



1001350 – Lista de exercícios

Jander Moreira*

Última revisão: 5 de Março de 2019

Conteúdo

1 Introdução

2 Conceitos gerais

3 Algoritmos

- 3.1 Algoritmos informais
- 3.2 Especificações de entradas e saídas
- 3.3 Algoritmos formais básicos
- 3.4 Testes de mesa
- 3.5 Estruturas condicionais
- 3.6 Estruturas de repetição
- 3.7 Refinamentos sucessivos
- 3.8 Variáveis compostas
- 3.9 Modularização

4 Programação

- 4.1 Tipos de dados e constantes
- 4.2 Expressões
- 4.3 Estruturas condicionais
- 4.4 Estruturas de repetição
- 4.5 Variáveis compostas
- 4.6 Ponteiros
- 4.7 Modularização
- 4.8 Arquivos

5 Comentários de exercícios selecionados

Lista de figuras

Lista de tabelas

Lista de quadros

Lista de algoritmos

Lista de programas

*Moreira, J. – Universidade Federal de São Carlos – Departamento de Computação – Rodovia Washington Luis, km 235 – 13565-905 - São Carlos/SP – Brasil – jander@dc.ufscar.br

1 Introdução

Esta lista de exercícios é proposta para a disciplina Construção de Algoritmos e Programação (CAP), ofertada pelo Departamento de Computação da Universidade Federal de São Carlos.

- 1 Ela é uma compilação de exercícios originais e também por inspirados em outras fontes.

- 1 É importante destacar que o aluno deve ser capaz de resolver os exercícios sozinho, porém o estudo em grupo é incentivado como ferramenta de aprendizado. Cada aluno individualmente deve tomar o cuidado para não apenas observar outros resolvendo as questões.

- 2 O nível dos exercícios varia bastante. Alguns são apenas para verificar se um conceito foi assimilado e outros propõem desafios mais ousados.

- 3 Exercícios importantes são marcados por uma ou mais estrelas: ★.

2 Conceitos gerais

- 2-1 Quando se fala em velocidade de transmissão de dados, são usadas taxas com valores como 100Mbps, por exemplo. O termo bps significa *bits por segundo* e o M indica *mega*, ou seja, 10^6 . Assim, 100Mbps são 100.000.000 bits por segundo.

- Por outro lado, ao se referir à memória de um computador, há outra interpretação envolvida. O valor de 100 megabytes de memória equivale a 104.857.600 bytes.

- 45 Pesquise as abreviações M, G, T e confronte-as com Mi, Gi e Ti.

- 46

- 46 2-2 Monte uma tabela em uma planilha eletrônica (como Excel, LibreOffice ou Google) sobre medidas de capacidade de memória. Use o byte como unidade básica e dê todas as outras medidas em função dela (e.g., 1 bit equivale a $\frac{1}{8}$ byte, ou seja, 0,125).

- 47 Nas linhas, coloque os valores para bit, byte e os múltiplos de byte (KiB, MiB, GiB, TiB, PiB, EiB, ZiB, YiB). Nas colunas, indique o nome da medida, sua abreviação, número de bytes, número de bytes em potência de 2 e número de bytes em potência de 10 (aproximado).

- 2-3 Monte uma tabela em uma planilha eletrônica (como Excel, LibreOffice ou Google) sobre medidas de tempo. Use o segundo como unidade básica e dê todas as outras medidas em função dela.

- Nas linhas, coloque os valores indo de picossegundos até segundos (para cada potência de 10), seguido de

Figura 3.1-1: Receita para preparação de moussaka.
 Fonte: <http://gnt.globo.com/receitas/receitas/como-fazer-moussaka.htm>, visitado em 17/5/2018.

Moussaka

- Corte as berinjelas em lâminas longitudinais e polvilhe com sal, deixando sorar por 10 minutos.
- Em uma panela bem quente, refogue a cebola, o alho, a hortelã, a canela e o louro com azeite.
- Adicione a carne moída, refogando até dourar.
- Polvilhe a farinha de trigo, acrescente o sal e a pimenta do reino, adicionando o vinho, o extrato de tomate e a água. Deixe ferver, cozinhando por 30 minutos.
- Lave as berinjelas e seque com papel toalha.
- Aqueça uma frigideira antiaderente com um fio de azeite e frite as berinjelas por 2 minutos de cada lado.
- Misture o queijo parmesão e os ovos, fora do fogo, no molho, mexa bem e tempere.
- Em uma assadeira, intercale as batatas, um terço da carne, as fatias de berinjela e molho. Repita até terminar (sic.) os ingredientes, finalizando com o molho branco.
- Polvilhe com parmesão e asse por 45 minutos.

minuto, hora, dia, semana, mês e ano. Nas colunas, indique o nome da medida, sua abreviação, número de segundos e número de bytes em potência de 10 (aproximado).

2-4 Explique a seguinte sentença: “quando um computador não funciona, software é o que você xinga e hardware é o que você chuta”.

3 Algoritmos

3.1 Algoritmos informais

3.1-1 ✓ Receitas culinárias são exemplos típicos de algoritmos: levam de um estado inicial (ingredientes separados) a um estado final (prato pronto).

Um exemplo para a preparação de uma moussaka, assumindo que ingredientes, utensílios e equipamentos estejam disponíveis, é apresentado na Figura 3.1-1.

Avalie se esta receita é um bom algoritmo, ou seja, se obedece a execução sequencial e se as instruções são claras e precisas (tanto quanto possível em uma receita).

3.1-2 Escreva um algoritmo informal com instruções para que uma pessoa, saindo pela porta da frente do DC, consiga chegar à Biblioteca Comunitária (BCo), fazendo a rota a pé.

Tendo a oportunidade, repasse seu algoritmo a um colega para que ele dê uma opinião se está adequado.

3.1-3 ✓ Com uma caneta (e não lápis, para não poder apagar), use o espaço reservado na sequência para executar as instruções do Algoritmo 3.1-1.



Algoritmo 3.1-1 Instruções para o desenho.

Desenhe cinco círculos lado a lado
 Escreva dentro dos círculos as letras de A a E

As instruções dadas estão claras o suficiente para que a execução gere o resultado esperado?

3.1-4 ✓ Usando uma caneta (e não lápis, para não poder apagar), execute as instruções do Algoritmo 3.1-2.

Algoritmo 3.1-2 Instruções para desenho.

Desenhe um quadrado
 Desenhe um círculo no centro do quadrado
 Desenhe um losango no canto superior esquerdo do quadrado
 Desenhe um círculo no canto inferior direito do quadrado
 Escreva sua idade no canto inferior esquerdo do quadrado
 Escreva o ano atual no canto superior direito do quadrado
 Calcule o valor de $e = 2i + 1$, sendo i sua idade em anos
 Se e for ímpar, ao invés do círculo, desenhe um triângulo no centro

O que foi desenhado no centro do quadrado: um círculo ou um triângulo?

3.1-5 Escreva um algoritmo informal que ensine uma pessoa a identificar o nome do aluno da turma que mora mais distante de São Carlos. Não se preocupe com notação formal; foque em escrever um algoritmo que funcione.

Tente executar seu algoritmo passo a passo para ver se ele funciona corretamente. Se possível, compartilhe com seus colegas e colha opiniões.

3.1-6 Execute o Algoritmo 3.1-3 para um grupo de colegas, excluindo você.

Algoritmo 3.1-3 Determinação do valor máximo.

Pergunte a idade de um dos colegas e a anote
Para cada colega restante

- Pergunte a idade do colega
- Se esta idade for maior que a atualmente anotada, substitua sua anotação pela idade maior

Anuncie a todos o valor da idade que está anotada

Você consegue verificar que o algoritmo faz com que seja apresentado o valor de idade máximo do grupo? Note que a única restrição é que exista pelo menos um colega no grupo (teste os passos considerando apenas um colega no grupo e veja que ainda assim o “para cada colega restante” ainda faz sentido).

3.2 Especificações de entradas e saídas

3.2-1 Identifique qual o espectro de valores abrangidos. Diga quantos valores diferentes são possíveis em cada caso.

- Um valor entre 0 e 10;
- Um valor inteiro no intervalo $[0, 10]$;
- Um valor real no intervalo $[0, 10]$, com apenas uma casa decimal.

3.2-2 Usando como exemplo os intervalos do Exercício 3.2-1, indique como poderiam ser especificados os valores para:

- A idade de uma pessoa em anos;
- O valor de um ano d.C. (e.g., 800, 1345, 1967...);
- O valor de um saldo bancário.

3.2-3 É preciso fazer o cálculo da velocidade média (em km/h) de um percurso feito por um veículo, dada a distância percorrida (em quilômetros) e o tempo gasto (em horas).

Forneça as especificações de entrada e saída para o problema. Não é para escrever o algoritmo.

3.2-4 Um professor precisa calcular a média das notas de duas provas de um determinado aluno, usando média aritmética simples.

Forneça as especificações de entrada e saída para o problema. Não é para escrever o algoritmo.

3.2-5 O cálculo da nota média dos relatórios entregues em uma disciplina é feito desconsiderando-se as duas piores notas de um total de 10. A média é aritmética simples, com pesos iguais.

Forneça as especificações de entrada e saída para o problema. Não é para escrever o algoritmo.

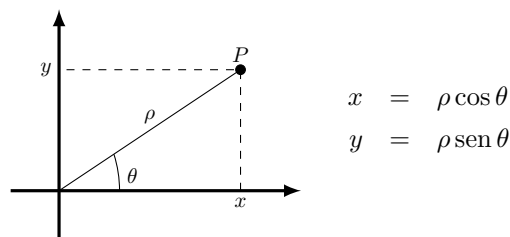
3.2-6 Um algoritmo deve fazer a conversão de uma entrada única em graus Kelvin para graus Fahrenheit. Verifique a escala válida para Kelvin, contemplando-a na especificação. Procure pela fórmula de conversão.

Forneça as especificações de entrada e saída para o problema. Não é para escrever o algoritmo.

3.2-7 Em um laboratório de pesquisa está sendo desenvolvido um robô que se move autonomamente. Seu sistema de navegação usa câmeras e sensores de proximidade para evitar colisões.

Seu sistema de navegação, em especial, é provido por radiofrequência, recebendo as coordenadas cartesianas (x, y) de sua localização atual e destino continuamente. Essas coordenadas são dadas em metros.

Entretanto, internamente, o processamento do robô utiliza coordenadas polares (ρ, θ) , com ρ dado em metros e θ em radianos, indo de 0 a 2π .



É preciso elaborar um algoritmo para converter as coordenadas cartesianas para coordenadas polares. Forneça as especificações das entradas e saídas para o problema. Não é para escrever o algoritmo.

3.2-8 Em um clube de tiro ao alvo foi instalado um sistema de sensores que atuam no monitoramento das sessões dos praticantes. Existe um sensor na arma que detecta um disparo. Há também um sensor no alvo, que gera automaticamente a cada disparo a pontuação referente à parte do alvo que foi atingida. No caso do praticante errar o alvo por completo, a pontuação é zero; no caso de acertar o alvo principal (áreas centrais do tórax e cabeça do zumbi no exemplo da Figura 3.2-1), a pontuação é 50. Todas as pontuações são dadas em valores inteiros.

Com o conjunto dos dois sensores, é possível saber quando houve cada disparo e a pontuação referente a ele.

É preciso de um algoritmo que apresente, a partir das pontuações de uma seção de 10 tiros, a pontuação total e a pontuação média do praticante. Forneça as especificações das entradas e saídas para o problema descrito. Não é para escrever o algoritmo.

Figura 3.2-1: Exemplo de alvo com pontuações. Fonte: https://tprairsoft.files.wordpress.com/2014/07/zombies_silhuetas_11.jpg.



3.3 Algoritmos formais básicos

3.3-1 Execute o Algoritmo 3.3-1 para cada uma das seguintes entradas (ângulos em radianos):

- $\rho = 10$ e $\theta = \pi/4$
- $\rho = \sqrt{2}$ e $\theta = 3\pi/8$
- $\rho = 0$ e $\theta = \pi$
- $\rho = 3,5$ e $\theta = 4,71238898$

Algoritmo 3.3-1 Algoritmo para conversão de coordenadas polares para cartesianas.

Input: A distância ρ de um ponto até a origem e o ângulo θ em radianos que este ponto faz com a abscissa ($\rho \in \mathbb{R}^+$ e $0 \leq \theta < 2\pi$)

Output: As coordenadas cartesianas x e y do ponto

Obtenha os valores de ρ e θ

Calcule x como $\rho \cos \theta$

Calcule y como $\rho \sin \theta$

Apresente os valores x e y calculados

3.3-2 Corrija a indentação dos Algoritmos 3.3-2 e 3.3-3. No caso do segundo algoritmo, supõe-se uma disponibilidade de espaço horizontal bastante reduzida (artificial, claro).

Algoritmo 3.3-2 Algoritmo com indentação inadequada.

Input: Dois valores $v_1, v_2 \in \mathbb{R}$

Output: A média m dos dois valores

Obtenha os valores v_1 e v_2

Calcule $m = (v_1 + v_2)/2$

Apresente m

Algoritmo 3.3-3 Algoritmo com indentação inadequada.

Input: O comprimento
 $c_1, c_2, c_3 \in \mathbb{R}^{+*}$ que formam um triângulo válido

Output: A área a do triângulo

Obtenha os valores

c_1, c_2 e c_3

Calcule p como
 $\frac{c_1 + c_2 + c_3}{2}$

Calcule a como
 $\frac{\sqrt{p(p - v_1)(p - v_2)(p - v_3)}}{2}$

Apresente a área calculada a

3.3-3 ☑ Compare o Algoritmo 3.3-1 com o Algoritmo 3.3-4. Há dúvidas quanto à equivalência entre eles? Qual deles prefere?

Algoritmo 3.3-4 Algoritmo para conversão de coordenadas polares para cartesianas.

Input: A distância ρ de um ponto até a origem e o ângulo θ em radianos que este ponto faz com a abscissa ($\rho \in \mathbb{R}^+$ e $0 \leq \theta < 2\pi$).

Output: As coordenadas cartesianas x e y do ponto.

read ρ, θ

$x \leftarrow \rho \cos \theta$

$y \leftarrow \rho \sin \theta$

write x, y

3.3-4 Considere os dados da Tabela 3.3-1. O objetivo é calcular a matriz de covariância dos dados.

Este cálculo considera cada coluna a amostra de de um tipo de dado. A covariância entre duas colunas X e Y é dada por

$$\text{cov}_{XY} = \frac{1}{n} \sum_{i=1}^n (x_i - \mu_X)(y_i - \mu_Y),$$

sendo os valores x_i as amostras da coluna X e y_i as amostras da coluna Y , com $i = 1, 2, \dots, n$. Os valores μ_X e μ_Y correspondem à média das amostras de X e Y , respectivamente.

A matriz de covariância é uma matriz quadrada formada pelas combinações de todos os tipos de dados.

Para o exemplo da Tabela 3.3-1, a matriz de covariância é

$$\begin{pmatrix} 23,71 & 32,57 & 0,14 \\ 32,57 & 48,53 & 0,96 \\ 0,14 & 0,96 & 0,24 \end{pmatrix},$$

com as linhas e colunas representando idade, nota e sexo, nesta ordem.

Tabela 3.3-1: Alguns dados sobre alunos ingressantes no ensino superior. A idade é dada em meses, a nota está no intervalo $[0, 100]$ e sexo usa 1 feminino e 2 para masculino.

Idade	Nota	Sexo
221	82	1
227	89	1
215	75	2
218	79	1
225	91	2
230	97	2
225	85	1

Reproduza esse resultado seguindo o Algoritmo 3.3-5.

Algoritmo 3.3-5 Cálculo da matriz de covariância. A notação A^T representa a transposta de uma matriz A .

Input: Uma matriz de dados numéricos, organizada por colunas

Output: Uma matriz quadrada com as covariâncias dos dados

Obtenha a matriz de dados M

▷ *Centralize os dados*

Calcule a média de cada coluna de M

Crie uma matriz M_c a partir de M subtraindo de cada dado de M a média sua respectiva coluna

▷ *Calcule a matriz de covariância*

Calcule a matriz C pelo produto $M_c^T \times M_c$

Divida cada valor de C pelo número de linhas de M

▷ *Resultado*

Apresente a matriz C

3.3-5 Escreva um algoritmo que apresente o índice de massa corporal (IMC) de uma pessoa, fornecidas sua massa e a sua altura.

O IMC é calculado, dada a massa m em quilogramas e a altura h em centímetros, por

$$I_{IMC} = \frac{m}{h^2}.$$

Tome os devidos cuidados com as especificações da entrada e da saída.

3.3-6 Escreva um algoritmo que apresente a média das notas de duas provas, dadas notas de 0 a 10. A primeira nota deve ter peso 4 e a segunda, 6. Tome os devidos cuidados com as especificações da entrada e da saída.

A média, claro, é

$$m = \frac{4p_1 + 6p_2}{10}.$$

3.3-7 Escreva um algoritmo que dados um mês inicial e um mês final, ambos no mesmo ano, apresente o número de meses envolvidos. Os valores a serem fornecidos são numéricos (i.e., de 1 a 12) e na ordem correta. O cálculo deve apresentar, por exemplo, o valor 3 para os meses de março (3) a maio (5).

Tome os devidos cuidados com as especificações da entrada e da saída.

3.3-8 Escreva um algoritmo que dados um mês inicial e um mês final, cada um em um ano diferente e consecutivo, apresente o número de meses envolvidos. Os valores a serem fornecidos são numéricos (i.e., de 1 a 12) e na ordem correta. O cálculo deve apresentar, por exemplo, o valor 13 para os meses de maio (5) a maio (5) e o valor 2 para dezembro (12) e janeiro (1).

3.3-9 ★★ Escreva um algoritmo de nível alto para, dada uma quantia monetária em reais (R\$), determinar a quantidade mínima de notas e moedas que a compõe.

Por exemplo, o valor R\$ 413,35 usa nove notas ou moedas, pois é formado por:

- quatro notas de 100;
- uma nota de 10;
- uma nota de 2;
- uma moeda de 1;
- uma moeda de 0,25;
- uma moeda de 0,10.

O objetivo deste exercício é conseguir pensar na manipulação de dados necessária, sem os detalhes dos cálculos. A versão mais detalhada será solicitada no Exercício 3.7-6.

3.3-10 Considere o Algoritmo 3.3-6. Você está confortável com o tipo da saída produzida por ele?

Algoritmo 3.3-6 Determinação de validade de intervalo de uma nota acadêmica.

Input: Um valor numérico qualquer em \mathbb{R}

Output: A condição de validade do valor como uma nota acadêmica no intervalo $[0, 10]$: **true** ou **false**

Obtenha o valor n

$v \leftarrow 0 \leq n$ e $n \leq 10$

Apresente a verificação v

3.3-11 ✓ Considerando a notação algorítmica utilizada nesta disciplina, marque os itens que apresentam verificações **incorretamente** estruturadas. Considere os x_i valores numéricos e os l_i valores lógicos.

- ☐ $x \geq 0$
- ☐ $x_1 < x_2$ e $x_1 < x_3$
- ☐ $x > 10$ ou $-1 < x < 1$
- ☐ $x = 1$
- ☐ l_1
- ☐ $x = 1$ ou 2 ou 3
- ☐ $x_1 > x_2$ ou $x_3 < 0$ e l_1 ou l_2
- ☐ $x_1, x_2, x_3 \geq 0$

3.3-12 Complete a tabela verdade do Quadro 3.3-1 com os resultados de cada expressão (colunas), dados os valores de l_1 , l_2 e l_3 especificados nas linhas.

Quadro 3.3-1 Tabela verdade. V indica verdadeiro (true) e F indica falso (false).

l_1	l_2	l_3	não l_1 ou l_2 e l_3	l_1 e (l_2 ou l_3)
F	F	F		
F	F	V		
F	V	F		
F	V	V		
V	F	F		
V	F	V		
V	V	F		
V	V	V		

3.3-13 Apresente a ordem em que as expressões seguintes são avaliadas, considerando os valores l_i como lógicos:

- l_1 e l_2 ou l_3
- l_1 ou l_2 e l_3
- $(l_1$ ou $l_2)$ e l_3
- não l_1 e l_2
- não $(l_1$ e $l_2)$

3.3-14 Fornecidos o comprimento dos três comprimentos de segmentos de reta (todos com valores maiores que zero), informe se os lados formam um triângulo válido ou não.

Neste exercício não é permitido o uso de estruturas condicionais; use apenas expressões lógicas, como as do Algoritmo 3.3-6.

3.3-15 ✓ Um número perfeito é todo $n > 0$ tal que n é a soma de todos seus divisores, exceto ele mesmo. Por exemplo, o número 6 tem os elementos do conjunto $\{1, 2, 3\}$ como divisores (excetuando-se o próprio 6) e $6 = 1 + 2 + 3$.

Escreva um algoritmo para, dado um valor inteiro maior que zero e menor ou igual a 10, apresentar um resultado lógico indicando se o valor é ou não um número perfeito.

Neste exercício não é permitido o uso de estruturas condicionais; use apenas expressões lógicas, como as do Algoritmo 3.3-6.

3.4 Testes de mesa

3.4-1 Escreva um algoritmo que, a partir duas coordenadas no plano (x_1, y_1) e (x_2, y_2) , apresente a área do retângulo definido por elas, considerando as arestas paralelas aos eixos.

Teste seu algoritmo para as seguintes entradas, executando cada instrução passo a passo e anotando os resultados:

- $(-1, -1)$ e $(1, 1)$
- $(1, 9)$ e $(4, 12)$
- $(8, 3)$ e $(5, 10)$
- $(10, 7)$ e $(8, -1)$

3.5 Estruturas condicionais

3.5-1 ✓ Acompanhe a execução do Algoritmo 3.5-1 e verifique que ele classifica corretamente um triângulo a partir dos comprimentos de seus lados, incluindo a verificação se os comprimentos permitem ou não a formação do triângulo.

