

Universidade Federal  
do Rio de Janeiro

---

Escola Politécnica

Luiz Felipe Léo  
Felipe Claudio da Silva Santos

Terceiro Relatório – Corretor Ortográfico  
12 de Maio de 2017

## SUMÁRIO

1. Introdução .....	3
2. Implementação do Programa.....	3
2.1. Classe perlWrapper .....	3
2.2. Classe biblioteca .....	4
2.3. Classe ExcecaoTamanhoPalavra.....	5
3. Casos de Uso .....	6
3.1. Instalação .....	6
3.2. Inserção de texto na Biblioteca.....	7
3.3. Exibir textos contidos na biblioteca .....	7
3.4. Selecionar texto para exibir informações .....	7
3.4.1. Correção de plural .....	9
3.4.2. Correção de palavras.....	9
3.4.3. Correção de pontuação .....	9
3.4.4. Colocar sinônimos em repetição.....	10
3.4.5. Correção de letras maiúscula .....	11
3.4.6. Correção completa .....	11
3.5. Excluir texto da biblioteca .....	12
4. Conclusão .....	12

## Lista de Figuras

Figura 1- Classe PerlWrapper .....	4
Figura 2 - Classe biblioteca .....	5
Figura 3 - Classe ExcecaoTamanhoPalavra.....	5
Figura 4 - Execução do Makefile .....	6
Figura 5 - Código do makefile.....	6
Figura 6 - Inserir texto na biblioteca .....	7
Figura 7 - Exibir textos .....	7
Figura 8 – Seleção de texto para exibir informações .....	8
Figura 9 - Menu de correções .....	8
Figura 10 - Código da correção de plural .....	9
Figura 11 - Código da função de substituição de repetição.....	10
Figura 12 - Identificação de palavra repetida .....	11
Figura 13 - Saída com todos os casos corrigidos.....	11
Figura 14 - Entrada com todos os tipos de erros .....	11

## 1. 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.

## 2. IMPLEMENTAÇÃO DO PROGRAMA

### 2.1. 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.

```

// Classe que torna transparente o uso da comunicacao entre o c++ e o perl
class PerlWrapper
{
public:
    // Precisa ser inicializado com uma referencia para o numero de argumentos
    // na entrada, referencia para o vetor com os textos de entrada
    // e a referencia para o ambiente
    PerlWrapper(int *argc, char ***argv, char ***env);
    ~PerlWrapper();

    // Arquivo de entrada pode ser diferente de arquivoDeEntrada.txt
    // porem, internamente o programa usa um arquivo de entrada e outro
    // de saida padrao

    // Esses sets e gets servem para informar ao programa em quais arquivos procurar conteudo
    // E em qual arquivo salvar o resultado processado

    // Lanca uma execucao caso a string seja maior que o comprimento maximo permitido
    // Ha possibilidade de alterar os arquivos utilizados para entrada e saida
    // durante o codigo utilizando estes metodos abaixo
    void setArquivoEntrada (const string);
    void getArquivoEntrada () const ;
    void setArquivoSaida (const string);
    void getArquivoSaida () const;

    // Funcoes presente no arquivo ModuloCorretorOrtografico.pm
    void ColocarPontoFinal ();

    // Necessita da palavra repetida a ser
    // lanca a execucao ExecucaoTamanhoPalavra caso a palavra colocada na entrada
    // seja maior que o tamanho permitido
    void SubstituirPalavrasRepetidas (string palavraExterna = "");
    void ColocarLetrasMaiusculas ();
    void CorrecaoDePalavras ();
    void CorrecaoDePlural ();

    // Chama todas as funcoes e realiza a correcao completa sobre o arquivo de entrada
    void CorrecaoCompleta ();

private:
    //Funcao auxiliar que so sera utilizada dentro da classe
    // copia o arquivo de entrada para o arquivo padrao
    void copiarArquivo (const char *, const char *);
    ifstream arquivoEntrada;
    ofstream arquivoSaida;
    string nomeEntrada;
    string nomeSaida;

    // Tamanho maximo de uma palavra repetida que pode ser tratada
    static const unsigned MAX_TAMANHO PALAVRA = 100;
    static const unsigned MAX_TAMANHO ARQUIVO = 100 + 9; //para adicionar "correcao_"
};

