==========

SSC0603 – ED1 – Prof. Fernando Osório – ICMC / USP 2020-2
Prof. Fernando Osório – Estrutura de Dados 1 (Eng.Comp.)
Run.Codes: Trabalho Prático - TP-01 Versão TP-01 v.01
Trabalho: CRIPTO - Criptografia com Listas Encadeadas

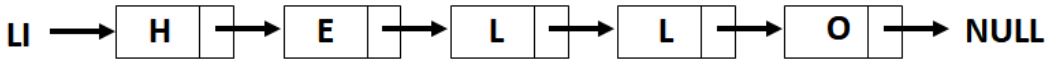
==========

TP01 - CRIPTO

>> Definição Geral do Trabalho: TADs LDES (simples) ou TAD LDED (dupla)
(*) Programa implementado na Linguagem de programação "C"
(**) Podem ser usadas as bibliotecas (rotinas) do TAD ProjListDinEncadeada (Simples ou Dupla) do livro do Backes, ou mesmo, rotinas de outros autores que implementem Listas Dinâmicas.

Este trabalho é relacionado ao uso de **listas encadeadas alocadas dinamicamente** (simples ou duplamente encadeadas). A lista encadeada irá ser usada para armazenar uma lista de caracteres que depois devem ser criptografados de diferentes formas (a serem descritas a seguir). Portanto, a lista encadeada é uma lista que representa uma sequência de caracteres representando um texto, como na figura abaixo:

TEXTO: "HELLO" Representação da Lista com o texto "HELLO" abaixo:



O programa irá receber como entrada um comando (número inteiro) indicando o tipo de operação a ser executado, podendo ser:

1 => Criptografa: Uso da "Cifra de César" (adiciona 1 ao código ASCII de cada caractere).
Quer saber o que é cifra de César? Acesse a Wikipedia (Português ou Inglês)
https://pt.wikipedia.org/wiki/Cifra_de_César

-1 => Descriptografa: Uso da "Cifra de César" (subtrai 1 do código ASCII de cada caractere).
Esta opção (-1) é o inverso da opção 1.

2 => Criptografa: Uso da "Cifra de César" (adiciona 3 ao código ASCII de cada caractere).

-2 => Descriptografa: Uso da "Cifra de César" (subtrai 3 do código ASCII de cada caractere).

3 => Criptografa: Uso da "Cifra de César", porém adiciona uma sequência de valores ao código ASCII de cada caractere. A sequência adotada será: +1, +2, +3, ou seja, soma +1 ao 1o. caractere, +2 ao segundo, +3 ao terceiro, +1 ao quarto e segue assim.

-3 => Descriptografa: Desfaz a criptografia da opção 3, ou seja, subtrai ciclicamente -1,-2,-3.

4 => Criptografa: Uso de uma chave de criptográfica (criptografia com chave única – simétrica). A chave é uma sequência de caracteres que representa a CHAVE, usada para codificar e para decodificar o texto. É como na opção 3, porém ao invés de somar valores pré-definidos, é o usuário que informa a sequência de valores. Vamos usar uma chave de 5 caracteres. Os 5 caracteres serão dígitos e serão usados como valores a serem somados.
Por exemplo: CHAVE "13524" irá gerar a sequência de somas cíclicas de +1,+3,+5,+2,+4.

-4 => Descriptografa: Usa a chave de criptografia informada para descriptografar o texto. Esta opção é o inverso da opção 4, e por usar uma chave simétrica, é adotada a mesma chave usada para criptografar, para então descriptografar o texto.

(*) Nota: obviamente que em sistemas "mais seguros" a chave criptográfica adotada poderia ser mais longa e composta de caracteres que não sejam restritos à dígitos.

