INSTITUTO FEDERAL DE MINAS GERAIS

Bacharelado em Ciência da Computação

Disciplina: Matemática Discreta

Trabalho Prático 03

Professor: Diego Mello da Silva

Formiga-MG
27 de março de 2014

Sumário

1 Informações Gerais

Este documento descreve a especificação do Trabalho Prático 03 da disciplina Matemática Discreta, e deve ser seguido de forma a contemplar os itens considerados na avaliação por parte do professor da disciplina. O trabalho deve ser feito em grupo de até 2 alunos e tem o valor de 10 pontos.

Em relação ao prazo o trabalho prático deverá ser realizado em 3 semanas (21 dias) contados a partir da data da publicação do mesmo em sala de aula pelo professor. O trabalho deverá ser entregue até as 23:59 hs da data limite. Trabalhos entregues após este prazo serão desconsiderados (isto é, valerão zero).

O documento é organizado como segue. Na Seção 2 será apresentado o problema que este trabalho prático pretende resolver; na Seção 3 serão apresentados os requisitos de *software* que, se implementados, permitirão resolver o problema proposto; na Seção 4 serão apresentados os critérios de avaliação do trabalho e respectiva pontuação; na seção 5 serão abordadas algumas considerações importantes sobre o trabalho em equipe e código de conduta.

2 O problema

A palavra criptografia vem do grego *kryptos* (esconder, ocultar) e *graphein* (escrita), e pode ser interpretada como 'escrita oculta' ou 'escrita secreta'. Trata-se de um conjunto de conceitos e técnicas que visa codificar uma mensagem de forma que somente o remetente e o destinatário possam compreendê-la, evitando que um terceiro consiga interpretá-la.

O envio e recebimento de mensagens sigilosas é uma necessidade antiga; segundo Tkotz[1] a primeira referência documentada de escrita criptográfica data de 1900 a.C., em uma vila egipcia perto do rio Nilo. De fato a história está repleta de casos onde a criptografia esteve presente: conflitos armados, interesses comerciais, sociedades secretas utilizaram massivamente da criptografia para trocar informações confidenciais. Em algumas situações conhecer o conteúdo das mensagens dos adversários era a diferença entre vencer ou perder uma guerra; ver o caso da máquina *Enigma*[2] usada pelos alemães na Segunda Guerra Mundial cuja contribuição vital do matemático Alan Turing, conhecido como Pai da Computação, foi essencial para decifrar as mensagens transmitidas pelos alemães para as tropas aliadas.

Nos dias atuais a criptografia também tem papel fundamental, embora a maioria das pessoas não se dê conta disso. Transações bancárias, contas de usuários de redes sociais, contabilidade corporativa, correspondências digitais, segredos de estado e outros são exemplos de situações presentes no mundo informatizado em que uso da criptografia é desejada ou essencial para garantir sigilo e privacidade. Milhares de informações trafegam diariamente nas redes de computadores, em especial na internet, de tal forma que proteger informações usando criptografia ao menos dificulta o entendimento sobre o conteúdo das mensagens trocadas quando interceptadas por pessoas não autorizadas.

Este documento descreve uma aplicação simples e didática de criptografia de dados usando o método de criptografia muito antigo denominado **cifra de Vigenère**,

[1] Tkotz, Viktoria. Criptografia: Segredos Embalados para Viagem, Novatec. ISBN 8575220713

[2] Link: `http://www.bbc.co.uk/history/topics/enigma`

publicada em 1585 no livro *Traicte de Chiffres* de Blaise de Vigenère. A cifra de Vigenère é um método bem conhecido e mais simples método de criptografia que utiliza técnicas de substituição polialfabética. Cifras de substituição polialfabéticas possuem múltiplas chaves de uma letra, cada qual usada para criptografar uma letra do texto plano. Entende-se por **texto plano** o texto original, limpo, sem códigos usados para enconder seu conteúdo; e por **texto cifrado** o texto plano transformado por algum método de cifragem em um texto criptografado.

