Anotação Trabalho de programação

Dividido em três arquivos .c e dois .h

.c: professor.c, disciplina.c main.c

.h: professor.h, disciplina.h

Funcionalidades do professor:

- Cadastrar, listar, listar em ordem alfabética, pesquisar, alterar, excluir e mostrar disciplinas
- Funções extras: obter professor, existe professor

Funcionalidades da disciplina:

- Cadastrar, listar, listar em ordem alfabética, pesquisar, alterar, excluir e ligar disciplina ao professor
- Funções extras: obter disciplina, existe disciplina

Nos headerers

foram usados typedef structs para facilitar a chamada de um tipo definido. No professor, t_professor é o tipo definido da struct professor. Na disciplina, t_disciplina é o tipo definido da struct disciplina.

Tem as declarações de todas as funções que aparecem no arquivo .c

Funções especiais obter, existe pegam o ponteiro do arquivo e a variável id como parâmetro, usar a função rewind para ir pro início do arquivo pegam a struct professor e com size_t calcula o tamanho do arquivo com o freed e com um if verifica se o id é igual a zero ou se tem algum valor. Posteriormente essas funções são colocadas dentro das outras funções para testas se o id existe, já que este é atribuido no início do cadastro.(como ele pega pelo id e o id é atribuído de forma automática é como uma busca linear de informações)

A função str_somente_numeros verifica se a string guando é pedido o id realmente contém somente numeros

As funções atualizar funcionam de modo semelhante, a diferença é que ao invés do parâmetro id ele pega o vetor do arquivo atualizar;

O main conta com três menus, o menu de iniciação que pergunta se que ir pra aba professor ou disciplina esse menu main leva para o menu1(professor) e menu2(disciplina) e esses por sua vez levam ao menu_professor e menu_disciplina respectivamente que tem apenas o printf para exibir as opções e quando digitada a opção ele volta pro menu que executa a função.

PROFESSOR.C

Cadastrar

- cria o file arq_professor como um ponteiro, abre com fopen e cria o arquivo binário com o “a+b”. assim que entra na função cadastrar ele cria o arquivo
- testa pra ver se o arquivo foi criado, se não foi sai do programa
- variável cont_bytes para contar os ids
- o fseek pega do inicio do arquivo até o final
- cont_bytes pega o tamanho do arquivo
- chama o typedef struct
- se o cont_bytes for igual a 0 coloca o id 1, se não for cria a variável ultimo_professor do (tipo que foi definido(typedef struct) no header) , usa o fseek pra posicionar até o ultimo cont_byte, lê o ultimo professor com o freed e o id do professor é o id do ultimo professor +1
- recebe o input do usuário para cadastrar as informações
- fseek(stdin, 0, SEEK_END); limpa o buffer (pra que nenhum espaço que tenha ficado entre no próximo cadastro por algum ripo de erro)
- fwrite escreve o input no arquivo e o fclose fecha o arquivo
- printa o que foi colocado no arquivo e limpa o buffer de novo

Listar

- rb abre apenas para leitura
- testa se o arquivo foi criado, cria a variável encontrou_professor, chama a struct e while (1)- loop para percorrer o arquivo pois sempre vai ser verdadeiro
- size_t (cria a variável result e mede seu tamanho do tamanho do arquivo, pega o tamanho do arquivo pelo freed)
- se não tiver nada sai do loop, se não sair do loop atribui o 1 ao encontrou professor e printa as informações
- se sair do loop vai pro if que diz que não tem professor cadastrado
- fecha com fclose e limpa o buffer

Pesquisar

- cria a variável nome pra usar como auxiliar
- rb abre pra leitura
- confere que o arquivo existe
- pede o nome, printa o nome
- while (1)- loop para percorrer o arquivo pois sempre vai ser verdadeiro
- size_t (cria a variável result e mede seu tamanho do tamanho do arquivo, pega o tamanho do arquivo pelo freed)
- declara a string nome aux, copia o professor.nome para o nome aux
- compara o nome aux com o nome do inicio do programa como parametro do if
- se forem iguais printa e atribui o 1 ao encontrou professor
- se sair do loop printa que não encontrou professor

