

Luiz Felipe Léo

Felipe Claudio da Silva Santos

Terceiro Relatório – Corretor Ortográfico

1. de Maio de 2017

Sumário

[1. Introdução 3](#_Toc484991959)

[2. Implementação do Programa 3](#_Toc484991960)

[2.1. Classe perlWrapper 3](#_Toc484991961)

[2.2. Classe biblioteca 4](#_Toc484991962)

[2.3. Classe ExcecaoTamanhoPalavra 5](#_Toc484991963)

[3. Casos de Uso 6](#_Toc484991964)

[3.1. Instalação 6](#_Toc484991965)

[3.2. Inserção de texto na Biblioteca 7](#_Toc484991966)

[3.3. Exibir textos contidos na biblioteca 7](#_Toc484991970)

[3.4. Selecionar texto para exibir informações 7](#_Toc484991971)

[3.4.1. Correção de plural 9](#_Toc484991972)

[3.4.2. Correção de palavras 9](#_Toc484991973)

[3.4.3. Correção de pontuação 9](#_Toc484991974)

[3.4.4. Colocar sinônimos em repetição 10](#_Toc484991975)

[3.4.5. Correção de letras maiúscula 11](#_Toc484991976)

[3.4.6. Correção completa 11](#_Toc484991977)

[3.5. Excluir texto da biblioteca 12](#_Toc484991978)

[4. Conclusão 12](#_Toc484991979)

**Lista de Figuras**

[Figura 1- Classe PerlWrapper 4](file:///C:\Users\felip\Documents\Materias\Linguagens_de_Programacao\t3\relatorio3_FelipeClaudio_LuizFelipe.docx#_Toc484993340)

[Figura 2 - Classe biblioteca 5](file:///C:\Users\felip\Documents\Materias\Linguagens_de_Programacao\t3\relatorio3_FelipeClaudio_LuizFelipe.docx#_Toc484993341)

[Figura 3 - Classe ExcecaoTamanhoPalavra 5](file:///C:\Users\felip\Documents\Materias\Linguagens_de_Programacao\t3\relatorio3_FelipeClaudio_LuizFelipe.docx#_Toc484993342)

[Figura 4 - Execução do Makefile 6](file:///C:\Users\felip\Documents\Materias\Linguagens_de_Programacao\t3\relatorio3_FelipeClaudio_LuizFelipe.docx#_Toc484993343)

[Figura 5 - Código do makefile 6](file:///C:\Users\felip\Documents\Materias\Linguagens_de_Programacao\t3\relatorio3_FelipeClaudio_LuizFelipe.docx#_Toc484993344)

[Figura 6 - Inserir texto na biblioteca 7](file:///C:\Users\felip\Documents\Materias\Linguagens_de_Programacao\t3\relatorio3_FelipeClaudio_LuizFelipe.docx#_Toc484993345)

[Figura 7 - Exibir textos 7](file:///C:\Users\felip\Documents\Materias\Linguagens_de_Programacao\t3\relatorio3_FelipeClaudio_LuizFelipe.docx#_Toc484993346)

[Figura 8 – Seleção de texto para exibir informações 8](file:///C:\Users\felip\Documents\Materias\Linguagens_de_Programacao\t3\relatorio3_FelipeClaudio_LuizFelipe.docx#_Toc484993347)

[Figura 9 - Menu de correções 8](file:///C:\Users\felip\Documents\Materias\Linguagens_de_Programacao\t3\relatorio3_FelipeClaudio_LuizFelipe.docx#_Toc484993348)

[Figura 10 - Código da correção de plural 9](file:///C:\Users\felip\Documents\Materias\Linguagens_de_Programacao\t3\relatorio3_FelipeClaudio_LuizFelipe.docx#_Toc484993349)

[Figura 11 - Código da função de substituição de repetição 10](file:///C:\Users\felip\Documents\Materias\Linguagens_de_Programacao\t3\relatorio3_FelipeClaudio_LuizFelipe.docx#_Toc484993350)

[Figura 12 - Identificação de palavra repetida 11](file:///C:\Users\felip\Documents\Materias\Linguagens_de_Programacao\t3\relatorio3_FelipeClaudio_LuizFelipe.docx#_Toc484993351)

[Figura 13 - Saída com todos os casos corrigidos 11](file:///C:\Users\felip\Documents\Materias\Linguagens_de_Programacao\t3\relatorio3_FelipeClaudio_LuizFelipe.docx#_Toc484993352)