Embora a cifra de Vigenère tenha sido considerada inquebrável por muitos anos na prática ela pode ser quebrada por criptoanálise. No entanto seu interesse se mantêm sob o ponto de vista matemático e didático, sendo abordado como tema deste trabalho prático.

Figura 1: Blaise de Vigenère (1523-1596), diplomata e criptógrafo francês

Em uma cifragem polialfabética, cada letra da chave é usada para cifrar uma letra do texto plano. Assim, a primeira chave encripta a primeira letra do texto plano, a segunda chave encripta a segunda letra do texto plano, e assim por diante. Após todas as chaves terem sido utilizadas elas são recicladas. Se uma dada chave possui 20 letras, então a cada letra do texto plano utilizar-se-á a mesma chave para criptografá-lo. Tal intervalo é denominado período da cifra; em criptografia clássica chaves com períodos longos são mais difíceis de quebrar do que chaves com períodos curtos. Cifras polialfabéticas foram inventadas por León Battista em 1568, e foram adotadas tanto em conflitos (como é o caso de seu uso pelo exército da União durante a Gerra Civil Americana) quanto em produtos comerciais (como no editor de texto comercial WordPerfect, da década de 90).

Métodos de cifragem baseados em substituição trabalham com valores numéricos associados com os caracteres alfabéticos do alfabeto que constitui o texto plano e o texto cifrado. Para os caso do alfabeto latino, que possui 26 letras, podemos associar a cada letra um valor numérico. Letras maiúsculas e minúsculas correspondentes possuem o mesmo valor numérico, não apresentando variação. As tabelas a seguir ilustram os códigos numéricos de cada letra.

Letra	Número
A, a	0
B, b	1
C, c	2
D, d	3
E, e	4
F, f	5
G, g	6
H, h	7
I, i	8
J, j	9
K, k	10
L, l	11
M, m	12

Letra	Número
N, n	13
O, o	14
P, p	15
Q, q	16
R, r	17
S, s	18
T, t	19
U, u	20
V, v	21
W, w	22
X, x	23
Y, y	24
Z, z	25

Podemos expressar a cifra de Vigenère da seguinte maneira. Assuma uma sequência de texto plano $P = \langle p_0, p_1, p_2, \ldots, p_{n-1} \rangle$ e de chaves $K = \langle k_0, k_1, k_2, \ldots, k_{m-1} \rangle$, onde em geral $m < n$. A sequência de letras do texto cifrado $C = \langle c_0, c_1, c_2, \ldots, c_{n-1} \rangle$ é calculada usando a função $E(P, K)$ que encripta o texto plano P usando a chave K, e pela função $D(C, K)$ que decripta o texto cifrado convertendo-o em texto plano. A função $E(P, K)$ pode ser implementada usando aritmética modular da seguinte maneira, assumindo que existem 26 caracteres diferentes no alfabeto em questão, e que a chave possui m caracteres:

$$c_i = \left[p_i + (k_i \mod m)\right] \mod 26$$

Para exemplificar o método, seja a mensagem `'Achamos o tesouro'`, que representa um texto claro, que deverá ser cifrado com a chave `'navio'`. Logo, $P = \langle achamosotesouro \rangle$ e $K = \langle navio \rangle$, com $m = 5$. Para encontrar o texto cifrado C usaremos a tabela a seguir, onde os caracteres da mensagem correspondente ao texto plano e seus valores numéricos, assim como os caracteres correspondentes à chave e seus valores numéricos estão dispostos. Após preencher a tabela, soma-se os valores numéricos de cada caracter do texto plano com o correspondente da chave (que é ciclada sempre que chegar no seu último caractere), aplicando-se a operação $(p_i + k_i) \mod 26$ para encontrar o caracter c_i correspondente. No exemplo em questão, a mensagem $P = \langle achamosotesouro \rangle$ transforma-se na mensagem $C = \langle dcidqrspwivovuc \rangle$ após aplicar a cifra de Vigenère usando a chave $K = \langle navio \rangle$.

