INSTITUTO FEDERAL MINAS GERAIS (IFMG) - CAMPUS BAMBUÍ
Cálculo Numérico
Prof. Marcos Roberto Ribeiro

Lista de Exercícios 00

Exercício 1:
Faça um código capaz de calcular as raízes de uma equação de segundo grau no formato $Ax^2+Bx+C=0$. Considere as seguintes observações:

- Se o termo $A=0$ então a equação não é uma equação de segundo grau;
- Seja $\Delta = B^2 - 4 \times A \times C$. Se $\Delta < 0$ então a equação não possui raízes. Se $\Delta = 0$ então a equação possui apenas uma raiz.

Exercício 2:
Faça um código para determinar se um ano é ou não bissexto. Um ano N é bissexto se N é múltiplo de 400, ou então se N é múltiplo de quatro, mas não é múltiplo de 100. Por exemplo, 2012 (múltiplo de 4, mas não múltiplo de 100) é bissexto, 1900 (múltiplo de quatro e de 100) não é bissexto, 2000 (múltiplo de 400) é bissexto).

Exercício 3:
Considere a função **range**(*início*, *fim*, *passo*). Utilize essa função algumas vezes com variações nos parâmetros e faça laços de repetições para imprimir os valores gerados.

Exercício 4:
Elabore um código que receba três notas considerando os pesos $0,3$, $0,3$ e $0,4$ para as notas 1, 2 e 3, respectivamente. A nota final é soma ponderada das três notas ($NotaFinal = Nota_1 * 0,3 + Nota_2 * 0,3 + Nota_4 * 0,3$). O código deve receber estas notas para 10 alunos. É necessário informar se cada aluno foi aprovado (nota final maior ou igual a 60) ou não. Também deve ser informada a média final de todas as notas finais.

Exercício 5:
Projete um código que leia uma quantidade indeterminada de números. A cada número informado, o usuário deve informar se deseja continuar ou parar. Ao final, o código deve retornar o maior e o menor número recebido.

Exercício 6:
Elabore um código que calcule o fatorial de um número recebido. O fatorial de um número n, representado por $n!$, é calculado da seguinte maneira $n! = n \times (n-1) \times ... \times 2 \times 1$. Sendo que $1! = 0! = 1$.

Exercício 7:
Implemente o algoritmo de Euclides[1] para calcular o máximo divisor comum (MDC) de dois números.

Exercício 8:
Crie um código que calcule o mínimo múltiplo comum (MMC) entre dois números. O MMC de dois números n_1 e n_2 pode ser calculado como $MMC = \frac{n_1 * n_2}{MDC}$, onde MDC é o máximo divisor comum entre n_1 e n_2.

Exercício 9:
Elabore um código capaz de simular uma calculadora simples. O código deve solicitar ao usuário a operação desejada (soma, multiplicação, divisão, subtração ou potência) ou então sair. Quando o usuário escolhe uma operação, o programa deve solicitar dois números, realizar a operação sobre estes números e exibir o resultado. O código deve sempre solicitar uma nova operação até que o usuário escolha sair.

Exercício 10:
Desenvolva um código que, considerando um conjunto de números naturais $A = \{1, 2, 3, ..., n\}$, gere todas as combinações com três elementos contidos em A. Antes de gerar as combinações o programa deve perguntar o número de elementos de A ao usuário.

[1] `https://pt.wikipedia.org/wiki/Algoritmo_de_Euclides`

Exercício 11:

Considerando um conjunto $A = \{1, 2, 3, ..., n\}$ com n informado pelo usuário, crie um código para obter os subconjuntos de três elementos contidos em A. Lembre-se que um conjunto não possui elementos repetidos.

Exercício 12:

A função **shuffle()** da biblioteca **random** permite embaralhar uma lista de elementos. Utilizando essa função crie um código capaz de sortear elementos de uma lista. O usuário deve informar o número total de elementos e quantos elementos devem ser sorteados.

Exercício 13:

Projete um código que preencha uma lista de 100 posições com números de 1 a 10 (utilize a função **randint()** da biblioteca **random**). Em seguida, conte o número de vezes que cada número aparece e armazene em um segunda lista. Por fim, exiba a quantidade de veses que cada número apareceu.

Exercício 14:

Construa um código capaz de calcular a velocidade média de uma viagem a partir das velocidades de cada trecho. Uma viagem pode possuir de 1 a 50 trechos. O cálculo da velocidade média n trechos é dado pela fórmula:

$$\frac{distância_1 \times velocidade_1 + ... + distância_n \times velocidade_n}{distância_1 + ... + distância_n}$$

Após o cálculo da velocidade média, o programa deve exibir quais trechos tiveram velocidade acima da média.

