



UNIVERSIDADE FEDERAL DE SANTA MARIA
CAMPUS CACHOEIRA DO SUL

CADERNO DE EXERCÍCIOS

Algoritmos Aplicados à Engenharia

Autores :
Prof. Dr. Vinícius Maran
Alisson Nascimento
Andrei Cunha Lima

Este caderno de exercícios tem como objetivo apresentar questões relacionadas à algoritmos e programação. As questões apresentadas neste material estão diretamente relacionadas à outras disciplinas de cursos de Engenharia, tais como: Física, Cálculo, Estatística, entre outras.

Este material é resultante da primeira etapa do projeto de ensino *Monitoria Não Subsidiada e Produção de Material sobre Algoritmos e Programação nos cursos de engenharia da Universidade Federal de Santa Maria (UFSM), Campus Cachoeira do Sul*.

As respostas para os algoritmos podem variar. Os exercícios foram resolvidos na linguagem de programação C.

Coordenador do Projeto: Prof. Dr. Vinícius Maran. Contato:
vinicius.maran@ufsm.br

Sugestões sobre o material podem ser informadas [nest link](#).

Este trabalho está licenciado sob uma licença [Creative Commons “Attribution-NonCommercial-ShareAlike 3.0 Unported”](#).



Conteúdo

1	Exercícios	5
1.1	Desvios Condicionais	5
1.1.1	Exercício 1 - Cálculo A	5
1.1.2	Exercício 2 - Estatística	5
1.1.3	Exercício 3 - Estatística	6
1.1.4	Exercício 4 - Estatística	6
1.1.5	Exercício 5 - Física	6
1.2	Laços de Repetição	7
1.2.1	Exercício 1 - Cálculo A	7
1.2.2	Exercício 2 - Cálculo A	7
1.2.3	Exercício 3 - Cálculo A	7
1.2.4	Exercício 4 - Álgebra Linear	7
1.2.5	Exercício 5 - Métodos Numéricos e Computacionais	8
1.3	Vetores e Matrizes	8
1.3.1	Exercício 1 - Álgebra Linear	8
1.3.2	Exercício 2 - Álgebra Linear	8
1.3.3	Exercício 3 - Álgebra Linear	9
1.3.4	Exercício 4 - Estatística	9
1.3.5	Exercício 5 - Métodos Numéricos e Computacionais	9
1.3.6	Exercício 6 - Métodos Numéricos e Computacionais	9
1.3.7	Exercício 7 - Métodos Numéricos e Computacionais	10
1.3.8	Exercício 8 - Métodos Numéricos e Computacionais	10
1.3.9	Exercício 9 - Métodos Numéricos e Computacionais	10
1.3.10	Exercício 10 - Estatística	10
1.4	Estruturas de Dados Heterogêneas	11
1.4.1	Exercício 1 - Física	11
1.4.2	Exercício 2 - Física	11
1.4.3	Exercício 3 - Métodos Numéricos e Computacionais	11
1.5	Funções	12
1.5.1	Exercício 1 - Métodos Numéricos e Computacionais	12
1.5.2	Exercício 2 - Cálculo A	12
1.5.3	Exercício 3 - Álgebra Linear	13
1.5.4	Exercício 4 - Cálculo A	13
1.5.5	Exercício 5 - Cálculo A	13
1.5.6	Exercício 6 - Métodos Numéricos e Computacionais	14
1.5.7	Exercício 7 - Métodos Numéricos e Computacionais	14
1.6	Ponteiros e Alocação Dinâmica	14
1.6.1	Exercício 1 - Estatística	14
1.6.2	Exercício 2 - Física	15

1.6.3	Exercício 3 - Álgebra Linear	15
1.6.4	Exercício 4 - Álgebra Linear	15
2	Algoritmos Resolvidos	16
2.1	Desvios Condicionais	16
2.1.1	Exercício 1	16
2.1.2	Exercício 2	17
2.1.3	Exercício 3	18
2.1.4	Exercício 4	18
2.1.5	Exercício 5	19
2.2	Laços de Repetição	21
2.2.1	Exercício 1	21
2.2.2	Exercício 2	21
2.2.3	Exercício 3	21
2.2.4	Exercício 4	22
2.2.5	Exercício 5	22
2.3	Vetores e Matrizes	23
2.3.1	Exercício 1	23
2.3.2	Exercício 2	24
2.3.3	Exercício 3	24
2.3.4	Exercício 4	26
2.3.5	Exercício 5	27
2.3.6	Exercício 6	28
2.3.7	Exercício 7	29
2.3.8	Exercício 8	30
2.3.9	Exercício 9	30
2.3.10	Exercício 10	31
2.4	Estruturas de Dados Heterogêneas	32
2.4.1	Exercício 1	32
2.4.2	Exercício 2	35
2.4.3	Exercício 3	35
2.5	Funções	36
2.5.1	Exercício 1	36
2.5.2	Exercício 2	37
2.5.3	Exercício 3	40
2.5.4	Exercício 4	42
2.5.5	Exercício 5	42
2.5.6	Exercício 6	43
2.5.7	Exercício 7	43
2.6	Ponteiros e Alocação Dinâmica	44
2.6.1	Exercício 1	44

2.6.2	Exercício 2	46
2.6.3	Exercício 3	47
2.6.4	Exercício 4	48

1 Exercícios

1.1 Desvios Condicionais

1.1.1 Exercício 1 - Cálculo A

Desenvolva um algoritmo que informe as relações trigonométricas em um triângulo retângulo. O usuário deve informar o que deseja encontrar através de um menu com as seguintes opções:

- 1 - Cateto Oposto;
- 2 - Cateto Adjacente;
- 3 - Hipotenusa;
- 4 - Ângulo formado entre um dos catetos e a hipotenusa.

Além disso, o programa deve considerar todas as possíveis alternativas para calcular o que o usuário deseja.

Exemplo: O usuário deseja calcular o ângulo entre os catetos e a hipotenusa de um triângulo. Deve-se empregar uma ou mais relações trigonométricas para o cálculo do ângulo:

- 1^a - Seno $\theta = \text{cateto_oposto}/\text{hipotenusa}$;
- 2^a - Cosseno $\theta = \text{cateto_adjacente}/\text{hipotenusa}$;
- 3^a - Tangente $\theta = \text{cateto_oposto}/\text{cateto_adjacente}$;

1.1.2 Exercício 2 - Estatística

Crie um algoritmo que calcule o intervalo de confiança para uma média populacional. O usuário deve fornecer a média amostral, o nível de confiança em porcentagem, e o número de indivíduos considerados na amostra analisada.

Além disso, o algoritmo deve permitir a realização de duas formas de cálculo, de acordo com a escolha do usuário:

- (i) Utilização da tabela de distribuição de *T Student*;
- (ii) Utilização da tabela de *Distribuição Z*.

Dependendo da escolha do usuário, ele deve informar o desvio padrão populacional (caso escolha a tabela Z) ou o desvio amostral (caso escolha a tabela T). Além disso, o programa deve requisitar ao usuário o valor da probabilidade associada a uma das tabelas, T ou Z.

1.1.3 Exercício 3 - Estatística

Elabore um programa que calcule o teste de hipóteses estatístico. O usuário deve informar a hipótese nula e a alternativa, o nível de significância desejado, as médias amostral e populacional, o desvio padrão populacional e o número de indivíduos que contém a amostra.

O teste de hipóteses deve considerar a tabela de distribuição normal, ou seja, a tabela Z . O usuário deve fornecer o valor da tabela ao programa para que sejam feitos os cálculos.

O programa deve testar se foi aceita a hipótese nula ou a alternativa. Após a realização do cálculo, o programa deve apresentar qual das duas alternativas foi aceita.

1.1.4 Exercício 4 - Estatística

Elabore um código que calcule o teste de hipóteses estatístico. Deve-se utilizar a tabela de T Student. O usuário deve informar o nível de significância, a média da amostra e a média da população, o desvio padrão amostral e a quantidade de elementos pertencentes à amostra.

O programa deve calcular através da tabela T , utilizando-se da fórmula do T calculado, que pode ser encontrada em livros de estatística ou até mesmo na internet. Considere apenas valores positivos para T calculado. Apresente se resultado do cálculo aceita H_0 ou H_1 . Apresente na tela qual foi a alternativa aceita, e qual foi rejeitada.

1.1.5 Exercício 5 - Física

Elabore um programa que aborde a lei da conservação de energia. Crie um código que, em cada desvio condicional, isola uma incógnita da fórmula do princípio da conservação de energia mecânica.

Utilize a fórmula $E = U + K$ (*Energia mecânica = Energia potencial + Energia cinética*). Isole uma incógnita por vez e utilize um indicador para saber qual desvio condicional o usuário deseja utilizar.

Exemplo: O usuário deseja encontrar a velocidade de uma partícula associada a uma conservação de energia. O programa deve ler o indicador que o usuário digitar, e isolar a velocidade na fórmula.

1.2 Laços de Repetição

1.2.1 Exercício 1 - Cálculo A

Elabore um código que deve calcular uma integral, na qual a função deve ser implementada no código fonte. O usuário deve fornecer os limites de integração da função considerada. Divida o intervalo em 10000 partições, e faça um somatório, avaliando a função e somando todos os resultados.

Exemplo: Calcule a integral da função $f(x) = 2x$ no intervalo de 0 a 1.

Para calcular esta integral, deve ser implementada no código fonte e os intervalos devem ser fornecidos pelo usuário. O programa deve dividir o intervalo em 10000 partes, e atribuir a x o valor do limite inferior, ou seja, 0.

A função deve ser avaliada para $x = 0$ e depois disso, para cada uma das 10000 partições feitas. Todas as vezes que x for avaliado, o resultado deve ser armazenado em um somatório. Este somatório é o resultado da integral.

1.2.2 Exercício 2 - Cálculo A

Elabore um programa que calcule o fatorial associado a um número, fornecido pelo usuário.

Caso o usuário forneça um número negativo, o programa deve apresentar a mensagem *Digite apenas valores maiores ou iguais a zero*. O fatorial é calculado através da multiplicação dos números anteriores até 1.

Exemplo: $5! = 5 * 4 * 3 * 2 * 1$.

1.2.3 Exercício 3 - Cálculo A

Elabore um programa que calcule $\sum_{j=1}^N (j)$, onde N é informado pelo usuário.

Caso o usuário forneça um valor negativo, o programa deve apresentar a mensagem: *Digite apenas valores maiores ou iguais a zero*.

1.2.4 Exercício 4 - Álgebra Linear

Crie um programa para calcular uma raiz da função $y = 2x - 2$, dado um intervalo informado pelo usuário.

O usuário deve fornecer a precisão do resultado, ou seja, qual o erro máximo permissível para o resultado da raiz obtida. Para realizar o cálculo, o programa deve implementar o 'Método da Bissecção'.

O programa deve somar os extremos do intervalo e dividir por 2. Logo após, a função deve ser avaliada para o valor obtido da operação feita (vamos chamar este valor de x) com os extremos do intervalo, para verificar se este é o zero da função.

1.2.5 Exercício 5 - Métodos Numéricos e Computacionais

Escreva um algoritmo que encontre o zero de $f(x) = e^x - 4x^2$ através do método da bissecção tendo a precisão exigida pelo usuário. O usuário deve inserir o intervalo do zero da função e a precisão exigida (Perceba que o algoritmo pode ser usado para qualquer outra função, sendo necessário somente modificar a expressão implementada no código fonte).

1.3 Vetores e Matrizes

1.3.1 Exercício 1 - Álgebra Linear

Faça um programa que calcule o produto vetorial entre dois vetores tridimensionais $(\vec{i}, \vec{j}, \vec{k})$ através das definições estudadas em álgebra linear. Lembre-se que a interpretação geométrica do produto vetorial é um vetor perpendicular aos dois vetores originários.

Para calcular o produto vetorial, utilize-se a fórmula:

$$\vec{a} \times \vec{b} = \vec{i}(a_2b_3 - a_3b_2) + \vec{j}(a_3b_1 - a_1b_3) + \vec{k}(a_1b_2 - a_2b_1)$$

onde \vec{a} é o primeiro vetor e \vec{b} o segundo.

Crie três vetores de inteiros com 3 elementos, sendo que um dos vetores deve ser utilizado para armazenar o resultado. Após, o programa deve realizar as operações e apresentar na tela os 3 vetores.

1.3.2 Exercício 2 - Álgebra Linear

Crie um código que calcule o determinante associado a uma matriz 3x3. O usuário deve digitar a matriz pelo teclado. O programa deve ler a matriz e calcular o determinante associado a matriz. Deve ser utilizada uma matriz de números reais, o determinante e os contadores de linhas (i) e de colunas (j) da matriz.

No final, o programa deve apresentar o determinante com 3 casas depois da vírgula. Utilize a regra de *Sarrus*, estudada na disciplina de álgebra linear, para calcular o determinante.

1.3.3 Exercício 3 - Álgebra Linear

Elabore um algoritmo que apresente a resolução de um sistema linear 3x3, dada uma matriz informada pelo usuário e os termos independentes. O usuário deve criar uma matriz 3x3 para guardar o valor das constantes que multiplicam as variáveis e colocar os termos independentes dentro de um vetor.

Deve-se utilizar o método direto de *Gauss* para resolver o sistema.

1.3.4 Exercício 4 - Estatística

Escreva um algoritmo que calcule:

- (i) Variância;
- (ii) Desvio padrão;
- (iii) Desvio médio, e
- (iv) Coeficiente de variância

para 8 intervalos usando vetores e variáveis do tipo *double* para desvio médio, variância e desvio padrão. O usuário deve inserir os 8 intervalos e a frequência de cada intervalo.

1.3.5 Exercício 5 - Métodos Numéricos e Computacionais

Implemente um algoritmo que utiliza o método de *Gauss-Seidel* para solucionar um sistema linear 4x4 utilizando 200 iterações.

O usuário deve informar a possível solução (x_0), os coeficientes (a_1, a_2, a_3, a_4) e o coeficiente linear (B).

1.3.6 Exercício 6 - Métodos Numéricos e Computacionais

Implemente um algoritmo que, usando o método de *Gauss-Jacobi*, solucione um sistema linear 4x4 utilizando 200 iterações.

O usuário deve informar a possível solução (x_0), os coeficientes (a_1, a_2, a_3, a_4) e o coeficiente linear (B).

1.3.7 Exercício 7 - Métodos Numéricos e Computacionais

Implemente um algoritmo que usando o método de *Newton-Raphson* encontre o zero da função

$$f(x) = \frac{x^2 + x - 6}{2x + 1}$$

utilizando vetores e laços de repetição.

O usuário deve informar o erro e o valor inicial x_0 .

1.3.8 Exercício 8 - Métodos Numéricos e Computacionais

Implemente um algoritmo que usando o método do ponto fixo encontre o zero da função

$$f(x) = \frac{x^3}{9} + \frac{1}{3}$$

utilizando vetores e laços de repetição além das operações matemáticas necessárias para a função dada.

O usuário deve informar a precisão e o valor inicial x_0 .

1.3.9 Exercício 9 - Métodos Numéricos e Computacionais

Implemente um algoritmo que usando o método da posição falsa encontre o zero da função $f(x) = e^x - 4x^2$ utilizando vetores e laços de repetição além das operações matemáticas necessárias para a função dada. O usuário deve informar o erro e o intervalo.

1.3.10 Exercício 10 - Estatística

Implemente um algoritmo que dê ao usuário a opção de escolher entre usar a média ou a variância para calcular a probabilidade de determinado evento.

Conforme a escolha do usuário, deve ser necessária a leitura das seguintes variáveis: número de ocorrências (x), porcentagem de ocorrências (p) e grupo de controle (n) ou variância.

1.4 Estruturas de Dados Heterogêneas

1.4.1 Exercício 1 - Física

Elabore um algoritmo que realize o cálculo de equações relacionadas ao conteúdo de oscilações. Utilize um registro e declare-o como *oscilações*. O programa deve ser capaz de calcular o seguinte:

- 1º - Período do movimento harmônico;
- 2º - Frequência do movimento harmônico;
- 3º - Frequência angular do movimento harmônico;
- 4º - Constante elástica de uma mola que realize um movimento harmônico;
- 5º - Massa do pêndulo ou bloco que está executando o movimento harmônico.

Declare dentro do registro, um contador, encarregado de guardar a informação a respeito da opção escolhida para o cálculo, fornecida pelo usuário. Declare um indicador (*char*) que deve receber '*s*' ou '*n*', dependendo das informações que o usuário possui.

Exemplo: O usuário deseja calcular o período T do movimento. O programa deve ser capaz de ler se o usuário possui a frequência para calcular o período, e caso não possua, deve ser calcular pelas equações do período de acordo com o tipo de pêndulo: torção, simples ou físico.

1.4.2 Exercício 2 - Física

Elabore um programa que realize o cálculo das velocidades finais de duas partículas após uma colisão. Considere que a segunda partícula esteja parada e seja o alvo.

Declare todas as variáveis como *float*, além disso, utilize um registro, no qual devem ser declaradas todas as variáveis que serão utilizadas no programa.

Utilize a fórmula para colisões em uma dimensão, e apresente na tela as velocidades finais das partículas.

1.4.3 Exercício 3 - Métodos Numéricos e Computacionais

Crie um algoritmo que calcule a integral de uma função por métodos numéricos. O algoritmo deve implementar o método chamado de trapézio repetido, que possibilita um resultado próximo ao real. Porém, como não é um resultado exato, existe um erro quando utiliza-se este método.

Através da fórmula, calcule a integral:

$$\int_a^b f(x), dx \approx \frac{b-a}{n} [f(a) + f(b) + 2 \sum_{i=1}^{n-1} (f(x_i))]$$

Sendo a o extremo inferior do intervalo, b o extremo superior e n o número de subdivisões. O usuário deve informar o intervalo e o número de subdivisões que deseja. Quanto mais subdivisões, mais exato o resultado será.

Considere $f(x) = 2e^{0,5x}$, intervalo de 0 a 1 com 10 subdivisões. Utilize um registro para armazenar todas as variáveis.

1.5 Funções

1.5.1 Exercício 1 - Métodos Numéricos e Computacionais

Escreva um algoritmo que solucione um polinômio de grau 4 através do metodo de *Birge-Vieta* usando uma função para o polinômio e outra para sua derivada.

As funções devem retornar variáveis do tipo *float* para que na função principal seja feito o cálculo. O usuário deve informar o erro máximo, os coeficientes e o $x0$ da função.

1.5.2 Exercício 2 - Cálculo A

Elabore um algoritmo que calcule a derivada pela regra do produto, dada uma função pelo usuário. O usuário deve digitar a função, e a mesma deve ser armazenada dentro de quatro vetores com no máximo 50 posições.

Primeiramente, o usuário deverá informar quantos termos possui a função, e logo após, deverá informar a constante associada ao x e o expoente em que o x está elevado. Quanto aos vetores, um deve conter todas as constantes da função 1, o outro todos os expoentes que estão associados a função 1, o terceiro deve conter todas as constantes que multiplicam x da função 2 e o último vetor deve conter os expoentes da função 2.

Devem ser criados mais dois vetores, um para armazenar a derivada da primeira função, e outro para armazenar a derivada da segunda função. Devem conter mais dois vetores, de inteiros, no qual irá conter o tipo da função a ser derivada, que irão armazenar números de 1 a 6, sendo eles:

- 1 - Função Polinomial
- 2 - Seno
- 3 - Cosseno

4 - Tangente

5 - Logaritmo Natural

6 - Exponencial Natural.

O usuário irá digitar o tipo de função existente através do índice acima. Crie uma função do tipo *void* para apresentar a derivada e a função, de acordo com a ordem expressa pela regra do produto. Derive as funções dentro do programa principal.

1.5.3 Exercício 3 - Álgebra Linear

Crie um algoritmo que calcule a subtração ou a adição de duas matrizes de dimensão máxima 10×10 . O usuário deve digitar as matrizes e realizar uma das duas operações. Lembre-se que as matrizes devem ser da mesma ordem para serem somadas ou subtraídas, logo, o programa deve identificar se as dimensões das matrizes são as mesmas.

Divida o programa em um programa principal e em 3 funções, sendo uma para ler as duas matrizes, outra para calcular a soma ou a subtração, e outra para apresentar todas as matrizes na tela.

1.5.4 Exercício 4 - Cálculo A

Elabore um programa que calcule a integral de uma função polinomial de no máximo 10 elementos. O programa deve receber as constantes da função dentro de um vetor, os expoentes dentro de outro vetor e o resultado dos expoentes e constantes integrado, armazenando dentro de outros dois vetores.

Utilize uma função do tipo *void* para realização do cálculo da integral. Apresente na tela, usando o programa principal, a função resultante.

1.5.5 Exercício 5 - Cálculo A

Crie um código no qual o usuário forneça y_0, x_0, y_1, x_1 (reta linear) e o programa calcule o coeficiente de inclinação desta reta. Utilize conhecimentos sobre geometria para análise do problema.

Execute os cálculos dentro de uma função, a qual deve devolver um valor ao programa principal, que posteriormente apresenta o resultado para o usuário.

1.5.6 Exercício 6 - Métodos Numéricos e Computacionais

Elabore um programa que calcule, através do método do trapézio, a integral da função $f(x) = 2e^{0,5x}$, com intervalo (a, b) definido de 0 a 1.

Considere a fórmula:

$$\int_a^b f(x), dx \approx \frac{particao}{2} [f(a) + f(b)]$$

Sendo $particao = b - a$.

Para o cálculo do erro, considere:

$$Erro = \frac{(b - a)^3}{12n^2} |\ddot{f}(x)|$$

Utilize uma função que devolve o valor numérico da função avaliada no intervalo (a, b) , e outra função que devolve o valor numérico da derivada segunda da função avaliada em b .

1.5.7 Exercício 7 - Métodos Numéricos e Computacionais

Calcule, através do método *1/3 de Simpson*, a integral de $f(x) = 2e^{0,5x}$, considerando $(a, b) = (0, 1)$. Utilize a fórmula:

$$Integral = (particoes/3) * (f(a) + 4 * f(a + particoes) + f(b))$$

Sendo o número de partições dado por: $particoes = (b - a)/2$. Para realização do cálculo do erro associado a este método, utilize: $Erro = ((particoes^5)/90) * (derivadaquarta(b))$.

Utilize uma função que devolve o valor numérico da função avaliada dentro do intervalo (a, b) , e outra função que devolve o valor numérico da derivada quarta da função avaliada em b .

1.6 Ponteiros e Alocação Dinâmica

1.6.1 Exercício 1 - Estatística

Escreva um algoritmo que realize: (i) a união de dois eventos ou, (ii) a intersecção desses eventos dando a opção ao usuário de escolher.

Além disso utilize funções do tipo *void* para as operações e realize as operações até que o usuário deseje sair do sistema.

Os eventos devem ser guardados em vetores alocados dinamicamente, os resultados devem ser apresentados na própria função.

1.6.2 Exercício 2 - Física

Elabore um programa que decomponha um número de forças, fornecido pelo usuário, em relação ao eixo x e ao eixo y . Deve ser alocado dinamicamente duas variáveis, uma delas é o módulo da força f , e a outra, a variável que irá conter o ângulo das forças produzidas em relação ao eixo x , denominada *ang*.

