

Contador_de_linhas

Gerado por Doxygen 1.8.14

Sumário

1	README	1
2	Índice dos Arquivos	3
2.1	Lista de Arquivos	3
3	Arquivos	5
3.1	Referência do Arquivo include/reader.hpp	5
3.1.1	Descrição detalhada	5
3.1.2	Funções	6
3.1.2.1	count_coment()	6
3.1.2.2	count_doc_lines()	8
3.1.2.3	readdoc()	8
3.2	Referência do Arquivo Source/Principal.cpp	9
3.2.1	Descrição detalhada	10
3.2.2	Funções	10
3.2.2.1	main()	10
3.3	Referência do Arquivo Source/reader.cpp	11
3.3.1	Descrição detalhada	11
3.3.2	Funções	11
3.3.2.1	count_coment()	12
3.3.2.2	count_doc_lines()	13
3.3.2.3	readdoc()	14
3.4	Referência do Arquivo Test/Test_reader.cpp	15
3.4.1	Descrição detalhada	15
	Sumário	17

Capítulo 1

README

Capítulo 2

Índice dos Arquivos

2.1 Lista de Arquivos

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

include/ reader.hpp	
Arquivo de includes	5
Source/ Principal.cpp	9
Source/ reader.cpp	11
Source/Sample/ program.cpp	??
Test/ Test_reader.cpp	15

Capítulo 3

Arquivos

3.1 Referência do Arquivo include/reader.hpp

Arquivo de includes.

```
#include <iostream>
#include <fstream>
#include <boost/algorithm/string.hpp>
```

Funções

- int [readdoc](#) (fstream *inFile, string readFile)
Função de leitura de arquivo.
- int [count_coment](#) (fstream *inFile, int &line)
Contagem de comentários ou linhas em branco.
- int [count_doc_lines](#) (fstream *inFile, int &line)
Contagem de todas as linhas do documento.

3.1.1 Descrição detalhada

Arquivo de includes.

Autor

Luís Eduardo (lightguy875@github.com)

Versão

0.1

Data

2018-10-06

Copyright

Copyright (c) 2018

3.1.2 Funções

3.1.2.1 count_coment()

```
int count_coment (
    fstream * inFile,
    int & line )
```

Contagem de comentários ou linhas em branco.

Parâmetros

<i>inFile</i>	
<i>line</i>	

Retorna

int

Contagem de comentários ou linhas em branco.

