

Relatório  
Linguagens de Programação

Eduardo Naslausky  
Igor F. do Nascimento  
  
18 de Julho de 2016

1. Introdução

Nosso trabalho consiste no desenvolvimento de um programa que se aproveita da cifra de Vigenère para criptografar, descriptografar listar e gerenciar os arquivos de textos fornecidos pelo usuário. Na parte III desenvolvemos a implementação da lógica em C++ que por sua vez irá fazer chamadas à função desenvolvida na parte II (programa em perl), para que a mesma faça a criptografia de forma organizada e ao mesmo tempo modifique e salve os arquivos.   
 O programa possui cinco funções: Listar todos os arquivos textos contidos na mesma pasta do programa (considera-se arquivo texto qualquer arquivo com extensão “.txt”). Criptografar e descriptografar um arquivo texto fornecido pelo usuário. Tentar descobrir a senha de um arquivo texto criptografado, fornecido um dicionário de palavras pelo usuário. Por último, exibir o conteúdo de arquivo texto na tela.

1. Implementação do programa

A linguagem utilizada foi C++ de forma que pudemos fazer uso da orientação a objeto de C++ para organizar e orientar nossos arquivos e funções de forma efetiva. As classes criadas foram: Arquivo, Dicionario, Mensagem e dirTools. O programa conta ainda com o módulo principal( CryptoCesar.cpp ) que integra todas as classes e funções que o programa executa, por meio da função main.

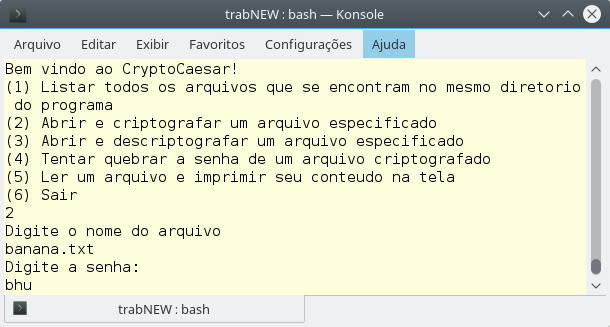
A classe Arquivo é responsável por Abrir o arquivo e fechá-lo. Esses métodos são reaproveitados nas classes que são herdeiras de arquivo, e recebem como argumento uma string constante que pode ser “w” para escrita e “r” para leitura. Temos também: getFilename que retorna o nome do arquivo, que está como atributo protegido “fileName”. Usamos o atributo protected justamente porque a classe Arquivo é uma classe mãe a outras classes do nosso programa, e sendo assim é extremamente proveitoso que somente suas classes “filhas” possuam acesso a esta instância. Arquivo possui também um construtor, no qual é passado o nome do arquivo externo ao programa, que o objeto está vinculado.

A classe Mensagem primeiramente é uma classe filha da classe arquivo, pois esta é a classe que modifica o arquivo de acordo com o que o usuário pede. Mensagem possui como métodos públicos as seguintes funções: Criptografar(Função que chama a função code em perl de criptografia) e Descriptografar(Função que chama a função “decode” em perl de descriptografia). Possui também como atributo privado um ponteiro my\_perl que serve somente para fazer a chamada do programa em perl. O mesmo foi posto como atributo privado porque é utilizado em todas as chamadas em perl, que acontecem em dois métodos da classe. Além dos atributos públicos e privado a classe possui um construtor que irá passar os argumentos da linha de comando passados pelo perl para o nosso programa, e como mensagem é “filha” de arquivo a mesma precisa fazer a chamada do construtor da classe mãe. Os comandos de inicialização do interpretador e de alocação de memória de perl foram postos no construtor, e as de liberação de memória foram postas no destrutor. Isso inclusive é bom, visto que esses comandos só podem ser usados uma vez por programa. Nas funções de criptografia e descriptografia em si, só são chamadas as funções “sub” propriamente ditas, em perl.

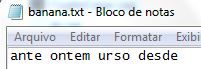
A classe Dicionário também é uma classe filha da classe arquivo, porque a mesma será usada para salvar palavras em um dicionário, ou mesmo fazer comparação com as palavras do arquivo. Dicionário possui a priori somente atributos públicos, que são: numVerbetes (número de palavras carregadas no dicionario), aberto(serve como estado para saber se o arquivo do dicionário está aberto) e a função que compara as palavras do arquivo com as do dicionário. Essa função verificaMatch retorna 1 caso a palavra passada como argumento se encontra no dicionário e 0 caso contrário. A mesma possui também um construtor que tem a mesma função do da classe Mensagem.

