BASES COMPUTACIONAIS DA CIÊNCIA

MARIA DAS GRAÇAS BRUNO MARIETTO
MÁRIO MINAMI
PIETER WILLEM WESTERA
(ORGS.)

UFABC

CATALOGAÇÃO NA FONTE
SISTEMA DE BIBLIOTECAS DA UNIVERSIDADE FEDERAL DO ABC
Responsável: Roberta Kelly Amorim de França CRB: 7660

511.3
BASE
Bases computacionais da ciência / Organizado por Maria das Graças Bruno Marietto, Mário Minami, Pieter Willem Westera. — Santo André:
Universidade Federal do ABC, 2013.

242 p.
ISBN: 987 - 85 - 65212 - 21

1. Computação – teoria e prática 2.Comunicação e Informação 3. Bacharelados Interdisciplinares de Ciências - UFABC I. MARIETTO, Maria das Graças Bruno. II. MINAMI, Mário. III. WESTERA, Pieter Willem.

Relação dos organizadores, por ordem alfabética, e suas respectivas Instituições

Maria das Graças Bruno Marietto
Universidade Federal do ABC

Mário Minami
Universidade Federal do ABC

Pieter Willem Westera
Universidade Federal do ABC

Relação de autores, por ordem alfabética, e suas respectivas Instituições

Alessandro S. Nascimento
Universidade de São Paulo

Aline Neves
Universidade Federal do ABC

Carlos da Silva dos Santos
Universidade Federal do ABC

Cristiane Otero Reis Salum
Universidade Federal do ABC

Delmo Alves de Moura
Universidade Federal do ABC

Edson Pinheiro Pimentel
Universidade Federal do ABC

Harlen Costa Batagelo
Universidade Federal do ABC

Humberto Luiz Razente
Universidade Federal de Uberlândia

Irineu Antunes Junior
Universidade Federal do ABC

João Paulo Gois
Universidade Federal do ABC

Juliana Cristina Braga
Universidade Federal do ABC

Letícia Rodrigues Bueno
Universidade Federal do ABC

Luiz Carlos da Silva Rozante
Universidade Federal do ABC

Maria Camila Nardini Barioni
Universidade Federal de Uberlândia

Maria das Graças Bruno Marietto
Universidade Federal do ABC

Márcio Eisencraft
Universidade Federal do ABC

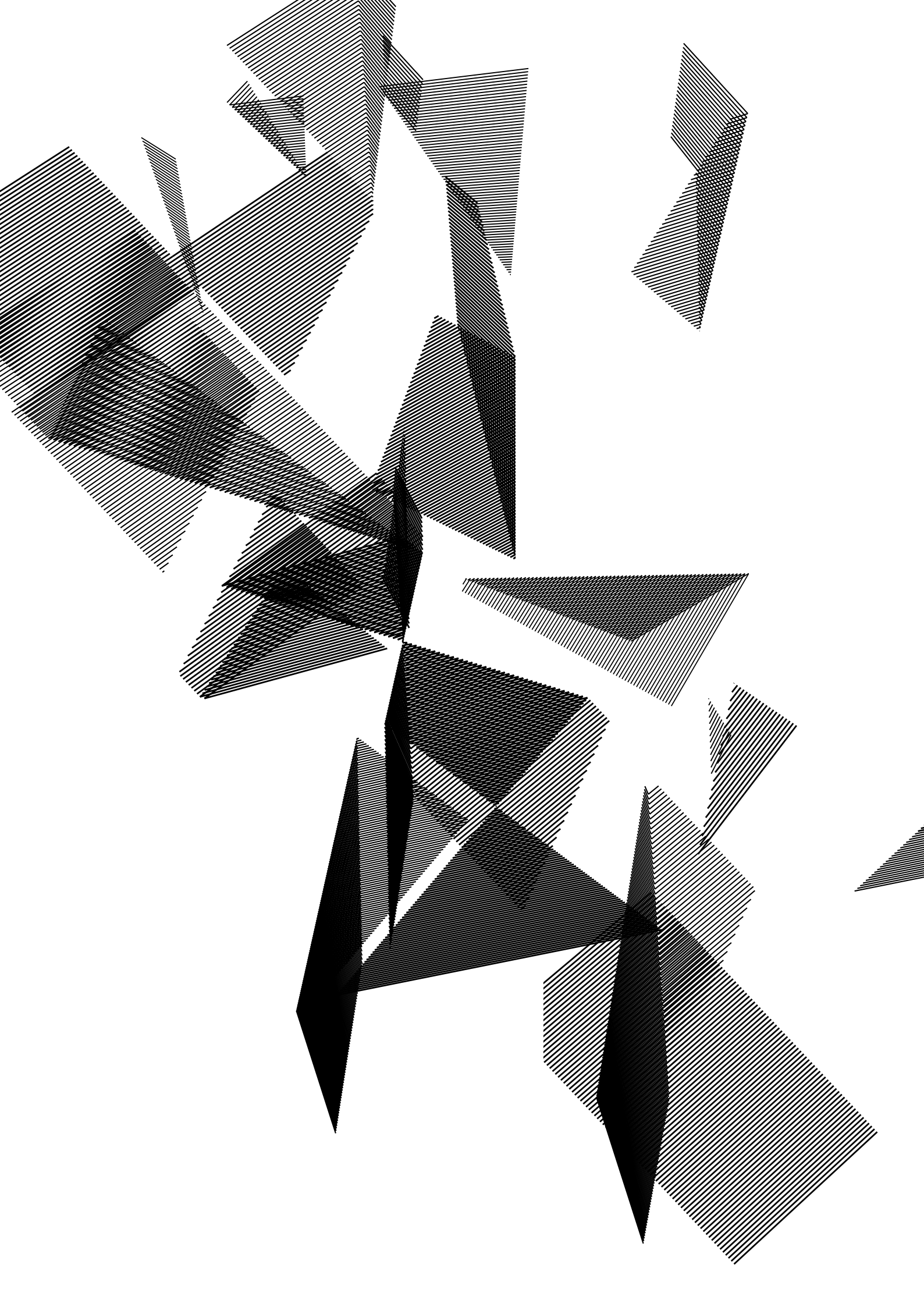
Márcio Kaitsumi Oikawa
Universidade Federal do ABC