P	a	c	h	a	m	o	s	o	t	e	s	o	u	r	o
P	00	02	07	00	12	14	18	14	19	04	18	14	20	17	14
K	n	a	v	i	o	n	a	v	i	o	n	a	v	i	o
K	13	00	21	08	14	13	00	21	08	14	13	00	21	08	14
$K \mod 5$	03	00	01	03	04	03	00	01	03	04	03	00	01	03	04
$P + K$	03	02	08	03	16	17	18	15	22	08	21	14	21	20	18
$P + K \mod 26$	03	02	08	03	16	17	18	15	22	08	21	14	21	20	18
C	d	c	i	d	q	r	s	p	w	i	v	o	v	u	c

De maneira semelhante, dados o texto cifrado C e a mesma chave K usada para

cifrar C, a função $D(C, K)$ que decripta o texto cifrado em texto plano pode ser implementada da seguinte maneira:

$$\boxed{p_i = \left[c_i - (k_i \mod m)\right] \mod 26}$$

Desde que a mesma chave seja utilizada no texto cifrado, aplicando a operação inversa obtêm-se o texto plano original. Fica como exercício para o grupo converter a mensagem cifrada $C = \langle dcidqrspwivovuc \rangle$ na mensagem original P usando a chave $K = \langle navio \rangle$ aplicando-se a operação inversa $D(C, K)$. Qualquer pessoa que intercepte a mensagem cifrada ficará sem compreender seu significado se não souber qual foi a chave usada para transformar o texto plano no texto cifrado, a menos que tenha conhecimentos sobre criptoanálise e disponha de algum dispositivo computacional de cálculo[3].

3 Especificação

Esta seção apresenta a especificação dos aplicativos `encrypt` e `decrypt` que realizam, respectivamente, as operações de criptografar e decriptografar textos planos no formato ASCII contendo apenas caracteres alfabéticos usando o método de criptografia conhecido como *cifra de Vigenère*, publicada em 1586 e descrita na seção introdutória deste documento.

Nesta seção serão apresentados os requisitos funcionais para implementar uma versão didática desta cifra nas linguagens `C` ou `Pascal`. Ambos os códigos fontes devem ser testados e compilados no ambiente Linux Ubuntu, cujas distribuições encontram-se disponíveis e instaladas nos computadores dos laboratórios de informática do IFMG Campus Formiga. O código fonte será compilado pelo professor usando os compiladores `gcc` e `fpc`, logo é essencial que o grupo verifique a codificação usando os compiladores e ambiente especificados. Caso o código fonte de ambos os aplicativos seja escrito em linguagem `C` deverão ser denominados de `encrypt.c` e `decrypt.c`; caso sejam escritos em linguagem `Pascal` deverão ser denominados `encrypt.pas` e `decrypt.pas`.

Quanto ao uso, após compilados os aplicativos deverão ser chamados em linha de comando, com a seguinte sintaxe:

```
user@machine$ encrypt <string-chave> <arquivo-entrada> <arquivo-saida>
```

para o aplicativo que aplica criptografia no texto plano, e

```
user@machine$ decrypt <string-chave> <arquivo-entrada> <arquivo-saida>
```

para o aplicativo que transforma o texto criptografado em texto plano. Os parâmetros de ambos os aplicativos são dados a seguir:

- `<string-chave>`: chave usada para criptografar o texto, sendo esta usada no método de criptografia por Vigenère. Não deve ter espaços em branco e deve ser composta exclusivamente por caracteres alfabéticos.

[3] Para exemplificar, ver `http://archives.math.utk.edu/ICTCM/VOL24/S094/paper.pdf`

- `<arquivo-entrada>`: nome do arquivo de entrada usado na transformação de texto plano para criptografado (ou vice-versa).

