRIO

1. INTRODUÇÃO

O problema proposto consiste basicamente em ler arquivos de texto contendo relações de amizade entre pessoas e músicas contidas em playlists e então armazenar esses dados em tipos abstratos de dados (TAD's).

Os TAD's criados consistem basicamente de um TAD principal, que contém a lista de pessoas lida. Cada pessoa é também um TAD que contém um nome, uma lista de amigos e uma lista de playlists. A figura 1 ilustra o funcionamento geral das estruturas de dados implementadas.

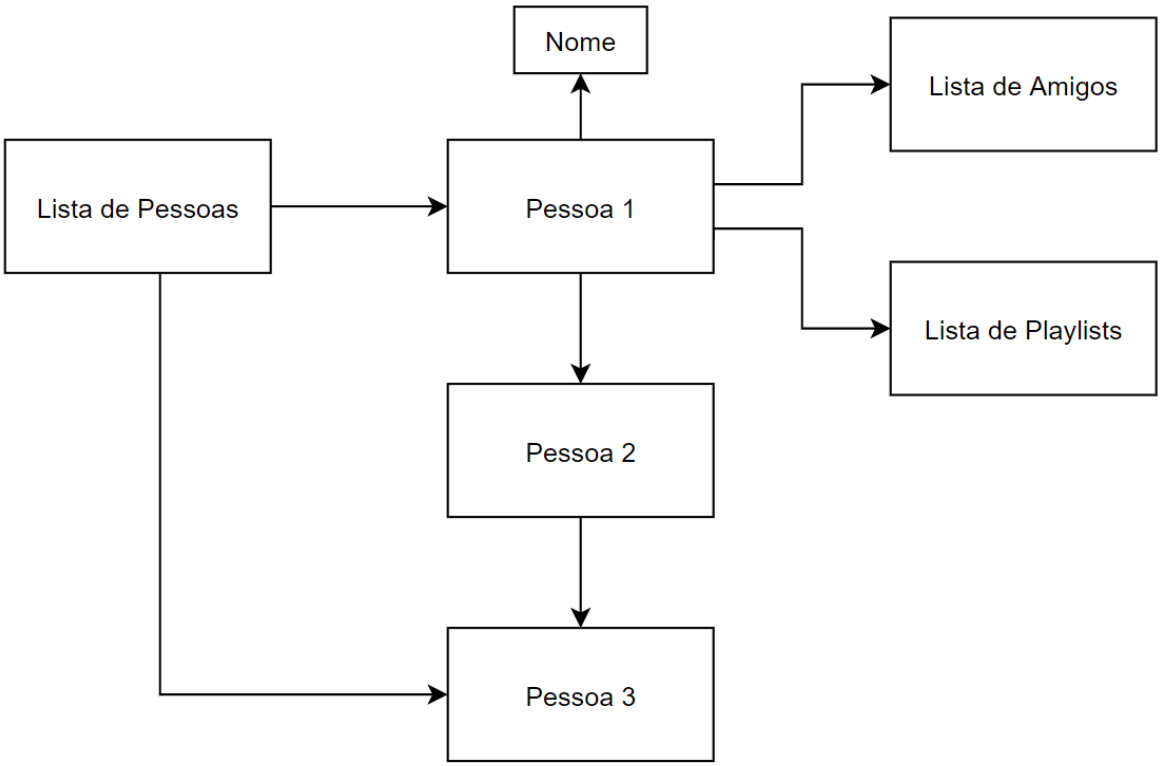

Figura 1 - visão geral das estruturas de dados implementadas.

2. IMPLEMENTAÇÃO

O programa consiste em basicamente 4 TAD's (musica.h, playlist.h, pesoas.h, listaPessoas.h) e 1 cliente (main.c), onde para podermos chegar ao TAD mais geral, no caso o listaPessoas.h, primeiro precisamos analisar o funcionamento do musica.h que o elemento mais simples da estrutura geral que foi proposta, e assim por diante.

2.1. musica.h

Primeiramente dentro desse TAD temos uma estrutura do tipo *musica_st* que contém os campos que armazenam os nomes do cantor/banda e o nome da música em si.

A partir disso temos a uma função que inicializa a estrutura *musica_st* na memória e preenche os campos com as informações passadas pelo parâmetro da função (eles também são alocados).

Também temos funções de retorno dos campos da estrutura para que outros arquivos também consigam acessá-los fora do musica.c.

E por fim uma função que libera toda a memória alocada oriunda dessa estrutura.

