Projeto II

1.0

Generated by Doxygen 1.8.17

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Namespace Documentation

4.1 structures Namespace Reference

Classes

- struct TrieNode

  Árvore de prefixos.

Functions

- struct TrieNode ∗ initNode ()

  Aloca memória para um novo TrieNode.
- void insert (struct TrieNode ∗root, std::string word, int pos, int len)

  Adiciona uma chave na árvore de prefixos.
- int findPrefix (struct TrieNode ∗index)
- std::tuple< int, int, int > search (struct TrieNode ∗, std::string word)

  Procura por uma palavra na árvore de prefixos.
- int countWords (std::istream &in)

4.1.1 Function Documentation

4.1.1.1 countWords()

```
int structures::countWords (
            std::istream & in )
```

4.1.1.2 findPrefix()

```
int structures::findPrefix (
            struct TrieNode * index )
```

4.1.1.3 initNode()

```
struct TrieNode* structures::initNode ( )
```

Aloca memória para um novo TrieNode.

A posição e o comprimento de cada TrieNode começam zerados. Além disso, todos as posições do vetor de nodos filhos começam nulas.

Returns

struct TrieNode* Ponteiro para o novo TrieNode criado.

4.1.1.4 insert()

```
void structures::insert (
            struct TrieNode * root,
            std::string word,
            int pos,
            int len )
```

Adiciona uma chave na árvore de prefixos.

Parameters

root	Root da árvore.
word	Palavra para inserir.
pos	Posicao no dicionario da palavra a ser inserida.
len	Comprimento da linha no dicionario que possui a palavra a ser inserida.

4.1.1.5 search()

```
std::tuple<int, int, int> structures::search (
            struct TrieNode * ,
            std::string word )
```

Procura por uma palavra na árvore de prefixos.

Parameters

word	Palavra a ser procurada na trie.
root	Raíz da árvore.

Returns

pair<int,int> Par que indica se a palavra pertence ao dicionário, é apenas um prefixo ou que não pertence ao dicionário. Se a palavra pertencer ao dicionário, a primeira entrada representa a posição da palavra enquanto a segunda, o comprimeto da linha.

Chapter 5

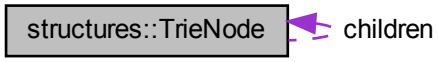
Class Documentation

5.1 structures::TrieNode Struct Reference

Árvore de prefixos.

`#include <trie.h>`

Collaboration diagram for structures::TrieNode:



Public Attributes

- struct TrieNode * children [ALPHABET_SIZE]
- int pos
- int len

5.1.1 Detailed Description

Árvore de prefixos.

A árvore de prefixos é uma estrutura de dados eficiente no que diz respeito à recuperação de informações. Por isso ela também é conhecida como Trie, de reTRIEval. Usando uma Trie bem organizada, pode-se alcançar complexidade de busca O(M), onde M representa o tamanho máximo de suas chaves. Ao passo que, uma B↩ ST balanceada teria tempo proporcional a MlogN, em que N representa o número de chaves na árvore. Essas vantagens todavia não são sem seus custos, uma vez que a Trie perde para a BST em espaço ocupado em memória.

5.1.2 Member Data Documentation

5.1.2.1 children

```
struct TrieNode* structures::TrieNode::children[ALPHABET_SIZE]
```

5.1.2.2 len

```
int structures::TrieNode::len
```

5.1.2.3 pos

```
int structures::TrieNode::pos
```

The documentation for this struct was generated from the following file:

- src/trie.h

Chapter 6

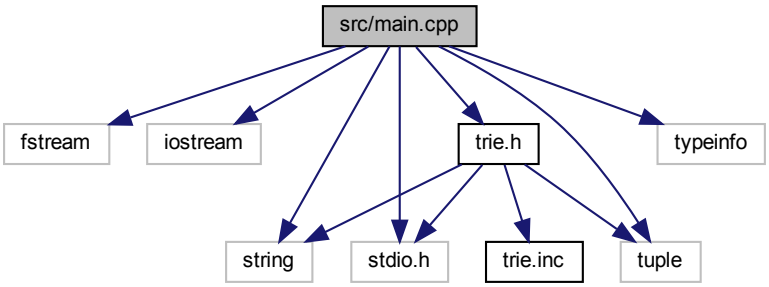
File Documentation

6.1 src/main.cpp File Reference

Código do programa principal.

```
#include <fstream>
#include <iostream>
#include <string>
#include <stdio.h>
#include <typeinfo>
#include <tuple>
#include "trie.h"
```

Include dependency graph for main.cpp:



Functions

- int main ()

  Programa principal, realiza a leitura e processamento dos dicionários e indica o que as palavras da entrada são, se a palavra pertence ao dicionário é impresso a sua posição e o comprimeto da linha em que a palavra está.

6.1.1 Detailed Description

Código do programa principal.

Author

Rafael Nilson Witt

Version

1.0

Date

2021-05-05

Copyright

Copyright (c) 2021

6.1.2 Function Documentation

6.1.2.1 main()

```
int main ( )
```

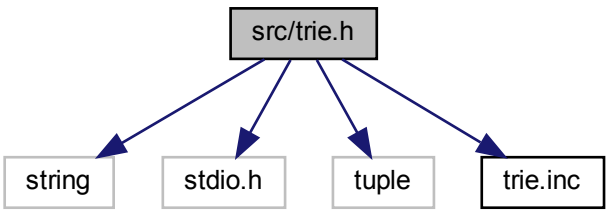
Programa principal, realiza a leitura e processamento dos dicionários e indica o que as palavras da entrada são, se a palavra pertence ao dicionário é impresso a sua posição e o comprimeto da linha em que a palavra está.

6.2 src/trie.h File Reference

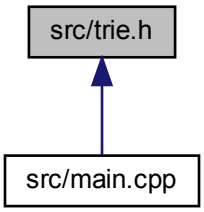
Código da trie.

```
#include <string>
#include <stdio.h>
#include <tuple>
#include "trie.inc"
```

Include dependency graph for trie.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct structures::TrieNode

  Árvore de prefixos.

Namespaces

- structures

Macros

- #define ALPHABET_SIZE 26

Functions

- struct TrieNode ∗ structures::initNode ()

  Aloca memória para um novo TrieNode.
- void structures::insert (struct TrieNode ∗root, std::string word, int pos, int len)

  Adiciona uma chave na árvore de prefixos.
- int structures::findPrefix (struct TrieNode ∗index)
- std::tuple< int, int, int > structures::search (struct TrieNode ∗, std::string word)

  Procura por uma palavra na árvore de prefixos.
- int structures::countWords (std::istream &in)

6.2.1 Detailed Description

Código da trie.

Author

Rafael Nilson Witt

Version

1.0

Date

2021-05-05

Copyright

Copyright (c) 2021

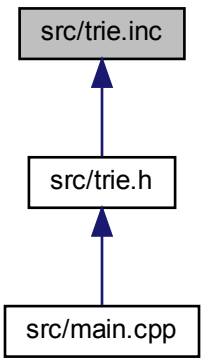
6.2.2 Macro Definition Documentation

6.2.2.1 ALPHABET_SIZE

```
#define ALPHABET_SIZE 26
```

6.3 src/trie.inc File Reference

This graph shows which files directly or indirectly include this file:



Index