

## Lista de exercícios #6 – manipulando lógicos e caracteres

Sobre a lista:

Aqui a nossa linguagem algorítmica estruturada de alto nível ganha novos tipos de dados: *caracteres* e *lógicos*, para melhor representar algumas situações-problema da vida real. Assim, além da introdução da tabela ASCII, são introduzidas novas funções para conversão entre inteiros (códigos), e caracteres (símbolo): `chr()` e `ascii()`.

1. [*dois caracteres iguais*] Faça um algoritmo para ler 2 caracteres e informar, ao final, se eles são iguais ou diferentes.
2. [*algum caracter é algarismo*] Faça um algoritmo para ler caracteres e informar, ao final, se pelo menos um dos caracteres digitados foi um dígito numérico. O caracter ‘\_’ indica o fim dos dados.
3. [*quatro caracteres iguais*] Faça um algoritmo para ler 4 caracteres e informar, ao final, se eles são todos iguais ou se há algum diferente dos outros.
4. [*qual tipo do caracter*] Faça um algoritmo que leia caracteres do usuário e, dependendo de qual seja o caracter lido, vá escrevendo na saída as mensagens: “**letra maiúscula**”, “**letra minúscula**”, “**dígito**” ou “**outro símbolo**”. O algoritmo termina quando o usuário digita o caracter de espaço.
5. [*maior caracter se diferentes*] Faça um algoritmo para ler 3 caracteres e informar, ao final, qual é o maior deles, mas apenas se os 3 forem diferentes. Se forem iguais, escreva uma mensagem dizendo que há caracteres repetidos.
6. [*caracteres do mesmo tipo*] Faça um algoritmo para ler 2 caracteres e escrever se eles são ou não são do mesmo tipo. Dois caracteres são do mesmo tipo se eles 2 são letras maiúsculas, se eles 2 são letras minúsculas, se eles 2 são números ou se eles 2 são outros símbolos.
7. [*verifica variedade de caracteres*] Faça um algoritmo para ler caracteres do usuário e informar, ao final, se o usuário digitou pelo menos uma letra maiúscula, uma letra minúscula e um dígito, em qualquer ordem. O caracter ‘\_’ indica o fim dos dados.
8. [*sequência crescente*] Faça um algoritmo para ler caracteres consecutivamente. verifique se cada caracter digitado é sempre maior que o anterior, escrevendo, ao final, “**sucesso**” (em caso de sucesso) ou “**falha**” (em caso contrário). O caracter ‘\_’ indica o fim dos dados. Letras minúsculas e maiúsculas são consideradas diferentes entre si.
9. [*todas as vogais apareceram*] Faça um algoritmo para ler caracteres do usuário e informar, ao final, se o usuário digitou todas as cinco vogais do alfabeto, independente se a vogal for maiúscula ou minúscula. O caracter ‘\_’ indica o fim dos dados.
10. [*sequência alternada*] Faça um algoritmo para ler caracteres consecutivamente. Verifique se cada caracter digitado é, alternadamente, maior e menor que o anterior, escrevendo, ao final, “**sucesso**” (em caso de sucesso) ou “**falha**” (em caso contrário). O caracter ‘\_’ indica o fim dos dados. Lembre-se que o usuário pode começar digitando uma sequência tanto ascendente quanto descendente. Letras maiúsculas e minúsculas são consideradas diferentes entre si.
11. [*frequência de conjuntos*] Faça um algoritmo que verifique quantos caracteres digitados pelo usuário são dígitos, letras sinais de pontuação ou outros caracteres, imprimindo a frequência (em porcentagem) de cada um deles ao final do algoritmo. O caracter ‘\_’ indica o fim dos dados e não deve ser computado na conta.