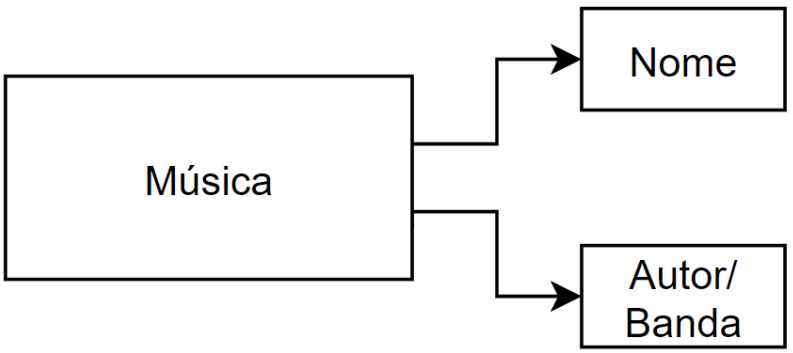

Figura 2 - TAD Musica.

2.2. playlist.h

Agora no TAD de playlists iremos utilizar a estrutura anteriormente criada (*musica_st*) como um dos campos das células da lista de playlists que iremos fazer, ou seja, criaremos uma célula que contém os campos *musica_st* e um ponteiro para próxima célula. Tendo a estrutura de célula criada, agora podemos definir uma a estrutura de *Playlist* que terá os campos com os ponteiros para a primeira e última célula da lista e o seu próprio nome, ou seja, ela será a sentinela dessa lista de músicas.

Como na *musica.h*, também teremos uma função que inicializa a estrutura da playlist na memória, mas já que essa estrutura é um lista encadeada com sentinela

então iremos inicializar seus campos com *NULL*. E sendo ela uma lista também haverá um função de inserção de músicas nessa lista, inserindo ao final da lista.

Além disso, também criamos uma função auxiliar para podermos percorrer a playlist e retornarmos um ponteiro para *música_st* em um determinado índice da lista. Isso será útil para quando precisarmos acessar a estrutura playlist fora desse arquivo (*playlist.c*).

Em relação a leitura de arquivos, temos uma função que lê as informações contidas no arquivo *playlists.txt* para criarmos uma estrutura de *playlists* com o nome da playlist contido no arquivo de entrada e criar as células de música de acordo com as músicas dentro do .txt do nome da playlist.

E também, por fim teremos uma função que destrói todos os blocos de memória alocados devido a estrutura de playlists, e também destrói as músicas de cada célula, usando a função de liberação que se observa no *musica.h*.

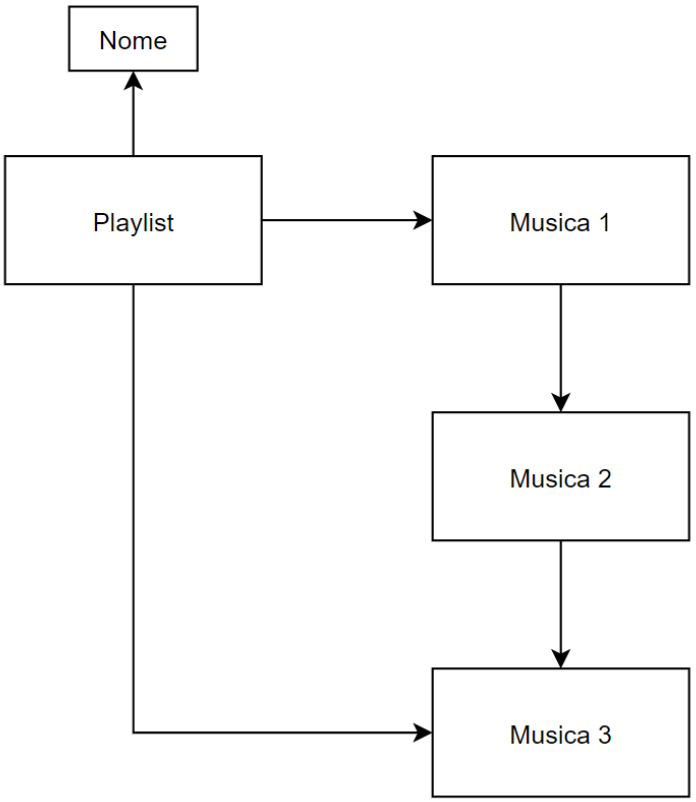

Figura 3 - TAD Playlist.

