#### 

#### UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE – UFRN

###### INSTITUTO METRÓPOLE DIGITAL

IMD0012 – Introdução às Técnicas de Programação

Professores: Julio Melo e Isaac Franco

Lista de exercícios 4

Para essa lista, verifique a diferença entre as palavras chave ***Função*** e ***Programa***. Quando a questão pedir para escrever uma ***Função***, escreva apenas a função de acordo com os requisitos. O código do main só é necessário quando a questão pedir para escrever um ***Programa***.

**Strings e Vetores**

No jargão de linguagem de programação uma *string* é, basicamente um vetor de caracteres que contenha, na sua última posição o elemento ‘\0’, chamado de finalizador de *strings*. Dessa forma, os programas podem trabalhar supondo, sempre, que as *strings* têm esse elemento para marcar seu fim.

Sabendo que as strings são vetores, é seguro afirmar que, para guardar a string “abcd” um vetor de caracteres precisa de 5 elementos, e não 4, como seria natural se esperar. Naturalmente, o compilador sabe dessa informação (é necessário por o ‘\0’ no fim da string) e faz isso automaticamente quando o programa declara: char \*string = “abcd”. Dessa forma, programas e funções que processam strings, não precisam receber argumentos adicionais informando seu tamanho, assim como era feito nos vetores e matrizes.

De porte dessa informação resolva as questões abaixo:

**QUESTÃO 1:**

Implemente os requisitos abaixo:

1. Implemente a função int tamanho(char \* string), que computa e retorna o tamanho de uma *string*. (O tamanho de uma *string* é a quantidade de elementos que existe antes do ‘\0’).
2. Implemente a função int compara(char \*stringA, char \*stringB), que retorna 1 caso as duas *strings* sejam iguais e 0 caso contrário. (Devido ao fato de serem ponteiros, duas *strings* não podem ser comparadas usando o operador ‘==’, caso seja usado, apenas os ponteiros serão comparados, dessa forma, para essa questão é necessário iterar em todos os elementos da *string* comparando caractere à caractere).
3. Implemente a função void copia(char \*dest, char \*base), que copia a string base para a string dest. (Da mesma forma que o ítem anterior, a cópia de strings deve ser feita caractere à caractere, caso contrário apenas os ponteiros serão copiados).
4. Implemente um programa principal para testar as funções tamanho, compara e copia. (Veja que, no caso da função copia, você precisará declarar a string base e outra string dest, com tamanho suficiente para caber a string base).

**Biblioteca ctype.h**

As funções para checagem de caracteres, as funções nessa biblioteca definem funções utilitárias para checagem de caracteres. Todas as funções recebem um caractere como argumento e retornam valores de acordo com a necessidade, abaixo, seguem algumas funções úteis dessa biblioteca:

|  |  |
| --- | --- |
| Função | Descrição |
| int isspac(char ch) | Retorna 1 se o caractere ch é um caractere é um separador de palavras (\n, \t, espaço em branco, etc). |
| int isdigit(char ch), int isalpha(char ch), int isalnum(char ch), int isxdigit(char ch) | Retorna 1 se o caractere ch for um digito (digit), uma letra (alpha), um caractere ou uma letra (alnum) ou um caractere hexadecimal (isxdigit). |
| int islower(char ch), int isuper(char ch) | Retorna 1 se o caractere ch estiver maiúsculo ou miniusculo |
| int touper(char ch), int tolower(char ch) | Retorna o valor do caractere ch em maiúsculo ou minúsculo. |

**QUESTÃO 2:**

Implemente os requisitos abaixo:

1. Implemente função que receba como argumento uma string e teste se a string é um número ou não (leve em conta números no formato decimal, como 10.5, ou inteiro 188899). A função deve retornar 1 caso a string represente um número e 0, caso contrário.
2. Implemente uma função com a seguinte assinatura: void toLower(char \*src, char \*dest). A função deve processar a string *src*, e armazenar na string dest outra string porém apenas com caracteres miúsculos. Ex: se src for igual à “ABCde”, a string dest deve ser “abcde”.
3. Implemente a função com a seguinte assinatura: void toLowerUper(char \*sr, char \*dest). A função deve processar a string src, de forma que palavras separadas por espaço comecem com letras maiúsculas. Ex: se src for igual a “eu sou julio”, a string dest deve ser “Eu Sou Julio”, porém se src for igual à “EU sou julio”, a string dest deve ser “Eu Sou Julio”.
4. Implemente um programa para testar as funções implementadas em a, b e c.