Parâmetros

<i>inFile</i>	
<i>line</i>	

Retorna

int

variáveis necessárias para analisar os comentários do arquivo

Definição na linha 76 do arquivo reader.cpp.

```
77 {
78     line = 0;
79     int longComment, shortComment, substringPointer;
80     string lineGetter;
81     string lineGetterAux;
82
83     if (inFile->is_open())
84     {
85         while (!inFile->eof())
86         {
87             getline(*inFile, lineGetter);
88             lineGetterAux = lineGetter;
89             boost::erase_all(lineGetterAux, " "); // Remove espaços em branco do programa
90             // procura se existe uma linha comentada dentro do arquivo principal
91             longComment = lineGetter.find("/*");
92             shortComment = lineGetter.find("//");
93             //exposição dos 4 casos possíveis de se encontrar ou não um comentário
94
95             // 1º caso existe /* e //
96             if (longComment != -1 && shortComment != -1)
97             {
98                 // Se // vem antes de /*
```

```

103         if (shortComment < longComment)
104         {
105             line++;
106             substringPointer = lineGetter.find("\\");
107             if (substringPointer != -1)
108             {
109                 while (substringPointer != -1)
110                 {
111                     line++;
112                     getline(*inFile, lineGetter);
113                     substringPointer = lineGetter.find("\\");
114                 }
115             }
116         }
117         // Se /* vem antes de //
118         else if (shortComment > longComment)
119         {
120             line++;
121             substringPointer = lineGetter.find("*/");
122             if (substringPointer == -1)
123             {
124                 while (substringPointer == -1)
125                 {
126                     line++;
127                     getline(*inFile, lineGetter);
128                     substringPointer = lineGetter.find("*/");
129                 }
130             }
131         }
132     }
133     // 2º caso existe /* mas não //
134     else if (longComment != -1 && shortComment == -1)
135     {
136         line++;
137         substringPointer = lineGetter.find("*/");
138         if (substringPointer == -1)
139         {
140             while (substringPointer == -1)
141             {
142                 line++;
143                 getline(*inFile, lineGetter);
144                 substringPointer = lineGetter.find("*/");
145             }
146         }
147     }
148     // 3º caso existe // mas não /*
149     else if (longComment == -1 && shortComment != -1)
150     {
151         line++;
152         // Procura por barra invertidas para encontrar comentários em linhas novas se já houve
153         //
154         substringPointer = lineGetter.find("\\");
155         if (substringPointer != -1)
156         {
157             while (substringPointer != -1)
158             {
159                 line++;
160                 getline(*inFile, lineGetter);
161                 substringPointer = lineGetter.find("\\");
162             }
163         }
164     }
165 }
166 }
167 // 4º caso se existe linhas vazias
168 else if (lineGetterAux.size() == 0)
169 {
170     line++;
171 }
172 }
173 // Retorno de controle se o arquivo foi lido
174 return 0;
175 }
176 else
177 {
178     // Retorno de controle se o arquivo não foi lido
179     return -1;
180 }
181 }

```

3.1.2.2 count_doc_lines()

```
int count_doc_lines (
    fstream * inFile,
    int & line )
```

Contagem de todas as linhas do documento.

Parâmetros

<i>inFile</i>	
<i>line</i>	

Retorna

int

Contagem de todas as linhas do documento.

Parâmetros

<i>inFile</i>	
<i>line</i>	

Retorna

int

Variáveis para analisar o arquivo

Definição na linha 43 do arquivo reader.cpp.

```
44 {
49     line = 0;
50     string a;
51     if (inFile->is_open())
52     {
53         while (!inFile->eof())
54         {
55             // contagem das linhas totais do programa
56             getline(*inFile, a);
57             line++;
58         }
59         // retorno de controle se o arquivo foi aberto
60         return 0;
61     }
62
63     else
64     {
65         // retorno de controle se o arquivo não foi aberto
66         return -1;
67     }
68 }
```

3.1.2.3 readdoc()

```
int readdoc (
    fstream * inFile,
    string stringfile )
```

Função de leitura de arquivo.

Parâmetros

<i>inFile</i>	
<i>readFile</i>	

Retorna

int

Função de leitura de arquivo.

Parâmetros

<i>inFile</i>	
<i>stringfile</i>	

Retorna

int

Definição na linha 20 do arquivo reader.cpp.

```
21 {  
22     //@brief Abre o arquivo especificado em stringFile  
23     inFile->fstream::open(stringfile, fstream::in | fstream::out);  
24     if (inFile->is_open())  
25     {  
26         // O arquivo foi aberto com sucesso  
27         return 0;  
28     }  
29  
30     else  
31     {  
32         // O arquivo não foi aberto  
33         return -1;  
34     }  
35 }
```

3.2 Referência do Arquivo Source/Principal.cpp

```
#include "../include/reader.hpp"  
#include <iostream>
```

Funções

- int [main](#) (int argc, char const *argv[])

Função principal do programa.

3.2.1 Descrição detalhada

Autor

Luís Eduardo (lightguy875@github.com)

Versão

0.1

Data

2018-10-06

Copyright

Copyright (c) 2018

3.2.2 Funções

3.2.2.1 main()

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

Função principal do programa.