Tenha claro quais condições são válidas nas linhas 5, 8 e 11.

3.5-2 ✓ Acompanhe a execução do Algoritmo 3.5-2. Há um erro na lógica da solução e ele não apresenta sempre o resultado esperado. Identifique o problema.

3.5-3 Verifique se o Algoritmo 3.5-3 está correto. Note que $a = 0$ não é uma entrada possível considerando-se a equação de segundo grau na forma $ax^2 + bx + c = 0$.

Algoritmo 3.5-1 Determinação de existência e tipo de triângulo a partir dos comprimentos de três segmentos de reta.

Input: Três comprimentos de segmentos de reta, todos maiores que zero

Output: A classificação de acordo com a relação entre os comprimentos: “inexistente” (se o triângulo não é possível), “equilátero” (todos os lados iguais), “isósceles” (apenas dois lados iguais) ou “escaleno” (todos os lados diferentes)

```

1► Obtenha os comprimentos
2► if a soma de quaisquer dois comprimentos for
   menor ou igual ao terceiro then
3►   Defina o tipo como “inexistente”
4► else
5►   if todos os comprimentos forem iguais then
6►     Defina o tipo como “equilátero”
7►   else
8►     if há quaisquer dois comprimentos
       iguais then
9►       Defina o tipo como “isósceles”
10►    else
11►      Defina o tipo como “escaleno”
12►    end if
13►  end if
14► end if
15► Apresente o tipo definido

```

Algoritmo 3.5-2 Determinação de existência e tipo de triângulo a partir dos comprimentos de três segmentos de reta (com erro de lógica).

Input: Três comprimentos de segmentos de reta, todos maiores que zero

Output: A classificação de acordo com a relação entre os comprimentos: “inexistente” (se o triângulo não é possível), “equilátero” (todos os lados iguais), “isósceles” (apenas dois lados iguais) ou “escaleno” (todos os lados diferentes)

```

1► Obtenha os comprimentos
2► if a soma de quaisquer dois comprimentos for
   menor ou igual ao terceiro then
3►   Defina o tipo como “inexistente”
4► else
5►   if todos os comprimentos forem iguais then
6►     Defina o tipo como “equilátero”
7►   end if
8►   if há quaisquer dois comprimentos
       iguais then
9►     Defina o tipo como “isósceles”
10►  end if
11►  if todos os lados são diferentes then
12►    Defina o tipo como “escaleno”
13►  end if
14► end if
15► Apresente o tipo definido

```

Algoritmo 3.5-3 Determinação do número de raízes reais e distintas de uma equação de segundo grau.

Input: Os coeficientes de uma equação de segundo grau válida

Output: O número de raízes reais distintas que a equação determina

```

Obtenha os coeficientes
if o discriminante da equação for negativo then
  n ← 0
else if o discriminante da equação for zero then
  n ← 1
else
  n ← 2
end if
Apresente n

```

Tabela 3.5-1: Relação entre conceitos e notas.

Conceito	Nota
A	10,0
B	8,5
C	7,0
D	5,0
E	2,0
F	0,0

Tabela 3.5-2: Relação entre notas e conceitos.

Nota	Conceito
$9 < n \leq 10$	A
$7,5 < n \leq 9$	B
$5 < n \leq 7,5$	C
$2 < n \leq 5$	D
$0 < n \leq 2$	E
$n = 0$	F

3.5-4 ★★★ Em uma instituição de ensino, o histórico é apresentado usando-se conceitos e não notas.

Os cálculos com conceitos são feitos convertendo-se o conceito para valor numérico (pela Tabela 3.5-1), fazendo-se o cálculo e depois retornando o resultado para conceito (Tabela 3.5-2).

Por exemplo, a média dos conceitos A e D é dada por

$$\frac{A + D}{2} = \frac{10 + 5}{2} = 7,5 = C.$$

O Algoritmo 3.5-4 apresenta uma solução de nível alto para o cálculo da média de dois conceitos. Observe-o e avalie se está correto, ou seja, se a sequência dos passos leva à solução de forma completa e coerente.

Note que uma versão mais detalhada deste exercício será solicitada no Exercício 3.7-6.

Algoritmo 3.5-4 Média de dois conceitos.

Input: Dois conceitos de A a F**Output:** O conceito resultante da média das entradas

Obtenha os dois conceitos
Converta ambos os conceitos para notas usando a
Tabela 3.5-1
Calcule a média das duas notas convertidas
Converta a média para conceito usando a Ta-
bela 3.5-2
Apresente o conceito da média

3.5-5 Um problema recorrente em soluções computacionais é a determinação de valor máximo (ou mínimo) em grupos de dados.

Alguns algoritmos são apresentados para a obtenção do máximo entre dois valores numéricos (Algoritmos 3.5-5 a 3.5-8), usando estratégias ligeiramente diferentes.

Todos estão corretos? Teste cada um deles executando-os passo a passo.

Algoritmo 3.5-5 Determinação de valor máximo.

Input: Dois valores reais v_1 e v_2 **Output:** O valor máximo entre as entradas

Obtenha v_1 e v_2
if $v_1 > v_2$ **then**
 $v_{\text{máx}} \leftarrow v_1$
else
 $v_{\text{máx}} \leftarrow v_2$
end if
Apresente $v_{\text{máx}}$

Algoritmo 3.5-6 Outra determinação de valor máximo.

Input: Dois valores reais v_1 e v_2 **Output:** O valor máximo entre as entradas

Obtenha v_1 e v_2
 $v_{\text{máx}} \leftarrow v_1$
if $v_2 > v_{\text{máx}}$ **then**
 $v_{\text{máx}} \leftarrow v_2$
end if
Apresente $v_{\text{máx}}$

3.5-6 Execute o Algoritmo 3.5-9 para as seguintes entradas:

- 3, 4 e 5
- 5, 3 e 4
- 8, 8 e 4
- 5, 5 e 5
- 1, 2 e 5

Analise, nas execuções, como cada condição dos **ifs** é avaliada. Em particular:

Algoritmo 3.5-7 Ainda outra determinação de valor máximo.

Input: Dois valores reais v_1 e v_2 **Output:** O valor máximo entre as entradas

Obtenha v_1 e v_2
 $v_{\text{máx}} \leftarrow 0$
if $v_1 > v_{\text{máx}}$ **then**
 $v_{\text{máx}} \leftarrow v_1$
end if
if $v_2 > v_{\text{máx}}$ **then**
 $v_{\text{máx}} \leftarrow v_2$
end if
Apresente $v_{\text{máx}}$

Algoritmo 3.5-8 Ainda outra determinação de valor máximo.

Input: Dois valores reais v_1 e v_2 **Output:** O valor máximo entre as entradas

Obtenha v_1 e v_2
 $v_{\text{máx}} \leftarrow 0$
if $v_1 > v_{\text{máx}}$ **then**
 $v_{\text{máx}} \leftarrow v_1$
else if $v_2 > v_{\text{máx}}$ **then**
 $v_{\text{máx}} \leftarrow v_2$
end if
Apresente $v_{\text{máx}}$

Algoritmo 3.5-9 Determinação de existência e tipo de triângulo a partir dos comprimentos de três segmentos de reta.

Input: Três comprimentos de segmentos de reta, l_1 , l_2 e l_3 ($l_1, l_2, l_3 \in \mathbb{R}^{*+}$)**Output:** Um tipo t de acordo com a relação entre os comprimentos, $t \in \{\text{"inexistente"}, \text{"equilátero"}, \text{"isósceles"}, \text{"escaleno"}\}$

Obtenha os comprimentos l_1 , l_2 e l_3
if $l_1 < l_2 + l_3$ ou $l_2 < l_1 + l_3$ ou $l_3 < l_1 + l_2$ **then**
 $t \leftarrow \text{"inexistente"}$ $\triangleright \Delta$ impossível
else
 if $l_1 = l_2$ e $l_2 = l_3$ **then**
 $t \leftarrow \text{"equilátero"}$ \triangleright todos iguais
 else
 if $l_1 = l_2$ ou $l_1 = l_3$ ou $l_2 = l_3$ **then**
 $t \leftarrow \text{"isósceles"}$ \triangleright apenas dois iguais
 else
 $t \leftarrow \text{"escaleno"}$ \triangleright todos diferentes
 end if
 end if
end if
Apresente o tipo m

- Note que as condições dos vários **ifs** detalham as mesmas verificações expressas no Algoritmo 3.5-1;
- Tenha claro porque a comparação da linha 8 é suficiente para determinar o tipo como isósceles, sem confrontar com equilátero.

3.5-7 Analise a execução do Algoritmo 3.5-10 para os seguintes dados:

- $v = 2.000,00$, $t = 0,001$, $m = 2$
- $v = 500,00$, $t = 0,005$, $m = 6$
- $v = 500,00$, $t = 0,005$, $m = 7$
- $v = 1.000,00$, $t = 1,02$, $m = 1$

Algoritmo 3.5-10 Projeção de rendimentos para aplicação com liquidez diária feita no dia 1º para o último dia do mês. Anos bissextos não são considerados.

Input: O valor $v \in \mathbb{R}$, $v > 0$ a ser aplicado; a taxa diária $t \in \mathbb{R}$, $0 < t \leq 1$; o mês $m \in \mathbb{N}$, $1 \leq m \leq 12$ que será considerado

Output: O valor da previsão de ganho $p \in \mathbb{R}$, $p > 0$

Obtenha os valores para v , t e m

switch m **of**

▷ *determinação do número de dias*

case 1, 3, 5, 7, 8, 10, 12 **do**

$n \leftarrow 31$

end case

case 2 do

$n \leftarrow 28$

end case

case 4, 6, 9, 11 do

$n \leftarrow 30$

end case

end switch

Calcule p como $v(1+t)^n$ ▷ *juros compostos*

Apresente a projeção p ▷ *resultado calculado*

3.5-8 A ordem cronológica é de imprescindível importância em algoritmos. Estruture a lógica de comparação para apresentar em ordem cronológica três datas.

Cada data é formada por três valores inteiros (dia, mês e ano) e sempre representam uma data válida.

Escreva um algoritmo de nível mais alto que, dadas as três datas (i.e., nove valores inteiros), apresente-as em ordem cronológica crescente. Se o nível do algoritmo ficou alto demais, fique à vontade para detalhar um pouco mais.

3.5-9 Um valor racional $q \in \mathbb{Q}$ é um valor que pode ser escrito na forma a/b , sendo $a \in \mathbb{Z}$ e $b \in \mathbb{Z}^*$.

A partir de dois valores inteiros separados, o primeiro representando o numerador e o segundo o denominador de uma fração, deseja-se o valor racional em sua

Tabela 3.5-3: Exemplos de valores de numerador e denominador de uma razão e a representação “canônica” esperada.

n	d	$\frac{n}{d}$
7	21	$\frac{1}{3}$
-6	7	$-\frac{6}{7}$
6	-7	$-\frac{6}{7}$
15	5	$\frac{3}{1}$
9	-3	$-\frac{3}{1}$
0	-3	$\frac{0}{1}$

forma irredutível (i.e., a forma simplificada da fração). Assuma que o denominador será sempre diferente de zero e use as seguintes regras adicionais para a representação:

- se o valor for positivo, deve ser representado com numerador e denominador positivos;
- se o valor for negativo, deve ser representado com o numerador negativo e o denominador positivo;
- se o valor representar um número inteiro, o denominador deve ser 1, o que inclui o caso do numerador nulo.

A Tabela 3.5-3 mostra exemplos de possíveis entradas e representações.

Escreva um algoritmo mostre a lógica para, a partir dos valores originais de n e d , sejam calculados os valores n' e d' , de forma que $\frac{n'}{d'} = \frac{n}{d}$ e os novos numerador e denominador sigam as regras estabelecidas.

Apresente o algoritmo em nível alto, a exemplo do Algoritmo 3.5-1. Ou seja, opte por “se ambos forem negativos” em lugar de “ $n < 0$ e $d < 0$ ”, por exemplo.

3.5-10 ★★ Em alguns processos químicos, determinadas medidas são colhidas três vezes, pois podem variar consideravelmente em um curto espaço de tempo.

Para cada coleta, é escolhida como medida para registro a de valor intermediário, sendo descartadas a mais baixa e a mais alta. Quando há medidas iguais, é irrelevante qual é escolhida.

Assim, para t_1 , t_2 e t_3 , é escolhida

$$t_m \mid t_i \leq t_m \leq t_s, \forall i, m, s \in \{1, 2, 3\}.$$

Escreva um algoritmo completo para, a partir de três medidas, apresentar a medida intermediária, detalhando a seleção da mais adequada.

Tabela 3.5-4: Fator de multiplicação para conversão de unidades de medida de velocidade: metros por segundo, metros por minuto, quilômetros por hora e milhas por hora.

	m/s	m/min	km/h	mi/h
m/s	1	60	3,6	2,237
m/min	0,01667	1	0,05	0,03728
km/h	0,02778	16,67	1	0,6214
mi/h	0,447	26,42	1,609	1

3.5-11 Uma equação de segundo grau é determinada por três coeficientes, a , b e c , com $a \neq 0$, escrevendo-se

$$ax^2 + bx + c = 0.$$

O discriminante da equação é dado por $\Delta = b^2 - 4ac$. O número de raízes reais depende do valor de Δ : se for nulo, há apenas uma raiz real; se for positivo, há duas raízes reais distintas; se for negativo, há duas raízes imaginárias.

Escreva um algoritmo completo que, a partir dos três coeficientes de uma equação de segundo grau, apresente quantas raízes reais ela possui. Elabore um algoritmo com detalhamento dos cálculos e decisões de fluxo. Assuma sempre $a \neq 0$.

3.5-12 Retome as descrições de notas e conceitos apresentadas no Exercício 3.5-4 e escreva um algoritmo completo que, a partir de duas notas (ambas no intervalo $[0, 10]$), calcule a média e apresente o conceito equivalente, detalhando a conversão da nota para conceito segundo a Tabela 3.5-1 e eliminando qualquer referência a ela.

3.5-13 Um sistema precisa realizar a conversão de unidades de velocidade de forma (um tanto quanto) genérica.

As conversões a serem realizadas são entre as unidades apresentadas na Tabela 3.5-4. A tabela indica o fator de conversão; para converter de m/s para milhas/h, basta multiplicar por 2,237, por exemplo.

Escreva um algoritmo para fazer as conversões necessárias, usando a seguinte codificação:

1. m/s
2. m/min
3. km/h
4. milhas/h

O algoritmo deve aceitar três valores: o código da unidade de entrada, o código da unidade de saída e o valor da velocidade.

Como exemplos, para converter de km/ para m/s, devem ser informados ao algoritmo os valores 3 e 1; de milhas/h para m/min, os valores 4 e 2. Seguido a estes dois valores, deve haver a velocidade a ser convertida. Como exemplo, a entrada com 2, 4, 1,2 deve fazer a conversão de 1,2m/min para milhas/h.

Tabela 3.5-5: Exemplos de alguns anos bissextos e não bissextos.

Ano	Múlt.4	Múlt.100	Múlt.400	Biss.
4	sim	não	não	sim
8	sim	não	não	sim
100	sim	sim	não	não
200	sim	sim	não	não
400	sim	sim	sim	sim
600	sim	sim	não	não
800	sim	sim	sim	sim
1992	sim	não	não	sim
1997	não	não	não	não
2000	sim	sim	sim	sim
2008	sim	não	não	sim
2010	não	não	não	não
2100	sim	sim	não	não
2900	sim	sim	não	não
3000	sim	sim	não	não
3004	sim	não	não	sim

Para este exercício, use exclusivamente a estrutura **if** para as verificações de condições e detalhe os cálculos e condições envolvidas.

3.5-14 Refaça o Exercício 3.5-13 usando exclusivamente a estrutura **switch** para tratar o fluxo condicional, detalhando os cálculos e condições envolvidas.

3.5-15 ★★ Um ano é considerado bissexto quando for múltiplo de quatro, exceto os múltiplos de 100, porém incluídos os múltiplos de 400. A Tabela 3.5-5 mostra uma lista de anos bissextos e não bissextos.

Escreva um algoritmo para, dado um ano (inteiro maior que zero), apresentar **true** para anos bissextos ou **false** para os demais anos. O algoritmo deve apresentar em detalhes os cálculos e verificações realizados.

Desafio: use apenas uma estrutura **if** e combine as diversas partes da verificação com **es** e **ous**.

3.5-16 ★★★ Dados os valores para dia, mês e ano, é preciso determinar se uma data é válida.


Uma data válida, composta por dia, mês e ano (todos inteiros), é aquela em que:

- o ano é diferente de zero, considerando-se que anos negativos indiquem a.C.;
- o mês pertence ao conjunto $\{1, 2, 3, \dots, 12\}$;
- o dia é de 1 a 31 para os meses 1, 3, 5, 7, 10 e 12;
- o dia é de 1 a 30 para os meses 4, 6, 9 e 11;
- o dia é de 1 a 28 para o mês 2 de ano não bissexto e de 1 a 29 para anos bissextos.

A definição para anos bissextos está no Exercício 3.5-15.

Escreva um algoritmo que, a partir de três valores inteiros quaisquer (respectivamente dia, mês e ano), determinem se eles formam uma data válida e apresente como saída **true** ou **false**. As verificações devem ser apresentadas em detalhes.

Desafio: Apresente seu algoritmo com apenas uma estrutura condicional **if**.

3.5-17  Um triângulo pode ser classificado, de acordo com seus ângulos internos, como:

- *acutângulo*, quando todos seus ângulos forem menores que 90° ;
- *retângulo*, quando um de seus ângulos internos for igual a 90° ;
- *obtusângulo*, quando um de seus ângulos internos for maior que 90° .

Sabe-se que a soma dos ângulos internos de um triângulo é sempre 180° e, a partir dos valores de dois ângulos internos de um triângulo, é possível classificá-lo adequadamente.

Escreva um algoritmo que resolva o problema exposto, dando como saída “acutângulo”, “retângulo” ou “obtusângulo”. Considere que somente serão informados ângulos válidos no contexto.

3.6 Estruturas de repetição

3.6-1 ★ Acompanhe os Algoritmos 3.6-1 a 3.6-3 e diga quais valores são apresentados pela execução de cada um.

Algoritmo 3.6-1 Apresentação de valores.

Output: Alguns valores

```

v ← 0
while v < 10 do
    write v
    v ← v + 1
end while

```

Algoritmo 3.6-2 Apresentação de valores.

Output: Alguns valores

```

v ← 0
while v < 10 do
    v ← v + 1
    write v
end while

```

3.6-2 ★ Acompanhe os Algoritmos 3.6-4 a 3.6-6 e diga quais valores são apresentados pela execução de cada um.

Algoritmo 3.6-3 Apresentação de valores.

Output: Alguns valores

```

v ← 10
while v > 0 do
    v ← v - 1
    write v
end while

```

Algoritmo 3.6-4 Apresentação de valores.

Output: Alguns valores

```

v ← 0
repeat
    write v
    v ← v + 1
until v = 10

```

Algoritmo 3.6-5 Apresentação de valores.

Output: Alguns valores

```

v ← 0
repeat
    v ← v + 1
    write v
until v = 10

```

Algoritmo 3.6-6 Apresentação de valores.

Output: Muitos valores

```

v ← 0
repeat
    v ← v + 3
    write v
until v = 10

```

3.6-3 ★★ Considere os Algoritmos 3.6-7 e 3.6-8. Para quais valores de entrada cada um deles não apresenta nenhuma saída?

Algoritmo 3.6-7 Apresentação de valores.

Input: Um valor $v \in \mathbb{Z}$

Output: Alguns valores

```

read v
while v > 0 do
    v ← v + 1
    write v
end while

```

Algoritmo 3.6-8 Apresentação de valores.

Input: Um valor $v \in \mathbb{Z}$

Output: Alguns valores

```

read v
repeat
    v ← v + 1
    write v
until v ≤ 0

```

3.6-4 Qual a saída produzida pelos Algoritmos 3.6-9 e 3.6-10?

Algoritmo 3.6-9 Apresentação de valores

Output: Alguns valores

```

for e ← 1 to 15 step 4 do
    write e
end for

```

Algoritmo 3.6-10 Apresentação de valores

Output: Alguns valores

```

for e ← -1 to 1 step 0.5 do
    write e
end for

```

3.6-5 (✓) Qual a saída produzida pelo Algoritmo 3.6-11?

Algoritmo 3.6-11 Apresentação de valores

Output: Alguns valores

```

a ← 10
b ← 1
for e ← a to b do
    write e
end for

```

3.6-6 Observe com atenção o Algoritmo 3.6-12, que assume a presença de um único funcionário na loja. Você está convencido que a condição da repetição **while** está correta? Ou a condição seria “há cliente e não deu o horário de fechamento”? Qual seria a diferença entre usar “e” e “ou”?

Algoritmo 3.6-12 Atendimento em estabelecimento comercial.

```

1► Abra o caixa
2► Abra o estabelecimento
3► while há cliente ou não deu o horário de fechamento do
4►     if deu o horário de fechamento then
5►         Feche o estabelecimento
6►     end if
7►     if há cliente na loja then
8►         Atenda o próximo cliente
9►     end if
10► end while
11► if o estabelecimento está aberto then
12►     Feche o estabelecimento
13► end if
14► Feche o caixa

```

3.6-7 Por que o **if** da linha 11 do Algoritmo 3.6-12 é necessário?

3.6-8 Haveria a necessidade do **if** da linha 11 do Algoritmo 3.6-12 caso a verificação da linha 4 viesse depois da da linha 7 (i.e., a ordem dos **ifs** fosse trocada)?

E qual seria a implicação dessa alteração na lógica geral?

3.6-9 Considere que uma empresa tenha entregue relatórios para os últimos 12 meses. Cada relatório contém os seguintes dados:

- Número do mês e ano;
- Uma lista das compras e vendas do mês, transação a transação, com valores (em R\$) positivos para as vendas e negativos para as compras, em ordem cronológica.

