

JOÃO PEDRO CORREIA

DOCUMENTAÇÃO

OBJETIVO

O objetivo do algoritmo é ler um arquivo onde tem a matrícula e o nome do estudante, e armazená-los em uma árvore binária. Também deve proporcionar para o usuário operações para manipular a árvore, podendo inserir, remover, printar, buscar e salvar a base de dados.

Para realizar a atividade em questão eu fiz apenas uma biblioteca, ela é responsável pelas operações que são feitas na árvore. Os arquivos usados são `arvore.c`, `arvore.h` e `main.c`, nos arquivos 'arvore' é onde se encontra o código fonte de manipulação da árvore binária. No 'main.c' é o código principal, nele é chamado a biblioteca 'arvore.h', faz as leituras e gravações no arquivo.

ESTRUTURAS USADAS

```
arvore.c  buffers
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4
5 // ESTRUTURA DA ARVORE BINARIA
6 struct ArvoreBinaria {
7     char *matricula;
8     char *nome_aluno;
9     struct ArvoreBinaria *esq;
10    struct ArvoreBinaria *dir;
11 };
12
```

Na estrutura da árvore binária tem 4 ponteiros, dois para char, onde vão ser armazenados o nome do aluno e sua respectiva matrícula, os outros dois são ponteiros para nós filhos.

```
arvore.c  main.c  arvore.h
1 #include "arvore.c"
2
3 typedef struct ArvoreBinaria Abb;
4
5 /*INSERE UM NOH NA ARVORE BINARIA*/
6 struct ArvoreBinaria *inserirNo(struct ArvoreBinaria **nodo, char *matricula,
7                                 char *nome);
8
9 /*REMOVE UM NOH DA ARVORE*/
10 struct ArvoreBinaria *removeNo(struct ArvoreBinaria **nodo, char *matricula);
11
12 /*PROCURA UM NO DA ARVORE*/
13 struct ArvoreBinaria *procurarNo(struct ArvoreBinaria *nodo, char *matricula);
14
15 /*VERIFICA SE O NODO ESTA VAZIO*/
16 int vazio(struct ArvoreBinaria *nodo);
17
18 /*RETORNA A MATRICULA NO NODO*/
19 char *getMatricula(struct ArvoreBinaria *nodo);
20
21 /*RETORNA O NOME NO NODO*/
22 char *getNome(struct ArvoreBinaria *nodo);
23
24 /*RETORNA O NODO DA ESQUERDA*/
25 Abb *noEsq(struct ArvoreBinaria *node);
26
27 /*RETORNA O NODO DA DIREITA*/
28 Abb *noDir(struct ArvoreBinaria *node);
29
30 /*LIMPA TODA A ARVORE*/
31 void freeArvore(struct ArvoreBinaria *arvore);
32
33 /*INICIALIZA*/
34 struct ArvoreBinaria *inicializa();
```

Essas são as funções responsáveis por manipular a árvore como um todo ou algum nó em específico. Todas estão presentes no arquivo 'main.c'. Seus códigos fonte são encontrados dentro do arquivo 'arvore.c'.

ESTRATÉGIA

```
arvore.c [] main.c [] arvore.h []
1 #include "arvore.h"
2 #include <stdio.h>
3 #include <stdlib.h>
4
5 /*CONSTANTES DE TAMANHO
6  * DOS ARRAY DE CHAR DOS NOMES E MATRICULA*/
7 #define NOME 60
8 #define MATRICULA 15
9
10 /*EXIBE O MENU NA TELA*/
```

Esse é o topo do arquivo 'main.c', é chamado as bibliotecas 'stdio.h', 'stdlib.h' e 'arvore.h'. É definido duas constantes com pré-processamento também, a constante NOME representa o tamanho que um array onde vai armazenar o nome, a outra constante representa o tamanho do array onde vai ser armazenado a matrícula. Os arrays de nome e matrícula vão ser ponteiros para char.

```
arvore.c [] main.c [] arvore.h []
44 int main(int argc, char *argv[]) {
45     Abb *raiz;
46     clearWindow();
47     raiz = inicializa();
48
49     if (argv[1] == NULL) { // USUARIO NAO PASSOU O ARQUIVO
50         printf("ERRO Arquivo não encontrado!"); format:
51         printf("\nForma de execução esperado 'arquivo.out local_do_arquivo.txt'\n"); format:
52         return 0;
53     }
54
55     long int linhas = contarEstudantes(argv[1]); // QUANTIDADE DE ALUNOS file_name:
56     lerArquivo(&raiz, argv[1], linhas); // LER ARQUIVO raiz: file_name:
57 }
```

No início da função 'main' é esperado um parâmetro, que nesse caso é o nome do arquivo. Posteriormente o arquivo é aberto para contar a quantidade de estudantes nele e posteriormente é feita a abertura e é inserido os alunos na árvore binária da variável 'raiz'.

```
57
58     int opcao;
59
60     do {
61         /*EXIBE O MENU NA TELA E RETORNA A OPCAO ESCOLHIDA*/
62         opcao = menu();
63         switch (opcao) {
64             case 1:
65                 inserir(&raiz); arvore:
66                 clear_buff();
67                 getchar();
68                 break;
69             case 2:
70                 remover(&raiz); raiz:
71                 break;
72             case 3:
73                 printLista(raiz); arvore:
74                 clear_buff();
75                 getchar();
76                 break;
77             case 4:
78                 buscar(raiz); arvore:
79                 clear_buff();
80                 getchar();
81                 break;
82             case 5:
83                 salvar(raiz, argv[1]); file_name:
84                 clear_buff();
85                 getchar();
86                 break;
87         }
88     } while (opcao != 0);
89 }
```

Após o armazenamento dos dados presente no arquivo na árvore, é exibido o menu de opções para o usuário e podendo escolher 6 opção que são indicadas, elas são selecionadas pelos valores discretos de 0 à 5.

Quando o programa é encerrado é chamado uma função para liberar toda a memória alocada pelos nós da árvore e suas variáveis.

COMPILAR E EXECUTAR

Para compilar basta digitar o seguinte comando:

→ **gcc main.c**

Na execução do programa é exigido que passe um arquivo como parâmetro, nesse caso arquivo de alunos:

→ **./<arquivo.out> lista_matricula_alunos_2023.txt**

Após executar esses comando você deve ver a seguinte mensagem na tela:

Leitura de arquivo concluida

1 - Inserir

2 - Remover

3 - Printar arvore

4 - Buscas elemento

5 - Salvar base de dados

0 - Sair

Option: