Count all Alphabetics Characters Existent in the String

acarlstein.com/

Posted by Alejandro G. Carlstein Ramos Mejia on October 15, 2010 November 12, 2010 About Programming / Algorithms / C++

Process input file by Count all alphabetic characters existent in the string.

I would advice to compile them in Linux as I did.

NOTIFICATION: These examples are provided for educational purposes. Using this code is under your own responsibility and risk. The code is given 'as is'. I do not take responsibilities of how they are used.

StreamOperation.h:

```
#ifndef STREAM_OPERATION_H
#define STREAM_OPERATION_H

#include <iostream>  // for cout

#include <fstream>  // for ifstream, ofstream

#include <string>       // for String
#include <cctype>       // for classify and transform individual characters

using namespace std;

/**
 * author: Alejandro G. Carlstein
 * Class: StreamOperation
 * Description: This class will read a textfile and produce an output
 *    in another text file. The other text file will contain
 *     a copy of the input file but all blank lines will be
 *    removed, the lines are going to be numbered, all
 *              semicolons will be replaces with the string 'SEMI-COLON',
 *    print the number of lines removed at the second to last
 *              line, and finally print the number of alphabetic characters
 */
class StreamOperation {

private:

 bool doRemoveBlankLines;

 bool doReplaceAllStrings;

 bool doCountLines;

 int numLinesRemoved;
```

```
 int numAlphaCharacters;

 string oldStr;

 string newStr;

 void replaceAll(string& str,
     string oldStr,
     string newStr);

 int countAlphaCharacters(string str);

 void copyStream(ifstream& fin,
       ofstream& fout);

public:

 // Default constructor
 StreamOperation(void);

 // * Get Methods *

 // Get the number of lines removed
 int getNumberLinesRemoved();

 // Get the number of alphabetic character
 int getNumberAlphaCharacters();

 // * Set Methods *

 // * Print Methods *

 // Print the number of blank lines removed
 void printNumberLinesRemoved(ofstream& output);

 // Print the number of alphabetic characters
 void printNumberAlphaCharacters(ofstream& output);

 // * Process Methods *

 // Copy the content from an input stream to an output stream
 void copy(ifstream& input,
     ofstream& output);

 // Copy the content from an input stream to an output stream.
    // This method can remove all the blank lines in the output
    // stream when copying.
 void copy(ifstream& input,
     ofstream& output,
     bool removeBlankLines);

 // Copy the content from an input stream to an output stream.
```

```
 // This method can remove all the blank lines in the output
  // stream when copying.
 // This method can number all the lines in the output stream.
 void copy(ifstream& input,
     ofstream& output,
     bool removeBlankLines,
              bool numberLines);

 // Copy the content from an input stream to an output stream.
 // This method can replace all old strings for a new string
 // This method can remove all the blank lines in the output
  // stream when copying.
 // This method can number all the lines in the output stream.
 void copy(ifstream& input,
     ofstream& output,
              string oldString,
     string newString,
     bool removeBlankLines,
     bool numberLines);

    // Default destructor
 ~StreamOperation(void);
};

#endif
```

StreamOperation.cpp:

```
/**
 * author: Alejandro G. Carlstein
 * Class: StreamOperation
 * Description: This class will read a textfile and produce an output
 *  in another text file. The other text file will contain
 *   a copy of the input file but all blank lines will be
 *  removed, the lines are going to be numbered, all
 *              semicolons will be replaces with the string 'SEMI-COLON',
 *  print the number of lines removed at the second to last
 *              line, and finally print the number of alphabetic characters
 */

#include 'StreamOperation.h'

/**
 * Public Methods
 */

/**
 * StreamOperation
 * @description: Default Constructor
 */
StreamOperation::StreamOperation(void){

 numLinesRemoved = 0;
```

```
 numAlphaCharacters = 0;

 doRemoveBlankLines = false;

 doReplaceAllStrings = false;

 doCountLines = false;

 oldStr = '';

 newStr = '';

}

// * Get Methods *

/**
 * getNumberLinesRemoved
 * @description: Get the number of lines removed
 * @return: integer
*/
int StreamOperation::getNumberLinesRemoved(){
 return numLinesRemoved;
}

/**
 * getNumberAlphaCharacters
 * @description: Get the number of alphabetic character
 * @return: integer
 */
int StreamOperation::getNumberAlphaCharacters(){
 return numAlphaCharacters;
}

// * Set Methods *

// * Print Methods *

// Print the number of blank lines removed
void StreamOperation:😜rintNumberLinesRemoved(ofstream& output){
 if (output.is_open()){
  output << numLinesRemoved;
 } else {
   cerr << '[X] Error: Program cannot write file!' << endl
     << 'Exit program!' << endl;
 }
}

// Print the number of alphabetic characters
void StreamOperation:😜rintNumberAlphaCharacters(ofstream& output){
 if (output.is_open()){
  output << numAlphaCharacters;
```

```
 } else {
   cerr << '[X] Error: Program cannot write file!' << endl
     << 'Exit program!' << endl;
 }
}

// * Process Methods *

/**
 * copy
 * @description: Copy the content from an input stream to an output stream
 * @param: input, output
 */
void StreamOperation::copy(ifstream& input,
         ofstream& output){

 doRemoveBlankLines = false;

 doReplaceAllStrings = false;

 doCountLines = false;

 copyStream(input, output);

}

/**
 * copy
 * @description: Copy the content from an input stream to an output stream.
 *               This method can remove all the blank lines in the output
 *     stream when copying.
 *
 * @param: input, output, removeBlankLines
 */
void StreamOperation::copy(ifstream& input,
         ofstream& output,
         bool removeBlankLines){

 doReplaceAllStrings = false;

 doCountLines = false;

 doRemoveBlankLines = removeBlankLines;

 copyStream(input, output);

}

/**
 * copy
 * @description: Copy the content from an input stream to an output stream.
 *               This method can remove all the blank lines in the output
 *     stream when copying.
```

```
 *               This method can number all the lines in the output stream.
 * @param: input, output, removeBlankLines, numberLines
 */
void StreamOperation::copy(ifstream& input,
         ofstream& output,
         bool removeBlankLines,
         bool numberLines){

 doReplaceAllStrings = false;

 doCountLines = numberLines;

 doRemoveBlankLines = removeBlankLines;

 copyStream(input, output);

}

/**
 * copy
 * @description: Copy the content from an input stream to an output stream.
 *     This method can replace all old strings for a new string
 *               This method can remove all the blank lines in the output
 *     stream when copying.
 *               This method can number all the lines in the output stream.
 * @param: input, output, oldstring, new string, removeBlankLines, numberLines
 */
void StreamOperation::copy(ifstream& input,
           ofstream& output,
                  string oldString,
          string newString,
           bool removeBlankLines,
         bool numberLines){

 doReplaceAllStrings = true;

 oldStr = oldString;

 newStr = newString;

 doCountLines = numberLines;

 doRemoveBlankLines = removeBlankLines;

 copyStream(input, output);

}

/**
 * StreamOperation
 * @description: Default Destructor
 */
StreamOperation::~StreamOperation(void){
```

```
};

/**
 * Private Methods
 */

/**
 * replaceAll
 * @description: This method will remove all substrings for a new substring
 *               inside the string
 * @param: str, oldStr, newStr
 */
void StreamOperation::replaceAll(string& str,
              string oldStr,
             string newStr){

 // The method find return the unsigned int string::npos
    // if substring not found. Therefore, string::size_type
 // type is used
 string::size_type position = 0;

 // Until the end of the string is reached, search for every
    // string that maches the old string and replace it with
 // the new string.
 while((position = str.find(oldStr, position)) != string::npos){
  str.replace(position,
     oldStr.length(),
     newStr);
  position++;
 }
}

/**
 * countAlphaCharacters
 * @description: Count all alphabetics characters existent in the string
 * @param: str
 * @return: integer
 */
int StreamOperation::countAlphaCharacters(string str){

 int countAlpha = 0;

 // Go thought the whole string, counting all
 // the alphabetic characters
 for (int position = 0;
   position < str.length();
   position++){

  countAlpha += (isalpha(str[position]) ? 1 : 0);

 }

 return countAlpha;
```

```
}

/**
 * copyStream
 * @description: This method copy the content from an input stream to
 *               an output stream.
 *     Base on the flags doRemoveBlankLines, doCountLines, and
 *               doReplaceAllStrings:
 *     This method can replace all old strings for a new string
 *               This method can remove all the blank lines in the output
 *     stream when copying.
 *               This method can number all the lines in the output stream.
 * @param: fin, fout
 */
void StreamOperation::copyStream(ifstream& fin,
            ofstream& fout){
 int lineCounter;

 string strBuffer;

 numLinesRemoved = 0;

 numAlphaCharacters = 0;

 lineCounter = 1;

 // Check if input and output stream can be open
 if (fin.is_open()){

  if (fout.is_open()){

   //Read one line at the time as a string until eof
   while(!fin.eof()){

    getline(fin, strBuffer);

    // If the string is empty and doRemoveBlankLines
    // is true, count the string as as a blank line
    // else process the string
    if (strBuffer.empty() && doRemoveBlankLines){

     numLinesRemoved++;

    }else{

     // Count the alphabetic character of the string
     numAlphaCharacters += countAlphaCharacters(strBuffer);

     // Replace all semicolons with the string SEMICOLON
     if (doReplaceAllStrings)
      replaceAll(strBuffer, oldStr, newStr);

     // Add a number to each line if doCountLines is true
```

```
     if (doCountLines)
      fout << lineCounter++ << ' ';

     fout << strBuffer << endl;

    }

   }

  } else {
   cerr << '[X] Error: Program cannot write file!' << endl
     << 'Exit program!' << endl;
  }

 }else{
  cerr << '[X] Error: Program cannot read file!' << endl
    << 'Exit Program!' << endl;
 }

}
```

processFile.cpp:

```
/**
 * Author: Alejandro G. Carlstein
 * Description: Use the file StreamOperations.cpp as an input file
 *    and process it using StreamOperation class
 */

#include <iostream>
#include <fstream>
#include 'StreamOperation.h'

static const string INPUT_FILE = 'StreamOperation.cpp';
static const string OUTPUT_FILE = 'Output.txt';
static const string SEMI_COLON = ';';
static const string STR_SEMI_COLON = 'SEMI-COLON';

/**
 * Main function
 * @param: argc, argv
 */

int main(int argc, char *argv[]){

 ifstream inputFile;
 ofstream outputFile;

 // If the program is executed with two parameters (file 1  and file 2)
    // used these parameters as input file and output file
 // If the program is executed without parameters use default files
    // If the program is executed with the -h parameter display help
 // If the program get more than two parameters or
```

```
   // wrong key display help
 if (argc == 1 || argc == 3){

  if (argc == 1){

   inputFile.open(INPUT_FILE.data());

   outputFile.open(OUTPUT_FILE.data());

  }else{

   inputFile.open(argv[1]);

   outputFile.open(argv[2]);
  }

  StreamOperation StrOp;

  // Copy the content from the input file to the output file
     // In the process, replace the ; with string SEMI-COLON,
  // remove the blank lines and number all the lines
  StrOp.copy(inputFile,
       outputFile,
       SEMI_COLON,
       STR_SEMI_COLON,
       true,
                true);

  outputFile << 'Lines Removed: '
       << StrOp.getNumberLinesRemoved() << endl;

  outputFile << 'Alphabetic Characters: '
       << StrOp.getNumberAlphaCharacters();

  inputFile.close();

  outputFile.close();

 } else {
  cout << argv[0] << ' input_file output_file ' << endl;
 }

 return 0;
}
```

input.txt:

```
23
This is
An example

of things;
That we can input;

;; aaa ;
```

output.txt:

```
1 23
2 This is
3 An example
4 of thingsSEMI-COLON
5 That we can inputSEMI-COLON
6 SEMI-COLONSEMI-COLON aaa SEMI-COLON
```

If you encounter any problems or errors, please let me know by providing an example of the code, input, output, and an explanation. Thanks.

```
/**
* author: Alejandro G. Carlstein
* Course: CS 240
* Class: StreamOperation
* Description: This class will read a textfile and produce an output
*       in another text file. The other text file will contain
*        a copy of the input file but all blank lines will be
*       removed, the lines are going to be numbered, all
*            semicolons will be replaces with the string 'SEMI-COLON',
*       print the number of lines removed at the second to last
*            line, and finally print the number of alphabetic characters
*/
#include "StreamOperation.h"

/**
* Public Methods
*/

/**
* StreamOperation
* @description: Default Constructor
*/
StreamOperation::StreamOperation(void){

numLinesRemoved = 0;

numAlphaCharacters = 0;
```

```
doRemoveBlankLines = false;

doReplaceAllStrings = false;

doCountLines = false;

oldStr = “”;

newStr = “”;

}

// * Get Methods *

/**
* getNumberLinesRemoved
* @description: Get the number of lines removed
* @return: integer
*/
int StreamOperation::getNumberLinesRemoved(){
return numLinesRemoved;
}

/**
* getNumberAlphaCharacters
* @description: Get the number of alphabetic character
* @return: integer
*/
int StreamOperation::getNumberAlphaCharacters(){
return numAlphaCharacters;
}

// * Set Methods *

// * Print Methods *

// Print the number of blank lines removed
void StreamOperation: rintNumberLinesRemoved(ofstream& output){
if (output.is_open()){
output << numLinesRemoved;
} else {
cerr << “[X] Error: Program cannot write file!” << endl
<< “Exit program!” << endl;
}
}
```

```
// Print the number of alphabetic characters
void StreamOperation: rintNumberAlphaCharacters(ofstream& output){
if (output.is_open()){
output << numAlphaCharacters;
} else {
cerr << “[X] Error: Program cannot write file!” << endl
<< “Exit program!” << endl;
}
}

// * Process Methods *

/**
* copy
* @description: Copy the content from an input stream to an output stream
* @param: input, output
*/
void StreamOperation::copy(ifstream& input,
ofstream& output){

doRemoveBlankLines = false;

doReplaceAllStrings = false;

doCountLines = false;

copyStream(input, output);

}

/**
* copy
* @description: Copy the content from an input stream to an output stream.
*               This method can remove all the blank lines in the output
*               stream when copying.
*
* @param: input, output, removeBlankLines
*/
void StreamOperation::copy(ifstream& input,
ofstream& output,
bool removeBlankLines){

doReplaceAllStrings = false;

doCountLines = false;

doRemoveBlankLines = removeBlankLines;
```

```
copyStream(input, output);

}

/**
* copy
* @description: Copy the content from an input stream to an output stream.
*              This method can remove all the blank lines in the output
*               stream when copying.
*              This method can number all the lines in the output stream.
* @param: input, output, removeBlankLines, numberLines
*/
void StreamOperation::copy(ifstream& input,
ofstream& output,
bool removeBlankLines,
bool numberLines){

doReplaceAllStrings = false;

doCountLines = numberLines;

doRemoveBlankLines = removeBlankLines;

copyStream(input, output);

}

/**
* copy
* @description: Copy the content from an input stream to an output stream.
*               This method can replace all old strings for a new string
*              This method can remove all the blank lines in the output
*               stream when copying.
*              This method can number all the lines in the output stream.
* @param: input, output, oldstring, new string, removeBlankLines, numberLines
*/
void StreamOperation::copy(ifstream& input,
ofstream& output,
string oldString,
string newString,
bool removeBlankLines,
bool numberLines){

doReplaceAllStrings = true;

oldStr = oldString;
```

```
newStr = newString;

doCountLines = numberLines;

doRemoveBlankLines = removeBlankLines;

copyStream(input, output);

}

/**
* StreamOperation
* @description: Default Destructor
*/
StreamOperation::~StreamOperation(void){
};

/**
* Private Methods
*/

/**
* replaceAll
* @description: This method will remove all substrings for a new substring
*               inside the string
* @param: str, oldStr, newStr
*/
void StreamOperation::replaceAll(string& str,
string oldStr,
string newStr){

// The method find return the unsigned int string::npos
// if substring not found. Therefore, string::size_type
// type is used
string::size_type position = 0;

// Until the end of the string is reached, search for every
// string that maches the old string and replace it with
// the new string.
while((position = str.find(oldStr, position)) != string::npos){
str.replace(position,
oldStr.length(),
newStr);
position++;
}
}
```

```
/**
* countAlphaCharacters
* @description: Count all alphabetics characters existent in the string
* @param: str
* @return: integer
*/
int StreamOperation::countAlphaCharacters(string str){

int countAlpha = 0;

// Go thought the whole string, counting all
// the alphabetic characters
for (int position = 0;
position < str.length();
position++){

countAlpha += (isalpha(str[position]) ? 1 : 0);

}

return countAlpha;
}

/**
* copyStream
* @description: This method copy the content from an input stream to
*               an output stream.
*                 Base on the flags doRemoveBlankLines, doCountLines, and
*               doReplaceAllStrings:
*                 This method can replace all old strings for a new string
*               This method can remove all the blank lines in the output
*                 stream when copying.
*               This method can number all the lines in the output stream.
* @param: fin, fout
*/
void StreamOperation::copyStream(ifstream& fin,
ofstream& fout){
int lineCounter;

string strBuffer;

numLinesRemoved = 0;

numAlphaCharacters = 0;

lineCounter = 1;
```

```
// Check if input and output stream can be open
if (fin.is_open()){

if (fout.is_open()){

//Read one line at the time as a string until eof
while(!fin.eof()){

getline(fin, strBuffer);

// If the string is empty and doRemoveBlankLines
// is true, count the string as as a blank line
// else process the string
if (strBuffer.empty() && doRemoveBlankLines){

numLinesRemoved++;

}else{

// Count the alphabetic character of the string
numAlphaCharacters += countAlphaCharacters(strBuffer);

// Replace all semicolons with the string SEMICOLON
if (doReplaceAllStrings)
replaceAll(strBuffer, oldStr, newStr);

// Add a number to each line if doCountLines is true
if (doCountLines)
fout << lineCounter++ << " ";

fout << strBuffer << endl;

}

}

} else {
cerr << "[X] Error: Program cannot write file!" << endl
<< "Exit program!" << endl;
}

}else{
cerr << "[X] Error: Program cannot read file!" << endl
<< "Exit Program!" << endl;
}

}

/**
```

```
 * author: Alejandro G. Carlstein
 * Course: CS 240
 * Class: StreamOperation
 * Description: This class will read a textfile and produce an output
 *  in another text file. The other text file will contain
 *   a copy of the input file but all blank lines will be
 *  removed, the lines are going to be numbered, all
 *              semicolons will be replaces with the string 'SEMI-COLON',
 *  print the number of lines removed at the second to last
 *              line, and finally print the number of alphabetic characters
 */

#include 'StreamOperation.h'

/**
 * Public Methods
 */

/**
 * StreamOperation
 * @description: Default Constructor
 */
StreamOperation::StreamOperation(void){

 numLinesRemoved = 0;

 numAlphaCharacters = 0;

 doRemoveBlankLines = false;

 doReplaceAllStrings = false;

 doCountLines = false;

 oldStr = '';

 newStr = '';

}

// * Get Methods *

/**
 * getNumberLinesRemoved
 * @description: Get the number of lines removed
 * @return: integer
*/
int StreamOperation::getNumberLinesRemoved(){
 return numLinesRemoved;
}

/**
 * getNumberAlphaCharacters
```

```
 * @description: Get the number of alphabetic character
 * @return: integer
 */
int StreamOperation::getNumberAlphaCharacters(){
 return numAlphaCharacters;
}

// * Set Methods *

// * Print Methods *

// Print the number of blank lines removed
void StreamOperation:😛rintNumberLinesRemoved(ofstream& output){
 if (output.is_open()){
  output << numLinesRemoved;
 } else {
   cerr << '[X] Error: Program cannot write file!' << endl
     << 'Exit program!' << endl;
 }
}

// Print the number of alphabetic characters
void StreamOperation:😛rintNumberAlphaCharacters(ofstream& output){
 if (output.is_open()){
  output << numAlphaCharacters;
 } else {
   cerr << '[X] Error: Program cannot write file!' << endl
     << 'Exit program!' << endl;
 }
}

// * Process Methods *

/**
 * copy
 * @description: Copy the content from an input stream to an output stream
 * @param: input, output
 */
void StreamOperation::copy(ifstream& input,
         ofstream& output){

 doRemoveBlankLines = false;

 doReplaceAllStrings = false;

 doCountLines = false;

 copyStream(input, output);

}

/**
 * copy
```

```
 * @description: Copy the content from an input stream to an output stream.
 *               This method can remove all the blank lines in the output
 *     stream when copying.
 *
 * @param: input, output, removeBlankLines
 */