- `<arquivo-saida>`: nome do arquivo de saída gerado a partir da transformação de texto plano para criptografado (ou vice-versa).

O grupo deverá compactar em um único arquivo no formato `.zip` os códigos fontes dos aplicativos `encrypt` e `decrypt`, assim como pelo menos 10 arquivos no formato texto plano ASCII (`.txt`) que serão usados nos exemplos. **Tais arquivos deverão estar no formato plano, ou seja, sem criptografia**. O arquivo compactado deverá ser enviado pelo email `diego.silva@ifmg.edu.br` **impreterivelmente** até as 23:59 do dia combinado para a entrega. As subseções de 3.1 até 3.10 a seguir descreverão em detalhes cada um dos requisitos funcionais esperados para este trabalho prático.

3.1 Requisito: Operador de Módulo ($x \mod y$)

Neste requisito espera-se que o grupo escreva uma função matemática usando apenas as operações elementares de soma ($+$) e subtração ($-$) para implementar o operador de congruência módulo m. O operador módulo é essencial em aritmética modular e na aplicação de alguns métodos de criptografia, como é o caso de cifras de substituição monoalfabéticas e polialfabéticas.

O operador módulo é o operador que quando aplicado sobre dois números inteiros x e y resulta no resto da divisão inteira de x por y. O operador de módulo $x \mod y$ é também um caso especial de relação de equivalência que cria uma partição no conjunto dos inteiros; se qualquer número x é dividido por um número y escolhido arbitrariamente do conjunto dos inteiros então a operação de módulo retornará resultados entre $0, \ldots, (y-1)$.

Para este trabalho o operador de módulo deverá ser implementado usando apenas as operações básicas de soma e subtração, com laços de repetição, em um algoritmo de divisão baseado em subtrações sucessivas do dividendo por quantidades discretas do divisor, processo este repetido enquanto o resultado da diferença for um inteiro positivo ou nulo. O requisito é considerado cumprido se a implementação especificada não utilizar nenhum constructo básico das linguagem de programação consideradas no trabalho (tais como os operadores `%` da linguagem `C` e o operador `MOD` da linguagem `Pascal`). Esta função será usada em ambos os aplicativos `encrypt` e `decrypt`.

3.2 Requisito: Entrada de Dados

Este requisito especifica a entrada de dados dos métodos `encrypt` e `decrypt` usados respectivamente para criptografar e decriptografar um texto plano. Neste requisito espera-se que o grupo faça a entrada de dados via console, obtendo os parâmetros informados via terminal. Se o requisito estiver completamente implementado será possível informar, via linha de comando, a *string* com a chave do método de Vigenère, o nome do arquivo de entrada e o nome do arquivo de saída. Tais parâmetros serão usados ao longo da codificação deste trabalho prático, e devem ser guardadas em estruturas de dados adequadas. Para implementar este requisito pesquisar as funções

`ParamCount()` e `ParamStr()` da linguagem `Pascal`, ou os argumentos da função `int main(int argc, char **argv)` da linguagem `C`. Observe que para cada um dos aplicativos `encrypt` ou `decrypt` a entrada de dados poderá ter significado distinto.

3.3 Requisito: Função `ConverteAscii2Numero()`

Este requisito implementa uma função que retorna um número associado com cada caracter alfabético ASCII. Por exemplo, após o uso da função o caracter `'A'` ou `'a'` torna-se 0; o caracter `'B'` ou `'b'` torna-se 1, e assim sucessivamente. Observe que as letras maiúscula e minúscula possuem código numéricos diferentes na tabela ASCII. Para maiores informações consultar a tabela ASCII e verificar que os caracteres alfabéticos sempre possuem o mesmo código numérico associado.

Observe que para a função em questão, todas as letras do alfabeto, maiúsculas ou minúsculas, devem ser representadas por 26 números diferentes no intervalo $0, \ldots, 25$, segundo a tabela de valores numéricos associados com os caracteres latinos apresentados na seção 2.

