UNIVERSIDADE FEDERAL DE LAVRAS

TRABALHO

Algoritmo e Estrutura de Dados III - GCC109

Alunos: Augusto Soares Pereira - 20120320
Álvaro dos Reis Cozadi - 201211012
Ueslei Marcelino da Guia - 201120619

Lavras - MG
2014

1. Introdução

Na ciência da computação uma **árvore B** é uma estrutura de dados projetada para funcionar especialmente em memória secundária como um disco magnético ou outros dispositivos de armazenamento secundário. Dentre suas propriedades ela permite a inserção, remoção e busca de chaves numa complexidade de tempo logarítmica e, por esse motivo, é muito empregada em aplicações que necessitam manipular grandes quantidades de informação tais como um banco de dados ou um sistema de arquivos.

Inventada por Rudolf Bayer e Edward Meyers McCreight em 1971 enquanto trabalhavam no Boeing Scientific Research Labs, a origem do nome (**árvore B**) não foi definida por estes. Especula-se que o B venha da palavra balanceamento, do nome de um de seus inventores Bayer ou de *Boeing*, nome da empresa.

2. Organização do código.

O código está dividido em 4 arquivos. No arquivo arvoreB.c estão implementados as função de manipulção e inicialização da arvore B. Já no arquivo arvoreB.h está implementados os arquivo referente ao cabeçalho de implementação da árvore B, as estruturas Index, DataIndex, Aluno, Register, Page, AuxPage e BinaryPage. A definição completa de cada uma está muito bem documentada no código.

O arquivo main.c corresponde ao arquivo principal com função main e interface implementadas. No final deste arquivo, estão os códigos para escrever o indice da arvore e fechar o arquivo.

Por fim o arquivo makefile, gera um arquivo de dados através do seguinte comandos:

```
make
./SysAlunos <nome_do_arquivo>
```

E através do comando `make limpar`, este arquivo é apagado.

3. Funcionalidade Implementadas

Informações completas sobre a implementação do trabalho pode ser encontradas no código fonte, que está muito bem documentado, indentado além de ser alto-explicativo.

3.1 Estrutura

Estrutura inicial da arvore.

```
typedef struct Index {
int TreeRRN;
int CurrentRRN;
      } Index;
```

Estrutura dos nós ativos

```
typedef struct DataIndex {
 int TotalRRN;
 int AtivosRRN;
 } DataIndex;
```

Estrutura referente aos Alunos

```
typedef struct Aluno {

char Nome[50];
char Identidade[TAM_CHAVE + 1];
char CPF[15];
char Matricula[TAM_CHAVE + 1];
double RSG;
}Aluno;
```

Estrutura referente ao Registro

```
typedef struct Register {

char Chave[TAM_CHAVE + 1];
int FileRRN; /*Reference for Structure on File.*/
} Register;
```

Estrutura da Pagina

```
typedef struct Page {
```

```
     /* data */
int RRN;
Register RegisterArray[MAXKEYS];
int RRNsArray[MAXKEYS + 1];
short NumberOfKeys;
} Page;
```

Estrutura da Pagina Auxiliar

```
typedef struct AuxPage {
/* data */
int RRN;
short NumberOfKeys;
Register RegisterArray[MAXKEYS + 1];
int RRNsArray[MAXKEYS + 2];
} AuxPage;
```

Estrutura da Pagina auxiliar que sera gravada no arquivo binario

```
typedef struct BinaryPage {

Register RegisterArray[MAXKEYS];
long RRNsArray[MAXKEYS + 1];
short NumberOfKeys;
} BinaryPage;
```

3.2 - Inserir um novo Aluno

Implementação do código responsável na inserção de um novo aluno. O código recebe os dados válidos de aluno para serem gravados na árvore e por fim, gravados no disco. Linhas 54 a 91 no arquivo **main.c**.

3.3 - Remover um Aluno

A remoção pode ser feita de duas formas. Pode ser realizada pelo número de identificação do aluno inserido ou pelo número de matrícula do aluno.
Primeiro o algoritmo faz a identificação de qual opção de remoção foi digitada e depois realiza a busca pelos dados do aluno a ser removido. Linhas 100 a 130 a remoção é por identidade. E linhas 132 a 163 a remoção é feita por matrícula. Ambas informações se encontram no arquivo **main.c**.

3.4 - Busca por Aluno

A busca pode ser feita de duas formas. Pode ser realizada pelo número de identificação do aluno inserido ou pelo número de matrícula do aluno.
O algoritmo faz a identificação de qual opção de busca foi digitada e depois realiza a busca pelos dados do aluno. Linhas 172 a 189 a busca é por identidade. E linhas 190 a 209 a busca é feita por matrícula. Ambas informações se encontram no arquivo **main.c**.

3.5 - Atualizar os dados

Na atualização dos dados do alunos já cadastrado, o algoritmo precisa de novos valores validos para o aluno. A atualização pode ser feita de duas formas. Pode ser realizada pelo número de identificação do aluno inserido ou pelo número de matrícula do aluno. Linhas 219 a 248 a atualização é por identidade. E linhas 249 a 284 a atualização é feita por matrícula. Ambas informações se encontram no arquivo **main.c**.

3.6 - Fechamento do arquivo

Por fim, o código deve realizar a escrita dos dados e fechar o arquivo para evitar qualquer tipo de erro. A implementação que realizará isso é a seguinte:

```
/* Escreve o indice da arvore e fecha o arquivo */
       Btree = BtreeID;
       ind = indID;
       WriteIndex();
       free(ind);
       fclose(Btree);

       Btree = BtreeMAT;
       ind = indMAT;
       WriteIndex();
       free(ind);
       fclose(Btree);

       fclose(Data);
       return (EXIT_SUCCESS);
```

4. Mecanismos

Os algoritmos do trabalho de referencia foram implementados na IDE NetBeans versão 7.2.2 utilizando o S.O Ubuntu 12.04. As modificações feitas para atender os requisitos foram feitas na IDE Geany 0.21 utilizando o S.O Ubuntu 14.04.

A execução do trabalho deve ser realizada utilizando o compilador GCC da seguinte forma:
Passo 1 - `make`
Passo 2 - `./Sysalunos <nome_arquivo>`
Passo 3 - `Entrada dos dados`