Peter Maurice Erna Claessens
Universidade Federal do ABC

Ricardo Suyama
Universidade Federal do ABC

Ronaldo Cristiano Prati
Universidade Federal do ABC

Wagner Tanaka Botelho
Universidade Federal do ABC



SUMÁRIO

7 Prefácio

11 Capitulo 1 Fundamentos da computação

41 Capitulo 2 Representação gráfica de funções

63 Capitulo 3 Noções de estatística, correlação e regressão

113 Capitulo 4 Base de dados

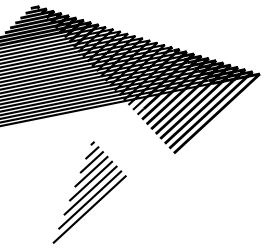
143 Capitulo 5 Lógica de programação: Variáveis e estruturas sequenciais

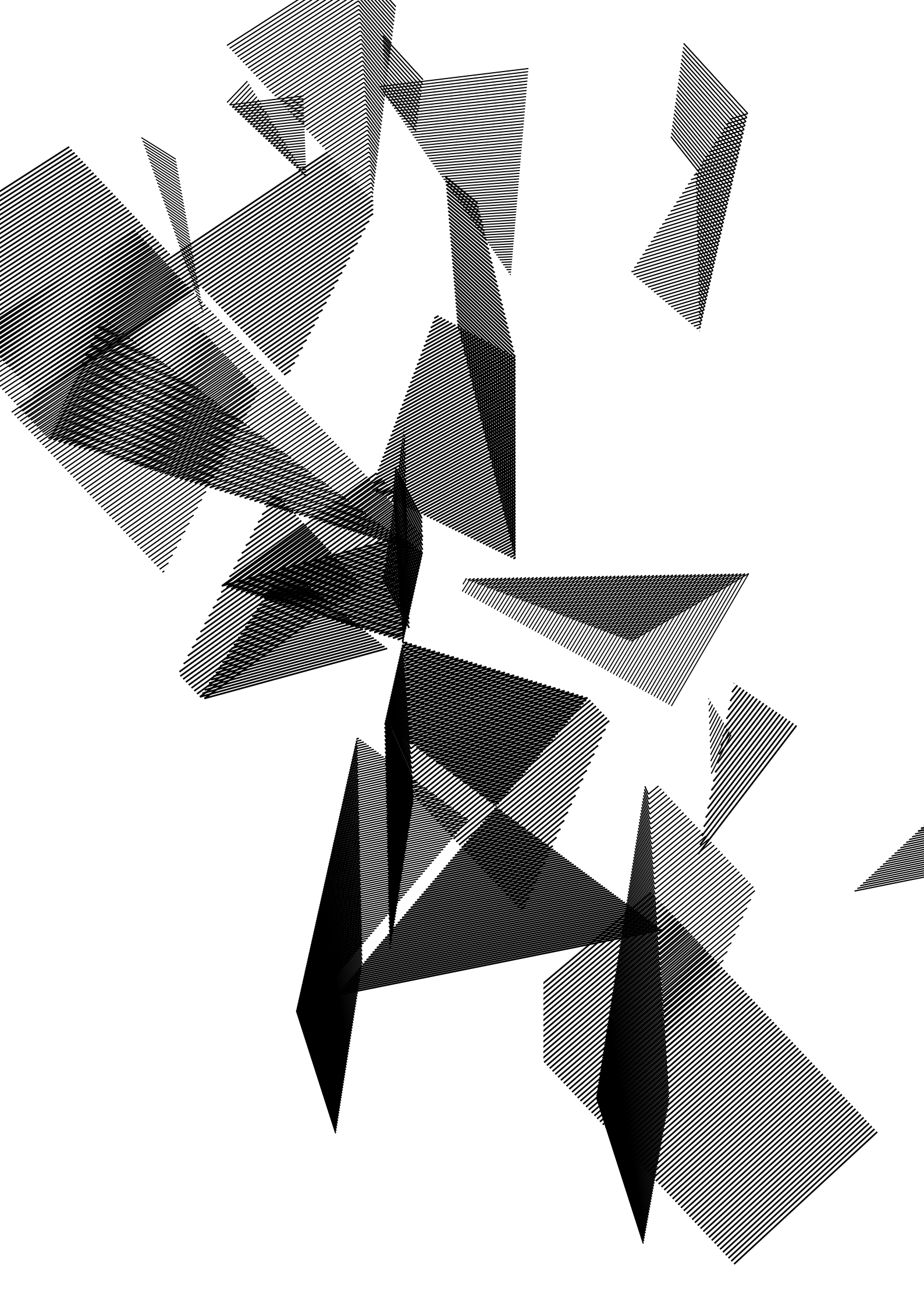
161 Capitulo 6 Lógica de programação: Estruturas condicionais

175 Capitulo 7 Lógica de programação: Estruturas de repetição

185 Capitulo 8 Modelagem e simulação computacional: Conceitos fundamentais

218 Capitulo 9 Modelagem e simulação computacional: A ciência na prática



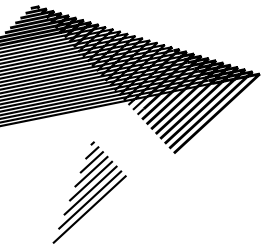


PREFÁCIO

Atualmente, utilizamos a tecnologia da informação em quase todos os domínios da vida. No ambiente acadêmico, por exemplo, o trabalho com computadores se faz necessário em áreas tais como Ciências Naturais, Matemática, Computação, Biologia, Química, Física, Engenharias, Licenciaturas, Ciências Sociais e Artes. Por esse motivo, o uso de computadores e ferramentas