el trabajo 4

Estruturas

O programa vai conter 3 structs principais

Usuário

A matrícula será a chave de busca nas operações nos arquivos

```
struct usuario
{
    int matricula;
    in rg;
    str cpf;
    str nome;
    str endereço;
    struct tm nascimento //explicado no final
    char sexo;
    str bairro;
    int fone;
    struct tm data_adesão;
    int status; //suspendo ou não
    struct tm termino_suspensao;
    int n_emprestado;
}
```

Livro

O código será a chave de busca nas operações nos arquivos

```
struct livro
{
    int codigo;
    str isbn;
    str titulo;
    str autores;
    str assunto;
    str editora;
    int ano_pubicação;
    int edição;
    int status; //disponivel, emprestado ou em manutenção
}
```

Empréstimo

Relaciona um usuário com um volume emprestado

```
struct emprestimo
{
    int matricula;
    int codigo;
    struct tm data_emprestimo;
    struct tm data_devolução;
}
```

Arquivos

Basicamente cada arquivo armazenará um tipo de struct.As streams dos arquivos serão declarad

Usuários

Armazenará as structs de cada usuário no arquivo *usuarios.dat*.As structs no arquivo devem

Livros

Armazenará as structs de cada livro no arquivo *livros.dat*.As structs no arquivo devem esta

Empréstimos

Armazenará as structs de cada empréstimo feito no arquivo *emprestimos.dat*.As structs no ar

Módulo de usuários:

- **int busca_repetido_usuario(int matricula, int rg, str cpf)**:Vai ser uma busca linear simples no arquivo, se encontrar uma struct que contenha um desses dados iguais retorna 1, caso não encontre retorna 0.
- **void sort_usuario()**: Vai pegar a última struct e trocar com a posição acima até que a posição acima tenha uma matrícula menor.
- **int busca_generica_usuario(int matricula, struct usuario *user)**: Vai procurar o usuario a gravar as informações na struct e retornar 1, se não encontrar retorna 0.
- **int cadastrar_usuario()**: Será lido todos os dados necessários, checado se esses não estão na base de dados(1) e se não a struct será armazenada no arquivo.Após armazenado, o arquivo deve ser ordenado(1) para que a última struct seja inserida no local correto.Se o cadastro for correto retorna 0, se não retorna 1.
  (1)Nesse caso será usada uma função de busca(linear, pois o arquivo vai estar ordenado somente pela matrícula) para checar se a Matrícula ou o RG ou o cpf existem na base de dados.
  (2)A ordenação vai trocando as posições do arquivo até o último alcançar seu local correto, tipo um bubblesort da vida que começa no final do

arquivo.

- **int atualizar_cadastro()**: Será lido uma matrícula, será realizada uma busca no arquivo por ela, e caso seja encontrada novos dados vão ser lidos e escritos sobre a struct anterior.Se a matrícula não existir a função retorna 1, se não retorna 0.
- **int consultar_usuario()**: Será lido uma matrícula, então essa será procurada no arquivo.Caso seja encontrada, a função printará as informações e retornará 0, caso contrário retornará 1.
- **int apagar_usuario()**: Será criado um novo arquivo, lido uma matrícula, e checado se essa matricula tem um livro emprestado(se tiver a função retorna 1),e então copiado todas as structs que não sejam da matrícula lida para o novo arquivo.Caso a cópia para o novo arquivo seja bem sucedida o arquivo antigo será apagado e o novo movido para o nome do antigo, caso contrário a função retornará 1.

Módulo de Obras:

- **int busca_repetido_obra(int codigo)**:Análoga a função busca_repetido_usuario() do módulo de usuários, porém com os dados únicos das obras.
- **void sort_obra()**:Igual a função sort_usuario(), mas com o arquivo de obras.
- **int busca_generica_obra(str titulo, struct livro *obra)**: Vai procurar o livro com o titulo e gravar os dados na struct e retornar 1, se não encontrar retorna zero.
- **int cadastrar_obra()**:Será lido os dados necessários, checado se esses não existem no arquivo de obras(1) e se não a struct será armazenada no arquivo.Depois de armazenado, o arquivo vai ser ordenado pelo título(2). (1)igual a função busca_repetido_usuario() do módulo de usuários. (2)igual a função sort_usuario_usuario() do módulo de usuários.
- **int alterar_status()**: Será lido um titulo, então esse será procurada no arquivo.Se for encontrado, será mudado o campo de DISP para MANUT e vice versa.Se o título não existir retorna 1, caso tudo seja feito corretamente retorna 1.
- **int consultar_obra()**:Análoga a função consultar_usuario() do módulo de usuários.
- **int apagar_obra()**: Análogo a função apagar_usuario() do módulo de usuários.Porém será lido um título e checado se ele existe e se não está emprestado.

Módulo de Emprestimos:

- **void sort_emprestimo()**:Análoga as funções de ordenação dos outros módulos, porém vai ordenar o arquivo de empréstimo por títulos.
- **void mudar_status_usuario(int matricula, int tipo, struct tm dia)**: Vai procurar a struct do usuário com esta matricula e mudar seu status.Se tipo for = 1 a função vai suspender o usúario e mudar sua data de suspenção para 30 dias após 'dia', se não vai só mudar o status.
- **void mudar_status_livro(int codigo)**: Vai procurar a struct do livro com esse codigo e mudar seu status.
- **struct tm ultimo_emprestimo(struct usuario *u)**: Vai procurar no arquivo de emprestimos os n livros emprestados pelo usuário e retornar a data de devolução do mais recente
- **int emprestar()**: Vai ler a matricula do usuario e buscar com a função busca_generica_usuario() e armazenar em uma struct, também vai chamar a função ultimo_emprestimo() e armazenar.Se o usuário estiver suspenso e o termino da suspensão não tiver chegado, a função printa o resultado e retorna 1.Se o usuario estiver suspenso e a data de suspensão tiver passado, chamar a função mudar_status_usuario().Se o usuario não estiver suspenso checar a data do ultimo emprestimo com a data atual, e se tiver um atraso suspendê-lo e retornar 1.Também checar numero máximo de empréstimos, se for 4 retornar 1. Se o usuário não estiver suspenso, chamar a função busca_generica_obra() e checar se o livro existe e está disponível, se não retorna 1.Após isso inserir uma struct de emprestimo no arquivo, mudar a situação do livro para emprestado, incrementar o numero de volumes do usuário.
- **int devolver()**: Vai ler o código do livro e verificar se ele existe, se não existir retornar 1.Procurar no arquivo de empréstimos a struct do empréstimo e atualizar a situação do livro e o número de livros que o usuário tem.Verificar a data de devolução, se for depois da data máxima suspender o usuário.Após isso copiar todas as structs de emprestimos que não seja a devolvida para um novo arquivo e renomeá-lo.
- **int relatorio()**:Vai printar todas as structs no arquivo(vão estar ordenadas por título).