Referência do Arquivo functions.h

```
#include "dados.h"
#include <bits/stdc++.h>
#include <iostream>
```

Vá para o código-fonte desse arquivo.

Funções

int	**funcaoHash** (int chave)
int	**contaChaves** (**noArvore** noDado)
int	**contaPonteiros** (**noArvore** noDado)
void	**apagaParesNo** (**noArvore** *no)
void	**copiaParesNo** (**noArvore** *no, **noArvoreTemp** *tmp, int inicioTemp, int fim)
void	**copiaTodosParesNo** (**noArvore** *no, **noArvoreTemp** *tmp)
void	**moveParesNo** (**noArvore** *no, int pos, int quntidadeChaves)
void	**moveParesNo** (**noArvoreTemp** *no, int pos, int quntidadeChaves)
int	**posicaoPai** (int node_offset)
void	**copiaPaiNo** (**noArvore** *P, **noArvoreTempPai** *TP, int chave, int offsetChave)
void	**copiaPorPonteiro** (**noArvore** *no, **noArvoreTempPai** *TP, int inicioTemp, int fim)
void	**copiaPorPonteiro2** (**noArvore** *no, **noArvoreTempPai** *TP, int inicioTemp, int fim)
void	**printDadosBusca** (**dadoBusca** d)

Variáveis

std::vector< int >	**vetorPais**
std::vector< int >	**vetorPaisFind**

Funções

◆ apagaParesNo()

void apagaParesNo (**noArvore** * no)

Apaga os valores de chave e ponteiro, setando para -1

Autor: Aldemir

```
53 {
54   for (int i = 0; i < QUANTIDADE_PONTEIROS - 1; i++)
55   {
56     (*no).pares[i].chave = -1;
57     (*no).pares[i].endereco = -1;
58   }
59   (*no).ponteiroM = -1;
60 }
```

◆ contaChaves()

int contaChaves (**noArvore** noDado)

Retorna a quantidade de chaves existentes em um nó *

Autor: Erlon

```
25 {
26   int counter = 0;
27   while (noDado.pares[counter].chave != -1 && counter != QUANTIDADE_PONTEIROS
   - 1)
28     counter++;
29   return counter;
30 }
```

◆ contaPonteiros()

int contaPonteiros (**noArvore** noDado)

Retorna a quantidade de ponteiros existente em um nó

Autor: Glenn

```
37 {
38   int counter = 0;
39   while (noDado.pares[counter].endereco != -1 && counter !=
  QUANTIDADE_PONTEIROS - 1)
40     counter++;
41   if (noDado.ponteiroM != -1)
42   {
43     counter++;
44   }
45   return counter;
46 }
```

◆ copiaPaiNo()

```
void copiaPaiNo ( noArvore *          P,
                  noArvoreTempPai *   TP,
                  int                 chave,
                  int                 offsetChave
                )
```

Copia todos os ponteiros e chavesde um nó **noArvore** para um nó **noArvoreTempPai**, e aadiciona chave e offsetChave no final

Autor: Erlon

```
133 {
134   int i;
135   for (i = 0; i < QUANTIDADE_PONTEIROS - 1; i++)
136   {
137     (*TP).pares[i] = (*P).pares[i];
138   }
139   (*TP).pares[i].endereco = (*P).ponteiroM;
140   (*TP).pares[i].chave = chave;
141   (*TP).ponteiroM = offsetChave;
142 }
```

◆ copiaParesNo()

```
void copiaParesNo ( noArvore *        no,
                    noArvoreTemp *    tmp,
                    int               inicioTemp,
                    int               fim
                  )
```

Copia os valores de um nó auxiliar (**noArvoreTemp**) para o nó **noArvore** de um determinada posição a outra

Autor: Erlon

```
67 {
68   for (int i = inicioTemp, j = 0; i <= fim; i++, j++)
69   {
70     (*no).pares[j] = (*tmp).pares[i];
71   }
72 }
```

◆ copiaPorPonteiro()

```
void copiaPorPonteiro ( noArvore *          no,
                        noArvoreTempPai *   TP,
                        int                 inicioTemp,
                        int                 fim
                      )
```

Copia os valores de um nó **noArvoreTempPai** para um nó **noArvore**, de 0 à teto(N/2)-1