Parâmetros

<i>argc</i>	
<i>argv</i>	

Retorna

int

Definição na linha 21 do arquivo Principal.cpp.

```
22 {
23     int inutializedLines = 0;
24
25     int alldocLines = 0;
26
27     int utilizedLines = 0;
28
29     fstream inFile;
30     // Definição do arquivo a ser lido
31     string readFile = "Sample/program.cpp";
32     // Leitura do arquivo
```

```
33     readdoc(&inFile, readFile);
34
35     count_doc_lines(&inFile, alldocLines);
36     // Retorno para a primeira linha do arquivo principal
37     inFile.clear();
38
39     inFile.seekg(0, ios::beg);
40     // Contagem de linhas comentadas ou em branco
41     count_coment(&inFile, inutializedLines);
42
43     utilizedLines = alldocLines - inutializedLines;
44     // Exposição de dados coletados
45     cout << "A quantidade de linhas totais do programa são:" << alldocLines << endl;
46
47     cout << "A quantidade de linhas de código efetivo são:" << utilizedLines << endl;
48
49     cout << "A quantidade de linhas comentadas ou em branco são:" << inutializedLines << endl;
50
51     return 0;
52 }
```

3.3 Referência do Arquivo Source/reader.cpp

```
#include "../include/reader.hpp"
```

Funções

- int `readdoc` (fstream *inFile, string stringfile)
Função de leitura do arquivo.
- int `count_doc_lines` (fstream *inFile, int &line)
Função que conta todas as linhas do programa.
- int `count_coment` (fstream *inFile, int &line)
Função que conta todos os comentários do programa juntamente com quebra de linha.

3.3.1 Descrição detalhada

Autor

Luís Eduardo (lightguy875@github.com)

Versão

0.1

Data

2018-10-06

Copyright

Copyright (c) 2018

3.3.2 Funções

3.3.2.1 count_coment()

```
int count_coment (
    fstream * inFile,
    int & line )
```

Função que conta todos os comentários do programa juntamente com quebra de linha.

Contagem de comentários ou linhas em branco.

Parâmetros

<i>inFile</i>	
<i>line</i>	

Retorna

int

variáveis necessárias para analisar os comentários do arquivo

Definição na linha 76 do arquivo reader.cpp.

```
77 {
78     line = 0;
79     int longComment, shortComment, substringPointer;
80     string lineGetter;
81     string lineGetterAux;
82
83     if (inFile->is_open())
84     {
85         while (!inFile->eof())
86         {
87             getline(*inFile, lineGetter);
88             lineGetterAux = lineGetter;
89             boost::erase_all(lineGetterAux, " "); // Remove espaços em branco do programa
90             // procura se existe uma linha comentada dentro do arquivo principal
91             longComment = lineGetter.find("/*");
92             shortComment = lineGetter.find("//");
93             //exposição dos 4 casos possíveis de se encontrar ou não um comentário
94
95             // 1º caso existe /* e //
96             if (longComment != -1 && shortComment != -1)
97             {
98                 // Se // vem antes de /*
99                 if (shortComment < longComment)
100                 {
101                     line++;
102                     substringPointer = lineGetter.find("\\");
103                     if (substringPointer != -1)
104                     {
105                         while (substringPointer != -1)
106                         {
107                             line++;
108                             getline(*inFile, lineGetter);
109                             substringPointer = lineGetter.find("\\");
110                         }
111                     }
112                 }
113                 // Se /* vem antes de //
114                 else if (shortComment > longComment)
115                 {
116                     line++;
117                     substringPointer = lineGetter.find("*/");
118                     if (substringPointer == -1)
119                     {
120                         while (substringPointer == -1)
121                         {
122                             line++;
123                             getline(*inFile, lineGetter);
124                             substringPointer = lineGetter.find("*/");
125                         }
126                     }
127                 }
128             }
129         }
130     }
```



```

129         }
130     }
131 }
132 }
133 // 2º caso existe /* mas não //
134 else if (longComment != -1 && shortComment == -1)
135 {
136     line++;
137     substringPointer = lineGetter.find("*/");
138     if (substringPointer == -1)
139     {
140         while (substringPointer == -1)
141         {
142             line++;
143             getline(*inFile, lineGetter);
144             substringPointer = lineGetter.find("*/");
145         }
146     }
147 }
148 // 3º caso existe // mas não /*
149 else if (longComment == -1 && shortComment != -1)
150 {
151     line++;
152     // Procura por barra invertidas para encontrar comentários em linhas novas se já houve
153     //
154     substringPointer = lineGetter.find("\\");
155     if (substringPointer != -1)
156     {
157         while (substringPointer != -1)
158         {
159             line++;
160             getline(*inFile, lineGetter);
161             substringPointer = lineGetter.find("\\");
162         }
163     }
164 }
165 }
166 // 4º caso se existe linhas vazias
167 else if (lineGetterAux.size() == 0)
168 {
169     line++;
170 }
171 }
172 // Retorno de controle se o arquivo foi lido
173 return 0;
174 }
175 else
176 {
177     // Retorno de controle se o arquivo não foi lido
178     return -1;
179 }
180 }
181 }