2.3. pessoa.h

Um dos TAD's principais do programa é o tipo *Pessoa*. Ele consiste de uma string (*char* nome*) que armazena o nome da pessoa, de uma lista de amigos (*ListaAmigos* listaAmigos*) e uma lista de playlists (*ListaPlaylists* listaPlaylists*). A *listaAamigos* é a sentinela de uma lista encadeada homogênea cujos elementos são strings com o nome de cada amigo. A *listaPlaylists* é a sentinela de uma lista

encadeada homogênea cujos elementos são TAD's do tipo *Playlist*. A figura 4 mostra um esquema do TAD *Pessoa*.

Em pessoas.c, existe uma função que inicializa a estrutura de *Pessoa* na memória, mas já que essa estrutura possui ponteiros para duas listas encadeadas (*listaAmigos* e *listaPlaylists*), então iremos inicializar também essas duas listas. Além disso, existe também uma função de inserção de playlists na pessoa (insere a playlist na lista de playlists) e uma função de inserção de amigo (insere o amigo na lista de amigos).

Além disso, existe em *pessoas.h* uma função que insere uma música em uma lista de playlists por nome (*insereMusicaPorNome()*), ou seja, basta passar um ponteiro para a lista de playlists, o de uma playlist dentro desta lista, e a música a ser adicionada, que a função adiciona a música na playlist correta dentro da lista de playlists. Essa função teve grande importância no desenvolvimento do programa e serviu de função de função auxiliar para várias outras funções.

Uma outra função muito importante foi *refatorada()*, que recebe um ponteiro para uma estrutura do tipo *Pessoa* e cria uma nova lista de playlists para pessoa, onde cada playlist possui apenas músicas do mesmo autor ou banda. Ao final, a lista de playlists original da pessoa é deletada e a nova lista é colocada no final. A função *refatorada()* serviu como função auxiliar para a função *refatoradaLista()* presente dentro do *listaPessoa.h*.

Existe também a função *similaridades()* recebe dois ponteiros para duas estruturas do tipo *Pessoa* e retorna quantas músicas iguais essas duas pessoas têm. A função *similaridades()* serviu de função auxiliar para a função *similaridadesLista()* presente em *listaPessoas.h.*

Existe também a função *imprimePlaylistsArquivo()* que recebe um ponteiro para uma estrutura do tipo *Pessoa* e cria um arquivo para cada uma de suas playlists e imprime dentro dele todas as músicas contidas na playlist.

Por fim, existe a função *destroiPessoa()*, que recebe um ponteiro para uma estrutura do tipo *Pessoa* e libera a memória da lista de amigos (e de todos os amigos contidos dentro dela) e da lista de playlists (e de todas as playlists contidas nela e de todas as músicas contidas nas playlists).

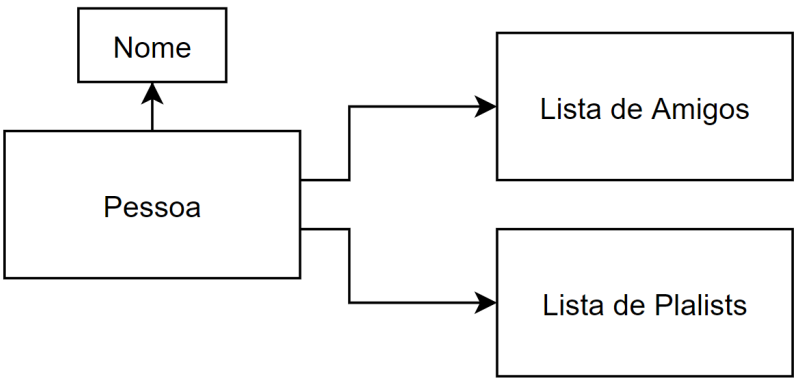

Figura 4 - TAD Pessoa.

2.4. listaPessoa.h

Por fim, no TAD de lista de pessoas é basicamente uma lista homogênea encadeada simples cujos elementos são estruturas do tipo Pessoa definidos anteriormente. A figura 5 mostra um esquemático do TAD *ListaPessoas*.
Dentro de *listaPessoa.h* existe uma função chamada *inicListaPessoa()* que aloca memória para a sentinela da lista de pessoas. Além disso, existe uma função de inserção que insere uma pessoa na lista de pessoas (ao final).

Uma das funções mais essenciais do trabalho é *leArquivos()*, que é responsável por ler todos os arquivos de entrada e inicializar as estruturas *pessoas* (e a lista de amigos e playlists de cada pessoa, além de inicializar as músicas e playlists também) e encadear cada pessoa na lista de pessoas.