```

Figura 1- Classe PerlWrapper

## 2.2. 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.

```
#include <iostream>
#include <string>
#include <string>
#include <fstream>
#include "perlWrapper.h"

using namespace std;

class Biblioteca {
public:
    Biblioteca(int ultimo);
    void adicionarTexto(string,int);
    void exibirBiblioteca();
    void exibirErros(string);
    void excluirTexto(string);
    void excluirTudo();

private:
    string textos[100];
    string nomeTexto;
    int ultimo;
};
```

Figura 2 - Classe biblioteca

### 2.3. 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.

```
#ifndef _EXCECAOTAMANHOPALAVRA_H
#define _EXCECAOTAMANHOPALAVRA_H "ExcecaoTamanhoPalavra.h"
#include <exception>

using namespace std;

class ExcecaoTamanhoPalavra : public exception
{
public:
    ExcecaoTamanhoPalavra () : exception() {}
    virtual const char * what () const throw()
    { return "texto excede o tamanho maximo permitido na entrada da funcao\n"; }
};

#endif
```

Figura 3 - Classe ExcecaoTamanhoPalavra

### 3. CASOS DE USO

#### 3.1. 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:

```
CC = g++
CCFLAGS = -Wall -std=c++11 $(shell perl -MExtUtils::Embed -e ccopts) -lstdc++
LD = g++
LDLFLAGS = -Wall $(shell perl -MExtUtils::Embed -e ldopts)

ALL = cperl\
      cperl2

WRAPPERC = perlWrapper.cpp
PROGRAMA_PRINCIPALC = Programa_Principal.cpp
BIBLIOTECAC = biblioteca.cpp
TRABALH0030BJS = Programa_Principal.o perlWrapper.o biblioteca.o
TRABALH003EXEC = corretor

EXECS = $(TRABALH003EXEC)

OBS = $(TRABALH0030BJS)

all: $(ALL)

trabalho03:
    $(CC) $(CCFLAGS) -c $(WRAPPERC)
    $(CC) $(CCFLAGS) -c $(BIBLIOTECAC)
    $(CC) $(CCFLAGS) -c $(PROGRAMA_PRINCIPALC)
    $(LD) -L/usr/lib -o $(TRABALH003EXEC) $(TRABALH0030BJS) $(LDLFLAGS)

install: trabalho03

rebuild: clean trabalho03

clean:
    rm -f $(EXECS) $(OBS)
```

Figura 5 - Código do makefile

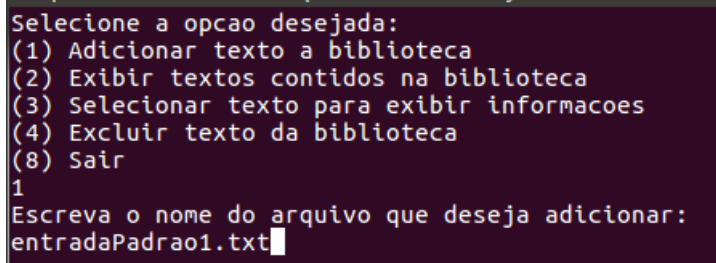
```
felipe@felipe-VirtualBox:~/LING_PROG/Linguagem_de_Programacao/trabalhoFinal/parte3/final$ make install
g++ -Wall -std=c++11 -D_REENTRANT -D_GNU_SOURCE -DDEBIAN -fwrapv -fno-strict-aliasing -pipe -I/usr/local/include -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64 -I/usr/lib/x86_64-linux-gnu/perl/5.22/CORE -lstdc++ -c perlWrapper.cpp
g++ -Wall -std=c++11 -D_REENTRANT -D_GNU_SOURCE -DDEBIAN -fwrapv -fno-strict-aliasing -pipe -I/usr/local/include -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64 -I/usr/lib/x86_64-linux-gnu/perl/5.22/CORE -lstdc++ -c biblioteca.cpp
g++ -Wall -std=c++11 -D_REENTRANT -D_GNU_SOURCE -DDEBIAN -fwrapv -fno-strict-aliasing -pipe -I/usr/local/include -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64 -I/usr/lib/x86_64-linux-gnu/perl/5.22/CORE -lstdc++ -c Programa_Principal.cpp
g++ -L/usr/lib -o corretor Programa_Principal.o perlWrapper.o biblioteca.o -Wall -Wl,-E -fstack-protector-strong -L/usr/local/lib -L/usr/lib/x86_64-linux-gnu/perl/5.22/CORE -lperl -ldl -lm -lpthread -lc -lcrypt
felipe@felipe-VirtualBox:~/LING_PROG/Linguagem_de_Programacao/trabalhoFinal/parte3/final$
```

