UFSM
Estruturas de Dados e Programação
Professor: Jonas Bulegon Gassen

Exercícios sobre Strings

1. Desenvolva um algoritmo com funções para ler uma string e aplicar a cifra de Cesar para criptografá-la. Esta técnica se baseia na substituição de cada letra da mensagem por outra a frente dela no alfabeto um número fixo de posições. Exemplo: CASA. Aplicando a técnica, substituindo por 3 posições posteriores, C → F, A → D, S → V. Logo, a saída será: FDVD.

    1. O usuário deve definir o número de posições para criptografia. Não esqueça de tratar casos especiais, como as últimas letras do alfabeto.
    2. O programa deve possibilitar o processo inverso para a mensagem.

2. Desenvolva um algoritmo para ler pelo menos duas strings e apresentá-las em ordem alfabética.

3. Crie um programa que leia duas strings (A e B) e mostre uma terceira string (C) formada pelos caracteres contidos em A e B de forma intercalada. Exemplo: Se A = “quarta” e B = “segunda”, a resposta obtida deverá ser “qsueagrutnada”.

4. Crie uma função que receba uma string e altere a primeira letra de cada uma das palavras para maíuscula. Dica, use a tabela ASCII para conversão para maiúscula.

5. Crie uma função que mascare caracteres **NUMÉRICOS** de uma string. A função deve receber a string e dois parâmetros numéricos. O primeiro define o início de onde **NÃO** será mascarado e o tamanho desse intervalo.
   Por exemplo, para a chamada de função com os parâmetros: “991.614.580-66”, 3, 5
   Resultado: ***.614.58*-**

6. Escreva uma função receba uma string por parâmetro. A função deverá verificar se a string recebida é um endereço da web válido (simplificado para o exercício), exemplos:

   https://www.google.com.br ou https://www.ufsm.br

   Se o endereço for válido, a função deve retornar o número 1, caso contrário, 0. Regras: iniciar por “https://“, conter pelo menos 3 letras (apenas letras) quaisquer após o início e antes do primeiro “.”. Apenas letras até o próximo “.”. Após o segundo ponto, conter pelo menos duas letras. Se houver mais caracteres após estes dois últimos, devem ser apenas letras ou “.”. Considerar apenas letras minúsculas. Dica: A tabela ASCII pode ser útil.

7. Faça uma função que verifique se uma string é valida com base em um padrão informado pelo usuário, por exemplo

   Entrada 1: “991.614.580-66”, {{“0-9”},{“.”},{“-”}} (Padrão é uma matriz de strings)
   Resultado: 1 (Válida)

   Entrada 2: “991.61**a**.580-66”, {{“0-9”},{“.”},{“-”}} (Padrão é uma matriz de strings)
   Resultado: 0 (Inválida)

   Entrada 3: “991.614.580-66”, {{“0-9”},{“.”},{“*”}} (Padrão é uma matriz de strings)
   Resultado: 1 (Inválida)

   Assuma que o padrão sempre terá 3 caracteres “0-9”, “a-z” para intervalos ou um caractere “.”, “-“ para caracteres individuais.

   Dicas:
   - Use a tabela ASCII para pegar os intervalos dos padrões e também para comparar.
   - typedef struct {
       int ini;
       int fim;
   } Intervalo;

   Crie uma estrutura como essa para decodificar os padrões e depois compará-los. Você precisará construir os dados de tipo Intervalo com base na matriz de regras e depois compará-los com a string a ser testada. Faça uma verificação caractere a caractere.