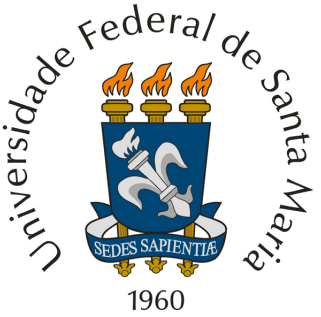


Universidade Federal de Santa Maria
Campus Cachoeira do Sul

Caderno de Exercícios

Algoritmos Aplicados nas Engenharias

Prof. Dr. Vinícius Maran
Profª. Drª. Silvana Maldaner
Alisson Nascimento
Andrei Cunha Lima
Henrique Jahnke Hoch
Wantuil Rodrigues

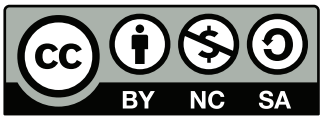
Este caderno de exercícios tem como objetivo apresentar questões relacionadas à algoritmos e programação. As questões apresentadas neste material estão diretamente relacionadas à outras disciplinas de cursos de Engenharia, tais como: Física, Cálculo, Estatística, entre outras.

Este material é resultado da segunda 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*. O projeto é desenvolvido no LUMAC (*Laboratory of Ubiquitous, Mobile and Applied Computing*) e teve apoio do Fundo Institucional de Apoio ao Ensino (FIEN-UFSM).

As respostas para os algoritmos podem variar. Os exercícios foram resolvidos na linguagem de programação C. Alguns exercícios foram baseados em teorias apresentadas em livros de outras disciplinas: [1, 2]

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

Deixe suas sugestões aqui, nos ajude a melhorar cada vez mais.



Documento criado em LaTeX

Conteúdo

1 Exercícios **5**
1.1 Desvios Condicionais . . . 5
1.1.1 Exercício 1 - Variável Complexa . . . 5
1.1.2 Exercício 2 - Fisica 1 . . . 5
1.1.3 Exercício 3 - Cálculo A . . . 5
1.1.4 Exercício 4 - Estatística . . . 6
1.1.5 Exercício 5 - Estatística . . . 6
1.1.6 Exercício 6 - Estatística . . . 6
1.1.7 Exercício 7 - Física . . . 7
1.2 Laços de Repetição . . . 7
1.2.1 Exercício 1 - Fisica III . . . 7
1.2.2 Exercício 2 - Circuitos Digitais I . . . 7
1.2.3 Exercício 3 - Física III . . . 8
1.2.4 Exercício 4 - Física III . . . 8
1.2.5 Exercício 5 - Cálculo A . . . 8
1.2.6 Exercício 6 - Cálculo A . . . 9
1.2.7 Exercício 7 - Cálculo A . . . 9
1.2.8 Exercício 8 - Álgebra Linear . . . 9
1.2.9 Exercício 9 - Métodos Numéricos e Computacionais . . . 9
1.3 Vetores e Matrizes . . . 10
1.3.1 Exercício 1 - Física III . . . 10
1.3.2 Exercício 2 - Estatística . . . 10
1.3.3 Exercício 3 - Álgebra Linear . . . 10
1.3.4 Exercício 4 - Álgebra Linear . . . 10
1.3.5 Exercício 5 - Álgebra Linear . . . 11
1.3.6 Exercício 6 - Estatística . . . 11
1.3.7 Exercício 7 - Métodos Numéricos e Computacionais . . . 11
1.3.8 Exercício 8 - Métodos Numéricos e Computacionais . . . 11
1.3.9 Exercício 9 - Métodos Numéricos e Computacionais . . . 12
1.3.10 Exercício 10 - Métodos Numéricos e Computacionais . . . 12
1.3.11 Exercício 11 - Métodos Numéricos e Computacionais . . . 12
1.3.12 Exercício 12 - Estatística . . . 12
1.3.13 Exercício 13 - Álgebra Linear . . . 12
1.3.14 Exercício 14 - Álgebra Linear . . . 13
1.3.15 Exercício 15 - Álgebra Linear . . . 13
1.4 Estruturas de Dados Heterogêneas . . . 13
1.4.1 Exercício 1 - Física . . . 13
1.4.2 Exercício 2 - Física . . . 14
1.4.3 Exercício 3 - Métodos Numéricos e Computacionais . . . 14

1.5 Funções . . . 14
1.5.1 Exercício 1 - Métodos Numéricos e Computacionais . . . 14
1.5.2 Exercício 2 - Cálculo A . . . 14
1.5.3 Exercício 3 - Álgebra Linear . . . 15
1.5.4 Exercício 4 - Cálculo A . . . 15
1.5.5 Exercício 5 - Cálculo A . . . 16
1.5.6 Exercício 6 - Métodos Numéricos e Computacionais . . . 16
1.5.7 Exercício 7 - Métodos Numéricos e Computacionais . . . 16
1.6 Ponteiros e Alocação Dinâmica . . . 17
1.6.1 Exercício 1 - Física III . . . 17
1.6.2 Exercício 2 - Física III . . . 17
1.6.3 Exercício 3 - Estatística . . . 17
1.6.4 Exercício 4 - Física . . . 17
1.6.5 Exercício 5 - Álgebra Linear . . . 18
1.6.6 Exercício 6 - Álgebra Linear . . . 18
1.6.7 Exercício 7 - Física III . . . 18
1.6.8 Exercício 8 - Física III . . . 18