void StreamOperation::copy(ifstream& input,
         ofstream& output,
         bool removeBlankLines){

 doReplaceAllStrings = false;

 doCountLines = false;

 doRemoveBlankLines = removeBlankLines;

 copyStream(input, output);

}

/**
 * copy
 * @description: Copy the content from an input stream to an output stream.
 *               This method can remove all the blank lines in the output
 *     stream when copying.
 *               This method can number all the lines in the output stream.
 * @param: input, output, removeBlankLines, numberLines
 */
void StreamOperation::copy(ifstream& input,
         ofstream& output,
         bool removeBlankLines,
         bool numberLines){

 doReplaceAllStrings = false;

 doCountLines = numberLines;

 doRemoveBlankLines = removeBlankLines;

 copyStream(input, output);

}

/**
 * copy
 * @description: Copy the content from an input stream to an output stream.
 *     This method can replace all old strings for a new string
 *               This method can remove all the blank lines in the output
 *     stream when copying.
 *               This method can number all the lines in the output stream.
 * @param: input, output, oldstring, new string, removeBlankLines, numberLines
 */
void StreamOperation::copy(ifstream& input,
```

```
           ofstream& output,
                  string oldString,
          string newString,
           bool removeBlankLines,
         bool numberLines){

 doReplaceAllStrings = true;

 oldStr = oldString;

 newStr = newString;

 doCountLines = numberLines;

 doRemoveBlankLines = removeBlankLines;

 copyStream(input, output);

}

/**
 * StreamOperation
 * @description: Default Destructor
 */
StreamOperation::~StreamOperation(void){
};

/**
 * Private Methods
 */

/**
 * replaceAll
 * @description: This method will remove all substrings for a new substring
 *               inside the string
 * @param: str, oldStr, newStr
 */
void StreamOperation::replaceAll(string& str,
              string oldStr,
             string newStr){

 // The method find return the unsigned int string::npos
    // if substring not found. Therefore, string::size_type
 // type is used
 string::size_type position = 0;

 // Until the end of the string is reached, search for every
    // string that maches the old string and replace it with
 // the new string.
 while((position = str.find(oldStr, position)) != string::npos){
  str.replace(position,
     oldStr.length(),
     newStr);
```

```
  position++;
 }
}

/**
 * countAlphaCharacters
 * @description: Count all alphabetics characters existent in the string
 * @param: str
 * @return: integer
 */
int StreamOperation::countAlphaCharacters(string str){

 int countAlpha = 0;

 // Go thought the whole string, counting all
 // the alphabetic characters
 for (int position = 0;
   position < str.length();
   position++){

  countAlpha += (isalpha(str[position]) ? 1 : 0);

 }

 return countAlpha;
}

/**
 * copyStream
 * @description: This method copy the content from an input stream to
 *               an output stream.
 *     Base on the flags doRemoveBlankLines, doCountLines, and
 *               doReplaceAllStrings:
 *     This method can replace all old strings for a new string
 *               This method can remove all the blank lines in the output
 *     stream when copying.
 *               This method can number all the lines in the output stream.
 * @param: fin, fout
 */
void StreamOperation::copyStream(ifstream& fin,
            ofstream& fout){
 int lineCounter;

 string strBuffer;

 numLinesRemoved = 0;

 numAlphaCharacters = 0;

 lineCounter = 1;

 // Check if input and output stream can be open
 if (fin.is_open()){
```

```
  if (fout.is_open()){

   //Read one line at the time as a string until eof
   while(!fin.eof()){

    getline(fin, strBuffer);

    // If the string is empty and doRemoveBlankLines
    // is true, count the string as as a blank line
    // else process the string
    if (strBuffer.empty() && doRemoveBlankLines){

     numLinesRemoved++;

    }else{

     // Count the alphabetic character of the string
     numAlphaCharacters += countAlphaCharacters(strBuffer);

     // Replace all semicolons with the string SEMICOLON
     if (doReplaceAllStrings)
      replaceAll(strBuffer, oldStr, newStr);

     // Add a number to each line if doCountLines is true
     if (doCountLines)
      fout << lineCounter++ << ' ';

     fout << strBuffer << endl;

    }

   }

  } else {
   cerr << '[X] Error: Program cannot write file!' << endl
     << 'Exit program!' << endl;
  }

 }else{
  cerr << '[X] Error: Program cannot read file!' << endl
    << 'Exit Program!' << endl;
 }

}
```