Introdução

Primeiramente o problema a ser solucionado e que preciso e ler os arquivos txts para inserir as pessoas, as amizades das mesmas, playlist e as músicas. No arquivo txt cada pessoas e separada por “;” e logo abaixo suas relações de amizade colocando a pessoa em primeiro e em seguida seu amigo. Fazendo isso monta-se uma função para ler o arquivo txt, verificando se está tudo certo de acordo com o que é pedido.

Após as amizades adiciona-se as playlists, sendo que a ordem no arquivo é nome das pessoas a adicionar as playlists, a quantidade de playlists a ser adicionada e o nome das playlists, sendo que todas as informações no final seguidas por “;”. Em seguida e feita uma função parecida com a de ler o arquivo txt de amizades verificando se está tudo correto no arquivo txt de playlists.

Dando continuidade ao programa são inseridas as músicas nas playlists através de arquivos txts com o estilo de musica da bandas (nacional,heavy metal,etc).

Logo após separa as playlist por artistas/banda. É baseado na relação de amizade de cada um faço a combinação da playlists de um cantor, podendo cada amigo ouvir e ver as músicas e playlists que cada um tem. Cria a pasta com os arquivos finais para ser utilizado no site tunimymuisic.com.

Implementação

Descrevendo as funções:

lista criaLista(){

cria a lista para pessoas(struct lista) para guardar as informações fornecidas, alocando dinamicamente a mesma e retorna ela mesma colocando valores default.

O(1)

}

TPessoa criaPessoa(char nome[]){

cria uma struct de TPpessoas para guardar informações fornecidas, alocando dinamicamente a mesma e retorna ela mesma colocando valores default.

O(1)

}

Void insereFim(lista l, TPessoa p){

Insere as informações fornecidas na struct TPessoa para guardar as mesma, colocando as informações no fim da struct através de ponteiros.

O(n2)

}

TPessoa pesquisaPessoa(lista l, char nome[]){

Pesquisa uma pessoa na struct TPessoa para verificar se a mesma informada e na struct se encontrar retorna essa pessoa informada(return p) caso n encontre retornal NULL.

O(1)

}

TAmigos criaAmigos(TPessoa p){

cria uma struct de TAmigos para guardar informações fornecidas, alocando dinamicamente a mesma e retorna ela mesma colocando valores default.

O(1)

}

Void insereAmigo(TPessoa p, TAmigo amig){

Insere as informações fornecidas na struct TPessoa para guardar as mesma, colocando as informações no fim da struct através de ponteiros.

O(n2)

}

TPlayLlist criaPlaylist(char *playlist){

cria uma struct de TPlaylist para guardar informações fornecidas, alocando dinamicamente a mesma e retorna ela mesma colocando valores default.

O(1)

}

TMusica criaMusica(char *musica, char *banda){

cria uma struct de TMusica para guardar informações fornecidas, alocando dinamicamente a mesma e retorna ela mesma colocando valores default.

O(1)

}

Void insereMusica(Playlist pl, TMusica music){

Insere as informações fornecidas na struct TMusica para guardar as mesma, colocando as informações no fim da struct através de ponteiros.

O(n2)

}

Void inserePlaylist(TPessoa p, TMusica music){

Insere as informações fornecidas na struct TPlaylist para guardar as mesma, colocando as informações no fim da struct através de ponteiros.

O(n2)

}

Int listaInserePessoaAmigo(lista l, char *amixade){

Abre o arquivo informado através do FILE para adicionar pessoas e os amigos na struct TPessoa e TAmigo respectivamente, logo apos abro o arquivo e verifico se seu tudo certo caso não retorno -1. Usando as funções fgets (coloca o tamanho da linha e o arquivo a ser lido), strtok(delimito ate onde o caracter vai se lido no caso ate o “;”), uso a função insereFim e insereFimAmigo para poder guardar as informações na struct. No final fecho o aquivo através do fclose;

0(n2)

}

Int listaInserePlayList(lista l, char playlistTXT){

Abre o arquivo informado através do FILE para adicionar as playlists na struct TPlayList, logo após abro o arquivo e verifico se seu tudo certo caso não retorno -1. Usando as funções fgets (coloca o tamanho da linha e o arquivo a ser lido), strtok(delimito ate onde o caracter vai se lido no caso ate o “;”), uso a função inserePlayList para poder guardar as informações na struct. No final fecho o aquivo através do fclose;