Exercício 15:

Implemente um código capaz de ordenar os elementos de uma lista. Inicialmente, o código deve receber a quantidade de posições e um valor para cada posição. Em seguida, o código deve ordenar os elementos da lista utilizando o método bolha[2]:

- Percorra a lista várias vezes ordenando pares de elementos, até que todos os elementos estejam ordenados;
- Observe o exemplo a seguir:
  1. Lista inicial desordenada `F,O,R,A,O,R,D,E,M`
  2. Primeira passagem até a última posição
     (a) `F,O,R,A,O,R,D,E,M` $F \not> O$ não troca
     (b) `F,O,R,A,O,R,D,E,M` $O \not> R$ não troca
     (c) `F,O,R,A,O,R,D,E,M` $R > A$ troca
     (d) `F,O,A,R,O,R,D,E,M` $R > O$ troca
     (e) `F,O,A,O,R,R,D,E,M` $R \not> R$ não troca
     (f) `F,O,A,O,R,R,D,E,M` $R > D$ troca
     (g) `F,O,A,O,R,D,R,E,M` $R > E$ troca
     (h) `F,O,A,O,R,D,E,R,M` $R > M$ troca
     (i) `F,O,A,O,R,D,E,M,R` Fim da primeira passagem
  3. Segunda passagem até a penúltima posição
     (a) `F,O,A,O,R,D,E,M,R` $F \not> O$ não troca
     (b) `F,O,A,O,R,D,E,M,R` $O > A$ troca
     (c) `F,A,O,O,R,D,E,M,R` $O \not> O$ não troca
     (d) `F,A,O,O,R,D,E,M,R` $O \not> R$ não troca
     (e) `F,A,O,O,R,D,E,M,R` $R > D$ troca
     (f) `F,A,O,O,D,R,E,M,R` $R > E$ troca
     (g) `F,A,O,O,D,E,R,M,R` $R > M$ troca
     (h) `F,A,O,O,D,E,M,R,R` Fim da segunda passagem
  4. Terceira passagem até a antepenúltima posição
     (a) `F,A,O,O,D,E,M,R,R` $F > A$ troca

[2] O método bolha é um método de ordenação bastante simples, existem diversos outros métodos de ordenação mais complexos e mais eficientes.

(b) `A,F,O,O,D,E,M,R,R` $F \not> O$ não troca
(c) `A,F,O,O,D,E,M,R,R` $O \not> O$ não troca
(d) `A,F,O,O,D,E,M,R,R` $O > D$ troca
(e) `A,F,O,D,O,E,M,R,R` $O > E$ troca
(f) `A,F,O,D,E,O,M,R,R` $O > M$ troca
(g) `A,F,O,D,E,M,O,R,R` Fim da terceira passagem

5. Quarta passagem até a quarta última posição
   (a) `A,F,O,D,E,M,O,R,R` $A \not> F$ não troca
   (b) `A,F,O,D,E,M,O,R,R` $F \not> O$ não troca
   (c) `A,F,O,D,E,M,O,R,R` $O > D$ troca
   (d) `A,F,D,O,E,M,O,R,R` $O > E$ troca
   (e) `A,F,D,E,O,M,O,R,R` $O > M$ troca
   (f) `A,F,D,E,M,O,O,R,R` Fim da quara passagem
6. Quinta passagem até a quinta última posição
   (a) `A,F,D,E,M,O,O,R,R` $A \not> F$ não troca
   (b) `A,F,D,E,M,O,O,R,R` $F > D$ troca
   (c) `A,D,F,E,M,O,O,R,R` $F > E$ troca
   (d) `A,D,E,F,M,O,O,R,R` $F \not> M$ não troca
   (e) `A,D,E,F,M,O,O,R,R` Fim da quinta passagem
7. Na sexta passagem não ocorre nenhuma troca e o código pode parar

Exercício 16:
Desenvolva um código que receba uma lista de números inteiros e, sem seguida, crie uma nova lista contendo os mesmos números sem repetições.

Exercício 17:
Crie as funções **input_int(mensagem)** e **input_float(mensagem)** semelhantes à função **input()**. As funções devem receber uma mensagem, exibi-la ao usuário e garantir que o número digitado seja válido. Enquanto o usuário digitar um número incorreto, as funções devem informar que o número é inválido e solicitar e digitação novamente. Faça uma chamada a cada função para testá-las. Dica: utilize a instrução **try** (tratamento de exceções).

Exercício 18:
Implemente o módulo **mat.py** contendo as seguintes funções:

impar(numero): Retorna **True** ou **False**, caso o **numero** seja impar ou não;

area_circulo(raio): retorna a área de um círculo a partir de seu **raio**, $\acute{a}rea = raio^2 \times \pi$ (dica, importe **pi** da biblioteca **math**)

Codifique as chamadas a essas funções em outro arquivo.

Exercício 19:
Implemente um módulo com as seguintes funções:

ano_bissexto(ano): retorna **True** se o ano for bissexto e **False**, em caso negativo;

dias_mes(ano, mes): retorna a quantidade de dias do mês (deve usar a função **ano_bissexto**);

nome_mes(mes): retorna o nome do mes informado (1 a 12)

Construa um programa que receba uma data do usuário (mês e ano) e mostre o resultado das funções implementadas sobre essa data.

Exercício 20:
Escreva um módulo com funções para calcular o máximo divisor comum (MDC) e o mínimo múltiplo comum (MMC) de uma lista de números. Dica: comece calculando o MDC dos dois primeiros números e, depois, para cada número, calcule o MDC desse número com o resultado anterior (o MMC pode ser feito de fórmula análoga). Implemente também chamadas a essas funções números digitados pelo usuário.