CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
CAMPUS VII - UNIDADE TIMÓTEO
Curso: Técnico em Informática
Disciplina: Linguagem de Programação I
Professor: Odilon Corrêa

Lista de Exercícios - Modularizando Algoritmos - (Parte II)

1. Construa um algoritmo que leia 3 números inteiros A, B e C e que, utilizando uma função, imprima o maior número.
2. Construa um algoritmo que verifique se um número inteiro, passado como parâmetro para uma função, é par, retornando verdadeiro ou falso.
3. Calcule o valor da expressão x2 - y + 10 através de um algoritmo, seguindo as seguintes regras:
   - o cálculo da expressão deve ser por um procedimento;
   - não é permitido utilizar variáveis globais;
   - a leitura dos dados e exibição dos resultados deve ser feita no algoritmo principal.
4. Desenvolva uma sub-rotina que retorne o máximo divisor comum (MDC) de dois números informados (o MDC de dois números informados é o maior número que os divide). Em seguida desenvolva um algoritmo que utilize esse procedimento e imprima o vetor ordenado.
5. Construa um procedimento que preencha uma matriz (3x3) com números inteiros. Em seguida desenvolva um algoritmo que utilize esse procedimento e imprima o conteúdo da matriz.
6. Elabore um procedimento que faça a ordenação de um vetor (3) com números inteiros. Em seguida desenvolva um algoritmo que utilize esse procedimento e imprima o vetor ordenado.
7. Construa um algoritmo que faça a ordenação, linha por linha, de uma matriz (3x3) utilizando os procedimentos criados nos exercícios 4 e 5.
8. Construa uma função que faça a conversão de um número inteiro (decimal) em números romanos. Em seguida desenvolva um algoritmo essa função. Por exemplo:

I = 1	XX = 20	CCC = 300
II = 2	XXX = 30	CD = 400
III = 3	XL = 40	D = 500
IV = 4	L = 50	DC = 600
V = 5	LX = 60	DCC = 700
VI = 6	LXX = 70	DCCC = 800
VII = 7	LXXX = 80	CM = 900
VIII = 8	XC = 90	M = 1.000
IX = 9	C = 100	MM = 2.000
X = 10	CC = 200	MMM = 3.000

9. Faça o rastreio do algoritmo e registre suas saídas.

```
programa
{
    inclua biblioteca Util --> util

    funcao inicio()
    {
        inteiro matrizA[3][2]
        inteiro matrizB[4][3]
        escreva("Elementos da primeira matriz\n")
        preencherMatriz(matrizA)
        imprimirMatriz(matrizA)
        escreva("Elementos da segunda matriz\n")
        preencherMatriz(matrizB)
        imprimirMatriz(matrizB)
    }

    funcao preencherMatriz(inteiro &matriz[][])
    {
        inteiro linha, coluna, quantLinha, quantColuna
        quantLinha = util.numero_linhas(matriz)
        quantColuna = util.numero_colunas(matriz)
        para(linha = 0; linha < quantLinha; linha++)
        {
            para(coluna = 0; coluna < quantColuna; coluna++)
            {
                matriz[linha][coluna] = util.sorteia(0, 5)
            }
        }
    }

    funcao imprimirMatriz(inteiro matriz[][])
    {
        inteiro linha, coluna, quantLinha, quantColuna
        quantLinha = util.numero_linhas(matriz)
        quantColuna = util.numero_colunas(matriz)
        para(linha = 0; linha < quantLinha; linha++)
        {
            para(coluna = 0; coluna < quantColuna; coluna++)
            {
                escreva(matriz[linha][coluna],"\t")
            }
            escreva("\n")
        }
    }
```

10. Faça o rastreio do algoritmo e registre suas saídas.

```
programa
{
    inclua biblioteca Util --> util

    funcao inicio()
    {
        inteiro numeros[3][4], numero
        escreva("Elementos da  matriz\n")
        preencherMatriz(numeros)
        imprimirMatriz(numeros)
        numero = util.sorteia(0, 5)
        escreva("A matriz contém ",contarNumero(numeros, numero)," ocorrência(s) do número ", numero)

    }

    funcao preencherMatriz(inteiro &matriz[][])
    {
        inteiro linha, coluna, quantLinha, quantColuna
        para(linha = 0; linha < util.numero_linhas(matriz); linha++)
        {
            para(coluna = 0; coluna < util.numero_colunas(matriz); coluna++)
            {
                matriz[linha][coluna] = util.sorteia(0, 5)
            }
        }
    }

    funcao imprimirMatriz(inteiro matriz[][])
    {
        inteiro linha, coluna, quantLinha, quantColuna
        para(linha = 0; linha < util.numero_linhas(matriz); linha++)
        {
            para(coluna = 0; coluna < util.numero_colunas(matriz); coluna++)
            {
                escreva(matriz[linha][coluna],"\t")
            }
            escreva("\n")
        }
    }

    funcao inteiro contarNumero(inteiro matriz[][], inteiro numero)
    {
        inteiro linha, coluna, quantidade = 0
        para(linha = 0; linha < util.numero_linhas(matriz); linha++)
        {
            para(coluna = 0; coluna < util.numero_colunas(matriz); coluna++)
            {
                se(matriz[linha][coluna] == numero)
                {
                    quantidade = quantidade + 1
                }
            }
        }
        retorne quantidade
    }
}
```