Uma outra função essencial é a *refatoradaLista()*, que é responsável por iterar pela lista de pessoas e aplicar a função refatorada para cada pessoa (ver função refatorada() explicada na seção 2.3).

Existe também a função *similaridadeiLista()*, que é responsável por iterar pela lista de pessoas e aplicar a função *similaridade()* (ver seção 2.3) para cada par de pessoas da lista de pessoas.

A função *imprimePlaylistsArquivoLista()* itera por toda a lista de pessoas e chama a função *imprimePlaylistsArquivo()* (ver seção 2.3) para cada pessoa da lista.

Por fim, existe a função *destroiListaPessoa()*, que recebe uma lista de pessoas e chama a função *destroiPessoa()* para cada pessoa da lista (ver seção 2.3) e ao final libera a memória alocada pela sentinela da lista.

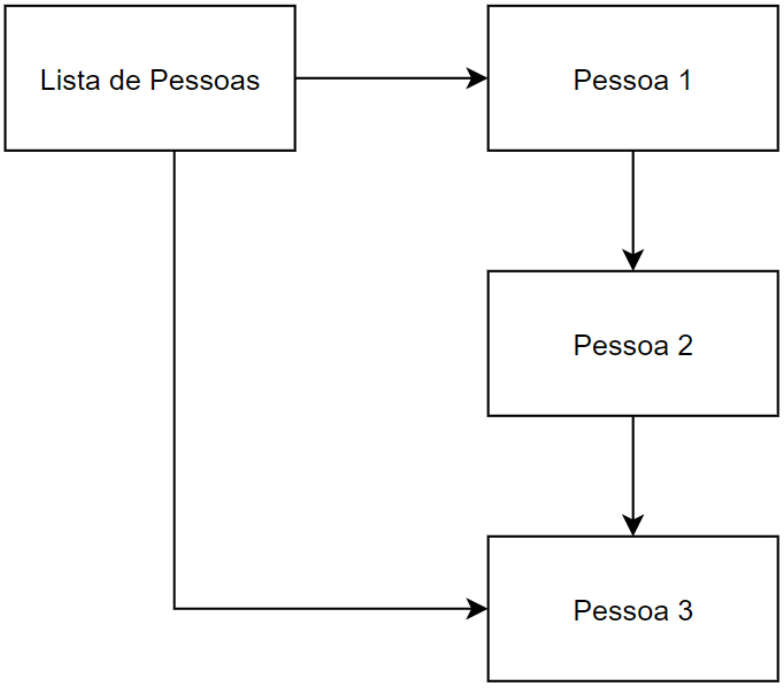

Figura 5 - TAD ListaPessoas.

Finalmente, a figura 6 mostra um esquema com todas as estruturas de dados explicitadas.

