



**UNIVERSIDADE ESTADUAL PAULISTA
“JÚLIO DE MESQUITA FILHO”**

Bruno Vedoveto Leandro
Giovanna Cazolato Pires
Wesley Otto Garcia Utsunomiya
Jonathan Henrique de Oliveira

Trabalho Prático I

Professora Vânia

Rio claro
2016

Sumário

Introdução.....	2
Seção 1.....	3
Seção 2.....	4
Procedimento Experimental.....	4
Código Fonte do Programa.....	10
Conclusão.....	15

Introdução

O trabalho propôs ao grupo implementar um programa capaz de armazenar em um arquivo dados sobre determinados posts, com a possibilidade de removê-los, adicionar novos dados, buscá-los utilizando uma chave primária ou uma chave secundária.

O objetivo do trabalho é pôr em prática tudo o que foi visto em sala de aula na disciplina de Estrutura de Dados 2, utilizando conceitos de armazenamento, buscas, organização, compactação e compressão do arquivo.

Seção 1

Para facilitar o uso da nossa estrutura, usamos *typedef* para declará-la:

```
typedef struct post{  
    ...  
} post;
```

Logo, sempre que queremos usá-la chamamos a estrutura com o tipo **post**.

Considerando os seguintes dados a respeito de um post : TEXT, USER, COORDINATES, LIKE_COUNT, LANGUAGE, SHARE_COUNT, VIEWS_COUNT. Definimos o corpo de nossas estruturas com um **text[150]**, que significa o texto do post, com no máximo 150 caracteres, para não ficar com um texto muito longo no formato **char**. **User[50]**, que contém o nome do usuário que postou o post, com no máximo 50 caracteres, que achamos um bom tamanho para poder dar um nome completo no formato **char**. Um **coordinates[20]**, que seria de onde os usuário fez o post, tipo **char**. Um contador de curtidas do post **like_count**, definido com o tipo **int**. O idioma do post, declarado no formato **char language[30]**, o tamanho do language está de bom tamanho para as linguagens mais comumente usadas nos posts. Temos também um contador de compartilhadas e visualizações do post declarados ambos com tipo **int** com o nome de **share_count** e **views_count** respectivamente.

Com este corpo de estrutura, acabaríamos com uma estrutura definida de acordo com a tabela 1 abaixo:

Com este corpo de estrutura, acabaríamos com uma estrutura definida de acordo com a tabela 1 abaixo:

Tipo	Char	Char	Char	Int	Char	Int	int
Nome	TEXT	USER	COORDINATES	LIKE_COUNT	LANGUAGE	SHARE_COUNT	VIEWS_COUNT
Tam.	150	50	20	-	30	-	-

Tabela 1 – Estrutura de um post.

Seção 2

Utilizamos uma lista dispo no início do arquivo para guardar os registros disponíveis. Caso tenha mais de um registro disponível, colocamos a lista dispo no *like_count* do registro e atualizamos a lista dispo.

Definimos que, para identificar um registro removido utilizaremos um “ * “ no lugar do nome do usuário(*user[50]*).

Achamos que seria melhor para o usuário se a cada inserção de um novo post fosse dado uma opção de continuar ou não, deixaria de um modo mais claro que ele está inserindo um novo post.

Procedimento Experimental

Utilizando o CodeBlocks em sua versão 13.12 no Windows 8.1, rodamos alguns testes:

- Inserindo dados do post:

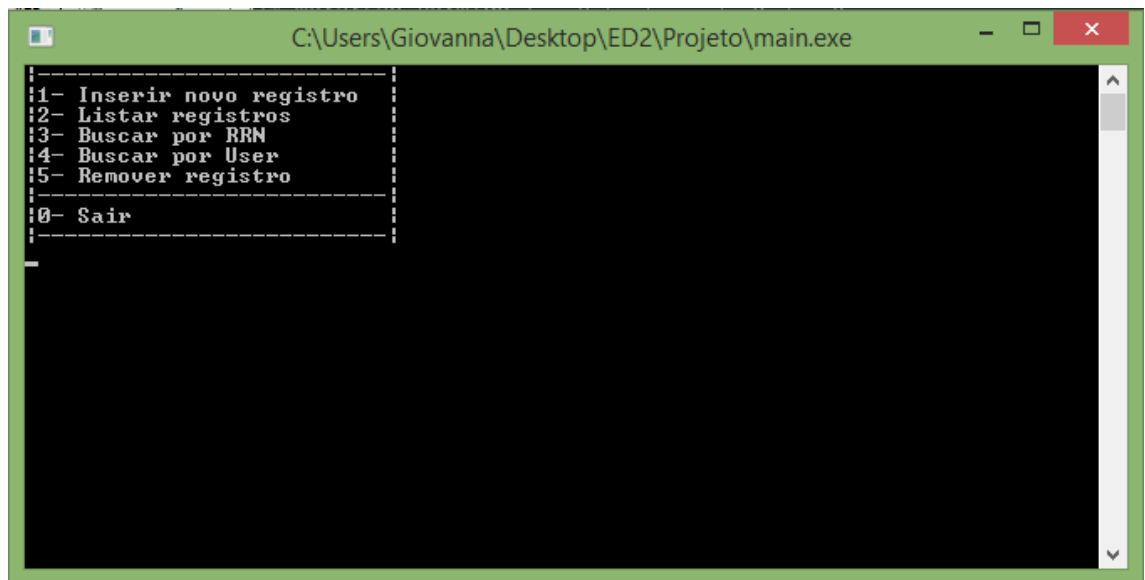


Figura 1 – Tela do menu de opções.

Teclando 1 temos:

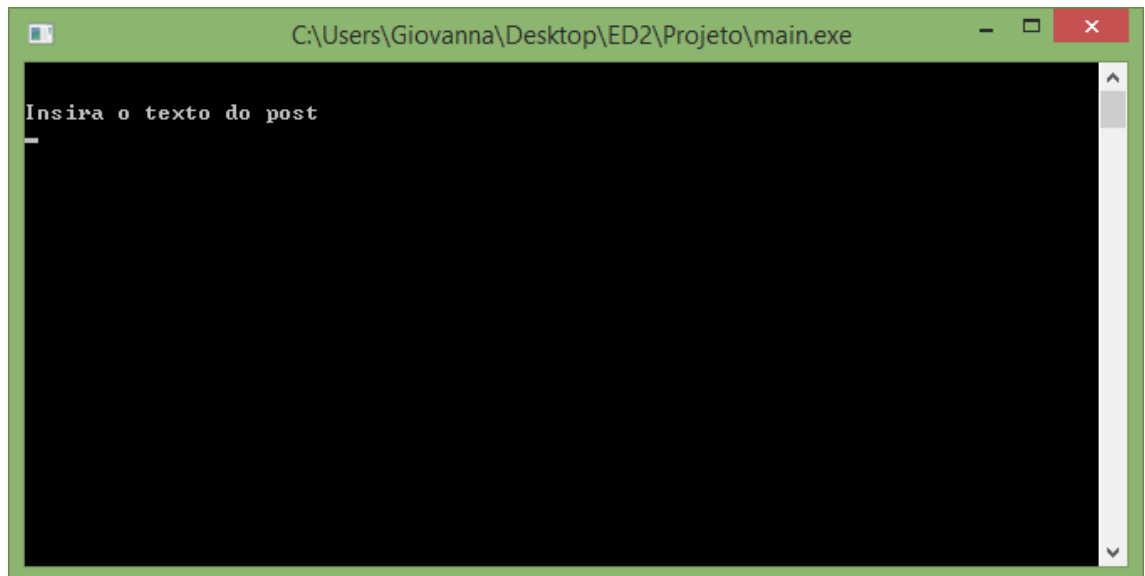


Figura 2 – Inserindo texto no post.

Após inserir o texto (Teste 1), nome do usuário (Teste 1), as coordenadas do post (Teste 1), a contagem de likes (1), a linguagem do post (Teste), o número de vezes que o post foi compartilhado (1) e o número de visualizações do post (1), aparece a opção de continuar inserindo novos posts ou não:

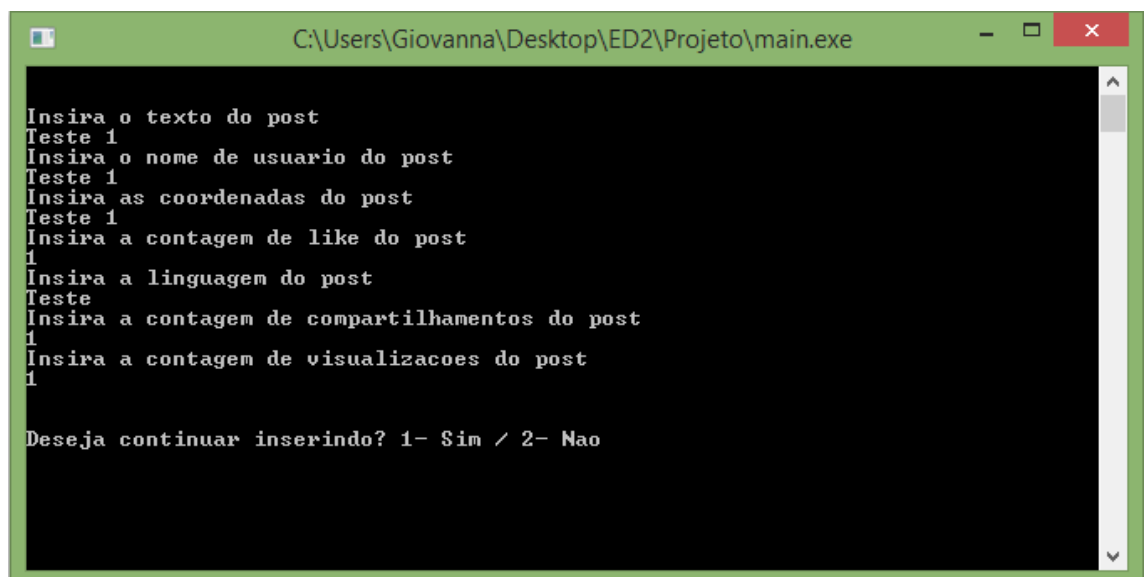


Figura 3 – Opção de continuar inserindo um post.

Para fins de teste, adicionamos mais 2 registros iguais o primeiro, mudando apenas o número do seu teste. Ex. Nome do usuário: Teste 2

Listamos todos os posts do arquivo teclando o número 2 no menu de opções (Figura 1):



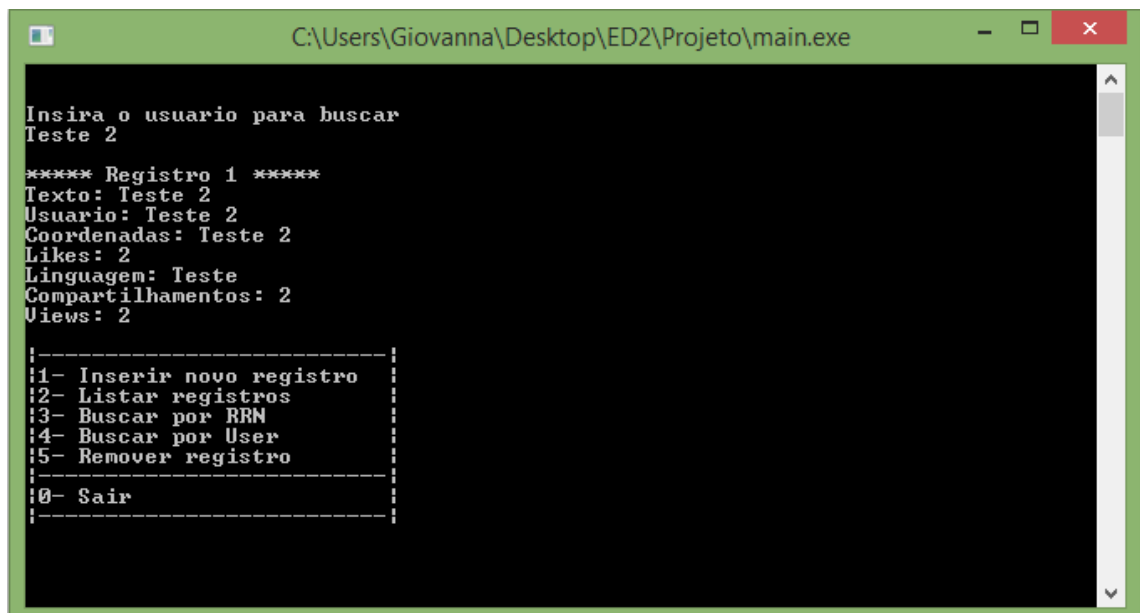
```
C:\Users\Giovanna\Desktop\ED2\Projeto\main.exe
Usuario: Teste 1
Coordenadas: Teste 1
Likes: 1
Linguagem: Teste
Compartilhamentos: 1
Views: 1

***** Registro 1 *****
Texto: Teste 2
Usuario: Teste 2
Coordenadas: Teste 2
Likes: 2
Linguagem: Teste
Compartilhamentos: 2
Views: 2

***** Registro 2 *****
Texto: Teste 3
Usuario: Teste 3
Coordenadas: Teste 3
Likes: 3
Linguagem: Teste
Compartilhamentos: 3
Views: 3
```