Deseja-se obter, a partir desses relatórios, as seguintes informações:

- A quantidade de meses em que houve mais vendas do que compras;
- O mês e ano em que houve maior déficit;
- O mês e ano em que houve maior superavit.

O Algoritmo 3.6-13 é uma solução para o problema? Há deficiências?

3.6-10 ★ No Algoritmo 3.6-13 (veja detalhes no Exercício 3.6-9), as duas últimas instruções incluem a ressalva “se existirem”. Por que essa ressalva é importante? Em que condições um mês e ano de déficit ou superavit não existem? Há a possibilidade de não

Algoritmo 3.6-13 Levantamento de dados sobre vendas e compras.

```
for each relatório do
    Obtenha o balanço do mês somando os valores da lista de compras e vendas
    if o balanço for positivo then
        Conte o mês atual como mês de superavit
    end if
    if o balanço atual for menor que o menor balanço registrado e foi deficit then
        Registre o balanço atual como o menor balanço
        Registre o mês e ano do relatório como o mês de maior deficit
    end if
    if o balanço atual for maior que o maior balanço registrado e foi superavit then
        Registre o balanço atual como o maior balanço
        Registre o mês e ano do relatório como o mês de maior superavit
    end if
end for
Apresente a contagem de meses com superavit
Apresente o mês e ano do maior superavit se existirem
Apresente o mês e ano do maior deficit se existirem
```

existirem mês e ano de deficit e superavit simultaneamente?

3.6-11 ★★★★★ Um leilão corresponde à oferta de um bem de valor v a uma plateia de potenciais compradores. Indivíduos interessados no bem fazem lances cada vez maiores e quem fizer o maior lance, compra o bem. No caso de um lance vencedor de valor b , o ganho do comprador é $v - b$.

Em um leilão específico, o leiloeiro apresenta o bem e estabelece um lance inicial mínimo de 10% do valor ofertado, começando então as rodadas de lances. A cada rodada, o leiloeiro pergunta “*quem dá mais?*” e há duas possibilidades: (a) um interessado faz uma oferta de valor b ou (b) não há lances. O leilão se encerra após uma rodada sem lances, vencendo quem fez a maior oferta ou não havendo venda se ninguém fez qualquer lance.

Escreva um algoritmo estruturado de nível alto que simule a oferta de um bem em leilão, desde a apresentação até o término, apresentando ao final o valor do lance ganhador ou indicando a ausência de interessados. Considere que todo o processo seja bem comportado e que, portanto, não há situações atípicas, como mais de um lance em uma rodada, um lance menor que o anterior, desistências durante o processo ou outras.

3.6-12 ★★★★★ Uma escola de idiomas na Nova Zelândia é especializada no ensino da língua inglesa para estrangeiros, usando a estratégia de “imersão”, sendo que os estudantes são levados a apenas usar o inglês para se comunicar durante todo o período de estadia.

É preciso determinar, para fins de planejamento da escola, quais são as origens de todos os atuais estudantes, indicadas pela combinação do nome da cidade e do país (exemplo: São Carlos/Brazil).

O relatório com os resultados deve ser apresentado na forma de uma lista contendo a cidade/país de origem e o número de alunos provenientes desta localização. A ordem de apresentação das cidades é irrelevante.

Escreva um algoritmo estruturado que, a partir da consulta da origem de cada estudante, apresente a lista com as quantidades. Apresente uma solução de nível mais alto, sem entrar demasiadamente nos detalhes, mas que organize corretamente as ações para apresentar as informações solicitadas.

3.6-13 ★★★ Um restaurante compra azeite de qualidade em tonéis e os transfere para pequenas garrafas para serem usadas nas mesas dos clientes. Para um controle de qualidade, nenhuma garrafa tem seu conteúdo completado com azeite novo, de forma a não misturar óleo novo com mais velho.

No final do expediente, todas as garrafas de azeite são colocada ao longo de uma estante, sem qualquer ordem específica. Essa estante permite apenas que haja uma garrafa ao lado da outra, formando uma grande fila.

Um funcionário é encarregado de separar as garrafas parcialmente usadas das cheias (intocadas), mantendo-as ainda na estante. O trabalho é feito manualmente e exclusivamente por um único empregado. Ele não tem outros recursos (como uma caixa ou mesa de apoio), de forma que, no máximo, ele consegue segurar apenas duas garrafas por vez, uma em cada mão. Ele também não tem como mover as garrafas se estiver segurando outras.

Escreva um algoritmo estruturado que, se seguido à risca pelo funcionário, faça com que todas as garrafas usadas fiquem à esquerda e todas as novas à direita na estante, instruindo-o a tirar e por as garrafas na estante. Assuma que o funcionário seja capaz de se lembrar de algumas coisas relativamente simples, como qual a posição da última garrafa que colocou na estante, por exemplo.

3.6-14 Em um estudo, é formado um grupo de 100 pessoas. Dentre vários dados, é preciso saber a quantidade de pessoas em cada faixa etária e a porcentagem de pessoas na quinta faixa etária em relação ao total de pessoas. As faixas estão definidas na Tabela 3.6-1.

Escreva um algoritmo de nível alto para resolver o problema e considere que serão fornecidas como entradas as idades de cada sujeito.

Tabela 3.6-1: Faixas etárias.

Faixa etária	Idade
1	Até 15 anos
2	De 16 a 25 anos
3	De 26 a 40 anos
4	De 41 a 55 anos
5	Maiores de 56 anos

3.6-15 ★★★ Em um estudo, são formados 20 grupos (numerados de 1 a 20), cada um com pelo menos 100 pessoas cada um e nunca vazios. Para cada grupo é informada a quantidade de pessoas no grupo, seguida das idades de cada uma.

É preciso saber informações relativas a faixas etárias e idades. As faixas estão definidas na Tabela 3.6-1.

Dados importantes que precisam ser levantados para o estudo:

- A média geral das idades;
- A maior amplitude de idades dentre os grupos (diferença entre a máxima e a mínima);
- O número do grupo com maior idade média;
- A porcentagem do número de pessoas na primeira faixa etária em relação ao total de pessoas no estudo;
- A faixa etária com menor número total de pessoas.

Escreva um algoritmo de nível alto que resolva este problema, apresentando os dados listados.

3.6-16 Execute o Algoritmo 3.6-14 para cada uma das seguintes sequências:

- $\langle 4, 16, -3, 8, 3, -20, 18 \rangle$
- $\langle \rangle$ (sequência vazia)
- $\langle 10, 20, 30, 40 \rangle$
- $\langle 40, 30, 20, 10 \rangle$

Algoritmo 3.6-14 Determinação do valor máximo.

Input: Uma sequência de valores inteiros quaisquer

Output: O valor máximo dessa sequência ou $-\infty$ se a sequência for vazia

Defina $máx$ como $-\infty$

```

for each valor  $n$  da entrada do
  if  $n > máx$  then
    Redefina  $máx$  com o valor de  $n$ 
  end if
end for

```

Apresente $máx$

3.6-17 Execute o Algoritmo 3.6-15 para as seguintes entradas (o primeiro valor é sempre o número de idades contidas na sequência):

- 3 $\langle 15, 25, 16 \rangle$

- 5 $\langle 18, 19, 25, 17, 13 \rangle$

- 1 $\langle 20 \rangle$

Algoritmo 3.6-15 Determinação do valor médio de idades, dados a quantidade de amostras e os valores individuais.

Input: A quantidade n , $n > 0$ de valores em uma sequência de idades seguida dos valores $\langle c_1, c_2, \dots, c_n \rangle$. $n, c_i \in \mathbb{N}^*$

Output: O valor médio das idades $m \in \mathbb{R}$

Obtenha o número de amostras n

Defina a soma s com 0 ▷ inicia o acumulador
for $i \leftarrow 1$ **to** n **do**

Obtenha um valor c da sequência

Some a s o valor de c ▷ acumule cada amostra

end for

Calcule m como $\frac{s}{n}$ ▷ cálculo da média

Apresente o valor m

3.6-18 Qual a diferença entre as entradas dos Exercícios 3.6-16 e 3.6-17? Indique o impacto dessa diferença na elaboração da solução algorítmica.

3.6-19 Escreva um algoritmo completo para determinar o valor mínimo de uma sequência (possivelmente vazia) de números inteiros. O algoritmo deve apresentar o valor mínimo ou o valor $+\infty$ caso não haja valores de entrada.

3.6-20 ★★★ Escreva um algoritmo completo para determinar simultaneamente os valores mínimo e máximo de uma sequência (possivelmente vazia) de valores numéricos. O algoritmo deve apresentar os valores mínimo e máximo ou então $-\infty$ e $+\infty$ caso não haja valores de entrada.

3.6-21 ☑ Um sistema de controle de temperatura registra, minuto a minuto, a temperatura de um sensor. Cada temperatura é registrada em graus Celsius ($^{\circ}\text{C}$) com casas decimais. Os registros são contínuos (i.e., não terminam nunca) e os valores são sempre maiores que zero.

O sistema de registro, de acordo com algumas regras internas, registra artificialmente na sequência de temperaturas o valor zero (0). Esse registro não é uma temperatura, mas apenas um marcador.

Escreva um algoritmo que determine, dada uma sequência de temperaturas registradas de acordo com a descrição, quantas das temperaturas são superiores a 25°C . Devem apenas ser interpretadas as temperaturas até a primeira marcação nula, sendo as demais ignoradas.

Por exemplo, o algoritmo deve dar 5 como resposta para a sequência

(24.73, 25.87, 25.95, 25.20, 24.74, 26.19, 24.93,
26.19, 0, 27.25, 28.51, 28.37, 0, 29.00, 27.52 ...)

O algoritmo já deve terminar ao encontrar a primeira ocorrência de zero.

3.6-22 Existem registros dos valores de fechamento do câmbio do dólar para cada dia do ano. Para os dias sem fechamento oficial (fins de semana e feriados, por exemplo) é replicado no registro o valor do último fechamento, perfazendo sempre 365 valores registrados. Anos bissextos ou outras situações atípicas devem ser ignoradas.

Escreva um algoritmo que determine para uma sequência de registros como o descrito seu valor médio, apresentando esse resultado.

3.6-23 ★★★★★ Considere os registros de câmbio descritos no Exercício 3.6-22 e escreva um algoritmo para determinar o número de vezes em que a alta foi superior a 5% entre dois dias consecutivos.

3.6-24 Considere uma entrada de dados em que, inicialmente, é apresentada a quantidade de valores que serão fornecidos e, em seguida, cada um dos valores. A quantidade é sempre maior ou igual a zero e cada valor individual é um número real qualquer.

Escreva um algoritmo completo para, a partir desses dados, determinar os valores máximo e mínimo. No caso da entrada ser “vazia” (i.e., se a quantidade indicada for zero), nenhuma saída deve ser apresentada.

3.6-25 ★★★★★ Um número N é dito primo se ele for divisível apenas por ele e pela unidade. Por esta definição, são primos: 1, 2, 3, 5, 7, 11, por exemplo.

Escreva um algoritmo para, dado um valor inteiro qualquer, determinar se ele é ou não primo. A saída produzida deve ser **true** ou **false**.

3.6-26 ★★★ A água possui algumas características interessantes. Ela tem menor densidade no estado sólido (0°C ou menos) que no estado líquido; indo de 0 a 4°C , há aumento da densidade e, depois dos 4°C , a densidade vai sendo reduzida com o aumento da temperatura.

Escreva um algoritmo que, a partir de uma sequência de temperaturas em Celsius, determine e apresente:

- Quantas das temperaturas são inferiores a 0°C (gelo);
- Quantas das temperaturas estão no intervalo de 0 a 4°C ;
- A média de todas as temperaturas (ou nada, caso a sequência seja vazia).

3.6-27 ★★★★★ O problema da “sub-lista contígua de soma máxima” (ou “segmento de soma máxima”) é descrito como se segue, de forma adaptada.

Deve-se considerar uma sequência de valores reais, contendo número negativos, positivos ou nulos. É preciso calcular o valor da maior soma possível considerando-se apenas valores consecutivos e apenas somas maiores ou iguais a zero.

Por exemplo, para a sequência abaixo¹, a maior soma é 72, somando-se do valor 20 até o 3 (inclusive).

(10, 5, -17, 20, 50, -1, 3, -30, 10)

Considere uma entrada de dados formada pelo número de dados de uma sequência (valor inteiro e positivo), seguido dos valores da sequência.

Escreva um algoritmo para apresentar a maior soma a partir dos dados de entrada.

3.6-28 ★ Na universidade existe uma lista com o número de matrícula (RA) e o índice de rendimento acadêmico (IRA) de todos os alunos. Dispõe-se também do número total de alunos. É preciso calcular o IRA médio de todos e reportar o número de alunos que estão abaixo de 60% do IRA médio.

O que esse algoritmo exige que torna todas as abordagens de processamento vistas até agora insuficientes para uma solução?

3.6-29 O Algoritmo 3.6-16 determina, para uma entrada de 10 valores inteiros, qual o valor máximo e se há pelo menos um valor par entre eles. A solução não está correta. Por quê?

3.7 Refinamentos sucessivos

3.7-1 Uma empresa de *marketing* pretende fazer uma campanha para venda de produtos. O público alvo são os torcedores de uma partida de futebol em um grande estádio e campanha será feita junto ao portão de saída de apenas uma das equipes.

Ao longo da partida, uma equipe da empresa fica encarregada de monitorar a transmissão feita pela televisão, dando pontos positivos e negativos a cada time, de acordo com o apoio da torcida a seu time (como incentivos, vaias, xingamentos etc.). A equipe que obteve o melhor saldo (pontos positivos menos negativos) é a escolhida para a campanha.

Como a montagem da propaganda no portão de saída deve ser rápida, o tempo entre a escolha do time e a preparação para a abordagem aos torcedores deve ser o mínimo possível.

O Algoritmo 3.7-1 ilustra uma solução para se decidir onde a campanha será realizada. Verifique se é adequado.

¹Fonte: Wikipedia (https://pt.wikipedia.org/wiki/Sublista_cont%C3%ADgua_de_soma_m%C3%A1xima).

Algoritmo 3.6-16 Determinação de valor máximo e existência de pares para 10 valores inteiros.

Input: Uma sequência de 10 valores inteiros

Output: O valor máximo da sequência e a existência de pares (**true** ou **false**)

```
▷ Primeiro valor
read v
vmáx ← v
if v é par then
    cpar ← true
else
    cpar ← false
end if

▷ Demais valores
for i ← 2 to 10 do
    read v
    if v > vmáx then
        vmáx ← v
    end if
    if v é par then
        cpar ← true
    else
        cpar ← false
    end if
end for

▷ Resultado
write vmáx, cpar
```

Algoritmo 3.7-1 Determinação da equipe que será escolhida para uma campanha publicitária.

Input: A transmissão televisiva de uma partida de futebol ao vivo

Output: O nome da equipe escolhida para a campanha

Acompanhe a transmissão da partida até o final
Identifique os pontos positivos da primeira equipe
Identifique os pontos negativos da primeira equipe
Identifique os pontos positivos da segunda equipe
Identifique os pontos negativos da segunda equipe

Contabilize o saldo da primeira equipe
Contabilize o saldo da segunda equipe

Informe a equipe com maior saldo

3.7-2 Uma empresa compra peças avulsas no exterior, monta o equipamento e o vende, também para o exterior. Nem sempre as vendas andam bem e a empresa tem que variar a quantidade de importação frente ao que é exportado.

É preciso determinar o número de meses em que essa empresa exportou mais do que importou, a partir de uma sequência de relatórios mensais. Cada relatório contém o valor, em dólares, de cada importação feita e cada exportação realizada em um dado mês, por ordem de data. Cada operação é identificada por seu tipo (importação ou exportação) e seguida pelo seu valor. Não há totais nem qualquer outra informação no relatório.

O Algoritmo 3.7-2 apresenta uma solução de nível mais alto para o problema. Refine os passos, aumentando o nível de detalhe da solução. Se julgar apropriado, reestruture o próprio Algoritmo 3.7-2.

Algoritmo 3.7-2 Contagem de meses em que houve maior volume de exportações que de importações

Input: Uma sequência de relatórios mensais com os valores das operações de importação e exportação

Output: O número de relatórios em que o total de exportações foi maior que o de importações

```
for each relatório disponível do
    Calcule a soma das exportações do relatório
    Calcule a soma das importações do relatório
    Conte o mês do relatório se a diferença das
        exportações e importações for positiva
end for
```

Apresente o número de meses contados

3.7-3 ★★★★★ Os alunos de computação se reuniram para uma palestra sobre novas tecnologias para *smartphones*. Grande parte dos presentes ficou animada com a palestra, tanto que, depois de terminada, permaneceram no local para conversar.

A determinada altura, a conversa foi para quais aparelhos eram os mais comuns entre os presentes (e.g., Moto G, Moto E, iPhone 7, Galaxy J7, One Touch U3...). O grupo então, se motivou a contabilizar, entre eles, quais eram os *smartphones* que tinham e a quantidade de cada um.

Apresente uma proposta de solução para o problema seguinte na forma de um algoritmo de nível mais alto. Expanda algumas partes, aumentando o nível de detalhamento quando possível.

3.7-4 ★★★★★ Analise o Algoritmo 3.7-3, comparando-o com o Algoritmo 3.7-2 (veja Exercício 3.7-2) e tentando entender qual a falha que ele apresenta.

Algoritmo 3.7-3 Contagem de meses em que houve maior volume de exportações que de importações. Solução incorreta.

Input: Uma sequência de relatórios mensais com os valores das operações de importação e exportação

Output: O número de relatórios em que o total de exportações foi maior que o de importações

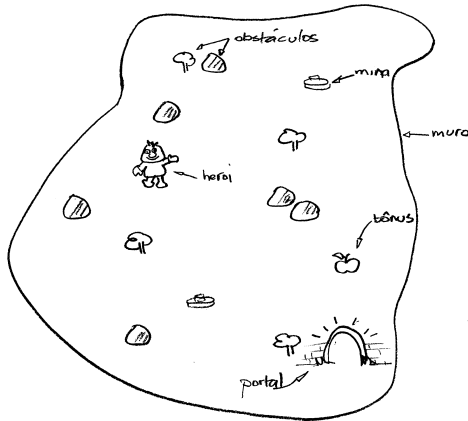
```

Inicie a soma das exportações com zero
Inicie a soma das importações com zero
for each relatório disponível do
    Acumule as exportações do relatório à soma de exportações
    Acumule as importações do relatório à soma de importações
    Conte o mês do relatório se a diferença da soma de exportações e da de importações for positiva
end for

```

Apresente o número de meses contados

Figura 3.7-1: Rascunho de um jogo simples.



3.7-5 ★★★★★ Uma empresa está desenvolvendo um jogo simples para dispositivos móveis, ilustrado na Figura 3.7-1.

O personagem pode se deslocar apenas em movimentos simples: para cima, para baixo, para a esquerda e para a direita (C, B, E e D), sem movimentos diagonais. A cada entrada, ele tenta dar um único passo na direção indicada.

Não há controle de tempo da partida e o personagem inicia o jogo com 100% de vida.

O espaço para o personagem atuar é em duas dimensões e consiste em um terreno cercado por muros. Nesse espaço existem obstáculos fixos (e.g., rochas, árvores, muro) que impedem o movimento, estando visíveis na tela. Quando se tenta mover o personagem na direção de um obstáculo, ele fica no mesmo lugar.

Há também objetos fixos invisíveis, na forma de minas escondidas no solo. A posição só é revelada se o personagem andar sobre uma delas. Quando ele ocupa a mesma posição de uma mina, há um decréscimo de 10pp (pontos percentuais) de vida.

Durante o jogo, aparecem objetos bônus que, se tocados, fortalecem o personagem com 15pp de vida, porém limitados aos 100%. Os bônus aparecem em posições aleatórias e por um tempo finito.

O objetivo do jogo é levar o personagem até um portal de escape que existe em algum lugar do terreno. A posição da saída é visível o tempo todo.

O jogo pode terminar por duas razões: (1) quando o personagem chega ao portal ou (2) quando a vida chega a 0%.

Escreva um algoritmo que gerencie uma partida do jogo, indicando ao final se houve sucesso (“YOU WIN!”) ou fracasso (“YOU LOSE!”). Proponha o algoritmo usando a técnica de refinamentos sucessivos, detalhando progressivamente sua solução.

Compartilhe sua solução com colegas e discutam as estratégias e dificuldades encontradas.

3.7-6 ★★★★★ Refaça os exercícios seguintes indicando em detalhes os cálculos efetuados:

- No Exercício 3.3-9, apresente todos os cálculos e somas explicitamente;
- No Exercício 3.5-4, detalhe as conversões média-conceito e conceito-média com todas as verificações necessárias no Algoritmo 3.5-4.

3.8 Variáveis compostas

3.8-1 Execute o Algoritmo 3.8-1 para cada uma das seguintes entradas:

- (2, 2, 2)
- (4, 1, 2)
- (-2, 0, 1)
- (-1, 0, 2)

Algoritmo 3.8-1 Determinação da distância de um ponto no espaço à origem.

Input: Um ponto $P = (x, y, z) \in \mathbb{R}^3$

Output: Sua distância $d \in \mathbb{R}^+$ à origem

read P

$d \leftarrow \sqrt{x^2 + y^2 + z^2}$

write d

3.8-2 Execute o Algoritmo 3.8-2 para as seguintes entradas:

- (2, 2, 2) e (4, 1, 2)
- (-2, 0, 1) e (-1, 0, 2)

3.8-3 ☑ Por que há diferença na notação usada para as coordenadas x , y e z dos pontos entre os Algoritmos 3.8-1 e 3.8-2?

Algoritmo 3.8-2 Determinação do ponto mais próximo da origem, sendo irrelevante a resposta no caso de equidistantes.