O número de forças deverá ser fornecido pelo usuário. Devem ser criadas duas variáveis, componente resultante da força em x (fx) e o componente resultante da força em y (fy), que irão conter o resultado de todas as forças decompostas em relação ao eixo x e y , respectivamente.

Exemplo: Se o usuário declarar que existem 3 forças agindo no sistema estudado, o programa deve fazer uma alocação dinâmica, e requisitar o valor da força e o ângulo entre a força e o eixo x do sistema. A variável resultante x guarda o somatório da decomposição das 3 forças em relação ao eixo x . A variável resultante y , guarda o somatório da decomposição das 3 forças em relação ao eixo y . Após serem realizadas todas as operações, apresente os resultados e libere a memória dos vetores criados.

1.6.3 Exercício 3 - Álgebra Linear

Escreva um código no qual realize o cálculo de um sistema linear, da mesma forma que o Exercício 3 (Seção 1.3.3), de Álgebra Linear. A única diferença, é que deve-se utilizar alocação dinâmica de memória.

A matriz deve ser alocada dinamicamente, juntamente com o vetor.

1.6.4 Exercício 4 - Álgebra Linear

Elabore um algoritmo que calcule o produto escalar entre dois vetores alocados dinamicamente, de dimensão especificada pelo usuário. O produto escalar consiste em multiplicar termos de mesmo índice dos vetores.

Logo após todas as multiplicações, o produto consiste em devolver um número escalar, ou seja, somar todas as multiplicações efetuadas anteriormente.

2 Algoritmos Resolvidos

2.1 Desvios Condicionais

2.1.1 Exercício 1

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main ()
{
    float cat_oposto, cat_adjacente, hipotenusa, angulo;
    char caractere;
    int indicador;
    printf ("Digite o numero de acordo com a operacao desejada:\n1-Encontrar o cateto
           oposto.\n2-Encontrar o cateto adjacente.\n3-Encontrar a hipotenusa.\n4-
           encontrar o valor do angulo em radianos: ");
    scanf ("%d",&indicador);
    if (indicador==1)
    {
        printf ("Possui a Hipotenusa?(s ou n): ");
        scanf (" %c",&caractere);
        printf ("Informe o angulo considerado em radianos: ");
        scanf ("%f",&angulo);
        if (caractere=='s')
        {
            printf ("Informe a hipotenusa: ");
            scanf ("%f",&hipotenusa);
            cat_oposto=sin(angulo)*hipotenusa;
        }
        else
        {
            printf ("Informe o cateto adjacente: ");
            scanf ("%f",&cat_adjacente);
            cat_oposto=tan(angulo)*cat_adjacente;
        }
        printf ("\nCateto oposto: %.4f\n\n",cat_oposto);
    }
    else if (indicador==2)
    {
        printf ("Possui a Hipotenusa?(s ou n): ");
        scanf (" %c",&caractere);
        printf ("Informe o angulo considerado em radianos: ");
        scanf ("%f",&angulo);
        if (caractere=='s')
        {
            printf ("Informe a hipotenusa: ");
            scanf ("%f",&hipotenusa);
            cat_adjacente=cos(angulo)*hipotenusa;
        }
        else
        {
            printf ("Digite o cateto oposto: ");
            scanf ("%f",&cat_oposto);
            cat_adjacente=cat_oposto/tan(angulo);
        }
        printf ("\nCateto Adjacente: %.4f\n\n",cat_adjacente);
    }
    else if (indicador==3)
    {
        printf ("Possui o cateto oposto?(s ou n): ");
        scanf (" %c",&caractere);
        printf ("Informe o angulo considerado em radianos: ");
        scanf ("%f",&angulo);
        if (caractere=='s')
```

```

        {
            printf ("Digite o cateto oposto: ");
            scanf ("%f",&cat_oposto);
            hipotenusa=cat_oposto/sin(angulo);
        }
        else
        {
            printf ("Informe o cateto adjacente: ");
            scanf ("%f",&cat_adjacente);
            hipotenusa=cat_adjacente/cos(angulo);
        }
        printf ("\nHipotenusa: %.4f\n\n",hipotenusa);
    }
    else
    {
        printf ("Possui o cateto oposto?(s ou n): ");
        scanf (" %c",&caractere);
        if (caractere=='s')
        {
            printf ("Possui a hipotenusa? (s ou n): ");
            scanf (" %c",&caractere);
            if (caractere=='s')
            {
                printf ("Digite o cateto oposto: ");
                scanf ("%f",&cat_oposto);
                printf ("Digite a hipotenusa: ");
                scanf ("%f",&hipotenusa);
                angulo=asin(cat_oposto/hipotenusa);
            }
            else
            {
                printf ("Digite o cateto oposto: ");
                scanf ("%f",&cat_oposto);
                printf ("Digite o cateto adjacente: ");
                scanf ("%f",&cat_adjacente);
                angulo=atan(cat_oposto/cat_adjacente);
            }
        }
        else
        {
            printf ("Digite o cateto adjacente: ");
            scanf ("%f",&cat_adjacente);
            printf ("Digite a hipotenusa: ");
            scanf ("%f",&hipotenusa);
            angulo=acos(cat_adjacente/hipotenusa);
        }
        printf ("\nAngulo procurado: %.4f\n\n",angulo);
    }
    system ("pause");
    return 0;
}

```

2.1.2 Exercício 2

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main ()
{
    float x,desvio_padrao,margem_erro,nivel_conf,P,lim_sup,lim_inf;
    int indicador,n;
    printf ("Digite 1 se deseja usar a tabela Z ou 2 caso deseje a T: ");
    scanf ("%d",&indicador);
    printf ("Digite a media amostral a ser considerada: ");
    scanf ("%f",&x);
    printf ("Digite o nivel de confianca considerado em porcentagem: ");
    scanf ("%f",&nivel_conf);
    printf ("Digite o numero de individuos considerados da amostra: ");

```

```

scanf ("%d",&n);
if (indicador==1)
{
    printf ("Digite a probabilidade de Z para Z %.3f: ",nivel_conf/200);
    scanf ("%f",&P);
    printf ("Digite o desvio padrao da populacao: ");
    scanf ("%f",&desvio_padrao);
}
if (indicador==2)
{
    printf ("Digite a probabilidade de T para T %.3f, considerando %d graus
de liberdade: ",1-nivel_conf/100,n-1);
    scanf ("%f",&P);
    printf ("Digite o desvio padrao da amostra: ");
    scanf ("%f",&desvio_padrao);
}
margem_erro=P*(desvio_padrao/sqrt(n));
printf ("\nMargem de erro: %f para mais ou para menos.\n",margem_erro);
lim_inf=x-margem_erro;
lim_sup=x+margem_erro;
printf ("Intervalo de confianca: [%.3f - %.3f].\n\n",lim_inf,lim_sup);
system ("pause");
return 0;
}

```

2.1.3 Exercício 3

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main ()
{
    float nivel_significancia,H0,H1,P,z_calc,media_amostra,media_pop,desvio_padrao;
    int n;
    printf ("Digite a hipotese nula,H0: ");
    scanf ("%f",&H0);
    printf ("Digite a hipotese alternativa, H1 (qualquer valor diferente de H0): ");
    scanf ("%f",&H1);
    printf ("Digite o nivel de significancia a ser considerado(em decimais): ");
    scanf ("%f",&nivel_significancia);
    printf ("Digite a media da amostra e a media da populacao, respectivamente!\n");
    scanf ("%f%f",&media_amostra,&media_pop);
    printf ("Digite o desvio padrao pertencente a populacao: ");
    scanf ("%f",&desvio_padrao);
    printf ("Digite o numero de individuos pertencentes a amostra: ");
    scanf ("%d",&n);
    z_calc=(sqrt(n)*(media_amostra-media_pop))/desvio_padrao;
    printf ("Informe o valor critico associado a z%.2f: ",z_calc);
    scanf ("%f",&P);
    if (z_calc<0&&P>nivel_significancia||z_calc>0&&P<nivel_significancia)
        printf ("Aceitamos a hipotese nula H0 e rejeitamos a hipotese
alternativa H1.\n");
    else
        printf ("Aceitamos a hipotese alternativa H1 e rejeitamos a hipotese nula
H0.\n\n");
    system ("pause");
    return 0;
}

```

2.1.4 Exercício 4

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main ()
{
    float nivel_significancia,H0,H1,P,t_calc,media_amostra,media_pop,desvio_padrao,
    t_tab;
    int n;
    printf ("Digite a hipotese nula,H0: ");
    scanf ("%f",&H0);
    printf ("Digite a hipotese alternativa, H1 (qualquer valor diferente de H0): ");
    scanf ("%f",&H1);
    printf ("Digite o nivel de significancia a ser considerado(em porcentagem): ");
    scanf ("%f",&nivel_significancia);
    printf ("Digite a media da amostra e a media da populacao, respectivamente!\n");
    scanf ("%f%f",&media_amostra,&media_pop);
    printf ("Digite o desvio padrao pertencente a amostra: ");
    scanf ("%f",&desvio_padrao);
    printf ("Digite o numero de individuos pertencentes a amostra: ");
    scanf ("%d",&n);
    t_calc=(media_amostra-media_pop)/(desvio_padrao)/(sqrt(n));
    if (t_calc<0)
        t_calc=-t_calc;
    P=t_calc;
    printf ("Informe o T tabelado, levando em consideracao o nivel de significancia
    de %.3f, %d graus de liberdade e levando em consideracao quantas caudas
    possui a distribuicao estudada: ",nivel_significancia,n-1);
    scanf ("%f",&t_tab);
    if (t_tab>t_calc)
        printf ("Aceitamos a hipotese nula H0 e rejeitamos a hipotese
        alternativa H1.\n");
    else
        printf ("Aceitamos a hipotese alternativa H1 e rejeitamos a hipotese nula
        H0.\n\n");
    system ("pause");
    return 0;
}

```

2.1.5 Exercício 5

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <locale.h>
#define G 9.80665

int main ()
{
    float energia_mecanica,energia_cinetica,energia_potencial,masse,velocidade,altura
    ;
    int indicador;
    setlocale(LC_ALL,"portuguese");
    printf ("Digite o número de acordo com a operação desejada:\n1-Energia Mecânica\n2
    -Energia Potencial\n3-Energia Cinética\n4-Massa\n5-Velocidade\n6-Altura\
    nOperação desejada: ");
    scanf ("%d",&indicador);
    if (indicador==1)
    {
        printf ("Digite a massa em KG associada ao sistema: ");
        scanf ("%f",&masse);
        printf ("Digite a velocidade em m/s associada ao sistema: ");
        scanf ("%f",&velocidade);
        printf ("Digite a altura H em metros: ");
    }
}

```

