Universidade do Estado do Amazonas - UEA
Escola Superior de Tecnologia - EST
Núcleo de Computação

Linguagem de Programação 2

Lista de Exercícios 7

Prof. Flávio José Mendes Coelho

Funções, caracteres e strings

Nesta lista você poderá fazer uso de uma ou mais das seguintes funções da biblioteca `ctype.h` (o parâmetro `int c`, na verdade, é um caracter, e todas as funções iniciando com `is` retornam *true* (1), se a resposta à pergunta for um sim, ou *false* (zero), em caso contrário):

Função	**O que faz?**
`isalnum(int c)`	`c` é um caractere alfanumérico?
`isalpha(int c)`	`c` é um caractere alfabético?
`isblank(int c)`	`c` é um caractere em branco (espaço ou tab)?
`isdigit(int c)`	`c` é um dígito?
`islower(int c)`	`c` é uma letra minúscula?
`ispunct(int c)`	`c` é um simbolo de pontuação (exceto espaço ou alpanumérico)?
`isspace(int c)`	`c` é um caractere de espaço em branco (espaço, nova linha, return, tab, etc.)?
`isupper(int c)`	`c` é uma letra maiúscula?
`int tolower(c)`	retorna o equivalente minúsculo de `c`.
`int toupper(c)`	retorna o equivalente maiúsculo de `c`.

1. Escreva uma função que imprima todos os caracteres comuns de um teclado e seus códigos ASCII (ou Unicode).

2. Escreva uma função que receba um caractere e retorne *true* se ele for minúsculo, ou retorne *false*, em caso contrário, sem utilizar a biblioteca `ctype.h`.

3. Escreva uma função que receba um caractere e retorne *true* se ele for maiúsculo, ou retorne *false*, em caso contrário, sem utilizar a biblioteca `ctype.h`.

4. Escreva uma função que receba um caractere e retorne *true* se ele for um dígito, ou retorne *false*, em caso contrário, sem utilizar a biblioteca `ctype.h`.

5. Escreva uma função que receba um caractere e retorne *true* se ele for um símbolo de pontuação, ou retorne *false*, em caso contrário, sem utilizar a biblioteca `ctype.h`.

6. Escreva uma função que receba um caracter e retorne uma cópia em minúsculo do caractere, se este for uma letra, ou retorne o próprio caractere, em caso contrário.

7. Escreva uma função que receba um caracter e retorne uma cópia em maiúsculo do caractere, se este for uma letra, ou retorne o próprio caractere, em caso contrário.

8. Escreva uma função que receba como parâmetro uma *string* e que a imprima de forma invertida, isto é, do último caractere até o primeiro.

9. Escreva uma função que receba uma *string* como parâmetro e que imprima somente as vogais da *string*.

10. Escreva uma função que receba uma *string* como parâmetro e que imprima somente suas consoantes.

11. Escreva uma função que receba uma *string* como parâmetro e que retorne seu tamanho (comprimento).

12. Escreva uma função que receba uma *string* e dois valores inteiros positivos a e b como parâmetros, e imprima a *substring* que vai da posição a até a posição b, se $a < b$, ou imprima um espaço em branco se $b < a$.

13. Crie uma nova versão da função do exercício anterior, adicionando um parâmetro do tipo *string* que armazene a *substring* obtida na função. Note que a segunda *string* será passada por referência.

14. Escreva uma função que receba duas *strings* s_1 e s_2, e que copie s_2 para s_1, sem espaços em branco. Note que a segunda *string* será passada por referência.

15. Crie uma nova versão da função do exercício anterior acrescentando um parâmetro do tipo `char` que conterá o caractere que não deve ser copiado de s_2 para s_1.

16. Escreva uma função que receba duas *strings* s_1 e s_2 de mesmo tamanho, e retorne *true* se ambas forem **iguais**, ou retorne *false*, em caso contrário.