2 Algoritmos Resolvidos 19
2.1 Desvios Condicionais . . . 19
2.1.1 Exercício 1 . . . 19
2.1.2 Exercício 2 . . . 20
2.1.3 Exercício 3 . . . 21
2.1.4 Exercício 4 . . . 22
2.1.5 Exercício 5 . . . 23
2.1.6 Exercício 6 . . . 24
2.1.7 Exercício 7 . . . 24
2.2 Laços de Repetição . . . 24
2.2.1 Exercício 1 . . . 24
2.2.2 Exercício 2 . . . 25
2.2.3 Exercício 3 . . . 27
2.2.4 Exercício 4 . . . 27
2.2.5 Exercício 5 . . . 27
2.2.6 Exercício 6 . . . 27
2.2.7 Exercício 7 . . . 28
2.2.8 Exercício 8 . . . 28
2.2.9 Exercício 9 . . . 29
2.3 Vetores e Matrizes . . . 30
2.3.1 Exercício 1 . . . 30
2.3.2 Exercício 2 . . . 30
2.3.3 Exercício 3 . . . 32

2.3.4 Exercício 4 . . . 32
2.3.5 Exercício 5 . . . 33
2.3.6 Exercício 6 . . . 34
2.3.7 Exercício 7 . . . 36
2.3.8 Exercício 8 . . . 37
2.3.9 Exercício 9 . . . 38
2.3.10 Exercício 10 . . . 38
2.3.11 Exercício 11 . . . 39
2.3.12 Exercício 12 . . . 40
2.3.13 Exercício 13 . . . 41
2.3.14 Exercício 14 . . . 41
2.3.15 Exercício 15 . . . 42
2.4 Estruturas de Dados Heterogêneas . . . 42
2.4.1 Exercício 1 . . . 42
2.4.2 Exercício 2 . . . 45
2.4.3 Exercício 3 . . . 45
2.5 Funções . . . 46
2.5.1 Exercício 1 . . . 46
2.5.2 Exercício 2 . . . 47
2.5.3 Exercício 3 . . . 50
2.5.4 Exercício 4 . . . 52
2.5.5 Exercício 5 . . . 52
2.5.6 Exercício 6 . . . 53
2.5.7 Exercício 7 . . . 53
2.6 Ponteiros e Alocação Dinâmica . . . 54
2.6.1 Exercício 1 . . . 54
2.6.2 Exercício 2 . . . 54
2.6.3 Exercício 3 . . . 55
2.6.4 Exercício 4 . . . 57
2.6.5 Exercício 5 . . . 58
2.6.6 Exercício 6 . . . 59
2.6.7 Exercício 7 . . . 59
2.6.8 Exercício 8 . . . 60

1 Exercícios

1.1 Desvios Condicionais

1.1.1 Exercício 1 - Variável Complexa

Elabore um algoritmo que faça a conversão de um número complexo de coordenadas cartesianas para polares, ou de coordenadas polares para cartesianas. O usuário deve informar com 0 ou 1 a opção de conversão que deseja realizar. Caso o usuário escolha converter de cartesianas para polares, o mesmo deve fornecer a parte real do número complexo (x), e a parte imaginária (iy). O resultado deve ser fornecido em função do ângulo e do módulo em coordenadas polares. Caso o usuário escolha converter para cartesianas, realize o inverso do que foi descrito logo acima.

1.1.2 Exercício 2 - Fisica 1

Crie um programa que faça a conversão de rumo para azimute e vice-versa. O usuário deve selecionar a opção desejada em um menu e, caso opte pela conversão rumo azimute, deverá informar a orientação do rumo. Os valores dos ângulos deverão ser informados e calculados na forma decimal (Ex: 90,89° ao invés de 90°53'24'').

1.1.3 Exercício 3 - 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:

1ª - $Seno\ \theta = cateto\_oposto/hipotenusa$;
2ª - $Cosseno\ \theta = cateto\_adjacente/hipotenusa$;
3ª - $Tangente\ \theta = cateto\_oposto/cateto\_adjacente$;

1.1.4 Exercício 4 - 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.

1.1.5 Exercício 5 - 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.

1.1.6 Exercício 6 - 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 $H0$ ou $H1$. Apresente na tela qual foi a alternativa aceita, e qual foi rejeitada.

1.1.7 Exercício 7 - 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 - Fisica III

Desenvolva um algoritmo que calcule a capacitância necessária em um capacitor para armazenar uma certa quantidade de energia. O programa deve ler a energia em Wh e a tensão em $Volts$. No fim da execução do algoritmo a capacitância deve ser apresentada em $Farad$, deve-se usar potência de base 3, onde:

$1Wh = 3600J$

$U = (1/2)CV^2$, onde:

U = energia (em *Joules*),

C = Capacitância (em *Farads*) e

V = diferenca de potências (em *Volts*).

Exemplo: O usuário informa as seguintes entradas 0.05Wh e 220V. A saída do programa deve ser: $50 * 10^{-3}$.