```
scanf ("%f",&altura);
energia_mecanica=massa*G*altura+0.5*massa*pow(velocidade,2);
printf ("Energia mecânica associada ao sistema: %.3f Joules\n\n",
        energia_mecanica);
}
else if (indicador==2)
{
printf ("Informe a Energia Mecânica em Joules associada ao sistema: ");
scanf ("%f",&energia_mecanica);
printf ("Digite a massa em KG associada ao sistema: ");
scanf ("%f",&massa);
printf ("Digite a velocidade em m/s associada ao sistema: ");
scanf ("%f",&velocidade);
energia_potencial=energia_mecanica-0.5*massa*pow(velocidade,2);
printf ("Energia Potencial associada ao sistema: %.4f Joules\n\n",
        energia_potencial);
}
else if (indicador==3)
{
printf ("Informe a Energia Mecânica em Joules associada ao sistema: ");
scanf ("%f",&energia_mecanica);
printf ("Digite a massa em KG associada ao sistema: ");
scanf ("%f",&massa);
printf ("Digite a altura H em metros: ");
scanf ("%f",&altura);
energia_cinetica=energia_mecanica-massa*G*altura;
printf ("Energia Cinética associada ao sistema: %.4f Joules\n\n",
        energia_cinetica);
}
else if (indicador==4)
{
printf ("Informe a Energia Mecânica em Joules associada ao sistema: ");
scanf ("%f",&energia_mecanica);
printf ("Informe a altura a ser considerada em metros: ");
scanf ("%f",&altura);
printf ("Digite a velocidade em m/s associada ao sistema: ");
scanf ("%f",&velocidade);
massa=energia_mecanica/(G*altura + 0.5*pow(velocidade,2));
printf ("Massa associada ao sistema: %.4f KG\n\n",massa);
}
else if (indicador==5)
{
printf ("Informe a Energia Mecânica em Joules associada ao sistema: ");
scanf ("%f",&energia_mecanica);
printf ("Informe a altura a ser considerada em metros: ");
scanf ("%f",&altura);
printf ("Digite a massa em KG associada ao sistema: ");
scanf ("%f",&massa);
velocidade=sqrt((2.0/massa)*(energia_mecanica-massa*G*altura));
printf ("Velocidade associada ao sistema: %.4f m/s\n\n",velocidade);
}
else if (indicador==6)
{
printf ("Informe a Energia Mecânica em Joules associada ao sistema: ");
scanf ("%f",&energia_mecanica);
printf ("Digite a massa em KG associada ao sistema: ");
scanf ("%f",&massa);
printf ("Digite a velocidade em m/s associada ao sistema: ");
scanf ("%f",&velocidade);
altura=(1.0/massa*G)*(energia_mecanica-0.5*massa*pow(velocidade,2));
printf ("Altura associada ao sistema no dado ponto: %.4f metros\n\n",
        altura);
}
else printf ("Operação inválida! Digite números de 1 a 6!\n\n");
system ("pause");
```

```
    return 0;
}
```

2.2 Laços de Repetição

2.2.1 Exercício 1

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main ()
{
    float integral=0,lim_inferior,lim_superior,particoes,x,y;
    printf ("Digite o Limite inferior e superior respectivamente:\n");
    scanf ("%f%f",&lim_inferior,&lim_superior);
    particoes=(lim_superior-lim_inferior)/10000;
    x=lim_inferior;
    do
    {
        y=2*x; //Função considerada, deve ser alterada.
        integral+=particoes*y;
        x+=particoes;
    }
    while (x<lim_superior);
    printf ("Resultado da Integral definida de %f a %f: %f\n\n",lim_inferior,
        lim_superior,integral);
    system ("pause");
    return 0;
}
```

2.2.2 Exercício 2

```
#include <stdio.h>
int main ()
{
    int x,fatorial=1,i;
    do
    {
        printf ("Digite o valor no qual deseja calcular o fatorial: ");
        scanf ("%d",&x);
        if (x<0)
            printf ("Valor invalido, digite um valor maior ou igual a 0!\n\n"
                );
    }
    while(x<0);
    for (i=x;i>=0;i--)
    {
        if (i==0)
            fatorial*=1;
        else
            fatorial*=i;
    }
    printf ("Fatorial: %d\n\n",fatorial);
    return 0;
}
```

2.2.3 Exercício 3

```
#include <stdio.h>
```

```

int main ()
{
    int x,somatorio,i;
    do
    {
        printf ("Digite o valor no qual deseja calcular o somatorio: ");
        scanf ("%d",&x);
        if (x<0)
            printf ("Valor invalido, digite um valor maior ou igual a 0!\n\n"
                );
    }
    while(x<0);
    for (i=x;i>0;i--)
    {
        somatorio+=i;
    }
    printf ("Somatorio: %d\n\n",somatorio);
    return 0;
}

```

2.2.4 Exercício 4

```

#include <stdio.h>
#include <math.h>
float function (float x)
{
    float funcao;
    funcao=2*x-2;           //Deve-se sempre ser alterada a função.
    return funcao;
}
int main ()
{
    float xa,xb,xc,erro,parada=1;
    printf ("Digite a margem de erro aceitavel: ");
    scanf ("%f",&erro);
    printf ("Digite o valor de Xa: ");
    scanf ("%f",&xa);
    printf ("Digite o valor de Xb: ");
    scanf ("%f",&xb);
    do
    {
        xc=(xa+xb)/2;
        if (function(xa)*function(xc)<0)
            xb=xc;
        if (function(xb)*function(xc)<0)
            xa=xc;
        parada=(xb-xa)/2;
        abs(function(xc));
    }
    while (parada>erro&&function(xc)>erro);
    printf ("\nRaiz da equacao no intervalo dado: %f\n\n",xc);
    return 0;
}

```

2.2.5 Exercício 5

```

#include <stdio.h>
#include <math.h>

int main ( )
{

```

```
double x, y;
double precisao;
int k=1;
double media, funcao_med, funcao_x, funcao_y, teste;

printf("-Insira o intervalo-\n");
printf("\nDigite o valor de x: ");
scanf("%lf", &x);
printf("\nDigite o valor de Y: ");
scanf("%lf", &y);
printf("\nDigite o valor de precisao: ");
scanf("%lf", &precisao);

while (teste>precisao)
{
    funcao_x=exp(x)-4*pow(x,2);
    funcao_y=exp(y)-4*pow(y,2);
    media=(x+y)/2;
    funcao_med=exp(media)-4*pow(media,2);

    if (funcao_x*funcao_med<0.0)
    {
        y=media;
    }
    if (funcao_y*funcao_med<0.0)
    {
        x=media;
    }

    teste=(y-x)/2;
    printf("Iteracao:%d \n", k);
    printf("Intervalo atualizado: x=%lf,y=%lf \n", x, y);
    k++;
}
printf ("\n O valor aproximado da raiz sera: %lf", media);
return 0;
}
```

2.3 Vetores e Matrizes

2.3.1 Exercício 1

```
#include <stdio.h>
#define N 3
int main ()
{
    int a[N],b[N],c[N],i,n=1;
    for (i=1;i<=N;i++)
    {
        printf ("\nDigite o valor associado a posicao %d do vetor A: ",i);
        scanf ("%d",&a[i]);
        printf ("Digite o valor associado a posicao %d do vetor B: ",i);
        scanf ("%d",&b[i]);
    }
    c[0]=a[2]*b[3]-a[3]*b[2];
    c[1]=a[3]*b[1]-a[1]*b[3];
    c[2]=a[1]*b[2]-a[2]*b[1];
    printf ("Vetor A:\n\n");
    for (i=1;i<=N;i++)
    {
        printf ("%d\n",a[i]);
    }
}
```



```

}
printf ("Vetor B:\n\n");
for (i=1;i<=N;i++)
{
    printf ("%d\n",b[i]);
}
printf ("Vetor C, resultante do produto vetorial(perpendicular aos vetores A e B)
:\n\n");
for (i=0;i<N;i++)
{
    printf ("%d\n",c[i]);
}
return 0;
}

```

2.3.2 Exercício 2

```

#include <stdio.h>
#define N 3
int main ()
{
    float matriz[N][N],determinante;
    int i,j;
    printf ("Digite a matriz a ser calculado o determinante:\n\n");
    for (i=0;i<N;i++)
    {
        for (j=0;j<N;j++)
        {
            printf ("Digite o termo associado a linha %d e coluna %d da
matriz: ",i+1,j+1);
            scanf ("%f",&matriz[i][j]);
        }
        printf ("\n");
    }
    determinante=matriz[0][0]*matriz[1][1]*matriz[2][2]+matriz[0][1]*matriz[1][2]*
matriz[2][0]+matriz[0][2]*matriz[1][0]*matriz[2][1]-(matriz[0][1]*matriz
[1][0]*matriz[2][2]+matriz[0][0]*matriz[1][2]*matriz[2][1]+matriz[0][2]*
matriz[1][1]*matriz[2][0]);
    printf ("Determinante associado a matriz: %.3f\n\n",determinante);
    return 0;
}

```

2.3.3 Exercício 3

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <locale.h>

int main(){

    double desviom=0,variancia=0,desviop=0;
    float a,b,vetmed[8],vetmodulo[8],vetmult[8],a1=0,b1=0,f=0,n=0,somatorio=0;
    float vetfreq[8],vetmodulo2[8],vetmult2[8],cv=0,somatorio2=0;
    int cont=0,w;
    setlocale(LC_ALL, "Portuguese");

    printf("Digite o intervalo\n");

    while(cont<8){

        scanf("\n%f ",&a);
        scanf("\n%f ",&b);
    }
}

```

```
system("CLS");

vetmed[cont]=((a+b)/2);

    if(cont==0){
        a1=a;
    }
    if(cont==7){
        b1=b;
    }
cont++;
}
f=(a1+b1)/2;

cont=0;
while(cont<8){
    vetmodulo[cont]=vetmed[cont]-f;
    vetmodulo2[cont]=pow(vetmodulo[cont],2);
    cont++;
}
cont=0;
while(cont<8){
    if(vetmodulo[cont]<0){
        vetmodulo[cont]=vetmodulo[cont]*(-1);
        cont++;
    }
    else{
        cont++;
    }
}

system("CLS");

printf("Digite a frequência de cada intervalo");
cont=0;
while(cont<8){
    scanf("\n%f ", &vetfreq[cont]);
    n=n+vetfreq[cont];
    cont++;
}
cont=0;
while(cont<8){
    vetmult[cont]=vetmodulo[cont]*vetfreq[cont];
    vetmult2[cont]=vetmodulo2[cont]*vetfreq[cont];
    cont++;
}
cont=0;
while(cont<8){
    somatorio=somatorio+vetmult[cont];
    somatorio2=somatorio2+vetmult2[cont];
    cont++;
}
system("CLS");
desviom=(somatorio/n);
variancia=(somatorio2/n);
desviop=sqrt(variancia);
cv=(desviop/f)*100;
printf("O desvio padrão é %lf\n", desviop);
printf("O desvio médio é %lf\n", desviom);
printf("A variância é %lf\n", variancia);
```

```
printf("O coeficiente de variância é %f", cv);
```

```
}
```

2.3.4 Exercício 4

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <locale.h>

int main(){

    double desviom=0, variancia=0, desviop=0;
    float a,b, vetmed[8], vetmodulo[8], vetmult[8], a1=0, b1=0, f=0, n=0, somatorio=0;
    float vetfreq[8], vetmodulo2[8], vetmult2[8], cv=0, somatorio2=0;
    int cont=0, w;
    setlocale(LC_ALL, "Portuguese");

    printf("Digite o intervalo\n");

    while(cont<8){

        scanf("\n%f ", &a);
        scanf("\n%f ", &b);

        system("CLS");

        vetmed[cont]=((a+b)/2);

        if(cont==0){
            a1=a;
        }
        if(cont==7){
            b1=b;
        }
        cont++;
    }
    f=(a1+b1)/2;

    cont=0;
    while(cont<8){
        vetmodulo[cont]=vetmed[cont]-f;
        vetmodulo2[cont]=pow(vetmodulo[cont], 2);
        cont++;
    }
    cont=0;
    while(cont<8){
        if(vetmodulo[cont]<0){
            vetmodulo[cont]=vetmodulo[cont]*(-1);
            cont++;
        }
        else{
            cont++;
        }
    }

    system("CLS");
```

```

printf("Digite a frequência de cada intervalo");
cont=0;
while(cont<8){
    scanf("\n%f ", &vetfreq[cont]);
    n=n+vetfreq[cont];
    cont++;
}
cont=0;
while(cont<8){
    vetmult[cont]=vetmodulo[cont]*vetfreq[cont];
    vetmult2[cont]=vetmodulo2[cont]*vetfreq[cont];
    cont++;
}
cont=0;
while(cont<8){
    somatorio=somatorio+vetmult[cont];
    somatorio2=somatorio2+vetmult2[cont];
    cont++;
}
system("CLS");
desviom=(somatorio/n);
variancia=(somatorio2/n);
desviop=sqrt(variancia);
cv=(desviop/f)*100;
printf("O desvio padrão é %lf\n", desviop);
printf("O desvio médio é %lf\n",desviom);
printf("A variância é %lf\n",variancia);
printf("O coeficiente de variância é %f", cv);
}