17. Escreva uma função que receba duas *strings* s_1 e s_2, e retorne *true* se ambas forem **iguais**, ou retorne *false*, em caso contrário.

18. Escreva uma função que receba duas *strings* a e b, e retorne *true* se ambas forem **inversas**, ou retorne *false*, em caso contrário. Por exemplo, "papagaio" e "oiagapap" são inversas.

19. Uma palavra ou frase é um **palíndromo** se puder ser lida da esquerda para a direita, ou da direita para a esquerda, e significar a mesma palavra ou frase. Por exemplo, "radar" e "roma é amor" são palíndromos. Escreva uma função que receba uma *string* s e retorne *true* se s for um **palíndromo**, ou retorne *false*, em caso contrário.

20. Escreva uma função que receba duas *strings* p e q, e que verifique se q ocorre ou não em p. Retorne *true* ou *false*, dependendo da verificação.

21. Escreva uma função que receba uma *string* s como parâmetro e retorne uma cópia de s convertida em caracteres maiúsculos.

22. Escreva uma função que receba uma *string* s como parâmetro e retorne uma cópia de s convertida em caracteres minúsculos.

23. Faça uma versão da função do exercício anterior que não retorne valor. É necessário criar mais um parâmetro?

24. Desenvolva um programa que leia três *strings* de mesmo comprimento, e um número inteiro positivo i, e imprima, em sequência, o i-ésimo caractere de cada uma das *strings*. Note que o valor de i não pode ser maior nem igual ao comprimento das *strings*. Utilize a técnica de *refinamentos sucessivos* para criar funções para resolver o problema.

25. Escreva uma função que receba uma *string* e um inteiro positivo k, e retorne o caracter na posição na k da *string*.

26. Escreva um programa que leia um valor inteiro positivo de até três digitos e que imprima o valor por extenso (textualmente). Por exemplo, para a entrada 123, a saída é "cento e vinte três". Utilize a técnica de *refinamentos sucessivos* para criar funções para resolver o problema.

27. A "Cifra de César" é uma técnica simples de criptografia que consiste em substituir cada letra da frase original por uma outra do mesmo alfabeto, obtida saltando-se um certo número fixo de letras após cada letra original. Escreva uma função que receba uma *string* s e um inteiro positivo $k \leq 26$, que retorne uma cópia de s cifrada com a Cifra de César.

Entradas:	**Saídas:**
$s =$ "CIFRA DE CESAR!", $k = 3$	"FLIUD GH FHVDU!"
$s =$ "CIFRA DE CESAR!", $k = 4$	"GMJVE HI GIWEV!"
$s =$ "CIFRA DE CESAR!", $k = 10$	"MSPBK NO MOCKB!"

28. Escreva uma função que funcione como a função `strcat` da biblioteca `string.h` da linguagem de programação **C**. (Note que você precisará criar a *string* concatenada dinamicamente).

29. Escreva uma função que receba três *strings* s_1, s_2 e s_3, e um inteiro $0 \leq k <$ `strlen(`s_1`)`. A função deve dividir a *string* s_1 em duas, de forma que $s_2 = s_1[0..k]$ e $s_3 = s_1[k+1..n-1]$, onde n é o tamanho de s_1. (Note que você precisará criar as *strings* s_2 e s_3 dinamicamente).

Entradas:	**Saídas:**
$s_1 =$"Os sonhos não envelhecem.", $k = 0$	$s_2 =$"Os sonhos não envelhecem.", $s_3 =$" "
$s_1 =$"Os sonhos não envelhecem.", $k = 24$	$s_2 =$" ", $s_3 =$"Os sonhos não envelhecem."
$s_1 =$"Os sonhos não envelhecem.", $k = 8$	$s_2 =$"Os sonhos", $s_3 =$" não envelhecem."

30. Escreva uma função que receba um inteiro positivo $n \leq 26$, que imprima a seguinte pirâmide de letras, conforme o valor de n.

```
a
bb
ccc
dddd
...
```