computacionais deve fazer parte do conhecimento de alunos no nível universitário desde o início dos seus estudos. Para esse fim, na UFABC, há a oferta da disciplina *Bases Computacionais da Ciência*.

Esta disciplina faz parte de um conjunto de seis disciplinas "de base" do currículo da UFABC. As outras disciplinas deste conjunto são *Bases Matemáticas*, *Base Experimental das Ciências Naturais, Estrutura da Matéria, Origem da Vida e Diversidade dos Seres Vivos* e *Bases Epistemológicas da Ciência Moderna*. A maior parte destas disciplinas é cursada pelos alunos dos dois bacharelados interdisciplinares da UFABC: *Bacharelado em Ciência e Tecnologia* (BC&T) e *Bacharelado em Ciências e Humanidades* (BC&H). O propósito é criar uma base homogênea de conhecimento para os alunos de todas as áreas de graduação oferecidas pela UFABC. Além de serem importantes para o andamento dos estudos seguintes, as disciplinas de base representam o primeiro contato do aluno com a interdisciplinaridade, ponto central do projeto pedagógico da UFABC. A interdisciplinaridade visa à percepção do aluno de que as áreas da ciência são interligadas, sendo que, para o bom entendimento de um dado assunto, o conhecimento das áreas relacionadas a ele é indispensável.

