Relatório sobre o Código Fonte do Projeto, Árvore B+

UFES Centro Universitário Norte do Espírito Santo

Daniel Morais
preencher@edu.ufes.br
Joilnen Leite
joilnen.leite@edu.ufes.br
Matheus Cruz
preencher@edu.ufes.br

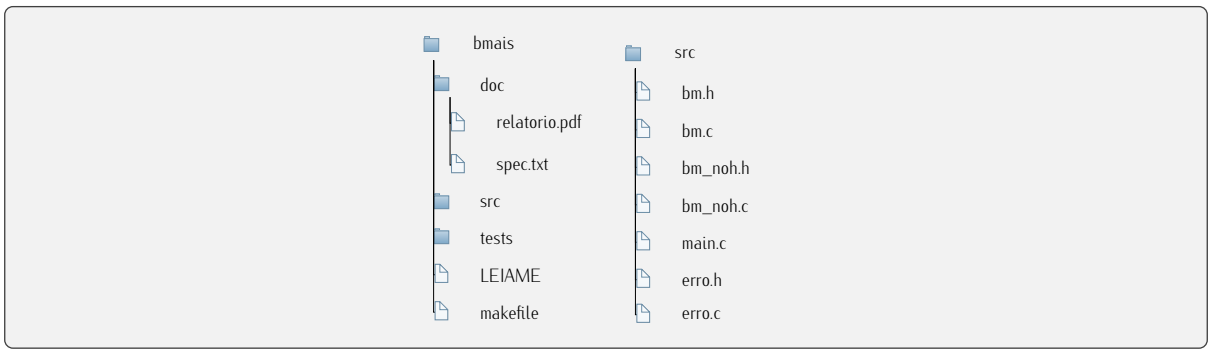
Resumo: Relatório básico sobre o conteúdo e processo de desenvolvimento da atividade sobre árvore B+

Feito em LaTeX

Palavras-chave: fontes, C, árvore B+

1 Introdução

Esta biblioteca é composta pelo os seguintes arquivos,



Em **src** estão os arquivos específicos da biblioteca

- bm.h
- bm.c
- bm_noh.h
- bm_noh.c
- main.h
- Makefile.h

Todos os arquivos estão listados nos anexos na sua íntegra.

Como foi implementado um número grande de testes, estes foram separados em quatro arquivos, **testa_item_1.c, testa_item_2.c, testa_item_3.c, testa_rb.c** e tem suas funções chamadas sequencialmente dentro da função **main**, no arquivo **main.c**

Seguiremos neste relatório uma abordagem **top-down** onde partiremos das estruturas e funções manipuladas pelo o código cliente em direção as estruturas e funções que implementam e operam na estrutura de dados, **árvore red-black**, que é totalmente ocultada do cliente ou seja poderíamos reimplementar as funcionalidades com outras estruturas de dados e manter a interface compatível com a existente.

O estilo do código fonte neste trabalho é o mais tradicional, chaves abrem e fecham do mesmo lado nas funções e instruções escritas em mais de uma linha, entre cada instrução e seus operandos há sempre espaços, com exceção das funções e seus parênteses, os espaços dividem visialmente os tokens, como em arrays, em equações, símbolo de ponteiro alinhado à variável, exceto em alguns ponteiros para função, todos os comentários seguem ANSI C, /* */.

Na leitura da documentação nos comentários vale a pena ressaltar que todos estam em ASCII, por isso não tem acentuação e a descrição dos parâmetros são antecedidas com **@param** que é tag utilizada pelo o sistema que gera documentação apartir do código fonte, documentação esta constante nos anexos.

2 As Estruturas

2.1 bm_noh

A primeira estrutura que veremos aqui é a **bm_noh** ela representa um nó na árvore B+

```
20 typedef struct bm_noh {
21   int *chaves, nchaves;
22   int mgrau;
23   struct bm_noh **filhos;
24   long **dfilhos;
25   char eh_folha;
26 } bm_noh;
27
28 bm_noh *bm_noh_inic(int i, char eh_folha);
29 void bm_noh_split(bm_noh *bmn, bm_noh *y, int i);
30 void bm_noh_insere(bm_noh *bmn, int k);
31 void bm_noh_escrutina(bm_noh *bmn);
32 bm_noh *bm_noh_pesquisa(bm_noh *bmn, int k);
33 int bm_noh_contem(bm_noh *bmn, int chave);
34 void bm_noh_escrevedisc(bm_noh *bmn);
35 s_artigo *bm_noh_ledisc(bm_noh *bmn, int chave);
36 void bm_noh_populaart(int chave, s_artigo *art);
```

Listing 1: Fragmento do bm_noh.h

Aqui bemos implementada a função que cria um bm_noh, um tipo que representa uma nó de uma árvore B+

```
12 bm_noh *bm_noh_inic(int i, char eh_folha)
13 {
14   bm_noh *n = (bm_noh *)malloc(sizeof(bm_noh));
15   n->mgrau = i;
16   n->chaves = (int *)malloc(sizeof(int) * NCHAVES(n->mgrau));
17   n->filhos = (bm_noh **)malloc(sizeof(bm_noh) * NFILHOS(n->mgrau));
18   n->dfilhos = (long **)malloc(sizeof(long) * NCHAVES(n->mgrau));
19
20   n->nchaves = 0;
21   n->eh_folha = eh_folha;
22   return n;
23 }
```

Listing 2: Fragmento do bm_noh.c

Fazendo uso da estrutura anterior em um nível acima temos a bm, aqui sua definição e os cabeçaçhos das suas funções correspondetes.

```
4  #include "bm_noh.h"
5
6  typedef struct bm {
7    struct bm_noh *raiz;
8    int mgrau, *felem;
9  } bm;
10
11 bm *bm_inic(int i);
12 void bm_escrutina(bm *b);
13 bm_noh *bm_pesquisa(bm *b, int chave);
14 void bm_insere(bm *b, int chave);
```

Listing 3: Fragmento do bm.h

3

4 Apêndices