Lista de exercícios #2 – algoritmos numéricos com operações inteiras

Sobre a lista:

Para encontrar o valor absoluto (módulo) de um número você pode usar a função `abs()`.

Para encontrar a raiz quadrada de um número você pode usar a função `raizq()`.

Para obter apenas a parte inteira, do tipo inteiro, de um número real você pode usar a função `trunc()`.

1. Faça um algoritmo que, primeiro, leia um número inteiro qualquer e, em seguida, leia um número natural de apenas 1 algarismo e, por fim, diga se o algarismo digitado aparece no número digitado. Use a frase "`O algarismo ☐ aparece no número ☐`" ou "`O algarismo ☐ não aparece no número ☐`", dependendo do resultado.

2. Faça um algoritmo que leia dois números inteiros e diga se a sequência exata dos dígitos que formam o segundo número aparece na sequência de dígitos que formam o primeiro número lido.

   Exemplo:

   primeiro número = `1234567890`
   segundo número = `456`

   Use a frase "`O número ☐ aparece como parte do número ☐`" ou "`O número ☐ não aparece como parte do número ...`", dependendo do resultado.

3. Faça um algoritmo que leia um número inteiro (em decimal) e escreva a quantidade de bits iguais a "`1`" nesse número.

4. Faça um algoritmo que leia 2 números inteiros e verifique se todos os algarismos que formam o segundo número aparecem em alguma posição do primeiro número.

   No exemplo, os algarismos 3, 5 e 9 do segundo número aparecem no primeiro número:

   Exemplo:

   primeiro número (X) = `1234567890`
   segundo número (Y) = `539`

   Use a frase "`Os algarismos do número Y aparecem no número X`" ou "`Os algarismos do número Y não aparecem no número X`", a depender do resultado.

5. Um número natural é formado por vários algarismos, ou seja, uma quantidade de 0s (significativos), uma quantidade de 1s, uma quantidade de 2s *etc.*

   Faça um algoritmo que leia um número natural e escreva na saída a quantidade (apenas quando for não nula) de cada algarismo existente no número.

6. Faça um algoritmo que leia 2 números inteiros e verifique quantos algarismos distintos que formam o segundo número aparecem no primeiro número, escrevendo a porcentagem de algarismos do primeiro número que foram efetivamente encontrados.

   Exemplo:

   primeiro número (X) = `-1234567890`
   segundo número (Y) = `2482`
   Resultado = 0.3 ou 30%

   Use a frase "`Em Y há p% dos algarismos que aparecem em X`".

7. Faça um algoritmo que leia dois números inteiros e descubra os algarismos da interseção entre os dois, ou seja, os algarismos que aparecem em ambos os números, escrevendo a quantidade deles e a soma destes algarismos.

Exemplo:

primeiro número (X) = `7245680`
segundo número (Y) = `-1835709`
Interseção = `7, 0, 8, 5`
Quantidade (q) = `4`
Soma (s) = `20`

Use a frase "`M e N têm q algarismos em comum e sua soma é s`".

8. Faça um algoritmo que leia um número natural e gere um segundo número que seja a inversão de posição dos algarismos do número lido, ou seja, a posição dos algarismos do segundo número está de trás para frente em relação ao primeiro número, escrevendo o número invertido resultante na saída.

   Exemplo:

   Número (X) = `12345678900`
   Resultado (Y) = `987654321`

   Use a frase "`O número X gera o número invertido Y`".

9. Faça um algoritmo que leia um número natural de no máximo 8 dígitos binários, representando um numeral binário de 8 dígitos (ou seja, você deve garantir que o o número deve ser formado apenas pelos dígitos 0s e 1s), monte o seu equivalente na base decimal e escreva este número resultante na saída.

   Use a frase "`O número binário B equivale ao número decimal D`".

10. Faça um algoritmo que leia um número natural representando uma quantidade de segundos e escreva o valor equivalente em horas, minutos e segundos.

    Use a frase "`X segundos equivalem a H horas, M minutos e S segundos`".

11. Faça um algoritmo que leia um número natural representando um número decimal, o converta para sua representação na base binária e escreva este binário resultante.

    Use a frase "`O numero decimal D é equivalente ao número binário B`".

12. Seja um número natural de no máximo 4 dígitos (no formato $abcd$) no qual o último dígito (d) serve para controle e é igual ao resto da divisão inteira de $a + 3b + 5c$ por 7. Para que fique claro, a, b e c podem ter valor igual a 0.