[Figura 14 - Entrada com todos os tipos de erros 11](file:///C:\Users\felip\Documents\Materias\Linguagens_de_Programacao\t3\relatorio3_FelipeClaudio_LuizFelipe.docx#_Toc484993353)

# Introdução

O programa desenvolvido na terceira parte do trabalho integra as funções desenvolvidas em perl durante a segunda parte do trabalho com a interface feita em c++. Além dessa integração entre essas duas linguagens, estão presentes na interface funções auxiliares ao processo de correção ortográfica. Estas funções têm como objetivo facilitar a inserção, remoção e visualização de textos salvos na biblioteca.

# Implementação do Programa

## Classe perlWrapper

O programa possui uma classe chamada*perlWrapper*, a qual tem como objetivo facilitar a troca de informações entre o programa em c++ e o programa em perl. O construtor desta classe depende das informações passadas à função *main* (argc, argv, env). Há também os métodos set e get para os arquivos de entrada e saída, visto que pode ser necessário utilizar como entrada ou saída do programa algum outro arquivo diferente dos definidos por padrão (entrada – arquivoDeEntrada.txt / saída – SaidaCorrigida.txt). Estes métodos disparam a exceção *ExceçãoTamanhoPalavra* caso o nome do arquivo de entrada seja maior que o máximo especificado na classe (100 caracteres).

Há também um método privado com o objetivo de somente auxiliar as funções *set* e *get* a copiar arquivos.



Figura - Classe PerlWrapper

## Classe biblioteca

O programa também possui uma classe chamada biblioteca, responsável pelo armazenamento dos textos, oferecendo ao usuário a possibilidade de guardar quantos textos quiser para serem corrigidos de modo persistente após a finalização do programa, salvando todos os textos em .txt e os carregando conforme solicitado, a partir de uma lista de texto guardada pelo programa. Caso o usuário tente adicionar um texto inexistente a biblioteca, o programa também pode contar com a opção de criar um novo texto na hora, que também poderá ser corrigido e será salvo junto aos demais da biblioteca.

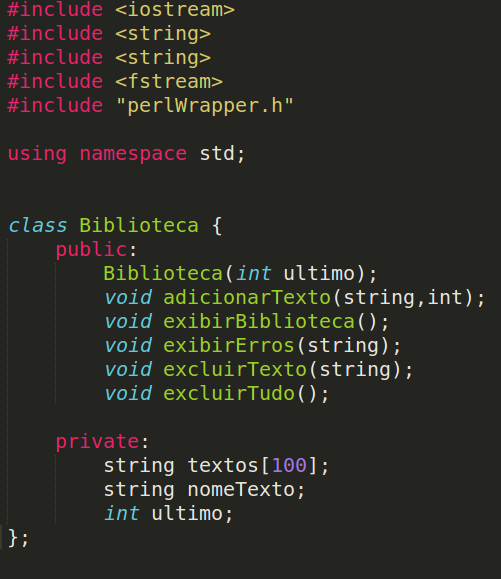
A classe possui 4 funções públicas, adicionar texto, excluir texto, exibir texto para correção e exibir biblioteca (Que exibe os títulos dos textos presentes na biblioteca e oferece um índice para selecionar o listado desejado), além de possuir 2 argumentos privados, a lista de texto, que armazena os nomes de cada txt presente na biblioteca para facilitar sua abertura futura sem a necessidade de ser informado novamente pelo usuário, e um inteiro que guarda a posição do último texto inserido na biblioteca para facilitar a ordenação e exclusão.

