

Projeto II - Identificação de prefixos e indexação de dicionários

Gerado por Doxygen 1.8.13

Sumário

1	Namespaces	1
1.1	Lista de Namespaces	1
2	Índice dos Componentes	3
2.1	Lista de Componentes	3
3	Índice dos Arquivos	5
3.1	Lista de Arquivos	5
4	Namespaces	7
4.1	Refência do Namespace structures	7
4.1.1	Descrição Detalhada	7
5	Classes	9
5.1	Referência da Classe structures::Trie	9
5.1.1	Descrição Detalhada	9
5.1.2	Construtores & Destrutores	9
5.1.2.1	Trie()	10
5.1.3	Métodos	10
5.1.3.1	count_children()	10
5.1.3.2	count_prefix()	10
5.1.3.3	count_words()	11
5.1.3.4	insert()	11
5.1.3.5	search()	11
6	Arquivos	13
6.1	Referência do Arquivo include/trie.h	13
6.1.1	Descrição Detalhada	13
6.1.2	Definições e macros	14
6.1.2.1	ALPHABET_SIZE	14
6.2	Referência do Arquivo main.cpp	14
6.2.1	Descrição Detalhada	14
6.2.2	Funções	14
6.2.2.1	main()	14
	Índice	15

Capítulo 1

Namespaces

1.1 Lista de Namespaces

Esta é a lista de todos os Namespaces com suas respectivas descrições:

structures	
Estruturas de dados	7

Capítulo 2

Índice dos Componentes

2.1 Lista de Componentes

Aqui estão as classes, estruturas, uniões e interfaces e suas respectivas descrições:

structures::Trie	
Trie 9

Capítulo 3

Índice dos Arquivos

3.1 Lista de Arquivos

Esta é a lista de todos os arquivos e suas respectivas descrições:

main.cpp	Código principal	14
include/ trie.h	Código da Trie	13

Capítulo 4

Namespaces

4.1 Refência do Namespace structures

Estruturas de dados.

Componentes

- class [Trie](#)
[Trie](#).

4.1.1 Descrição Detalhada

Estruturas de dados.

Capítulo 5

Classes

5.1 Referência da Classe `structures::Trie`

`Trie`.

```
#include <trie.h>
```

Métodos Públicos

- `Trie ()`
Construtor de um novo `Trie`.
- void `insert` (std::string word, int index, int length)
Adiciona uma chave na árvore.
- std::pair< int, int > `search` (std::string word)
Procura uma palavra na árvore.
- int `count_prefix` (std::string word)
Conta o número de vezes que a palavra é um prefixo na árvore.
- int `count_children` ()
Conta o número de filhos.
- int `count_words` ()
Conta o número de palavras a partir do nodo.

5.1.1 Descrição Detalhada

`Trie`.

5.1.2 Construtores & Destrutores

5.1.2.1 Trie()

```
structures::Trie::Trie ( )
```

Construtor de um novo [Trie](#).

Na construção do [Trie](#) a posição (index) e o comprimento (length) são definidos inicialmente como 0 e as posições do vetor de nodos filhos (children) são definidos inicialmente como nulas.

Retorna

[Trie](#) criado.

5.1.3 Métodos

5.1.3.1 count_children()

```
int structures::Trie::count_children ( )
```

Conta o número de filhos.

Retorna

Inteiro com número de filhos.

5.1.3.2 count_prefix()

```
int structures::Trie::count_prefix (
    std::string word )
```

Conta o número de vezes que a palavra é um prefixo na árvore.

Parâmetros

<i>word</i>	Palavra a ser procurada na árvore.
-------------	------------------------------------

Retorna

Inteiro com número de vezes que a palavra é prefixo na árvore.

5.1.3.3 `count_words()`

```
int structures::Trie::count_words ( )
```

Conta o número de palavras a partir do nó.

Retorna

Inteiro com número de palavras.

5.1.3.4 `insert()`

```
void structures::Trie::insert (
    std::string word,
    int index,
    int length )
```

Adiciona uma chave na árvore.

Parâmetros

<i>word</i>	Palavra para ser inserida.
<i>index</i>	Posição no dicionário da palavra a ser inserida.
<i>length</i>	Comprimento da linha do dicionário que possui a palavra a ser inserida.

5.1.3.5 `search()`

```
std::pair< int, int > structures::Trie::search (
    std::string word )
```

Procura uma palavra na árvore.

Parâmetros

<i>word</i>	Palavra a ser procurada na árvore.
-------------	------------------------------------

Retorna

Um par (`std::pair<int, int>`) que indica se a palavra pertence ao dicionário ou se é um prefixo. Caso a palavra pertença ao dicionário, o primeiro valor do par representa a posição da palavra enquanto o segundo representa o comprimento da linha.

A documentação para esta classe foi gerada a partir do seguinte arquivo:

- `include/trie.h`

Capítulo 6

Arquivos

6.1 Referência do Arquivo include/trie.h

Código da Trie.

```
#include <string>
```

Componentes

- class `structures::Trie`
Trie.

Namespaces

- `structures`
Estruturas de dados.

Definições e Macros

- `#define ALPHABET_SIZE 26`

6.1.1 Descrição Detalhada

Código da Trie.

Autores

Alisson Fabra da Silva e Eduardo Vinicius Betim.

Copyright

Copyright [2021] <Alisson Fabra da Silva, Eduardo Vinicius Betim>

6.1.2 Definições e macros

6.1.2.1 ALPHABET_SIZE

```
#define ALPHABET_SIZE 26
```

6.2 Referência do Arquivo main.cpp

Código principal.

```
#include <fstream>
#include <iostream>
#include <string>
#include "../include/trie.h"
```

Funções

- int `main` ()
Função main, ponto de partida para a execução do programa.

6.2.1 Descrição Detalhada

Código principal.

Autores

Alisson Fabra da Silva e Eduardo Vinicius Betim.

Copyright

Copyright [2021] <Alisson Fabra da Silva, Eduardo Vinicius Betim>

6.2.2 Funções

6.2.2.1 main()

```
int main ( )
```

Função main, ponto de partida para a execução do programa.

Recebe um arquivo dic e coloca as palavras em uma árvore de prefixos e faz a busca das palavras.

Retorna

Retorna 0 caso não ocorra erro e -1 caso o arquivo não abra.

Índice Remissivo

ALPHABET_SIZE

trie.h, [14](#)

count_children

structures::Trie, [10](#)

count_prefix

structures::Trie, [10](#)

count_words

structures::Trie, [10](#)

include/trie.h, [13](#)

insert

structures::Trie, [11](#)

main

main.cpp, [14](#)

main.cpp, [14](#)

main, [14](#)

search

structures::Trie, [11](#)

structures, [7](#)

structures::Trie, [9](#)

count_children, [10](#)

count_prefix, [10](#)

count_words, [10](#)

insert, [11](#)

search, [11](#)

Trie, [9](#)

Trie

structures::Trie, [9](#)

trie.h

ALPHABET_SIZE, [14](#)