    Faça um algoritmo que leia um número natural de no máximo 4 dígitos, no qual o último dígito deveria ser um dígito de controle calculado conforme a explicação anterior, e escreva na saída se o número digitado pelo usuário está correto, ou seja, se é um número cujo dígito de controle está correto.

    Obs: é com um método semelhante a esse que se verifica a digitação correta de um CPF ou de um número de matrícula, por exemplo.

    Use a frase "`O número X foi digitado corretamente`" ou "`O número X foi digitado com erro`", dependendo do caso.

13. Faça um algoritmo que escreva todos os números naturais pares entre 1 e 1001, que não sejam múltiplos de 3 nem de 5. Os números da saída devem ser escritos em uma única linha e devem ser separados por apenas um caractere de espaço em branco.

14. O crivo de Euclides para encontrar o MDC entre dois números inteiros estritamente positivos pode ser descrito como a seguir:

    Se r é o resto da divisão inteira de a por b, com $a > b$, então o máximo divisor comum de a e b é igual ao máximo divisor comum de b e r, ou seja, $MDC(a, b) = MDC(b, r)$, enquanto a for diferente de b.

    Então, com esta informação, dados dois valores a e b, é possível calcular iterativamente o MDC entre eles, através de subtrações sucessivas ou, melhor, restos sucessivos, até que

tenhamos $a = b$, caso em que o $MDC = a$ (ou $= b$) ou, alternativamente, até que um dos valores chegue a zero, caso em que o MDC será igual ao outro valor diferente de zero.

Assim, faça um algoritmo para calcular o MDC de dois números inteiros estritamente positivos, lidos do usuário, usando o crivo de Euclides como algoritmo, escrevendo-o na saída como "`O MDC entre a e b é mdc`".

15. Suponha um terminal PDV (ponto-de-venda) que, para facilitar a vida do operador, diga quantas moedas têm que ser dadas de troco ao cliente. As moedas existentes (em número suficiente) são de \$0.01, \$0.05, \$0.25 e \$1.00.

    Faça um algoritmo que leia o valor dado pelo cliente como pagamento, o valor total da compra e escreva na saída a menor combinação possível de moedas a serem devolvidas ao cliente como troco na forma "`x moedas de y centavos`", em uma linha separada para cada moeda, ou "`sem troco`":

    Exemplo:

    "`x moedas de 1 centavo`"
    "`y moedas de 10 centavos`"

16. Faça um algoritmo que inverta um número natural de 8 algarismos, gerando-o em uma nova variável, escrevendo o valor dessa variável ao fim do algoritmo. Para números menores que 10000000 considere que ele possua zero's à esquerda.

17. O dia da semana para uma data qualquer pode ser calculado pela seguinte fórmula:

$$r(q((2.6 \cdot M - 0.2); 1) + D + A + q(A; 4); 7)$$

    - $r(n; z)$ representa a função que fornece o resto módulo z do número n;
    - $q(n; z)$ representa a função que fornece o quociente da divisão de n por z;
    - M representa o número do mês. Janeiro e fevereiro são os meses 11 e 12 do **ano precedente**, março é o mês 1 e dezembro é o mês 10;
    - D representa o dia do mês;
    - A representa o número formado pelos dois últimos algarismos do ano;
    - S representa o número formado pelos dois primeiros algarismos do ano.

    Os dias da semana são numerados de zero a seis, com domingo = 0 , segunda = 1, e assim por diante até sábado = 6.

    Fazer um algoritmo que leia um conjunto de 50 datas (dia, mês, ano), determine o dia da semana correspondente à data lida, segundo o método especificado e escreva, para cada data lida, o dia, mês, ano e dia da semana calculado.

18. No contexto de armazenamento e telecomunicações, *paridade* se refere ao número de bits "`1`" de um determinado número binário. Para assinalar a paridade par, toda vez que a quantidade de bits iguais a "`1`" encontrados no número for ímpar, adiciona-se um bit "`1`" ao final deste número para que essa quantidade se torne par.

    Resumindo, uma função que analise os bits de um número e encontre uma quantidade ímpar de bits "`1`" deve retornar "`1`" ou `VERDADEIRO` . Em caso contrário deve retornar "`0`" ou `FALSO`.

    Assim, faça um algoritmo que leia um número inteiro e retorne a paridade desse número, escrevendo na saída se ela é "`par`" ou "`ímpar`".

19. Uma cadeia *palíndrome* é aquela que pode ser lida exatamente da mesma maneira tanto no sentido normal de leitura quanto no sentido inverso de leitura. Assim, faça um algoritmo que leia um número inteiro (em decimal) e diga se sua representação em binário é *palíndrome*.