Input: Dois pontos P_1 e P_2 em \mathbb{R}^3

Output: O ponto mais próximo da origem

```

read  $P_1, P_2$ 

 $d_1 \leftarrow \sqrt{P_1.x^2 + P_1.y^2 + P_1.z^2}$ 
 $d_2 \leftarrow \sqrt{P_2.x^2 + P_2.y^2 + P_2.z^2}$ 
if  $d_1 \geq d_2$  then
     $R \leftarrow P_1$ 
else
     $R \leftarrow P_2$ 
end if

write  $R$ 
```

3.8-4 Escreva um algoritmo para apresentar, para uma sequência de 10 pontos em \mathbb{R}^2 , o mais próximo a um outro ponto de referência R .

3.8-5 Qual o resultado apresentado pelo Algoritmo 3.8-3? Há dúvidas de interpretação?

Algoritmo 3.8-3 Processo seletivo com dados de candidatos entregues na ordem de término das provas. Os empates são decididos em favor do que terminou antes.

Input: Uma sequência de dados sobre candidatos em registros ($nome, nota_1, nota_2$), com $0 \leq nota_i \leq 10$

Output: O nome do candidato escolhido

```

 $p_{m\acute{a}x} \leftarrow 0$  ▷ pontuação máxima
for each candidato  $c$  da entrada do
     $p \leftarrow 0.2nota_1 + 0.8nota_2$  ▷ pontuação
    if  $p > p_{m\acute{a}x}$  then
         $p_{m\acute{a}x} \leftarrow p$ 
         $c_{sel} \leftarrow c$  ▷ registra selecionado
    end if
end for
```

Apresente o nome de c_{sel}

O algoritmo deve apresentar, ao final, o numerador e denominador corretamente estruturados.

3.8-6 ★★ Um sistema controla a tensão em uma linha de alta potência. Ele faz o registro da tensão (em kV) uma vez a cada minuto, perfazendo 1440 leituras por dia.

Escreva um algoritmo para, a partir das leituras de um dia completo, determinar e apresentar:

- A tensão média (v_{md});
- A porcentagem de leituras que ficaram acima de $1,5v_{md}$.

3.8-7 ★★ Considere o problema do Exercício 3.3-9, que propõe determinar o número mínimo de cédulas e moedas para uma dada quantidade monetária.

O Algoritmo 3.8-4 apresenta uma solução usando uma sequência (e.g., vetor) como variável composta. Compare esta solução com a que fez para Exercício 3.3-9.

Algoritmo 3.8-4 Determinação do número mínimo de cédulas e moedas para uma dada quantidade em reais (R\$).

Input: Um valor monetário $v \in \mathbb{R}^+$ expresso com até duas casas decimais (centavos)

Output: O número $n \in \mathbb{N}$ de cédulas ou moedas mínimo que equivale ao valor v

```

Obtenha  $v$ 
 $n \leftarrow 0$ 
for each  $u$  em
     $\langle 100, 50, 20, 10, 5, 2, 1, 0.50, 0.25, 0.10, 0.05, 0.01 \rangle$  do
         $n_u \leftarrow \lfloor v/u \rfloor$  ▷ quantidade de cédulas/moedas
         $n \leftarrow n + n_u$  ▷ acumula
         $v \leftarrow v - n_u u$  ▷ calcula o que ainda resta
    end for
Apresente  $n$ 
```

3.8-8 Escreva um algoritmo para, dados 100 valores inteiros positivos quaisquer (\mathbb{Z}^{+*}), apresentar primeiro todos os pares e depois todos os ímpares. Entre os pares e os ímpares não é preciso manter a ordem de entrada.

3.8-9 Escreva um algoritmo para, dados 100 valores inteiros positivos quaisquer, apresentar inicialmente o valor mínimo e depois os demais 99 valores, em qualquer ordem. Havendo múltiplas instâncias do mínimo, apenas uma delas deve ser apresentada no início e as demais juntamente com os demais, em qualquer ordem.

3.9 Modularização

3.9-1 ★ Você entende o Algoritmo 3.9-1? Quais os resultados esperados para as instruções considerando-se a função especificada?

```

write RAIZEQUAÇÃO(5, -8)
write RAIZEQUAÇÃO(-7.2, 0)
write RAIZEQUAÇÃO(-9, 3)
write RAIZEQUAÇÃO(0, 2)
```

3.9-2 ☑ ★★ Escreva uma função para retornar o valor máximo entre dois valores reais. Defina com clareza também as entradas e saídas.

Algoritmo 3.9-1 Código exemplo.

Input: Os coeficientes a e b ($a, b \in \mathbb{R}$) de uma equação linear $ax + b = 0$

Output: ∞ , se $a = 0$, ou a raiz $x \in \mathbb{R}$ da equação, caso contrário

function RAIZEQUAÇÃO(a, b)

if $a \neq 0$ **then**

$x \leftarrow -b/a$

else

$x \leftarrow \infty$

end if

return x

end function

3.9-3 ✓ Escreva uma função para retornar o valor máximo entre três valores reais. Sua solução deve usar, necessariamente a função desenvolvida no Exercício 3.9-2.

3.9-4 ★★ A função **abs** é usada em algumas linguagens para indicar o “valor absoluto” de um número, ou seja, seu módulo:

$$\text{abs } x = |x|.$$

Suponha que, por qualquer razão, não existisse o módulo de um valor na matemática. Escreva uma função **ABS**(v) que tenha esse comportamento, ou seja, que retorne o valor positivo apenas.

3.9-5 ★★★ Um número perfeito n é um número natural (\mathbb{N}) tal que n é igual à soma de seus divisores, exceto ele mesmo. Por exemplo, 6 é perfeito, pois $6 = 1 + 2 + 3$. (Veja Exercício 3.3-15.)

Escreva a versão algorítmica da função **PERFEITO**(k), que retorna **true** se k for um número perfeito ou **false**, caso contrário.

4 Programação

4-1 Em programas em C, podem ser encontradas variações para a função **main**, como ilustrado na sequência. Pesquise e tente entender a diferença entre elas.

```
int main()
int main(void)
int main(int argc, char *argv[])
void main()
void main(void)
```

4-2 Escreva um programa em C simples e o execute. Tente, então, alterar o nome da função **main** para outra coisa qualquer, como **principal** ou **mymain**. Interprete a mensagem de erro apresentada e justifique porque o nome da função não pode ser mudado.

4.1 Tipos de dados e constantes

4.1-1 Considere os tipos básicos comuns na computação, identifique o tipo de cada um dos valores no Quadro 4.1-1, assinalando com \times a coluna correta.

Quadro 4.1-1 Tipos das constantes: inteiros (I), reais (R), cadeias de caracteres (C) e lógicos (L). O símbolo \square é usado para indicar explicitamente um espaço em branco.

Valor	I	R	C	L
25				
170				
25.				
25.2				
''Peter''				
31415				
''120''				
-16.5				
''1.2''				
true				
''''				
'''''				
''false''				
''False''				
0				

4.1-2 ★★ Em C, as constantes 603 e ''603'' representam coisas diferentes. Descreva, sucintamente, o que cada valor representa e o que os torna tão diferentes.

4.1-3 Os tipos de dados são voltados para representar valores do mundo real. Dadas as situações apresentadas no Quadro 4.1-2, assinale com \times a coluna que indica o tipo adequado para representar cada uma delas.

4.1-4 ✓ Considere a afirmação: “Valores inteiros são aqueles cuja parte fracionária é zero”. Esta afirmação está correta? Justifique.

4.1-5 O que significa a constante numérica 1.05E10 na linguagem C? Faça uma pesquisa por “notação científica”.

Dê a versão da notação científica em C para os seguintes valores:

- $6,02 \cdot 10^{23}$
- 0,000 000 000 000 01
- $3,25 \cdot 10^{-5}$
- $-5 \cdot 10^{-4}$

Quadro 4.1-2 Opção por tipo de dados: inteiros (I), reais (R), cadeias de caracteres (C) e lógicos (L).

Valor	I	R	C	L
O nome do mês atual				
A idade de uma pessoa				
O valor de um dos coeficientes de uma equação de segundo grau				
O ano de contratação de um trabalhador				
A velocidade média de um veículo				
Se o aluno foi aprovado após o término uma disciplina				
A quantidade de tubos de pasta de dente contido em um lote				
O número de alunos de uma turma				
Se o período de inscrição em um curso está atualmente aberto				
A rua de residência de um cliente				
O preço de um dado produto de uma loja				
O volume de produto contido num frasco de xampú				

4.1-6 ☑ Todas as opções de tipos de dados listadas são válidas na linguagem C. Assinale aquelas que correspondam a valores inteiros:

- ☐ int
- ☐ short int
- ☐ unsigned short int
- ☐ long long int
- ☐ char
- ☐ unsigned char

4.1-7 Preencha o quadro do Quadro 4.1-3 com o número de bytes usados para cada tipo. O Programa 4.1-1 apresenta o código em C que pode ser usado para a verificação, com poucas modificações.

Programa 4.1-1 Programa para apresentar o tamanho do tipo double.

```
/*
   Output: o numero de bytes do tipo
   double
*/
#include <stdio.h>

int main(){

    printf("%ld\n", sizeof(double));
    return 0;
}
```

Arquivo: material/sizeofdouble.c.

Quadro 4.1-3 Número de bytes para alguns tipos de dados em C.

Tipo	Tamanho
int	
long long int	
char	
unsigned char	
float	
double	

4.1-8 ☑ Por que a segunda linha do código seguinte gera um erro de compilação, enquanto a primeira é considerada correta?

```
int i = 070;
int j = 080;
```

4.1-9 O comando seguinte apresenta um valor do tipo int.

```
printf("valor = %d\n", 80);
```

Altere o formato de escrita %d para os valores seguintes, executando e observando os resultados. Experimente também alterar o valor da expressão incluindo números negativos.

- %i
- %o
- %x
- %X

4.1-10 O formato %d é usado para apresentar valores inteiros e %f para valores float. Pesquise e veja quais especificações de formato devem ser usadas com:

- short int
- unsigned int
- unsigned short int
- long long int
- double
- long double

4.1-11 ☑ Qual o tipo de cada uma das constantes seguintes, quando usadas em um programa em C?

- 10
- 10u
- 10l
- 10ul
- 010
- 010l
- '1'
- "10"

4.1-12 ✓ Considere a afirmação: “Se uma variável inteira foi declarada, mas ainda não houve leitura ou atribuição para ela, então seu valor é zero”. Ela está correta? Justifique.

4.1-13 Em C, um caractere texto é expresso 'A' e usa o tipo `char` para a variável. Qual é o tipo de uma variável e como deve ser expressa uma constante quando se trata de uma cadeia de caracteres?

4.1-14 Observe o Programa 4.1-2. O que pode dizer da saída que ele produzirá?

Programa 4.1-2 Comparação de literais.

```
/*
   Output: 0 resultado da funcao strcmp
   para duas cadeias de caracteres
   fixas
*/
#include <stdio.h>
#include <string.h>

int main(){
    char texto1[] = "pedro";
    char texto2[] = "paulo";

    printf("%d", strcmp(texto1, texto2));

    return 0;
}
```

Arquivo: material/comparaliteral.c.

4.1-15 Uma das formas de se criar uma variável com caracteres em C é a seguinte:

```
char nome[] = "karen";
```

Alternativamente, é possível usar o seguinte código (que requer o arquivo de cabeçalho `string.h`):

```
char nome[20];
strcpy(nome, "karen");
```

Em que uma forma difere da outra?

4.1-16 ★★★★★ Faça um programa para armazenar valores arbitrários em duas variáveis inteiras. Na sequência, faça a troca dos valores dessas variáveis, um com o da outra. Apresente os valores de ambas as variáveis tanto antes quanto depois da troca.

4.1-17 A linguagem C permite que a iniciação das variáveis seja feita juntamente com as declarações.

Por exemplo, pode-se escrever:

```
int i = 10, j = 11;
printf("%d e %d\n", i, j);
```

Teste declarações com iniciação para variáveis de outros tipos.

4.2 Expressões

4.2-1 Qual a função do operador `%` na linguagem C? A que tipo de dados pode ser aplicado?

4.2-2 Considere as expressões na sequência.

```
v1 + v2
v1 + v2/2
(v1 + v2)/2
0.5 * (v1 + v2)
2 * v1 + 3 * v2
v1/v2
(10 * v1) % v2
```

Calcule o resultado de cada uma delas para as seguintes declarações:

```
int v1, v2;
• v1 = 8;
  v2 = 11;

float v1, v2;
• v1 = 8;
  v2 = 11;
```

Codifique um programa em C para verificar seus resultados.

4.2-3 ✓ Observe o Programa 4.2-1 e diga porque o programa escreve 0 (i.e., falso).

Programa 4.2-1 Operações comutativas?

```
/*
   Output: Resultado da comparacao
   3/2 * 0.75 = 0.75 * 3/2
*/
#include <stdio.h>

int main(){

    printf("%d\n", 3/2*0.75 == 0.75*3/2);
    return 0;
}
```

Arquivo: material/divisaointeira.c.

4.2-4 Codifique em C as seguintes expressões, considerando todas as variáveis reais (cada letra é uma variável distinta):

$$a \frac{b+c}{d} \quad \frac{a+b}{c} \quad \sqrt{\frac{a}{b+c}} \quad a \ln b$$

$$|a-b| \quad \frac{|a-b|}{c} \quad \left| \frac{a-b}{c} \right|$$

Implemente as expressões em um programa e teste-as com valores diversos atribuídos às variáveis.

4.2-5 Escreva um programa para ler duas notas, ambas no intervalo de 0 a 10 e calcular e apresentar a média delas.

Teste-o para alguns valores e inclua as seguintes notas: 6,0 e 5,95.

4.2-6 Codifique o Programa 4.2-2 e o execute. A saída é a que você esperava? Por que a comparação de igualdade resulta em falso?

Programa 4.2-2 Comparação de valores reais.

```
/*
   Output: resultado da comparacao de
   igualdade do valor da variavel
   f com 3.77
*/
#include <stdio.h>

int main(){
    float f;

    f = 3.77;
    printf("%d\n", f == 3.77);

    return 0;
}
```

Arquivo: material/precisaofloat.c.

4.2-7 Os operadores ++ e - podem ser prefixados ou pós-fixados a uma variável. Porém, é preciso algum cuidado no uso deles para evitar o chamado *comportamento indefinido* (*undefined behaviour*, ou UB).

Codifique o Programa 4.2-3 e o execute. Teste algumas alterações do uso dos incrementos e acrescente decrementos. Dois testes interessantes com comportamento indefinido:

```
i = i++;
i = i++ * i++;
```

4.3 Estruturas condicionais

4.3-1 Implemente em C o Algoritmo 4.3-1 e teste com alguns valores para os coeficientes. Lembre-se:

- O coeficiente do termo quadrático (a) nunca é nulo;
- Forneça valores para os quais você consiga verificar o resultado.

4.3-2 Codifique o Programa 4.3-1 e teste-o para os valores de entrada 6,01 e 5,95. O que está errado com a saída apresentada? Como contornaria o problema, mantendo ainda a nota apresentada com apenas uma casa decimal?

Programa 4.2-3 Testes com o operador ++.

```
/*
   Output: valores de incrementos com ++i e i++
*/
#include <stdio.h>

int main(){
    int i;

    i = 10;
    printf("i = %d\n", i);

    i = 10;
    i++;
    printf("i = %d\n", i);

    i = 10;
    printf("i++ = %d\n", i++);
    printf("i = %d\n", i);

    i = 10;
    printf("++i = %d\n", ++i);
    printf("i = %d\n", i);

    i = 10;
    printf("i = %d, i++ = %d, ++i = %d\n",
           i, i++, ++i);

    return 0;
}
```

Arquivo: material/incrementos.c.

Algoritmo 4.3-1 Apresentação das raízes reais de uma equação de segundo grau.

Input: Os coeficientes a , b e c de uma equação de segundo grau, $a \in \mathbb{R}^*$ e $b, c \in \mathbb{R}$

Output: Os valores das duas raízes reais r_1 e r_2 caso existam ou nada, caso não haja raízes reais

Obtenha a, b, c

$\Delta \leftarrow b^2 - 4ac$

if $\Delta \geq 0$ **then**

$r_1 \leftarrow \frac{-b - \sqrt{\Delta}}{2a}$

$r_2 \leftarrow \frac{-b + \sqrt{\Delta}}{2a}$

Apresente r_1, r_2

end if

Programa 4.3-1 Cálculo da média.

```
/*
   Input: duas notas reais, de 0.0 a 10.0
   Output: a média das notas e mensagem
           "aprovado" ou "reprovado"
*/
#include <stdio.h>

int main(){
    float nota1, nota2, media;

    scanf("%f %f", &nota1, &nota2);
    media = (nota1 + nota2)/2;

    printf("media = %.1f", media);

    if(media >= 6)
        printf(" - aprovado\n");
    else
        printf(" - reprovado\n");

    return 0;
}
```

Arquivo: material/medianotas.c.

4.3-3 Considere o Algoritmo 4.3-2 e preencha o Quadro 4.3-1 com os valores esperados de saída, dadas as entradas indicadas. Use um traço (—) para indicar que não há saída.

Faça a implementação desse algoritmo em C e verifique se as saídas produzida coincidem com as do quadro.

Quadro 4.3-1 Entradas e saídas para o Algoritmo 4.3-2.

<i>a</i>	<i>b</i>	<i>c</i>	Saída
1	2	3	
1	2	2	
−1	2	3	
−1	3	3	

Algoritmo 4.3-2 Algoritmo com condicionais.

Input: Três valores $a, b, c \in \mathbb{R}$

Output: Uma letra ou nada, conforme as verificações

```
read a, b, c
if a > 0 then
    write "X"
    if b = c then
        write "Y"
    end if
end if
```

4.3-4 ★★★★★ Considere o Algoritmo 4.3-3 e preencha o Quadro 4.3-2 com os valores esperados de saída, dadas as entradas indicadas. Use um traço (—) para indicar quando não há saída.

Faça a implementação desse algoritmo em C e verifique se as saídas produzida coincidem com as do quadro.

Quadro 4.3-2 Entradas e saídas para o Algoritmo 4.3-3.

<i>a</i>	<i>b</i>	Saída
1	1	
−1	1	
1	−1	
−1	−1	

Algoritmo 4.3-3 Algoritmo com condicionais.

Input: Dois valores $a, b \in \mathbb{R}$

Output: Uma letra ou nada, conforme as verificações

```
read a, b
if a > 0 then
    if b > 0 then
        write "X"
    end if
else
    write "Y"
end if
```

4.3-5 O Programa 4.3-2 (com código fonte denominado `erroif.c`) produz, na compilação, a mensagem de erro seguinte. Qual o problema do código?

```
erroif.c:20:2: error: 'else' without a
previous 'if'
```

4.3-6 (✓) ★★★★★ Analise o Algoritmo 4.3-4 e tenha certeza que entende o que ele representa.

Feito isso, faça sua implementação na linguagem C.

4.3-7 ★ Em determinados sistemas, as datas são apresentadas pela concatenação do ano, do mês e do dia, nesta ordem, usando quatro dígitos para o ano e dois dígitos para dia e mês. Por exemplo, 22/4/1500 seria representado pelo valor inteiro 15000422.

Esse formato é interessante, pois permite comparar facilmente as datas. Por exemplo, 4/11/2015 é anterior a 30/12/2015, pois 20151104 < 20151230.

Considere duas datas válidas, cada uma expressa por três valores inteiros indicando dia, mês e ano. É preciso indicar se as datas são *iguais* (são exatamente a mesma data), ou é *anterior* (a primeira data é cronologicamente anterior à segunda) ou *posterior* (se a segunda preceder a primeira).

Escreva um programa em C que apresente como saídas “anterior”, “igual” ou “posterior” usando a estratégia de comparação indicada.

4.3-8 ★★★★★ Codifique em C os desenvolvimentos feitos para os exercícios ou algoritmos (a ordem indica tendência no aumento da complexidade de implementação):

- Exercício 3.5-11;
- Exercício 3.5-12;

Programa 4.3-2 Programa em C sem objetivo definido com erro de compilação.

```

1  /*
2   Input: Dois valores reais a e b
3   Output: A média dos dois valores e
4   uma indicação "bom" ou "ruim" ou nada
5  */
6  #include <stdio.h>
7  #include <stdlib.h> // para abs()
8
9  int main(){
10     double a, b, media;
11
12     scanf("%lf%lf", &a, &b);
13
14     media = (a + b)/2;
15
16     if(a < 0 || b < 0)
17         if(media > 6.0)
18             printf("bom\n");
19             media = abs(media);
20     else
21         printf("ruim\n");
22
23     printf("%g\n", media);
24
25     return 0;
26 }
```

Arquivo: material/erroif.c.

Algoritmo 4.3-4 Número de dias segundo o nome do mês, ignorando anos bissextos.

Input: O nome de um mês $m \in \{\text{"janeiro", "fevereiro", ..., "dezembro"}\}$

Output: O número de dias que o mês possui, ignorando-se anos bissextos

Obtenha o nome do mês m

switch m **of**

 case "janeiro", "março", "maio", "julho",
 "agosto", "outubro", "dezembro" **do**
 $n \leftarrow 31$

end case

 case "abril", "junho", "setembro",
 "novembro" **do**
 $n \leftarrow 30$

end case

 case "fevereiro" **do**
 $n \leftarrow 28$

end case

end switch

 Apresente n

▷ número de dias

- Algoritmo 3.5-10;
- Exercício 3.5-6;
- Exercício 3.3-9;
- Exercício 3.5-16.
- E outros que quiser fazer...

4.4 Estruturas de repetição

4.4-1 ✓ Codifique em C os Algoritmos 3.6-1 a 3.6-5, 3.6-9 e 3.6-10, verificando os valores apresentados. Use as estruturas **while**, **do-while** e **for**.

4.4-2 Codifique o Programa 4.4-1 e o execute. Remova o erro para que ele produza o resultado esperado.

Dica: A combinação de teclas **Ctrl-C** interrompe a execução do programa.