```

3.3.2.2 count_doc_lines()

```

int count_doc_lines (
    fstream * inFile,
    int & line )

```

Função que conta todas as linhas do programa.

Contagem de todas as linhas do documento.

Parâmetros

<i>inFile</i>	
<i>line</i>	

Retorna

int

Variáveis para analisar o arquivo

Definição na linha 43 do arquivo reader.cpp.

```
44 {  
49     line = 0;  
50     string a;  
51     if (inFile->is_open())  
52     {  
53         while (!inFile->eof())  
54         {  
55             // contagem das linhas totais do programa  
56             getline(*inFile, a);  
57             line++;  
58         }  
59         // retorno de controle se o arquivo foi aberto  
60         return 0;  
61     }  
62     else  
63     {  
64         // retorno de controle se o arquivo não foi aberto  
65         return -1;  
66     }  
67 }  
68 }
```

3.3.2.3 readdoc()

```
int readdoc (  
    fstream * inFile,  
    string stringfile )
```

Função de leitura do arquivo.

Função de leitura de arquivo.

Parâmetros

<i>inFile</i>	
<i>stringfile</i>	

Retorna

int

Definição na linha 20 do arquivo reader.cpp.

```
21 {  
22     // @brief Abre o arquivo especificado em stringFile  
23     inFile->fstream::open(stringfile, fstream::in | fstream::out);  
24     if (inFile->is_open())  
25     {  
26         // O arquivo foi aberto com sucesso  
27         return 0;  
28     }
```

```
29
30     else
31     {
32         // O arquivo não foi aberto
33         return -1;
34     }
35 }
```

3.4 Referência do Arquivo Test/Test_reader.cpp

```
#include "../include/reader.hpp"
#include <gtest/gtest.h>
```

Funções

- **TEST** (Reading_file_successfully, reading)
- **TEST** (GET_ALL_DOC_LINES, all_lines_count)
- **TEST** (find_comment_in_a_file, commentary)
- int **main** (int argc, char *argv[])

3.4.1 Descrição detalhada

Autor

Luís Eduardo (lightguy875@github.com)

Versão

0.1

Data

2018-10-06

Copyright

Copyright (c) 2018

Índice Remissivo

- count_coment
 - reader.cpp, [11](#)
 - reader.hpp, [6](#)
- count_doc_lines
 - reader.cpp, [13](#)
 - reader.hpp, [7](#)
- include/reader.hpp, [5](#)
- main
 - Principal.cpp, [10](#)
- Principal.cpp
 - main, [10](#)
- readdoc
 - reader.cpp, [14](#)
 - reader.hpp, [8](#)
- reader.cpp
 - count_coment, [11](#)
 - count_doc_lines, [13](#)
 - readdoc, [14](#)
- reader.hpp
 - count_coment, [6](#)
 - count_doc_lines, [7](#)
 - readdoc, [8](#)
- Source/Principal.cpp, [9](#)
- Source/reader.cpp, [11](#)
- Test/Test_reader.cpp, [15](#)