**Funções de conversão de strings para números**

Uma operação bem comum, na prática de processamento de strings é a conversão de strings para números ou de números para strings. Nesse caso existem muitas funções para realizar o trabalho, além de muitas bibliotecas. No entanto, duas funções bem flexíveis e disponíveis na biblioteca padrão (stdio.h) são:

|  |  |
| --- | --- |
| Função | Descrição |
| int sprintf(char \*s, const char \*format, ....) | Funciona como o printf, porém imprime os valores na string s, ao invés de imprimir na saída padrão, da mesma forma que o printf, você pode imprimir inteiros, floats, strings e outros dependendo apenas da string format. Note que a string s precisa ser declarada à priori e ter espaço suficiente para armazenar a string desejada. |
| int sscanf(char \*s, const char\* format, ....) | Funciona como o scanf, porém lê os valores da string s ao invés de ler da saída padrão, da mesma forma que o scanf, você pode armazenar os elementos lidos em ints, floats, strings e outros, dependendo apenas da string format. |

Exemplo de código:

|  |
| --- |
| #include <stdlib.h>  #include <stdio.h>  int main**(){**  char **\*** num\_str **=** "10.5"**;**  float num\_float**;**  /\*tranforma a string "10.5" em um float\*/  sscanf**(**num\_str**,** "%f"**,** **&**num\_float**);**  printf **(**"Eu li a string %s e transformei no número %f\n"**,** num\_str**,** num\_float**);**  /\*tranforma o int 10 em uma string\*/  int num\_int **=** 10**;**  /\*str\_restult tem que ser grande suficiente para  caber a string resultante da conversão, nesse caso  terá 3 caracteres (1, 0 e \0)\*/  char str\_result**[**20**];**  sprintf**(**str\_result**,** "%d"**,** num\_int**);**  printf **(**"Li o numero %d e transformei na string %s\n"**,** num\_int**,** str\_result**);**  **return** 0**;**  **}** |

**QUESTÃO 3:**

Implemente os requisitos abaixo:

1. Implemente uma função que processe strings no formato: “<numero1> <op> <numero2>”, onde op pode variar entre >, >=, <=, <. A função deve processar a string e retornar 1 ou 0 dependendo do valor de <op>. ex: “10 > 30”, e retorna 0, pois o numero1 é maior que numero2, “10 <= 10”, retorna 1, pois o número1 é menor ou igual a numero2.
2. Implemente uma função que tenta a seguinte assinatura: void pessoaToString (char \*nome, int idade, char sexo, char \*dest). A função deve receber os argumentos e colocar na string *dest* o valor “Meu nome é <nome>, minha idade é <idade> e meu sexo é <sexo>”, exemplo voidPessoaToString (“Julio”, 31, ‘m’, dest), nesse caso *dest* é igual a: “Meu nome é Julio, minha idade é 31 e meu sexo é m”.
3. Escreva uma função que receba um vetor de inteiros, seu tamanho, e o converta em uma string. A função deve ter a assinatura: void intVecToString(int tam, int vec[tam], char\* dest). A função deve processar o vetor vec e colocar em dest uma string no formato “[ <num1> <num2> ... <num\_tam>]”. Exemplo: caso vec = { 0, 2, 3 }, a função seria chamada com intVecToString(3, vec, dest), nesse caso o valor de dest seria “[ 0 2 3 ]”
4. Implemente um programa para testar as funções implementadas em a, b e c.

**Biblioteca string.h**

As funções, tamanho, compara e copia implementadas nos exercícios anteriores são apenas três das muitas funções que alguém pode precisar para manipular strings. Por isso, algumas funções já são predefinidas dentre as bibliotecas padrão da linguagem C.