Autor: Aldemir

```
149 {
150   int i, j;
151   for (i = inicioTemp, j = 0; i < fim; i++, j++)
152   {
153     (*no).pares[j] = (*TP).pares[i];
154   }
155   (*no).pares[j].endereco = (*TP).pares[i].endereco;
156 }
```

◆ copiaPorPonteiro2()

```
void copiaPorPonteiro2 ( noArvore *          no,
                         noArvoreTempPai *   TP,
                         int                 inicioTemp,
                         int                 fim
                       )
```

Copia os valores de um nó **noArvoreTempPai** para um nó **noArvore**, de teto(N/2) à até o fim(N)

Autor: Glenn

```
163 {
164   int i, j;
165   for (i = inicioTemp, j = 0; i < fim; i++, j++)
166   {
167     (*no).pares[j] = (*TP).pares[i];
168   }
169   (*no).pares[j].endereco = (*TP).ponteiroM;
170 }
```

◆ copiaTodosParesNo()

void copiaTodosParesNo (**noArvore** * no,
noArvoreTemp * tmp
)

Copia todos os valores de **noArvore** para uma **noArvoreTemp**

Autor: Glenn

```
79 {
80   for (int i = 0; i < QUANTIDADE_PONTEIROS - 1; i++)
81   {
82     (*tmp).pares[i] = (*no).pares[i];
83   }
84   (*tmp).pares[QUANTIDADE_PONTEIROS - 1].chave = -1;
85   (*tmp).pares[QUANTIDADE_PONTEIROS - 1].endereco = -1;
86 }
```

◆ funcaoHash()

int funcaoHash (int chave)

Cria um hash para uma determinada chave

A implementação foi a mais simples possível com uma função mod %

Autor: Aldemir

```
16 {
17   return (chave - 1) % QUANTIDADE_BUCKETS;
18 }
```

◆ moveParesNo() [1/2]

void moveParesNo (**noArvore** * no,
int pos,
int quntidadeChaves
)

Desloca os valores de um determinado nó para a inserção de uma nova chave

Autor: Aldemir

```
93 {
94   for (int j = quntidadeChaves + 1; j > pos; j--)
95   {
96     (*no).pares[j] = (*no).pares[j - 1];
97   }
98 }
```

◆ moveParesNo() [2/2]

```
void moveParesNo ( noArvoreTemp *  no,
                   int             pos,
                   int             quntidadeChaves
                 )
```

Desloca os valores de um determinado nó para a inserção de uma nova chave

Autor: Erlon

```
105 {
106   for (int j = quntidadeChaves; j > pos; j--)
107   {
108     (*no).pares[j] = (*no).pares[j - 1];
109   }
110 }
```

posicaoPai()

```
int posicaoPai ( int  node_offset )
```

Retorna a posição do pai de um determinado nó em relação ao caminho feito na inserção ou busca

Autor: Glenn

```
117 {
118   for (int i = 0; i < vetorPais.size(); i++)
119   {
120     if (node_offset == vetorPais[i])
121     {
122       return vetorPais[i - 1];
123     }
124   }
125   return 0;
126 }
```

printDadosBusca()

```
void printDadosBusca ( dadoBusca  d )
```

Imprime os dados de um bloco no arquivo de hash se a consulta for bem sucedida

Autor: Erlon

```
178 {
179   std::cout << "ID: " << d.artigoDado.ID << std::endl;
180   std::cout << "Titulo: " << d.artigoDado.Titulo << std::endl;
181   std::cout << "Ano : " << d.artigoDado.Ano << std::endl;
182   std::cout << "Autores: " << d.artigoDado.Autores << std::endl;
183   std::cout << "Citacoes: " << d.artigoDado.Citacoes << std::endl;
184   std::cout << "Atualizacao: " << d.artigoDado.Atualizacao << std::endl;
185   std::cout << "Snippet: " << d.artigoDado.Snippet << std::endl;
186   std::cout << "Blocos lidos: " << d.node_level << std::endl;
187   std::cout << "Quantidade de blocos: " << d.quantidadeBlocos + 1 << std::endl
188   std::cout << "-----------------------------------------------" << std::endl;
189 }
```

Variáveis

◆ vetorPais

std::vector<int> vetorPais

◆ vetorPaisFind

std::vector<int> vetorPaisFind