Figura 4 – Lista de todos os posts.

Utilizamos agora a opção de mostrar um registro utilizando o nome do usuário como critério de busca teclando a opção 4 no menu de opções (Figura 1). Para realizar o teste escolhemos o usuário Teste 2:



```
C:\Users\Giovanna\Desktop\ED2\Projeto\main.exe
Insira o usuario para buscar
Teste 2

***** Registro 1 *****
Texto: Teste 2
Usuario: Teste 2
Coordenadas: Teste 2
Likes: 2
Linguagem: Teste
Compartilhamentos: 2
Views: 2

|-----|
|1- Inserir novo registro|
|2- Listar registros    |
|3- Buscar por RRN     |
|4- Buscar por User    |
|5- Remover registro   |
|-----|
|0- Sair               |
|-----|
```

Figura 5 – Busca de post por nome do usuário.

Para realizar a busca, dado o RRN do post, que no caso é o registro do post, teclamos a opção 3 do menu de opções (Figura 1). A fins de teste buscamos o post que tem o RRN igual a 0:



```
C:\Users\Giovanna\Desktop\ED2\Projeto\main.exe

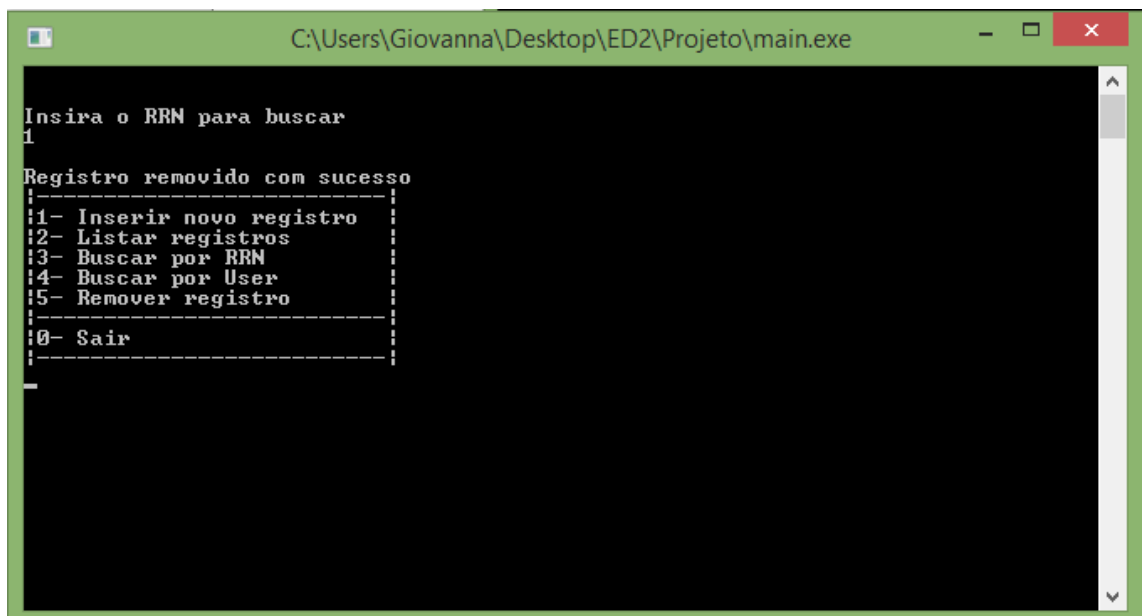
Insira o RRN para buscar
0

***** Registro 0 *****
Texto: Teste 1
Usuario: Teste 1
Coordenadas: Teste 1
Likes: 1
Linguagem: Teste
Compartilhamentos: 1
Views: 1

-----
1- Inserir novo registro
2- Listar registros
3- Buscar por RRN
4- Buscar por User
5- Remover registro
-----
0- Sair
-----
```

Figura 6 – Buscando o post dado o RRN.