1.2.2 Exercício 2 - Circuitos Digitais I

Faça um algoritmo que converta um numero decimal inteiro para: Binario , hexadecimal e BCD

Exemplo: Caso o usuario digite 87523 ele deve ter as seguintes respostas:
Binário: 10101010111100011
Hexadecimal: 155F3
BCD: 10000111010100100011

1.2.3 Exercício 3 - Física III

Suponha que você deseja medir a corrente elétrica que percorre um fio de cobre, porém que você apenas conheça da função distribuição densidade de corrente e as característica geométricas do fio. A densidade de corrente é dada por $J = a * r^2$, onde $a = 3 * 10^{11} A/m^4$. Considere que o fio conduza de $R/2$ até R, onde $R = 2mm$. A geometria do fio é cilíndrica. Utilize os conceitos de integral, e utilize a fórmula

$$\int_{R/2}^{R} J * DA$$

para calcular a corrente no condutor. Lembre-se que o significado físico da integral é o somatório de valores infinitesimais dentro de um intervalo de integração. Este exercício foi baseado no exercício resolvido contido no livro fundamentos de física, volume 3, página 138.

1.2.4 Exercício 4 - Física III

Elabore um programa que calcule a força total que uma particula está submetida devido a existência de 20 particulas ao seu redor, cada uma exercendo uma determinada força, de acordo com sua carga e distância.

O programa deve fornecer o resultado em relação aos eixos x e y, além de fornecer o módulo da força. O usuario deve fornecer também o ângulo em relação ao semi-eixo positivo x. Utilize a lei de *Coulomb* para calcular a força resultante.

1.2.5 Exercício 5 - Cálculo A

Elabore um algoritmo 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, a mesma deve ser codificada 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.

1.2.6 Exercício 6 - Cálculo A

Elabore um programa que calcule o fatorial associado à 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: $5! = 5 * 4 * 3 * 2 * 1$.

1.2.7 Exercício 7 - 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.*

1.2.8 Exercício 8 - Álgebra Linear

Crie um programa para calcular uma raíz 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 raíz 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.

1.2.9 Exercício 9 - Métodos Numéricos e Computacionais

Escreva um algoritmo que encontre o zero de $f(x) = e^x - 4x^2$ através do metodo 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 somento modificar a expressão implementada no código fonte).

1.3 Vetores e Matrizes

1.3.1 Exercício 1 - Física III

Desenvolva um algoritmo que calcule a resistência equivalente da associação de 5 resistores em paralelo, faça esse algoritmo de forma que seja fácil modificar a quantia de resistores que deseja associar.

1.3.2 Exercício 2 - Estatística

Crie um algoritmo que mostre ao usuário uma tabela de frequência. A tabela deverá possui intervalo de classes, frequência simples, frequência acumulada, frequência relativa simples, frequência relativa acumulada e o ponto médio do intervalo de classes (podendo estender colocando as medidas de tendências centrais). O usuário deve informar a quantidade de dados a ser inserido no programa. A média deverá ser informada ao usuário antes do programa ser encerrado. O número de classes e a amplitude de classes deve ser arredondado para o inteiro superior imediato. Entende-se que todos os dados serão do tipo inteiro.

1.3.3 Exercício 3 - Álgebra Linear

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

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

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

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 ao usuário os 3 vetores.

1.3.4 Exercício 4 - Álgebra Linear

Crie um código que calcule o determinante associado a uma matriz 3x3. O usuário deve informar os valores da matriz. 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* para calcular o determinante.

1.3.5 Exercício 5 - 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.

1.3.6 Exercício 6 - 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 reais para desvio médio, variância e desvio padrão. O usuário deve inserir os 8 intervalos e a frequência de cada intervalo.

1.3.7 Exercício 7 - 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 ($x0$), os coeficientes ($a1, a2, a3, a4$) e o coeficiente linear (B).

1.3.8 Exercício 8 - Métodos Numéricos e Computacionais

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

O usuário deve informar a possível solução ($x0$), os coeficientes ($a1, a2, a3, a4$) e o coeficiente linear (B).

1.3.9 Exercício 9 - Métodos Numéricos e Computacionais

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

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

O usuário deve informar o erro e o valor inicial $x0$.

1.3.10 Exercício 10 - Métodos Numéricos e Computacionais

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

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

O usuário deve informar a precisão e o valor inicial $x0$.

1.3.11 Exercício 11 - Métodos Numéricos e Computacionais

Implemente um algoritmo que usando o método da posição falsa encontre o zero da função $f(x) = e^x - 4x^2$. O usuário deve informar o erro e o intervalo.

1.3.12 Exercício 12 - Estatística

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

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

1.3.13 Exercício 13 - Álgebra Linear

Crie um código que calcule a matriz transposta, dada uma matriz 2X4, utilizando os conceitos de matriz. Crie uma matriz 4x2 para armazenar os resultados obtidos. Ambas as matrizes devem ser apresentadas na tela.

1.3.14 Exercício 14 - Álgebra Linear

Elabore um programa que calcule a norma de um vetor com 3 dimensões, informado pelo usuário, e posteriormente calcule o vetor uniário, dividindo todos os termos do vetor informado pela sua norma.