O objetivo deste livro é dar apoio à disciplina *Bases Computacionais da Ciência*, apresentando os seus conceitos teóricos elementares juntamente com um grande número de exercícios, já que o uso do computador se aprende principalmente através da prática. Esperamos que este livro seja uma ferramenta didática importante para os alunos da disciplina e outras pessoas que gostariam de aprender os conceitos básicos da área de Computação. Desejamos um aprendizado frutífero e prazeroso.



ESTRUTURA DO LIVRO

A disciplina consiste, além das provas e do projeto final, de nove aulas, cada uma das quais se encontra representada nos capítulos deste livro. Em cada capítulo há uma introdução sobre o conteúdo da aula, enunciando-se seus objetivos. Na sequência, há uma seção que apresenta a parte teórica da aula, ilustrada com exemplos práticos. Na sequência, a Seção "Atividades em aula" oferece exercícios práticos a serem realizados em sala de aula, sob a orientação do professor. Na Seção "Considerações finais" são apresentadas as observações finais, relacionadas aos conteúdos apresentados. Por fim, a Seção "Exercícios" apresenta alguns exercícios para assimilação do conteúdo estudado.

RECURSOS COMPUTACIONAIS DA DISCIPLINA

Além deste texto, outro recurso educacional importante é o emprego de um computador pessoal, preferencialmente com acesso à Internet. Na Internet, encontra-se o ambiente virtual de aprendizagem TIDIA, disponível em http://tidia-ae.ufabc.edu.br/portal, o qual é utilizado nesta disciplina para atividades tais como: disponibilização do material didático, transferência de arquivos, comunicação entre os alunos e entre aluno e professor, entrega de exercícios e fóruns interativos. Dentre os materiais didáticos disponíveis no TIDIA, citamos a versão em PDF deste livro, bem como tutoriais dos *softwares* usados nos exercícios práticos.

Para fazer os exercícios práticos é necessário instalar no computador os seguintes *softwares*:

- Um navegador de Internet, para acessar o TIDIA e fazer pesquisas online;
- Scilab, um *software* para computação numérica e a visualização de funções matemáticas;
- BrOffice Calc, um programa de cálculos de planilha, que faz parte do pacote OpenOffice (ou BrOffice);
- RoboMind, um programa para o ensino de Lógica de Programação.
- Os primeiros três *softwares* são gratuitos e podem ser baixados, instalados e utilizados para fins não-comerciais sem pagamento de licenças de uso. O último, RoboMind, é licenciado para o uso em universidades na versão mais recente. A UFABC está em processo de adquirir uma licença. Para o uso em casa, a versão atual do *software* RoboMind é gratuita e pode ser baixada, instalada e utilizada sem pagamento de licenças de uso.

AGRADECIMENTOS

A elaboração deste livro só foi possível graças ao apoio, confiança, dedicação e colaboração de muitas pessoas que ajudaram a torná-lo uma realidade. Por isso, fazemos questão de registrar aqui nossos agradecimentos.

Em primeiro lugar, agradecemos a quem acreditou no projeto quando este era apenas uma ideia. Agradecemos a Pró-Reitoria de Graduação, em especial a Professora Denise Consonni e os Professores Dácio Roberto Matheus, José Fernando Queiruga Rey e Derval dos Santos Rosa, pela iniciativa em estruturar uma discussão integrada em toda a UFABC sobre as seis disciplinas de base dos bacharelados interdisciplinares.

De maneira especial, gostaríamos de agradecer nominalmente os professores que elaboraram este livro, sempre de maneira cooperativa e interativa. São eles: Alessandro S. Nascimento, Aline Neves, Carlos da Silva dos Santos, Cristiane Otero Reis Salum, Delmo Alves de Moura, Edson Pinheiro Pimentel, Harlen Costa Batagelo, Humberto Luiz Razente, Irineu Antunes Junior, João Paulo Gois, Juliana Cristina Braga, Letícia Rodrigues Bueno, Luiz Carlos da Silva Rozante, Maria Camila Nardini Barioni, Maria das

Graças Bruno Marietto, Márcio Eisencraft, Márcio Kaitsumi Oikawa, Peter Claessens, Ricardo Suyama, Ronaldo Cristiano Prati e Wagner Tanaka Botelho.