```

2.3.5 Exercício 5

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main(){

    float xo[3],a1[3],a2[3],a3[3], a4[3],b[3];
    int cont=0;
    float c[3];
    while(cont<4){
        printf("digite o valor de x0");
        scanf("%f", &xo[cont]);
        cont++;
    }
    cont=0;
    while(cont<4){
        printf("digite o valor de B");
        scanf("%f", &b[cont]);
        cont++;
    }
    cont=0;
}

```

```

        while(cont<4){
        printf("digiter o valor de A1");
            scanf("%f", &a1[cont]);
            cont++;
        }

        cont=0;
        while(cont<4){
        printf("digiter o valor de A2");
            scanf("%f", &a2[cont]);
            cont++;
        }

        cont=0;
        while(cont<4){
        printf("digiter o valor de A3");
            scanf("%f", &a3[cont]);
            cont++;
        }

        cont=0;
        while(cont<4){
        printf("digiter o valor de A4");
            scanf("%f", &a4[cont]);
            cont++;
        }
        cont=0;

        while (cont<200){
            xo[0]=(b[0]-a1[1]*xo[1]-a1[2]*xo[2]-a1[3]*xo[3])/a1[0];
            xo[1]=(b[1]-a2[0]*xo[0]-a2[2]*xo[2]-a2[3]*xo[3])/a2[1];
            xo[2]=(b[2]-a3[0]*xo[0]-a3[1]*xo[1]-a3[3]*xo[3])/a3[2];
            xo[3]=(b[3]-a4[0]*xo[0]-a4[1]*xo[1]-a4[2]*xo[2])/a4[3];
            cont++;
        }
        printf("%f\n", xo[0]);
            printf("%f\n", xo[1]);
                printf("%f \n", xo[2]);
                    printf("%f\n", xo[3]);

    }

```

2.3.6 Exercício 6

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main(){

    float xo[3], a1[3], a2[3], a3[3], a4[3], b[3], r1, r2, r3, r4;
    int cont=0;

    while(cont<4){
        printf("digite o valor de x0");
        scanf("%f", &xo[cont]);
        cont++;
    }
    cont=0;
    while(cont<4){
        printf("digiter o valor de B");
        scanf("%f", &b[cont]);
    }
}

```

```

        cont++;
    }
    cont=0;
    while(cont<4){
        printf("digite o valor de A1");
        scanf("%f", &a1[cont]);
        cont++;
    }

    cont=0;
    while(cont<4){
        printf("digite o valor de A2");
        scanf("%f", &a2[cont]);
        cont++;
    }

    cont=0;
    while(cont<4){
        printf("digite o valor de A3");
        scanf("%f", &a3[cont]);
        cont++;
    }

    cont=0;
    while(cont<4){
        printf("digite o valor de A4");
        scanf("%f", &a4[cont]);
        cont++;
    }
    cont=0;

    while (cont<100){
        r1=(b[0]-a1[1]*xo[1]-a1[2]*xo[2]-a1[3]*xo[3])/a1[0];

        r2=(b[1]-a2[0]*r1-a2[2]*xo[2]-a2[3]*xo[3])/a2[1];

        r3=(b[2]-a3[0]*r1-a3[1]*r2-a3[3]*xo[3])/a3[2];

        r4=(b[3]-a4[0]*r1-a4[1]*r2-a4[2]*r3)/a4[3];

        xo[0]=r1;
        xo[1]=r2;
        xo[2]=r3;
        xo[3]=r4;

        cont++;
    }
    printf("%f\n\n", xo[0]);
    printf("%f\n\n", xo[1]);
    printf("%f\n\n", xo[2]);
    printf("%f\n\n", xo[3]);

}

```

2.3.7 Exercício 7

```

#include<stdio.h>
#include<math.h>
int main (){

```

```
double e,x[99];
int k=1;
printf("Digite o valor de x0:");
scanf("%lf",&x[0]);
printf("Informe o valor do E:");
scanf("%f",&e);
x[1]=x[0]-((pow(x[0],2)+x[0]-6)/(2*x[0]+1));

while((fabs(x[k]-x[k-1])>e))
{
    printf("Iteracao: %d\n ", k);
    k++;
    x[k]=x[k-1]-((pow(x[k-1],2)+x[k-1]-6)/(2*x[k-1]+1));
    printf("x%d= %lf\n ",k-1, x[k]);
}
printf("o resultado final se deu em x%d = %lf ",k-1,x[k]);
}
```

2.3.8 Exercício 8

```
#include<stdio.h>
#include<math.h>
int main ()
{
double precisao, x[99];
int k=1;
printf("Digite o valor de x0: ");
scanf("%lf",&x[0]);

printf("\nInforme o valor da precisao: ");
scanf("%lf",&precisao);

x[1]=(pow(x[0],3))/9+(0.3333);

while((fabs(x[k]-x[k-1])>precisao))
{
    printf("\nIteracao: %d\n", k);
    k++;

    x[k]=(pow(x[k-1],3)/9)+(0.333);

    printf("\nx%d = %lf\n ",k-1, x[k]);
}
printf("\n o resultado final se deu em x%d=%lf",k-1,x[k]);
}
```

2.3.9 Exercício 9

```
#include <stdio.h>
#include <math.h>
```

```
int main ( )
{
    double x, y;
    double precisao;
    int k=1;
    double media=0, funcao_med=1, funcao_x=0, funcao_y=0;

    printf("-Insira o intervalo-\n");
    printf("\nDigite o valor de x: ");
    scanf("%lf", &x);
    printf("\nDigite o valor de Y: ");
    scanf("%lf", &y);
    printf("\nDigite o valor de precisao: ");
    scanf("%lf", &precisao);

    while (fabs(funcao_med)>precisao)
    {

        funcao_x=exp(x)-4*pow(x,2);
        funcao_y=exp(y)-4*pow(y,2);
        media=(x*fabs(funcao_y)+y*fabs(funcao_x))/(fabs(funcao_x)+fabs(funcao_y))
        ;
        funcao_med=exp(media)-4*pow(media,2);

        if (funcao_x*funcao_med<0)
        {
            y=media;
        }
        else
        {
            x=media;
        }

        printf("Iteracao:%d \n", k);
        printf("Intervalo atualizado: x=%lf,y=%lf \n", x, y);
        k++;

    }
    printf ("\n O valor aproximado da raiz sera: %lf", media);
    return 0;
}
```

2.3.10 Exercício 10

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <locale.h>
#define e 2.718828

int main(){

    int z,cont=0,x=0;
    float p,n,lambda,variancia,xfat=1;
    double probabilidade=0;
    setlocale(LC_ALL,"Portuguese");
```



```

printf("digite 1 para usar a média e 2 para usar a variância:\n");
scanf("%d", &z);
switch (z){
    case 1:
        system("CLS");
        printf("digite o n ");
        scanf("%f", &n);
        printf("\ndigite o p ");
        scanf("%f", &p);
        printf("\ndigite x ");
        scanf("%d", &x);
        lambda=n*p;
        cont=x;
        while (cont>=1){
            xfat=xfat*cont;
            cont--;
        }
        probabilidade=(pow(e,(-1*lambda))*pow(lambda,x))/xfat;
        break;
    case 2:
        system("CLS");
        printf("digite a variância: ");
        scanf("%f", &variancia);
        printf("\ndigite x ");
        scanf("%d", &x);
        lambda=variancia;
        cont=x;
        while (cont>=1){
            xfat=xfat*cont;
            cont--;
        }
        probabilidade=(pow(e,(-1*lambda))*pow(lambda,x))/xfat;
}
system("CLS");
printf("\na probabilidade de acontecer é:\n\n%lf", probabilidade );
}

```

2.4 Estruturas de Dados Heterogêneas

2.4.1 Exercício 1

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <locale.h>
#define PI 3.141592654
#define G 9.81
struct registro{
    float T,f,w,m,k,I,l;
    int cont;
    char ind;
};
int main ()
{
    struct registro oscilacoes;
    setlocale(LC_ALL,"portuguese");
    printf ("O programa realiza calculos de problemas de fisica relacionados à
    oscilações.\n");
}

```

```

printf ("Digite o número de acordo com o que deseja calcular:\n1-Período do
movimento harmônico.\n2-Frequência do movimento harmônico.\n3-Frequência
Angular do movimento harmônico.\n4-Constante elástica da mola.\n5-Massa
associada à uma oscilação.\nOpção escolhida: ");
scanf ("%d",&oscilacoes.cont);
if (oscilacoes.cont==1)
{
    printf ("Possui a frequência?(s ou n): ");
    scanf (" %c",&oscilacoes.ind);
    if (oscilacoes.ind=='s')
    {
        printf ("Digite a frequência associada ao movimento(em HZ): ");
        scanf ("%f",&oscilacoes.f);
        oscilacoes.T=1.0/oscilacoes.f;
        printf ("Para encontrar o período, levamos em consideração que o
mesmo é o inverso da frequência, logo: T=1/f, resultando em
T=%f segundos.\n",oscilacoes.T);
    }
    else
    {
        printf ("Digite 1 caso o problema se trate de um pêndulo
de torção, 2 para pêndulo simples e 3 para pêndulo físico e
4 para oscilador linear: ");
        scanf ("%d",&oscilacoes.cont);
        if (oscilacoes.cont==1)
        {
            printf ("Digite o momento de inércia do objeto
considerado: ");
            scanf ("%f",&oscilacoes.I);
            printf ("Digite a constante de torção: ");
            scanf ("%f",&oscilacoes.k);
            oscilacoes.T=2*PI*sqrt(oscilacoes.I/oscilacoes.k)
;
            printf ("O período de um pêndulo de torção é dado
pela seguinte relação: T=2*pi*sqrt(I/k),
resultando em T=%f segundos.",oscilacoes.T);
        }
        else if (oscilacoes.cont==2)
        {
            printf ("Digite a distância L entre o eixo de
rotação do pêndulo e o objeto considerado: ");
            scanf ("%f",&oscilacoes.l);
            oscilacoes.T=2*PI*sqrt(oscilacoes.l/G);
            printf ("O período de um pêndulo simples é dado
pela seguinte relação: T=2*pi*sqrt(L/G), onde
G é a gravidade, tomada como 9.81 m/s²,
resultando em T=%f segundos.",oscilacoes.T);
        }
        else if (oscilacoes.cont==3)
        {
            printf ("Digite o momento de inércia associado ao
pêndulo físico: ");
            scanf ("%f",&oscilacoes.I);
            printf ("Digite a massa do objeto em KG, e a
altura H respectivamente, dada pela
distância em metros entre o centro de
oscilações e o centro de massa do objeto: ")
;
            scanf ("%f%f",&oscilacoes.m,&oscilacoes.l);
            oscilacoes.T=2*PI*sqrt(oscilacoes.I/(oscilacoes.m
*G*oscilacoes.l));
            printf ("O período de um pêndulo físico é dado
pela seguinte relação: T=2*pi*sqrt(I/mGh),
onde G é a gravidade, tomada como 9.81 m/s²,
resultando em T=%f segundos.",oscilacoes.T)