1.3.15 Exercício 15 - Álgebra Linear

Escreva um código que leia números informados pelo usuário e realize a soma ou a subtração dos mesmos. O usuário deverá escrever o número com o sinal negativo na frente para realizar a subtração. Para realizar a soma, o usuário deverá entrar com o número sem sinal, ou com o sinal positivo na frente do número considerado. Essa operação deverá ser realizada até o usuário informar que deseja encerrar o programa. O somatório deve ser informado após o usuário informar o último número ao programa.

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 perído 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.

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.

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 informada 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 que não retorna dados 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 10x10. O usuário deve digitar os valores das 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 integrados, armazenando dentro de outros dois vetores.

Utilize uma função do tipo *double* 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 $y0, x0, y1, x1$ (reta linear) e o programa calcule o coeficiente de inclinação desta reta.

Execute os cálculos dentro de uma função, que deve retornar 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 - Física III

Desnvolva um algoritmo que calcule a resistência equivalente de N resistores associados em série

Exemplo: O usuário deseja calcular a resistência equivalente da associação de 27 resistores em série, ele informara ao programa o número 27, e digitará a resistência de cada um dos resistores.

1.6.2 Exercício 2 - Física III

Elabore um programa que calcule a indutância equivalente de um circuito elétrico. O programa deve considerar o número de indutores informados pelo usuário. Assim, deve-se alocar dinamicamente um vetor de indutâncias, que armazenará os valores de indutâncias. Deve-se estabelecer dentro do código também, um desvio condicional, onde o usuário deve informar se a indutância informada está em série ou em paralelo com as demais indutâncias informadas anteriormente.

1.6.3 Exercício 3 - 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 usuario deseje sair do sistema.

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

1.6.4 Exercício 4 - Física

Elabore um programa que decomponha um número de forças, fornecido pelo usuário, em relação ao eixo x e ao eixo y. Devem ser alocadas dinamicamente duas variáveis, uma deles é o módulo da força f, e a outra, a variável que irá armazenar 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.

1.6.5 Exercício 5 - Álgebra Linear

Escreva um programa que 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.

1.6.6 Exercício 6 - 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.

1.6.7 Exercício 7 - Física III

Elabore um programa que calcule a resistência equivalente de um circuito elétrico. O programa deve considerar o número de resistores informados pelo usuário, portanto, deve-se alocar dinamicamente um vetor de resistências, que armazenará os valores de resistências. Deve-se estabelecer dentro do código também, um desvio condicional, onde o usuário deve informar se a resistência informada está em série ou em paralelo com as demais resistências informadas anteriormente.

1.6.8 Exercício 8 - Física III

Elabore um progroma que calcule a capacitância equivalente de um circuito elétrico. O programa deve considerar o número de capacitores informados pelo usuário, portanto, deve-se alocar dinamicamente um vetor de capacitâncias, que armazenará os valores de capacitâncias. Deve-se estabelecer dentro do código também, um desvio

condicional, onde o usuário deve informar se a capacitância informada está em série ou em paralelo com as demais capacitâncias informadas anteriormente.

2 Algoritmos Resolvidos

2.1 Desvios Condicionais

2.1.1 Exercício 1

```
//O programa faz conversao entre valores complexos em coordenas cartesianas para polares,
     e vice-versa.
#include <stdio.h>
#include <math.h>
int main ()
{       float real, complexo, angulo, modulo;
        bool operacao;
        printf ("Digite o tipo de operacao desejada, 0 para converter de coordenadas
            cartesianas para polares, e 1 de polares para cartesianas: ");
        scanf ("%d",&operacao);
        if (operacao == 0)
        {       printf ("\nInforme o valor da parte real: ");
                scanf("%f",&real);
                printf ("\nInforme o valor da parte complexa: ");
                scanf ("%f", &complexo);
                if (complexo>=0)
                        printf ("\nNumero complexo em coordenadas cartesianas: %f + %fi",
                            real,complexo);
                else
                        printf ("\nNumero complexo em coordenadas cartesianas: %f  %fi",
                            real,complexo);
                angulo = atanf(complexo/real);
                modulo = sqrt (pow(real,2)+pow(complexo,2));
                printf ("\nNumero complexo na forma polar: %f<%f radianos",modulo,angulo)
                    ;
        }
        else
        {       printf ("\nInforme o ,ngulo (em radianos) e o modulo do numero que deseja
             converter, respectivamente:\n");
                scanf ("%f%f",&angulo,&modulo);
                real = modulo*cos(angulo);
                complexo = modulo*sin(angulo);
                printf ("\nNumero complexo na forma polar: %f<%f radianos",modulo,angulo)
                    ;
                if (complexo>=0)
                        printf ("\nNumero complexo em coordenadas cartesianas: %f + %fi",
                            real,complexo);
                else
                        printf ("\nNumero complexo em coordenadas cartesianas: %f  %fi",
                            real,complexo);
        }
        return 0;
}
```

