

Trabalho 3 tdd

Generated by Doxygen 1.8.14

Contents

1	File Index	1
1.1	File List	1
2	File Documentation	3
2.1	contador.cpp File Reference	3
2.1.1	Detailed Description	3
2.1.2	Function Documentation	4
2.1.2.1	abrir_arquivo()	4
2.1.2.2	conta_comments()	4
2.1.2.3	conta_final()	5
2.1.2.4	conta_linhas_branco()	6
2.1.2.5	conta_total_linhas()	6
2.1.2.6	eh_vazio()	7
2.1.2.7	nao_eh_vazio_indice()	8
2.2	contador.hpp File Reference	9
2.2.1	Detailed Description	9
2.2.2	Function Documentation	9
2.2.2.1	abrir_arquivo()	9
2.2.2.2	conta_comments()	10
2.2.2.3	conta_final()	11
2.2.2.4	conta_linhas_branco()	11
2.2.2.5	conta_total_linhas()	12
2.2.2.6	eh_vazio()	13
2.2.2.7	nao_eh_vazio_indice()	13
	Index	15

Chapter 1

File Index

1.1 File List

Here is a list of all documented files with brief descriptions:

contador.cpp	Arquivo com as funcoes implementadas para resolver problema referente a contar numeros de linhas de um programa	3
contador.hpp	Arquivo com as funcoes que sao usadas para implementar as funcoes que contaram o numero de linhas	9

Chapter 2

File Documentation

2.1 contador.cpp File Reference

Arquivo com as funcoes implementadas para resolver problema referente a contar numeros de linhas de um programa.

```
#include "../include/contador.hpp"
```

Functions

- int [abrir_arquivo](#) (string nome_arq, ifstream &arq)
Funcao que abri o arquivo de nome "nome_arq" e retorna em arq seu ponteiro.
- int [conta_total_linhas](#) (ifstream &arq)
Conta o numero total de linhas do arquivo.
- int [conta_linhas_branco](#) (ifstream &arq)
Conta o numero total de linhas em branco.
- int [eh_vazio](#) (const char *st)
Dado uma linha verifica se eh vazia ou se nao eh.
- int [nao_eh_vazio_indice](#) (const char *st)
Dado uma linha verifica qual o primeiro char que nao eh nulo "" ou tab.
- int [conta_comments](#) (ifstream &arq)
Conta o numero total de linhas que sao comentadas.
- int [conta_final](#) (ifstream &arq)
Calcula o numero de linhas realmente usadas.
- int [fecha_arquivo](#) (ifstream &arq)

2.1.1 Detailed Description

Arquivo com as funcoes implementadas para resolver problema referente a contar numeros de linhas de um programa.

Arquivo com os testes feitos no arquivo [contador.cpp](#) e sua respectiva biblioteca.

2.1.2 Function Documentation

2.1.2.1 abrir_arquivo()

```
int abrir_arquivo (
    string nome_arq,
    ifstream & arq )
```

Funcao que abri o arquivo de nome "nome_arq" e retorna em arq seu ponteiro.

Parameters

<i>nome_arq</i>	-> String contendo o nome do arquivo
<i>arq</i>	Representa o tipo stream que sera retornado

Returns

Retorna um se deu certo e zero caso de errado

Attention

Assertivas entradas – dado pelos clientes:

Precondition

nome_arq != NULL (correspondendo a um nome de arquivo)

Postcondition

arq != NULL, deve estar preparado para receber ponteiro, deve ser do tipo correto

Attention

Assertiva Saída - dado pelo Servidor:

Postcondition

se criado arq com sucesso - retorno 1
se deu erro - retorno 0

2.1.2.2 conta_comments()

```
int conta_comments (
    ifstream & arq )
```

Conta o numero total de linhas que sao comentadas.

Parameters

<i>arq</i>	Representa o tipo stream que sera usado
------------	---

Returns

Retorna o total de linhas que foram comentadas do arquivo em questao

Attention

Assertivas entradas - pelo Clientes:

Precondition

arq != NULL, referente ao *arq* atual, devendo ser do certo

Attention

Assertiva Saida - pelo Servidor :

Postcondition

Se deu erro retorna zero ou
Ou o numero de linhas comentadas

2.1.2.3 conta_final()

```
int conta_final (
    ifstream & arq )
```

Calcula o numero de linhas realmente usadas.

Parameters

<i>arq</i>	Representa o tipo stream que sera usado
------------	---

Returns

Retorna o total de linhas realmente uteis

Attention

Assertivas entradas - pelos clientes :

Precondition

arq != NULL, referente ao arq atual devendo ser do tipo certo

Attention

Assertiva Saida - pelo Servidor:

Postcondition

Retorna total - comentada - branco

2.1.2.4 conta_linhas_branco()

```
int conta_linhas_branco (  
    ifstream & arq )
```

Conta o numero total de linhas em branco.