A contribuição de cada um destes professores demonstra o domínio científico e técnico nos temas abordados em cada capítulo. Entretanto, a preocupação de todos foi muito além da transmissão pura e direta dos conhecimentos. O processo de formação acadêmica dos alunos que utilizarão este material didático sempre foi o norte de nossas discussões. Sendo assim, as decisões relacionadas à forma de transmissão dos conhecimentos foram direcionadas por perguntas tais como: (i) como apresentar conhecimentos amplos e complexos, de maneira que os alunos iniciantes entendam seus significados e abrangência? (ii) como apresentar os conhecimentos de Computação não como um fim em si mesmo, mas como uma importante e indispensável ferramenta para todos os profissionais do século XXI? (iii) como expor os alunos, de maneira gradativa e coerente, às várias situações de estudo e pesquisa envolvidas na vida acadêmica?

A elaboração de um material didático o mais adequado possível para apoio às aulas é um grande desafio, mas também uma boa motivação para todos os docentes e para toda a comunidade acadêmica. Esperamos ter colaborado nesse processo de construção e que essas iniciativas colham prontamente bons frutos e produzam novas sementes.

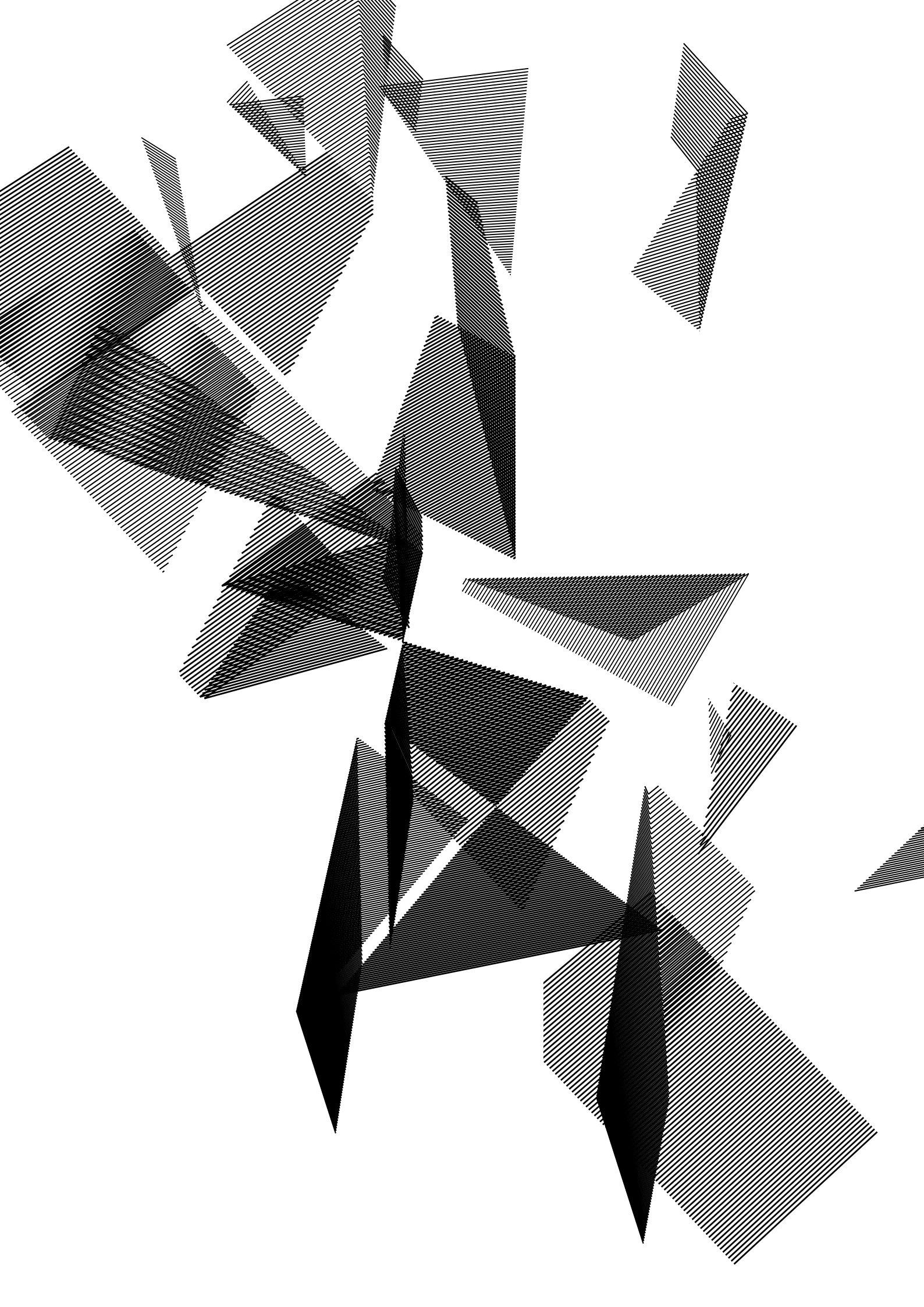
Maria das Graças Bruno Marietto

Mário Minami

Pieter Willem Westera

Comissão da Disciplina Bases Computacionais da Ciência

Santo André, São Paulo, Julho de 2013



CAPÍTULO 1

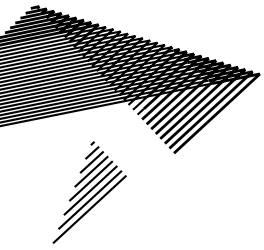
FUNDAMENTOS DA COMPUTAÇÃO

Edson Pinheiro Pimentel
Juliana Cristina Braga

1.1 INTRODUÇÃO

Atualmente, é praticamente impensável fazer pesquisa científica sem o uso de computação. A presença de instrumentos computadorizados coletando dados o tempo todo e em todo lugar gera dados científicos em volumes que não podem mais ser entendidos com cálculos simples, sendo necessárias, muitas vezes, computações complexas. Por exemplo, a Sloan Digital Sky Survey está mapeando o céu com um telescópio e nos primeiros cinco anos de operação, gerou cerca de 6 TB de dados. Da mesma forma, simulações em grande escala de modelos climáticos a reatores de fusão geram enormes conjuntos de dados em semanas ou mesmo dias devido à disponibilidade de computadores cada vez mais rápidos. Tal volume de dados tem de ser analisado por técnicas de computação. À luz desta evolução da ciência, as futuras gerações de cientistas devem compreender que precisarão envolver cada vez mais computação em seu trabalho (Hambrusch *et al.*, 2009:183).