Figura 4 - Execução do Makefile

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

### 3.2. 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,



```
Selecione a opcao desejada:
(1) Adicionar texto a biblioteca
(2) Exibir textos contidos na biblioteca
(3) Selecionar texto para exibir informacoes
(4) Excluir texto da biblioteca
(8) Sair
1
Escreva o nome do arquivo que deseja adicionar:
entradaPadrao1.txt
```

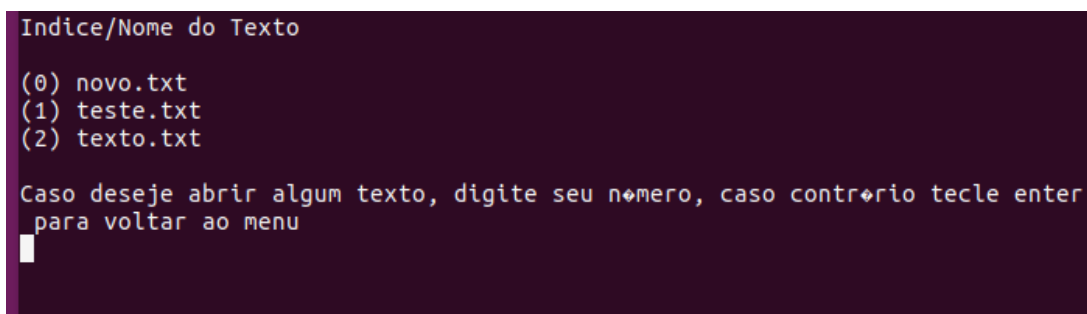
Figura 6 - Inserir texto na biblioteca

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.

### 3.3. 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.



```
Indice/Nome do Texto
(0) novo.txt
(1) teste.txt
(2) texto.txt

Caso deseje abrir algum texto, digite seu numero, caso contrario tecle enter
para voltar ao menu

```

Figura 7 - Exibir textos

### 3.4. 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”.

```
Selecione a opcao desejada:
(1) Adicionar texto a biblioteca
(2) Exibir textos contidos na biblioteca
(3) Selecionar texto para exibir informacoes
(4) Excluir texto da biblioteca
(8) Sair
3
Escreva o nome do texto que deseja exibir:
novo.txt
```

Figura 8 – Seleção de texto para exibir informações

```
Os meninos cantava e nos cantava muito tb O anderson e
ra feliz e o luiz era mais feliz que ele. rafael queria falar com vc Nao sei
o que ele queria falar, mas nao me parecia feliz.
André estava triste e procurou ajuda para nunca mais ficar triste, o cauã er
a triste Hoje cauã não é triste mais

O texto tem 302 caracteres
Que tipo de correcao deseja fazer?
(1) Correcao de plural
(2) Correcao de palavras
(3) Correcao de pontuacao
(4) Colocar sinonimos em repeticao
(5) Correcao de letras maiusculas
(6) Correcao completa
(8) Voltar ao Menu principal
```

