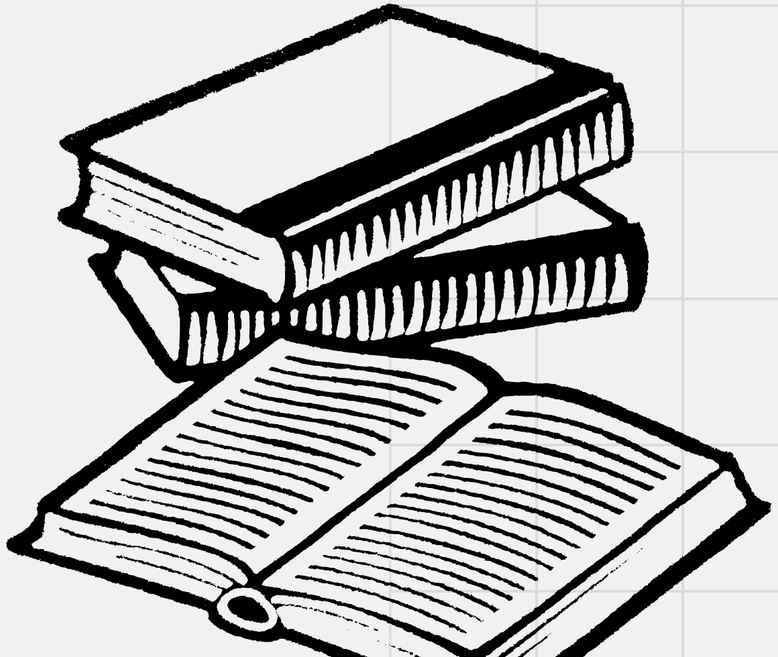
DOCENTE

DR. Carlos Benedito Barreiros Gutierrez

DATA

12/06

SISTEMA DE GERENCIAMENTO DE LIVROS COM ÁRVORE



DISCENTES:

Diego Gabriel Pessoa Amorim

Ryan Ricardo de Souza

Athus Gabriel Guarany Soares

Marcello Costa dos Santos

SUMÁRIO

3 ESTRUTURA DO LIVRO

4 ESTRUTURA DA ÁRVORE BINÁRIA

5 FUNÇÃO PARA CRIAR UM NOVO NÓ

6 FUNÇÃO PARA INSERIR UM LIVRO

7 FUNÇÃO PARA BUSCAR UM LIVRO POR ID

8 FUNÇÃO PARA LISTAR OS LIVROS EM ORDEM

9 FUNÇÃO PARA REMOVER LIVROS

10 FUNÇÃO PRINCIPAL

```c
typedef struct {
    int id;
    char titulo[100];
    char autor[100];
} Livro;
```

ESTRUTURA DO LIVRO

A estrutura do Livro é utilizada para armazenar as informações de cada livro. Cada livro possui um identificador único (`id`), um título (`titulo`) e um autor (`autor`). Essa estrutura serve como a unidade básica de dados que será armazenada na árvore binária de busca.

```c
typedef struct No {
    Livro livro;
    struct No* esquerda;
    struct No* direita;
} NoArvore;
```

ESTRUTURA DA ARVORE BINÁRIA

A estrutura “NoArvore” define os nós da árvore binária de busca. Cada nó contém um livro e dois ponteiros (`esquerda` e `direita`) que apontam para os filhos esquerdo e direito, respectivamente. Essa estrutura permite a organização hierárquica dos livros com base em seus `id`s.

```
NoArvore* criarNo(Livro livro) {
    NoArvore* novoNo =
(NoArvore*)malloc(sizeof(NoArvore));
    novoNo->livro = livro;
    novoNo->esquerda = NULL;
    novoNo->direita = NULL;
    return novoNo;
}
```

FUNÇÃO PARA CRIAR UM NOVO NÓ

A função “criarNo” é responsável por alocar memória para um novo nó na árvore e inicializar seus valores. Ela recebe um livro como parâmetro e retorna um ponteiro para o novo nó criado, com os filhos inicializados como `NULL`.

```
NoArvore* inserirLivro(NoArvore* raiz, Livro
livro) {
    if (raiz == NULL) {
        return criarNo(livro);
    }
    if (livro.id < raiz->livro.id) {
        raiz->esquerda = inserirLivro(raiz-
>esquerda, livro);
    } else if (livro.id > raiz->livro.id) {
        raiz->direita = inserirLivro(raiz-
>direita, livro);
    }
    return raiz;
}
```