Programa 4.4-1 Programa com repetição simples e um erro de codificação.

```

/*
   Output: os valores de 1 a 10
*/
#include <stdio.h>

int main(){
    int i;

    for(i = 1; i <=10; i++){
        printf("%d ", i);
        printf("\n");
    }

    return 0;
}
```

Arquivo: material/comandovazio.c.

4.4-3 Considere o Programa 4.4-2. Codifique-o e execute o código gerado. A saída produzida é a esperada?

Dica: A combinação de teclas **Ctrl-C** interrompe a execução do programa.

Programa 4.4-2 Programa com repetição simples.

```

/*
   Output: A sequencia 0.3, 0.6, ..., 3.0
*/
#include <stdio.h>

int main(){
    float v;

    v = 0;
    while(v != 3){
        v += 0.3;
        printf("v = %.2f\n", v);
    }

    return 0;
}
```

Arquivo: material/problema_precisao.c.

4.4-4 Implemente o Algoritmo 4.4-1 em C e o teste.
Responda: quais os fatoriais de 10 e 15? É possível que $15! = 2.004.310.016$?

Algoritmo 4.4-1 Cálculo do fatorial de n .

Input: Um valor $n \in \mathbb{Z}^+$
Output: O fatorial de n

```

read n
f ← 1
for i ← 2 to n do
    f ← f · i
end for
write f

```

4.4-5 Implemente o Algoritmo 4.4-2 em C e o teste.

Algoritmo 4.4-2 Cálculo do fatorial de n .

Input: Um valor $n \in \mathbb{Z}^+$
Output: O fatorial de n

```

read n
f ← 1
while n > 1 do
    f ← f · n
    n ← n - 1
end while
write f

```

4.4-6 Implemente o Algoritmo 4.4-3 em C e o teste. Dois testes interessantes: 46.623.939 e 538.279.023 deve resultar 3 e 784.631.389.193 e 67.962.161.749 deve resultar 4373. Seu programa consegue lidar com estes valores? Quer testar mais? Confira seu programa usando uma calculadora online².

Algoritmo 4.4-3 Máximo divisor comum pelo método de Euclides.

Input: Dois valores inteiros positivos n_1 e n_2
Output: O valor correspondente ao máximo divisor comum de n_1 e n_2

```

Obtenha  $n_1$  e  $n_2$ 
repeat
    Calcule o resto  $r$  da divisão de  $n_1$  por  $n_2$ 
    Substitua  $n_1$  pelo o valor de  $n_2$ 
    Substitua  $n_2$  pelo o valor de  $r$ 
until o resto  $r$  ser nulo
Apresente o valor de  $n_1$        $\triangleright$   $n_1$  contém o MDC

```

4.4-7 É necessário calcular a média de três provas para cada aluno de uma turma, fornecidos o número de alunos da turma e as notas das provas de cada aluno. Cada média deve ser apresentada logo após a entrada das três notas.

Codifique um programa em C que resolva este problema.

4.4-8 Escreva um programa em C para apresentar uma tabela de conversão de Celsius para Fahrenheit. A tabela deve apresentar de 0 a 50 °C, de 5 em 5 graus.

A conversão de c graus Celsius para f Fahrenheit é dada por

$$f = \frac{9}{5}c + 32.$$

4.4-9 Considere os seguintes dados sobre alunos: RA, IRA, idade e sexo (sendo “M” para sexo masculino e “F” para feminino). Os dados são disponibilizados iniciando-se com o número total de alunos e seguidos de uma sequência de tuplas, uma por aluno. É preciso calcular quantas das entradas apresentam idade no intervalo [18, 22]. Como resultado, deve ser apresentado o número de entradas no intervalo especificado.

Escreva um programa em C para processar os dados e mostrar a informação solicitada.

4.4-10 ★★★★★ Uma entrada de dados é formada por diversos pares de valores inteiros. O primeiro valor do par usa o código 1 para o sexo masculino e 2 para o feminino e o segundo valor é a altura da pessoa em centímetros. Considere que a entrada possa ser vazia, ou seja, não existirem dados.

Escreva um programa em C que apresente a altura média dos homens e das mulheres, caso existam. Nenhuma saída deve ser produzida se não existir a informação correspondente.

4.4-11 Um número natural (\mathbb{N}) é dito perfeito se ele for igual à soma de todos seus divisores, exceto ele mesmo. Os números 6 e 28 são perfeitos, assim como o 33.550.336.

$$6 = 1 + 2 + 3$$

$$28 = 1 + 2 + 4 + 7 + 14$$

Um programa para, a partir de um número inteiro positivo, determinar se ele é perfeito, apresentando 1 (**true**) ou 0 (**false**) como saída.

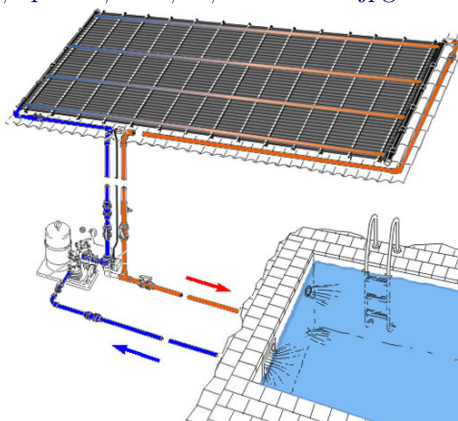
4.4-12 Apresente dois algoritmos para calcular $S = 1 + 2 + \dots + n$, um com fórmula e outro com uma repetição que faça as sucessivas somas. Compare os desempenhos.

A soma dos n primeiros termos da sequência S é dado por

$$\sum_{i=1}^n i = \frac{n(n+1)}{2}.$$

²<http://ecalculator.blogspot.com.br/p/mmc-e-mdc.html>.

Figura 4.4-1: Esquema de aquecimento de água.
 Fonte: <http://www.netaquecedores.com.br/wp-content/uploads/2013/01/instalacao1.jpg>.



Escreva um programa em C para cada uma das soluções e tente avaliar se há diferença sensível no tempo de execução para valores altos de n .

4.4-13 Um sistema de aquecimento de piscinas usa controle diferencial de temperatura. Ele possui um coletor no telhado onde é feito o aquecimento da água, que é bombeada para a piscina, conforme o esquema na Figura 4.4-1. Há dois sensores de temperatura: um no coletor e outro na piscina. O sistema liga automaticamente a bomba quando a temperatura do coletor está 8°C acima da temperatura da água da piscina e a desliga quando a diferença cai para 4°C .

Considere que uma sequência de pares de temperatura, a primeira do coletor e a segunda da piscina, é coletada em intervalos de tempo regulares. O fim da sequência ocorre quando a temperatura da piscina for superior a 40°C ou quando a temperatura do coletor for inferior a 0°C . O sistema se autodesliga nestas condições por questões de segurança.

A Tabela 4.4-1 ilustra uma possível situação de dados. A primeira coluna indica a temperatura do coletor, a segunda a da piscina e a terceira a diferença entre elas. Os dados disponíveis correspondem apenas às duas primeiras colunas.

Escreva um algoritmo de nível alto que determine, a partir dos dados descritos, em quantas leituras a bomba foi acionada (ou permaneceu acionada). Use a lógica do algoritmo para, então, fazer a codificação na linguagem C.

Tabela 4.4-1: Exemplo de temperaturas coletadas por sensores e a diferença entre elas ($^{\circ}\text{C}$).

Temp. coletor	Temp. piscina	Diferença
35,0	25,0	10,0
36,0	26,0	10,0
36,4	27,0	9,4
37,5	27,9	9,6
37,0	28,9	8,1
38,5	29,7	8,8
38,9	30,6	8,3
37,9	31,4	6,5
36,9	32,1	4,8
34,2	32,6	1,6
31,0	32,8	-1,8
27,4	32,6	-5,2
24,7	32,1	-7,4
24,8	31,4	-6,6
27,5	30,7	-3,2
29,6	30,4	-0,8
31,9	30,3	1,6
36,4	30,5	5,9
39,3	31,1	8,2
41,4	31,9	9,5
43,7	32,9	10,8
44,3	34,0	10,3
45,7	35,0	10,7
45,4	36,1	9,3
44,8	37,0	7,8
43,4	37,8	5,6
43,5	38,4	5,1
44,2	40,9	3,3

4.4-14 Dado o número de termos n (inteiro e maior que zero), é preciso que sejam apresentados os n primeiros elementos de uma sequência.

Dadas essas condições, considere separadamente cada uma das sequências (os sinais positivos (+) são usados apenas para clareza):

- $\langle +1, -2, +3, -4, +5, -6, \dots \rangle$
- $\langle 1/2, 1/3, 1/4, 1/5, \dots \rangle$
- $\langle +4, -4/3, +4/5, -4/7, +4/9, \dots \rangle$

- ★★ $\langle 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, \dots \rangle$

Elabore um algoritmo para cada sequência e codifique o programa em C correspondente. O formato da apresentação de cada termo é irrelevante.

4.4-15 ★★ Um relatório de notas de alunos contém as seguintes informações:

- O número total de alunos;
- Para cada aluno, os seguintes dados: nome e notas de duas provas (de 0,0 a 10,0).

A nota individual de um aluno é calculada pela expressão

$$m = \frac{35p_1 + 65p_2}{100}.$$

É preciso determinar e apresentar, para um relatório deste tipo:

- A nota máxima individual;
- A média de todas as notas individuais;
- O nome do aluno com nota máxima (em caso de empate, apenas um nome deve ser apresentado, indistintamente)

Escreva um algoritmo que apresente uma solução para o problema e o codifique em C.

4.4-16 Um sistema de monitoramento usa um radar de velocidade para aferir quanto rápido trafegam os veículos em uma dada via, com registros em quilômetros arredondados para valores inteiros.

Para levantamentos estatísticos, devem processados lotes de 100 registros de velocidade por vez, produzindo como resultado os valores das velocidades mínima e máxima do lote, juntamente com o número de veículos com cada uma dessas velocidades.

Escreva um algoritmo para processar um único lote de 100 registros e apresentar os resultados solicitados. Codifique sua solução em programa em C.

4.4-17 Segundo a formulação de Leibniz, o valor de π pode ser dado pelo somatório:

$$\pi = 4 - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \dots$$

Escreva programa em C para estimar uma aproximação de π com n casas decimais.

Nesta aproximação, considere que somente os termos cujos módulos sejam maiores que $5 \cdot 10^{-(n+1)}$ sejam somados para um dado n de entrada.

4.4-18 ★★★★★ A sequência de Fibonacci é a sequência numérica proposta pelo matemático Leonardo Pisa (conhecido por Fibonacci):

$$\langle 1, 1, 2, 3, 5, 8, 13, 21, \dots \rangle$$


Os termos são definidos como se segue:

$$f_1 = 1$$

$$f_2 = 1$$

$$f_n = f_{n-1} + f_{n-2}, \quad n > 2$$

Escreva um programa em C para, dada a posição de um termo n ($n \geq 1$), apresentar o valor correspondente. Exemplo: para $n = 7$, apresentar o sétimo termo, 13.

4.4-19  Codifique e execute o Programa 4.4-3.

Acrescente o seguinte código nos locais indicados e execute novamente.

```
for(i = 0; i < 20; i++) // desenha uma "linha"
    printf("-");
printf("\n");
```

O resultado esperado continua correto? Porque uma modificação tão simples pôde comprometer tão significativamente o resultado?

Programa 4.4-3 Soma simples de valores positivos.

```
/*
Input: Uma sequência de valores
       inteiros positivos terminada com zero
Output: A soma dos valores entrados
*/
#include <stdio.h>

int main(){
    int i = 0; // soma dos valores
    int j; // valor de entrada

    /* local de insercao do codigo novo */

    do{
        printf("Valor (0 para terminar): ");
        scanf("%d", &j);

        i = i + j; // soma
    }while(j != 0);

    printf("Soma: %d\n", i);

    return 0;
}
```

Arquivo: material/somasimples.c.

4.5 Variáveis compostas

4.5-1 Implemente o Programa 4.5-1 e o execute. Entenda bem a forma de declaração das variáveis `pto1` e `pto2`.

Programa 4.5-1 Programa com registro.

```
/*
   Input: Dois pontos em R^2
   Output: A distancia entre esses pontos
*/
#include <stdio.h>
#include <math.h> // para sqrt e pow

int main(){
    struct{
        double x, y;
    } pto1, pto2;
    double distancia;

    /* Entrada */
    printf("Coordenadas de P1: ");
    scanf("%lf%lf", &pto1.x, &pto1.y);
    printf("Coordenadas de P2: ");
    scanf("%lf%lf", &pto2.x, &pto2.y);

    /* Distancia */
    distancia = sqrt(pow(pto1.x - pto2.x, 2) +
                    pow(pto1.y - pto2.y, 2));

    /* Resultado */
    printf("Distancia: %g\n", distancia);

    return 0;
}
```

Programa 4.5-2 Programa com registro.

```
/*
   Input: Dois pontos em R^2
   Output: A distancia entre esses pontos
*/
#include <stdio.h>
#include <math.h> // para sqrt e pow

int main(){
    struct ponto2d{
        double x, y;
    }

    struct ponto2d pto1, pto2;
    double distancia;

    /* Entrada */
    printf("Coordenadas de P1: ");
    scanf("%lf%lf", &pto1.x, &pto1.y);
    printf("Coordenadas de P2: ");
    scanf("%lf%lf", &pto2.x, &pto2.y);

    /* Distancia */
    distancia = sqrt(pow(pto1.x - pto2.x, 2) +
                    pow(pto1.y - pto2.y, 2));

    /* Resultado */
    printf("Distancia: %g\n", distancia);

    return 0;
}
```

4.5-2 Implemente o Programa 4.5-2 e o execute. Entenda bem a forma de declaração das variáveis `pto1` e `pto2`.

4.5-3 Implemente o código do Programa 4.5-3 e corrija o erro, mantendo a estrutura do registro.

4.5-4 Na linguagem C, o conteúdo de todo um registro pode ser copiado para outro. Assim, se `r1` e `r2` tiverem o mesmo tipo, a atribuição seguinte é válida e os conteúdos das variáveis ficam idênticos.

```
r1 = r2; // copia de registros
```

Porém, o Programa 4.5-4, ao ser compilado, apresenta o seguinte erro:

```
registrocopia.c: In function 'main':
registrocopia.c:21:14: error: incompatible
types when assigning to type 'struct <
anonymous>' from type 'struct <anonymous>'
ponto_saida = ponto_entrada;
~
```

Por que a atribuição falha neste caso?

Programa 4.5-3 Leitura e escrita de um ponto em \mathbb{R}^2

```
/*
   Input: um ponto em R^2
   Output: o mesmo ponto
*/
#include <stdio.h>

int main(){
    struct{
        float x, y;
    } ponto;

    /* Entrada */
    printf("Coordenadas x e y: ");
    scanf("%f%f", &x, &y);

    /* Saida */
    printf("Ponto: (%g, %g)\n", x, y);

    return 0;
}
```

Arquivo: material/ponto2d.c.

Programa 4.5-4 Programa com registro.

```
/*
   Input: Um ponto em R^2
   Output: O mesmo ponto
*/
#include <stdio.h>
#include <math.h> // para sqrt e pow

int main(){
    struct{
        double x, y;
    } ponto_entrada;
    struct{
        double x, y;
    } ponto_saida;

    /* Entrada */
    printf("Coordenadas do ponto: ");
    scanf("%lf%lf", &ponto_entrada.x, &
        ponto_entrada.y);


    /* Cópia de registros */
    ponto_saida = ponto_entrada;

    /* Resultado */
    printf("Ponto: (%g, %g)\n", ponto_saida.x,
        ponto_saida.y);

    return 0;
}
```

4.5-5 Um número complexo z pode ser expresso na forma $z = a + bi$, sendo a a parte real e b a parte imaginária, com $i = \sqrt{-1}$. O módulo de um valor z , é dado por $|z| = \sqrt{a^2 + b^2}$.

Usando registros para estruturar os dados, escreva um programa em C que leia separadamente os valores da parte real e da imaginária de um número complexo e escreva seu módulo.

4.5-6  Considerando a apresentação de números complexos do Exercício 4.5-5, escreva um programa em C para ler um número complexo (lendo separadamente a parte real e a imaginária) e escrevê-lo de forma padronizada e bem formatada. Use o conceito de registro para organizar os dados.

A Tabela 4.5-1 mostra os resultados esperados para alguns exemplos de entrada. Atenção aos espaços internos usados na formatação, em especial para a diferença esperada para o sinal negativo entre os valores $-3 - 5i$ e $-5i$ ("3_-5i" versus "-5i").

4.5-7 ★ Escreva um programa em C que use registros para representar pontos no espaço cartesiano \mathbb{R}^3 . O programa deve ler os dados de dois pontos e apresentar o ponto médio entre eles.

O ponto médio entre (x_1, y_1, z_1) e (x_2, y_2, z_2) é dado por

$$\left(\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2}, \frac{z_1 + z_2}{2} \right).$$

Quadro 4.5-1 Formatação de números complexos. O símbolo _ indica um espaço em branco.

\Re	\Im	Saída
3,1	5	3.1_5i
-3	5,5	-3_+5.5i
3,1	-5,5	3.1_-5.5i
-3	-5	-3_-5i
3	0	3
0	5	5i
-3	0	-3
0	-5	-5i

4.5-8 Implemente o Programa 4.5-5 e teste sua execução. Ele é uma implementação para o Algoritmo 3.8-4.

Programa 4.5-5 Determinação do número mínimo de cédulas ou moedas para uma dada quantidade monetária

```
/*
   Input: Um valor monetario v real expresso
   com ate duas casas decimais (centavos)
   Output: O numero n de cedulas ou moedas
   minimo equivalente a v
*/
#include <stdio.h>

int main(){
    float valor; // valor em R$
    int numero_total, // contador total
        numero_dinheiro; // contador especifico
    float dinheiro[] = {
        100, 50, 20, 10, 5, 2, 1,
        0.5, 0.25, 0.1, 0.05, 0.01
    };

    scanf("%f", &valor);
    numero_total = 0;
    for(int i = 0; i < 12; i++){
        numero_dinheiro = valor/dinheiro[i];
        numero_total += numero_dinheiro;
        valor -= numero_dinheiro * dinheiro[i];
    }
    printf("%d\n", numero_total);

    return 0;
}
```

Arquivo: material/monetario.c.

4.5-9 Implemente o Programa 4.5-6 e certifique-se que entende os mecanismos de acesso aos dados individuais do vetor.

4.5-10 Escreva um programa em C que leia 20 valores reais e os escreva na ordem inversa à qual foram digitados. Teste-o.

Programa 4.5-6 Porcentagem de dados acima da média.

```
/*
   Input: 10 valores inteiros maiores que zero
   Output: A porcentagem de valores maiores
           que a média deles
*/
#include <stdio.h>

int main(){
    int i;
    int dado[10]; // 10 inteiros

    /* Leitura e calculo da media */
    int soma = 0;
    for(i = 0; i < 10; i++){
        printf("Valor %2d: ", i + 1);
        scanf("%d", &dado[i]);
        soma += dado[i];
    }
    double media = (double)soma/10;

    /* Contagem de dados acima da media */
    int conta_acima = 0;
    for(i = 0; i < 10; i++)
        if(dado[i] > media)
            conta_acima++;

    /* Resultado */
    double porcentagem = (double)conta_acima
        /10;
    printf("Acima da media %g: %g%%\n",
        media, porcentagem * 100);

    return 0;
}
```

Arquivo: material/acimamedia.c.

4.5-11 Escreva um programa em C que execute as seguintes ações, na ordem dada:

- Leia 10 inteiros em um vetor;
- Apresente todos os dados do vetor;
- Troque o primeiro valor com o último;
- Reapresente todos os dados.

┘

4.5-12 Escreva um programa em C que:

- Leia em 20 valores inteiros quaisquer em um vetor;
- Modifique todos os valores iguais ao último para zero, exceto o último;
- Apresente os valores.

┘

4.5-13 ✓ Escreva um programa em C que:

- Leia em 10 valores inteiros quaisquer em um vetor;
- Inverta a ordem dos dados no vetor;
- Apresente os valores.

┘

4.5-14 Escreva um programa em C que:

- Leia dois vetores com tamanho máximo de 30 itens inteiros;
- Escreva se os vetores são ou não iguais, dando mensagem “iguais” ou “diferentes”.

Um vetor é igual ao outro se contiver os mesmos valores nas mesmas posições

┘

4.5-15 ★★ Enquanto pontos em \mathbb{R}^2 e \mathbb{R}^3 sejam usualmente representados em programas usando registros (como o do Programa 4.5-2, por exemplo), espaços com dimensionalidade maior acabam ficando impraticáveis. Assim, para um ponto em \mathbb{R}^{10} é difícilmente estruturado um registro com 10 campos.

Para estes casos, é mais prático que seja usado um vetor com tamanho igual à dimensão para guardar as coordenadas.

Escreva um programa em C para criar ler dois pontos em \mathbb{R}^8 e determinar a distância entre eles, escrevendo o resultado. A distância entre $P = (p_1, p_2, \dots, p_8)$ e $Q = (q_1, q_2, \dots, q_8)$ é dada por

$$d_{PQ} = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \dots + (p_8 - q_8)^2}.$$

Teste seu programa.

┘

4.5-16 Um vetor (no sentido matemático) em \mathbb{R}^n é um valor $\vec{v} = [v_1, v_2, \dots, v_n]$.