A classe dirTools serve para realizar a primeira função de exibir os txts que estão na pasta do programa, para que assim o usuário possa escolher de forma mais fluída qual arquivo o mesmo deseja criptografar. Esta classe possui como instâncias privadas mostrarTxts(função que exibe os txts) e o nome do arquivo. Como atributo privado este possui o path do arquivo a ser comparado. Essa função de exibir os “\*.txt” dentro de uma pasta é baseada na biblioteca “dirent.h” que teve seu funcionamento pesquisado e aprendido pelo grupo.

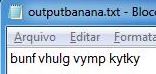
O CryptoCesar tem como principal função integrar todas as classes e instâncias para dessa forma fazer com que o programa trabalhe de forma uníssona desempenhando o papel de um sistema de criptografia simples, que como já foi dito anteriormente, se baseia a cifra de Vigenère.

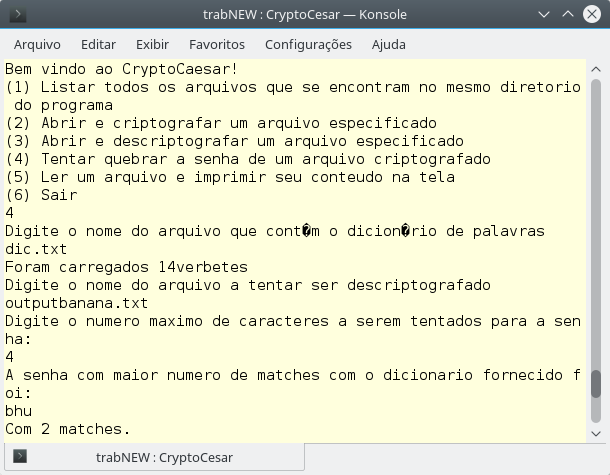
1.  Casos de Uso

No primeiro caso, podemos ver um usuário que criptografou um texto simples que estava contido dentro do banana.txt.



Teve seu output criptografado e salvo em “outputbanana.txt”:



Após isso o programa foi finalizado (pois acreditamos que finalizar o programa após funções de criptografar e descriptografar fosse mais seguro, em função da senha ser uma variável do sistema). Após isso ele re-abriu o programa e selecionou a opção de quebrar senha:

Primeiramente ele disse ao programa em qual arquivo continha as palavras a serem dadas como existentes, dic.txt. Foram carregadas 14 palavras, como a foto mostra.

Após isso, o programa perguntou quantas letras no máximo deverão ser tentadas para “chute”, pelo método de força bruta.

O programa tem como método de quebra de senha a tentativa subsequente de senhas. Ele tenta primeiramente a senha “a”, depois a “b” e assim por diante. A cada senha incrementada, ele re-roda o método descriptografar e, com o resultado (neste caso outputoutputbanana.txt) verifica então dentre as palavras de saída quantas estão contidas no dicionário. A senha com maior resultado de “matches” deve ser a funcional.

Podemos perceber que o mesmo conseguiu quebrar a senha bhu fornecida pelo usuário com quatro matches, exatamente 4 palavras contidas no dicionário.

1. Conclusão

Apesar do programa ter apresentado uma interação com o perl muito complicada de ser alinhada, o mesmo foi realizado com sucesso. Se fez uso de diversos conceitos aprendidos em sala de aula, tais como herança, construtores, destrutores, orientação a objeto etc. Adicionalmente usamos classes ifstream, ofstream, contidas na biblioteca fstream, e que se mostraram extremamente úteis e versáteis em nossa implementação. Fizemos uso também da classe vector para tratamento do dicionário, e principalmente da classe string que já possuía sobrecarga em vários operadores, nos quais facilitou muito a implementação de várias lógicas do programa. A parte com maiores problemas foram as partes que envolviam a integração com Perl pois era necessário fazer acesso de forma externa a essa linguagem, o que acabou limitando bastante alguns usos.

Referências

<http://stackoverflow.com/>

[www.cplusplus.com/](http://www.cplusplus.com/)

<http://www.perlmonks.org/>

<http://www.gta.ufrj.br/~miguel/lingprog.html>

<http://perldoc.perl.org/perlop.html#Auto-increment-and-Auto-decrement>