FUNÇÃO PARA INSERIR UM LIVRO

A função `inserirLivro` insere um novo livro na árvore binária de busca. Ela compara o `id` do livro a ser inserido com o `id` dos nós existentes para encontrar a posição correta e insere o livro, mantendo a propriedade de ordenação da árvore.

```
NoArvore* buscarLivro(NoArvore* raiz, int
id) {
    if (raiz == NULL || raiz->livro.id ==
id) {
        return raiz;
    }
    if (id < raiz->livro.id) {
        return buscarLivro(raiz->esquerda,
id);
    }
    return buscarLivro(raiz->direita, id);
}
```

FUNÇÃO PARA BUSCAR UM LIVRO POR ID

A função `buscarLivro` busca um livro na árvore binária de busca com base no `id` fornecido. Ela navega pela árvore comparando o `id` alvo com os `id`s dos nós, e retorna o nó que contém o livro desejado ou `NULL` se o livro não for encontrado.

```
void listarLivros(NoArvore* raiz) {
    if (raiz != NULL) {
        listarLivros(raiz->esquerda);
        printf("ID: %d, Título: %s, Autor:
%s\n", raiz->livro.id, raiz->livro.titulo,
raiz->livro.autor);
        listarLivros(raiz->direita);
    }
}
```

FUNÇÃO PARA LISTAR OS LIVROS EM ORDEM CRESCENTE

A função `listarLivros` imprime os livros armazenados na árvore binária de busca em ordem crescente de `id`. Ela realiza uma travessia in-order, visitando recursivamente a subárvore esquerda, o nó atual, e a subárvore direita.

```
NoArvore* removerLivro(NoArvore* raiz, int
id) {
    if (raiz == NULL) {
        return raiz;
    }

    if (id < raiz->livro.id) {
        raiz->esquerda = removerLivro(raiz-
>esquerda, id);
    } else if (id > raiz->livro.id) {
        raiz->direita = removerLivro(raiz-
>direita, id);
    } else {
        if (raiz->esquerda == NULL) {
            NoArvore* temp = raiz->direita;
            free(raiz);
            return temp;
        } else if (raiz->direita == NULL) {
            NoArvore* temp = raiz->esquerda;
            free(raiz);
            return temp;
        }

        NoArvore* sucessor =
encontrarSucessor(raiz);
        raiz->livro = sucessor->livro;
        raiz->direita = removerLivro(raiz-
>direita, sucessor->livro.id);
    }
    return raiz;
}
```

FUNÇÃO PARA REMOVER LIVROS

A função “removerLivro” remove um livro da árvore binária de busca com base no `id` fornecido. Ela lida com três casos de remoção: nó com zero ou um filho, e nó com dois filhos, utilizando o sucessor para manter a integridade da árvore.

```
    char linha[100];

    while (1) {
        printf("\nMenu de Opções:\n");
        printf("1. Inserir Livro\n");
        printf("2. Buscar Livro\n");
        printf("3. Remover Livro\n");
        printf("4. Listar Livros\n");
        printf("5. Verificar se a árvore
está Vazia\n");
        printf("6. Sair\n");
        printf("Digite a opções desejada:
");

        fgets(linha, sizeof(linha), stdin);
        sscanf(linha, "%d", &opcao);

        switch (opcao) {
            case 1:
                printf("\nInsira as
informações do livro:\n");
                printf("ID: ");
                fgets(linha, sizeof(linha),
stdin);
                sscanf(linha, "%d",
&novoLivro.id);
                printf("Título: ");
                fgets(linha, sizeof(linha),
stdin);
                strcpy(novoLivro.titulo,
linha);
```

FUNÇÃO PRINCIPAL

A função “main” gerencia o menu de opções que permite ao usuário interagir com a árvore binária de busca. Ela oferece opções para inserir, buscar, remover, listar livros, verificar se a árvore está vazia e sair do programa. A função `main` utiliza um loop `while` infinito que continua até o usuário escolher sair, processando a entrada do usuário e chamando as funções apropriadas com base na escolha.

AGRADECIMENTOS

Gostaríamos de expressar nossa sincera gratidão a todos que nos acompanharam durante esta apresentação. Agradecemos imensamente pela atenção e pelo tempo dedicado a entender o nosso trabalho sobre a implementação de uma árvore binária de busca em C para a gestão de livros.