O módulo (ou norma) de um vetor é dado por

$$\begin{aligned} \|\vec{v}\| &= \sqrt{\sum_{i=1}^n v_i^2} \\ &= \sqrt{v_1^2 + v_2^2 + \dots + v_n^2}. \end{aligned}$$

A normalização é a obtenção de um vetor \vec{w}

$$\vec{w} = \frac{\vec{v}}{\|\vec{v}\|} = \left[\frac{v_1}{\|\vec{v}\|}, \frac{v_2}{\|\vec{v}\|}, \dots, \frac{v_n}{\|\vec{v}\|} \right].$$

A soma de dois vetores é o vetor

$$\vec{v} + \vec{w} = [v_1 + w_1, v_2 + w_2, \dots, v_n + w_n].$$

O produto escalar de dois vetores é:

$$\begin{aligned}\vec{v} \cdot \vec{w} &= \sum_{i=1}^n v_i w_i \\ &= v_1 w_1 + v_2 w_2 + \dots + v_n w_n\end{aligned}$$

Considere um vetor no espaço de \mathbb{R}^{10} e escreva um programa para ler dois vetores e determinar e apresentar:

- o módulo do primeiro vetor;
- o vetor normalizado do primeiro vetor (se o módulo for nulo, apresente "impossível normalizar");
- o vetor soma dos dois vetores;
- o produto escalar dos dois vetores.

4.5-17 Complemente o Exercício 4.7-24, acrescentando as seguintes funcionalidades:

- Multiplicação de vetor por escalar
- A subtração de dois vetores

A multiplicação de \vec{v} por um escalar $a \in \mathbb{R}$ é dada por

$$a\vec{v} = [av_1, av_2, \dots, av_n]$$

No caso da subtração,

$$\vec{v} - \vec{w} = [v_1 - w_1, v_2 - w_2, \dots, v_n - w_n].$$


4.5-18 Escreva um programa em C que:

- Leia uma quantidade $q \in \{1, 2, \dots, 20\}$;
- Leia q inteiros quaisquer em um vetor;
- Inverta a ordem dos dados no vetor;
- Apresente os valores.

4.5-19 Escreva um programa em C que:

- Leia uma quantidade $q \in \{1, 2, \dots, 20\}$;
- Leia q inteiros quaisquer em um vetor;
- Rearranje a ordem dos dados no vetor de forma que todos os ímpares fiquem antes dos pares;
- Apresente os valores.

Neste exercício, a ordem entre os pares e entre os ímpares é irrelevante. Note a equivalência deste exercício com o Exercício 3.6-13.

4.5-20  Escreva um programa em C que leia um texto qualquer e determine o número de palavras. Considere que o texto segue as restrições:

- Não há espaços antes ou depois do texto;
- Palavras hifenizadas contam como uma única palavra (e.g., *sente-se*, *sub-rotina*);
- Não existem sinais de pontuação;

- Não há ocorrência de dois ou mais espaços consecutivos;
- O comprimento máximo do texto é limitado a 100 caracteres.

4.5-21 Conjunto é uma reunião de elementos, podendo ter qualquer quantidade ou ser vazio (\emptyset). Em conjuntos, não faz sentido a repetição de elementos nem há ordem entre eles. Por exemplo,

$$\{1, 2, 3\} = \{3, 2, 1\} = \{1, 1, 2, 2, 3\}.$$

Em um programa, um conjunto pode ser representado em um vetor que contenha seus elementos.

Escreva um programa em C que:

- Leia uma quantidade $n \geq 0$ de dados que serão digitados;
- Crie um conjunto a partir dos n dados de serão lidos;
- Escreva os elementos do conjunto (a ordem é irrelevante).

Assuma que os elementos sejam valores inteiros qualquer. Se $n = 0$, então o conjunto será vazio. O número de elementos do conjunto pode ser menor ou igual ao valor de n , pois se forem informados números repetidos, apenas uma ocorrência deve ser mantida.

Como exemplo, caso a entrada indique $n = 5$ e os valores dos elementos digitados sejam $\langle 3, 8, 3, 1, 3 \rangle$, os valores escritos para o conjunto devem ser 1, 3 e 8 (em qualquer ordem).

4.5-22 Implemente e execute o Programa 4.5-7, entendendo bem o acesso a cada elemento da matriz.

4.5-23 ★★ Escreva um programa em C que leia uma matriz quadrada 10×10 e apresente:

- os elementos da diagonal principal;
- os elementos da diagonal secundária;
- apenas os elementos acima da diagonal principal;
- apenas os elementos abaixo da diagonal secundária.

Tente fazer seu código tão eficiente quanto possível. Por exemplo, para escrever os elementos da diagonal, não é preciso passar por todos os elementos da matriz.

4.5-24 Escreva um programa que leia uma matriz quadrada 10×10 e escreva seu traço, que é definido como a soma dos elementos da diagonal principal.

4.5-25 ★★★★★ Escreva um programa em C que leia uma matriz 5×5 e a transforme em sua transposta. O objetivo deste exercício é trocar os valores de posição na mesma matriz, sem uso de uma matriz auxiliar.

Programa 4.5-7 Manipulação simples de matriz.

```
/*
   Output: o conteúdo de uma matriz em vários
   formatos e ordem
*/
#include <stdio.h>

#define LINHAS 5
#define COLUNAS 4
int main(){
    int matriz[][COLUNAS] = {
        1, 2, 3, 4,
        4, 3, 2, 1,
        0, 0, 0, 0,
        1, 0, 1, 0,
        1, 3, 5, 7
    };

    printf("Matriz inteira:\n");
    for(int i = 0; i < LINHAS; i++){
        for(int j = 0; j < COLUNAS; j++){
            printf("%d ", matriz[i][j]);
            printf("\n");
        }

        printf("\nLinha 2:\n");
        for(int j = 0; j < COLUNAS; j++){
            printf("%d ", matriz[2][j]);
            printf("\n");
        }

        printf("\nColuna 0:\n");
        for(int i = 0; i < LINHAS; i++){
            printf("%d\n", matriz[i][0]);
        }

        return 0;
    }
}
```

Arquivo: material/matrizsimples.c.

4.5-26 Escreva um programa em C para ler duas matrizes de dimensões máximas 20×20 , considerando a dimensão mínima 1×1 . As duas matrizes podem ter dimensões diferentes entre si.

O usuário deve então escolher uma linha da primeira matriz e uma coluna da segunda.

A saída do programa é a soma dos valores dos elementos da linha da primeira matriz multiplicados pelos da coluna da segunda matriz. Em outras palavras, equivale ao produto escalar do vetor linha pelo vetor coluna (veja o Exercício 4.7-24).

Caso o número de colunas da primeira matriz seja diferente do número de linhas da segunda, apresente a mensagem “impossível”.

4.5-27 Complemente o Exercício 4.5-23, acrescentando as seguintes funcionalidades:

- Cálculo da soma de todos os elementos da matriz;
- Cálculo da média de todos os elementos da matriz.

4.5-28 ★★★★★ Complemente o Exercício 4.5-23, acrescentando as seguintes funcionalidades:

- Verificação se a matriz é a matriz identidade
- Verificação se a matriz é simétrica

Uma matriz é identidade se os elementos da diagonal forem todos iguais a 1 e todos os demais elementos

Figura 4.5-1: Ilustração de um “círculo” desenhado com caracteres. Neste caso, o diâmetro escolhido foi 11.

```
....***....
..**.....*
.*.....*
.*.....*
*.....*
*.....*
*.....*
*.....*
.*.....*
.*.....*
..**.....*
....***....
```

iguais a 0. Uma matriz é simétrica quando $m_{ij} = m_{ji}$, $\forall i, j$.

Importante: não faça comparações além das necessárias! Por exemplo: se já sabe que $m_{3,2} = m_{2,3}$, porque verificar se $m_{2,3} = m_{3,2}$?

4.5-29 ★★★★★ Escreva um programa em C que desenhue um “círculo” na tela usando caracteres (veja a Figura 4.5-1). Para tanto, use uma matriz de caracteres e a preencha com espaços. Depois, substitua os espaços por asteriscos, de forma que, ao escrever a matriz, seja visualizado um círculo.

O desenho do círculo deve ser feito solicitando o diâmetro, ou seja, o número de linhas e colunas da matriz, que deve ser quadrada. Tome suas decisões sobre como representar matrizes de tamanhos diversos.

O traçado não precisa ser perfeito ou bonito.

4.6 Ponteiros

4.6-1 Uma variável do tipo ponteiro guarda o endereço do primeiro byte de alguma coisa na memória.

Implemente e execute o Programa 4.6-1. Modifique-o para fazer alguns testes seus, com outras variáveis e funções, por exemplo.

Neste programa, o tipo do endereço é `void`, que significa “sem tipo”.

4.6-2 O operador `*` pode ser usado para se ter acesso ao conteúdo armazenado em um endereço. Implemente e execute o Programa 4.6-2. Entenda bem o que significa `*ponteiro`.

4.6-3 O operador `*` pode ser usado para se ter acesso ao conteúdo armazenado em um endereço. Implemente e execute o Programa 4.6-3. Entenda bem o que significa `*ponteiro`.

Programa 4.6-1 Apresentação de alguns endereços de memória.

```
/*
   Output: O endereço de diversas coisas
*/
#include <stdio.h>

int main(){
    void *endereco;
    int i;
    float f;

    endereco = &main;
    printf("main() inicia em %p\n", endereco);

    endereco = &endereco;
    printf("variavel endereco inicia em %p\n",
        endereco);

    endereco = &i;
    printf("variavel i inicia em %p\n",
        endereco);

    endereco = &f;
    printf("variavel f inicia em %p\n",
        endereco);

    endereco = &printf;
    printf("funcao printf inicia em %p\n",
        endereco);

    endereco = NULL;
    printf("ponteiro nulo: %p\n", endereco);

    return 0;
}
```

Arquivo: material/enderecos.c.

Programa 4.6-2 Inspeção de conteúdo de memória com o operador *.

```
/*
   Output: O valor armazenado em algumas
           posicoes de memoria
*/
#include <stdio.h>

int main(){
    int inteiro1, inteiro2;
    int *ponteiro;

    inteiro1 = 11;
    inteiro2 = 22;

    ponteiro = &inteiro1;
    printf("inteiro1 = %d e *ponteiro %d\n",
        inteiro1, *ponteiro);

    ponteiro = &inteiro2;
    printf("inteiro2 = %d e *ponteiro %d\n",
        inteiro2, *ponteiro);

    return 0;
}
```

Arquivo: material/conteudomemoria.c.

Programa 4.6-3 Modificação de conteúdo de memória com o operador *.

```
/*
   Output: Conteudo da memoria
*/
#include <stdio.h>

int main(){
    int inteiro;
    int *ponteiro;

    inteiro = 100;

    ponteiro = &inteiro;
    printf("inteiro = %d e *ponteiro %d\n",
        inteiro, *ponteiro);

    *ponteiro = 200; // alteracao direta
    printf("inteiro = %d e *ponteiro %d\n",
        inteiro, *ponteiro);

    return 0;
}
```

Arquivo: material/modificacaomemoria.c.

4.6-4 Escreva um programa em C, baseado no Programa 4.6-1, que declare uma variável do tipo registro (**struct**). Use o operador **&** para mostrar o endereço tanto da variável como um todo quanto dos seus campos.

4.6-5 Escreva um programa em C, baseado no Programa 4.6-1, que declare um vetor (ou matriz). Use o operador **&** para mostrar o endereço tanto do vetor quanto de alguns de seus elementos.

Em particular, verifique que o endereço do vetor e de seu elemento na posição zero coincidem.

4.6-6 ★★★★★ Escreva um programa em C para ler duas variáveis inteiras e dobrar o valor da menor delas (em caso de igualdade, dobrar o valor da primeira variável). As comparações e modificações devem ser feitas exclusivamente usando ponteiros. Os valores devem ser apresentados no final.

4.6-7 ★★★★★ Escreva um programa em C para ler duas variáveis do tipo **double** e trocar o conteúdo de uma com o da outra usando exclusivamente ponteiros para essa ação. Os valores devem ser apresentados no final.

4.6-8 ★★ Observe atentamente o Programa 4.6-4 e tente prever os valores que serão escritos caso a leitura seja feita para o valor 10.

Depois, implemente e execute o código. Sua previsão foi correta? Se não, entenda o que aconteceu.

Programa 4.6-4 Uso de múltiplos ponteiros (dois, neste caso).

```
/*
   Input: um valor inteiro
   Output: tres valores inteiros
*/
#include <stdio.h>

int main(){
    int i, *p1, *p2;

    printf("i = ");
    scanf("%d", &i);

    p1 = &i;
    p2 = p1;
    *p2 = *p1 * 2;

    printf("%d, %d e %d\n", i, *p1, *p2);

    return 0;
}
```

Arquivo: material/variosponteiros.c.

4.6-9 Considere a declaração seguinte para um registro.

```
struct reg {
    int i;
    float f;
};
```

Suponha as seguintes declarações de variáveis:

```
struct reg r;
struct reg *pont_r;
```

Há dúvidas de que ambas as partes abaixo são equivalentes?

```
/* parte 1: acesso direto */
r.i = 10;
r.f = 1.25;

/* parte 2: acess por ponteiro */
pont_r = &r;
pont_r->i = 10;
pont_r->f = 1.25;
```

Uma atenção especial ao uso do operador -> é importante.

4.6-10 Considere a declaração seguinte para um registro e um ponteiro:

```
struct reg {
    char nome[100];
    char cpf[11];
};
struct reg r1, r2, *pont_r;
```

O que o código seguinte faz? Ele é válido?

```
pont_r = &r1;
r2 = *pont_r;
```

4.6-11 Implemente o código do Programa 4.6-5 e o execute. Acostume-se e entenda o acesso a um registro por meio de ponteiros.

Programa 4.6-5 Acesso a registro por ponteiro.

```
/*
   Input: Nome e ano de nascimento de uma
         pessoa
   Output: Sua idade, assumindo-se que ja tenha
           feito aniversario no ano atual
*/
#include <stdio.h>
#include <string.h> // para strlen()
#include <time.h> // para determinar o ano
                  atual

int main(){
    struct pessoa{
        char nome[100];
        int ano_nascimento;
    };
    struct pessoa uma_pessoa;

    /* Entrada de dados */
    printf("Nome: ");
    fgets(uma_pessoa.nome,
        sizeof(uma_pessoa.nome), stdin);
    uma_pessoa.nome[strlen(uma_pessoa.nome)-1] =
        '\0';
    printf("Ano de nascimento: ");
    scanf("%d", &uma_pessoa.ano_nascimento);

    /* obtencao do ano atual */
    /* (ignore esta parte do codigo) */
    time_t tempo = time(NULL);
    struct tm *data = localtime(&tempo);
    int ano_atual = 1900 + data->tm_year;

    /* Calculo e apresentacao da idade */
    /* usando um ponteiro para o registro */
    /* (essa parte do codigo eh importante!) */
    struct pessoa *ponteiro = &uma_pessoa;
    int idade = ano_atual -
        ponteiro->ano_nascimento;
    printf("%s tem %d anos\n", ponteiro->nome,
        idade);

    return 0;
}
```

Arquivo: material/ponteiroregistro.c.

4.6-12 ★★★★★ Escreva um programa em C para ler dois registros contendo RG e ano de nascimento de duas pessoas, escrevendo os dados da mais velha (em caso de mesma idade, apresentar os dados da primeira). Os dados de cada indivíduo devem ser mantidos em registros e as comparações e apresentações devem usar exclusivamente ponteiros. Use inteiros para RG e ano de nascimento.

4.6-13 ✓ Qual a saída esperada para o Programa 4.6-6? Tente fazer sua previsão e depois implemente e verifique se acertou.

Programa 4.6-6 Exemplos de uso de ponteiros com vetores.

```
/*
   Output: dados de um vetor manipulado com
           ponteiros
*/
#include <stdio.h>

int main(){
    int vet[] = {1, 2, 3, 4, 5, 6, 7, 8};

    /* Antes */
    printf("original: [ ");
    for(int i = 0; i < 8; i++)
        printf("%2d ", vet[i]);
    printf("]\n");

    /* Modificacoes */
    vet[4] = 40;

    int *pont = vet;
    *pont = 99;
    *(pont + 1) = 33;
    pont[6] = 10 * pont[7];

    /* Depois */
    printf("modificado: [ ");
    for(int i = 0; i < 8; i++)
        printf("%2d ", vet[i]);
    printf("]\n");
}
```

Arquivo: material/ponteirovetor.c.

4.6-14 Suponha que um vetor `vet` de floats não possua nenhum elemento igual a zero. Considerando-se que todas as instruções abaixo manipulem posições válidas desse vetor, determine quantas se tornam nulas depois da execução. Assuma que `p` seja um ponteiro para float.

```
p = &vet[5];
*p = 0;
p[1] = 0;
p[-1] = 0;
p++;
p[0] = 0;
*(p + 1) = 0;
```

4.6-15 Assuma que `vet` seja um vetor de valores numéricos que contenha pelo menos um valor negativo. O que o contador `cont` na sequência de comandos seguinte determina, sendo `p` um ponteiro para o tipo base do vetor?

```
cont = 0;
p = vet;
while(*p >= 0){
    cont++;
    p++;
}
```

4.6-16 Suponha uma matriz `mat` de inteiros com dimensões 15×12 e que `i` e `j` sejam valores inteiros. Considere que `p` seja um ponteiro para inteiro e que aponte para o primeiro elemento da matriz (primeira linha, primeira coluna).

Quando se escreve `*(p + 12*i + j)`, a qual elemento da matriz se está referenciando? Considere tanto `i` quanto `j` no intervalo $[0, 11]$.

4.7 Modularização

4.7-1 ★ Considere o Programa 4.7-1. Qual o valor que você espera que seja escrito pelo código?

Transcreva o programa e teste-o. O resultado foi o esperado? Se não foi, entenda o porquê!

Programa 4.7-1 Programa para teste.

```
/*
   Output: Valores da variavel i em dois
           contextos diferentes
*/
#include <stdio.h>

/*
   p(i)
   Input: um valor inteiro i
   Output: o valor de i na tela
*/
void p(int i){
    i = 10;
    printf("Em p: %d\n", i);
}

/*
   Programa principal
*/
int main(){
    int i;
    i = 20;
    p(i);
    printf("Em main: %d\n", i);
    return 0;
}
```

Arquivo: material/testeescopo.c.

4.7-2 Escreva uma função em C para retornar o valor máximo entre dois doubles. Escreva um pequeno programa para testar sua função.

4.7-3 Teste a função escrita para o Exercício 4.7-2 usando variáveis inteiras. Diga se funcionou e porquê.

4.7-4 Escreva uma função em C para retornar o sinal de um valor real v . A função $\text{sign } x$ é definida por

$$\text{sign } x = \begin{cases} -1, & \text{se } x < 0 \\ 0, & \text{se } x = 0 \\ 1, & \text{se } x > 0. \end{cases}$$

Escreva um pequeno programa para testar sua função.

4.7-5 Escreva uma função para retornar o fatorial de um valor $n \in \mathbb{Z}^+$. Lembre-se que $0! = 1$ e que $n! = n \cdot (n-1)!$, por definição.

Teste seu programa calculando $15!$, que vale 1.307.674.368.000. Lembre-se que esse código está coberto no Exercício 4.4-4.

4.7-6 Escreva uma função que aceite as coordenadas nos eixos ortogonais x , y e z de um ponto em \mathbb{R}^3 e retorne sua distância até a origem. Entre de um ponto (x, y, z) à origem é

$$d = \sqrt{x^2 + y^2 + z^2}.$$

Escreva um pequeno programa para testar sua função.

4.7-7 Escreva uma função que determine se um número inteiro n é ou não é primo. Um número N é dito primo se ele for divisível apenas por ele e pela unidade. Por esta definição, são primos: 1, 2, 3, 5, 7, 11, por exemplo.

A função deve retornar 1 caso o número seja primo ou 0, caso contrário.

Use o seguinte protótipo de função:

```
int testa_primo (int n);
```

Retome o Exercício 3.6-25 para referência.

Quadro 4.7-1 Estruturação de um ponto no espaço com um registro.

O registro seguinte é usado para descrever um ponto em \mathbb{R}^3 .

```
struct ponto3d {
    double x, y, z;
};
```

4.7-8 Considere o ponto definido no Quadro 4.7-1 e escreva:

- uma função sem parâmetros que leia do teclado três coordenadas e retorne um ponto;
- um procedimento que escreva um ponto na tela, apresentando suas coordenadas.

Escreva um programa simples para testar suas funções.

4.7-9 Escreva uma função que aceite dois pontos como o descrito no Quadro 4.7-1 e retorne a distância entre eles.

Adicione esta função ao código desenvolvido no Exercício 4.7-8 para testá-la.

4.7-10 Escreva uma função que retorne o ponto O (a origem do espaço de coordenadas), usando a estruturação de dados do Quadro 4.7-1. A origem é o ponto $(0, 0, 0)$.

Adicione esta função ao código desenvolvido no Exercício 4.7-8 para testá-la.

4.7-11 Use as funções escritas para os Exercícios 4.7-9 e 4.7-10 para escrever uma nova função, que aceita um ponto (Quadro 4.7-1) como parâmetro e retorna sua distância à origem.

4.7-12 Escreva uma função que aceite dois pontos como os do Quadro 4.7-1 e retorne o ponto médio entre eles. O ponto médio entre (x_1, y_1, z_1) e (x_2, y_2, z_2) é o ponto M tal que

$$M = \left(\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2}, \frac{z_1 + z_2}{2} \right).$$

Adicione esta função ao código desenvolvido no Exercício 4.7-8 para testá-la.

4.7-13 ★★ Codifique em C uma função que aceite um registro como o descrito abaixo e retorne a quantidade de valores não nulos antes do primeiro zero contido no vetor `dados`. Assuma que sempre haverá pelo menos um zero no vetor (não precisa verificar).

```
struct lista{
    char rotulo[10]; // descricao dos dados
    int dados[300]; // os dados em si
};
```

Escreva um programa simples que permita testar a função.

4.7-14 ★★ Considere um registro que represente um valor racional $q \in \mathbb{Q}$, no formato (s, n, d) , sendo s o sinal ($s = 1$ ou $s = -1$), $n \in \mathbb{N}$ o numerador e $d \in \mathbb{N}^*$ o denominador.

Assim, $-2/5$ será representado por $(-1, 2, 5)$, por exemplo.

