ICEI | Instituto de Ciências Exatas e Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

INSTITUTO DE CIÊNCIAS EXATAS EXATAS E INFORMÁTICA

Caderno de Exercícios Algoritmos C e C++

CURSOS

Ciência de Dados, Ciência da Computação e afins

DISCIPLINA

Algoritmos e Estrutura de Dados

PROFESSOR

Prof. Dr. Gustavo Luís Soares

ALUNOS COLABORADORES

Paulo Henrique Braga Pessoa
Vinícius Henrique Dutra Goddard

Versão 2.0 - 7 de fevereiro de 2024

Sumário

1 Notas gerais - Gustavo Soares

Fundamentação 1: Padrões de escrita e notação

A notação empregada foi baseada em padrões de representação normalmente encontrados na literatura matemática, ciência da computação e engenharia. Entretanto, mesmo nestas áreas, é comum encontrar trabalhos científicos utilizando notações distintas para denotar, por exemplo, um mesmo parâmetro, grandeza ou operação matemática. Nestes casos, foram definidos padrões que convenientemente atendesse este trabalho. Em alguns casos, o leitor pode discordar quanto à escolha da notação. Justifica-se que as definições e escolhas de símbolos pela tentativa de descrever os algoritmos, formulação e outras partes do texto de forma clara, objetiva e não ambígua. Infelizmente, devido à grande quantidade de símbolos, talvez o leitor considere que as metas acima não tenham sido atingidas. Os critérios de notação e algumas nomenclaturas básicas empregados neste texto estão descritos logo abaixo.

Como de costume, a fonte $\mathbb{R}$ é utilizada para representar conjuntos matemáticos sendo neste caso dos números reais, e $\mathbb{Z}$ dos números inteiros, por exemplo. O uso de expoentes numéricos nesses símbolos denota a ordem do espaço matemático que eles representam, por exemplo $\mathbb{R}^2$. Conjuntos numéricos são apresentados com letras maiúsculas em negrito, como em $\mathbf{A}$. Essa mesma notação é empregada para matrizes. O contexto descarta a possível confusão entre conjuntos e matrizes. O negrito em letras minúsculas denota vetores como é o caso de $\mathbf{x}$. A representação das casas decimais é marcada pelo símbolo "." como em $\pi = 3.14$.

Vocábulos ou trechos em destaque no texto ou que representam uma linguagem menos formal se encontram entre aspas. O itálico é reservado para apresentação inicial de termos técnicos dentro do texto corrente como em "... a verificação de erros em unidades se dá por *testes caixa branca*". Sem perda de generalidade, o itálico também marca os termos em língua estrangeira como *software*. Os termos de linguagem de programação estão em negrito e itálico, como em ***printf***. Os intervalos são delimitados por colchetes, como $[\underline{x}\ \ \overline{x}]$, sendo os limites inferiores de intervalos são denotados pelo acréscimo de grifos acima e abaixo símbolo. De modo particular, a notação $\overline{x}$ também significa média aritmética. As letras i, j e k denotam iterações de processos, identificam elementos de conjuntos e particularizam termos de vetores, entre outros usos. Por exemplo, a_{ij} significa o elemento da linha i e coluna j da matriz $\mathbf{A}$. O símbolo $'$ é utilizado como uma notação suplementar ou de transformação do termo assinalado. Por exemplo, se $\mathbf{A}$ é uma matriz, então $\mathbf{A}'$ é a matriz resultante de uma transformação.

Provavelmente, existam exceções que fujam às regras apresentadas. Nestes casos, espera-se que o contexto e o rigor das definições sejam suficientes para deixar claras quaisquer outras classes de nomenclatura não cobertos aqui.

Fundamentação 2: Listas e organização

As listas estão classificadas em três grupos, a saber:

a) Exercícios de fixação - Trata-se de um conjunto de exercícios de uso direto dos comandos, operações e funções da linguagem de programação em foco. Sugere-se que o estudante se dedique a resolver todos os exercícios propostos tendo em mente a necessidade do domínio completo (sem apoio de ferramentas computacionais e/ou anotações) da resolução de cada exercício para aprendizagem básica de AEDs I.

b) Exercícios de complementares - As listas têm finalidade de treinar o estudante com problemas extras sobre implementação de programas e entendimento de problemas um pouco mais complexos.

c) Exercícios de aplicação - Considero este grupo como "especial", pois os exercícios buscam conectar o estudante no mundo real. Os exercícios, muitas vezes simples, trazem reflexões nos postulados das questões. Em alguns casos as informações para resolver estão em artigos científicos que, por sua vez, adicionam uma nova dimensão no problema.

(Nota: Os exercícios das listas têm numeração única e permanente.

2 Termos técnicos da etapa

Fundamentação 3: Algoritmos e programação - parte 1

Desenvolver software envolve uma complexa combinação de habilidades e conhecimentos. Comecemos pelo básico: a *lógica de programação*. Ela é a capacidade de se resolver problemas através de sequências lógicas, determinando como um programa deve ser estruturado. Às vezes, pode-se cometer um *erro de lógica*, o que leva o software a se comportar de maneira não intencional. Outros erros incluem o *erro de execução*, frequentemente relacionado a problemas de recurso ou ambiente, e o *erro de compreensão do problema*, quando o desenvolvedor não entende completamente a demanda.

Toda solução começa com um *algoritmo*, um conjunto ordenado de passos para resolver um problema específico. Muitas vezes, antes de começarmos a escrever código, ilustramos nossos algoritmos através de *pseudocódigo* ou *fluxograma*. Estes são mecanismos simplificados que facilitam a visualização do processo.

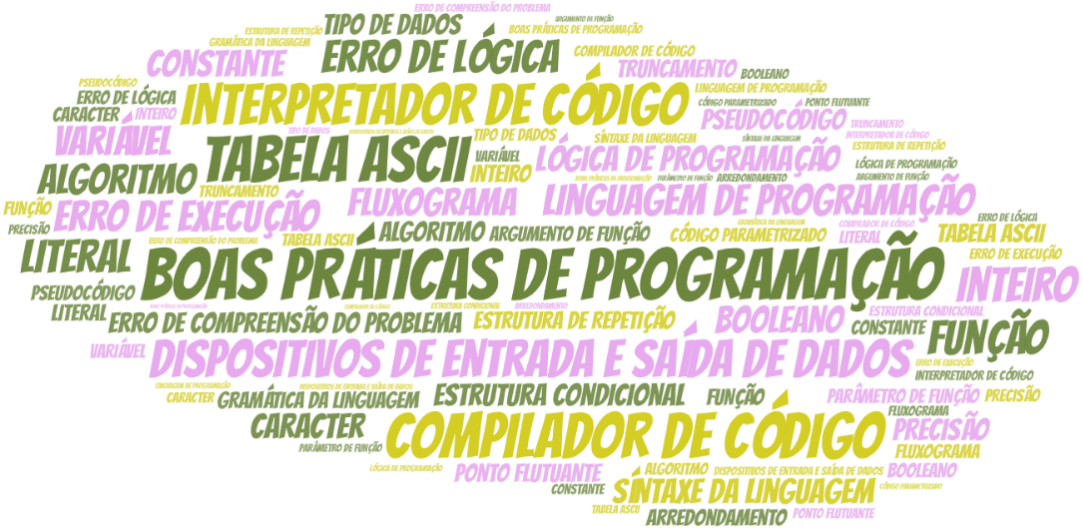
Ao passar para a codificação, escolhe-se uma *linguagem de programação* específica. Cada linguagem possui sua *síntaxe* e *gramática*, que determinam como as instruções devem ser escritas e organizadas. Durante a codificação, usamos diversas estruturas como *estruturas condicionais* (***if***, ***else***) e *estruturas de repetição* (***for***, ***while***).

Uma *função* é um conjunto de instruções que realizam uma tarefa específica e podem ser reutilizadas. Funções podem ter *parâmetros* (definido na declaração da função), que são informações necessárias para sua execução, e quando chamamos uma função, fornecemos *argumentos* (valores dos parâmetros durante a execução) para esses parâmetros. Um *código parametrizado* permite maior flexibilidade, pois pode ser ajustado ou expandido através de parâmetros.

Variáveis e constantes são elementos fundamentais. Uma *variável* é uma área de armazenamento que pode mudar ao longo da execução, enquanto uma *constante* mantém seu valor fixo. Cada variável ou constante possui um *tipo de dados*, que pode ser *booleano*, *inteiro*, *ponto flutuante*, entre outros. O *literal* é a representação direta de um valor, como o número 10 ou a letra "A". Falando em letras, a *tabela ASCII* é um padrão de codificação que mapeia caracteres para números. Para obter precisão nos cálculos, precisamos entender conceitos como *precisão*, *truncamento* e *arredondamento*.

Em tempo, é fundamental reconhecer a importância dos *dispositivos de entrada e saída de dados* para interagir com usuários ou outros sistemas. E, por fim, quando o código está pronto, ele pode ser transformado em software executável por meio de um *compilador de código* ou ser executado linha a linha por um *interpretador de código*.

Ao longo de todo do processo de desenvolvimento de software, é essencial seguir *boas práticas de programação* para garantir que o software seja eficiente, legível e livre de erros.



3 Lógica e pseudocódigos - Exercícios de fixação N°01

Exercício 1

Considerando a linguagem C, assinale as alternativas que são possíveis nomes para variáveis.

a) Somatorio	b) Somatório	c) X	d) Tipo.1	e) aux
f) printf	g) valor final	h) Tipo_1	i) valor-total	j) int

Exercício 2

Considerando a linguagem C, determine o tipo das variáveis que armazenam os seguintes valores:

a) 7	b) P	c) 3.141592653	d) 78.3	e) 0
f) 3.3333	g) TRUE	h) FALSO	i) 128742	j) 0.0

Exercício 3

João, um aluno de ensino médio, precisa tirar 10 pontos na sua prova final para poder passar na etapa. Crie um programa, em Portugol e em C, que receba do usuário a nota de João e, caso seja menor do que 10, imprima "reprovado". Caso seja igual ou maior que 10, imprima "aprovado".

Clique aqui para saber mais - sintaxe de Portugol

Exercício 4

Escreva um programa em Portugol e em C que receba através da entrada do usuário, dois números a serem comparados. O programa deve imprimir o maior dos dois, mas caso sejam iguais, deve imprimir "os números são iguais".

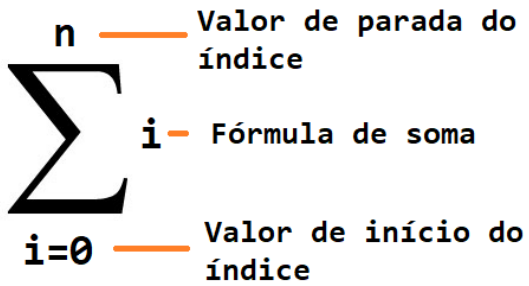
Clique aqui para saber mais - sintaxe de Portugol

Exercício 5

Escreva um programa em Portugol e em C que receba dois números através da entrada do usuário e imprima: a) o maior deles; b) seu produto; c) a média aritmética; e d) a soma do antecessor do primeiro com o sucessor do segundo.

Exercício 6

A notação matemática é uma forma de linguagem universal que pode ser reconhecida independente da região. Nela, existem *Somatórios* e *Produtórios*, que são formas recursivas de somar e multiplicar números gradualmente.



O somatório ao lado utiliza a *Notação Sigma*. Nele, o *índice* "i" começa em 0 e é somado à "i + 1" até que o valor de "i" seja igual a N (nesse exemplo: 10). Na programação, o resultado do *somatório* é armazenado em uma variável inicializada em 0 (nesse exemplo: X).

$X = 0 + 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10$

Crie um programa em Portugol ou C que calcule e imprima o resultado de:

a) $\sum_{i=1}^{10} i$ b) $\sum_{i=0}^{10} i + 1$ c) $\prod_{i=0}^{15} i$ d) $\prod_{i=1}^{10} i$

e) Em (d) a fórmula representa qual operação/função matemática?

4 Lógica e pseudocódigos - Exercícios complementares N° 01

Exercício 7

Crie um programa em Portugol e em C que receba, através da entrada do usuário, um número ***inteiro*** e um número ***decimal*** e, então, faça a soma dos dois, de modo a imprimir um resultado como ***inteiro***.

Clique aqui para saber mais - sintaxe de Portugol

Exercício 8

Uma loja de brinquedos vende bolas compostas de duas cores dentre branco, amarelo e azul. Crie um programa em Portugol ou C que receba, através da entrada de usuário, as cores que compõem uma bola e imprima “true” caso essa bola tenha as cores azul e branco. Caso contrário, imprima "false".

Clique aqui para saber mais - sintaxe de Portugol

Exercício 9

Crie um programa em Portugol ou C que calcule e imprima o resultado de:

a) $\prod_{i=0}^{5} \frac{i}{i+1}$ b) $\prod_{i=1}^{7} i$ c) $\sum_{i=4}^{17} i+1$

Exercício 10

Para cada um dos problemas a seguir, expresse um algoritmo que possa ser utilizado para solucioná-lo. Utilize o Portugol e passe para a linguagem C.

a) Leia três notas e seus respectivos pesos, calcule e mostre a média ponderada dessas notas.

b) Receba uma temperatura dada na escala Celsius (C) e imprima o equivalente em Fahrenheit (F).

c) Leia uma quantidade em horas e imprima seu valor em segundos.

(Nota: Em (b) use a fórmula de conversão: $F = \frac{9}{5}C + 32$)

Exercício 11

Numeração reservada; exercício ainda em elaboração.

Exercício 12

Numeração reservada; exercício ainda em elaboração.

5 Lógica e pseudocódigos - Exercícios de aplicação N°01

Exercício 13

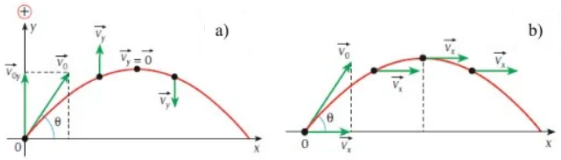
Apresentar os passos necessários para encontrar as raízes de uma equação do segundo grau, caso exita em $\mathbb{R}$.

Exercício 14: revisado 1x

Apresentar os passos necessários para elaborar uma planilha Excel para contabilizar as despesas de uma família. Considere as rubricas Utilidades (energia elétrica, água e esgoto), Educação, Supermercado, Transporte, Restaurante, Faxina, Vestuário, Outros. Cada linha da planilha deve ser preenchida segundo o cabeçalho das colunas Data, Descrição, Rubrica, Valor. Ao final a planilha deve apresentar o valor total das despesas, valores numéricos e percentuais das despesas por rubrica. Fazer gráfico de barras dos valores percentuais com título e eixos nomeados.

Exercício 15: revisado 1x

Na Física, sub-área Mecânica, considere o lançamento de objetos no ar a uma dada velocidade inicial v_o e um ângulo de lançamento θ, como mostrado na Fig.15. Simplificadamente, pode-se desconsiderar o atrito com ar e aproximar as equações de movimentos horizontais por MRU (Movimento Retilíneo Uniforme) e verticais MRUV (Movimento Retilíneo Uniforme Variado) como mostrado na expressões matemáticas abaixo.



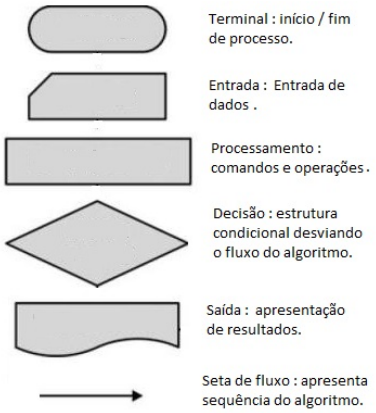
a) Mov. vertical. (b) Mov. horizontal.

MRU	MRUV
$x = v_o \cos\theta t$	$y = v_o \sin\theta t + 0.5 a_y t^2$
$v_x = v_o \cos\theta$	$v_y = v_o \sin\theta + a_y t$
v_x constante	$v_y^2 = (v_o \sin\theta)^2 + 2 a_y y$
$a_x = 0$	$a_y = -g = -9,8 m/s^2$

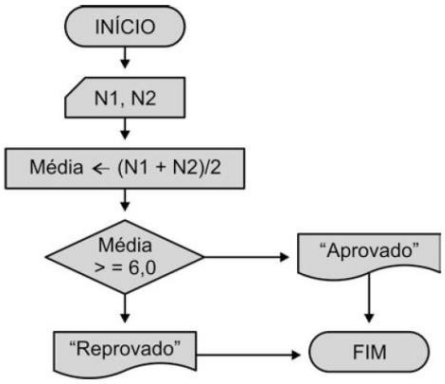
Considere que um projétil foi lançado verticalmente para cima a partir da origem. Sabendo-se a distância percorrida x de um ponto P da trajetória, o tempo t que levou para atingir P e o ângulo de lançamento θ, desenvolver uma lógica de programação para encontrar y em P.

Exercício 16

Fluxogramas (veja Fig.16) são representações gráficas de algoritmos cujos elementos geométricos indicam comandos e operações sobre os dados e variáveis. Deseja-se, com o uso de fluxogramas, organizar as ideias e facilitar o entendimento de algoritmos.



a) Elementos de fluxogramas.



a) Fluxograma para computação de notas.

Elabore um fluxograma para descrever:

a) o sistema de avaliação da disciplina Algoritmos e Estrutura de Dados 1;

b) a feitura de um bolo Floresta Negra.

6 Entrada e saída de dados - Exercícios de fixação N°02

Exercício 17

Interprete o código abaixo e aponte sua função.

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

int main()
{
    int numero_1, numero_2, aux;

    printf("Entre com o primeiro numero:");
    scanf("%d", &numero_1);

    printf("\nEntre com o segundo numero:");
    scanf("%d", &numero_2);

    aux = numero_1;
    numero_1 = numero_2;
    numero_2 = aux;

    printf("\nPrimeiro numero: %d // Segundo numero: %d", numero_1, numero_2);

    return 0;
}
```

Exercício 18

Interprete o código abaixo e aponte sua função.

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

int main()
{
    float a, b, c, d, resultado_1, resultado_2;

    printf("entre com o valor de a:");
    scanf("%f", &a);

    printf("\nentre com o valor de b:");
    scanf("%f", &b);

    printf("\nentre com o valor de c:");
    scanf("%f", &c);

    d = pow(b,2)-4*a*c;

    resultado_1 = (-b + sqrt(d))/ (2*a);
    resultado_2 = (-b - sqrt(d))/ (2*a);

    printf("\nOs resultados sao: %f %f", resultado_1, resultado_2);

    return 0;
}
```

(Nota: Sintaxe funções ***pow***(*base,expoente*) e ***sqrt***(*radicando*))

Exercício 19: revisado 1x

Crie um programa em C que:

a) Receba 2 números inteiros e imprima o resto da divisão do primeiro número pelo segundo.

b) Receba um intervalo numérico inteiro e imprima os números inteiros contidos no intervalo.

c) Receba 2 números dupla precisão e imprima o resultado da potenciação do primeiro numero (base) pelo segundo (expoente).

d) Receba um número dupla precisão e imprima sua raiz quadrada.

(Nota: Sintaxe função ***mod*** como *dividendo % divisor*.)

Exercício 20

Crie um programa em C que receba os dados necessários, calcule e imprima a área das seguintes formas geométricas:

a) quadrado;

b) triângulo;

c) círculo;

d) trapézio.

(Nota: Considere a fórmula área do trapézio: $(B + b) * h/2$ sendo B, b e h, a base maior, a base menor e a altura respectivamente.)

Exercício 21

Desenvolva um programa em C que receba um número através da entrada do usuário e imprima o dobro da raiz do resto da divisão desse número por 5.

Exercício 22

Um determinado restaurante oferece a seus clientes x opções de pratos principais, y opções de acompanhamentos e z opções de bebidas. Crie um programa em C que receba os valores de x, y e z e, então, calcule o número total de possíveis refeições desse restaurante.

Exercício 23

Um aluno de física do ensino médio estava aprendendo sobre o cálculo de pressão de gases, quando teve a ideia de facilitar sua vida através da programação. Para ajudá-lo, crie um programa em C que receba os seguintes dados sobre um determinado gás: volume, número de mols e temperatura (em kelvin). E, então, imprima o valor da pressão desse gás.

(Nota: Considere que o cálculo de pressão P possa ser obtido pela correlação $PV = NRT$, sendo V, N e T, o volume, o número de mols, e a temperatura respectivamente; e a constante $R = 0.082$.)

Exercício 24

O ***fgets()*** é uma função da biblioteca C que nos permite receber uma linha de caracteres (incluindo espaços) e, então, armazená-la em uma variável do tipo ***char*** com um vetor de n posições. Sua sintaxe é: ***fgets*** (variavel, n, stdin); Em que "variavel" é o nome da variável ***char*** que armazenará o texto, "n" é o número máximo de caracteres a serem armazenados e "stdin" significa *standard input* ou "entrada padrão", o que determina que os caracteres serão recebidos pela entrada do usuário. A partir dessa informação, crie um programa em C utilizando a função ***fgets*** para receber um texto de até 100 caracteres e imprimi-lo na tela.

(Nota: Declaração de variável ***char*** com vetor de n posições: char variavel[n];)

7 Entrada e saída de dados - Exercícios complementares N° 02

Exercício 25

O dono de uma pequena loja de roupas permite que o cliente pague por seu produto em parcelas mensais a uma determinada taxa de juros compostos. Com o intuito de apresentar imediatamente ao cliente seu gasto total, o dono busca uma ferramenta que automatize o cálculo desse gasto. Para ajudá-lo, crie um programa em C que receba os seguintes dados sobre uma determinada compra: capital inicial, taxa de juros e tempo de pagamento (em meses). E, então, imprima o valor do montante dessa compra.

(Nota: Considere a fórmula juros compostos $M = C(1+i)^t$, sendo M, C, i e t, o montante, o capital inicial, a taxa de juros e o período de tempo respectivamente.)

Exercício 26

Os tipos de variável ***float*** e ***double*** são utilizadas para armazenar números decimais e possuem capacidade de alocação de 4 e 8 bytes, respectivamente. Crie um programa em C que receba, através da entrada do usuário, um valor para uma variável f e outro para a variável d. O primeiro valor deve ser um decimal com até 7 casas decimais, e o outro, a partir de 7 casas. Logo após, o programa deve somar os dois valores e imprimir o resultado com 8 casas decimais.

Exercício 27

Faça um programa em C que leia o dia, mês e ano inserido por um usuário e imprima esses dados no formato convencional brasileiro de data "dia/mês/ano".

Exercício 28

Faça um programa em C que imprima parte da tabela ASCII correspondente às letras minúsculas. A impressão deve conter duas colunas. A primeira o valor decimal e a segunda o caracter.

Exercício 29

Desenvolver um algoritmo ou programa em C que leia um número inteiro e verifique se o número é divisível por 5 e por 3 ao mesmo tempo.

Exercício 30

Um triângulo é um polígono de três lados, sendo que cada lado é menor que a soma dos outros dois lados. Assim, para que um triângulo seja válido, é necessário que o comprimento de seus lados a, b e c obedeçam à seguinte regra:

$a < |b+c|$, $b < |a+c|$ e $c < |a+b|$.

Escreva um programa C que leia os três lados de um triângulo e verifique se tais valores realmente formam um triângulo. Caso afirmativo, informe se o triângulo é isósceles, escaleno ou equilátero. Caso negativo, imprima mensagem na tela sobre a conclusão do teste.

Exercício 31: Interpolação linear entre dois pontos

Dados os pontos (x_0, y_0) e (x_1, y_1), a *interpolação linear* é a linha entre os dois pontos. Assim, para um valor x no intervalo $[x_0 \ \ x_1]$, o valor y é dado por:

$$y = y_0 + (y_1 - y_0)\frac{x - x_0}{x_1 - x_0}.$$

Faça programa em C que leia os limites inferiores e superiores de dois intervalos, um valor interno ao segundo intervalo e calcule o valor interpolado. Se o valor for externo, então imprima mensagem de erro.

(Nota: Fora do intervalo $[x_0 \ \ x_1]$ denominamos de *extrapolação linear*.)

8 Entrada e saída de dados - Exercícios de aplicação N°02

Fundamentação 4: Cálculo Numérico

O cálculo numérico é uma disciplina que relaciona métodos computacionais na resolução de problemas em diversas ciências, mediante a execução de uma sequência finita de operações aritméticas e/ou cálculo de funções matemáticas. Como resultado, os valores encontrados são aproximações, cuja diferença do resultado esperado ou exato denomina-se erro. Como consequência, os erros serão propagados e talvez aumentados se houver etapas subsequentes.

Fundamentação 5: Erros de arredondamento e de truncamento

Dois tipos de erro destacam-se em cálculo numérico. Os erros de arredondamento são aqueles relacionados com as limitações existentes na forma de representar números em máquina. Por outro lado, o erro de truncamento com a substituição de um processo infinito de operações por outro finito. Em tempo, podemos ainda apresentar o valor arredondado que trata-se do uso de número de dígitos reduzido para simplificar uma grandeza. Por exemplo, se considerarmos o número de Euler (ou número neperiano) $e = 2.71828182845904523536028 7\ldots$ como mostrato nas correlações:

$$\underbrace{e = \sum_{n=0}^{\infty} \frac{1}{n!}}_{\text{formulação matemática}} \qquad e = \underbrace{\frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \ldots + \frac{1}{k!}}_{\text{função aproximada}} + \underbrace{\frac{1}{(k+1)!} + \ldots + \frac{1}{(n)!}}_{\text{erro arredondamento}}$$

a) haverá $n = k+1$ que extrapola o tipo de variável, ocorrendo o erro de arredondamento; b) $e = 2.71828$ é valor arredondado, sendo a diferença com valor conhecido também chamado de erro de arredondamento; c) para $n = 7$, $e = 2.71825396825397$ com erro de truncamento $0.00002786020507523536028 7$. Uma questão do programador é atentar-se, de acordo com o problema, sobre a possibilidade de ocorrência dos erros e providenciar o tratamento ou aceitação.