2.1.2 Exercício 2

```
#include<stdio.h>

int main (){
    float rumo, azimute;
    int opcao, direcao;
    printf("Selecione uma opcao para conversao:");
    printf("\n1 - Azimute para rumo;");
    printf("\n2 - Rumo para azimute.\n");
    scanf("%d", &opcao);
    switch(opcao){

        case 1:
            printf("Informe o azimute: ");
            scanf("%f", &azimute);
            if(azimute >= 0 && azimute <= 90){
                rumo = azimute;
                printf("O rumo corresponte ao azimute %.4f eh %.4f NE", azimute, rumo);
                break;
            }
            else if(azimute > 90 && azimute <= 180){
                rumo = 180.00 - azimute;
                printf("O rumo corresponte ao azimute %.4f eh %.4f SE", azimute, rumo);
                break;
            }
            else if(azimute > 180 && azimute <= 270){
                rumo = azimute - 180.00;
                printf("O rumo corresponte ao azimute %.4f eh %.4f SO", azimute, rumo);
                break;
            }
            else if(azimute > 270 && azimute <= 360){
                rumo = 360.00 - azimute;
                printf("O rumo corresponte ao azimute %.4f eh %.4f NO", azimute, rumo);
                break;
            }
            else{
                printf("Azimute invalido");
                break;
            }
        case 2:
            printf("Informe o rumo: ");
            scanf("%f", &rumo);
            printf("Selecione a orientacao do rumo:");
            printf("\n1 - NE;");
            printf("\n2 - SE;");
            printf("\n3 - SO;");
            printf("\n4 - NO.");
            scanf("%d", &direcao);
            if(rumo >=0 && rumo <=90 && direcao == 1){
                azimute = rumo;
                printf("O azimute corresponte ao rumo %.4f NO eh %.4f.", rumo, azimute);
                break;
            }
            else if(rumo >=0 && rumo <=90 && direcao == 2){
                azimute = 180.00 - rumo;
                printf("O azimute corresponte ao rumo %.4f NO eh %.4f.", rumo, azimute);
                break;
            }
            else if(rumo >=0 && rumo <=90 && direcao == 3){
                azimute = rumo + 180.00;
```

```
                printf("O azimute corresponte ao rumo %.4f NO eh %.4f.", rumo, azimute);
                break;
            }
            else if(rumo >=0 && rumo <=90 && direcao == 4){
                azimute = 360.00 - rumo;
                printf("O azimute corresponte ao rumo %.4f NO eh %.4f.", rumo, azimute);
                break;
            }
            else{
                printf("Dados invalidos");
                break;
            }
    }
    return 0;
}
```

2.1.3 Exercício 3

```
#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;
}
```

2.1.4 Exercício 4

```
#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;
}
```

2.1.5 Exercício 5

```
#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;
}
```

2.1.6 Exercício 6

```
#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;
}
```

2.1.7 Exercício 7

2.2 Laços de Repetição

2.2.1 Exercício 1

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

```
        double u, v, c; // u = energia  v = tensao      c = capacitancia
        int k=0; // Variavel que serah utilizada para contar a base
        printf("\nDigite a energia em Wh: ");
        scanf("%lf",&u);
        printf("\nDigite a diferenca de potencial em Volts: ");
        scanf("%lf",&v);
        c= u*7200/pow(v,2);
        while(c > 1000){
                c = c / 1000;
                k= k+1;//Conta quantas vezes o numero pode ser dividido por 1000 ate ser
                    menor que 1000
        }
        while(c<1){
                c = c *1000;
                k = k-1;//Conta quantas vezes o numero poder ser multiplicado por 1000
                    ate fica maior que 1
        }
        printf("\nCapacitancia necessaria: %.2lf*10^%i F",c,k*3);
        return 0;
}
```

2.2.2 Exercício 2

```
#include <stdio.h>

int main(){
        long long int dec, aux;
        int resto, n, i, tam, j=0;
        char bin [32], b[32], hexa[8], h[8];
        int deca[10], bcdaux[40], bcd[40];

        printf("\nDigite um numero decimal para ser convertido para binario e hexadecimal
            : ");
        scanf("%ld",&dec);

        aux = dec;

        //Inicio conversao binario
        for(i=0;i<32;i++){
                if(dec % 2 == 1)
                        bin[i] = '1';
                else bin [i] = '0';
                        dec = dec/2;
        }



        for(i=0;i<32;i++){
                b[31-i] = bin[i];
        }

        printf("\nNumero digitado em binario: %s ",b);

        dec = aux;

        //Inicio conversao hexadecimal
        for(i=0;i<8;i++){
                resto = dec % 16;
                dec = dec/16;
                if(resto > 9){
```

```
                        h[i] = 'A'+(resto-9);
                }
                else
                        h [i] = '0'+resto;
        }

        for(i=0;i<8;i++){
                hexa[7-i] = h[i];
        }
        printf("\nNumero digitado em hexadecimal: %s",hexa);

        dec = aux;
        //Inicio convercao para BCD
        for(i=0;i<10;i++){
                deca[i] = dec % 10;
                dec = dec/10;
        }
        for(i=0;i<10;i++){
                bcdaux[j] = deca[i]%2;
                deca[i]= deca[i]/2;
                j++;
                bcdaux[j] = deca[i]%2;
                deca[i]= deca[i]/2;
                j++;
                bcdaux[j] = deca[i]%2;
                deca[i]= deca[i]/2;
                j++;
                bcdaux[j] = deca[i]%2;
                deca[i]= deca[i]/2;
                j++;
        }
        printf("\n");
        for(i=0;i<40;i++)
                bcd[39-i] =bcdaux[i];
        for(i=0;i<40;i++)
                printf("%d",bcd[i]);
        return 0;
}
```

```
#include <stdio.h>
#include <math.h>
#define PI 3.1415
int main ()
{       int i;
        float r, corrente,dr,a,x;
        a=3*pow(10,11);
        dr=(0.001)/100000;      //comprimento infinitesimal
        r=0.001;
        for (i=0;i<100000;i++)
        {       x=pow(r,3);
                corrente+=x*dr;
                r+=dr;
        }
        corrente=2*PI*a*corrente;
        printf ("Corrente percorrendo o fio: %f A.\n",corrente);
        return 0;
}
```

```
include <stdio.h>
#include <math.h>
#define N 20
```

```
#define K 8990000000
int main ()
{       float forca_x,forca_y,modulo_forca,carga[N],distancia[N],angulo[N],
    carga_particula;
        int i;
        printf ("Digite os dados para que o conjunto de forcas possam ser calculados:\n\n
            ");
        printf ("Digite o valor da carga da particula central do sistema: ");
        scanf ("%f",&carga_particula);
        for (i=0;i<N;i++)
        {       printf ("Digite a carga, em Coulomb, da particula %d: ",i+1);
                scanf ("%f",&carga[i]);
                printf ("Digite a distancia da particula %d em relacao a carga central (
                    em metros): ",i+1);
                scanf ("%f",&distancia[i]);
                printf ("Digite o angulo que a particula %d faz em relacao ao semi-eixo
                    positivo x (em radianos): ",i+1);
                scanf ("%f",&angulo[i]);
                forca_x+=(K*carga_particula*carga[i]*cos(angulo[i]))/pow(distancia[i],2);
                forca_y+=(K*carga_particula*carga[i]*sin(angulo[i]))/pow(distancia[i],2);
        }
        modulo_forca=sqrt(pow(forca_x,2)+pow(forca_y,2));
        printf ("Forca em X: %.4f\nForca em Y: %.4f\nModulo: %.4f\n\n",forca_x,forca_y,
            modulo_forca);
        return 0;
}
```