```

```

        }
        else
        {
            printf ("Digite a massa em KG e a constante
elástica da mola em N/m, respectivamente: ");
            scanf ("%f%f",&oscilacoes.m,&oscilacoes.k);
            oscilacoes.T=2*PI*sqrt(oscilacoes.m/oscilacoes.k)
            ;
            printf ("O período de um oscilador harmônico
simples é dado pela seguinte relação: T=2*pi
*sqrt(m/k), resultando em T=%f segundos.",
            oscilacoes.T);
        }
    }
}
if (oscilacoes.cont==2)
{
    printf ("Forneça o período em Segundos: ");
    scanf ("%f",&oscilacoes.T);
    oscilacoes.f=1.0/oscilacoes.T;
    printf ("A frequência, por ser o inverso do período, é dada pela equação
f=1/T, resultando em %f Hz.\n",oscilacoes.f);
}
if (oscilacoes.cont==3)
{
    printf ("Possui a Frequência do movimento? (s ou n): ");
    scanf (" %c",&oscilacoes.ind);
    if (oscilacoes.ind=='s')
    {
        printf ("Digite-a em Hz: ");
        scanf ("%f",&oscilacoes.f);
        oscilacoes.w=2*PI*oscilacoes.f;
        printf ("A frequência angular do movimento pode ser dada pela
seguinte relação: w=2*pi*f, onde f é a frequência informada
e w é a frequência angular em rad/s. Logo, w=%f rad/s.\n",
            oscilacoes.w);
    }
    else if (oscilacoes.ind=='n')
    {
        printf ("Possui o período associado ao movimento? (s ou n): ");
        scanf (" %c",&oscilacoes.ind);
        if (oscilacoes.ind=='s')
        {
            printf ("Digite-o em Segundos: ");
            scanf ("%f",&oscilacoes.T);
            oscilacoes.w=(2*PI)/oscilacoes.T;
            printf ("A frequência angular do movimento pode ser dada
pela seguinte relação: w=2*pi/T, onde T é o período
dado e w é a frequência angular em rad/s. Logo, w=%f
rad/s.\n",oscilacoes.w);
        }
        else if (oscilacoes.ind=='n')
        {
            printf ("Trata-se de um oscilador linear, onde a massa e
a constante elástica são conhecidas? (s ou n): ");
            scanf (" %c",&oscilacoes.ind);
            if (oscilacoes.ind=='s')
            {
                printf ("Digite a constante elástica em N/m e a
massa em KG, respectivamente:");
                scanf ("%f%f",&oscilacoes.k,&oscilacoes.m);
                oscilacoes.w=sqrt(oscilacoes.k/oscilacoes.m);
                printf ("A frequência angular é dada pela equação
: w=sqrt (k/m), onde K é a constante
elástica e m a massa associada ao oscilador
linear, logo w=%f rad/s\n",oscilacoes.w);
            }
        }
    }
}
}

```

```

    }
}
if (oscilacoes.cont==4)
{
    printf ("Digite a frequência angular associada ao movimento: ");
    scanf ("%f",&oscilacoes.w);
    printf ("Digite a massa do objeto: ");
    scanf ("%f",&oscilacoes.m);
    oscilacoes.k=pow(oscilacoes.w,2)*oscilacoes.m;
    printf ("Temos a expressão  $k=w^2*m$ , que resulta na constante elástica da
        mola. Logo,  $K=%f$  N/m.\n",oscilacoes.k);
}
if (oscilacoes.cont==5)
{
    printf ("Digite a frequência angular associada ao movimento: ");
    scanf ("%f",&oscilacoes.w);
    printf ("Digite a constante elástica da mola: ");
    scanf ("%f",&oscilacoes.k);
    oscilacoes.m=oscilacoes.k/pow(oscilacoes.w,2);
    printf ("Temos a expressão  $m=k/w^2$ , obtemos então como resultado a massa
        associada ao objeto estudado. Logo,  $m=%f$  KG.\n",oscilacoes.m);
}
system ("pause");
return 0;
}

```

2.4.2 Exercício 2

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
struct registro{
    float v_final_1,v_final_2,masa_1,masa_2,v_inicial_1;
};
int main ()
{
    struct registro colisoes;
    printf ("Digite a velocidade inicial da primeira partícula (projétil) em m/s: ");
    scanf ("%f",&colisoes.v_inicial_1);
    printf ("Digite a massa da primeira partícula em KG: ");
    scanf ("%f",&colisoes.masa_1);
    printf ("Digite a massa da segunda partícula em KG(alvo): ");
    scanf ("%f",&colisoes.masa_2);
    colisoes.v_final_1=((colisoes.masa_1-colisoes.masa_2)/(colisoes.masa_1+
        colisoes.masa_2))*colisoes.v_inicial_1;
    printf ("Velocidade Final da partícula 1: %.4f m/s.\n\n",colisoes.v_final_1);
    colisoes.v_final_2=(2*colisoes.masa_1/(colisoes.masa_1+colisoes.masa_2))*
        colisoes.v_inicial_1;
    printf ("Velocidade Final da partícula 2: %.4f m/s\n\n",colisoes.v_final_2);
    system ("pause");
    return 0;
}

```

2.4.3 Exercício 3

```

#include <stdio.h>
#include <math.h>
struct registro{ //Estrutura do tipo registro
    float Integral,erro,h,x,a,b,particoes;
    int subdivisoes,i;
};
int main ()

```

```

{
    struct registro trapezio_repetido;
    trapezio_repetido.Integral=0;
    trapezio_repetido.i=0;
    printf ("Digite o valor de Xa: ");
    scanf ("%f",&trapezio_repetido.a);
    printf ("Digite o valor de Xb: ");
    scanf ("%f",&trapezio_repetido.b);
    printf ("Digite o numero de subdivisoões: ");
    scanf ("%d",&trapezio_repetido.subdivisoões);
    trapezio_repetido.particoes=(trapezio_repetido.b-trapezio_repetido.a)/
        trapezio_repetido.subdivisoões;
    trapezio_repetido.x=trapezio_repetido.a;
    while (trapezio_repetido.i<=trapezio_repetido.subdivisoões)
    {
        if (trapezio_repetido.i==0||trapezio_repetido.i==trapezio_repetido.
            subdivisoões)
            trapezio_repetido.Integral+=2*exp(0.5*trapezio_repetido.x);
            //Função considerada
        else
            trapezio_repetido.Integral+=2*2*exp(0.5*trapezio_repetido.x);
            trapezio_repetido.x+=trapezio_repetido.particoes;
            trapezio_repetido.i++;
    }
    trapezio_repetido.Integral*=trapezio_repetido.particoes/2;
    trapezio_repetido.erro=trapezio_repetido.subdivisoões*(pow(trapezio_repetido.
        particoes,3)/12)*0.5*exp(0.5*trapezio_repetido.b); //deve ser inserido aqui
        a derivada segunda;
    printf ("Integral por Trapezio Repetido: %f\n",trapezio_repetido.Integral);
    printf ("Erro Trapezio Repetido: +- %f\n\n",trapezio_repetido.erro);
    return 0;
}

```

2.5 Funções

2.5.1 Exercício 1

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <conio.h>

float eps, p[50], resultado, parada;
int i, k;

float r(float x){
    float a[10],b[10];
    int cont=0;

    printf("Digite o valor dos coeficientes de X\n");
    while(cont<5){
        scanf("%f", &a[cont]);
        cont++;
    }

    b[4]=a[4];
    b[3]=a[3]+(x*(b[4]));

```

```

        b[2]=a[2]+(x*(b[3]));
        b[1]=a[1]+(x*(b[2]));
        b[0]=a[0]+(x*(b[1]));

        return (b[0]);
}

float rderivada(float x){

    float a[10],b[10];
    int cont=0;
    printf("digite o valor dos coeficientes de X para a derivada da função\n");

    while(cont<4){
        scanf("%f\n", a[cont]);
        cont++;
    }

    b[3]=a[3];
    b[2]=a[2]+(x*b[3]);
    b[1]=a[1]+(x*b[2]);
    b[0]=a[0]+(x*b[1]);

    return (b[0]);
}

int main(){
    printf("Metodo de Birge-Vieta\n\nDigite o x0: ");
    scanf("%f", &p[0]);

    printf("\nInsira o Epsion: ");
    scanf("%f", &eps);
    k=1;

    do{

        p[k]=(p[k-1])-(r(p[k-1]))/(rderivada(p[k-1]));
        printf("\nInteracao = %d \nParcial = %f\nR= %f\nR(derivada)= %f\n\n", k
            , p[k], r(p[k-1]), rderivada(p[k-1]));
        resultado = p[k];
        parada=p[k]-p[k-1];

        k++;

    }while(eps<=(fabs(parada)));
    printf("\n\n A solucao eh: %f", resultado);
}

```

2.5.2 Exercício 2

```

#include <stdio.h>
#include <stdlib.h>
#define N 50 //tamanho da função

```

```

void tipos(float constantes_f2[],float expoentes_f2[],float derivada_f2[],int tipo_2[],
int tamanho,int indicador,int i) //subprograma
{
    if (indicador==0) //printa a função f2 (não deriva)
    {
        if (tipo_2[i]==1) //Analisa
            qual é o tipo da função 2 a ser considerada (primeiro if associado
            a uma função polinomial)
            printf ("%.2fX%.2f) + ",constantes_f2[i],expoentes_f2[i]+1);
        else if (tipo_2[i]==2) //Segundo if
            associado a uma função seno declarada no programa principal
            printf ("sen(%.2fX%.2f) + ",constantes_f2[i],expoentes_f2[i]+1);
        else if (tipo_2[i]==3) //Terceiro
            associado a função cosseno
            printf ("cos(%.2fX%.2f) + ",constantes_f2[i],expoentes_f2[i]+1);
        else if (tipo_2[i]==4) //Quarto
            associado a função tangente
            printf ("tan(%.2fX%.2f) + ",constantes_f2[i],expoentes_f2[i]+1);
        else if (tipo_2[i]==5) //Quinto
            associado a função Logaritmo Natural
            printf ("ln(%.2fX%.2f) + ",constantes_f2[i],expoentes_f2[i]+1);
        else if (tipo_2[i]==6) //Sexto e
            último associado à função exponencial
            printf ("exp(%.2fX%.2f) + ",constantes_f2[i],expoentes_f2[i]+1)
            ;
    }
    else //Else das derivadas, printa as derivadas associadas à função 2.
    {
        if (tipo_2[i]==1)
            printf ("(%.2fX%.2f)",derivada_f2[i],expoentes_f2[i]);
        else if (tipo_2[i]==2)
            printf ("(%.2fX%.2f)cos(%.2fX%.2f)",derivada_f2[i],expoentes_f2
            [i],constantes_f2[i],expoentes_f2[i]+1);
        else if (tipo_2[i]==3)
            printf ("-(%.2fX%.2f)sen(%.2fX%.2f)",derivada_f2[i],
            expoentes_f2[i],constantes_f2[i],expoentes_f2[i]+1);
        else if (tipo_2[i]==4)
            printf ("(%.2fX%.2f)(sec(%.2fX%.2f))^2",derivada_f2[i],
            expoentes_f2[i],constantes_f2[i],expoentes_f2[i]+1);
        else if (tipo_2[i]==5)
            printf ("(%.2fX%.2f)/(%.2fX%.2f)",derivada_f2[i],expoentes_f2[i
            ],constantes_f2[i],expoentes_f2[i]+1);
        else if (tipo_2[i]==6)
            printf ("(%.2fX%.2f)exp(%.2fX%.2f)",derivada_f2[i],expoentes_f2
            [i],constantes_f2[i],expoentes_f2[i]+1);
    }
}
int main ()
{
    float constantes_f1[N],expoentes_f1[N],derivada_f1[N],derivada_f2[N],expoentes_f2
    [N],constantes_f2[N]; //constantes_f1-Vetor com N elementos, que guarda a constante
    que multiplica X da primeira função.
    int tamanho,tipo_1[N],i,tipo_2[N],indicador=0;
    //Expoentes_f1-vetor com N elementos que guarda na memória informações a
    respeito do expoente associado à função 1.Derivada_f1-Vetor que contém a
    derivada da função 1
    printf ("Digite o quantos elementos possui a funcao: "); //Tipo_1
    e Tipo_2 - Vetor com N elementos de inteiros no qual guarda qual o 'modelo'
    da função(seno,cosseno,tangente,ln,exp,polinomial,etc).
    scanf ("%d",&tamanho);
    printf ("Digite o numero, de acordo com o tipo de funcao que deseja derivar:\n1-
    Polinomial\n2-seno\n3-cosseno\n4-tangente\n5-logaritmo natural\n6-