Exercício 32

Em 1991, durante a Guerra do Golfo, um sistema antiaéreo americano de mísseis na Arábia Saudita falhou ao interceptar um míssil vindo do Iraque. O míssil destruiu acampamentos americanos deixando 28 soldados mortos e 100 feridos. A causa apontada foi um erro de arredondamento no software calculou incorretamente o tempo de ação dos mísseis Patriot em relação aos os mísseis inimigos Scud. Percebe-se que a utilização correta de tipos de dados pode influenciar decisivamente nos sistemas computacionais. As operações matemáticas, como no exemplo de Rump [1] abaixo, podem gerar valores que extrapolam os limites definidos pelos tipos de dados. Considere o problema abaixo:

$$f = 333.75b^6 + a^2(11a^2b^2 - b^6 - 121b^4 - 2) + 5.5b^8 + \frac{a}{2b}$$

Com a = 77617 e b = 33096, computadores utilizando simples, dupla e precisão estendidas produzem os seguintes resultados:

$$\begin{aligned} float \quad f &= 1.172603\ldots \\ double \quad f &= 1.1726039400531\ldots \\ long\ double \quad f &= 1.172603940053178\ldots \end{aligned}$$

Entretanto, $f = -0.827396059946821368141165095479816$. Importante observar que além do valor errado o sinal também está. Escrever um programa para computar f com 20 casas decimais.

Exercício 33

Considere que alguns amigos decidiram fazer um churrasco que será rateado entre todas os n adultos e k crianças participantes, entretanto, as crianças pagam metade. Sabendo que as despesas custaram D, qual o valor de cada participante? Desta forma pede-se elaborar um programa em C que leia do teclado as variáveis e imprima o valor da contribuição de adultos e crianças.

9 Estruturas condicionais - Exercícios de fixação N°03

Exercício 34

Analise atentamente o código em C a seguir:

```c
#include <stdio.h>

int main(){
    int opcao = 0;
    scanf("%d", &opcao);

    switch (opcao) {
    case 1:
        printf("\nvoce digitou 1 e escolheu a primeira opcao.\n");
        break;
    case 2:
        printf("\nvoce digitou 2 e escolheu a segunda opcao.\n");
        break;
    case 7:
        printf("\nvoce digitou 7 e escolheu a terceira opcao.\n");
        break;
    default:
        printf("\nopcao invalida\n");
    }
    return 0;
}
```

Analise o código, descreva brevemente seu funcionamento e responda qual a saída quando a entrada for: a) 7; b) 8; c) 10.

Exercício 35

Desenvolva um programa em C que receba um número e retorne "true"caso ele atenda o seguinte critério: a) seja par; seja ímpar; c) seja negativo; d) seja múltiplo de 7; d) seja divisor de 100.

Exercício 36

Uma determinada loja de salgados vende a unidade de coxinha por 2.00 reais. Porém, esse preço cai para 1.50 em encomendas com mais de 20 unidades. Crie um programa em C que receba a quantidade de coxinhas em uma encomenda e imprima o preço total da compra.

Exercício 37

Uma família procura organizar suas alternativas de viagem para as férias. Para isso, ela pretende levar em conta a taxa de probabilidade de chuva local *Pchuva* e a distância *Dist*. Sabe-se que:

- *Pchuva* é contado em uma escala de 0% a 100% e, para distâncias maiores que 200Km, deve ser no máximo 58%.
- Como a família vai de carro, a distância máxima são 400Km.

Crie um programa em C que receba o nome, distância e probabilidade de chuva local de uma alternativa, criada pelo usuário, assim julgando e imprimindo se é uma opção viável com base nos critérios estabelecidos.

Exercício 38

Um determinado banco demanda que clientes novos criem uma senha de 5 dígitos, utilizando caracteres de 0 a 9, para que acessem seu site. Crie um programa em C que receba uma possível senha para esse site e retorne "true" caso a senha seja válida ou "false" caso seja inválida

(Nota: O primeiro caractere de uma senha deve ser diferente de 0)

10 Estruturas condicionais - Exercícios complementares N°03

Exercício 39

Uma churrasqueira consegue preparar uma carne de boi em 15 minutos, se for mal passada, 20 minutos, se for ao ponto e 25 minutos, se for bem passada. Sabendo que a churrasqueira deve ser preaquecida por 10 minutos antes de ser utilizada, crie um programa em C que receba o ponto da carne desejado e imprima o tempo total de preparo

Exercício 40

Analise atentamente o código a seguir. Qual será o resultado esperado?

```c
#include <stdio.h>

int main() {
    int num1 = 2;
    float num2 = 50.11;
    num1 = num1 * (int)num2;
    if (num1 > 100 && num1 < 100.23){
        printf("verde");
    } else if (num1 < num2 || num1 < 100){
        printf("amarelo");
    } else if (num1 == 100.22){
        printf("amarelo");
    } else {
        printf("azul");
    }
    return 0;
}
```

Exercício 41

Utilizando estruturas condicionais, crie um programa em C que receba 3 números reais e os imprima em ordem crescente.

Exercício 42

Uma empresa de transporte urbano deseja automatizar o cálculo das informações do frete de cada pedido, de forma a estimar qual são as opções possíveis para cada usuário, dado as seguintes regras:

- Existem 3 tipos de transporte: **Caminhão**, **Van**, **Moto**.
- **Moto** possui distância de entrega *DIST* de até 5km e peso máximo de carga de 20kg.
- **Van** possui peso máximo de carga de 1000kg.
- Caso seja necessário, o cliente pode designar a entrega como "Urgente"(URG), o que aumenta em 30% o preço do frete com **Moto**, 70% com **Van** e 100% com **Caminhão**.
- O preço por km *PKM* é R$ 1, tendo acréscimo de seguro no valor de 20% mais para **Motos**.
- Fórmula básica de cálculo do preço: $P = (PKM * DIST) * URG$

Escreva um programa em C que receba os dados *distancia*, *peso* e *urgente* do usuário e imprima o preço do frete nas opções **Moto**, **Van**, **Caminhão**. Caso alguma dessas opções sejam inválidas, imprima "invalido".

Exercício 43

Crie um programa em C que leia as medidas dos lados de um triângulo e imprima se ele é equilátero, isósceles ou escaleno.

11 Estruturas condicionais - Exercícios de aplicação N°03

Exercício 44

Considerando edificações residenciais, a fatura da companhia de água e esgoto tem taxação fixa (*R*$ 305.55 água; *R*$ 226.05 esgoto) e variável, conforme apresentado na tabela abaixo.

Volume água (×1000 (litros)	Tarifa água (R$ / 1000 l)	Tarifa esgoto (R$ / 1000 l)
0 - 75	2.11	1.56
75 - 150	4.496	3.327
150 - 225	6.968	5.156
225 - ∞	9.512	7.039

Exemplo de fatura. Considere que uma edificação consumiu 120000 litros. Então a composição da fatura será :

Faixa (k litros) (x 1000 litros)	Consumido na faixa (x 1000 litros)	Tarifa água (R$)	Tarifa esgoto (R$)
		305,55	226,05
0 - 75	75	158,25	117,00
75 -150	45	202,32	149,72
150 - 225		0,00	0,00
225 - inf		0,00	0,00
		666,12	492,77
Fatura total	**1158,89**		

Escreva um código em C que leia a quantidade de litros consumida e imprima o valor **Fatura total**.

Exercício 45

Construir um código em C que leia um valor de salário bruto e calcule o valor líquido após de dedução do INSS e imposto de renda.

Nota: Há necessidade de consulta de tabelas de INSS e imposto de renda para obter alíquotas corretas. Tais tabelas são de livre acesso pela Internet.

Exercício 46

Para se determinar o número de lâmpadas necessárias para cada cômodo em uma residência, existem experimentos empíricos que dão o mínimo de potência W de iluminação LED por m^2, conforme o tipo de utilização do cômodo dado na tabela:

Cômodo	Classe	Potência / m^2
quarto	1	3
sala TV	1	3
sala jantar	1	3
varanda	4	2
escritório	3	7
banheiro	4	2

a) leia um cômodo e sua área;

b) classifique o cômodo;

c) calcule o número mínimo de lâmpadas necessárias para deixar o cômodo com a claridade sugerida na tabela apresentada.

d) imprima: o nome do cômodo, sua classe e o número (inteiro) de lâmpadas.

12 Estruturas de Repetição - Exercícios de fixação N°04

Exercício 47

Elabore um programa em C que inicialize uma variável inteira com o valor 0 para, então, incrementar seu valor de 5 em 5, até que passe a valer 100. Para isso, utilize a estrutura de repetição: a) ***for***; b) ***while***; c) ***do while***.

Exercício 48

Elabore um código em C que receba um número inteiro qualquer e imprima todos os números ímpares de 1 até esse número.

Exercício 49

Crie um programa em C que receba um número inteiro de 1 a 10 e, então, imprima sua tabuada na tela. Utilize estruturas de repetição.

Exercício 50

Desenvolva um programa em C que calcule e imprima o resultado da seguinte soma: $1^{20} + 2^{19} + 3^{18} + ... + 19^2 + 20^1$.

(Nota: Para inteiros com mais de 10 casas, é recomendado que se declare a variável como ***double***)

Exercício 51

Crie um programa em C que calcule e imprima o resultado de:

a) $\sum_{i=4}^{10} 4i - 5$ b) $\sum_{i=1}^{8} (-10)^i$ c) $\prod_{i=4}^{15} 3i - 20$ $\prod_{i=1}^{7} i^3 - 9$

Exercício 52

Beatriz possui 37 anos e é mãe de Clarisse de apenas 7 anos. Desenvolva um programa em C que calcule quantos anos terá de Clarisse quando sua mãe tiver o dobro de sua idade.

Exercício 53

Um determinado laboratório estudava o crescimento de duas colônias de bactérias. A colônia A possuía 10.000 indivíduos e crescia a uma taxa de 50% por hora, enquanto a colônia B consistia de 30.000 indivíduos e aumentava a uma taxa de 40% por hora. Desenvolva um programa em C que calcule quantas horas serão necessárias para que a quantidade de indivíduos da colônia A ultrapasse a da B.

(Nota: Considere a fórmula juros compostos $M = C(1+i)^t$, sendo M, C, i e t, o montante, o capital inicial, a taxa de juros e o período de tempo respectivamente.)

Exercício 54

Na linguagem C, o ***rand***() é uma função da biblioteca ***time.h*** que permite gerar um número pseudo-aleatório entre 0 e um limite determinado. Sua sintaxe consiste em: ***rand***()%x. Sendo x o limite do intervalo no qual será gerado um número aleatório. Considere as seguintes observações:

- O valor máximo assumido pelo número aleatório é $x - 1$. Portanto, caso queira gerar um número de 0 a 100, deve utilizar ***rand***() % 101.
- Ao utilizar a função ***rand***(), os números aleatórios gerados são sempre os mesmos quando se roda novamente o programa. Para resolver isso, inclua a seguinte linha em seu código: $srand(time(NULL));$

Crie um programa em C que gere e imprima 10 números aleatórios de 0 a 1000.