12. [*sequência de letras e dígitos*] Faça um algoritmo que leia caracteres da entrada e escreva, ao final, uma mensagem dizendo “sucesso” se foram digitados, alternadamente, letras e dígitos (começando por qualquer deles) ou “falha” em caso contrário. Qualquer símbolo diferente de letras e dígitos indica o término do algoritmo.
13. [*o ASCII é múltiplo de M*] Faça um algoritmo para ler caracteres e escrever apenas aqueles cujo código ASCII são um múltiplo de um valor constante  $M$ , já definido no início do algoritmo. Faça o algoritmo parar quando o usuário tiver digitado exatamente 25 caracteres.
14. [*bits 0 e bits 1 do caracter*] Faça um algoritmo para ler um caracter e informar, ao final, quantos bits 0 e quantos bits 1 tem a representação numérica do seu código ASCII na base binária.
15. [*caracter com bits invertidos*] Dado um caracter lido do usuário, produza um algoritmo que encontre o caracter imprimível que é resultante da inversão dos sete bits (onde há 0 para a ser 1 e vice-versa) menos significativos do código ASCII daquele caracter, escrevendo o caracter resultante na saída ou uma mensagem que tal caracter é não-imprimível. Lembre-se que um caracter não-imprimível tem seu código entre 0 e 31.
16. [*bit shift*] Faça um algoritmo para ler um caracter e gerar um outro caracter na saída que seja o resultado da rotação de 1 bit para a esquerda dos sete bits menos significativos do seu código ASCII. Por exemplo, se os 8 bits do código são  $\alpha\beta\gamma\delta\epsilon\zeta\eta\theta$  o código a ser gerado deve ser  $\alpha\gamma\delta\epsilon\zeta\eta\theta\beta$ .
17. [*contém ASCII equidistante*] Faça um algoritmo que leia 3 caracteres do usuário e diga se as distâncias (diferenças) que separam o código de um deles dos outros dois são iguais, escrevendo, neste caso, qual caracter é o equidistante dos outros dois, ou uma mensagem de que tal fato não acontece. Observe que não se sabe em que ordem os caracteres são digitados pelo usuário.
18. [*paridade do código*] Faça um algoritmo que leia um caracter do usuário e escreva uma mensagem dizendo se a paridade dos bits do seu código ASCII é par ou ímpar. Assuma que a paridade de um número é par quando a quantidade de bits iguais a 1 de sua representação na base binária é par.
19. [*modifica caixa do caracter*] Faça um algoritmo que leia caracteres do usuário e, sendo letra maiúscula, escreva a minúscula correspondente e, sendo minúscula, escreva a maiúscula correspondente. Qualquer caracter que não seja uma letra indica o fim do algoritmo.
20. [*bits diferentes nos caracteres*] Dados dois caracteres  $x$  e  $y$  lidos do usuário, faça um algoritmo que verifique de quantos bits, posição por posição, diferem as suas respectivas representações numéricas no código ASCII em binário e qual o primeiro bit, da esquerda para a direita, onde aparece a diferença. Escreva os resultados ao final.
21. [*troca bits do caracter*] Faça um algoritmo que leia um caracter da entrada e escreva o caracter transformado de forma que o primeiro bit, da esquerda para a direita, seja trocado de posição com o sétimo bit da representação binária do seu código ASCII.
22. [*rotacao de bits*] Faça um algoritmo para ler um caracter e gerar um outro caracter na saída que seja o resultado da rotação de 1 bit para a esquerda dos sete bits menos significativos do seu código ASCII. Por exemplo, se os 8 bits do código são  $\alpha\beta\gamma\delta\epsilon\zeta\eta\theta$  o código a ser gerado deve ser  $\alpha\gamma\delta\epsilon\zeta\eta\theta\beta$ .

17 - tomando  $x$ ,  $y$  e  $z$ , basta verificar se a distância de  $x$  para  $y$  (modulo da diferença) é igual a distância de  $x$  para  $y$ , se for,  $x$  é equidistante em relação aos outros 2. Para resolver o problema, é necessário verificar esse pensamento para  $x$ , para  $y$  e para  $z$ .

21 -> faço uma "cirurgia", separo em uma variável o bit1, em outra o bit7, em outra o byte original com os "buracos" dos dois novos bits faltando, desloco o bit1 para a posição 7, desloco o bit7 para a posição 1 e depois junto tudo

16/22 -> separo o bit 8 do resto dos 7, guardo o bit 7, guardo os três, desloco os menos significativos para esquerda uma casa, recoloco o 7 na posição do bit 1 (simulando a rotação) e devolvo o bit 8 pro lugar dele de novo.