Para realizar a remoção de um post, teclamos o número 5 no menu de opções (Figura 1). Escolhemos testar remover o post com RRN igual a 1:



```
C:\Users\Giovanna\Desktop\ED2\Projeto\main.exe

Insira o RRN para buscar
1

Registro removido com sucesso

-----
1- Inserir novo registro
2- Listar registros
3- Buscar por RRN
4- Buscar por User
5- Remover registro
-----
0- Sair
-----
```

Figura 7 – Removendo um post dado o seu RRN.

Após remover o post de RRN igual a 1, listamos novamente os registros na tela e obtemos a lista de posts atualizada:



```
C:\Users\Giovanna\Desktop\ED2\Projeto\main.exe

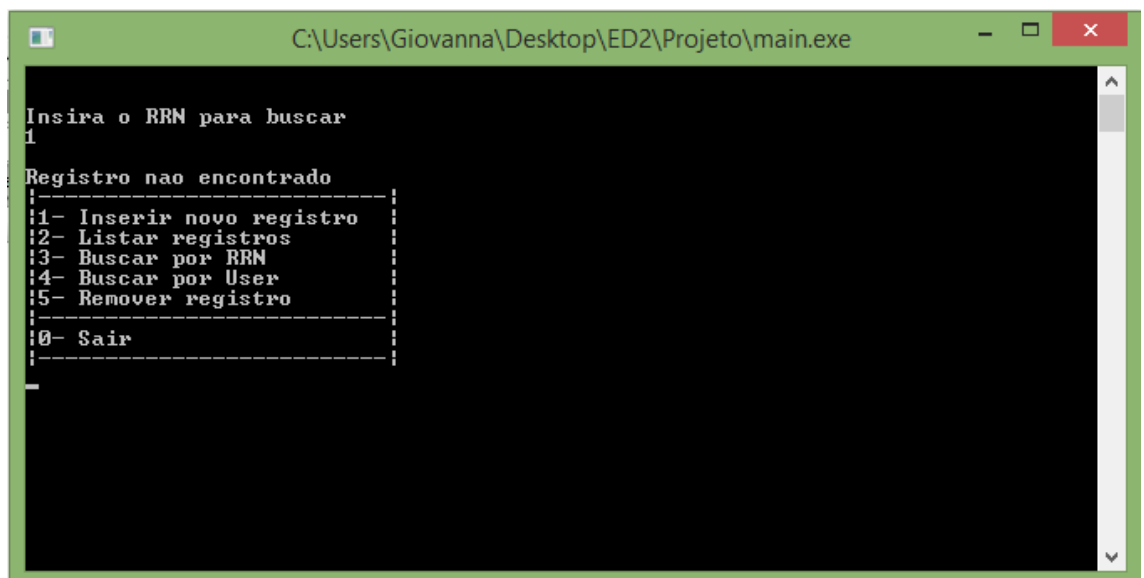
***** Registro 0 *****
Texto: Teste 1
Usuario: Teste 1
Coordenadas: Teste 1
Likes: 1
Linguagem: Teste
Compartilhamentos: 1
Views: 1

***** Registro 2 *****
Texto: Teste 3
Usuario: Teste 3
Coordenadas: Teste 3
Likes: 3
Linguagem: Teste
Compartilhamentos: 3
Views: 3

-----
1- Inserir novo registro
2- Listar registros
3- Buscar por RRN
4- Buscar por User
5- Remover registro
-----
```

Figura 8 – Lista atualizada de posts. (Após a remoção do RRN igual a 1)