13 Estruturas de repetição - Exercícios complementares N°04

Exercício 55

Interprete o código abaixo e aponte sua função:

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

int main() {
    int num;
    printf("Entre com o numero desejado:\n");
    scanf("%d", &num);
    for(int i = 1; i <= num; i++){
            if(num % i == 0){
                    printf("\n%d", i);
            }
    }
    return 0;
}
```

Exercício 56

Elabore um programa em C que receba um número inteiro qualquer e imprima seu fatorial. Para isso, utilize a estrutura de repetição: a) ***for***; b) ***while***; c) ***do while***.

Exercício 57

Numeração reservada; exercício ainda em elaboração.

Exercício 58

Numeração reservada; exercício ainda em elaboração.

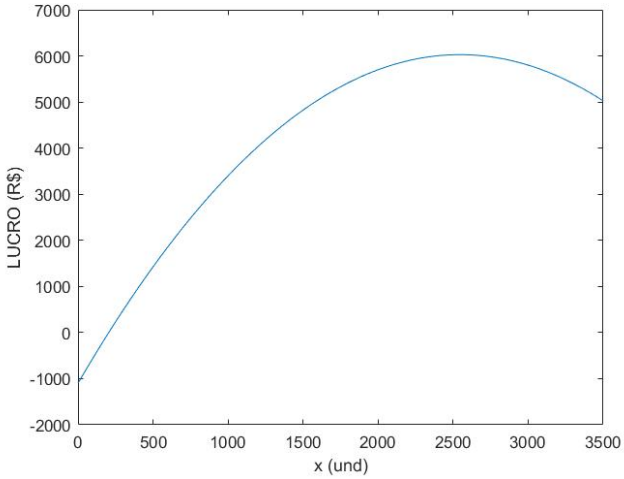
Exercício 59

Numeração reservada; exercício ainda em elaboração.

14 Estruturas de repetição - Exercícios de aplicação N°04

Exercício 60

Um publicitário resolveu fazer canetas personalizadas com estampa de times de futebol para vender. Se nenhuma caneta for vendida ele terá prejuízo de R$ 1100.00, por causa dos custos de matéria prima. Ele planeja criar preços promocionais (política de descontos) que podem acelerar as vendas. Entretanto, há um limite de descontos que deixa o publicitário atingir ou quase atingir o ponto de equilíbrio, ou seja, nem lucro nem prejuízo. O modelo matemático e a representação gráfica estão apresentados abaixo.



$$L = -0.0011x^2 + 5.6x - 1100$$

Desenvolva código em C ou C++, deve responder, qual a quantidade x que gera um dado lucro L. Personalize esse código para descobrir: a) o valor de x_{max} que gera o lucro máximo L_{max}, e b) qual a quantidade mínima x_{min} de venda para que não ocorra prejuízo.

Nota: Há diferentes formas de encontrar L_{max}, como por exemplo o método de busca unidirecional Golden Section e bem como por fórmula $L_{max} = -\frac{\Delta}{4a}$.

Exercício 61

Nos tempos da Grécia antiga, as catapultas eram máquinas de guerra construídas de madeira que utilizavam um recipiente em formato de concha para lançar objetos a uma grande distância, ultrapassando obstáculos como muralhas e fossos. Pede-se construir um programa em C ou C++ para possibilitar a simulação do desempenho de uma catapulta a respeito do impacto do projétil frente um anteparo (muralha, montanha, muro, etc). No nosso modelo de catapulta, consideramos:

a) o estrago no anteparo e, é descrito em função da área do projétil a e da velocidade v dele no momento do impacto como $e = a^2v$;

b) a velocidade no momento do impacto é medida por $v^2 = v_o^2 + 2gdk$, sendo v_o a velocidade inicial, k uma constante de frenagem pelo atrito e g a aceleração da gravidade;

c) a constante k não tem unidade e, neste problema, pode ser aproximado como $k = (a + 0.2)^2$.

d) os projéteis são esféricos, formados rocha talhada com raio r variando de 0,12 m a 0.22 m;

e) a gravidade $g = -10m/s^2$, a distância da catapulta ao anteparo $d = 30m$, $v_o = 20m/s$, a área de uma esfera $4\pi r^2$ e o ângulo de lançamento em 45 graus constante ao longo de todas as simulações.

O programa lê os raios de projéteis r de arquivo, sendo um por linha, até que não haja mais dado a ser lido. O programa deve apresentar qual valor do raio que provocou maior estrago.

15 Prova de AEDs I - etapa 1 - valor 100% pontos

Questão 1: (30% pontos)

Considerando edificações residenciais, a fatura da companhia de água e esgoto tem taxação fixa (*R*$ 305.55 água; *R*$ 226.05 esgoto) e variável, conforme apresentado na tabela abaixo.

Volume água (×1000 (litros)	Tarifa água (R$ / 1000 l)	Tarifa esgoto (R$ / 1000 l)
0 - 75	2.11	1.56
75 - 150	4.496	3.327
150 - 225	6.968	5.156
225 - ∞	9.512	7.039

Exemplo de fatura. Considere que uma edificação consumiu 120000 litros. Então a composição da fatura será :

Faixa (k litros) (x 1000 litros)	Consumido na faixa (x 1000 litros)	Tarifa água (R$)	Tarifa esgoto (R$)
		305,55	226,05
0 - 75	75	158,25	117,00
75 -150	45	202,32	149,72
150 - 225		0,00	0,00
225 - inf		0,00	0,00
		666,12	492,77
Fatura total	**1158,89**		

Escreva um código em C que leia a quantidade de litros consumida e imprima o valor **Fatura total**.

Questão 2: (15% pontos)

Escreva algoritmo para ler n e calcular a soma *soma* dos n primeiros termos, conforme expressão:

$$soma = 1^2 - 3^2 + 5^2 - 7^2 \ldots$$

Questão 3: (15% pontos)

Para compreender o algoritmo ao lado sugere-se utilizar uma ou mais entradas na linha N do quadro abaixo. Bem explicado, cada execução representa o comando "iniciar o programa".

Execução ☞	1	2	3	4	5	6
N	17	10	20	25	37	38
Impressão	?	?	?	?	?	?

Qual a finalidade do algoritmo? Apresente o resultado para $N = 10$.

```
1: Leia (N)
2: x ← 1;
3: repita
4:     r ← Resto(N, x);
5:     se(r==0) então
6:         Imprima (x)
7:     fim se
8:     x ← x + 1
9: até(x > N)
```

Questão 4: (15% pontos)

Explique por que, na construção de um algoritmo, devem ser observados itens como MODULARIZAÇÃO, FLEXIBILIDADE e DESENVOLVIMENTO TOP-DOWN. (Obs.: cada um desses itens devem ser explicados)

Questão 5: 25% pontos

Escreva um programa em C que solicita ao usuário a dimensão n e vetor $\mathbf{x}$ para então calcular $f(\mathbf{x})$, conforme expressão:

$$f(\mathbf{x}) = 10n + \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i)) \text{ , sendo } -5.12 \leq x_i \leq 5.12$$

16 Modularidade - Exercícios de fixação N°05

Exercício 62

Interprete o código abaixo e aponte sua função.

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

double funcao(double a, double b) {
    double resultado;
    resultado = pow(a, b);
    return resultado;
}

int main(){

    double num1, num2, result;

    printf("entre com o primeiro numero:");
    scanf("%lf", &num1);

    printf("entre com o segundo numero:");
    scanf("%lf", &num2);

    result = funcao(num1, num2);

    printf("\nO resultado e: %lf", result);

    return 0;
}
```

Exercício 63

Desenvolva um programa em C que receba dois números quaisquer, chame uma função "soma" que receba ambos como parâmetros, faça sua soma e retorne o resultado a ser impresso na tela.

Exercício 64

Escreva um código em C que receba 3 números, chame uma função para calcular a sua média e imprima o resultado.

Exercício 65

Crie um programa em C que, para calcular a área de um triângulo retângulo, chame uma função externa ao programa principal, recebendo os parâmetros *Base* e *Altura*. Por fim, imprima o resultado do cálculo.

Exercício 66

Elabore um código em C que receba as notas de um aluno em 3 provas diferentes e chame uma função para calcular sua nota final, dado que a nota da segunda prova tem peso "2" e a nota da terceira tem peso "3".

Exercício 67

Desenvolva um programa em C que receba o valor de um ângulo em graus, chame uma função que converta esse valor para radianos e imprima o resultado.

(Nota: Utilize a relação: $180° = \pi rad$.)

17 Modularidade - Exercícios complementares N°05

Exercício 68

Interprete o código abaixo e aponte sua função.

```c
#include <stdio.h>

void encontrar_m(float x[100], int total)
{
    float num_max = x[0];

    for(int i = 0; i < total; i++)
    {
        if(x[i] > num_max)
        {
            num_max = x[i];
        }
    }

    printf("\n\n%f", num_max);
}

int main(){
    int total;
    float numeros[100];

    printf("Entre com o numero de elementos a serem comparados: \n");
    scanf("%d", &total);
    printf("\nEntre com os elementos a serem comparados:\n");

    for(int i = 0; i < total; i++)
    {
        scanf("%f", &numeros[i]);
    }

    encontrar_m(numeros, total);
    return 0;
}
```

Exercício 69

Crie um programa em C que receba uma sequência de **n** números e chame uma função para organizá-los e imprimi-los em ordem crescente.

Exercício 70

Desenvolva um programa em C que receba uma quantidade de tempo em dias e chame 3 funções distintas. A primeira para calcular o valor dessa quantidade de tempo em horas, a segunda em minutos e a terceira em segundos. Imprima os resultados.

Exercício 71

Elabore um código em C que receba um valor inteiro qualquer e chame 3 funções distintas. A primeira retorna "TRUE" caso o número seja divisível por "7", A segunda retorna "TRUE" caso o número seja par e a terceira retorna "TRUE" caso o número esteja entre 38 e 433. Imprima os resultados.

Exercício 72

Numeração reservada; exercício ainda em elaboração.

18 Modularidade - Exercícios de aplicação N°05

Fundamentação 6: Algoritmos de otimização

Algoritmos de otimização tem a finalidade de encontrar um ou mais vetores $\boldsymbol{x}$, dentro do domínio viável, que resultem em valores ótimos da função objetivo $f(\boldsymbol{x})$. Por exemplo, quando minimizamos, a solução do problema é o valor do vetor $\boldsymbol{x}$ que deixa $f(\boldsymbol{x})$ menor possível. Outras notações para vetor $\boldsymbol{x}$: x_i, $i = 1 \ldots n$ e também $[x_1, x_2, \ldots, x_n]$. Maximização é conceituado analogamente. Então, uma das áreas de Ciência da Computação é construir algoritmos eficientes e eficazes para resolver problemas de otimização (maximização ou minimização). Entretanto, há diversos problemas de otimização no mundo real que não é possível garantir que a resposta encontrada é a solução ótima. Por outro lado, graças a baterias de testes, podemos investigar sobre a eficiência, a eficácia e o desempenho computacional do algoritmo no enfrentamento de funções teste; se o algoritmo foi testado ele se torna mais confiável ou menos confiável, por exemplo.

Fundamentação 7: Exemplo Função Teste

Para exemplificar o entendimento de funções teste, considere o caso a seguir:

Expressão algébrica	Domínio nome da função	Dim. n	Valor ótimo vetor solução
$\min f(\boldsymbol{x}) = \sum_{i=1}^{3} x_i^2$	$-100 \leq x_i \leq 100$ sphere (quadratic)	3	0 [0,0,0]

Interpretação da tabela:

- o problema é de minimização devido ao termo min e maximização seria max;
- o número de dimensões do problema é $n = 3$, logo um possível $\boldsymbol{x}$ é $\boldsymbol{x} = [-33.23, 4.12, 100]$;
- a expressão $f(\boldsymbol{x}) = \sum_{i=1}^{3} x_i^2$ é equivalente a $f(\boldsymbol{x}) = x_1^2 + x_2^2 + x_3^2$;
- o domínio é $-100 \leq x_i \leq 100$, logo $\boldsymbol{x} = [-33.2, 4.1, 100]$ é viável e $\boldsymbol{x} = [-105, 14.1, 99]$ é inviável;
- a avaliação da função se dá da seguinte forma: $\boldsymbol{x} = [2, -5, 10]$, então $f(\boldsymbol{x}) = 129$;
- o mínimo ocorre em $\boldsymbol{x} = [0, 0, 0]$, quando $f(\boldsymbol{x}) = 0$.

Exercício 73

Os exercícios desta lista consideram o desenvolvimento de uma biblioteca de funções teste, que temos a intenção de deixar modular. Desenvolva algoritmos para computar as funções testes se seguem. Para simplificar a tabela, todas as funções estão no formato de minimização.

Expressão algébrica	Domínio nome da função	Dim. n	Valor ótimo vetor solução
$f_1(\mathbf{x}) = \sum_{i=1}^{n} x_i^2$	$-100 \leq x_i \leq 100$ Sphere (quadratic)	3	0 [0,0,0]
$f_2(\mathbf{x}) = \sum_{i=1}^{n-1} [100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2]$	$-2.048 \leq x_i \leq 2.048$ Rosenbrock	2	0 [1,1]
$f_3(\mathbf{x}) = \sum_{i=1}^{n} inteiro(x_i)$	$-5.12 \leq x_i \leq 5.12$ Degrau - De Jong	5	0 [0,0,0,0,0]
$f_4(\mathbf{x}) = \sum_{i=1}^{n} i x_i^4 + Gauss(0,1)$	$-1.28 \leq x_i \leq 1.28$ Gauss - De Jong	30	0 [0, ...,0]
$f_5(\mathbf{x}) = 0.002 + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ji})^6}$	$-65.536 \leq x_i \leq 65.536$ Shekel's Foxholes	2	0.9980038 [-31.9784576, -31.9786271]
$f_6(\mathbf{x}) = (x_1^2 + x_2^2)^{0.25} [\sin^2 (50(x_1^2 + x_2^2)^{0.1}) + 1.0]$	$-100 \leq x_i \leq 100$ Schaffer	2	0 [0,0]
$f_7(\mathbf{x}) = (x_1^2 + x_2^2)/2 - \cos(20\pi x_1)\cos(20\pi x_2) + 2$	$-10 \leq x_i \leq 10$	2	1 [0,0]
$f_8(\mathbf{x}) = 10n + \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i))$	$-5.12 \leq x_i \leq 5.12$ Rastrigin	20	0 [0, ...,0]

19 Termos técnicos da etapa

Fundamentação 8: Estrutura de dados e testes - parte 2

Ao iniciar um novo projeto de software, é comum começar definindo os *protótipos de funções*, permitindo ao programador uma visão global do que o projeto final. Através do processo de *dividir para conquistar*, fragmentam o problema em partes gerenciáveis, promovendo a *modularidade* - um princípio fundamental que facilita a manutenção e atualizações futuras.