Parameters

<i>arq</i>	Representa o tipo stream que sera usado
------------	---

Returns

Retorna o total de linhas em branco do arquivo em questao

Attention

Assertivas entradas - pelos clientes :

Precondition

arq != NULL, referente ao arq atual, deve ser do mesmo tipo

Attention

Assertiva Saida - pelo Servidor :

Postcondition

Se deu erro retorna zero ou
Ou o numero de linhas totais em branco

2.1.2.5 conta_total_linhas()

```
int conta_total_linhas (  
    ifstream & arq )
```

Conta o numero total de linhas do arquivo.

Parameters

<i>arq</i>	Representa o tipo stream que sera usado
------------	---

Returns

Retorna o total de linhas do arquivo em questao

Attention

Assertivas entradas - dado pelos Clientes:

Precondition

arq != NULL e do tipo correto, referente ao *arq* atual

Attention

Assertiva Saida - dado pelo Servidor:

Postcondition

Se deu erro retorna zero ou
Ou o numero de linhas totais

2.1.2.6 eh_vazio()

```
int eh_vazio (  
    const char * st )
```

Dado uma linha verifica se eh vazia ou se nao eh.

Parameters

<i>*st</i>	-> string que contem a linha toda
------------	-----------------------------------

Attention

Assertivas entradas - pelos clientes:

Returns

1: linha esta vazia "em branco" e 0: nao esta em branco

Precondition

st != NULL, a string deve existir

Attention

Assertiva Saida - pelo Servidor:

Postcondition

Se so houver espaco ou tab retorna 1, ou
retorna 1 caso contrario

2.1.2.7 nao_eh_vazio_indice()

```
int nao_eh_vazio_indice (
    const char * st )
```

Dado uma linha verifica qual o primeiro char que nao eh nulo "" ou tab.

Parameters

<code>*st</code>	-> string que contem a linha toda
------------------	-----------------------------------

Returns

-1 caso nao seja encontrada, ou retorna o indice que foi encontrado algum char

Attention

Assertivas entradas - pelos Clientes:

Precondition

st != NULL, a string deve existir e deve ser string

Attention

Assertiva Saida - pelo Servidor:

Postcondition

retorna -1 caso nao tenha, ou
retorna o indice do primeiro elemento diferente de tab ou espaco

2.2 contador.hpp File Reference

Arquivo com as funcoes que sao usadas para implementar as funcoes que contaram o numero de linhas.

```
#include <iostream>
#include <fstream>
#include <string>
#include <cstdlib>
#include <cstring>
```

Functions

- int [abrir_arquivo](#) (string nome_arq, ifstream &arq)
Funcao que abri o arquivo de nome "nome_arq" e retorna em arq seu ponteiro.
- int [conta_total_linhas](#) (ifstream &arq)
Conta o numero total de linhas do arquivo.
- int [conta_linhas_branco](#) (ifstream &arq)
Conta o numero total de linhas em branco.
- int [eh_vazio](#) (const char *st)
Dado uma linha verifica se eh vazia ou se nao eh.
- int [nao_eh_vazio_indice](#) (const char *st)
Dado uma linha verifica qual o primeiro char que nao eh nulo "" ou tab.
- int [conta_comments](#) (ifstream &arq)
Conta o numero total de linhas que sao comentadas.
- int [conta_final](#) (ifstream &arq)
Calcula o numero de linhas realmente usadas.
- int [fecha_arquivo](#) (ifstream &arq)

2.2.1 Detailed Description

Arquivo com as funcoes que sao usadas para implementar as funcoes que contaram o numero de linhas.

2.2.2 Function Documentation

2.2.2.1 abrir_arquivo()

```
int abrir_arquivo (
    string nome_arq,
    ifstream & arq )
```

Funcao que abri o arquivo de nome "nome_arq" e retorna em arq seu ponteiro.

Parameters

<i>nome_arq</i>	-> String contendo o nome do arquivo
<i>arq</i>	Representa o tipo stream que sera retornado

Returns

Retorna um se deu certo e zero caso de errado

Attention

Assertivas entradas – dado pelos clientes:

Precondition

nome_arq != NULL (correspondendo a um nome de arquivo)

Postcondition

arq != NULL, deve estar preparado para receber ponteiro, deve ser do tipo correto

Attention

Assertiva Saida - dado pelo Servidor:

Postcondition

se criado arq com sucesso - retorno 1
se deu erro - retorno 0

2.2.2.2 conta_comments()

```
int conta_comments (
    ifstream & arq )
```

Conta o numero total de linhas que sao comentadas.