Figura 9 - Menu de correções

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



### 3.4.1. Correção de plural

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

```
void PerlWrapper::CorrecaoDePlural ()
{
    char *my_argv[] = {(char *)"", (char *) "perlMain.pl"};
    perl_parse(my_perl, NULL, 0, my_argv, (char **)NULL);
    perl_run(my_perl);

    dSP;
    ENTER;
    SAVETMPS;
    PUSHMARK (SP);
    XPUSHs(sv_2mortal(newSViv(PLURAL)));
    PUTBACK;
    call_pv("principal", G_SCALAR);
    SPAGAIN;

    PUTBACK;
    FREETMPS;
    LEAVE;
    copiarArquivo ("SaidaCorrigida.txt", "arquivoDeEntrada.txt");
}
```

Figura 10 - Código da correção de plural

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

### 3.4.2. Correção de palavras

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

### 3.4.3. 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.

#### 3.4.4. 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.

```
void PerlWrapper::SubstituirPalavrasRepetidas (string palavraExterna)
{
    char *my_argv[] = {(char *)"", (char *) "perlMain.pl"};
    perl_parse(my_perl, NULL, 0, my_argv, (char **)NULL);
    perl_run(my_perl);
    char palavra[MAX_TAMANHO_PALAVRA];

    if (palavraExterna == "")
    {
        cout << "Digite a Palavra Repetida" << endl;
        cin >> palavra;

        if (strlen(palavra) > MAX_TAMANHO_PALAVRA)
            throw ExcecaoTamanhoPalavra();
    }
    else
    {
        if (strlen(palavraExterna.c_str()) > MAX_TAMANHO_PALAVRA)
            throw ExcecaoTamanhoPalavra();

        strcpy(palavra, palavraExterna.c_str());
    }

    dSP;
    ENTER;
    SAVETMPS;
    PUSHMARK (SP);
    XPUSHs(sv_2mortal(newSViv(REPETICA0)));
    XPUSHs(sv_2mortal(newSVpv(palavra, strlen(palavra))));
    PUTBACK;
    call_pv("principal", G_SCALAR);
    SPAGAIN;

    PUTBACK;
    FREETMPS;
    LEAVE;
    copiarArquivo ("SaidaCorrigida.txt", "arquivoDeEntrada.txt");
}
```

Figura 11 - Código da função de substituição de repetição

```

tamanho: 20 tamanho: 180s meninos cantava e nos cant
ava muito tb O anderson era feliz e o luiz era mais
feliz que ele. rafael queria falar com vc Nao sei
o que ele queria falar, mas nao me parecia feliz.
André estava triste e procurou ajuda para nunca mai
s ficar triste, o cauã era triste Hoje cauã não é t
riste mais

0 texto tem 302 caracteres
Que tipo de correcao deseja fazer?
(1) Correcao de plural
(2) Correcao de palavras
(3) Correcao de pontuacao
(4) Colocar sinonimos em repeticao
(5) Correcao de letras maiusculas
(6) Correcao completa
(8) Voltar ao Menu principal
4
Qual palavra repetida deseja alterar?
feliz

```

Figura 12 - Identificação de palavra repetida

### 3.4.5. Correção de letras maiúscula

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

### 3.4.6. 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”.

```

entradaPadrao1.txt x
Os meninos cantava e nos cantava muito tb O anderson era
feliz e o luiz era mais feliz que ele O rafael queria falar
com vc Nao sei o que ele queria falar, mas nao me parecia
feliz.
André estava triste e procurou ajuda para nunca mais ficar
triste, o cauã era triste Hoje cauã não é triste mais

```

Figura 14 - Entrada com todos os tipos de erros

```

correcao_entradaPadrao1.txt x
Os meninos cantavam e nos cantávamos muito também. O Anderson
era feliz e o Luiz era mais contente que ele. O Rafael queria
falar com você. Nao sei o que ele queria falar mas nao me
parecia alegre.
André estava triste e procurou ajuda para nunca mais ficar
triste o Cauã era triste. Hoje Cauã não é triste mais.

```

Figura 13 - Saída com todos os casos corrigidos

### 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.

## 4. 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.