Em programação, a busca pela precisão e eficiência é recorrente. Desenvolvedores de software, usam frequentemente *testes caixa preta* e *testes caixa branca*, como métodos de teste de software que verificam as funcionalidades e qualidade da precisão das soluções que produzem. Se pudermos dividir um sistema, as *unidades* seriam os componentes onde pode-se avaliar isoladamente o comportamento. Assim, à medida que os programas evoluem, os *testes de unidade* tornam-se vitais, pois garantem o funcionamento correto antes de prosseguir para o *teste de integração*, onde diferentes unidades são combinadas e testadas como um grupo. Concomitantemente, o *teste de regressão* assegura que as modificações recentes não introduziram novos erros em áreas já testadas.

À medida que o projeto avança, *armazenamento em arquivos* torna-se uma necessidade, permitindo que dados sejam armazenados de forma permanente e recuperáveis para futuras consultas e operações. Às vezes, pode-se encontrar uma *variável heterogênia*, uma entidade que pode armazenar múltiplos tipos de dados, proporcionando flexibilidade no manejo de informações complexas.

Uma técnica elegante e às vezes complexa, conhecida como *recursividade*, permite que funções chamem a si mesmas. Estruturas de dados como *árvores de decisão* podem ser navegada para encontrar soluções para problemas complexos usando recursividade. Recursividade também é fundamental na navegação e tratamento de *grafos* e *listas encadeadas* oferecem estruturas de dados flexíveis, facilitando a organização e armazenamento de informações em formatos hierárquicos e sequenciais. A gestão eficiente desses recursos é feita através, por exemplo, com uso de *pilhas* e *filas*, estruturas de dados que auxiliam na organização e processamento de informações.

No entanto, o desenvolvimento de software não está isento de desafios. *loops infinitos* podem surgir, muitas vezes devido a uma *variável ponteiro* mal gerenciada, levando a um *uso intensivo de memória*. Esse tipo de falha pode resultar em *overflow*, uma condição onde o espaço de memória alocado é excedido, levando a falhas no sistema. O *tempo de execução* é uma métrica crucial, influenciada pela *ordem de complexidade* do algoritmo, refletindo a quantidade de recursos computacionais necessários.

Programação e desenvolvimento de software é mais que simples criação de códigos. A tarefa de encontrar uma solução inovadora, precisa e eficiente é cada vez mais complexa. Devemos ficar atentos às novas tecnologias e metodologias para acompanhar quem sai na frente.

20 Recursividade - Exercícios de fixação N°06

Exercício 74

Elabore o código de uma função em C que receba **n** como parâmetro e imprima todos os números de **n** até 0. Utilize recursividade.

Exercício 75

Desenvolva um programa em C que receba um número inteiro e chame uma função que calcule seu fatorial. Utilize recursividade.

Exercício 76

Elabore o código de uma função em C que determina quantas vezes um dígito **k** ocorre em um número natural **n**. Utilize recursividade.

Exercício 77

Desenvolva um programa em C que receba *n* valores inteiros do usuário e os armazene em um ***array***. Tais valores serão necessários para calcular a média ponderada do ***array***. O programa também deve receber *n* pesos e armazená-los em outro vetor, com cada um representando o respectivo peso de cada valor, de acordo com sua posição.

Exercício 78

Crie o código de uma função em C que receba um vetor de 20 posições e inverta sua ordem. Utilize recursividade.

Exercício 79

Crie um código em C que receba um vetor de *n* posições do usuário e o organize do menor para o maior, imprimindo o resultado no console. Utilize recursividade.

Exercício 80

Elabore o programa de uma função em C que converta um número decimal em um número binário. Utilize recursividade.

Exercício 81

Desenvolva o programa de uma função em C que calcule a soma dos algarismos de um número inteiro. Utilize recursividade.

Exercício 82

Numeração reservada; exercício ainda em elaboração.

Exercício 83

Numeração reservada; exercício ainda em elaboração.

21 Recursividade - Exercícios complementares N°06

Exercícios Complementares N°6: Recursividade

Exercício 84

Analise atentamente o programa abaixo.

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

float calc (float v_1, float v_2) {
    float result = (v_1 + v_2) / 2;
    return result;
}

int calcMenu () {
    int v1, v2, result;

    printf("------Calculo de media------\n");
    printf("------insira o primeiro valor\n");

    scanf("%d", &v1);

    printf("------insira o segundo valor\n");

    scanf("%d", &v2);

    printf("\n------resultado: %.2f", calc(v1, v2));

    int opcao;

    printf("\n------deseja calcular novamente?\nsim -> 1\nnao -> 2\n");

    scanf("%d", &opcao);

    if (opcao != 1 || opcao != 2){
        if (opcao == 1){
            calcMenu();
        } else {
            printf("fim do programa");
        }
    }
}

int main ()
{
    calcMenu();
}
```

a) Qual o objetivo do programa? b) Como o programa se repete, caso *opcao* seja 1?

Exercício 85

Numeração reservada; exercício ainda em elaboração.

Exercício 86

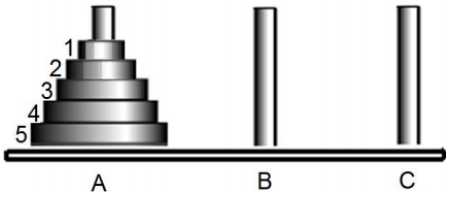
Numeração reservada; exercício ainda em elaboração.

22 Recursividade - Exercícios de aplicação N°06

Exercício 87

O problema da Torre de Hanói consiste em movimentar discos de diâmetros diferentes em três hastes seguindo a regra central de que se mova um disco por vez e um disco não pode ser posicionado em cima de outro de diâmetro menor. Este jogo foi popularizado em 1883 como uma invenção do matemático francês François Édouard Anatole Lucas. Em programação, a torre de Hanói é um clássico no estudo de recursividade, servindo também como um atividade para o desenvolvimento do raciocínio.

Nesta tarefa, pede-se estudar características de repetição sobre a movimentação das peças, mostrar que é possível mover todos os discos para uma nova haste e que a quantidade mínima de movimentos necessários para isso é $2^n - 1$. Validar a quantidade mínima de movimentos através de um programa em C.



Exercício 88

Um sorteio proporcional pode ser considerado uma forma justa de distribuir prêmios ou oportunidades entre os participantes, levando em consideração sua proporção de participação ou contribuição. Considere um sorteio com n prêmios em uma festa de confraternização entre m acionistas, sendo cada acionista com percentual p de ganhar, tal que, $\sum p_i = 1$, sendo i um índice que diferencia os acionistas. Utilizando linguagem C e laços recursivos, apresentar os n acionistas ganhadores. Para testar, use a tabela a seguir.

acionista i	1	2	3	4	5	6
número de ações	1	3	7	17	23	29

(Nota: cada acionista pode ganhar um ou mais prêmios.)

Exercício 89

Utiliza-se o número neperiano $e = 2.718282$ em cálculos de problemas de física e engenharia.

Escreva um programa em C ou C++ para calcular e com precisão mínima de 0.00001 e escreva na tela no número de iterações mínimo para atingir a precisão solicitada.

$$e = \sum_{i=0}^{n} \frac{1}{n!}$$

Utilizando recursividade, escreva uma função que calcule o fatorial de n em C ou C++.

$$n! = n \times (n-1) \times (n-2) \times (n-3) \times \ldots \times 2 \times 1, \qquad 0! = 1.$$

Exercício 90

Método bolha ordenação

Numeração reservada; exercício ainda em elaboração.

23 Ponteiros - Exercícios de fixação N°07

Exercício 91

Interprete o código abaixo e responda qual será a saída de cada ***printf***

```c
#include <stdio.h>

void main(void) {
    int m=10;
    int *z=&m;

    printf("%p\n", z);
    printf("%d\n", *z);
    printf("%p\n", &m);
    printf("%p\n", &z);
}
```

Exercício 92

Utilize ponteiros para desenvolver um programa em C que:

a) determine o tamanho de uma string.

b) ponha em ordem crescente os elementos de um ***array***.

c) inverta a ordem dos elementos de um ***array***.

Exercício 93

Crie um programa em C que receba 2 números pela entrada de usuário e chame uma função que, utilizando ponteiros, troque os valores das duas variáveis.

Exercício 94

Elabore o código de uma função em C que recebe como parâmetro um vetor de **n** posições e, então, imprime o maior número e a quantidade de vezes em que ele ocorre. utilize ponteiros.

Exercício 95

Crie um programa em C que gere 100 números aleatórios, os armazene em um vetor de 100 posições e imprima os ímpares com suas respectivas posições no vetor. Utilize ponteiros.

Exercício 96

Desenvolva um programa em C que receba dois ***arrays*** de **n** posições e os some, gerando um terceiro ***array***, também de **n** posições, que deve ser impresso na tela. Utilize ponteiros.

Exercício 97

Elabore um código em C que receba dois valores inteiros: **m** e **n**. O programa, então, deve criar um vetor de **m** posições, preenchê-lo com números aleatórios e imprimi-lo na tela com **n** elementos por linha. Utilize ponteiros.
(Nota: Caso **m < n** imprima “erro”.)

24 Ponteiros - Exercícios complementares N°07

Exercício 98

Interprete o código abaixo e responda qual será sua saída.

```c
#include <stdio.h>

void f1 (int a, int *b, int *c) {
    *b += a;
    a++;
    *c = a + *b;
    b = &a;
}

int main() {
    int x = 2;
    int y = 3;
    int z = -1;

    while (y < 10)
    {
        f1(x, &y, &z);
        printf("\n%d %d %d", x, y, z);
    }

    return 0;
}
```

Exercício 99

Utilize ponteiros para desenvolver um programa em C que:

a) percorra um vetor de **n** posições, imprimindo-o na tela.

b) crie uma variável inteira e imprima seu valor.

c) crie uma variável inteira e imprima seu endereço.

Exercício 100

Desenvolva um programa em C que receba duas variáveis inteiras, compare seus endereços e exiba o conteúdo do maior endereço. Utilize ponteiros.

Exercício 101

Numeração reservada; exercício ainda em elaboração.

25 Ponteiros - Exercícios de aplicação N°07

Fundamentação 9: Algoritmos força bruta

Em ciência da computação, o termo "força bruta" está relacionado a algoritmos que implementam métodos diretos de resolução de problemas por meio de testar todas as possibilidades. Um exemplo tradicional para força bruta é o cadeado de segredo com 3 algarismos. Se a combinação é desconhecida, é necessário um método de força bruta para abrir a fechadura sem repetir combinações.

Fundamentação 10: Otimização combinatória e permutações

A otimização combinatória busca as melhores soluções dentre um conjunto finito de possibilidades que podem ser obtidos por permutação da ordem de ocorrência das variáveis. Assim, o resultado desejado dos algoritmos de otimização é um ou mais arranjos com os elementos organizados em uma ordem específica. Por exemplo, em problemas de sequenciamento, como o problema do caixeiro viajante, as cidades a serem visitadas podem ser representadas como um arranjo. A otimização combinatória tem aplicações em diversas áreas, como logística, planejamento de produção, escala de horários e layout de posição de equipamentos. Os arranjos são fundamentais para modelar e resolver problemas nessas áreas.

Exercício 102

Usando ponteiros e recursividade, escreva um programas em C para imprimir todas as permutações de um vetor *string*. Faça adequações no programa para que a ordenação seja feita um vetor de inteiros.

Um resultado esperado para *string* "abcd´´: abcd abdc acbd acdb adcb adbc bacd badc bcad bcda bdca bdac cbad cbda cabd cadb cdab cdba dbca dbac dcba dcab dacb dabc

Fundamentação 11: O problema do caixeiro viajante

O problema do caixeiro viajante ou TSP (travelling salesman problem) é um problema NP-difícil em otimização combinatória, importante *benchmark* em ciência da computação teórica e pesquisa operacional, e pode ser enunciado como: "Dada uma lista de cidades e as distâncias entre cada par de cidades, qual é a rota mais curta possível que visita cada cidade exatamente uma vez e retorna à cidade de origem?"

Exercício 103

Considere um caixeiro viajante percorre n cidades e registra a distância entre elas, na sequência de visitação em vetor com n posições. Escreva um programa em C ou C++ que compute a distância total percorrida e apresente o valor do maior trecho entre cidades. Considere a leitura pelo teclado e impressão da distância total na tela. É obrigatório uso de ponteiro para percorrer o vetor e acessar os dados. As distâncias são valores inteiros.

Exercício 104

Elabore um algoritmo força bruta para resolver o problema do caixeiro viajante. Para isso considere que sejam n cidades e as distâncias entre elas seja dada em uma matriz bidimensional. Imprima na tela o resultado da sequência de cidades que minimiza o percurso total com retorno a cidade de origem. Para testar faça testes com $n = 3$, 4, 5 *e* 6.

26 Arquivos - Exercícios de fixação N°08

Exercício 105

Crie um programa em C que abra um arquivo de texto qualquer e imprima na tela “sucesso”, caso o arquivo seja aberto com sucesso ou “erro”, caso haja algum erro na abertura do arquivo.

Exercício 106

Desenvolva um programa em C que:

a) Abra um arquivo de texto com o nome “arquivo.txt”.

b) Permita que o usuário entre com uma string de até 100 caracteres para ser armazenada no arquivo.

c) feche o arquivo.

Exercício 107

Existem diversas maneiras de ler o conteúdo de um arquivo para imprimi-lo na tela. Uma possível forma de fazer isso seria utilizando a seguinte estrutura:

```
char linha[100];

while (fgets(linha, 100, arquivo) != NULL)
{
    printf("%s", linha);
}
```

Com base nessa informação, desenvolva um código em C que leia todas as linhas de um arquivo e as imprima na tela. Utilize o método de sua preferência.

(Nota: Lembre-se, primeiramente, de abrir o arquivo em modo de leitura.)

Exercício 108

Crie um programa em C que leia todas as linhas de um arquivo e imprima na tela quantos números pares existem nesse nele.

Exercício 109

Crie um programa em C que leia todas as linhas de um arquivo e imprima na tela quantas vogais existem nesse nele.

Exercício 110

Elabore um código em C que receba o raio de uma esfera, calcule sua área e volume e, então, imprima esses dados em um arquivo de texto.

Exercício 111

Desenvolva o programa de uma função em C que abra um arquivo em modo de escrita e nele imprima uma determinada ***string***. A função deve receber como parâmetros:

- Uma ***string*** de até 20 caracteres que será o nome do arquivo a ser criado.
- Uma ***string*** de até 100 caracteres que será o texto a ser impresso no arquivo.

A chamada da função deve ser como a do seguinte exemplo:
nome_da_funcao(“NomeDoArquivo.txt”, “esse texto sera impresso em um arquivo”);

27 Arquivos - Exercícios complementares N°08

Exercício 112

Crie um programa em C que receba um numero inteiro de 1 a 10, calcule sua tabuada e a imprima em um arquivo de texto.

Exercício 113

Uma casa de boliche teve a ideia de fazer um campeonato entre seus membros. Para apresentar o resultado, precisariam fazer uma tabela de classificação com o nome e a pontuação de cada participante. Desenvolva um programa em C que receba quantos membros participaram do evento, o nome de cada um e sua respectiva pontuação para, então, armazenar esses dados em um arquivo e imprimir o vencedor na tela.

Exercício 114

Considere um arquivo de texto que apresente informações a cerca de possíveis destinos de viagem a carro. O arquivo possui **n** linhas, cada uma contendo os seguintes dados: "nome_do_destino", "velocidade_maxima" e "distancia". Desenvolva um programa que leia a totalidade do arquivo,

calcule o tempo de viagem para cada caso e imprima-o ao lado de seu respectivo destino. Assuma que o carro sempre mantenha a velocidade máxima permitida no trajeto.

Exercício 115

Numeração reservada; exercício ainda em elaboração.

Exercício 116

Numeração reservada; exercício ainda em elaboração.

28 Arquivos - Exercícios de aplicação N°08

Exercício 117

Considere um modelo, no formato de expressão matemática, para avaliação a qualidade da distribuição de policiais nas n regiões de uma cidade. Neste modelo, é necessário considerar tanto o número de policiais disponíveis p como o índice de segurança em cada região i, como mostrado a seguir:

$$f_i = (\frac{x_i}{is_i} - \frac{\overline{x}}{\overline{is}}), \qquad f = \sum_{i=1}^{n} f_i, \qquad p = \sum_{i=1}^{n} x_i \ ,$$

sendo:

- x_i o número de policiais na região i;
- is_i índice de segurança na região i, variando de 1 (mais segura) a 5 (menos segura);
- $\overline{x}$ média do número de policiais em relação ao número de regiões;
- $\overline{is}$ média do índice de segurança.

Pede-se:

a) Elaborar programa em C para calcular f, sendo as informações paramétricas são lidas de arquivo, armazenadas em matrizes e f deve ser armazenada em vetor.

b) Elaborar código para gerar banco de dados aleatório de testes para testar (a). O banco de dados de teste deve ser escrito em arquivo, e mantenha fixo $n = 4$ e $p = 1200$.

Fundamentação 12: Testes de software

O teste de software é uma das etapas do processo do desenvolvimento de software que se relaciona diretamente a validação da qualidade do mesmo. O objetivo é de fato, através de baterias de testes, verificar se o software funciona, é eficiente, eficaz e se entrega as saídas esperadas.

Há vários tipos de teste, dentre os quais citam-se os testes de unidade, de integração, de regressão e de aceitação. Esses testes podem ser classificados como caixa preta, caixa cinza e caixa branca. Testes de unidade, por exemplo, são usualmente caixa branca. Enquanto testes caixa preta confrontam saídas do sistema com saídas esperadas, os testes caixa branca fazem análises a linhas de comando para encontrar as falhas.

Testar software considera planejamento para tratar a etapa como um experimento, conhecimento de estatística para validar os resultados, atenção e dedicação ser eficaz.

Exercício 118

Considerando as funções teste Sphere e Rastrigin desenvolvidas na Lista de Exercícios Modularidade, leia os arquivos os parâmetros de entrada sphere.txt e rastrigin.tex para validar as funções implementadas.

Nota: Utilizou-se linhas duplas como entrada nos arquivos sphere.txt e rastrigin.tex. Bem explicado, a primeira linha contem o vetor de dados $\mathbf{x}$ e a próxima linha a função $f(\mathbf{x})$.

Exercício 119

Numeração reservada; exercício ainda em elaboração.

29 Vetores e matrizes - Exercícios de fixação N°09

Exercício 120

Matrizes são tabelas organizadas em linhas e colunas no formato *i* x *j*, onde *i* representa o número de linhas (horizontal) e *j* o número de colunas (vertical).

$$\mathbf{A_{3\times5}} = \begin{array}{c} \begin{matrix} col.0 & col.1 & col.2 & col.3 & col.4 \end{matrix} \\ \begin{bmatrix} 0 & 9 & 8 & 0 & 7 \\ 3 & 5 & 6 & 5 & 1 \\ 1 & 0 & 0 & 3 & 2 \end{bmatrix} \end{array} \begin{matrix} lin.0 \\ lin.1 \\ lin.2 \end{matrix}$$

Crie um programa em c que receba 4 números e os armazene em uma matriz 2x2. Por fim, o programa deve imprimir tais números na ordem padrão das matrizes: linhas e colunas, respectivamente.

Exercício 121

Crie um programa em C que gere 100 números aleatórios, os armazene em um vetor de 100 posições e imprima o maior deles, juntamente a sua posição no vetor.

Exercício 122

Desenvolva um programa em C que receba uma sequência de *n* números reais e os imprima em ordem crescente.

Exercício 123

Elabore um código em C que receba um vetor inteiro de 50 posições e:

a) Imprima a quantidade de números ímpares, os números ímpares e suas respectivas posições.

b) Imprima a quantidade de múltiplos de 5, os múltiplos de 5 e suas respectivas posições.

c) Elimine os números repetidos, imprimindo cada número do vetor apenas uma vez.

d) Crie um novo vetor de 20 posições composto pelos últimos 20 elementos pares do vetor de 50 posições.

Exercício 124

Desenvolva um programa em C que receba 9 números, os armazene em uma matriz 3x3, some todos os valores contidos em cada linha e coluna (assim resultando em 6 números) e os armazene em um novo vetor, que deve ser impresso posteriormente.

Exercício 125

Elabore um código em C que receba dois números inteiros **i** e **i** para, então, criar uma matriz de **i** linhas e **j** colunas utilizando números aleatórios.

30 Vetores e matrizes - Exercícios complementares N°09

Exercício 126

Interprete o código abaixo e aponte qual será sua saída quando a entrada for "9"

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

int main()
{
    int numeros[3];

    scanf("%d", &numeros[0]);

    numeros[1] = 5;
    numeros[2] = numeros[1] + numeros[0];
    numeros[3] = numeros[2] * 2;

    printf("%d", numeros[3]);

    return 0;
}
```

Exercício 127

Desenvolva um programa em C que receba uma sequência de 5 (valor fixo) números reais e os imprima em ordem crescente.

Exercício 128

Crie um programa em C que receba uma matriz 3x3 e:

a) Calcule e imprima sua transposta;

b) Some a outra matriz 3x3;

c) Multiplique por outra matriz 3x3;

d) Calcule a soma da soma da primeira linha com a soma da segunda.

Exercício 129

Desenvolva um programa em C que leia uma matriz de 5x4 contendo as seguintes informações sobre 5 alunos de uma disciplina:

- Primeira coluna: números de matrícula;
- Segunda coluna: média das provas;
- Terceira coluna: média dos trabalhos;

A quarta coluna é composta pela nota final de cada aluno e deve ser calculada através da soma da média das provas com a média do trabalhos. Por fim, imprima a matricula do aluno com a maior nota final e a media aritmética das notas do 5 alunos.

31 Vetores e matrizes - Exercícios de aplicação N°09

Exercício 130

Saaty (1987) [3] apresentou o método de apoio a tomada de decisão AHP (Analytic Hierarchy Process) largamente utilizado atualmente. AHP utiliza manipulação de pessos organizados em matrizes que relacionam alternativas e critérios para apontar a solução do problema. Dentre as operações sobre as matrizes em AHP destacam-se a normalização de colunas e médias de elementos de linhas. Considerando essas operações para executar AHP, pede-se elaborar funções em C que:

a) receba uma matriz A_{nm} (dimensão $n \times m$), calcule a normalização $\bar{a}_{ij}$ dos elementos a_{ij}:

$$\bar{a}_{ij} = \frac{a_{ij}}{\sum_{i=1}^{n} a_{ij}}$$

b) calcule a média de cada linha da matriz normalizada $\bar{A}_{nm}$ resultando no vetor $\bar{A}_n$:

$$\bar{a}_i = \frac{\sum_{j=1}^{m} \bar{a}_{ij}}{m}, \qquad \bar{A}_n = [\bar{a}_1, \bar{a}_2, \ldots, \bar{a}_n]$$

Elabore um programa em C para testar as funções (a) e (b).

Exercício 131

Fonte: Soares (2008) [4]. Consider a target T traveling in the 2D space. This target can be represented by the following state vector:

$$\mathbf{X}_n = [x_n, y_n, \dot{x}_n, \dot{y}_n]^t \tag{1}$$

where x_n and y_n are the target's Cartesian coordinates, and $\dot{x}_n$, $\dot{y}_n$ their derivatives with respect to time. The evolution of the target is described by a state evolution equation:

$$\mathbf{X}_{n+1} = f(\mathbf{X}_n) + \mathbf{V}_n \tag{2}$$

where f is a deterministic state evolution function, and $\mathbf{V}_n$ is a noise vector representing unpredictable changes in the target's motion. Denoting the time between the states n and $n+1$ by Δt, a simple approximation for f and $\mathbf{V}_n$ can be:

$$\mathbf{X}_{n+1} = \underbrace{\mathbf{A}\mathbf{X}_n}_{f(\mathbf{X}_n)} + \underbrace{\mathbf{B}\mathbf{N}_n}_{\mathbf{V}_n} \tag{3}$$

where matrices $\mathbf{A}$ and $\mathbf{B}$ are given by:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} \frac{\Delta t^2}{2} & 0 \\ 0 & \frac{\Delta t^2}{2} \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} \tag{4}$$

respectively, and $\mathbf{N}_n$ is a vector of dimensions 2×1 which entries have mean zero and standard deviation σ_x to horizontal axis, and σ_y to vertical axis.

Dados o vetor de estado inicial de um corpo viajando na atmosfera $X_0 = [x_0, y_0, \dot{x}_0, \dot{y}_0]^t$ com desvios padrão σ_x e σ_y, elaborar programa em C para computar e armazenar em arquivo todas as posições do corpo até $\mathbf{X}_{500}$. Construa um gráfico da trajetória com ferramenta a escolha.