2.2.3 Exercício 3

2.2.4 Exercício 4

2.2.5 Exercício 5

```
#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;
}
```

2.2.6 Exercício 6

```c
#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;
}
```

2.2.7 Exercício 7

```c
#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;
}
```

2.2.8 Exercício 8

```c
#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 invervalo dado: %f\n\n",xc);
        return 0;
}
```

2.2.9 Exercício 9

```
#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 5

int main(){
    float r[5], rt;
    printf("Digite as resistencias: ");
    for(int i=0; i<N; i++){
        printf("\nResistor %d",i+1);
        scanf("%f",&r[i]);
        rt+=1/r[i];
    }
    printf("\nA resistencia equivalente eh: %f",rt);
    return 0;
    }
```

2.3.2 Exercício 2

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

int main (){
    int i, j,m ,n, aux, aux2, aux3;//i = variavel contadora, m = numero de elementos, n =
         numero de classes, aux = variavel auxiliar;
    int maior, menor;
    float media = 0;
    int A, a;//A = Amplitude dos dados, a = amplitude das classes
    printf("Insira a quantidade de dados a serem inseridos: ");
    scanf("%d", &m);
    int vetor[m];//Armazenamento de dados;
    system("pause");

    for(i=0; i<m; i++){
        system("cls");
        printf("Insira o valor %d: ", i+1);
        scanf("%d", &vetor[i]);
    }

    n = (int)sqrt(m);//O número de classes é um número inteiro; Não se pode ter 7,5
        classes !
    n++;
    menor = vetor[0];// Base para encontrar o menor número
    maior = vetor[0];//Base para encontrar o maior número

    for(i=0; i<m; i++){
        if(vetor[i] >= maior){
            maior = vetor[i];//Encontrando o maior numero dos dados inseridos
        }
```

```
    else if(vetor[i] <= menor){
        menor = vetor[i];//Encontrando o menor número do dados inseridos
    }
}

A = maior - menor;
a = (int)(A/n);//Calculando a amplitude das classes
a++;

int f[n], fa[n];//Declarand vetores para aramazenar dados das frequências

float pm[n], fr[n], fra[n], pmf[n];// Cada classe terá um número de cada tipo de
    frequência, pm = ponto medio; pmf = pm*f

for(i=0; i<n; i++){ // Atribuindo 0
    f[i] = 0;
    fa[i] = 0;
    fr[i] = 0;
    fra[i] = 0;
}

aux = menor;
aux2 = aux + a;

for(i=0; i<n; i++){
    for(j=0; j<m; j++){//Frequencia
        if(vetor[j]>=aux && vetor[j]<aux2){
            f[i]++;
        }
    }
    pm[i] = (float)(aux2 + aux)/2;
    pmf[i]= (pm[i]*f[i]);
    aux = aux2;
    aux2 = aux2 + a;
}

for(i=0; i<n; i++){
    media = media + pmf[i];
    if(i == 0){
        fa[i] = f[i];
    }
    else{
        fa[i] = f[i] + fa[i-1];
    }
}

media = media/m;

for(i=0; i<n; i++){
    fr[i] = (float)f[i]/m;

    if(i == 0){
        fra[i] = fr[i];
    }
    else{
        fra[i] = fr[i] + fra[i-1];
    }
}

aux = menor;
aux2 = aux + a;
```

```
    printf("Intervalo\tF\tFa\tFr\tFra\tPm\tPm*f\n\n");
    for(i=0; i<n; i++){
        printf("%d  |-  %d\t%d\t%d\t%.2f\t%.2f\t%.2f\t%.2f\n",aux, aux2, f[i], fa[i], fr[
            i], fra[i], pm[i], pmf[i]);

        aux = aux2;
        aux2 = aux2 + a;