Caso o caracter informado esteja fora do intervalo dos caracteres alfabéticos deve-se retornar um código especial (por exemplo, -1) para informar à quem invocou esta função que o caracter em questão é um caracter especial que deve ser mantido intacto, isto é, sem cifragem. Isso permitirá aos aplicativos identificar caracteres especiais como pontuação (pontos finais, virgulas, ponto-e-vírgulas, interrogração, exclamação), caracteres de operação lógica e matemática (sinais de mais, menos, asteriscos, barras, parênteses, colchetes, chaves) e outros (contrabarras, percentos, cerquilhas, arrobas, 'e' comerciais, travessões, etc).

Esta função será usada em ambos os aplicativos `encrypt` e `decrypt` para obter o valor numérico associado com cada caracter do texto plano ou cifrado.

3.4 Requisito: Função `IdxChave2Numero()`

Conforme mencionado na parte introdutória deste documento, a cifra de Vigenère utiliza uma chave cujo tamanho m é menor do que o texto plano, em geral. Portanto a chave cicla a cada m caracteres utilizados na função $E(P, K)$ usado para cifrar o texto plano.

Neste requisito exige-se que o grupo implemente uma função capaz de retornar o caracter da chave correspondente ao i-ésimo caracter do texto plano, informado para a função. Usando-se a função de módulo descrita no requisito da sub-seção 3.1 é possível escolher qualquer caracter da chave fazendo-se com que o índice da chave esteja no intervalo $0, \ldots, m$. Para exemplificar, se a chave é $K = \langle navio \rangle$ e o usuário solicita o caracter da 12^a posição, então teremos $12 \mod 5 = 2$, que corresponde ao caracter `'v'`. Esta função será usada pelas funções descritas a seguir para transformar o texto plano e texto cifrado, ou vice-versa.

3.5 Requisito: Função `Encripta()`

Neste requisito o grupo deverá implementar a função que recebe o i-ésimo caracter do texto plano e o i-ésimo caracter da chave, e aplica a transformação

$$\boxed{c_i = \left[p_i + (k_i \mod m)\right] \mod 26}$$

para transformar o caracter informado no i-ésimo caracter do texto cifrado. Deve-se lembrar que a chave possui m caracteres, enquanto a mensagem possui n caracteres onde, em geral, $m < n$. Isso implica que a chave deverá ser tratada de maneira cíclica usando a função descrita no requisito da sub-seção 3.4. Tal função deverá ser usada pelo aplicativo `encrypt` para realizar a cifragem do texto plano usando a cifra de Vigenère.

3.6 Requisito: Função `Decripta()`

Este requisito implementa a função que, dados o i-ésimo caracter da mensagem cifrada e da chave, retorna o i-ésimo caracter correspondente do texto plano original, segundo a expressão:

$$\boxed{p_i = \left[c_i - (k_i \mod m)\right] \mod 26}$$

Tal função será usada pelo aplicativo `decrypt` para realizar a operação inversa da cifragem de Vigenère.

3.7 Requisito: Função Principal

Este requisito descreve a necessidade de implementação de uma 'função principal' da aplicação, que pode ser mapeada em uma ou mais subfunções na linguagem de programação escolhida pelo grupo. Esta 'função principal' tem como objetivo abrir o arquivo de entrada para leitura, percorrer cada um de seus caracteres e, quando encontrar algum caractere alfabético, transformá-lo em outro caracter que deverá ser gravado em no arquivo de saída especificado. Caracteres diferentes dos alfabéticos devem ser copiados na íntegra, sem aplicar qualquer transformação. Entende-se por transformação qualquer operação de conversão descrita neste documento, seja para transformar texto plano em texto cifrado, ou vice-versa. Para cada um dos aplicativos esta função principal deverá usar a função `Encripta()` ou `Decripta()` correspondente, assim como quaisquer outras funções auxiliares usadas para modularizar o código ou subdividir o problema em problemas menores.

