

## Tarefa – Strings, Recursão e Arquivos

Uma técnica bem simples de criptografia é aquela onde cada letra de um texto é substituída por outra letra, presente no alfabeto, porém deslocada um certo número de posições à esquerda ou à direita. O número de posições é chamado de **chave** da criptografia. Por exemplo, para o alfabeto inglês de 26 letras e **chave** = 3, a letra A é substituída pela letra D, B por E, e C por F, como a seguir:

[illegible]

Note que a contagem é cíclica (e.g., a letra Z é substituída pela C). Para **encriptar** fazemos um deslocamento à direita; isto é, somamos a posição atual  $x$  com a *chave*  $k$  e calculamos o resto da divisão por 26:  $y = (x + k) \% 26$ . Assim, quando a soma ultrapassar a última posição, as letras iniciais são alcançadas. Por exemplo, para encriptar a letra Y com chave = 3, temos:  $y = (24 + 3) \% 26 = 27 \% 26 = 1$  (que é a letra B). Note que o resto para um número menor que 26 é o próprio número, e.g., para a letra W:  $y = (22 + 3) \% 26 = 25 \% 26 = 25$  (que é a letra Z).

Para **decifrar**, fazemos um deslocamento à esquerda:  $x = (y - k + 26) \% 26$ . Por exemplo, para a letra B encriptada, temos  $y = 1$  e  $x = (1 - 3 + 26) \% 26 = 24 \% 26 = 24$  que é a Y.

Nesta tarefa você vai primeiro desenvolver uma **função RECURSIVA** para encriptar e decifrar uma sentença. Esta função recebe uma cadeia de caracteres e a chave da criptografia. Se a chave for positiva, então a função encripta a *string* recebida. Porém se a chave for negativa, a função decifra (decodifica) o código. Encriptar ou decifrar modificam a *string* recebida (sem criar um novo espaço na memória). Esta função deve ser *void*. Por exemplo, se a *string* for "Uma imagem vale 1000 palavras" e *chave* = 10, então a função altera a *string* para "Ewk swkqow fkvo 1000 zkvkfbkc". E para decifrar essa última *string*, basta chamar a função com *chave* = -10. Note que letras minúsculas se mantêm minúsculas e as maiúsculas se mantêm maiúsculas<sup>1</sup>. E os dígitos numéricos ou qualquer outro símbolo não se alteram. Considere que as sentenças nunca são maiores que 80 caracteres.

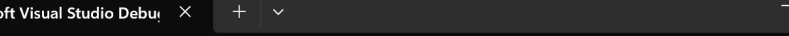
Depois você vai escrever dois programas completos que usam a função de encriptar e decifrar mencionada acima. No primeiro programa você vai, usando **feof()**, ler um ARQUIVO TEXTO contendo algumas sentenças e vai gravar um ARQUIVO BINÁRIO contendo o tamanho da sentença seguido da sentença encriptada (2 registros por sentença). No segundo programa, você vai ler, sem usar **feof()**, o ARQUIVO BINÁRIO, decodificar cada sentença e concatenar as sentenças em um único texto. Considere que as sentenças isoladas nunca são maiores do que 80 caracteres e a soma de todas elas não chega a 10000 caracteres. Coloque pontos finais nas sentenças. Por exemplo, para o arquivo texto:

Sabedorias

Uma imagem vale 1000 palavras

Mestre nao eh quem sempre ensina, mas quem de repente aprende

o primeiro programa grava o arquivo binário encriptado e o segundo programa apresenta o seguinte *output*:



The screenshot shows a terminal window with the following text:

```
Cklonybskc
Ewk swkqow fkvo 1000 zkvkfbkc
Wocdbo xky or aeow cowzbo oxcsxk, wkc aeow no bozoxdo kzboxno

TEXT0:
Sabedorias. Uma imagem vale 1000 palavras. Mestre nao eh quem sempre ensina, mas
quem de repente aprende.
```

Use obrigatoriamente sentenças diferentes das do exemplo acima (em quantidade e conteúdo). Deposite **4 (quatro) arquivos**: o arquivo texto, os dois programas e a saída do segundo programa (em PDF, capturando a tela). Para resolver esta tarefa, você deve usar **aritmética de ponteiros**; NÃO use a sintaxe de colchetes.

<sup>1</sup> Dica: para obter a posição de cada caractere da sentença no alfabeto, subtraia o caractere da 'A' maiúscula ou da 'a' minúscula, conforme o caractere seja letra maiúscula ou minúscula. Essa subtração é o  $x$  da equação  $y = (x + k) \% 26$ .