0(n2)

}

Int carregaMusicaPlayList(lista l){

Abre o arquivo informado através do FILE para adicionar as musicas na struct TMusica que no caso se conecta a TPlayList por poteiros, logo após abro o arquivo e verifico se seu tudo certo caso não

retorno -1. Usando as funções fgets (coloca o tamanho da linha e o arquivo a ser lido), strtok(delimito ate onde o caracter vai se lido no caso ate o “;”), uso a função insereMusica para poder guardar as informações na struct. No final fecho o aquivo através do fclose;

0(n2)

}

TPlayList pesquisaPlayListNome(TPessoa p, char *nomePlayList){

Pesquisa uma playLista na struct TPlayList para verificar se a mesma informada e na struct se encontrar retorna essa playList informada(return pl) caso n encontre retornal NULL.

O(1)

}

TMusica recortaMusicaPlayList(TPlayList pl, char*nomeBanda){

“Recorta” as musicas das playlista colocando as musicas em suas respectivas bandas, primeiro verifico se a struct TMusica não esta vazia caso esteja retorno NULL, caso não, percorro a struct musica buscando as musicas e nome das bandas

O(1)

}

Void refatoraPlayList(lista l){

“Refatoro” a playlist colocando as musicas com suas devidas bandas, primeiro a struct pessoa para colocar as bandas e as musicas no local das playlists dessa pessoa, para fazer isso percorro a struct TPlaylist e a struct TMusica com a juda do while. Dentro do while da struct TMusica crio as playlists com os nomes das bandas e as musicas em sua respectivas bandas.

O(n3)

}

Void retiraEspaco(char *vetor){

Retira o espaço deixado pelo vetor através do strlen, quando entra o mesmo coloca o /0 no lugar.

O(1)

}

Int guardaPlayListRefatora(lista l){

Printa a playlist refatorada em um arquivo txt. Primeiramente abro o arquivo(FILE) através da função fopen para escrita, caso não consiga

abrir o arquivo retorna -1.Percorro a struct TPessoa através de um while para poder pintar no arquivo as informações, no caso a pessoa e o tamanho da playlist usando o fprintf. Logo em seguida percorro a struct TPlayList com o while para poder printar no mesmo arquivo como se pede no enunciado o nome das padas também usando fprintf

O(n2)

}

Int calculaSimilaridade(lista l){

Calcula a quantidade de musicas que os amigos tem iguais entre si. Primeiramente abro o arquivo(FILE) através da função fopen para escrita caso não consiga abrir o arquivo retorna -1. Percorre a struct TPessoa,TAmigo através de um while para poder encontrar as pessoas que são amigas, logo após verifica se as playlists e as musicas são iguais contando as mesma e escrevendo no arquivo.

O(n)

}

Int margePlayList(lista l){

Faz a união da musicas e das playlists dos amigos. Primeiramente colocamos uma variável dentro de um do while para ficar percorrendo os whiles das structs TPessoa, TAmigo, TPlayList, TMusica com o intuito de todas as pessoas e seus amigos possam compartilhar as musicas entre si, fazendo o mesmo processo de criarMusica e inserirMusicaFim.

O(n)

}

Int gerarArquivo(lista l){

Gera o arquivo txt com as musicas compartilhadas com os amigos. Primeiramente percorro a struct TPessoa printando o nome da pessoa com a função sprintf no arquivo, também percorrendo a TPlayList e a TMusica para printa - las no arquivo também através da função sprintf no final fecho o arquivo com o flcose.

O(n2)

}

Void desalocaMemoriaALL(lista l){

Desaloca a memoria de todas as structs alocadas dinamicamente começando pelo TPessoa em seguida as demias:TAmigo,TPlayList,TMusica e por fim a struct lista.

O(n2)

}

Conclusões

Conseguir fazer tudo apesar de umas dificuldades no começo mexendo com a leitura do arquivo, percebi uma maior dificuldade na parte do arquivo de fazer as musicas e bandas que os amigos tem iguais. Mas com pesquisas consegui atingir o que esperava.

Referências

C: How to Program: with an introduction to C++ Global Edition 8th Edition: https://faculty.ksu.edu.sa/sites/default/files/c_how_to_program_with_an_introduction_to_c_global_edition_8th_edition.pdf

C plus plus references: https://cplusplus.com/reference/

GitHub: https://github.com