Figura - Classe biblioteca

## Classe ExcecaoTamanhoPalavra

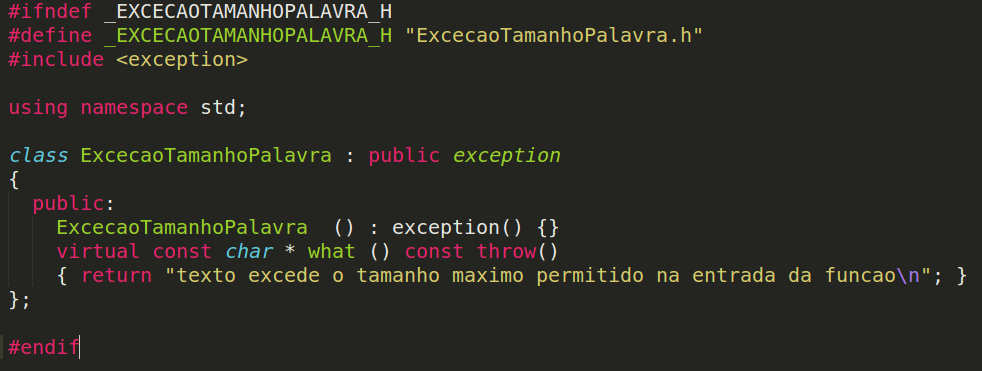
É uma exceção herdada da classe *exception* e com o método *what* sobrecarregado. Retorna um texto indicando que o tamanho da palavra é maior do que o permitido pela função. A classe é toda implementada em um único arquivo .h.

Figura - Classe ExcecaoTamanhoPalavra

# Casos de Uso

## Instalação

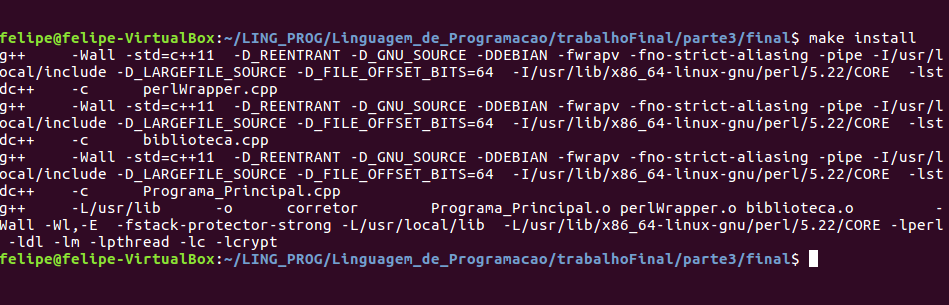
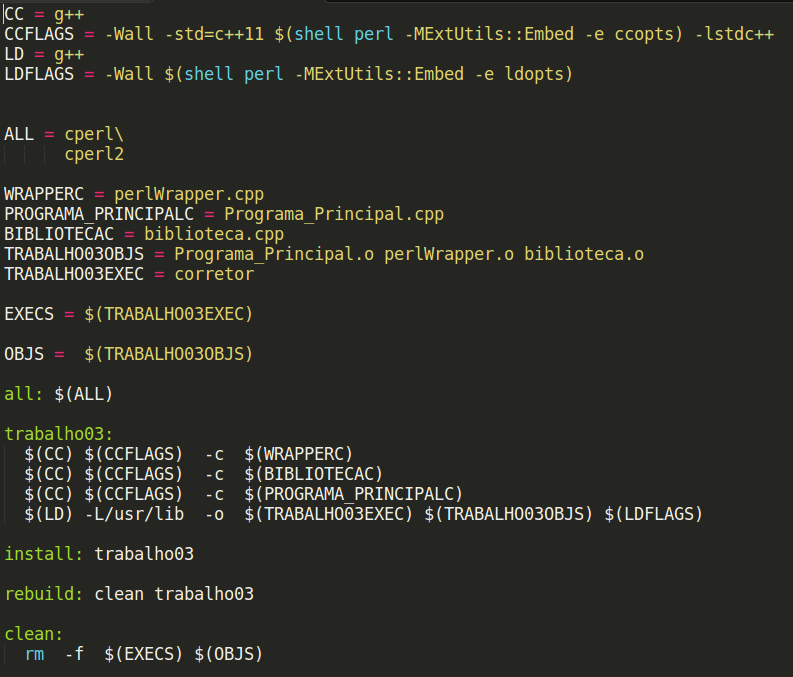
Para instalar o programa é necessário abrir a pasta onde estão todos os arquivos relacionados ao programa e executar o comando make install, conforme a figura abaixo:

Figura - Execução do Makefile

Figura - Código do makefile

O arquivo final gerado deverá se chamar “**corretor**”e será responsável por executar as funções comentadas**.**

## Inserção de texto na Biblioteca

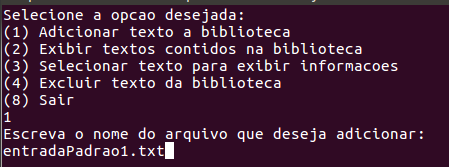
 Há duas maneiras de inserir um novo texto na biblioteca, a primeira delas serve para inserir um arquivo de texto externo que deverá ser copiado para a pasta do programa,

Figura - Inserir texto na biblioteca



Feito isso, basta selecionar a opção “1” do menu principal e inserir o nome do arquivo .txt para que o programa o inclua em sua biblioteca e ele seja habilitado para correção e funcione de forma persistente.