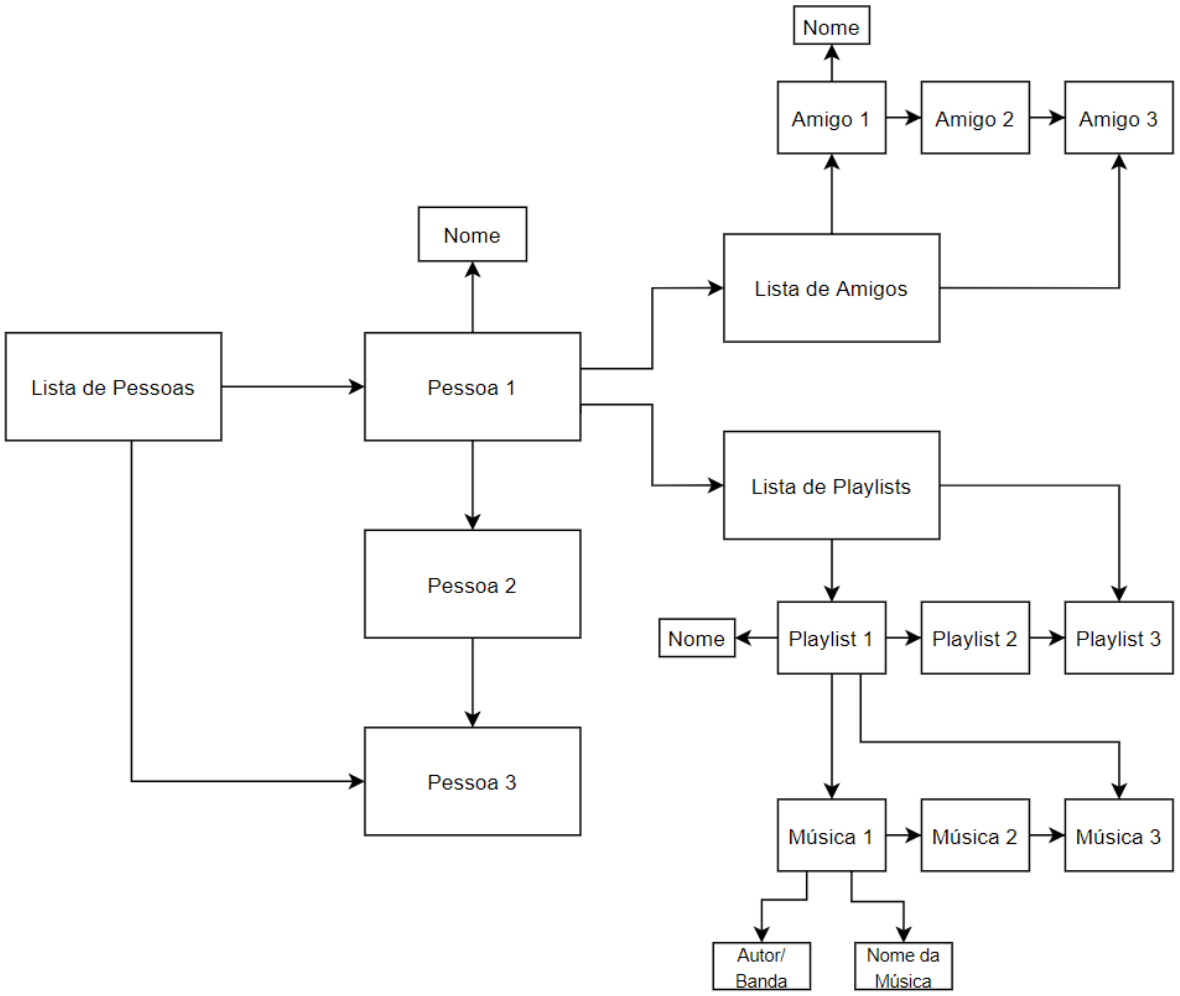

Figura 6 - estruturas de dados implementadas.

3. CONCLUSÃO

O trabalho foi desafiador, pois envolveu o encadeamento de listas dentro de outras listas, o que requer um nível de entendimento profundo das estruturas de dados envolvidas. Além disso, foi necessário usar a criatividade para resolver os problemas propostos usando listas.

A principal dificuldade encontrada no trabalho foi na construção das listas encadeadas. Inicialmente a ideia era construir apenas um arquivo *lista.h* que iria conter uma lista heterogênea, e então todas as estruturas que usassem listas iriam usar a implementação da *lista.h*. No entanto, surgiram problemas ao tentar incluir *lista.h* em *pessoas.h*, e o *lista.h* em *pessoas.h*, uma vez que era preciso usar o tipo de dados *Pessoa* para fazer uma função de inclusão da estrutura *Pessoa* na lista. Isso causou um problema de recursão, em que as bibliotecas ficavam incluindo umas às outras em um loop infinito. A solução encontrada foi criar listas homogêneas independentes dentro de cada TAD.

Um outro problema encontrado foi ao criar as funções *refatoradaLista()* e *similaridadesLista()*, que eram funções que precisavam ficar em *listaPessoa.c*, pois precisavam acessar a lista de pessoas, mas no entanto elas não tinham acesso às estruturas que não estavam no mesmo arquivo, como *pessoa_st ou playlist_st*. Então precisamos fazer a maioria das funções dentro de *pessoas.c* para acessarmos as estruturas mais elementares de maneira mais direta. Para isso contamos com funções auxiliares que servem de retorno em cada estrutura para podermos acessar os conteúdos dos campos dessas estruturas diretamente e assim fazer as devidas operações para obtermos os resultados desejados.

4. BIBLIOGRAFIA

[1] CELES, W; CERQUEIRA, R; RANGEL NETTO, JM. Introdução a estruturas de dados: com técnicas de programação em C.

[2] Rio de Janeiro: Campus, 2004., 2004. (Série Editora Campus/SBC). ZIVIANI, N. Projeto de algoritmos: com implementações em PASCAL e C. São Paulo, SP: Cengage Learning, 2011., 2011.

[3] SZWARCFITER, JL; MARKENZON, L. Estruturas de dados e seus algoritmos. Rio de Janeiro: Livros Técnicos e Científicos, c1994., 1994.