De fato, diversas áreas de pesquisa estão se tornando cada vez mais dependentes da computação. Segundo George Johnson, no artigo "All Science is Computer Science", publicado no *New York Times on the Web* em 2001, toda ciência, ao que parece, está se tornando ciência da computação. Eis alguns depoimentos que embasam esta afirmação:



- "Física é quase inteiramente computacional agora", disse Thomas B. Kepler, vice-presidente para assuntos acadêmicos do Santa Fe Institute, um centro de pesquisa multidisciplinar, no Novo México, EUA. "Ninguém sonharia em fazer esses grandes experimentos do acelerador sem uma tremenda quantidade de poder computacional para analisar os dados".

- Mas a maior mudança, segundo ele, foi em biologia. "Dez anos atrás, os biólogos desconsideravam a necessidade de computação", disse o Dr. Kepler. "Agora eles estão cientes de que não podem realmente fazer biologia sem ele".

- Há química computacional, neurociência computacional, genética computacional, imunologia computacional e biologia molecular computacional. Também, áreas como a sociologia e a antropologia estão lentamente sucumbindo à mudança. No Instituto Santa Fé, modelos de computador são usados para estudar os fatores que podem ter levado à ascensão e queda de culturas complexas, uma espécie de arqueologia artificial (Johnson, 2001:1).

O "Pensamento Computacional" ou "Raciocínio Computacional", do inglês *Computational Thinking,* tem sido considerado essencial para o aprendizado de técnicas como decomposição de tarefas ou modelagem de problemas, necessárias em diversas tarefas nas mais diversas áreas. De forma simplista seria como aprender a "pensar" como um computador. Segundo Henderson e colaboradores (2007:195) o "Pensamento Computacional" é o núcleo de todas as disciplinas modernas nas áreas de Ciências, Tecnologia, Engenharia e Matemática e é intrínseco a todas as outras disciplinas, de A a Z. É utilizado na vida cotidiana desde ao se fazer um bolo, ao se trocar um pneu ou quando escovamos nossos dentes. O cérebro humano está preparado para pensar computacionalmente, assim como estão os dispositivos de computação moderna. De certo modo, precisamos apenas despertar o raciocínio computacional para melhor aplicá-lo, quando e onde precisarmos.