Para testar, tentamos buscar o post que foi removido, e vem a mensagem de “Registro não encontrado”:



```
C:\Users\Giovanna\Desktop\ED2\Projeto\main.exe

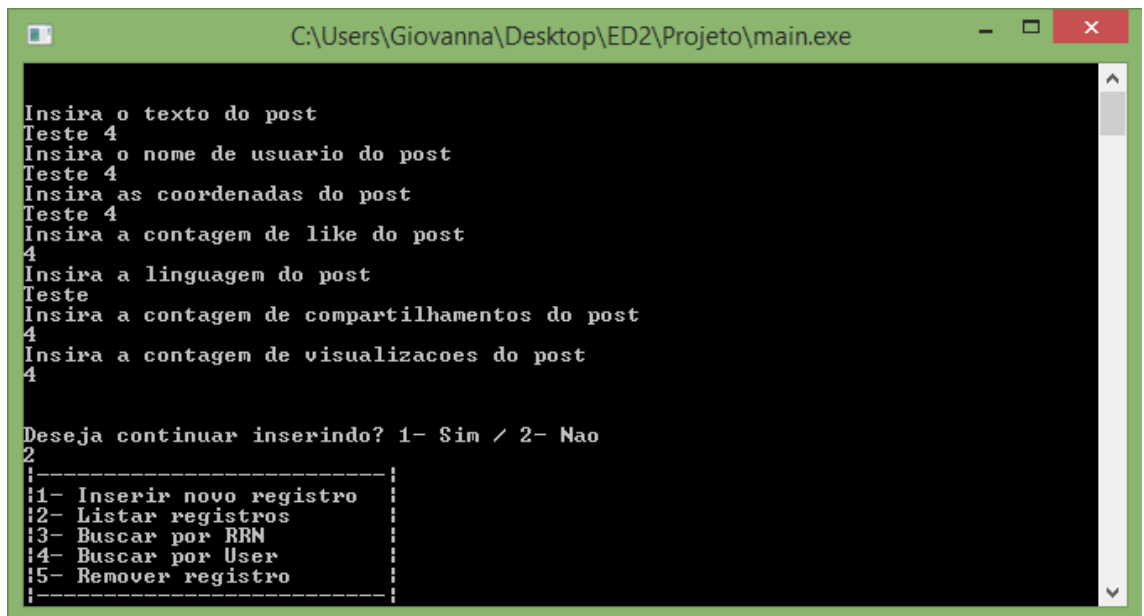
Insira o RRN para buscar
1

Registro nao encontrado

-----
1- Inserir novo registro
2- Listar registros
3- Buscar por RRN
4- Buscar por User
5- Remover registro
0- Sair
-----
```

Figura 9 – Busca de post não existente.

Após remover o post, gostaríamos de inserir um novo para testar, utilizamos um post com os seguintes dados:



```
C:\Users\Giovanna\Desktop\ED2\Projeto\main.exe

Insira o texto do post
Teste 4
Insira o nome de usuario do post
Teste 4
Insira as coordenadas do post
Teste 4
Insira a contagem de like do post
4
Insira a linguagem do post
Teste
Insira a contagem de compartilhamentos do post
4
Insira a contagem de visualizacoes do post
4

Deseja continuar inserindo? 1- Sim / 2- Nao
2

-----|
:1- Inserir novo registro |
:2- Listar registros      |
:3- Buscar por RRN       |
:4- Buscar por User      |
:5- Remover registro     |
:                         |
-----|
```

Figura 10 – Inserindo um novo post.

Ao listarmos novamente todos os posts, o novo post fica registrado no lugar do que removemos. Com seu RRN igual a 1:



```
C:\Users\Giovanna\Desktop\ED2\Projeto\main.exe

***** Registro 0 *****
Texto: Teste 1
Usuario: Teste 1
Coordenadas: Teste 1
Likes: 1
Linguagem: Teste
Compartilhamentos: 1
Views: 1

***** Registro 1 *****
Texto: Teste 4
Usuario: Teste 4
Coordenadas: Teste 4
Likes: 4
Linguagem: Teste
Compartilhamentos: 4
Views: 4

***** Registro 2 *****
Texto: Teste 3
Usuario: Teste 3
Coordenadas: Teste 3
Likes: 3
Linguagem: Teste
Compartilhamentos: 3
Views: 3
```

Figura 11 – Lista do post atualizada. (Após a inserção de um novo post)