Caso o texto informado pelo usuário não exista na pasta principal do programa, é oferecida a opção da criação de um novo texto escrito na hora, assim o usuário poderá escrever o texto e salvar com o nome descrito anteriormente, para que o mesmo seja adicionado a biblioteca, esse novo texto também será salvo em .txt e funcionará de forma persistente como o resto da biblioteca.

## Exibir textos contidos na biblioteca

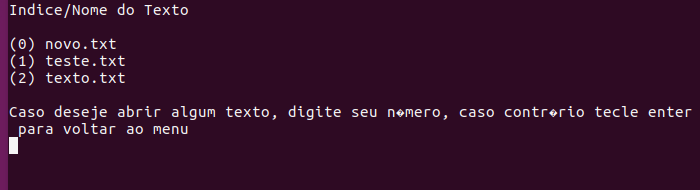
A função de exibição de texto visa facilitar o acesso e correção dos textos contidos na biblioteca. Ao ser selecionada exibe uma lista contendo o título de cada texto inserido e seu respectivo índice, caso o usuário deseje visualizar ou corrigir algum deles basta selecioná-los pelo índice e o programa irá direto para a área de exibição e correção, onde o texto cujo título foi selecionado será exibido, junto com informações de número de caracteres, e o programa entrará no menu de correção.

Figura - Exibir textos

## Selecionar texto para exibir informações

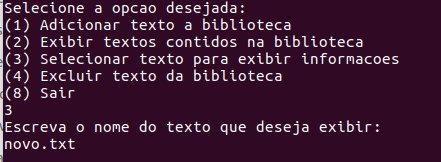
A função de exibir informações abre o texto especificado pelo usuário e exibe as informações de número de caracteres e opções de correção, ao selecionar uma opção de correção o texto será corrigido (através da interação com o Perl) e será exibida na tela a saída de texto pós correção, o texto corrigido também será salvo em forma de .txt com o nome de “correcao\_NOMEDOTEXTO”.

Figura – Seleção de texto para exibir informações

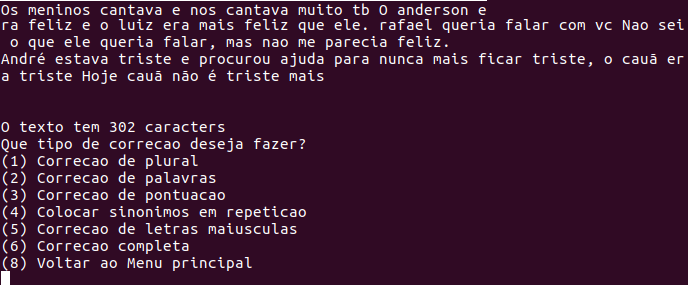


Figura - Menu de correções

As funções são as mesmas apresentadas no trabalho 2, logo serão explicadas de forma superficial.

### **Correção de plural**

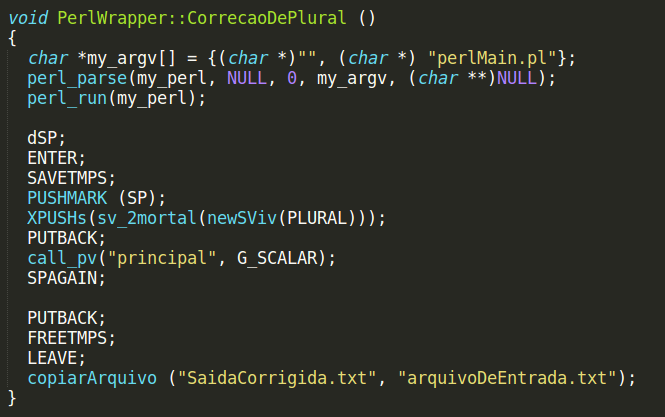
Corrige casos onde a regra do plural não é aplicada ex: nós vai.