Pensamento Computacional é um processo de resolução de problemas que inclui as seguintes características, mas que não se limita a elas:

- Formulação de problemas de modo que permita fazer uso do computador e outras ferramentas para ajudar a resolvê-los;
- Representação de dados através da abstração, tais como modelagem e simulação;
- Automatização de soluções através do pensamento algorítmico (uma sequência de instruções);
- Generalização e transferência do processo de solução de um problema para uma grande variedade de problemas relacionados (Wing, 2006:33).

Assim, seja qual for o campo de atuação escolhido, será inevitável estudar e entender um pouco de computação ou informática. De fato, esse "pouco" deve ir além de uma "alfabetização digital básica", e incluir a compreensão de termos como *software* ou *hardware* e também saber ligar um computador e navegar pela rede mundial de computadores.

A organização americana CSTA – *Computer Science Teachers Association* (Associação de professores da Ciência da Computação) defende que o ensino da computação, como ciência, deve começar já no ensino médio. O documento "*K–12 Computer Science Standards*", que estabelece padrões para a educação em Ciência da Computação no ensino médio americano apregoa que:

> Para ser cidadãos bem-educados em um mundo com utilização intensiva de computação e para ser preparado para as carreiras no século 21, os nossos estudantes devem ter uma compreensão clara dos princípios e práticas da ciência da computação. Nenhum outro assunto vai abrir tantas portas no século 21 como a ciência da computação, independentemente da área fim de estudo ou ocupação do estudante (Seehorn et al., 2011:ii).

No entanto, o ideal é que esse processo educativo seja realizado não apenas com uma visão tecnicista, mas também seja amparado por diversas abordagens disciplinares e científicas.

Dentro desse espírito, o livro *Computer Science Unplugged* (Bell, 2011), traduzido para diversas línguas, inclusive o Português, dispõe de uma coleção de atividades desen-

volvidas com o objetivo de ensinar os fundamentos da ciência da computação sem a necessidade de computadores. A concepção desse livro foi motivada pela necessidade de implementação de métodos lúdicos e simples que dispensem a utilização de computadores na sala de aula. Para Bell, "[...] as "atividades desplugadas" são passíveis de aplicação em localidades remotas com acesso precário de infraestrutura (i.e., sem energia elétrica ou computadores disponíveis) e podem até ser ministradas por não especialistas em computação" (2011:i).

A palavra Computação deriva do latim *computare*, que significa contar e que existia muito antes do primeiro computador. O termo Informática provém da contração das palavras "*Information automatique*" (Informação automática) e foi criado em 1962 na França, portanto após o primeiro computador.

De acordo com Lancharro e colaboradores, "A informática nasceu da ideia de auxiliar o homem nos trabalhos rotineiros e repetitivos, em geral de cálculo e gerenciamento" (2004:1). Uma das definições mais comumente aceitas: "Informática é a ciência que estuda o tratamento automático e racional da informação". Entre as principais funções da informática destacam-se: a) desenvolvimento de novas máquinas; b) desenvolvimento de novos métodos de trabalho; c) construção de aplicações automáticas; d) melhoria de métodos e aplicações existentes.

Nesse sentido, este capítulo abordará alguns conceitos introdutórios relacionados aos fundamentos da computação de modo que o estudante seja capaz de:

- Reconhecer nas máquinas computadorizadas os elementos essenciais que as tornam capazes de processar dados;
- Perceber a evolução histórica dos computadores;
- Identificar os Sistemas Computacionais e seus componentes;
- Distinguir as diversas áreas que compõem a Ciência da Computação.