Observação: caracteres fora do intervalo dos caracteres alfabéticos da tabela ASCII não devem ser cifrados pela aplicação.

3.8 Requisito: Saída de Dados

Como saída de dados é esperado que a aplicação retorne um arquivo escrito em texto plano (ASCII) contendo os caracteres alfabéticos transformados. Este requisito se aplica a ambas as aplicações `encrypt` e `decrypt`.

3.9 Requisito: Corretude dos Resultados

Neste requisito verificar-se-á a corretude dos resultados. Para o problema específico deste trabalho prático pretende-se verificar se os aplicativos `encrypt` e `decrypt` conseguem codificar e decodificar o arquivo ASCII informado pelo usuário corretamente dado a chave informada. Se a implementação resultar em um código cifrado reversível então o requisito será considerado cumprido. O professor da disciplina

irá utilizar um ou mais arquivos de testes cujo resultado é conhecido segundo a codificação de caracteres do alfabeto latino apresentado neste documento.

3.10 Requisito: Modularização/Documentação do Código

Um aplicativo bem projetado faz uso de boas práticas de programação. Uma das práticas mais importantes consiste na modularização de código, onde um problema é dividido em sub-problemas menores tais que a solução do problema maior pode ser feita em partes, módulos, funções e procedimentos. Neste requisito requisita-se que o código fonte seja bem modularizado para permitir um entendimento de seu funcionamento por terceiros.

O código fonte de ambas as aplicações `encrypt` e `decrypt` também devem ser bem documentadas com uso de comentários de código. Exige-se por este requisito que cada arquivo de código fonte contenha (i) um cabeçalho com nome completo, matrícula e endereço eletrônico (email) de cada membro do grupo, além do nome do arquivo, explicação sobre o módulo (para que serve) e informações sobre compilação do mesmo; (ii) cada função escrita em linguagem `C` ou `Pascal` tenha um cabeçalho indicando o nome da função e objetivo da função, podendo incluir os parâmetros de entrada e saída esperada; (iii) dentro de cada função exista algum comentário explicando as partes mais fundamentais do código em **quantidade de comentários adequada**.

4 Barema de Correção

Conforme mencionado, o trabalho prático tem valor de 10 pontos. A correção seguirá o barema apresentado a seguir, que lista os requisitos do trabalho prático e suas respectivas pontuações.

Requisito	**Pontos**
Requisito: Operador Módulo ($x \mod y$)	1.0 pto
Requisito: Função `ConverteAscii2Numero()`	1.0 pto
Requisito: Função `IdxChave2Numero()`	1.0 pto
Requisito: Entrada de Dados	1.0 pto
Requisito: Saída de Dados	1.0 pto
Requisito: Função `Encripta()`	1.0 pto
Requisito: Função `Decripta()`	1.0 pto
Requisito: Função Principal	1.0 pto
Requisito: Corretude dos Resultados	1.0 pto
Requisito: Modularização/Documentação de Código	1.0 pto
TOTAL	**10.0 ptos**

5 Considerações Finais

O trabalho prático especificado neste documento procura desenvolver habilidades técnicas e interpessoais entre os membros da equipe de maneira saudável. Para tal, é fundamental que o trabalho seja feito de maneira ética e profissional. Desta forma algumas considerações finais devem ser feitas:

- Este **é um trabalho de programação**, portanto aqueles que possuem dificuldades devem estudar o assunto, procurando livros, tutoriais e outros materiais que complementem sua deficiência em programação por conta própria;

- Trabalhos plagiados ou feitos por terceiros valerão zero;

- Todos os membros da equipe deverão participar efetivamente da implementação do trabalho de forma a aumentar suas perícias em matemática e programação;

- Caso o professor julgue necessário poderá fazer uma arguição dos membros de um ou mais grupos;

- Trabalhos entregues fora do prazo serão desconsiderados;

- Dúvidas sobre o trabalho devem ser tiradas com no máximo 1 semana do prazo de entrega.