Código Fonte do Programa

main.c

```
#include "stdio.h"
#include "myFile.h"
#include "stdlib.h"

//Função para printar interface
void printInterface(){
    printf(" |-----|\n");
    printf(" |1- Inserir novo registro  |\n");
    printf(" |2- Listar registros        |\n");
    printf(" |3- Buscar por RRN          |\n");
    printf(" |4- Buscar por User         |\n");
    printf(" |5- Remover registro        |\n");
    printf(" |-----|\n");
    printf(" |0- Sair                    |\n");
    printf(" |-----|\n");
}

int main(){
    int op = 0;

    FILE* file = fopen("file.dat", "r+");

    if(!file)
        file = fopen("file.dat", "w+");

    fread(&dispo, sizeof(dispo), 1, 0);

    while(1){
        printInterface();
        scanf("%d", &op);

        //Lê opção do usuário
        switch(op){
            case 1: system("cls");
                    inserirVarios(file);
                    break;

            case 2: system("cls");
                    listar(file);
                    break;

            case 3: system("cls");
                    buscarRRN(file);
                    break;

            case 4: system("cls");
                    buscarUser(file);
                    break;

            case 5: system("cls");
                    remover(file);
                    break;

            case 0: exit(0);

            default: printf("Opcao invalida!\n");
        }
    }
}
```

```

    }

    getch();
    close(file);
}

```

myFile.h

```

#include "string.h"

//Registro que guarda informações do post
typedef struct post{
    char user[50]; //Preenchido com * quando registro for excluído
    char text[150];
    char coordinates[20];
    int like_count; //Utilizado para manter a dispo se o registro
estiver excluído
    char language[30];
    int share_count;
    int views_count;
} post;

int dispo = -1;

//Utiliza função para inserir
//Pergunta ao usuário se deseja continuar inserindo após cada iteração
// 2) Não há uma condição de parada de fato, mas ainda sim há uma
opção de parar de adicionar arquivos.
// Desse modo o programa fica mais fácil para o usuário.
void inserirVarios(FILE*file){
    int op = 0;
    fseek(file, 0, 0);

    fread(&dispo, sizeof(dispo), 1, file);
    if(dispo == -1)
        atualizaDispo(file, dispo);

    do{
        inserir(file);

        fflush(stdin);
        printf("\n\nDeseja continuar inserindo? 1- Sim / 2-
Nao\n");
        scanf("%d", &op);
    }while(op != 2);
}

//Função responsável por inserir os dados de post no arquivo
// 7) Insere os registros em espaço vazio se houver.
void inserir(FILE* file){
    post lpost;
    printf("\n\n");

    //Realiza leitura de dados do post a partir do teclado
    fflush(stdin);
    printf("Insira o texto do post\n");
    gets(lpost.text);

    fflush(stdin);
    printf("Insira o nome de usuario do post\n");
    gets(lpost.user);

    fflush(stdin);

```

```

printf("Insira as coordenadas do post\n");
gets(lpost.coordinates);

fflush(stdin);
printf("Insira a contagem de like do post\n");
scanf("%d", &lpost.like_count);

fflush(stdin);
printf("Insira a linguagem do post\n");
gets(lpost.language);

fflush(stdin);
printf("Insira a contagem de compartilhamentos do post\n");
scanf("%d", &lpost.share_count);

fflush(stdin);
printf("Insira a contagem de visualizacoes do post\n");
scanf("%d", &lpost.views_count);

//Se existe valor excluído utiliza espaço para gravar, senão vai
para fim de arquivo
if(dispo != -1){
    int temp = dispo;
    post auxPost;
    fread(&auxPost, sizeof(post), 1, file);
    atualizaDispo(file, auxPost.like_count);
    fseek(file, posicaoRRN(temp), 0);
}
else{
    fseek(file, 0, 2);
}

//Escreve registro no arquivo
fwrite(&lpost, sizeof(post), 1, file);
}
// 3)Lê todos os registros no arquivo. E mostra na tela.
void listar(FILE* file){
    post lpost;
    fseek(file, posicaoRRN(0), 0);

    //Variável para contagem do RRN
    int RRN = 0;

    //Enquanto houver dados no arquivo, ler para o registro
    while(fread(&lpost, sizeof(post), 1, file) > 0){
        //Imprime registro na tela
        if(strcmp(lpost.user, "") != 0){
            printPost(lpost, RRN);
        }/*else{
            printf("\n----- Registro %d deletado -----
\n\n",RRN);
        }*/

        RRN++;
    }
}
// 5)Permite a busca de posts que contem o campo user.
void buscarUser(FILE* file){
    post lpost;
    char find[50];

    fseek(file, posicaoRRN(0) , 0);

