Departamento de Ciência da Computação – IME-USP

MAC 115 – Introdução à Computação para Ciências Exatas e Tecnologia

Escola Politécnica – Primeiro Semestre de 1997

Terceiro Exercício-Programa — Data de entrega: até 22 de maio de 1997.

Turmas: Produção, Civil A, Civil C, Minas/Metal/Materiais

MANIPULAÇÃO DE CADEIAS DE CARACTERES

Neste exercício-programa você vai escrever em C algumas funções de manipulação de strings (cadeias de caracteres) que servirão bastante para o quarto exercício-programa, quando você vai fazer transformações em um arquivo de texto.

As funções que você deve escrever em C são as seguintes :

1) int localiza (char s[ ], char t[ ])

Esta função recebe dois strings **s** e **t** e procura a *ocorrência* de **t** em **s** mais à esquerda. Para entender o que é ocorrência de um string em outro, repare nos seguinte exemplos :

(i) o string GRAMA ocorre uma vez no string PROGRAMAÇÃO, na posição 3 (observação importante: lembre que em C os vetores – um string é um vetor de caracteres – têm os elementos numerados a partir do 0);

(ii) o string ANA ocorre duas vezes no string BANANA (a primeira na posição 1 – BANANA – e a segunda na posicão 3 – BANANA) ;

(iii) o string CHATO não ocorre nenhuma vez no string PROGRAMAÇÃO.

A função deve retornar a posição da primeira ocorrência (isto é, a mais à esquerda) se houver alguma, ou então -1 caso não haja nenhuma.

Assim, a função deve retornar 3 no caso (i), 1 no caso (ii), e -1 no caso (iii).

2) int remove (char s[ ], int pos, int num)

Esta função recebe um string **s** e dois inteiros **pos** e **num**, e remove **num** caracteres de **s**, começando na posição **pos** (deslocando para a esquerda os caracteres porventura existentes em **s** a partir da posição **pos+num**), se **pos+num** for menor ou igual ao comprimento de **s** — neste caso a função retorna 0, para sinalizar que não houve erro. Caso contrário (isto é, não sendo possível remover **num** caracteres de **s** a partir da posição **pos**), a função deixa **s** do jeito que está, retornando um código de erro que pode ser :

1 - se não existe a posição **pos** em **s** ;

2 - se existe a posição **pos** em **s**, mas **pos+num** é maior que o comprimento de **s** (e portanto, não dá para remover **num** caracteres a partir da posição **pos**).

Exemplos :

i) se **s** tem como conteúdo "Gooool !", a chamada **remove (s, 2, 3)** deixará **s** com conteúdo "Gol !", retornando 0;

ii) se **s** tem como conteúdo "Gol !", a chamada **remove (s, 5, 3)** deixará **s** com conteúdo "Gol !", retornando 1;

iii) se **s** tem como conteúdo "Gol !", a chamada **remove (s, 2, 5)** deixará **s** com conteúdo "Gol !", retornando 2.

3) int insere (char s[ ], char t[ ], int pos)

Esta função recebe dois strings **s, t** e um inteiro **pos**, e insere na posição **pos** de **s** o string **t**, deslocando para a direita os caracteres porventura existentes no string **s** a partir da posição **pos**, se esta for menor ou igual ao comprimento de **s** — neste caso a função retorna 0, para sinalizar que não houve erro. Caso contrário (isto é, se **pos** é maior que o comprimento de **s**), a função deixa **s** como está, retornando 1 como código de erro.

Exemplos :

i) se **s** tem como conteúdo "A,E,I são vogais", e **t** tem como conteúdo ",O,U" , então a chamada **insere (s, t, 5)** deixará **s** com conteúdo "A,E,I,O,U são vogais", retornando 0;

ii) se **s** tem como conteúdo "Nenhuma novidade", e **t** tem como conteúdo ", por enquanto", então a chamada **insere (s, t, 16)** deixará **s** com conteúdo "Nenhuma novidade, por enquanto", retornando 0;

iii) se **s** tem como conteúdo "Nenhuma novidade", e **t** tem como conteúdo ", por enquanto", então a chamada **insere (s, t, 17)** deixará **s** com conteúdo "Nenhuma novidade", retornando 1.

4) int substitui_primeira (char s[ ], char t[ ], char r[ ])

Esta função recebe três strings **s, t, r**, e faz o seguinte :

i) procura **a primeira** ocorrência de **t** em **s** ;

ii) se não houver ocorrência, não faz mais nada, devolvendo -1 ;

iii) caso contrário, substitui **esta primeira** ocorrência de **t** em **s** pelo string **r**, devolvendo a posição onde ocorreu a substituição.

Exemplos :

i) se **s** tem como conteúdo "A,...,U são vogais", **t** tem como conteúdo "...", e **r** tem como conteúdo "E,I,O", então a chamada **substitui_primeira (s, t, r)** deixará **s** com conteúdo "A,E,I,O,U são vogais", e devolverá 2;

ii) se **s** tem como conteúdo "Goool !", **t** tem como conteúdo "oooo", e **r** tem como conteúdo "o", então a chamada **substitui_primeira (s, t, r)** deixará **s** com conteúdo "Goool !" (ou seja, o conteúdo de **s** não foi alterado, pois "oooo" não ocorre em **s**), e devolverá -1.

Atenção : nesta função você deve usar as funções anteriores.

5) int substitui_todas (char s[ ], char t[ ], char r[ ])

Esta função recebe três strings **s, t, r**, e substitui **todas** as ocorrências do substring **t** no string **s** pelo substring **r**. Para isso, basta chamar a função anterior até que ela devolva -1 (ou seja, até que não exista mais ocorrência de **t** em **s**). A função deve retornar o número de substituições feitas em **s**.

Exemplos :

i) se **s** tem por conteúdo "Tristão, porque teu nome é Tristão ?", **t** tem por conteúdo "Tristão", e **r** tem por conteúdo "Isolda", então a chamada **substitui_todas (s, t, r)** deixará **s** com conteúdo "Isolda, porque teu nome é Isolda ?", retornando 2;

ii) se **s** tem por conteúdo "cada caso é um caso", **t** tem por conteúdo "caso", e **r** tem por conteúdo "descaso", então a chamada **substitui_todas (s, t, r)** deixará **s** com conteúdo "cada descaso é um descaso", retornando 2.

Atenção : nesta função você deve usar a função anterior, mas é preciso tomar cuidado — veja o exemplo acima : se você chamar substitui_primeira passando **s, t, r** como parâmetros, você terá **s** mudado para "cada descaso é um caso", se você chamar outra vez substitui_primeira passando

s, t, r como parâmetros, você terá **s** mudado para "cada desdescaso é um caso", e assim você não acaba nunca, pois **s** tomará a forma "cada desdes...descaso é um caso". Para evitar isso, será preciso, ao passar **s** como parâmetro, avançar a sua base para a primeira posição após a última substituição operada.

Finalmente, faça um programa (isto é, uma função *main*) só para testar as funções que você escreveu, testando a função substitui_todas. O programa deve ler um string "principal" **s**, e dois strings **t, r**, a fim de substituir por **r** todas as ocorrências de **t** em **s**, e depois mostrar como ficou o string **s**. Teste seu programa com todos os exemplos dados neste enunciado.

IMPORTANTE: Todo exercício-programa deve seguir as observações contidas no folheto distribuído juntamente com o primeiro exercício-programa. Neste folheto estão descritas as diretrizes para forma de entrega do exercício, aspectos importantes na avaliação, etc.