

EXERCÍCIOS DE PROGRAMAÇÃO NO OCTAVE

GABRIELA C SCHMITT

19 DE JUNHO DE 2025



**PROF. ANA PAULA OENING
SOFTWARE GNU OCTAVE**

1. Desenvolva um algoritmo para calcular as raízes reais de uma equação quadrática $ax^2 + bx + c = 0$ usando a fórmula de Baskara, para valores dados de a, b e c. Teste seu programa para os seguintes casos:

i. $a=2$, $b=10$, $c=12$

ii. $a=3$, $b=24$, $c=48$

iii. $a=4$, $b=24$, $c=100$

```
clear
```

```
function res = bhaskara(A, B, C)
```

```
    n = length(A);
```

```
    res = NaN(n, 2); % Inicializa a matriz com NaN
```

```
    for i = 1:n
```

```
        a = A(i);
```

```
        b = B(i);
```

```
        c = C(i);
```

```
        delta = b^2 - 4*a*c;
```

```
        if delta > 0
```

```
            x1 = (-b + sqrt(delta)) / (2*a);
```

```
            x2 = (-b - sqrt(delta)) / (2*a);
```

```
            res(i, :) = [x1, x2];
```

```
        elseif delta == 0
```

```
            x = -b / (2*a);
```

```
            res(i, :) = [x, x];
```

```
        else
```

```
            % Mantém NaN para indicar que não há raízes reais
```

```
            res(i, :) = [NaN, NaN];
```

```
        end
```

```
    end
```

```
end
```

```
A1 = [2;3;4];
```

```
b1 = [10; 24; 24];
```

```
c1 = [12; 48; 100];
```

```
resultados = bhaskara(A1, b1, c1)
```

```
disp('Resultados das raízes:')
```

```
disp(resultados)
```

2. Escreva um programa que aceite um valor numérico x de 0 a 100 como entrada e que calcule e exiba a letra correspondente à nota de acordo com a seguinte tabela

- a. $x \geq 90$**
- b. $80 \leq x \leq 89$**
- c. $70 \leq x \leq 79$**
- d. $60 \leq x \leq 69$**
- e. $x < 60$**

```
function letra_nota = converterNota(x)
    % Esta função converte um valor numérico de nota em uma letra
    % correspondente.
    % x: Valor numérico da nota (de 0 a 100).

    if x >= 90
        letra_nota = 'A';
    elseif x >= 80 && x <= 89
        letra_nota = 'B';
    elseif x >= 70 && x <= 79
        letra_nota = 'C';
    elseif x >= 60 && x <= 69
        letra_nota = 'D';
    else
        letra_nota = 'E';
    end
end

nota1 = 95;
disp(['A nota ', num2str(nota1), ' corresponde à letra: ',
converterNota(nota1)]);

nota2 = 0;
disp(['A nota ', num2str(nota2), ' corresponde à letra: ',
converterNota(nota2)]);
```

3. Dado um número x e o quadrante q ($q = 1, 2, 3, 4$), escreva -1 um programa para calcular $\sin^{-1}x$ em graus, levando em consideração o quadrante. O programa deve exibir uma mensagem de erro se $|x| > 1$.

Para exibir avisos na tela:

`disp('Este é um aviso na tela do octave.')`

```
function angulo = sen_inverso_quadrante(x, q)
```

```
% Esta função calcula o arco seno de um número 'x' em graus, levando em  
% consideração o quadrante 'q' (1, 2, 3, ou 4).
```

```
if abs(x) > 1
```

```
    disp('Erro: O valor de |x| deve ser menor ou igual a 1.');
```

```
    angulo = NaN;
```

```
    return;
```

```
end
```

```
alpha = asind(x);
```

```
switch q
```

```
    case 1
```

```
        if x < 0
```

```
            disp('Aviso: Para o 1º quadrante, o valor de x deveria ser positivo.');
```

```
        end
```

```
        angulo = alpha;
```

```
    case 2
```

```
        if x < 0
```

```
            disp('Aviso: Para o 2º quadrante, o valor de x deveria ser positivo.');
```

```
        end
```

```
        angulo = 180 - alpha;
```

```
    case 3
```

```
        if x > 0
```

```
            disp('Aviso: Para o 3º quadrante, o valor de x deveria ser negativo.');
```

```
        end
```

```
        angulo = 180 - alpha;
```

```
    case 4
```

```
        if x > 0
```

```
            disp('Aviso: Para o 4º quadrante, o valor de x deveria ser negativo.');
```

```
        end
```

```
        angulo = 360 + alpha;
```

```
    otherwise
```

```
        disp('Erro: Quadrante inválido. Forneça um valor de 1 a 4.');
```

```
        angulo = NaN;
```

```
end
```

```
end
```

4. Escreva um arquivo de script para determinar o número de termos necessários para que a série $5k^2 - 2k$, $k = 1, 2, \dots$ exceda 10.000. Qual é a soma para esse número de termos?

```
soma = 0;
k = 0;
limite = 10000;

while soma <= limite
    k = k + 1;

    termo_atual = (5 * k^2) - (2 * k);

    soma = soma + termo_atual;
end
```