32 Prova de AEDs I - etapa 2 - valor 100% pontos

Questão 6: (20% pontos)

AHP (Analytic Hierarchy Process) é um método de apoio a tomada de decisão que utiliza manipulação de dados organizados em matrizes para relacionar pesos entre alternativas e critérios e, então, apontar a solução do problema. Dentre as operações sobre as matrizes destacam-se soma e médias de elementos de linhas e colunas. Considerando essas operações para executar AHP, pede-se:

a elaborar uma função em C que receba uma matriz de dimensão $n \times m$ como parâmetro, calcule a média de cada linha e escreva o resultado em um vetor de dimensão n;

b apresentar a sintaxe da chamada da função (a) no programa principal.

Questão 7: (20% pontos)

Escreva um código em C tal que, dado um vetor com n elementos numéricos reais positivos lidos em arquivo, obter a maior diferença entre dois elementos consecutivos neste vetor. Escreva o resultado na tela.

Questão 8: (20% pontos)

Para compreender o algoritmo ao lado sugere-se utilizar as entradas do quadro abaixo.

6	17	10	-5	25	37	20

Qual a finalidade do algoritmo? Apresente o resultado utilizando as entradas.

```
#include <stdio.h>

void main() {

int *a, i, j, tmp, n;
scanf (" %d", &n);
a = (int *)malloc(n * sizeof(int));

for (i=0; i<n; i++)
    scanf(" %d",a+i);

for (i=0; i<n; i++) {
    for (j=i+1;j<n; j++) {
        if (*(a+i)>*(a+j)) {
            tmp = *(a+i);
            *(a+i)=*(a+j);
            *(a+j)=tmp;
}}}

for (i=0;i<n ;i++)
   printf ("%d : %d \n", i+1,*(a+i));

free(a); }
```

Questão 9: (20% pontos)

Faça um programa que realize a leitura dos seguintes dados relativos a um conjunto de alunos: Matricula, Nome, CodigodaDisciplina, Nota1 e Nota2. Considere uma turma de até 10 alunos. Após ler todos os dados digitados, e depois de armazená-los em um vetor tipo *struct*, exibir na tela a listagem final dos alunos com as suas respectivas médias finais.
(Nota: use uma média ponderada, sendo Nota1 com peso=1.0 e Nota2 com peso=2.0).

Questão 10: (20% pontos)

Crie um programa em C que leia do teclado 10 temperaturas de um termômetro e armazene os valores em variável *array*, utilizando manipulação de ponteiros. Sabendo que o termômetro necessita ajuste de dois graus a maior, corrija as temperaturas e grave os resultados em arquivo.

33 Variáveis heterogêneas - Exercícios de fixação N°10

Exercício 132

Desenvolva um programa em C que crie uma ***struct*** com uma variável inteira de nome "teste". Então, no programa principal, crie uma variável do tipo ***struct***, atribua um valor qualquer a "teste" e imprima essa variável na tela.

Exercício 133

Interprete o código abaixo e aponte sua função.

```c
#include <stdio.h>

struct Contatos
{
    char nome[50];
    int numero;
};

int main() {
    int num_contatos;

    printf("entre com o numero de contatos:\n");
    scanf("%d", &num_contatos);

    struct Contatos contato[num_contatos];

    for(int i = 0; i < num_contatos; i++)
    {
        printf("entre com o nome:\n");
        scanf("%s", &contato[i].nome);
        printf("entre com o numero:\n");
        scanf("%s", &contato[i].numero);
    }

    return 0;
}
```

Exercício 134

Elabore um código em C que receba o nome e número de matrícula de um aluno e armazene esses dados em uma ***struct***.

Exercício 135

Escreva um programa em C que leia, de um arquivo, o nome e peso de **n** bois e, então, armazene esses dados em uma ***struct***

Exercício 136

Faça um programa em C que receba os seguintes dados sobre **n** carros: marca, ano e preço. Então, leia um valor "preço máximo" e imprima os dados de todos os carros com preço menor que "preço máximo".

(Nota: Os dados recebidos devem ser armazenados em uma ***struct***.)

34 Variáveis heterogêneas - Exercícios complementares N°10

Exercício 137

Desenvolva um código em C que receba o nome e preço de **n** produtos de um supermercado, armazene esses dados em uma ***struct*** e então, armazene essa ***struct*** em um arquivo.

Exercício 138

Elabore um código em C que receba, de um arquivo, os nomes dos participantes e suas respectivas pontuações em uma competição de corrida. Esses dados devem ser armazenados em uma ***struct*** "Participantes"para, então, serem armazenados em um novo arquivo de texto.

Exercício 139

Desenvolva um programa em C que receba o nome de **n** alunos, bem como suas notas nas provas 1, 2 e 3, de pesos 2, 1.5 e 1, respectivamente. Então, chame uma função para o cálculo da média ponderada das notas de cada aluno, resultando em sua nota final. Caso a nota seja maior que 60, imprima o nome do aluno seguido de "aprovado". Caso contrário, imprima o nome do aluno seguido de "reprovado".

(Nota: Os dados recebidos devem ser armazenados em uma ***struct***. O valor da nota em cada prova deve estar entre 0 e 100.)

Exercício 140

Crie um programa em C que sirva como uma agenda de compromissos. Para isso, ele deve:

- receber **n** ***structs*** com os dados: compromisso (máximo de 50 letras) e data. A data deve ser outra ***struct*** contendo os dados: dia, mês e ano;
- ler dois inteiros e atribui-los às variáveis "M" e "A" para, então, imprimir todos os compromissos do mês "M" do ano "A".

Exercício 141

Numeração reservada; exercício ainda em elaboração.

Exercício 142

Numeração reservada; exercício ainda em elaboração.

Exercício 143

Numeração reservada; exercício ainda em elaboração.

35 Variáveis heterogêneas - Exercícios de aplicação N°10

Exercício 144

Faça um programa que realize a leitura dos seguintes dados relativos a um conjunto de alunos: Matricula, Nome, CodigodaDisciplina, Nota1 e Nota2. Considere uma turma de até 10 alunos. Após ler todos os dados digitados, e depois de armazená-los em um vetor tipo *struct*, exibir na tela a listagem final dos alunos com as suas respectivas médias finais.

(Nota: use uma média ponderada, sendo Nota1 com peso=1.0 e Nota2 com peso=2.0).

Exercício 145

Escreva um programa em C que registre os eventos de uma agenda em arquivo .txt. Os eventos têm entrada pelo teclado e devem ser colocados em ordem temporal. É necessário que cada evento tenha pelo menos os seguintes tipos de dados:

a) horario: hora, minutos e segundos;

b) data: dia, mês e ano;

c) evento: data, horario, local e título do evento.

Exercício 146

Crie um programa em C utilizando variável do tipo registro para representar grupo de projetos de uma empresa. A variável deve conter a código do projeto, nome do projeto, fator de impacto (1 a 5, sendo 5 maior impacto) orçamento ano 1, orçamento ano 2 e orçamento ano 3.

a) Permita ao usuário entrar com os dados de todos os projetos descritos em arquivo.

b) Encontre o projeto com maior orçamento total.

c) Encontre o projeto com maior métrica fator de impacto por orçamento total.

Exercício 147

Faça um programa em C para armazenar um livro de receitas considerando que:

a) o número de receitas pode ser até 10, o nome pode ter no máximo 25 letras, a quantidade de ingredientes no máximo 12 e o processo (um parágrafo);

b) as receitas serão lidas por arquivo, assim como as outras informações;

c) cada ingrediente contém nome e quantidade;

d) seja necessário imprimir na tela uma receita.

Exercício 148

Numeração reservada; exercício ainda em elaboração.

36 Classes em C++ parte A - Exercícios de fixação N°11

Exercício 149

Uma clínica veterinária deseja automatizar seu processo de cadastro de animais. Sabe-se que os animais são documentados por nome, tipo, peso e se possui deficiência. Desenvolva um programa em C++ com a classe *Animais* para realizar tal cadastro, contendo:

a) variáveis privadas *nome* (***string***), *nomeDono* (***string***), *tipo* (***string***), *peso* (***float***) e *def* (***bool***);

b) construtor e destrutor;

c) "getters" e "setters";

d) programa principal.

Exercício 150

Na linguagem C++, o ***getline*** é uma função da biblioteca ***string*** que permite receber uma linha inteira de caracteres (incluindo espaços) e armazená-la em uma variável do tipo ***string***. Sua sintaxe consiste em "***getline***(entrada, variavel)", sendo "entrada" o meio por onde os caracteres serão recebidos e "variavel" a variável do tipo ***string*** que receberá os caracteres.

Considerações:

- O meio de entrada de caracteres pode ser ***cin*** para entrada do teclado ou o nome de uma variável de arquivo, para leitura deste;
- Apesar de terem funções análogas, o ***getline*** se diferencia do ***fgets*** no fato de que não é necessário definir o número máximo de caracteres a serem lidos.

Sabendo disso, desenvolva um programa em C++ que utilize a função ***getline*** para receber uma linha de caracteres através da:

a) Entrada de usuário.

b) Leitura de um arquivo.

Exercício 151

Desenvolver programa com C++ para computar se um boi está acima de peso fixado. Deverá iniciar com uma classe tipo Boi. Para cada cada objeto boi declare os seguintes atributos (private) (1 ponto): nome (string) e peso (double), e com os métodos (public) (10 pontos): 02 construtores (um sem parâmetro com inicialização padrão; o outro com parâmetros para todos os atributos), get/set para cada atributo, método exibe que mostra na saída padrão as informações atuais dos atributos formatadas.
Implemente em C++ os seguintes módulos para manipulação desse objeto:

- (4 pontos) um procedimento que recebe o objeto e o nome do arquivo de entrada, preenchendo o objeto a partir dos dados lidos deste arquivo;
- (2 pontos) uma função que recebe o objeto e retorna 1 (true), caso o boi possua mais de 15 arrobas e, 0 (false), caso contrário.

No programa principal (4 pontos), declare um objeto do tipo Boi, leia o nome do arquivo de entrada da informação e o nome do arquivo de saída, acione o procedimento e a função, gravando no arquivo de saída uma das mensagens conforme o resultado da função ("Boi para venda", "Boi deve voltar para confinamento"). Em algum momento no programa principal as informações do boi são exibidas.

37 Classes em C++ parte A - Exercícios complementares N°11

Exercício 152

Numeração reservada; exercício ainda em elaboração.

Exercício 153

Crie uma classe chamada "Ponto" que representa um ponto no espaço tridimensional. A classe deve ter os seguintes atributos privados: "x", "y" e "z", representando as coordenadas do ponto. Implemente os métodos públicos necessários para definir e obter as coordenadas do ponto (getters e setters), bem como um método para calcular a distância entre dois pontos. Em seguida, crie objetos da classe "Ponto" e demonstre o uso desses métodos.

Exercício 154

Qual a finalidade do algoritmo? Detalhe o funcionamento.

```
#include <iostream>
#include <cmath>
class Montante {
    private:
        double principal, taxa;
        int tempo;
    public:
        Montante(double principal, double taxa, int tempo)
        : principal(principal), taxa(taxa), tempo(tempo) {}
        double calcular() {
            return principal * pow((1 + taxa), tempo);
        }
};
class PopulacaoZumbi {
    private:
        int populacacao_inicial;
        Montante* capitalizacao;
    public:
        PopulacaoZumbi(int populacacao_inicial, double taxa, int tempo) {
            this->populacacao_inicial = populacacao_inicial;
            this->capitalizacao = new Montante(populacacao_inicial, taxa, tempo);}

        ~PopulacaoZumbi() {delete capitalizacao;}

        int calcular_population(int dias, int mortes_por_dia) {
            int populacao_zumbi = (int)this->capitalizacao->calcular();
            if(dias > 10) {
                Montante capitalizacao_equipes(2, 0.02, dias - 10);
                int cont_equipes = (int)capitalizacao_equipes.calcular();
                int mortes = cont_equipes * 7 * mortes_por_dia;
                populacao_zumbi -= mortes;}
            return populacao_zumbi > 0 ? populacao_zumbi : 0;
        }
};

int main() {
    PopulacaoZumbi zombies(50, 0.1, 30);
    int final_population = zombies.calcular_population(30, 20);
    std::cout << "A populacao de zumbis apos 30 dias eh: " << final_population << std::endl;
    return 0;
}
```

38 Classes em C++ parte A - Exercícios de aplicação N°11

Exercício 155

Utilizando linguagem C++, crie uma classe denominada Elevador para armazenar as informações de um elevador dentro de um prédio. A classe deve armazenar o andar atual (sendo térreo o andar número 0), total de andares no prédio (excluindo o térreo), capacidade do elevador em número de pessoas, e quantas pessoas estão presentes nele. A classe deve também disponibilizar os seguintes métodos:

- construtor: que deve receber como parâmetros a capacidade do elevador e o total de andares no prédio (os elevadores sempre começam no térreo e vazios);
- entra: para acrescentar uma pessoa no elevador (só deve acrescentar se ainda houver espaço);
- sai: para remover uma pessoa do elevador (só deve remover se houver alguém dentro dele);
- sobeUm: para subir um andar (não deve subir se já estiver no último andar);
- desceUm: para descer um andar (não deve descer se já estiver no térreo);
- exibir: apresenta quanta pessoas está no elevador no instante atual;
- getters: métodos para obter cada um dos os dados armazenados.

