UTFPR - Universidade Tecnológica Federal do Paraná
DACOM - Departamento de Computação
BCC31A :: Algoritmos

Exercícios 06 :: Strings

Instruções Gerais

- Faça cada exercício em uma função distinta, deixando-as em um único arquivo (**lista06.c**).
- Utilize a extensão .c, o compilador gcc e o editor de sua preferência: Code Blocks, VS Code, etc.
  - Ao final, compacte o arquivo em ZIP e envie pelo Moodle.
- Você pode utilizar as seguintes funções disponíveis na biblioteca padrão de C:
  - rand() <stdlib.h>
  - printf() e scanf() <stdio.h>
  - strlen(), strcpy(), strcmp() e strcat() <string.h>

1. Escreva uma função que recebe uma string (como parâmetro) e a imprime na tela.

   `void printString(char str[])`

2. Escreva uma função que recebe uma string e a imprime ao contrário. Obs: a função **strlen()** <string.h> poderá ajudar a encontrar a posição do último caractere da string.

   `void printStringReversed(char str[])`

3. Escreva uma função que recebe uma string e imprime seus caracteres separados por asterisco (*****).

   `void printStringSparse(char str[])`

   ```
   Ex: char s[] = “hyphenization”
       printStringSparse(s); // saída: h*y*p*h*e*n*i*z*a*t*i*o*n
   ```

4. Escreva uma função que recebe uma string e imprime seus caracteres separados por um caractere, fornecido como segundo parâmetro da função.

   `void printStringCustom(char str[], char separator)`

   ```
   Ex: char s[] = “hyphenization”
       printStringCustom(s, '$'); // saída: h$y$p$h$e$n$i$z$a$t$i$o$n
   ```

5. Escreva uma função que recebe uma string e imprime:
   a. A quantidade de letras (maiúsculas ou minúsculas);
   b. A quantidade de dígitos;
   c. A quantidade de símbolos.

   ```
   OBS: considere apenas a porção dos caracteres imprimíveis da tabela ASCII, isto
        é, dos índices 32 ao 126.
   ```

   `void stringReport(char str[])`

6. Escreva uma função que recebe uma string e a converte para letras maiúsculas. Atenção: a string poderá conter letras maiúsculas. OBS: observe a diferença de posição entre caracteres maiúsculos e minúsculos na tabela ASCII (ex: 'A' e 'a').

   void stringToUpper(char str[])

   ```
   Ex: char s[] = "All your BASE are Belong to US!";
       stringToUpper(s);
       printf("%s", s); // saída: ALL YOUR BASE ARE BELONG TO US!
   ```

7. A função strcmp(str1, str2) compara duas strings alfabeticamente. Ela devolve:
   a. res < 0 se str1 < str2 << str1 vem antes de str2
   b. res = 0 se str1 = str2
   c. res > 0 se str1 > st2 << str1 vem depois de str2

   Escreva uma função que compara duas strings independente do caso (maiúsculo ou minúsculo). Ela deve retornar os mesmos tipos de valores que strcmp(). Dica: com a função do exercício anterior, você poderá passar ambas strings para maiúsculas e, então, compará-las com strcmp().

   int strcmpNoCase(char s1[], char s2[])

   ```
   Ex: int res = strcmpNocase("JOanna", "joanna"); // res:0 (strings iguais)
   ```

8. Escreva uma função que conta e devolve o número de palavras em uma string. Considere que haverá somente um espaço entre as palavras.

   int countWords(char str[])

   ```
   Ex: char s[] = "first things first, second things latter";
       printf("%d", countWords(s)); // saída: 6
   ```

9. Escreva uma função que conta e devolve o número de palavras em uma string. Considere que poderá haver mais de um espaço entre as palavras, bem como, no início e final da string.

   int countWordsPlus(char str[])

   ```
   Ex: char s[] = "  first     things  first, second  things latter ";
       printf("%d", countWordsPlus(s)); // saída: 6
   ```

10. Escreva uma função que recebe uma string composta de várias palavras. A função deve modificar a letra inicial de cada palavra para maiúscula e, as demais, para minúsculas. Considere que sempre haverá ao menos um espaço entre cada palavra.

    void stringCapitalize(char str[])

    ```
    Ex: char s[] = "welCOME  To     COMPUTER programming!!";
        stringCapitalize(s);
        printf("%s", s); // saída: Welcome  To     Computer Programming!!
    ```

11. Escreva uma função que implementa um comportamento similar à strcmp(). Entretanto, você não deve utilizar strcmp() em sua implementação. A função criada deve retornar:
    a. res = -1 se str1 < str2 << str1 vem antes de str2
    b. res = 0 se str1 = str2
    c. res = 1 se str1 > st2 << str1 vem depois de str2

    int stringCompare(char str1[], char str2[])

12. Escreva uma função que remove os espaços que possam existir antes e depois de uma string.

    void stringTrim(char str[])

    ```
    Ex: char s[] = “    hello world    “;
        stringTrim(s);
        printf(“%s”, s); // saída: “hello world”
    ```

13. Escreva uma função que informa, com 1 (true) ou 0 (false), se uma string está contida em outra.

    int findSubstring(char str[], char sub[])

    ```
    Ex: char s[] = “first things first, second things latter“;
        int check = findSubstring(s, “second”);
        // neste caso, deve devolver 1, pois a string contém a palavra “second”
    ```

14. Escreva uma função que corta uma string após uma dada palavra. As posições após a palavra devem ser preenchidas com o caractere '\0' (int 0), até o final do vetor.

    void cutString(char str[], char target[])

    ```
    Ex: char s[] = “first things first, second things latter“;
        cutString(s, “second”);
        printf(“%s”, s); // s = “first things first, second”
    ```

15. Escreva uma função que converte um número inteiro de até 10 dígitos (parâmetro de entrada **number**) para uma string (parâmetro de saída **converted**). Dica: utilize módulo (%) e divisão (/) para obter os dígitos do número inteiro.

    void intToString(int number, char converted[])

    ```
    Ex: char num[11];
        intToString(512, num);
        printf(“%s”, num); // saída: “512” (string)
    ```

16. Escreva uma função que converte uma string (de qualquer tamanho) para um inteiro. Utilize a notação posicional para montar o número inteiro. Ex: 2506 = $2 \times 10^3 + 5 \times 10^2 + 0 \times 10^1 + 6 \times 10^0$.

    int stringToInt(char textNumber[])

    ```
    Ex: int n = stringToInt(“1024”);
        printf(“%d”, n); // saída: 1024 (inteiro)
    ```