Figura - Código da correção de plural

Os códigos para outros métodos são similares a este mostrado acima, com exceção do método que substitui palavras repetidas.

### **Correção de palavras**

Corrige palavras escritas fora do formato padrão da escrita ex: tb -> também, vc -> você.

### **Correção de pontuação**

Corrige casos onde há falta do ponto final: ex: Dormi em casa Acordei no outro dia -> Dormi em casa. Acordei no outro dia.

### **Colocar sinônimos em repetição**

Este método necessita da palavra de referência que será substituída, sendo esta passada pela linha de comando para a interface em c++ e após isto, o programa em c++ passa para o programa em perl a palavra que deverá ser substituída. A substituição é feita de forma circular, tendo como finalidade evitar novamente o problema da repetição de palavras ex: feliz feliz feliz feliz feliz -> feliz contente alegre feliz contente.

Figura - Código da função de substituição de repetição

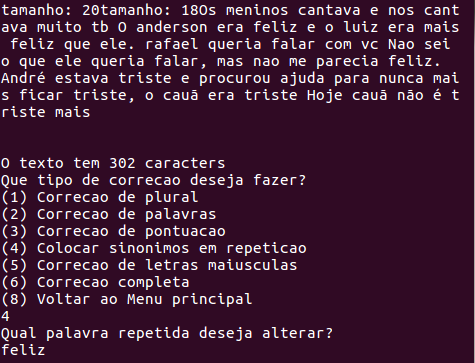


Figura - Identificação de palavra repetida

### **Correção de letras maiúscula**

Substitui a letra inicial de um substantivo próprio pela versão maiúscula desta letra ex: marcos -> Marcos

### **Correção completa**

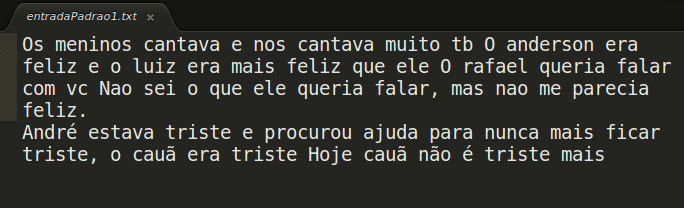
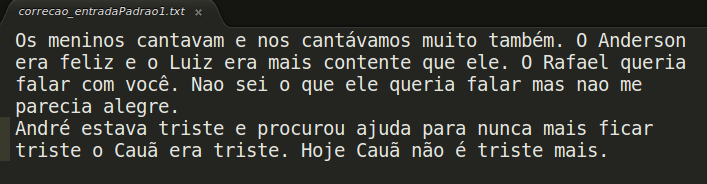
Executa todas as funções acima. Também pede que seja colocada a palavra repetida que se deseja alterar. A figura abaixo mostra um caso onde todas as opções de correção foram utilizadas e a palavra substituída foi “feliz”.

Figura - Saída com todos os casos corrigidos

Figura - Entrada com todos os tipos de erros

## 3.5. Excluir texto da biblioteca

Por último o programa também oferece a possibilidade de limpar da biblioteca um texto que não será mais usado ou que já foi corrigido de acordo com a vontade do usuário, ao ser selecionada a opção procura o título informado pelo usuário e o remove da biblioteca, porém mantém o arquivo original .txt gerado ou fornecido pelo usuário, apenas tirando-o dos registros de persistência, podendo assim ser incluído novamente caso desejado.

Ao ser selecionada, será removido da lista de títulos da biblioteca o texto especificado, e serão recarregados os demais textos mantendo assim organizado o índice de exibição de textos da função de exibição, evitando assim que a biblioteca fique bagunçada ou com índices em branco.

# Conclusão

Ao final do trabalho podemos observar as facilidades e a portabilidade oferecida pela linguagem C++ em suas interações com outras linguagens, como no nosso caso, a Perl, onde pudemos usar da interface simples e comandos específicos da Perl para fazer os tratamentos de texto, e das classes e funções mais robustas do C++ para realizar o gerenciamento de texto e facilitar a programação através da criação de um objeto simples que comandasse todo o programa e chamasse as funções em Perl quando necessário, poupando assim muito esforço de cópias de códigos semelhantes e evitando espalhar possíveis erros pelo código, além de também usufruir da possibilidade de tratamento de erro e exceção para contornar possíveis problemas em tempo de execução sem oferecer riscos de travamento e perda de dados.

# 