5. Determine quanto tempo será necessário para que você acumule pelo menos R\$10.000 em uma conta bancária se você depositar inicialmente R\$500 e mais R\$500 ao final de cada ano, com um rendimento anual de 5%.

```
saldo = 500;
deposito_anual = 500;
taxa_juros = 0.05;
meta = 10000;
anos = 0;

while saldo < meta
    anos = anos + 1;
    saldo = saldo * (1 + taxa_juros) + deposito_anual;
end

fprintf('Serão necessários %d anos para acumular pelo menos R$ %.2f.\n', anos, meta);
fprintf('Ao final desse período, o saldo será de R$ %.2f.\n', saldo);
```

6. Reescreva o exemplo 17 utilizando um laço while para evitar o uso do comando break.

```
clear
clc

k = 1;
continuar_loop = true;

while k <= 10 && continuar_loop
    x = 50 - k^2;
    if x >= 10
        y = sqrt(x)
        k = k + 1;
    else
        continuar_loop = false;
    end
end
```

7 - Verificador de Número Primo Crie uma função `eh_primo(n)` que retorna verdadeiro (`true`) se `n` for primo e falso (`false`) caso contrário. Use `if` e `for`.

```
function resultado = eh_primo(n)
    # Números menores ou iguais a 1 não são primos.
    if n <= 1
        resultado = false;
        return;
    end

    # Passo 2: Verifica se existe algum divisor entre 2 e n-1.
    for i = 2:n-1
        if mod(n, i) == 0
            # Se encontrar qualquer divisor, o número não é primo.
            resultado = false;
            return;
        end
    end

    # Se o laço terminar sem encontrar divisores, o número é primo.
    resultado = true;
end

disp(eh_primo(7)) # 1 == (true)
```

8. Soma até Limite com `while` Crie uma função `soma_limite(limite)` que soma números inteiros consecutivos a partir de 1 até que a soma ultrapasse o valor do `limite`

```
function [soma_final, ultimo_numero] = soma_limite(limite)
# Soma números inteiros consecutivos a partir de 1
# até que a soma ultrapasse o valor do 'limite'.

    soma = 0;
    numero_atual = 0;

    while soma <= limite
        numero_atual = numero_atual + 1;
        soma = soma + numero_atual;
    end

    soma_final = soma;
    ultimo_numero = numero_atual;
end

limite_teste = 50;
[s, u] = soma_limite(limite_teste);

disp(['Para o limite de ', num2str(limite_teste), ', a soma que o
ultrapassou foi: ', num2str(s)])
disp(['Isso foi alcançado somando até o número: ', num2str(u)])
```

9. Classificação de Idades

Crie uma função `classifica_idade(idade)` que retorna:

- **'Criança'** se < 12
- **'Adolescente'** se entre 12 e 17
- **'Adulto'** se entre 18 e 59
- **'Idoso'** se ≥ 60

```
function categoria = classifica_idade(idade)
    #classifica um valor numérico de idade em uma categoria.
    if idade < 12
        categoria = 'Criança';
    elseif idade >= 12 && idade <= 17
        categoria = 'Adolescente';
    elseif idade >= 18 && idade <= 59
        categoria = 'Adulto';
    else
        categoria = 'Idoso';
    end
end

idade1 = 10;
disp(['A idade ', num2str(idade1), ' corresponde à categoria: ',
classifica_idade(idade1)]);
```


10. Fatorial com `for` Crie uma função `fatorial(n)` que calcula o fatorial de `n` usando um laço `for`

```
function resultado = fatorial(n)
    % Esta função calcula o fatorial de um número n usando um
    laço for.

    if n < 0
        disp('Erro: Fatorial não é definido para números negativos.');
```

resultado = NaN;

return;

end

resultado = 1;

for i = 1:n

 resultado = resultado * i;

end

end

n1 = 5;

disp(['O fatorial de ', num2str(n1), ' é: ', num2str(fatorial(n1))]);

11. Contador de Pares e Ímpares Crie uma função `conta_pares_impares(vetor)` que recebe um vetor numérico e retorna a quantidade de números pares e ímpares.

```
function [pares, impares] = conta_pares_impares(vetor)
# Esta função conta a quantidade de números pares e ímpares
em um vetor.
```

```
    pares = 0;
    impares = 0;
```

```
    for i = 1:length(vetor)
        numero_atual = vetor(i);

        if mod(numero_atual, 2) == 0
            pares = pares + 1;
        else
            impares = impares + 1;
        end
    end
end
```

```
vetor_de_teste = [1, 2, 3, 4, 5, 6, 7, 8, 9];
```

```
disp('Contando pares e ímpares no vetor [1, 2, 3, 4, 5, 6, 7, 8, 9]:')
[p, i] = conta_pares_impares(vetor_de_teste);
```

```
fprintf(' -> Quantidade de números pares: %d\n', p);
fprintf(' -> Quantidade de números ímpares: %d\n', i);
```