Essas funções são definidas na biblioteca string.h, dentre suas muitas funções a tabela abaixo lista as que usaremos mais na disciplina:

|  |  |
| --- | --- |
| Função | Descrição |
| size\_t strlen(const char \*str) | (**str**ing **len**gth) Calcula o tamanho da string str e retorna o valor. |
| char \*strcpy(char \*dest, const char \*src) | (**str**ing **c**o**py**) Copia os caracteres na string src, para a string dest. Essa função retorna um ponteiro para a string dest. |
| char \*strncpy(char \*dest, const char \*src, size\_t n) | (**str**ing **n c**o**py**) Copia os n caracteres da string src, para a string dest. Essa função retorna um ponteiro para a string dest. |
| int strcmp(const char \*str1, const char \*str2) | (**str**ing **comp**are) Compara as duas strings str1 e str2. Caso str1 seja igual à str2, retorna 0; Caso str1 seja, lexicograficamente, menor que str2, retorna -1; Caso str1 seja, lexicograficamente, maior que str2, retorna 1. Note que a função retorna 0 apenas quando as strings são iguais, o que implica que nos outros casos elas são, obrigatoriamente, diferentes. |
| int strncmp(const char \*str1, const char \*str2, size\_t n) | (**str**ing **n** **comp**are) Similar à strcmp, porém compara apenas os n primeiros caracteres das strings. |
| char \*strcat(char \*dest, const char \*src) | (**str**ing con**cat**enate) Concatena (junta) a string src no final da string dest. Retorna um ponteiro para a string resultante. |
| char \*strncat(char \*dest, const char \*src, size\_t n) | (**str**ing **n** con**cat**enate) Similar à strcat, porém concatena apenas os n primeiros caracteres de src em dest. |
| char \*strstr(const char \*str1, const char \*str2) | (**Str**ing in **str**ing) Encontra a primeira ocorrência da string str2 na string str1. Retorna um ponteiro apontando para a posição em str1 onde a string encontrada começa ou NULL caso a string str2 não for encontrada em str1. |

**QUESTÃO 4:**

Implemente os requisitos abaixo:

1. Escreva um programa que ordene o seguinte conjunto de palavras {Vela, Google, Trovão, Agricultor, Anchova, Secretário, Ciclopes, Árbitro, Festival, Hollywood} e imprima o conjunto na ordem crescente. Dica: utilize o strcmp para decidir qual palavra é “maior” que a outra.
2. Escreva uma função com a seguinte assinatura: void sortString(char \*strings[50], char \*sorted[50]) a função recebe o conjunto *strings* e deve ordená-lo e armazenar o resultado do conjunto ordenado no parâmetro *sorted*. (dica: use strcpy para copiar uma string de um conjunto para outro, e use a função strcmp para comparar as palavras como no item a).
3. Escreva uma função com a seguinte assinatura: void subString(char \*src, int start, int end, char \*substr). A função deve processar uma substring da string src, que inicia na posição start e termina na posição end, após isso a função deve armazenar o resultado no parâmetro *substr*.
4. Escreva um programa que gere uma frase aleatória a partir de um conjunto aleatório de palavras. O programa deve conter um conjunto de palavras, na forma de um vetor de strings, e deve pedir ao usuário a quantidade, n, de palavras para a frase gerada. Depois disso o programa deve gerar uma frase aleatória, armazenar a frase em uma string, frase\_aleatoria, e imprimi-la. Dica: Declare um vetor de strings e use a função rand para acessar o vetor aleatoriamente. Dica2: Usa a função strcat para concatenar as palavras sorteadas aleatoriamente do vetor de palavras.
5. Escreva uma função a seguinte assinatura: int stringnfind(char \*str1, char \*str2). A função deve computar todas as ocorrências da string *str2* na string *str1* e retornar a quantidade de vezes que str2 se repete em str1. Ex: caso str1 for “**m**eu co**m**putador” e str2 for “m”, a função deve retornar 2. Caso str1 for “Passo meu **Natal** em **Natal**, pois sou **Natal**ense” e str2 for “Natal”, a função deve retornar 3. Dica: Use a função strstr, e use o retorno da função para continuar a busca.