```

```

    exponencial natural\n\n");
for (i=0;i<tamanho;i++) //Lê a função a ser derivada e, logo em
seguida, já executa a derivada.
{
    printf ("Tipo de funcao: ");
    scanf ("%d",&tipo_1[i]); //lendo o tipo da função
    1
    printf ("Digite a constante associada ao termo %d da funcao: ",i+1);
    scanf ("%f",&constantes_f1[i]); //Constante associada à função 1
    printf ("Digite o expoente associado a esta constante: ");
    scanf ("%f",&expoentes_f1[i]); //Expoente associado à função 1
    printf ("Tipo de funcao que esta multiplicando a primeira: ");
    scanf ("%d",&tipo_2[i]); //Lendo o tipo da função
    2
    printf ("Digite a constante da funcao que esta multiplicando a funcao
    declarada anteriormente: ");
    scanf ("%f",&constantes_f2[i]); //Constante associada à função 2
    printf ("Digite o expoente associado a esta constante: ");
    scanf ("%f",&expoentes_f2[i]); //Expoente associado à função 2
    derivada_f1[i]=constantes_f1[i]*expoentes_f1[i]; //
    Calculando a derivada da função 1
    derivada_f2[i]=constantes_f2[i]*expoentes_f2[i]; //
    Calculando a derivada da função 2
    expoentes_f1[i]--; //Subtrai 1 do expoente da função
    1
    expoentes_f2[i]--; //Subtrai 1 do expoente da função
    2
}
printf ("Derivada da funcao considerada:\n\n");
for (i=0;i<tamanho;i++) //For responsável por printar a
regra do produto, ou seja: f'(x)*g(x) + f(x)*g'(x), respectivamente, onde f(
x) é a função 1 e g(x) a função 2.
{
    indicador=0;
    if (tipo_1[i]==1)
    {
        printf ("(%.2fX^%.2f) * ",derivada_f1[i],expoentes_f1[i]);
        tipos(constantes_f2,expoentes_f2,derivada_f2,tipo_2,tamanho,
        indicador,i);
        indicador++;
        printf ("(%.2fX^%.2f) * ",constantes_f1[i],expoentes_f1[i]+1);
        tipos(constantes_f2,expoentes_f2,derivada_f2,tipo_2,tamanho,
        indicador,i);
    }
    if (i!=tamanho-1&&tipo_1[i]==1)
        printf (" + ");
    if (tipo_1[i]==2)
    {
        indicador=0;
        printf ("(%.2fX^%.2f)cos(%.2fX^%.2f) * ",constantes_f1[i],
        expoentes_f1[i]+1,derivada_f1[i],expoentes_f1[i]);
        tipos(constantes_f2,expoentes_f2,derivada_f2,tipo_2,tamanho,
        indicador,i);
        indicador++;
        printf ("sen(%.2fX^%.2f) * ",constantes_f1[i],expoentes_f1[i]+1);
        tipos(constantes_f2,expoentes_f2,derivada_f2,tipo_2,tamanho,
        indicador,i);
    }
    if (i!=tamanho-1&&tipo_1[i]==2)
        printf (" + ");
    if (tipo_1[i]==3)
    {
        indicador=0;
        printf ("-(%.2fX^%.2f)sen(%.2fX^%.2f) * ",constantes_f1[i],
        expoentes_f1[i]+1,derivada_f1[i],expoentes_f1[i]);
        tipos(constantes_f2,expoentes_f2,derivada_f2,tipo_2,tamanho,

```



```

        indicador,i);
        indicador++;
        printf ("cos(%.2fX^%.2f) * ", constantes_f1[i], expoentes_f1[i]+1);
        tipos(constantes_f2, expoentes_f2, derivada_f2, tipo_2, tamanho,
            indicador,i);
    }
    if (i!=tamanho-1&&tipo_1[i]==3)
        printf (" + ");
    if (tipo_1[i]==4)
    {
        indicador=0;
        printf ("(%.2fX^%.2f)(sec(%.2fX^%.2f))^2 * ", constantes_f1[i],
            expoentes_f1[i]+1, derivada_f1[i], expoentes_f1[i]);
        tipos(constantes_f2, expoentes_f2, derivada_f2, tipo_2, tamanho,
            indicador,i);
        indicador++;
        printf ("tan(%.2fX^%.2f) * ", constantes_f1[i], expoentes_f1[i]+1);
        tipos(constantes_f2, expoentes_f2, derivada_f2, tipo_2, tamanho,
            indicador,i);
    }
    if (i!=tamanho-1&&tipo_1[i]==4)
        printf (" + ");
    if (tipo_1[i]==5)
    {
        indicador=0;
        printf ("(%.2fX^%.2f)/(%.2fX^%.2f) * ", derivada_f1[i],
            expoentes_f1[i], constantes_f1[i], expoentes_f1[i]+1);
        tipos(constantes_f2, expoentes_f2, derivada_f2, tipo_2, tamanho,
            indicador,i);
        indicador++;
        printf ("ln(%.2fX^%.2f) * ", constantes_f1[i], expoentes_f1[i]+1);
        tipos(constantes_f2, expoentes_f2, derivada_f2, tipo_2, tamanho,
            indicador,i);
    }
    if (i!=tamanho-1&&tipo_1[i]==5)
        printf (" + ");
    if (tipo_1[i]==6)
    {
        indicador=0;
        printf ("(%.2fX^%.2f)exp^(%.2fX^%.2f) * ", derivada_f1[i],
            expoentes_f1[i], constantes_f1[i], expoentes_f1[i]+1);
        tipos(constantes_f2, expoentes_f2, derivada_f2, tipo_2, tamanho,
            indicador,i);
        indicador++;
        printf ("exp(%.2fX^%.2f) * ", constantes_f1[i], expoentes_f1[i]+1);
        tipos(constantes_f2, expoentes_f2, derivada_f2, tipo_2, tamanho,
            indicador,i);
    }
    if (i!=tamanho-1&&tipo_1[i]==6)
        printf (" + ");
}
return 0;
}

```

2.5.3 Exercício 3

```

#include <stdio.h>
#include <stdlib.h>
#define MAX 10
void leitor_matriz(float matriz[][MAX],int linhas,int colunas)
{
    int i,j;
    for (i=0;i<linhas;i++)
    {
        for (j=0;j<colunas;j++)

```

```
        {           printf ("Digite o termo associado a linha %d e coluna %d da
                      matriz: ",i+1,j+1);
                      scanf ("%f",&matriz[i][j]);
        }
    }
}
void operacao(float a[][MAX],float b[][MAX],float c[][MAX],int linhas,int colunas,int
indicador)
{
    int i,j;
    for (i=0;i<linhas;i++)
    {
        for (j=0;j<colunas;j++)
        {
            if (indicador==1)
                c[i][j]=a[i][j]+b[i][j];
            else
                c[i][j]=a[i][j]-b[i][j];
        }
    }
}
void printa_matriz(float matriz[][MAX],int linhas,int colunas)
{
    int i,j;
    for (i=0;i<linhas;i++)
    {
        for (j=0;j<colunas;j++)
        {
            printf ("%10.2f",matriz[i][j]);
            if (j==colunas-1)
                printf ("\n");
        }
    }
}
int main ()
{
    float a[MAX][MAX],b[MAX][MAX],c[MAX][MAX];
    int tam_linhas_a,tam_colunas_a,tam_linhas_b,tam_colunas_b,indicador;
    do
    {
        printf ("Digite o numero de linhas e colunas associado a matriz A,
                respectivamente:\n");
        scanf ("%d%d",&tam_linhas_a,&tam_colunas_a);
        printf ("Digite o numero de linhas e colunas associado a matriz B,
                respectivamente:\n");
        scanf ("%d%d",&tam_linhas_b,&tam_colunas_b);
        if (tam_linhas_a!=tam_linhas_b||tam_colunas_b!=tam_colunas_a)
            printf ("Dimensao invalida! Digite duas matrizes com as mesmas dimensoes
                    para realizar o calculo.\n");
    }
    while (tam_linhas_a!=tam_linhas_b||tam_colunas_b!=tam_colunas_a);
    printf ("Digite a matriz A:\n\n");
    leitor_matriz(a,tam_linhas_a,tam_colunas_a);
    printf ("Digite a matriz B:\n\n");
    leitor_matriz(b,tam_linhas_b,tam_colunas_b);
    printf ("\nDigite 1 para realizar a soma entre matrizes e 2 para realizar
            subtracao: ");
    scanf ("%d",&indicador);
    operacao(a,b,c,tam_linhas_b,tam_colunas_b,indicador);
    printf ("\nMatriz A digitada pelo usuario:\n\n");
    printa_matriz(a,tam_linhas_a,tam_colunas_a);
    printf ("\nMatriz B digitada pelo usuario:\n\n");
    printa_matriz(b,tam_linhas_b,tam_colunas_b);
    printf ("\nMatriz C:\n\n");
    printa_matriz(c,tam_linhas_a,tam_colunas_a);
    return 0;
}
```

2.5.4 Exercício 4

```

#include <stdio.h>
#include <stdlib.h>
#define MAX 10
void integral(float constantes[],float expoentes[],float constantes_integral[],float
expoentes_integral[],int tamanho)
{
    int i;
    for (i=0;i<tamanho;i++)
    {
        constantes_integral[i]=constantes[i]/expoentes[i];
        expoentes_integral[i]=expoentes[i]+1;
    }
}
int main ()
{
    float constantes[MAX],expoentes[MAX],constantes_integral[MAX],expoentes_integral[
MAX];
    int i, tamanho;
    printf ("Digite o tamanho da funcao (max 10 termos): ");
    scanf ("%d",&tamanho);
    printf ("Digite a funcao logo abaixo:\n\n");
    for (i=0;i<tamanho;i++)
    {
        printf ("Digite a constante %d associada ao x: ",i+1);
        scanf ("%f",&constantes[i]);
        printf ("Digite o expoente %d associado ao x: ",i+1);
        scanf ("%f",&expoentes[i]);
        printf ("\n");
    }
    integral(constantes,expoentes,constantes_integral,expoentes_integral,tamanho);
    printf ("Funcao resultante:\n\n");
    for (i=0;i<tamanho;i++)
    {
        if (expoentes[i]!=0)
            printf ("(%.2fX^%.2f)",constantes_integral[i],expoentes_integral[
i]);
        if (i<tamanho-1)
            printf (" + ");
    }
    return 0;
}

```

2.5.5 Exercício 5

```

#include <stdio.h>
float inc(float y0, float x0, float y1, float x1) //Função inclinação
{
    float inclinacao;
    inclinacao=(y1-y0)/(x1-x0);
    return inclinacao;
}
int main()
{
    float y0,x0,y1,x1,inclinacao;
    printf ("Digite Y0: ");
    scanf ("%f",&y0);
    printf ("Digite X0: ");
    scanf ("%f",&x0);
    printf ("Digite Y1: ");
    scanf ("%f",&y1);
    printf ("Digite X1: ");
    scanf ("%f",&x1);
    inclinacao=inc(y0,x0,y1,x1);
    printf ("\nInclinacao da reta: %.3f\n",inclinacao);
}