    }

    printf("\n\n%.2f", media);



    return 0;
}
```

2.3.3 Exercício 3

```
#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;
}
```

2.3.4 Exercício 4

```
#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;
}
```

2.3.5 Exercício 5

```
#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);


}
```

2.3.6 Exercício 6

```
#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);


}
```

2.3.7 Exercício 7

```
#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]);

}
```

2.3.8 Exercício 8

```
#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("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<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]);


}
```

2.3.9 Exercício 9

```
#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]);

}
```

2.3.10 Exercício 10

```
#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]);
}
```

2.3.11 Exercício 11

```
#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;
}
```

2.3.12 Exercício 12

```
#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 );

}
```

2.3.13 Exercício 13

```
#include <stdio.h>
#define MAX_colunas 4
#define MAX_linhas  2
int main ()
{       float matriz[MAX_linhas][MAX_colunas],transposta[MAX_colunas][MAX_linhas];
        int i,j;
        printf ("Digite a matriz na qual deseja-se obter sua transposta:\n");
        for (i=0;i<MAX_linhas;i++)
        {       for (j=0;j<MAX_colunas;j++)
                {       printf ("\nLinhas %d e coluna %d: ",i+1,j+1);
                        scanf ("%f",&matriz[i][j]);
                        transposta[j][i]=matriz[i][j];
                }
        }
        printf ("\nMatriz digitada pelo usuario:\n");
        for (i=0;i<MAX_linhas;i++)
        {       for (j=0;j<MAX_colunas;j++)
                {       printf ("%.3f   ",transposta[i][j]);
                        if (j==MAX_colunas-1)
                                printf ("\n");
                }
        }
        printf ("\nMatriz transposta obtida:\n");
        for (i=0;i<MAX_colunas;i++)
        {       for (j=0;j<MAX_linhas;j++)
                {       printf ("%.3f   ",transposta[j][i]);
                        if (j==MAX_linhas-1)
                                printf ("\n");
                }
        }
}
```

2.3.14 Exercício 14

```
#include <stdio.h>
#include <math.h>
#define N 3
int main ()
{       float vetor[N],R[N],somatorio=0,norma;
        int i;
        printf ("Digite os valores correspondentes ao vetor:\n");
        for (i=0;i<N;i++)
```

```
        {       printf ("Posicao %d: ",i+1);
                scanf ("%f",&vetor[i]);
                somatorio+=pow(vetor[i],2);
        }
        norma=sqrt(somatorio);
        somatorio=0;
                printf ("\nVetor Resultante:\n");
        for (i=0;i<N;i++)
        {       R[i]=vetor[i]/norma;
                if (i==0)
                        printf ("[");
                printf ("  %.3f",R[i]);
                if (i==2)
                        printf ("  ]");
        }
        return 0;
}
```

2.3.15 Exercício 15

```
#include <stdio.h>
int main ()
{       float somatorio=0,numero;
        bool i;
        do
        {       printf ("Digite o valor a ser somado/subtraido: ");
                scanf ("%f",&numero);
                somatorio+=numero;
                printf ("Deseja continuar somando/subtraindo numeros? 0 para continuar e
                    1 para encerrar: ");
                scanf ("%d",&i);
        }
        while (i==0);
        printf ("Soma total: %f.",somatorio);
        return 0;
}
```

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-Periodo 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 inercia 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 perído 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 perído 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=w²*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/w², 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,massa_1,massa_2,v_inicial_1;
};
int main ()
{       struct registro colisoes;
        printf ("Digite a velocidade inicial da primeira particula (projetil) em m/s: ");
        scanf ("%f",&colisoes.v_inicial_1);
        printf ("Digite a massa da primeira particula em KG: ");
        scanf ("%f",&colisoes.massa_1);
        printf ("Digite a massa da segunda particula em KG(alvo): ");
        scanf ("%f",&colisoes.massa_2);
        colisoes.v_final_1=((colisoes.massa_1-colisoes.massa_2)/(colisoes.massa_1+
            colisoes.massa_2))*colisoes.v_inicial_1;
        printf ("Velocidade Final da particula 1: %.4f m/s.\n\n",colisoes.v_final_1);
        colisoes.v_final_2=(2*colisoes.massa_1/(colisoes.massa_1+colisoes.massa_2))*
            colisoes.v_inicial_1;
        printf ("Velocidade Final da particula 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 subdivisoes: ");
        scanf ("%d",&trapezio_repetido.subdivisoes);
        trapezio_repetido.particoes=(trapezio_repetido.b-trapezio_repetido.a)/
            trapezio_repetido.subdivisoes;
        trapezio_repetido.x=trapezio_repetido.a;
        while (trapezio_repetido.i<=trapezio_repetido.subdivisoes)
        {       if (trapezio_repetido.i==0||trapezio_repetido.i==trapezio_repetido.
            subdivisoes)
                        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.subdivisoes*(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 Epson: ");
        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\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>

int main(){
    float *r, rt;
    int n;

    printf("Digite o numero de resistores associados em serie: ");
    scanf("%i",&n);

    r = (float*) calloc(n,sizeof(float));

    printf("Digite as resistencias: ");
    for(int i=0; i<n; i++){
        printf("\nResistor %d: ",i+1);
        scanf("%f",&r[i]);
        rt+=r[i];
    }

    printf("Resistencia equivalente = %f",rt);


    return 0;
}
```

