# Referência do Arquivo seek1.cpp

```
#include "csv.h"
#include "functions.h"
#include <cmath>
#include <iostream>
```

## Funções

| | |
|---|---|
| **dadoBusca** | **buscaNaArvore** (int chave) |
| int | **main** (int argc, char *argv[]) |

## Variáveis

| | |
|---|---|
| FILE * | **ponteiroArvore** |

## Funções

### ◆ buscaNaArvore()

**dadoBusca** buscaNaArvore ( int chave )

Busca uma chave na árvore (índice).

- Se a busca for bem sucedida, retorna as informações da registro cuja chave corresponde à chave de busca
- Caso contrário, retorna uma mensagem informando que ao registro não pode ser encontrado

Autor: Erlon

```
17 {
18   FILE *ponteiroHash = fopen("./hash.bin", "r");
19   cabecalhoArvore dadoCabecalho;
20   noArvore noDado;
21   tipoBloco bloco;
22   int noAtual;
23   fseek(ponteiroArvore, 0, SEEK_SET);
24   fread(&dadoCabecalho, sizeof(cabecalhoArvore), 1,
25         ponteiroArvore);
26   if (dadoCabecalho.enderecoRaiz == -1)
27   {
28     std::cout << "Error: Ávore vazia!!" << std::endl;
29   }
30   else
31   {
32     fseek(ponteiroArvore, 0, SEEK_SET);
33     fread(&dadoCabecalho, sizeof(cabecalhoArvore), 1, ponteiroArvore);
34     fseek(ponteiroArvore, (dadoCabecalho.enderecoRaiz * TAMANHO_BLOCO),
35           SEEK_CUR);
36
37     int nivelAtual = 1;
38     noAtual = dadoCabecalho.enderecoRaiz;
39     while (nivelAtual !=
40            dadoCabecalho.alturaArvore)
41     {
42       int i = 0;
43       fread(&noDado, sizeof(noArvore), 1,
44             ponteiroArvore);
45       while (true)
46       {
47         if (i == QUANTIDADE_PONTEIROS - 1 && noDado.pares[i - 1].chave <
chave)
48         {
49           noAtual = noDado.ponteiroM;
50           fseek(ponteiroArvore, sizeof(cabecalhoArvore) + noAtual * TAMANHO_BL
51                 SEEK_SET);
52           nivelAtual++;
53           break;
54         }
55         else if (noDado.pares[i].chave > chave ||
56                  noDado.pares[i].chave ==
57                      -1)
58         {
59           noAtual = noDado.pares[i].endereco;
60           fseek(ponteiroArvore, sizeof(cabecalhoArvore) + noAtual * TAMANHO_BL
61                 SEEK_SET);
62           nivelAtual++;
63           break;
64         }
65         i++;
66       }
67     }
68     fread(&noDado, sizeof(noArvore), 1, ponteiroArvore);
69     int quntidadeChaves = contaChaves(noDado);
70     for (int i = 0; i < quntidadeChaves; i++)
71     {
72       if (noDado.pares[i].chave == chave)
73       {
74         int offsetChave = noDado.pares[i].endereco;
75         fseek(ponteiroHash, offsetChave * TAMANHO_BLOCO,
76               SEEK_SET);
77         fread(&bloco, TAMANHO_BLOCO, 1,
78               ponteiroHash);
79         for (int j = 0; j < bloco.quantidadeArtigos; j++)
```

```
80          {
81            if (bloco.vetorArtigos[j].ID == chave) {
82              dadoBusca result = {bloco.vetorArtigos[j], dadoCabecalho.quantidad
83                                  dadoCabecalho.alturaArvore};
84              printDadosBusca(result);
85              fclose(ponteiroHash);
86              return result;
87            }
88          }
89        }
90      }
91    }
92    std::cout << "Não encontrou a chave" << std::endl;
93    fclose(ponteiroHash);
94    return {{}, dadoCabecalho.quantidadeBlocos, dadoCabecalho.alturaArvore};
95  }
```

## ◆ main()

int main ( int argc,
char * argv[]
)

```
98  {
99
100   std::string sEntrada = argv[1];
101   int nomeEntrada = stoi(sEntrada);
102   ponteiroArvore = fopen("./primarytree.bin", "r");
103   buscaNaArvore(nomeEntrada);
104
105   return 0;
106 }
```

# Variáveis

## ◆ ponteiroArvore

FILE* ponteiroArvore