```

```

        return 0;
}

```

2.5.6 Exercício 6

```

#include <stdio.h>
#include <math.h>
float derivada_segunda(float x)
{
    float y;
    y=0.5*exp(0.5*x); //Derivar duas vezes a função considerada
    return y;
}
float f(float x)
{
    float y;
    y=2*exp(0.5*x); //Alterar função
    return y;
}
int main ()
{
    float trapezio=0,particao,erro,a,b;
    printf ("Digite o valor de Xa: ");
    scanf ("%f",&a);
    printf ("Digite o valor de Xb: ");
    scanf ("%f",&b);
    particao=b-a;
    trapezio=(particao/2)*(f(a)+f(b));
    printf ("\nIntegral por Trapezio: %f\n",trapezio);
    erro=(pow(particao,3)/3)*derivada_segunda(b);
    printf ("Erro Trapezio: +- %f\n\n",erro);
    return 0;
}

```

2.5.7 Exercício 7

```

#include <stdio.h>
#include <math.h>
float derivada_quarta(float x)
{
    float y;
    y=0.125*exp(0.5*x); //Derivar a função 4 vezes
    return y;
}
float f(float x)
{
    float y;
    y=2*exp(0.5*x); //Alterar função
    return y;
}
int main ()
{
    float a,b,particoes,I1_3simpson,erro;
    printf ("Digite o valor de Xa: ");
    scanf ("%f",&a);
    printf ("Digite o valor de Xb: ");
    scanf ("%f",&b);
    particoes=(b-a)/2;
    I1_3simpson=(particoes/3)*(f(a)+4*f(a+particoes)+f(b));
    erro=(pow(particoes,5)/90)*derivada_quarta(b);
    printf ("Integral por 1/3 de Simpson: %f\nErro 1/3 Simpson: +- %f\n\n",
        I1_3simpson,erro);
    return 0;
}

```

2.6 Ponteiros e Alocação Dinâmica

2.6.1 Exercício 1

```
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>

void uniao (){
    float *v1;
    float *v2;
    float *vetuniao;
    int cont2=0,flag=0,cont3=0,cont4=0,n,n2,n3=0;
    printf("Digite o tamanho do vetor 1 ");
    scanf("%d", &n);
    v1=(float*)malloc(sizeof(int)*n);
    printf("\nDigite o tamanho do vetor 2 ");
    scanf("%d", &n2);
    v2=(float*)malloc(sizeof(int)*n2);
    cont2=0;

    while(cont2<n){
        printf("\nDigite os valores do vetor 1 ");
        scanf("%f", &v1[cont2]);
        cont2++;
    }
    cont2=0;
    while(cont2<n2){
        printf("Digite os valores do vetor 2 ");
        scanf("%f", &v2[cont2]);
        cont2++;
    }
    cont3=0;
    cont4=0;
    while(cont3<n){
        while(cont4<n2){
            if(v1[cont3]==v2[cont4]){
                n3--;
            }
            cont4++;
        }
        cont3++;
        cont4=0;
    }
    n3=n+n2;
    printf("%i", n3);
    vetuniao=(float*)malloc(sizeof(int)*n3);
    cont4=0;
    while(cont4<n){
        vetuniao[cont4]=v1[cont4];
        cont4++;
    }
    cont2=0;
    cont3=0;
    while(cont2<n2){
        cont3=0;
        while(cont3<n){
            if(v2[cont2]==v1[cont3]){
```

```
                flag=1;
            }
            cont3++;
        }
        if(flag==0){
            vetuniao[cont4]=v2[cont2];
            cont4++;
        }
        cont2++;
        flag=0;
    }
    cont2=0;
    printf("\nA uniao dos dois vetores e");
    while(cont2<n3){

        printf("\n%f", vetuniao[cont2]);
        cont2++;
    }
}
void interseccao(){

    float *v1;
    float *v2;
    float *vetinterseccao;
    int cont2=0, flag=0, cont3=0, cont4=0, n, n2, n3;
    printf("Digite o tamanho do vetor 1 ");
    scanf("%d", &n);
    v1=(float*)malloc(sizeof(int)*n);
    printf("\nDigite o tamanho do vetor 2 ");
    scanf("%d", &n2);
    v2=(float*)malloc(sizeof(int)*n2);
    cont2=0;

    while(cont2<n){
        printf("\nDigite os valores do vetor 1 ");
        scanf("%f", &v1[cont2]);
        cont2++;
    }

    cont2=0;
    while(cont2<n2){
        printf("\nDigite os valores do vetor 2 ");
        scanf("%f", &v2[cont2]);
        cont2++;
    }

    n3=n+n2;
    vetinterseccao=(float*)malloc(sizeof(int)*n3);
    cont2=0;
    cont3=0;
    cont4=0;
    while(cont2<n){
        while(cont3<n2){
            if(v1[cont2]==v2[cont3]){
                vetinterseccao[cont4]=v1[cont2];
                cont4++;
                cont3=n2;
            }
            cont3++;
        }
        cont2++;
        cont3=0;
    }
}
```

```

    }
    cont2=0;
    printf("\nA interseccao dos dois vetores eh\n ");
    while(cont2<cont4){
        printf(" %f", vetinterseccao[cont2]);
        cont2++;
    }

}

int main(){
int w,cont=0;

    while (cont!=1){
        printf("\nDigite 1 para fazer a uniao de vetores, 2 para fazer
            interseccao ou 3 para sair");
        scanf("%d", &w);
        switch (w){
            case 1:
                uniao();
                break;
            case 2:
                interseccao();
                break;
            case 3:
                cont=1;
        }
    }

}

```

2.6.2 Exercício 2

```

#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
#include <math.h>
int main ()
{
    float *f,*ang,resultante_x=0,resultante_y=0;
    int tamanho,i;
    setlocale(LC_ALL,"portuguese");
    printf ("Digite o número de forças associado ao sistema: ");
    scanf ("%d",&tamanho);
    f=(float *)calloc(tamanho,sizeof(float)); //Alocação utilizando a função
        calloc
    ang=(float *)malloc(tamanho*sizeof(float)); //Alocação utilizando a função
        malloc, ambas desempenham a mesma função, mudando apenas a sintaxe.
    for (i=0;i<tamanho;i++)
    {
        printf ("Digite F%d: ",i+1);
        scanf ("%f",&f[i]);
        printf ("Digite o ângulo de F%d em relação ao eixo x (em radianos): ",i
            +1);
        scanf ("%f",&ang[i]);
        resultante_x+=cos(ang[i])*f[i];
        resultante_y+=sin(ang[i])*f[i];
    }
    printf ("Força resultante associada ao eixo x (Fx):%f\n",resultante_x);
}

```

```

printf ("Força resultante associada ao eixo y (Fy):%f\n",resultante_y);
free(f);
free(ang);
return 0;
}

```

2.6.3 Exercício 3

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main ()
{
    float **coeficientes,*independentes,multiplicador,*resultados=0,somatorio;
    int i,j,contador=0,varia_matriz=0,k,ind_matriz=0,tamanho;
    printf ("Digite o tamanho da matriz: ");
    scanf ("%d",&tamanho);
    coeficientes=(float **)calloc(tamanho,sizeof(float *));
    for (i=0;i<tamanho;i++)
        coeficientes[i]=(float *)calloc(tamanho,sizeof(float));
    independentes=(float *)calloc(tamanho,sizeof(float));
    resultados=(float *)calloc(tamanho,sizeof(float));
    printf ("Digite os valores que estao multiplicando as incognitas:\n");
    for (i=0;i<tamanho;i++)
    {
        for (j=0;j<tamanho;j++)
        {
            printf ("\nDigite o valor associado a linha %d e coluna %d: ",i
                +1,j+1);
            scanf ("%f",&coeficientes[i][j]);
        }
        printf ("Digite o termo independente associado a linha %d: ",i+1);
        scanf ("%f",&independentes[i]);
    }
    for (k=0;k<tamanho-1;k++)
    {
        contador=0;
        for (i=varia_matriz;i<tamanho;i++)
        {
            if (i!=0&&i>varia_matriz)
            {
                multiplicador=coeficientes[i][varia_matriz]/coeficientes[
                    varia_matriz][varia_matriz];
                independentes[i]-=independentes[i-contador]*multiplicador
                    ;
            }
            for (j=0;j<tamanho;j++)
            {
                if (i!=0&&i>varia_matriz)
                    coeficientes[i][j]-=coeficientes[i-contador][j]*
                        multiplicador;
            }
            contador++;
        }
        varia_matriz++;
    }
    printf ("\n");
    for (i=tamanho-1;i>=0;i--)
    {
        somatorio=0;
        for (j=0;j<tamanho;j++)
        {
            if (i!=j)
                somatorio+=coeficientes[i][j]*resultados[j];
        }
        resultados[i]=(independentes[i]-somatorio)/coeficientes[i][i];
    }
    for (i=0;i<tamanho;i++)
    {
        printf ("Valor de X%d: %f\n",i+1,resultados[i]);
    }
}

```



```

    }
    system ("pause");
    return 0;
}

```

2.6.4 Exercício 4

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main ()
{
    float **coeficientes,*independentes,multiplicador,*resultados=0,somatorio;
    int i,j,contador=0,varia_matriz=0,k,ind_matriz=0,tamanho;
    printf ("Digite o tamanho da matriz: ");
    scanf ("%d",&tamanho);
    coeficientes=(float **)calloc(tamanho,sizeof(float *));
    for (i=0;i<tamanho;i++)
        coeficientes[i]=(float *)calloc(tamanho,sizeof(float));
    independentes=(float *)calloc(tamanho,sizeof(float));
    resultados=(float *)calloc(tamanho,sizeof(float));
    printf ("Digite os valores que estao multiplicando as incognitas: ");
    for (i=0;i<tamanho;i++)
    {
        for (j=0;j<tamanho;j++)
        {
            printf ("\nDigite o valor associado a linha %d e coluna %d: ",i
                +1,j+1);
            scanf ("%f",&coeficientes[i][j]);
        }
    }
    independentes[0]=1;
    do
    {
        for (k=0;k<tamanho-1;k++)
        {
            contador=0;
            for (i=varia_matriz;i<tamanho;i++)
            {
                if (i!=0&&i>varia_matriz)
                {
                    multiplicador=coeficientes[i][varia_matriz]/coeficientes[
                        varia_matriz][varia_matriz];
                    independentes[i]-=independentes[i-contador]*multiplicador
                        ;
                }
                for (j=0;j<tamanho;j++)
                {
                    if (i!=0&&i>varia_matriz)
                        coeficientes[i][j]-=coeficientes[i-contador][j]*
                            multiplicador;
                }
                contador++;
            }
            varia_matriz++;
        }
        for (i=tamanho-1;i>=0;i--)
        {
            somatorio=0;
            for (j=0;j<tamanho;j++)
            {
                if (i!=j)
                    somatorio+=coeficientes[i][j]*resultados[j];
            }
            resultados[i]=(independentes[i]-somatorio)/coeficientes[i][i];
        }
        printf ("\nColuna %d da matriz inversa:\n\n",ind_matriz+1);
        for (i=0;i<tamanho;i++)
            printf ("Valor de X%d: %f\n",i+1,resultados[i]);
    }
}

```

```
    ind_matriz++;  
    for (i=0;i<tamanho;i++)  
        independentes[i]=0;  
    independentes[ind_matriz]=1;  
}  
while (ind_matriz<tamanho);  
printf ("\n");  
system ("pause");  
return 0;  
}
```

Referências