**QUESTÃO 5:**

Implemente os quesitos abaixo:

1. Implemente uma função com a seguinte assinatura: int startsWith(char \*src, char \*starting). A função deve retornar 1 quando a string *src* começa com a string *starting*.
2. Implemente uma função com a seguinte assinatura: int endsWith(char \*src, char \*ending). A função deve retornar 1 quando a string *src* termine com a string *ending*.

**QUESTÃO 6:**

Implemente uma função, e um programa para testá-la, que receba como argumento uma string e compute um vetor de strings contendo todas as sub-strings que são separadas por um determinado delimitador. A assinatura da função deve ser a seguinte: void split(char\* string, char\* delim, char \*ret[20], int \*sizeRet).

Nesse caso, suponha que a string seja:

string = “int main(){ int a = 10; int b = 20; int c = 30; a = a + b + c; }”

Se a função for chamada split(string, “;”, ret, size), o vetor *ret* e o valor de *sizeRet* devem conter:

ret[0] = “int main(){ int a = 10”

ret[1] = “ int b = 20”

ret[2] = “ int c = 30”

ret[3] = “ a = a + b + c”

ret[4] = “ }”

sizeRet = 5

**(Extra, valendo 0,5 pontos) QUESTÃO 7:**

Implemente os quesitos abaixo:

1. Implemente um programa que compute a quantidade de palavras com um mesmo tamanho em um texto. O programa deve receber como entrada um texto, no formato char\*, e deve computar uma matriz onde cada linha representa um tamanho de palavra, a primeira coluna representa a quantidade de ocorrências, e a segunda coluna a porcentagem de ocorrências.
2. Implemente um programa que compute a quantidade de vezes que uma mesma palavra se repete em um texto. Gere uma tabela semelhante ao item a, porém com as linhas representando cada uma das palavras. Dica: Primeiro separe o texto em um vetor contendo cada uma das palavras (mesmo as repetidas); depois transforme esse vetor em um conjunto, sem palavras repetidas; por fim processe o texto com as palavras do conjunto, construindo os valores da tabela.

**Exemplo para o item a**

Entrada:

“Lorem ipsum dolor sit amet, consectetur adipiscing elit.”

Matriz de tamanhos:

|  |  |  |
| --- | --- | --- |
| Índice | Qtdade | % |
| 0 | - |  |
| 1 | 0 | 0 |
| 2 | 0 | 0 |
| 3 | 1 | 12,5% |
| 4 | 2 | 25% |
| 5 | 3 | 37,5% |
| ... | 0 | 0 |
| 10 | 1 | 12,5% |
| 11 | 1 | 12,5% |

**Exemplo para o item b**

Entrada:

“Proin sapien quam, porttitor id diam a, scelerisque auctor diam. Sed vel elementum libero, non pellentesque nibh. Fusce venenatis magna a sem hendrerit fringilla. Fusce cursus nunc metus, eu maximus elit dictum id. Sed mauris.”

Conjunto = {“Proin”, “sapien”, “quam”, “porttitor”, “id”, “diam”, “a”, “scelerisque”, “auctor”, “Sed”, “vel”, “elementum”, “libero”, “non”, “pellentesque”, “nibh”, “Fusce”, “venenatis”, “magna”, “sem”, “hendrerit”, “fringilla”, “Fusce”, “cursus”, “nunc”, “metus”, “eu”, “maxinus”, “elit”, “dictum”, “id”}

Matriz de repetições:

|  |  |  |
| --- | --- | --- |
| Índice | Qtdade | % |
| 0 (“Proin”) | 1 | 2,8% |
| 1 (“sapien”) | 1 | 2,8% |
| 2 (“quam”) | 1 | 2,8% |
| 3 (“porttitor”) | 1 | 2,8% |
| 4 (“id”) | 2 | 5,6% |
| 5 (“diam”) | 1 | 2,8% |
| 6 (“a”) | 2 | 5,6% |
| ... | ... | ... |
| 30 (“id”) | 2 | 5,6% |