

## Projeto II

1.0

Generated by Doxygen 1.8.17



<b>1 Namespace Index</b>	<b>1</b>
1.1 Namespace List . . . . .	1
<b>2 Class Index</b>	<b>3</b>
2.1 Class List . . . . .	3
<b>3 File Index</b>	<b>5</b>
3.1 File List . . . . .	5
<b>4 Namespace Documentation</b>	<b>7</b>
4.1 structures Namespace Reference . . . . .	7
4.1.1 Function Documentation . . . . .	7
4.1.1.1 countWords() . . . . .	7
4.1.1.2 findPrefix() . . . . .	8
4.1.1.3 initNode() . . . . .	8
4.1.1.4 insert() . . . . .	8
4.1.1.5 search() . . . . .	8
<b>5 Class Documentation</b>	<b>11</b>
5.1 structures::TrieNode Struct Reference . . . . .	11
5.1.1 Detailed Description . . . . .	11
5.1.2 Member Data Documentation . . . . .	12
5.1.2.1 children . . . . .	12
5.1.2.2 len . . . . .	12
5.1.2.3 pos . . . . .	12
<b>6 File Documentation</b>	<b>13</b>
6.1 src/main.cpp File Reference . . . . .	13
6.1.1 Detailed Description . . . . .	14
6.1.2 Function Documentation . . . . .	14
6.1.2.1 main() . . . . .	14
6.2 src/trie.h File Reference . . . . .	14
6.2.1 Detailed Description . . . . .	16
6.2.2 Macro Definition Documentation . . . . .	16
6.2.2.1 ALPHABET_SIZE . . . . .	16
6.3 src/trie.inc File Reference . . . . .	16
<b>Index</b>	<b>17</b>



# Chapter 1

## Namespace Index

### 1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">structures</a> . . . . .	7
--------------------------------------	---



## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">structures::TrieNode</a>	
Árvore de prefixos	11





## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

src/ <a href="#">main.cpp</a>		
	Código do programa principal . . . . .	13
src/ <a href="#">trie.h</a>		
	Código da trie . . . . .	14
src/ <a href="#">trie.inc</a>	. . . . .	16



## 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

<i>root</i>	Root da árvore.
<i>word</i>	Palavra para inserir.
<i>pos</i>	Posicao no dicionario da palavra a ser inserida.
<i>len</i>	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

<i>word</i>	Palavra a ser procurada na trie.
<i>root</i>	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 comprimento da linha.



## Chapter 5

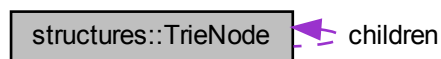
# Class Documentation

### 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  $M \log N$ , 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

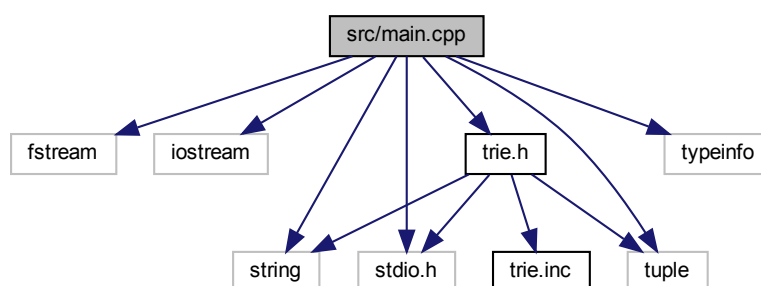
# File Documentation

### 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 comprimento 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()

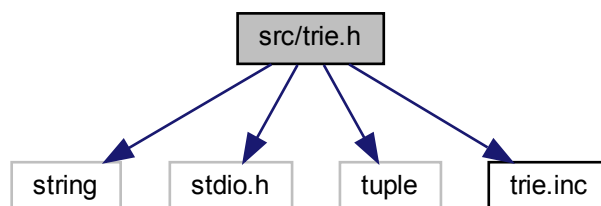
```
int main ( )
```

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 comprimento da linha em que a palavra está.

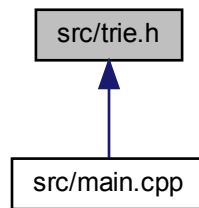
## 6.2 src/trie.h File Reference

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

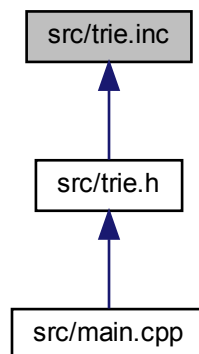
## 6.2.2 Macro Definition Documentation

### 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

ALPHABET\_SIZE

trie.h, [16](#)

children

structures::TrieNode, [12](#)

countWords

structures, [7](#)

findPrefix

structures, [7](#)

initNode

structures, [8](#)

insert

structures, [8](#)

len

structures::TrieNode, [12](#)

main

main.cpp, [14](#)

main.cpp

main, [14](#)

pos

structures::TrieNode, [12](#)

search

structures, [8](#)

src/main.cpp, [13](#)

src/trie.h, [14](#)

src/trie.inc, [16](#)

structures, [7](#)

countWords, [7](#)

findPrefix, [7](#)

initNode, [8](#)

insert, [8](#)

search, [8](#)

structures::TrieNode, [11](#)

children, [12](#)

len, [12](#)

pos, [12](#)

trie.h

ALPHABET\_SIZE, [16](#)