2.6.2 Exercício 2

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main ()
{       float indutancia, *vet_ind;
        int opcao,n,i;
        printf ("Digite o numero de indutores que existem no circuito: ");
        scanf ("%d",&n);
        vet_ind=(float *)malloc(n * sizeof(float));
        for (i=0;i<n;i++)
        {       printf ("Digite a resistencia (em Henrys) do indutor %d: ",i+1);
                scanf ("%f",&vet_ind[i]);
                if (i==0)
                        indutancia=vet_ind[i];
                else
                {
                        printf ("Digite 1 para indicar ligacao em paralelo com os
                            indutores anteriores e 2 para indicar ligacao em serie: ");
                        scanf ("%d",&opcao);
                        if (opcao==1)
                        {       indutancia=(indutancia*vet_ind[i])/(indutancia+vet_ind[i
                            ]);
                        }
                        else
                                indutancia+=+vet_ind[i];
                }
                if (i!=n-1)
```

```
                        printf ("Indutancia calculada: %.3f\n",indutancia);
        }
        printf ("Indutancia total oferecida pelo circuito: %.3f",indutancia);
        return 0;
}
```

2.6.3 Exercício 3

```
#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;
                }
}



}
```

2.6.4 Exercício 4

```
#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;
}
```

2.6.5 Exercício 5

```
#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;
}
```

2.6.6 Exercício 6

```
#include <stdio.h>
#include <stdlib.h>
int main ()
{
        float *vet1, *vet2, res;
        int tam, i;

        printf("\nDigite o tamanho dos vetores: ");
        scanf("%d",&tam);

        vet1 = (float*) malloc(tam*sizeof(float));
        vet2 = (float*) malloc(tam*sizeof(float));

        printf("\nDigite os valores dos vetores: ");
        for(i=0;i<tam;i++){
                printf("\nvetor1[%d]: ", i+1);
                scanf("%f",&vet1[i]);
                printf("\nvetor2[%d]: ", i+1);
                scanf("%f",&vet2[i]);
                res = res+ vet1[i]*vet2[i];
        }

        printf("\nO produto escalar do vetor eh: %f",res);

        return 0;
}
```

2.6.7 Exercício 7

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main ()
{       float resistencia, *vet_res;
        int opcao,n,i;
        printf ("Digite o numero de resistores que existem no circuito: ");
        scanf ("%d",&n);
        vet_res=(float *)malloc(n * sizeof(float));
        for (i=0;i<n;i++)
        {       printf ("Digite a resistencia (em ohlms) do resistor %d: ",i+1);
                scanf ("%f",&vet_res[i]);
                if (i==0)
                        resistencia=vet_res[i];
                else
                {
                        printf ("Digite 1 para indicar ligacao em paralelo com os
                            resistores anteriores e 2 para indicar ligacao em serie: ");
                        scanf ("%d",&opcao);
```

```
                        if (opcao==1)
                        {       resistencia=(resistencia*vet_res[i])/(resistencia+vet_res
                            [i]);
                        }
                        else
                                resistencia+=+vet_res[i];
                }
                if (i!=n-1)
                        printf ("Resistencia calculada: %.3f\n",resistencia);
        }
        printf ("Resistencia total oferecida pelo circuito: %.3f",resistencia);
        return 0;
}
```

2.6.8 Exercício 8

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main ()
{       float capacitancia, *vet_cap;
        int opcao,n,i;
        printf ("Digite o numero de capacitores que existem no circuito: ");
        scanf ("%d",&n);
        vet_cap=(float *)malloc(n * sizeof(float));
        for (i=0;i<n;i++)
        {       printf ("Digite a capacitancia (em faraday) do capacitor %d: ",i+1);
                scanf ("%f",&vet_cap[i]);
                if (i==0)
                        capacitancia=vet_cap[i];
                else
                {
                        printf ("Digite 1 para indicar ligacao em paralelo com os
                            capacitores anteriores e 2 para indicar ligacao em serie: ")
                            ;
                        scanf ("%d",&opcao);
                        if (opcao==1)
                                capacitancia+=+vet_cap[i];
                        else
                                capacitancia=(capacitancia*vet_cap[i])/(capacitancia+
                                    vet_cap[i]);
                }
                if (i!=n-1)
                        printf ("Capacitancia calculada: %.3f\n",capacitancia);
        }
        printf ("Capacitancia total oferecida pelo circuito: %.3f",capacitancia);
        return 0;
}
```

Referências

[1] Luís Damas. Linguagem c. *10a. Edição, LTC*, 2007.

[2] David HALLIDAY, Robert RESNICK, and Jearl WALKER. Fundamentos de física iii. *Rio de Janeiro: LTC*, 2009.