Contador_de_linhas

Gerado por Doxygen 1.8.14

Sumário

Capítulo 1

Contador de linhas de código c/c++ com framework de teste gtest

Observação

O arquivo Principal apenas roda o programa.cpp que está na pasta Sample que serve como teste para a aplicação principal

Como compilar o software e o arquivo de teste

Para compilar o software e necessário estar na pasta Source e usar o seguinte comando:

```
$ make
```

Para compilar o arquivo de teste do software é necessário rodar o seguinte comando:

```
$make test
```

Como rodar o programa principal

```
$make run
```

Como rodar os testes e analisar a cobertura

Ainda na pasta de Source e necessário executar o seguinte código:

```
$ make run_test
```

Para ver a cobertura de testes

```
$ make gcov
```

Como limpar o programa depois da execução

Para limpar a compilação é necessário rodar o seguinte comando:

```
$ make clean
```

Para limpar o arquivo de cobertura de testes:

```
$ make clean_coverage
```

Para limpar todos os arquivos compilados juntamente com os de cobertura

```
$ make clean_all
```

Arquivos de documentação

Site html

Para acessar o site contendo as informações do programa basta acessar o arquivo index.html a abri-lo com um navegador da pasta html contida dentro de cada pasta:

```
$ html/index.html
```

Arquivo pdf

Para acessar o arquivo pdf basta entrar na pasta latex e abrir o arquivo refman contendo toda a documentação em arquivo latex:

```
$ latex/refman.pdf
```

Capítulo 2

Índice dos Arquivos

2.1 Lista de Arquivos

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

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

inFile	
line	

Retorna

int

Contagem de comentários ou linhas em branco.

Parâmetros

inFile	
line	

Retorna

int

variáveis necessárias para analisar os comentários do arquivo

Definição na linha 76 do arquivo reader.cpp.

```
77 {
82     line = 0;
83     int longComment, shortComment, substringPointer;
84     string lineGetter;
85     string lineGetterAux;
86 
87     if (inFile->is_open())
88     {
89         while (!inFile->eof())
90         {
91             getline(*inFile, lineGetter);
92             lineGetterAux = lineGetter;
93             boost::erase_all(lineGetterAux, " "); // Remove espaços em branco do programa
94             // procura se existe uma linha comentada dentro do arquivo principal
95             longComment = lineGetter.find("/*");
96             shortComment = lineGetter.find("//");
97             //exposição dos 4 casos possíveis de se encontrar ou não um comentário
98 
99             // 1° caso existe /* e //
100             if (longComment != -1 && shortComment != -1)
101             {
102                 // 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             // 2° 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             // 3° caso existe // mas não /*
149             else if (longComment == -1 && shortComment != -1)
150             {
151 
152                     line++;
153                     // Procura por barra invertidas para encontrar comentários em linhas novas se ja ouve
    //
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             // 4° 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

inFile	
line	

Retorna

int

Contagem de todas as linhas do documento.

Parâmetros

inFile	
line	

Retorna

int

Variáveis para analisar o aquivo

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

inFile	
readFile	

Retorna

int

Função de leitura de arquivo.

Parâmetros

inFile	
stringfile	

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

3.2.2 Funções

3.2.2.1 main()

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

Função principal do programa.

Parâmetros

argc	
argv	

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     // Expoxição 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

inFile	
line	

Retorna

int

variáveis necessárias para analisar os comentários do arquivo

Definição na linha 76 do arquivo reader.cpp.

```
77 {
82     line = 0;
83     int longComment, shortComment, substringPointer;
84     string lineGetter;
85     string lineGetterAux;
86 
87     if (inFile->is_open())
88     {
89         while (!inFile->eof())
90         {
91             getline(*inFile, lineGetter);
92             lineGetterAux = lineGetter;
93             boost::erase_all(lineGetterAux, " "); // Remove espaços em branco do programa
94             // procura se existe uma linha comentada dentro do arquivo principal
95             longComment = lineGetter.find("/*");
96             shortComment = lineGetter.find("//");
97             //exposição dos 4 casos possíveis de se encontrar ou não um comentário
98 
99             // 1° caso existe /* e //
100             if (longComment != -1 && shortComment != -1)
101             {
102                 // 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             // 2° 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             // 3° caso existe // mas não /*
149             else if (longComment == -1 && shortComment != -1)
150             {
151 
152                     line++;
153                     // Procura por barra invertidas para encontrar comentários em linhas novas se ja ouve
    //
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             // 4° 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 }
```

3.3.2.2 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

inFile	
line	

Retorna

int

Variáveis para analisar o aquivo

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 }
```

3.3.2.3 readdoc()

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

Função de leitura do arquivo.

Função de leitura de arquivo.

Parâmetros

inFile	
stringfile	

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)

  Teste da função readdoc.
- TEST (GET_ALL_DOC_LINES, all_lines_count)

  Teste da função de leitura de todas as linhas.
- TEST (find_comment_in_a_file, commenttary)

  Teste da função de leitura de comentários e linhas em branco.
- 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