Parameters

<i>arq</i>	Representa o tipo stream que sera usado
------------	---

Returns

Retorna o total de linhas que foram comentadas do arquivo em questao

Attention

Assertivas entradas - pelo Clientes:

Precondition

arq != NULL, referente ao arq atual, devendo ser do certo

Attention

Assertiva Saida - pelo Servidor :

Postcondition

Se deu erro retorna zero ou
Ou o numero de linhas comentadas

2.2.2.3 conta_final()

```
int conta_final (
    ifstream & arq )
```

Calcula o numero de linhas realmente usadas.

Parameters

<i>arq</i>	Representa o tipo stream que sera usado
------------	---

Returns

Retorna o total de linhas realmente uteis

Attention

Assertivas entradas - pelos clientes :

Precondition

arq != NULL, referente ao *arq* atual devendo ser do tipo certo

Attention

Assertiva Saida - pelo Servidor:

Postcondition

Retorna total - comentada - branco

2.2.2.4 conta_linhas_branco()

```
int conta_linhas_branco (
    ifstream & arq )
```

Conta o numero total de linhas em branco.

Parameters

<i>arq</i>	Representa o tipo stream que sera usado
------------	---

Returns

Retorna o total de linhas em branco do arquivo em questao

Attention

Assertivas entradas - pelos clientes :

Precondition

arq != NULL, referente ao *arq* atual, deve ser do mesmo tipo

Attention

Assertiva Saida - pelo Servidor :

Postcondition

Se deu erro retorna zero ou
Ou o numero de linhas totais em branco

2.2.2.5 *conta_total_linhas()*

```
int conta_total_linhas (  
    ifstream & arq )
```

Conta o numero total de linhas do arquivo.

Parameters

<i>arq</i>	Representa o tipo stream que sera usado
------------	---

Returns

Retorna o total de linhas do arquivo em questao

Attention

Assertivas entradas - dado pelos Clientes:

Precondition

arq != NULL e do tipo correto, referente ao arq atual

Attention

Assertiva Saida - dado pelo Servidor:

Postcondition

Se deu erro retorna zero ou
Ou o numero de linhas totais

2.2.2.6 eh_vazio()

```
int eh_vazio (
    const char * st )
```

Dado uma linha verifica se eh vazia ou se nao eh.

Parameters

*st	-> string que contem a linha toda
-----	-----------------------------------

Attention

Assertivas entradas - pelos clientes:

Returns

1: linha esta vazia "em branco" e 0: nao esta em branco

Precondition

st != NULL, a string deve existir

Attention

Assertiva Saida - pelo Servidor:

Postcondition

Se so houver espaco ou tab retorna 1, ou
retorna 1 caso contrario

2.2.2.7 nao_eh_vazio_indice()

```
int nao_eh_vazio_indice (
    const char * st )
```

Dado uma linha verifica qual o primeiro char que nao eh nulo "" ou tab.

Parameters

<code>*st</code>	-> string que contem a linha toda
------------------	-----------------------------------

Returns

-1 caso nao seja encontrada, ou retorna o indice que foi encontrado algum char

Attention

Assertivas entradas - pelos Clientes:

Precondition

st != NULL, a string deve existir e deve ser string

Attention

Assertiva Saida - pelo Servidor:

Postcondition

retorna -1 caso nao tenha, ou
retorna o indice do primeiro elemento diferente de tab ou espaco

Index

- abrir_arquivo
 - contador.cpp, [4](#)
 - contador.hpp, [9](#)
- conta_comments
 - contador.cpp, [4](#)
 - contador.hpp, [10](#)
- conta_final
 - contador.cpp, [5](#)
 - contador.hpp, [11](#)
- conta_linhas_branco
 - contador.cpp, [6](#)
 - contador.hpp, [11](#)
- conta_total_linhas
 - contador.cpp, [6](#)
 - contador.hpp, [12](#)
- contador.cpp, [3](#)
 - abrir_arquivo, [4](#)
 - conta_comments, [4](#)
 - conta_final, [5](#)
 - conta_linhas_branco, [6](#)
 - conta_total_linhas, [6](#)
 - eh_vazio, [7](#)
 - nao_eh_vazio_indice, [8](#)
- contador.hpp, [9](#)
 - abrir_arquivo, [9](#)
 - conta_comments, [10](#)
 - conta_final, [11](#)
 - conta_linhas_branco, [11](#)
 - conta_total_linhas, [12](#)
 - eh_vazio, [13](#)
 - nao_eh_vazio_indice, [13](#)
- eh_vazio
 - contador.cpp, [7](#)
 - contador.hpp, [13](#)
- nao_eh_vazio_indice
 - contador.cpp, [8](#)
 - contador.hpp, [13](#)