Alterar

- crias as variáveis para pegar o id do professor

- abre o arquivo para leitura e atualização
- confere se o arquivo existe
- pede a string id do professor
- limpa o buffer
- como parâmetro do if chama a função que verifica se a string só tem numero
- sscanf bota o valor da string para o int id_professor
- if chama a função existe professor
- sscanf bota o valor da string para o int id_professor
- bota um ponteiro da struct(alocação dinâmica) e obtem o professor pelo id
- se não estiver vazio muda com a função atualizar e libera o professor
- fecha o arquivo limpa buffer

Excluir

- crias as variáveis para pegar o id do professor
- pede a string id do professor
- limpa o buffer
- como parâmetro do if chama a função que verifica se a string só tem numero
- sscanf bota o valor da string para o int id_professor
- abre o arquivo para leitura com o rb
- confere se o arquivo existe
- usa como parametro do if a função existe professor
- se existir cria um arquivo temporário
- testa se o arquivo temporário foi criado
- rewind vai pro inicio do arquivo
- chama a struct
- while (1)- loop para percorrer o arquivo pois sempre vai ser verdadeiro
- size_t (cria a variável result e mede seu tamanho do tamanho do arquivo, pega o tamanho do arquivo pelo freed)
- só escreve no arquivo temporário se os ids forem diferentes(para remover apenas o professor que deseja excluir e manter o outros no arquivo temporário)
- se os ids forem iguais copia o professor.c nome para o nome_professor
- fecha o arquivo professor e o temporário
- usa a função remove para remover o professor.bin, se o arquivo tiver alguma coisa, printa erro
- quando vai pro else renomeia o arquivo temporário para professor.bin
- se a renomeação não for bem sucedida pede para renomear manualmente
- se for bem sucedida printa que o professor foi deletado
- coloca o else dos ifs de cima

listar disciplina do professor(é igual ao listar disciplina literalmente)

obter

- obter professor é um ponteiro
- rewind vai pro inicio do arquivo
- pega o ponteiro da struct professor

- com alocação dinâmica (loop para percorrer o arquivo, busca linear O(n), como o ID é crescente é possível fazer uma busca binária O(log(n)), aloca espaço mesmo sem saber se o professor existe)
- freed retorna o numero de elementos lidos
- if(professor->id...) verifica se são iguais

atualizar

- rewind vai pro inicio do arquivo
- t_professor chama a struct
- freed lê o numero de elementos
- se for 0 sai do loop se ids forem iguais feek posiciona o arquivo e fwrite atualiza

existe

- rewind vai pro inicio do arquivo
- t_professor chama a struct
- freed lê o numero de elementos
- retorna 1 se ids forem iguais

DISCIPLINA.C

Cadastrar

- mesma coisa do professor, diferença é que atribui o valor -1 a disciplina.id_professor porque posteriormente se a disciplina não for ligada vai continuar -1 e imprimir que a disciplina não tem professor

Ligar professor

- crias as variáveis para pegar o id do professor
- abre arq_professor e arq_disciplina para leitura e atualização
- confere se os arquivos existem
- pede a string id do professor
- limpa o buffer
- como parâmetro do if chama a função que verifica se a string só tem numero
- sscanf bota o valor da string para o int id_professor
- if chama a função existe professor
- pede a string id da disciplina
- como parâmetro do if chama a função que verifica se a string só tem numero
- sscanf bota o valor da string para o int id_disciplina
- bota um ponteiro da struct(alocação dinâmica) e obtem o professor pelo id
- se disciplina não estiver vazia e ainda for igual a -1 atualiza o professor da disciplina
- free e setas é por causa da alocação dinâmica

Alterar disciplina

- igual ao alterar professor

Listar

- igual ao listar disciplina com o passo a mais de verificar se ela tem professor
- free e setas é por causa da alocação dinâmica do obter professor

Pesquisar

- igual ao listar disciplina com o passo a mais de verificar se ela tem professor
- free e setas é por causa da alocação dinâmica do obter professor

Excluir

- igual ao listar professor