Escreva uma função que retorne um valor lógico conforme dois valores racionais sejam ou não iguais. Lembre-se que $1/5 = 2/10$ e, portanto, $(1, 1, 5)$ e $(1, 2, 10)$ devem ser considerados iguais.

4.7-15 É preciso, para um conjunto de n pontos em \mathbb{R}^3 , determinar qual o mais próximo da origem. No caso de equidistância, o primeiro ponto entrado deve prevalecer. Assuma $n \in \mathbb{Z}^+$.

Escreva um algoritmo de nível alto que apresente uma solução para o problema. Então, faça uma codificação em C, usando a estruturação do Quadro 4.7-1 e reaproveitando o código escrito para os Exercícios 4.7-8 a 4.7-12.

4.7-16 Considere a definição a seguir.

```
struct vinho{
    int cod_produto,
        cod_uva;
    double preco;
};
```

Escreva:

- Um procedimento para ler os dados sobre uma garrafa de vinho (passagem por referência);
- Uma função para retornar o vinho mais caro entre dois vinhos;
- Um procedimento para, dados dois registros de vinhos, trocar o conteúdo de um registro com o de outro.

Escreva um programa simples para testar suas rotinas.

4.7-17 Considere a definição a seguir.

```
struct fracao{
    int numerador, denominador;
};
```

Escreva uma função que pegue um registro qualquer que descreva uma fração, com numerador e denominador, e o transforme para uma forma canônica, seguindo as seguintes regras:

- Caso a fração seja um número negativo, o termo negativo deve ficar sempre no numerador, sendo o denominador sempre positivo;
- A fração, se válida, deve sempre ficar na forma mais simples (veja o Algoritmo 4.4-3 para calcular o MDC);
- Se o denominador for nulo:
 - Se o numerador for nulo, deixá-lo como está;
 - Se o numerador for positivo, deixá-lo com valor 1;
 - Se o numerador for negativo, deixá-lo com valor -1.

Quadro 4.7-2 Descrição do vetor.

Considere um vetor com um total de t posições, cujas $n < t$ primeiras posições contenham dados úteis, sendo o conteúdo das demais irrelevantes. Os dados mantidos são numéricos.

Por exemplo, para $n = 4$, os valores válidos são os das posições de 0 a 3, como na ilustração seguinte.

0	1	2	3	4	...	$t-1$
5	-2	13	2	?	...	?


Nas codificações, assuma que o vetor tem tamanho máximo igual a 100.

4.7-18 Considere um vetor como o descrito no Quadro 4.7-2.

Escreva uma função que receba um vetor e o número de elementos válidos e retorne o valor mínimo. Caso o número de valores válidos seja zero, retornar $-\infty$.

4.7-19 Considere um vetor como o descrito no Quadro 4.7-2.


Escreva uma função que receba um vetor e o número de elementos válidos e retorne a média dos válidos (\mathbb{R}).

4.7-20  Considere um vetor como o descrito no Quadro 4.7-2.

Escreva um procedimento que receba um vetor, o número de elementos válidos e o valor de um novo elemento, modificando o vetor para que o novo elemento faça parte dos dados válidos.

4.7-21 Considere um vetor como o descrito no Quadro 4.7-2.

Escreva uma função que aceite como parâmetros o vetor, o número de elementos válidos e um elemento de consulta e retorne **true** ou **false** conforme a consulta esteja ou não presente no vetor.

4.7-22  Considere um vetor como o descrito no Quadro 4.7-2 e suponha que não haja valores repetidos entre os dados válidos.

Escreva um procedimento que receba um vetor, seu número de válidos e um novo elemento para ser inserido, fazendo a inserção do novo elemento apenas se ainda não houver um igual a ele.

4.7-23 Considere um vetor como o descrito no Quadro 4.7-2 e suponha que tanto o vetor quanto o número de elementos válidos formem uma única variável, na forma de um registro. Ou seja:

```
struct dado{
    int numero_validos;
    double dados[100];
};
```

Refça os Exercícios 4.7-18 a 4.7-22, usando um único parâmetro para as sub-rotinas e ajustando a notação adequadamente.

4.7-24 Um vetor (no sentido matemático) em \mathbb{R}^n é um valor $\vec{v} = [v_1, v_2, \dots, v_n]$.

Defina um registro que armazene um vetor, que pode estar no espaço de \mathbb{R}^1 até \mathbb{R}^{10} . No registro deve constar a dimensão e um arranjo de 10 posições deve ser usado para armazenar as coordenadas.

Escreva um programa para ler dois vetores (dimensão e demais coordenadas) usando um procedimento com passagem por referência.

Escreva também funções/procedimentos para:

- Escrever um vetor;
- Retornar um módulo do vetor;
- Normalizar o vetor (passando por referência);
- Retornar a soma de dois vetores;
- Retornar o produto escalar de dois vetores.

O módulo (ou norma) de um vetor é dado por

$$\|\vec{v}\| = \sqrt{\sum_{i=1}^n v_i^2} = \sqrt{v_1^2 + v_2^2 + \dots + v_n^2}.$$

A normalização é a obtenção de um vetor \vec{w}

$$\vec{w} = \frac{\vec{v}}{\|\vec{v}\|} = \left[\frac{v_1}{\|\vec{v}\|}, \frac{v_2}{\|\vec{v}\|}, \dots, \frac{v_n}{\|\vec{v}\|} \right].$$

A soma de dois vetores é o vetor

$$\vec{v} + \vec{w} = [v_1 + w_1, v_2 + w_2, \dots, v_n + w_n].$$

O produto escalar de dois vetores é:

$$\vec{v} \cdot \vec{w} = \sum_{i=1}^n v_i w_i = v_1 w_1 + v_2 w_2 + \dots + v_n w_n$$

A soma e o produto escalar somente fazem sentido se os vetores estiverem em espaços de mesma dimensão. Quando se tentar realizar uma dessas operações e a dimensão dos vetores não for igual, deve-se “promover” o vetor de menor dimensão para o espaço de maior dimensão. Por exemplo, se $\vec{v} = [4, 5]$ (isto é, em \mathbb{R}^2) e $\vec{w} = [1, -2, 3]$ (em \mathbb{R}^3), deve considerar $\vec{v} = [4, 5, 0]$, resultando a soma em $[5, 3, 3]$. O produto escalar $\vec{v} \cdot \vec{w}$ resultará em $4 \cdot 1 + 5 \cdot -2 + 0 \cdot 3$, ou seja, -6 .

4.8 Arquivos

4.8-1 Digite e execute o Programa 4.8-1. Entenda como ele funciona. Em particular:

- Identifique a criação de um arquivo vazio (modo “w” do `fopen`)
- Entenda a gravação de cada linha digitada no arquivo (`fprintf`)
- Identifique o encerramento de acesso o arquivo (`fclose`)

Programa 4.8-1 Manipulação básica de arquivo.

```
/*
Input: Um nome de arquivo texto, uma
sequencia de linhas de texto nao
vazias seguida por uma linha vazia
Output: A mesma sequencia (exceto pela
linha vazia) escrita em arquivo
*/
#include <stdio.h>
#include <string.h>

int main(){
    char linha[300], nome[200];
    FILE *arquivo;

    // nome do arquivo
    printf("Nome do arquivo: ");
    fgets(nome, sizeof nome, stdin);
    nome[strlen(nome) - 1] = '\0';

    // criacao de um arquivo vazio
    arquivo = fopen(nome, "w");

    // gravacao de dados no arquivo
    do{
        printf("> ");
        fgets(linha, sizeof linha, stdin);
        if(strcmp(linha, "\n") != 0)
            fprintf(arquivo, "%s", linha);
    }while(strcmp(linha, "\n") != 0);

    // encerramento
    fclose(arquivo);
    return 0;
}
```

Arquivo: material/criaarquivotexto.c.

4.8-2 Escreva um programa em C que peça ao usuário um nome de arquivo e crie um arquivo vazio com o nome dado.

A criação pode ser feita abrindo-se um arquivo (modo criação) e fechando-o logo em seguida, sem escrever nada como conteúdo.

4.8-3 Escreva um programa em C que solicite ao usuário um nome de arquivo (modo texto) e dois valores inteiros como número de linhas inicial e final. O programa deve apresentar as linhas do arquivo começando na linha inicial e indo até a final indicadas.

Assuma, para este exercício, que os números de linha serão sempre em ordem (i.e., a linha inicial será sempre menor ou igual à final) e que ambos os valores são linhas válidas no arquivo. Portanto, não faça teste de consistência.

O arquivo `livro.txt` pode ser usado para testes; ele possui 48.452 linhas no total. Como exemplo, as linhas de 203 a 209 contêm o seguinte conteúdo:

Beatrice Tarleton was a busy woman, having on her hands not only large cotton plantation, a hundred negroes and eight children, the largest horse-breeding farm in the state as well. She was hot-tempered and easily plagued by the frequent scrapes of her four sons, and while no one was permitted to whip a horse or a slave, she felt that a lick now and then didn't do the boys any harm.

Além disso, testes podem ser feitos com os códigos fonte dos programas em C, que também são arquivos texto simples.

4.8-4 Analise o código fonte do Programa ???. Entenda-o e depois compile e execute. Note que ele usa o arquivo de dados `dados.txt` como arquivo de entrada. O código exige que, a primeira linha do arquivo contenha a quantidade de dados, seguida pelos dados, um por linha.

Programa 4.8-2 Média de dados de um arquivo texto, com especificação da quantidade de dados.

```
/*
   Output: A soma dos valores inteiros contidos
   em
   um arquivo texto com nome 'dados.txt'
   formatado
   como se segue:
   - primeira linha: quantidade de linhas
   com dados
   - demais linhas: valores dos dados, um
   por linha
*/
#include<stdio.h>

int main(){
    int n, i, quantidade;
    double valor, soma, media;
    FILE *arqdados;

    // acesso ao arquivo (nome fixo)
    arqdados = fopen("dados.txt", "r");

    // obtencao da quantidade (1a linha)
    fscanf(arqdados, "%d", &quantidade);

    // leitura e processamento de cada dado
    soma = 0;
    for(int i = 0; i < quantidade; i++){
        fscanf(arqdados, "%lf", &valor);
        soma += valor;
    }
    media = soma/quantidade;

    // encerramento de acesso ao arquivo
    fclose(arqdados);

    // apresentacao do resultado na tela
    if(quantidade > 0)
        printf("media = %g\n", media);

    return 0;
}
```

Arquivo: material/mediaquantidade.c.

4.8-5 Considere o código do Programa 4.8-2, que usa o arquivo `dados.txt`.

Altere o arquivo de dados de forma a torná-lo inconsistente, avaliando como o programa se comporta:

- Coloque **mais** dados do que a quantidade especificada na primeira linha;
- Coloque **menos** dados do que a quantidade especificada na primeira linha.

No segundo caso, especificamente, você consegue identificar o que leva o programa a escrever o que ele escreve?

4.8-6 Analise o código fonte do Programa 4.8-3. Entenda-o e depois compile e execute. Note que ele usa o arquivo de dados `dados2.txt` como arquivo de entrada, o qual contém os valores um por linha.

Programa 4.8-3 Média de dados de um arquivo texto.

```
/*
   Output: A media real dos valores inteiros
   contidos
   em um arquivo texto com nome 'dados.txt',
   com
   um valor por linha
*/
#include<stdio.h>

int main(){
    int n, i, quantidade;
    float valor, soma, media;
    FILE *arqdados;

    // acesso ao arquivo
    arqdados = fopen("dados.txt", "r"); //
    arquivo fixo

    // leitura e processamento de cada dado
    soma = 0;
    quantidade = 0;
    while(fscanf(arqdados, "%f", &valor) !=
        EOF){
        quantidade++;
        soma += valor;
    }
    media = soma/quantidade;

    // encerramento de acesso ao arquivo
    fclose(arqdados);

    // apresentacao do resultado na tela
    if(quantidade > 0)
        printf("media = %g\n", media);

    return 0;
}
```

Arquivo: material/mediaarquivo.c.

4.8-7 Considere arquivos texto contendo dados inteiros, um por linha, podendo haver valores positivos ou negativos. Deseja-se saber a soma de todos os valores contidos no arquivo.

Escreva um programa em C que leia o nome de um arquivo que siga essa especificação e apresente a soma de todos os valores.

Tabela 4.8-1: Tabela verdade para a operação de ou-exclusivo.

a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

Edite em um editor de texto simples alguns exemplos para realizar os testes.

4.8-8 Criptografia³ ou criptologia (significando “escrita escondida”) é o estudo e prática de princípios e técnicas para comunicação segura na presença de terceiros.

Este exercício consiste em criptografar o conteúdo de um arquivo usando uma *chave*. Neste caso, a chave será de apenas um caractere dado pelo usuário.

Esta criptografia simples é baseada na operação *ou-exclusivo*. A operação binária chamada *ou-exclusivo* (*exclusive-or*, ou *xor*, com operador \oplus) está apresentada na Tabela 4.8-1.

Basicamente, a operação *ou-exclusivo* resulta em 0 se os operadores forem iguais e em 1 caso sejam diferentes.

Cada caractere tem sua representação na tabela ASCII⁴ e, portanto, tem sua representação binária com 8 bits (1 byte). Estas são consideradas neste exercício.

Em C, existe o operador \sim (circunflexo), que faz a operação *xor* bit a bit. Considerando-se, por exemplo, os caracteres '\$' e 'Z' possuem codificações binárias 00100100 e 01011010, respectivamente⁵. A operação '\$' \sim 'Z' resulta no padrão 01111110, que é o caractere '~' (til).

Uma propriedade interessante do *ou-exclusivo* é

$$(a \oplus b) \oplus b = a.$$

Ou seja, se aplicada a mesma operação duas vezes, o valor original é restaurado. Assim, '\$' \sim 'Z' resulta em '\$'.

Escreva um programa que solicite ao usuário o nome de um arquivo já existente e o nome do arquivo onde ele será criptografado. Um caractere chave deve também ser solicitado. O programa deve ler o arquivo de entrada caractere por caractere e, no arquivo de saída, escrever o resultado do *ou-exclusivo* entre o caractere de entrada e o caractere chave.

Experimente criptografando o arquivo `livro.txt`. Outro experimento interessante é tentar criptografar um arquivo que não seja texto e verificar se é possível recuperá-lo; como sugestão, crie um arquivo `.doc` ou `.zip` para testar.

4.8-9 Escreva um programa em C que solicite ao usuário um nome de arquivo (modo texto) e um texto qualquer, escrevendo todas as linhas em que o texto indicado aparecer.

O código deve fazer comparação exata, ou seja, “Abc” \neq “ABC”, “Pele” \neq “Pelé” etc.

Um arquivo chamado `livro.txt` foi disponibilizado para este exercício. Sugestões de texto para pesquisa: “bonneted” (que aparece em 2 linhas), “self-reliance” (1 linha), “foundations” (7 linhas) e “birthday” (9 linhas). Procure também por sentenças, incluindo espaços e pontuações (e.g, “then, as”, em 3 linhas, ou “about_Ashley”, em 23 linhas).

4.8-10 Digite e execute o Programa 4.8-10. Entenda como ele funciona. Em particular:

- Atente para o modo de abertura como dado binário: “rb”;
- Entenda cada parâmetro do `fread` e seu uso no programa.

Programa 4.8-4 Apresentação de um arquivo byte a byte, mostrando os valores em hexadecimal, decimal e caractere.

```
/*
Input: Um nome de arquivo
Output: O conteúdo do arquivo, byte a byte,
expresso em valor hexadecimal e decimal
*/
#include <stdio.h>
#include <string.h>

int main(){
    unsigned char byte;
    int cont, n_itens;
    char nome_arquivo[100];
    FILE *arquivo;

    // obtencao de dados
    printf("Nome do arquivo: ");
    fgets(nome_arquivo, sizeof nome_arquivo,
        stdin);
    nome_arquivo[strlen(nome_arquivo) - 1] =
        '\0';

    // apresentacao do arquivo, byte a byte
    arquivo = fopen(nome_arquivo, "rb");

    cont = 0; // numero do byte
    do{
        n_itens = fread(&byte, sizeof byte,
            1, arquivo);
        if(n_itens > 0)
            printf("%05d: %02x; %3d\n", cont++,
                byte, byte);
    }while(n_itens > 0);

    fclose(arquivo);

    return 0;
}
```

Arquivo: material/printbyte.c.

Para testes, use qualquer arquivo, pois será considerado um arquivo binário de bytes.

³<https://pt.wikipedia.org/wiki/Criptografia>.

⁴Isso é válido para caracteres gerais; há representações em que acentuações ou outros símbolos usam mais que um byte.

⁵<https://pt.wikipedia.org/wiki/ASCII>.

4.8-11 Execute o Programa 4.8-5. Nele, analise e compreenda:

- O uso da função `fread` com o tipo `int`;
- O uso do `fread` dentro da condição do `if`.

Programa 4.8-5 Apresentação de um arquivo de ints, inteiro a inteiro.

```
/*
   Input: O nome de um arquivo binario contendo
          valores do tipo int
   Output: Todos os valores inteiros contidos
          no
          arquivo na tela
*/
#include <stdio.h>
#include <string.h> // para strlen

int main(){
    int inteiro; // para leitura
    int cont, n_itens;
    char nome_arquivo[100];
    FILE *arquivo;

    // obtencao de dados
    printf("Nome do arquivo: ");
    fgets(nome_arquivo,
          sizeof nome_arquivo, stdin);
    nome_arquivo[strlen(nome_arquivo) - 1] =
        '\0';

    // apresentacao do arquivo, int a int
    arquivo = fopen(nome_arquivo, "rb");

    cont = 0;
    while(fread(&inteiro, sizeof inteiro,
        1, arquivo) > 0)
        printf("%05d: %12d\n", cont++, inteiro);

    fclose(arquivo);

    return 0;
}
```

Arquivo: material/printint.c.

Teste este programa com o arquivo `dados.int`.

4.8-12 Observe e execute o Programa 4.8-6 e:

- Observe os valores escritos para a posição corrente (chamadas a `ftell`);
- Tenha certeza da razão do valor ir de quatro em quatro;
- Entenda que a posição corrente é atualizada automaticamente a cada leitura.

Teste este programa com o arquivo `dados.float`.

4.8-13 Crie um arquivo binário que armazene dados de jogadores de um time de futebol. Cada jogador deve ter os seguintes dados armazenados: número da camisa, nome completo, apelido, posição que joga, altura (em metros) e peso (em quilogramas).

Escreva um programa em C para criar o arquivo de dados e outro para mostrar seu conteúdo. As escolhas dos tipos dos campos e os comprimentos das cadeias de caracteres pode ser arbitrária.

Programa 4.8-6 Apresentação dos dados em um arquivo contendo floats e das posições corrente do arquivo em pontos de interesse.

```
/*
   Input: O nome de um arquivo contendo float's
   Output: Todos os valores contidos no arquivo
           juntamente com a posicao corrente do
           arquivo em pontos de interesse
*/
#include <stdio.h>
#include <string.h>

int main(){
    float valor; // para leitura
    int cont, n_itens;
    char nome_arquivo[100];
    FILE *arquivo;

    // obtencao de dados
    printf("Nome do arquivo: ");
    fgets(nome_arquivo,
          sizeof nome_arquivo, stdin);
    nome_arquivo[strlen(nome_arquivo) - 1] =
        '\0';

    // apresentacao do arquivo, float a float
    arquivo = fopen(nome_arquivo, "rb");
    printf("Arquivo aberto; posicao "
        "corrente: %ld\n", ftell(arquivo));

    cont = 0;
    while(fread(&valor, sizeof valor, 1,
        arquivo) > 0){
        printf("%05d: %g\n", cont++, valor);
        printf("Leitura realizada; posicao "
            "corrente: %ld\n", ftell(arquivo));
    }
    fclose(arquivo);

    return 0;
}
```

Arquivo: material/printfloat2.c.

4.8-14 Considere o arquivo de dados gerado no Exercício 4.8-13. Escreva um programa em C para solicitar a posição do registro e apresentar os dados do jogador correspondente. O primeiro registro deve ser considerado com 1 (apesar do seu RRN ser 0). Caso seja fornecido um número inválido, a mensagem “Entrada inválida” deve ser apresentada.

4.8-15 Crie um arquivo texto que armazene dados de jogadores de um time de futebol, um jogador por linha. Cada jogador deve ter os seguintes dados armazenados: número da camisa, nome completo, apelido, posição que joga, altura (em metros) e peso (em quilogramas).

Escreva um programa em C para criar o arquivo de dados e outro para mostrar seu conteúdo.

4.8-16 Considere o arquivo de dados gerado no Exercício 4.8-15. Escreva um programa em C para solicitar a posição do registro e apresentar os dados do jogador correspondente. O primeiro registro deve ser considerado com 1 (apesar do seu RRN ser 0). Caso seja fornecido um número inválido, a mensagem “Entrada inválida” deve ser apresentada.

4.8-17 Observe e execute o Programa 4.8-7. Em um primeiro momento:

- Entenda como o único `fread` do programa funciona, em especial pela quantidade de elementos a serem lidos ser igual a `MAX`;
- Entenda o papel da variável `n_lidos`.

Em um segundo momento, experimente alterar o valor de `MAX` para mais ou para menos, observando o que o programa escreve.

Como testes sugeridos, use `MAX` igual a 1, que transforma o vetor em um único `float` e `MAX` igual a 1000, valor certamente superior à quantidade de valores contidas no arquivo.

Teste este programa com o arquivo `dados.float`.

4.8-18 Crie um programa para salvar registros no seguinte formato:

```
struct dado{
    char nome[40];
    char genero;    // 'M', 'F' ou 'O'
    int idade;
    float salario;
};
```

Lembre-se de que, ao gravar com o `fwrite`, o uso de `sizeof` garante que a quantidade correta de bytes será transferida.

4.8-19 Em relação ao Exercício 4.8-18, altere seu programa para que escreva na tela o tamanho do registro em bytes, usando o operador `sizeof`.

O tamanho apresentado é tem o número de bytes que você esperava?