Não há necessidade de criar programa principal, entretanto, considere que as as informações, a partir do andar térreo, de subir e descer e o número de pessoas que saem e entram em cada andar como a tabela:

sobeUm	desceUm	Entra	Sai
true	false	5	0
true	false	3	2
false	true	4	1

Exercício 156

O quadro de distribuição de circuitos (QDC), é uma caixa desenvolvida para organizar dispositivos elétricos de proteção para instalações elétricas. No QDC encontram-se disjuntores comuns, disjuntores diferencial residual, para-raios de baixa tensão e condutores de alimentação e distribuição. Quando é necessário realizar troca de uma lâmpada ou uma tomada, por exemplo, uma ação preliminar é desligar o disjuntor, desenergizando o circuito. Ele também é conhecido no mercado por variações de nome como painel elétrico, quadro do disjuntor e quadro de luz.

Num projeto elétrico, uma tabela representa o QDC é detalhada iniciando-se com o preencimento do levantamento de carga (qualquer dispositivo elétrico que pode entrar em funcionamento) para cada circuito. Então, pede-se elaborar código em C++ que tenha uma classe principal "cargaInstalada" com membros privados "Circuito", "tensao", "corrente" e "potencia". Os membros públicos são: "lampada10W", "tomada100W", "tomada600W", "tomada1300W" e "chuveiro5000W".

Há necessidade de transformação da potência descrita em Watt (W) para Volt Ampère (VA). Para isso é necessário dividir as variáveis membro lâmpada e tomada pelo fator de potência estabelecido aqui como 0,92. No caso do chuveiro o Fator de potência é unitário. Nosso interesse é encontrar a corrente em cada circuito, então para cada circuito com sua potência em VA, calcula-se a corrente dividindo a potência VA pela tensão. A tabela exemplifica situações.

Circuito	Lâmpada 10W	Tomada 100W	Tomada 600W	Tomada 1300W	Chuveiro 5000W	Tensão V	Potência kW	Potência kVA	Corrente A
1	20	0	0	0	0	127	2000	217.4	1.7
2	0	10	2	0	0	127	2200	2391.3	18.8
3	0	0	0	1	0	127	1300	1413.0	11.1
4	0	0	0	0	1	220	5000	5000	22.7

O programa principal tem um vetor de objetos deve-se fazer uma estrutura de repetição para tratar n circuitos lidos em arquivo. Em cada repetição o usuário deve entrar com o número do circuito, quantidade dos dispositivos "lampada10W", "tomada100W", "tomada600W", "tomada1300W" e "chuveiro5000W" de cada circuito, e a tensão do circuito. A corrente, as potências em W e e VA de cada circuito devem ser computadas. É necessário criar destrutor de objeto.

39 Classes em C++ parte B - Exercícios de fixação N°12

Exercício 157

Desenvolva um programa C++ que represente um círculo, contendo:

a) Construtor e destrutor;

b) variável privada *raio* (***int***);

c) métodos para definir o raio do círculo, calcular área e comprimento;

d) programa principal para realizar as operações.

40 Classes em C++ parte B - Exercícios de aplicação N°12

Exercício 158

Deseja-se escolher o transporte de mais barato para encomendas pequenas (volume inferior a 10 litros). Os tipos de transporte são moto, bicicleta, patinete. Bicicleta e patinete têm restrição de distância máxima 5km. Moto tem acréscimo de taxa de seguro para o entregador, sendo valor fixo por viagem. Se usuário marcar "urgência", o *preco_por_km* dobra. Implemente em C++ um programa completo para calcular frete em cada tipo. O programa deverá iniciar com uma classe tipo Frete, sendo:

- os atributos (private) *tipo* (string);
- os atributos (public) *distancia* (double), *urgente* (booleano), *velocidade* (double) e *preco_por_km* (double) e *preco_final* (double);
- os métodos (public): 01 construtor, 01 destrutor, 01 método exibe que mostra na saída padrão as informações atuais dos atributos formatadas, 01 método que calcula o frete e tempo de entrega (em minutos) segundo as variáveis pertinentes.

No programa principal, declare objetos segundo os tipos apresentados, conforme construtores. Leia do teclado todos os dados pertinentes ao frete utilizando-os na função construtora quando necessário. Em algum momento no programa principal as informações do frete são exibidas. O frete mais barato será escolhido para contratação.

Fundamentação 13: Algoritmos Genéticos

Dentre os algoritmos de otimização, destacam-se os algoritmos evolucionários para resolver problemas NP-difíceis, sendo os Algoritmos Genéticos (GAs) o grupo mais popular. Os GAs que utilizam métodos computacionais com metáfora ao processo de evolução natural proposto por Darwin como detalhado em Ribeiro Filho (1994) [2].

Um modelo básico de GAs é mostrado no algoritmo ao lado (Soares (2008) [5]). De acordo com Soares (2008) [5], a analogia ao processo evolutivo das espécies inicia-se nos passos 1 e 2, com inicialização do contador de gerações e com a criação dos primeiras soluções candidatas. Cada solução é chamada de *indivíduo*, e seu grupo, de *população*. No passo 3, os indivíduos recebem uma nota de desempenho obtida da função objetivo que simula um "ambiente de sobrevivência".

entrada (*parâmetros*)
saída(*população*)

1 inicialize t com 0.
2 inicialize a *população* com *parâmetros*.
3 compute o desempenho *população*.
4 se critério de parada é atingido siga para 11.
5 incremente t.
6 aplique método de *seleção*.
7 aplique método de *cruzamento*.
8 aplique método de *mutação*.
9 compute o desempenho da nova *população*.
10 siga para 4.
11 retorne a *população* corrente.

O ciclo evolutivo, marcado pelos passos $4-10$, prossegue com a *seleção* de indivíduos para gerar a próxima população. Os métodos de seleção tentam preservar, durante as gerações, os indivíduos considerados mais aptos para a evolução. Os indivíduos selecionados participam do processo de propagação de informação genética *cruzamento*, segundo uma dada probabilidade. Sobre o resultado do cruzamento, aplica-se o operador de diversidade e finaliza-se a geração com a nova população. Os passos anteriores se repetem até que algum critério seja satisfeito. Termos como população, cruzamento e mutação variam de algoritmo para algoritmo. No entanto, o mais importante é que esta estrutura atende aos passos principais de um algoritmo evolucionário.

Exercício 159

Adaptar o código desenvolvido por Ribeiro Filho [2] para C++ e utilizar as funções teste da Lista de Exercícios Modularidade. Criar classes para possibilitar adicionar novos métodos de cruzamento e seleção.

41 Prova de AEDs I - etapa 3 - valor 100% pontos

Questão 11: (40% pontos)

Nos tempos da Grécia antiga, as catapultas eram máquinas de guerra construídas de madeira que utilizavam um recipiente em formato de concha para lançar objetos a uma grande distância, ultrapassando obstáculos como muralhas e fossos. Pede-se construir um programa em C ou C++ para possibilitar a simulação do desempenho de uma catapulta a respeito do impacto do projétil frente um anteparo (muralha, montanha, muro, etc). No nosso modelo de catapulta, consideramos:

a o estrago no anteparo e, é descrito em função da área do projétil a e da velocidade v dele no momento do impacto como $e = a^2v$;

b a velocidade no momento do impacto é medida por $v^2 = v_o^2 + 2gdk$, sendo v_o a velocidade inicial, k uma constante de frenagem pelo atrito e g a aceleração da gravidade;

c a constante k não tem unidade e, neste problema, pode ser aproximado como $k = (a + 0,2)^2$.

d os projéteis são esféricos, formados rocha talhada com raio r variando de 0,12 m a 0,22 m;

e a gravidade $g = -10m/s^2$, a distância da catapulta ao anteparo $d = 30m$, $v_o = 20m/s$, a área de uma esfera $4\pi r^2$ e o ângulo de lançamento em 45 graus constante ao longo de todas as simulações.

O programa lê os raios de projéteis r de arquivo, sendo um por linha, até que não haja mais dado a ser lido. Após as simulações, o programa deve apresentar qual valor do raio que provocou maior estrago.

(Pontuação: elaborar função para computar estrago n_1 pts; entrada e saída de dados n_2 pts; programa principal n_3 pts.)

Questão 12: (40% pontos)

Deseja-se escolher o transporte de mais barato para encomendas pequenas (volume inferior a 10 litros). Os tipos de transporte são moto, bicicleta, patinete. Bicicleta e patinete têm restrição de distância máxima 5km. Moto tem acréscimo de taxa de seguro para o entregador, sendo valor fixo por viagem. Se usuário marcar "urgência", o *preco_por_km* dobra. Implemente em C++ um programa completo para calcular frete em cada tipo. O programa deverá iniciar com uma classe tipo Frete, sendo:

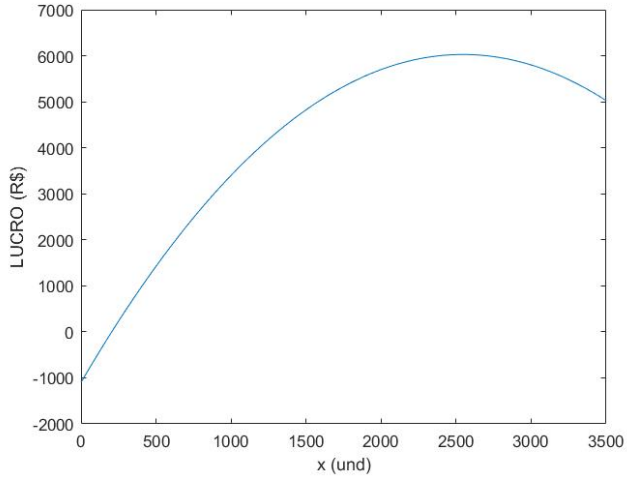
- os atributos (private) *tipo* (string);
- os atributos (public) *distancia* (double), *urgente* (booleano), *velocidade* (double) e *preco_por_km* (double) e *preco_final* (double);
- os métodos (public): 01 construtor, 01 destrutor, 01 método exibe que mostra na saída padrão as informações atuais dos atributos formatadas, 01 método que calcula o frete e tempo de entrega (em minutos) segundo as variáveis pertinentes.

No programa principal, declare objetos segundo os tipos apresentados, conforme construtores. Leia do teclado todos os dados pertinentes ao frete utilizando-os na função construtora quando necessário. Em algum momento no programa principal as informações do frete são exibidas. O frete mais barato será escolhido para contratação.

(Pontuação: declarações de atributos e implementação dos métodos construtor e destrutor n_1 pts; implementação método calcular frete e tempo de entrega n_2 pts; declaração e implementação para entrada e saída de dados n_3 pts; programa principal n_4 pts.)

Questão 13: (20% pontos)

Um publicitário resolveu fazer canetas personalizadas com estampa de times de futebol para vender. Se nenhuma caneta for vendida ele terá prejuízo de R$ 1.100,00, por causa dos custos de matéria prima. Ele planeja criar preços promocionais (política de descontos) que podem acelerar as vendas. Entretanto, há um limite de descontos que deixa o publicitário sem lucro e sem prejuízos. O modelo matemático e a representação gráfica estão apresentados abaixo.



$$L = -0.0011x^2 + 5.6x - 1100$$

Desenvolva código em C ou C++, deve responder, qual a quantidade x que gera um dado lucro L. Personalize esse código para descobrir: a) o valor de x_{max} que gera o lucro máximo L_{max}, e b) qual a quantidade mínima x_{min} de venda para que não ocorra prejuízo.

(Pontuação: declarações n_1 pts; implementação método calcular lucro n_2 pts; declaração e restante e programa principal n_3 pts.)

Referências

[1] Eugene Loh e G. William Walster. "Rump's example revisited". Em: *Reliable Computing* (2002). issn: 13853139. doi: 10.1023/A:1015569431383.

[2] J.L. Ribeiro Filho, P.C. Treleaven e C. Alippi. "Genetic-algorithm programming environments". Em: *Computer* 27.6 (1994), pp. 28–43. doi: 10.1109/2.294850.

[3] R. W. Saaty. "THE ANALYTIC HIERARCHY PROCESS-WHAT AND HOW IT IS USED". Em: 9.3 (1987), pp. 161–176.

[4] G. L. Soares et al. "An Interval-Based Target Tracking Approach for Range-Only Multistatic Radar". Em: *IEEE Transactions on Magnetics* 44.6 (2008), pp. 1350–1353. doi: 10.1109/TMAG.2007.916286.

[5] Gustavo Luís Soares. "Algoritmos Determinístico e Evolucionário Intervalares para Otimização Robusta Multi-Objetivo". Tese de dout. Universidade Federal de Minas Gerais/Université de Bretagne Occidentale, 2008, p. 243.