```

```

//Realiza leitura de qual usuario buscar a partir do teclado
fflush(stdin);
printf("\n\nInsira o usuario para buscar\n");
gets(&find);

//Variável lógica para indicar se encontrou o usuário buscado
int achou = 0;

//Variável para contagem do RRN
int RRN = 0;

//Laço nos dados do arquivo
while(fread(&lpost, sizeof(post), 1, file) > 0){

    //Se o usuário que está sendo lido é igual ao informado,
    muda valor da variável para 1 e sai do laço
    if (strcmp(lpost.user, find) == 0){
        achou = 1;
        break;
    }

    RRN++;
}

//Se a busca encontrou usuário, imprime. Senão, mostra mensagem de
que não encontrou
if(achou)
    printPost(lpost, RRN);
else
    printf("\nRegistro nao encontrado\n");
}

// 4) Permite a busca de um post dado o RRN
void buscarRRN(FILE* file){
    post lpost;

    //Variável para ler o RRN a partir do teclado
    int find=-1;

    fseek(file, posicaoRRN(0) , 0); // posiciona no começo do arquivo
    novamente

    //Realiza leitura do RNN para buscar a partir do teclado
    printf("\n\nInsira o RRN para buscar\n");
    scanf("%d", &find);

    //Usa o RRN para dar um seek no arquivo e localizar o registro em
    questão
    fseek(file, posicaoRRN(find), 0);

    //Se a busca encontrou usuário, imprime. Senão, mostra mensagem
    de que não encontrou
    if(fread(&lpost, sizeof(post), 1, file) > 0)
        printPost(lpost, find);
    else
        printf("\nRegistro nao encontrado\n");
}

// 6) Permite a remoção do registro dado o RRN
void remover(FILE* file){
    post lpost;

    //Variável para ler o RRN a partir do teclado
    int find = -1;

```

```

//Realiza leitura do RNN para buscar a partir do teclado
printf("\n\nInsira o RNN para buscar\n");
scanf("%d", &find);

//Usa o RNN para dar um seek no arquivo e localizar o registro em
questão
fseek(file, posicaoRRN(find), 0);

//Se a busca encontrou usuário, remove. Senão, mostra mensagem de
que não encontrou
if(fread(&lpost, sizeof(post), 1, file) > 0){
    //Marca usuário com simbolo especial para informar remoção
    strcpy(lpost.user, "*");

    //Armazena valor atual da dispo no registro e atualiza topo
da dispo
    lpost.like_count = dispo;
    atualizaDispo(file, find);

    fseek(file, posicaoRRN(find), 0);

    //Escreve registro no arquivo
    fwrite(&lpost, sizeof(post), 1, file);

    printf("\nRegistro removido com sucesso\n");
}
else
    printf("\nRegistro nao encontrado\n");
}

//atualiza dispo no arquivo e na variavel, ATENCAO: posiciona ponteiro
no inicio do arquivo
void atualizaDispo(FILE *file, int rrn){
    dispo = rrn;
    fseek(file, 0, 0);
    fwrite(&rrn, sizeof(int), 1, file);
}

// posicao considerando a dispo no inicio do arquivo
int posicaoRRN(int rrn){
    return ((rrn * sizeof(post)) + sizeof(dispo));
}

//Função para mostrar o post na tela.
void printPost(post lpost, int RRN){
    if(strcmp(lpost.user, "*") != 0){ //Se não está removido
        printf("\n***** Registro %d *****\n", RRN);
        printf("Texto: %s\n", lpost.text);
        printf("Usuario: %s\n", lpost.user);
        printf("Coordenadas: %s\n", lpost.coordinates);
        printf("Likes: %d\n", lpost.like_count);
        printf("Linguagem: %s\n", lpost.language);
        printf("Compartilhamentos: %d\n", lpost.share_count);
        printf("Views: %d\n", lpost.views_count);
        printf("\n");
    }else{
        printf("\nRegistro nao encontrado\n");
    }
}

```

Conclusão

O objetivo do trabalho era pôr em prática todos os conceitos dados na aula de Estrutura de Dados 2 até agora. Aparentemente, de acordo com os testes realizados, não foi encontrado nenhum erro, logo o objetivo foi alcançado com êxito. O trabalho em si não gerou dúvidas, apenas erros ligados a lógica que logo foram solucionados.

Não houve reunião com o grupo, nosso método de comunicação e troca de arquivos foi feito a partir de um repositório no GitHub (um site de versionamentos de código - <https://github.com>).