(**) Nota: Chave de criptografia simétrica: consulte wikipedia https://en.wikipedia.org/wiki/Key_(cryptography)

5 => Criptografa: Usa a mesma técnica da opção 4, porém com a inserção de caracteres no meio do texto, para dificultar a decodificação do mesmo. A cada 3 caracteres, será inserido um novo caractere, no caso um '*' (asterisco). Usa chave simétrica de 5 dígitos.

(*) Nota: poderia ser inserido qualquer caractere aleatório, pois depois será descartado, porém, para que possamos corrigir a saída gerada no Run.Codes vamos adotar o caractere '*' para poder avaliar se foi inserido corretamente.

-5 => Descriptografa Usa a chave de criptografia informada para descriptografar o texto. Além disto, "pula" (ignora) o caractere inserido a cada 3 caracteres.

Observações importantes sobre o funcionamento do programa:

A) O programa recebe, como se fossem digitados pelo teclado (stdin) os seguintes dados:
Opções 1, 2, 3 e -1, -2, -3
<nro_da_opção> [enter]
<texto_até_1024_caracteres> [enter]
Opções 4, 5 e -4, -5
<nro_da_opção> [enter]
<texto_com_chave_de_5_dígitos> [enter]
<texto_até_1024_caracteres> [enter]

B) O programa exibe na saída (stdout) o resultado da operação realizada sobre o texto, seja o texto criptografado ou descriptografado (conforme a opção).

O texto deve usar caracteres ASCII com valores de 8 bits por caractere (0 a 255). As entradas de teste serão realizadas de forma a evitar "problemas" com o uso de caracteres não padrão (ASCII Estendido). Portanto, não se preocupe, os casos de teste devem estar restritos a caracteres ASCII padrão entre SPACE ' ' (valor ASCII = 32) e 'z' minúsculo (valor ASCII = 122).

Lembre-se que a chave de criptografia simétrica é um texto, ou seja, usa dígitos '0' (valor ASCII 48) a '9' (valor ASCII 57) que devem ser convertidos em valores numéricos de 0 a 9.

C) O trabalho TP-01 deverá ser implementado usando os conceitos de modularidade e abstração vistos em aula, podendo ser feito a partir dos TADs disponibilizados aos alunos (TAD LDES, TAD LDED como os TADs do Livro do Backes, ou outros similares), sendo programado em linguagem "C" (enviar .c se for um arquivo único, ou, .zip se usar Makefile com mais arquivos – projeto completo .c, .h). **Tem que ser com lista dinâmica!**

>> O trabalho deve ser entregue via RUN.CODES

>> A definição do trabalho e exemplos de casos de teste estão disponíveis na página da Wiki:

http://wiki.icmc.usp.br/index.php/TrabPrat_SSC0603_2020(fosorio)

>> Exemplos de Dados dos Casos de Teste:

CASO 1
Entrada:
```
1
HELLO
```
Saída: `IFMMP`

CASO 2
Entrada:
```
-1
IFMMP
```
Saída: `HELLO`

CASO 3
Entrada:
```
2
HELLO
```
Saída: `KHOOR`

CASO 4
Entrada:
```
3
HELLO
```
Saída: `IGOMQ`

CASO 5
Entrada:
```
4
13524
HELLO WORLD! 01234567890 BYE BYE.
```
Saída: `IHQNS!ZTTPE$%253697:8;>2$C\J"FZH3`

CASO 6
Entrada:
```
-4
13524
IHQNS!ZTTPE$%253697:8;>2$C\J"FZH3
```
Saída: `HELLO WORLD! 01234567890 BYE BYE.`

CASO 7
Entrada:
```
5
13524
HELLO WORLD! 01234567890 BYE BYE.
```
Saida: `IHQ*NS!*ZTT*PE$*%25*369*7:8*;>2*$C\*J"F*ZH3`

CASO 8
Entrada:
```
-5
13524
IHQ*NS!*ZTT*PE$*%25*369*7:8*;>2*$C\*J"F*ZH3
```
Saida: `HELLO WORLD! 01234567890 BYE BYE.`