Programa 4.8-7 Leitura de um vetor de floats para um vetor em memória principal.

```
/*
    Leitura de dados para um vetor
    Jander, para CAP
*/
#include <stdio.h>
#define MAX 5

int main(){
    float vetor[MAX];
    int n_lidos, i;
    FILE *arquivo;

    // obtencao dos dados
    arquivo = fopen("dados.float", "rb");
    n_lidos = fread(vetor, sizeof (float),
        MAX, arquivo);
    fclose(arquivo);

    // apresentacao dos dados do vetor
    for(i = 0; i < n_lidos; i++)
        printf("vetor[%3d] = %g\n", i, vetor[i]);

    return 0;
}
```

Arquivo: material/readvec.c.

4.8-20 A criptografia do Exercício 4.8-8 pode ser quebrada com no máximo 255 tentativas (ou seja, todas as possibilidades). Optando-se pela ordem mais natural, como tentar letras e dígitos antes de caracteres menos usuais pode reduzir bastante o número médio de tentativas.

Essa criptografia simples pode ser incrementada usando-se uma frase (*passphrase*) no lugar de um único caractere. Cada caractere do arquivo de entrada seria “xored” com um caractere diferente da frase chave, ciclicamente.

Por exemplo, se a frase chave for “cAp”, a primeira saída seria o `xor` com `c`, a segunda o `xor` com `A`, seguida do `xor` com `p`, depois novamente com `c`, `A`, `p`, `c`, `A`. . . E o efeito seria que um mesmo caractere na entrada geraria saídas diferentes, dependendo de com qual elemento da frase fosse feita a operação. Quanto maior for o comprimento da frase, mais complicado fica de quebrar a criptografia por tentativa e erro, pois aumenta muito o número de combinações.

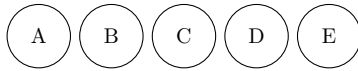
Crie uma nova versão do programa de criptografia usando uma *passphrase*.

5 Comentários de exercícios selecionados

✓ **3.1-1** A receita é um algoritmo bastante preciso, considerando-se que quem vai executá-la possui alguma experiência na cozinha.

O uso do imperativo indica que se tratam de comandos a serem executados e cada passo deve ser executado totalmente antes do início do próximo.

✓ **3.1-3** Sua resposta ao exercício foi a seguinte?



Ou foi alguma das abaixo?

- 1.
- 2.
- 3.
- 4.

É possível ver que qualquer uma das alternativas apresentadas também é uma resposta correta?

✓ **3.1-4** O algoritmo é confuso em relação à execução sequencial. Seguido à risca, a última instrução, que contém uma condição, somente será executada depois das demais, o que a torna incoerente, pois o círculo já foi desenhado.

Este não é um bom algoritmo. Uma alternativa seria o Algoritmo 5-1.

Algoritmo 5-1 Instruções para desenho.

- Desenhe um quadrado
 - Desenhe um losango no canto superior esquerdo do quadrado
 - Desenhe um círculo no canto inferior direito do quadrado
 - Escreva sua idade no canto inferior esquerdo do quadrado
 - Escreva o ano atual no canto superior direito do quadrado
 - Calcule o valor de $e = 2i + 1$, sendo i sua idade em anos
 - Se e for ímpar, desenhe um triângulo no centro do quadrado; senão, desenhe um círculo
-

✓ **3.3-2** Os Algoritmos Algoritmos 5-2 e 5-3 fornecem uma solução de indentação para os Algoritmos 3.3-2 e 3.3-3, respectivamente.

✓ **3.3-3** Os algoritmos são completamente equivalentes. Há somente o uso de uma notação mais compacta no caso do Algoritmo 3.3-4.

Algoritmo 5-2 Algoritmo 3.3-2 com indentação corrigida.

Input: Dois valores $v_1, v_2 \in \mathbb{R}$

Output: A média m dos dois valores

- Obtenha os valores v_1 e v_2
 - Calcule $m = (v_1 + v_2)/2$
 - Apresente m
-

Algoritmo 5-3 Algoritmo 3.3-3 com indentação corrigida.

Input: O comprimento

$c_1, c_2, c_3 \in \mathbb{R}^{*+}$ que formam um triângulo válido

Output: A área a do triângulo

Obtenha os valores

c_1, c_2 e c_3

Calcule p como

$$\frac{c_1 + c_2 + c_3}{2}$$

Calcule a como

$$\sqrt{p(p - v_1) \cdot (p - v_2)(p - v_3)}$$

Apresente a área calculada a

✓ **3.3-11** Estão incorretas as expressões seguintes:

- $x > 10$ ou $-1 < x < 1$
 $-1 < x < 1$ deve ser escrito $-1 < x$ e $x < 1$
- $x = 1$ ou 2 ou 3
deve ser escrita $x = 1$ ou $x = 2$ ou $x = 3$
- $x_1, x_2, x_3 \geq 0$
deve ser escrita $x_1 \geq 0$ e $x_2 \geq 0$ e $x_3 \geq 0$

Note que nas especificações de entradas e saídas, expressões como $x_1, x_2, x_3 \geq 0$ podem ser usadas. A restrição ocorre apenas nas instruções.

✓ **3.3-15** Veja o Algoritmo 5-4. Você esperava por esta solução?

Algoritmo 5-4 Determinação se um valor de 1 a 10 é um número perfeito.

Input: Um valor $n \in \{1, 2, \dots, 10\}$

Output: **true**, se n for perfeito, **false** caso contrário.

Obtenha n

$v \leftarrow n = 6$

▷ Único perfeito no intervalo

Apresente v

✓ **3.5-1** No Algoritmo 3.5-1:

- Na linha 5 apenas se sabe que segmentos dos comprimentos dados podem formar um triângulo;
- Na linha 8, sabe-se que há pelo menos um dos lados com comprimento diferente dos demais;

- Na linha 11 somente é possível que todos os comprimentos sejam diferentes.

✓ **3.5-2** Qual o resultado do **if** da linha 8 quando todos os comprimentos são iguais?

✓ **3.5-15** Dica: Veja se uma tabela verdade como a do Exercício 3.3-12 pode ajudar a chegar na expressão desejada para o ano bissexto.

✓ **3.5-17** Em sua solução ficou claro que os ângulos são dados em graus e não em radianos?

✓ **3.6-5** O Algoritmo 3.6-11 não produz qualquer saída, pois o **for** é considerado sempre crescente com incremento um, a não ser que explicitamente indicado o contrário.

Assim, para se ter uma repetição decrescente, há duas opções:

```
for e ← 10 downto 1 do
end for
```

ou

```
for e ← 10 to 1 step -1 do
end for
```

✓ **3.6-12** Uma dica para desenvolver este algoritmo é pensar em uma situação tão concreta quanto possível. Pode-se supor que quem executará o algoritmo dispõe de papel e lápis para as anotações (Figura 5-1); as instruções indicam como, quais e quando as diversas anotações serão feitas.

Figura 5-1: Contagem. Fonte: https://farm7.static.flickr.com/6220/6274747634_e62efe11b2_b.jpg.



✓ **3.6-21** O Algoritmo 5-5 apresenta uma solução.

✓ **3.8-3** Um algoritmo deve ser claro e preciso quanto à solução que ele apresenta. Usar x no lugar de $P.x$ no Algoritmo 3.8-1 não prejudica seu entendimento. Entretanto, usar apenas x no Algoritmo 3.8-2 pode gerar dúvidas, pois pode ser referência à primeira coordenada tanto de P_1 quanto de P_2 .

Se a entrada tivesse sido formulada de forma diversa, a dúvida poderia não existir.

Algoritmo 5-5 Determinação da quantidade de temperaturas superiores a 25°C até o primeiro valor nulo.

Input: Uma sequência infinita de temperaturas $t \in \{x \in \mathbb{R} \mid x > 25 \text{ ou } x = 0\}$

Output: A quantidade q de temperaturas acima de 25°C até o primeiro valor nulo

$c \leftarrow 0$ ▷ contador inicia com zero

read t ▷ primeira leitura

while $t \neq 0$ **do**

if $t > 25$ **then**

$c \leftarrow c + 1$

▷ conta +1

end if

read t

▷ próxima leitura

end while

write c

▷ resultado

Input: Os pontos $P = (p_x, p_y, p_z)$ e $Q = (q_x, q_y, q_z)$ em \mathbb{R}^3

...

...

$$d_p = \sqrt{p_x^2 + p_y^2 + p_z^2}$$

...

✓ **3.9-2** A sua solução não deveria incluir nenhuma instrução envolvendo **read** ou **write**. Houve uso desse recurso?

✓ **3.9-3** Uma solução pode ser dada pelo Algoritmo 5-6, no qual se supõe que $\text{MÁXIMO}(x, y)$ retorna o máximo entre x e y .

Algoritmo 5-6 Determinação do máximo entre três valores reais

Input: Três valores $a, b, c \in \mathbb{R}$

Output: O valor máximo entre eles

function MÁXIMO3(a, b, c)

return MÁXIMO(MÁXIMO(a, b), c)

end function

✓ **4.1-4** Um valor que possua parte decimal é considerado real, mesmo que essa parte seja nula. Inteiros simplesmente não possuem parte decimal.

Na maior parte das linguagens de programação, 2.0 é representado de forma diferente de 2.

✓ **4.1-6** Todas as opções podem ser usadas como valores inteiros, incluindo **char** e **unsigned char**. Apenas o número de bits de cada representação varia.

✓ **4.1-8** Todos as constantes numéricas iniciadas com o dígito 0 são interpretadas na base 8 (octal). Números octais usam os dígitos de 0 a 7 e, portanto, 080 não é um número octal válido.

O valor octal 70 (escrito usualmente 70₈ ou 70₍₈₎) equivale a 56 na base decimal.

✓ **4.1-11** Uma referência interessante: https://www.tutorialspoint.com/cprogramming/c_constants.htm.

✓ **4.1-12** A afirmação não é correta, pois não há como garantir que uma dada posição da memória tenha um valor específico *a priori*. Variáveis não iniciadas devem ser consideradas como contendo “lixo”, ou seja, um valor desconhecido.

Usar uma variável assim pode levar a resultados inesperados.

✓ **4.2-3** O problema **não** é a precisão, como é o caso do Exercício 4.2-6 e Programa 4.2-2.

Dica: insira comandos para escrever cada uma das duas expressões comparadas.

✓ **4.2-6** O número de bits para armazenar um valor `float` ou mesmo `double` é limitado. Assim, o valor efetivamente armazenado pode ser apenas uma aproximação do valor desejado.

Para o exemplo do Programa 4.2-2, o valor realmente armazenado em `f` é 3,769999980926513671875 (erro de cerca de $1,9 \cdot 10^{-8}$ em relação a 3,77).

Visite <https://www.h-schmidt.net/FloatConverter/IEEE754.html> e faça alguns testes.

Note, ainda, que esta é uma situação diferente da do Exercício 4.2-3 e do Programa 4.2-1.

✓ **4.3-6** Em C, a implementação direta faria uso da estrutura homônima `switch`. Porém a linguagem limita essa estrutura aos tipos escalares (i.e., `int`, `char` etc.) e, portanto, não se aplica a cadeias de caracteres. A implementação exige que se opte pela estrutura `if`, com seus `elses`.

✓ **4.4-1** Cuidado ao codificar o **repeat-until** para **do-while**: a condição de parada não é a mesma. Uma é a negação da outra: **repeat ... until** $t = 0$ deve ficar **do ... while** ($t \neq 0$), por exemplo.

✓ **4.4-11** Sua solução considerou que o maior divisor possível de um número n é $n/2$? Seu algoritmo procura divisores para valores maiores que $n/2$?

✓ **4.4-19** Há dois pontos importantes a salientar.

O primeiro é que as variáveis não possuem identificadores significativos. O uso de uma variável *soma* para acumular os valores já ajudaria e a separaria da variável *i*.

O segundo ponto é a “distância” da iniciação da variável *i* com zero da sua real utilização. Embora a linguagem C permita declarações já com iniciações, se a iniciação $i = 0$ estivesse junto com a repetição que efetivamente faz a soma, a inserção do novo código (o `for`) seria provavelmente feita *antes* da iniciação e a chance de prejudicar a lógica do programa ficaria em muito reduzida. Manter comandos relacionados próximos no código é importante para a organização do programa.

✓ **4.5-6** Lembre-se que, para escrever *i*, basta usar:

```
printf("i");
```

Assim, para escrever $2 + 7i$ pode-se usar o comando seguinte, assumindo-se que `real` e `imaginario` sejam variáveis do tipo `double` com valores 2 e 7, respectivamente:

```
printf("%g + %gi", real, imaginario);
```

Note o uso do formato `“%g”`, já ajusta o número necessário de casas decimais.

✓ **4.5-13** Este exercício parece trivial, mas... os dados ficaram mesmo invertidos na sua solução? Escreva seu programa e o execute. Verifique que realmente os dados ficam no vetor na ordem inversa à original.

✓ **4.5-20** Dicas: (a) use as funções de manipulação de cadeias de caracteres, disponíveis pela inclusão do cabeçalho `string.h` para determinar o número total de caracteres ou verifique o final da cadeia localizando o sentinela de caractere nulo `‘\0’`.

✓ **4.6-13** O resultado da execução do Programa 4.6-6 é

```
original:  [ 1 2 3 4 5 6 7 8 ]
modificado: [ 99 33 3 4 40 6 80 8 ]
```

Em particular:

- `vet[4]` é a quinta posição do vetor;
- `*pont` é o início do vetor e equivale a `vet[0]`;
- `*(pont + 1)` é um elemento a mais em relação ao início do vetor e equivale a `vet[1]`;
- `pont[i]` equivale a `*(pont + i)`, de forma que `pont[6]` e `pont[7]` equivalem a `vet[6]` e `vet[7]`, respectivamente.

✓ **4.7-11** Veja que sua solução pode ser semelhante ao código seguinte. A função `origem()` retorna o registro com todas as coordenadas nulas.

```
/*
   Input: Um ponto (ponto3d)
   Output: A distancia do ponto a origem
*/
double distancia_origem(struct ponto3d ponto){
    return distancia_pontos(ponto, origem());
}
```

✓ **4.7-20** Seu procedimento também alterou o número de elementos válidos ou apenas colocou o novo elemento na parte “irrelevante” dos dados?

O problema não diz o que fazer caso não haja espaço para novos elemento no vetor (i.e., quando $n = t$). Você considerou essa hipótese?

✓ **4.7-22** Você considerou, na sua resposta, em usar a função escrita para o Exercício 4.7-21?

Lista de figuras

3.1-1	Receita para preparação de moussaka.	2
3.2-1	Exemplo de alvo com pontuações.	4
3.7-1	Rascunho de um jogo simples.	17
4.4-1	Esquema de aquecimento de água.	26
4.5-1	Círculo desenhado com caracteres.	32
5-1	Contagem.	44

Lista de tabelas

3.3-1	Alguns dados sobre alunos ingressantes no ensino superior.	5
3.5-1	Relação entre conceitos e notas.	7
3.5-2	Relação entre notas e conceitos.	7
3.5-3	Exemplos de valores de numerador e denominador de uma razão e a representação “canônica” esperada.	9
3.5-4	Fator de multiplicação para conversão de unidades de medida de velocidade.	10
3.5-5	Exemplos de alguns anos bissextos e não bissextos.	10
3.6-1	Faixas etárias.	14
4.4-1	Exemplo de temperaturas coletadas por sensores e a diferença entre elas.	26
4.8-1	Tabela verdade para a operação de ou-exclusivo.	40

Lista de quadros

3.3-1	Tabela verdade. V indica verdadeiro (true) e F indica falso (false).	6
4.1-1	Tipos das constantes	19
4.1-2	Opção por tipo de dados.	20
4.1-3	Número de bytes para alguns tipos de dados em C.	20
4.3-1	Entradas e saídas para o Algoritmo 4.3-2.	23
4.3-2	Entradas e saídas para o Algoritmo 4.3-3.	23
4.5-1	Formatação de números complexos.	29
4.7-1	Estruturação de um ponto no espaço com um registro.	36
4.7-2	Descrição do vetor.	37

Lista de algoritmos

3.1-1	Instruções para o desenho.	2
3.1-2	Instruções para desenho.	2
3.1-3	Determinação do valor máximo.	3
3.3-1	Algoritmo para conversão de coordenadas polares para cartesianas.	4
3.3-2	Algoritmo com indentação inadequada.	4
3.3-3	Algoritmo com indentação inadequada.	4
3.3-4	Algoritmo para conversão de coordenadas polares para cartesianas.	4
3.3-5	Cálculo da matriz de covariância.	5
3.3-6	Determinação de validade de intervalo de uma nota acadêmica.	5

3.5-1	Determinação de existência e tipo de triângulo a partir dos comprimentos de três segmentos de reta.	7
3.5-2	Determinação de existência e tipo de triângulo a partir dos comprimentos de três segmentos de reta (com erro de lógica).	7
3.5-3	Determinação do número de raízes reais e distintas de uma equação de segundo grau.	7
3.5-4	Média de dois conceitos.	8
3.5-5	Determinação de valor máximo.	8
3.5-6	Outra determinação de valor máximo.	8
3.5-7	Ainda outra determinação de valor máximo.	8
3.5-8	Ainda outra determinação de valor máximo.	8
3.5-9	Determinação de existência e tipo de triângulo a partir dos comprimentos de três segmentos de reta.	8
3.5-10	Projeção de rendimentos para aplicação com liquidez diária.	9
3.6-1	Apresentação de valores.	11
3.6-2	Apresentação de valores.	11
3.6-3	Apresentação de valores.	11
3.6-4	Apresentação de valores.	11
3.6-5	Apresentação de valores.	11
3.6-6	Apresentação de valores.	11
3.6-7	Apresentação de valores.	12
3.6-8	Apresentação de valores.	12
3.6-9	Apresentação de valores	12
3.6-10	Apresentação de valores	12
3.6-11	Apresentação de valores	12
3.6-12	Atendimento em estabelecimento comercial.	12
3.6-13	Levantamento de dados sobre vendas e compras.	13
3.6-14	Determinação do valor máximo.	14
3.6-15	Determinação do valor médio de idades, dados a quantidade de amostras e os valores individuais.	14
3.6-16	Determinação de valor máximo e existência de pares para 10 valores inteiros.	16
3.7-1	Determinação da equipe que será escolhida para uma campanha publicitária.	16
3.7-2	Contagem de meses em que houve maior volume de exportações que de importações	16
3.7-3	Contagem de meses em que houve maior volume de exportações que de importações. Solução incorreta.	17
3.8-1	Determinação da distância de um ponto no espaço à origem.	17
3.8-2	Determinação do ponto mais próximo da origem, sendo irrelevante a resposta no caso de equidistantes.	18
3.8-3	Processo seletivo com dados de candidatos entregues na ordem de término das provas. Os empates são decididos em favor do que terminou antes.	18

3.8-4	Determinação do número mínimo de cé- dulas e moedas para uma dada quanti- dade em reais (R\$).	18	4.8-2	Media de dados de um arquivo texto, com especificação da quantidade de dados. . .	39
3.9-1	Código exemplo.	19	4.8-3	Média de dados de um arquivo texto. . .	39
4.3-1	Apresentação das raízes reais de uma equação de segundo grau.	22	4.8-4	Apresentação de um arquivo byte a byte, mostrando os valores em hexade- cimal, decimal e caractere.	40
4.3-2	Algoritmo com condicionais.	23	4.8-5	Apresentação de um arquivo de <code>ints</code> , in- teiro a inteiro.	41
4.3-3	Algoritmo com condicionais.	23	4.8-6	Apresentação dos dados em um arquivo contendo <code>floats</code> e das posições corrente do arquivo em pontos de interesse. . . .	41
4.3-4	Número de dias segundo o nome do mês, ignorando anos bissextos.	24	4.8-7	Leitura de um vetor de <code>floats</code> para um vetor em memória principal.	42
4.4-1	Cálculo do fatorial de n	25			
4.4-2	Cálculo do fatorial de n	25			
4.4-3	Máximo divisor comum pelo método de Euclides.	25			
5-1	Instruções para desenho.	43			
5-2	Algoritmo 3.3-2 com indentação corrigida. . .	43			
5-3	Algoritmo 3.3-3 com indentação corrigida. . .	43			
5-4	Determinação se um valor de 1 a 10 é um número perfeito.	43			
5-5	Determinação da quantidade de tempe- raturas superiores a 25 °C até o primeiro valor nulo.	44			
5-6	Determinação do máximo entre três va- lores reais	44			

Lista de programas

4.1-1	Programa para apresentar o tamanho do tipo <code>double</code>	20
4.1-2	Comparação de literais.	21
4.2-1	Operações comutativas?	21
4.2-2	Comparação de valores reais.	22
4.2-3	Testes com o operador <code>++</code>	22
4.3-1	Cálculo da média.	23
4.3-2	Programa em C sem objetivo definido com erro de compilação.	24
4.4-1	Programa com repetição simples e um erro de codificação.	24
4.4-2	Programa com repetição simples.	24
4.4-3	Soma simples de valores positivos.	27
4.5-1	Programa com registro.	28
4.5-2	Programa com registro.	28
4.5-3	Leitura e escrita de um ponto em \mathbb{R}^2 . . .	28
4.5-4	Programa com registro.	29
4.5-5	Determinação do número mínimo de cé- dulas ou moedas para uma dada quan- tidade monetária	29
4.5-6	Porcentagem de dados acima da média. . . .	30
4.5-7	Manipulação simples de matriz.	32
4.6-1	Apresentação de alguns endereços de memória.	33
4.6-2	Inspeção de conteúdo de memória com o operador <code>*</code>	33
4.6-3	Modificação de conteúdo de memória com o operador <code>*</code>	33
4.6-4	Uso de múltiplos ponteiros.	34
4.6-5	Acesso a registro por ponteiro.	34
4.6-6	Exemplos de uso de ponteiros com vetores. .	35
4.7-1	Programa para teste.	35
4.8-1	Manipulação básica de arquivo.	38