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

Exercícios 08 :: Strings

Instruções Gerais

- Faça cada exercício em uma função distinta, deixando-as em um único arquivo (main.c).
- Ao final, compacte em ZIP e envie pelo Moodle.
- Você pode utilizar as funções: rand() <stdlib.h>, printf() e scanf() <stdio.h>, e as funções da biblioteca <string.h>.

1. Escreva uma função que lê uma string simples (uma palavra) do teclado e a imprime na tela.

   ```
   void printString()
   ```

2. Escreva uma função que recebe uma string e imprime ao contrário. Dica: a função **strlen()** <string.h> será de grande ajuda.

   ```
   void printStringReversed(char str[])
   ```

3. Escreva uma função que imprime os caracteres de uma string separados por um segundo caractere, fornecido.

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

   Ex: char s[] = “hyphenization”
       stringHyphen(s, ‘*’); // saída: h*y*p*h*e*n*i*z*a*t*i*o*n*
   ```

4. Escreva uma função que imprime a quantidade de letras, dígitos e caracteres especiais em uma string. Lembre-se da tabela ASCII.

   ```
   void stringReport(char str[])
   ```

5. Escreva uma função que recebe uma string e a converte para letras maiúsculas. Atenção: a string pode já conter letras maiúsculas. Dica: 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!
   ```

6. 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 stringCompareNoCase(char s1[], char s2[])

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

7. 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
    ```

8. 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
    ```

9. 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!!
    ```

10. Escreva uma função que implementa o comportamento da strcmp(). Devolve -1, 0 ou 1.

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

11. 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"
    ```

12. Escreva uma função que informa, com 1 ou 0, se uma string está contida em outra.

```
int stringContains(char str[], char sub[])

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

13. Escreva uma função que corta uma string após uma dada palavra. A posição imediatamente após a palavra deve ser definida como ‘\0’.

```
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”
```

14. Escreva uma função que busca e remove uma substring dentro de outra string.

```
void removeSubstring(char str[], char sub[])

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

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 x 10^3 + 5 x 10^2 + 0 x 10^1 + 6 x 10^0.

```
int stringToInt(char textNumber[])

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

17. Escreva uma função que conta as ocorrências de cada letra em um texto.

    Dica: de forma similar à contagem das ocorrências em um vetor de inteiros, utilize um vetor auxiliar de tamanho 256 como uma tabela para contar as ocorrências de cada caractere (ASCII: http://www.asciitable.com/) . Lembre-se que cada caractere pode ser tratado como um int, que indica sua posição na tabela ASCII (de 0 à 255). Observe que aqui também existe o problema de termos caracteres maiúsculos ou minúsculos no texto.

    Desafio: Numa solução mais otimizada em termos de uso de memória extra, a tabela de ocorrências pode ter apenas 126-32+1 = 95 elementos, uma vez que os caracteres imprimíveis estão no intervalo 32 (espaço) ao 126 (~).

    ```
    void lettersFrequency(char str[])

    Ex: char s[] = “bacon and EGGS”;
        Saída:
           : 2        a: 2        b: 1        c: 1        d: 1        e: 1
          g: 2        n: 2        o: 1        s: 1
    ```