O capítulo está organizado da seguinte forma. A Seção 1.2 apresenta o computador como uma máquina capaz de processar dados e gerar informações. A Seção 1.3, numa visão do computador como um sistema, detalha os componentes de *hardware* e *software* e a interação entre eles. A Seção 1.4 apresenta um breve resumo da evolução histórica dos computadores e da própria computação. A Seção 1.5 descreve a organização da Ciência da Computação em 14 áreas do conhecimento a partir do Currículo de Referência da ACM *(Association for Computer Machinery)* e demonstra, através de exemplos, o potencial de aplicação da área da Ciência da Computação em outras áreas do conhecimento. Na Seção 1.6 o aluno aprende a utilizar o Ambiente Virtual de Aprendizagem denominado TIDIA-AE. A Seção 1.7 apresenta as considerações finais do capítulo, enquanto exercícios para fixação dos conceitos discutidos neste capítulo são fornecidos na Seção 1.8.

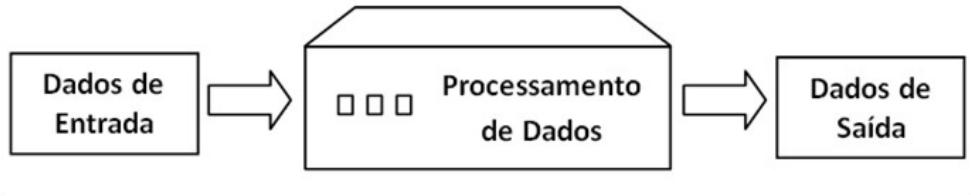
1.2 O COMPUTADOR E SEUS ELEMENTOS BÁSICOS

Do ponto de vista da informática, o dispositivo utilizado para o processamento de dados e a obtenção de informação é o computador. O computador é uma máquina composta de elementos físicos do tipo eletrônico, capaz de realizar uma grande va-

riedade de operações com alta velocidade e precisão, a partir de instruções adequadas (Lancharro *et al.,* 2004:2).

A Figura 1.1 apresenta um esquema básico representando o computador, elemento central, como um dispositivo que transforma dados de entrada em dados de saída, ou seja, que age como um "processador de dados".

Figura 1.1: O computador transformando dados de entrada em saída.

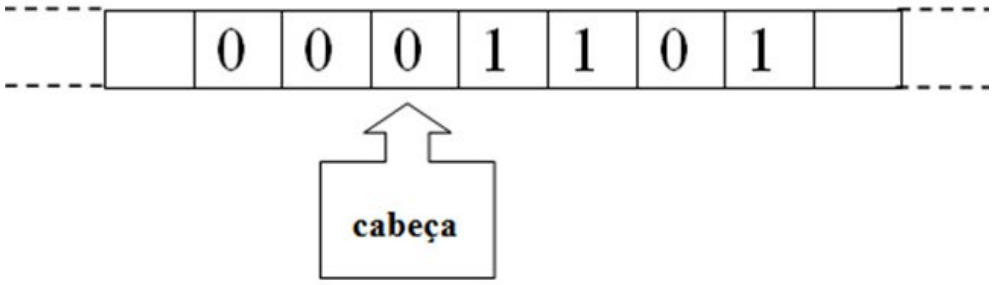


O fluxo de processamento de dados da Figura 1.1 é didático, na medida em que apresenta os elementos principais do funcionamento de uma máquina que faz cálculos como, por exemplo, um computador. Entretanto, tal fluxo é muito genérico, pois não diferencia um computador de máquinas que realizam apenas um conjunto restrito de funcionalidades como, por exemplo, uma máquina de calcular, ou o sistema de controle de uma máquina de lavar roupas. Sabemos, entretanto, que os computadores atuais são máquinas de propósito geral, pois estão preparados para realizar uma variedade de tarefas, dentro de um amplo escopo de complexidade.

A fim de simular procedimentos mais gerais, aproximando-se da concepção de um dispositivo de computação de propósito geral, o matemático inglês Alan M. Turing propôs em 1936 um tipo de máquina abstrata, um modelo matemático teórico para o computador universal, chamada de Máquina de Turing. Ela se tornou um dos principais elementos para amparar o conceito da computabilidade, ou seja, se algo é computável num tempo finito. Alan M. Turing (1912 – 1954) é considerado um dos pais da Computação.

Basicamente, o dispositivo lógico que Turing chamou de *automatic machine* (máquina automática) era capaz de ler, escrever e apagar símbolos binários (zeros e uns) em uma fita de comprimento ilimitado, dividida por quadrados de igual tamanho. Uma cabeça de leitura/gravação se moveria em qualquer direção ao longo da fita, um quadrado por vez, e uma unidade de controle poderia interpretar uma lista de instruções simples, movendo-se para a direita ou para a esquerda (Gersting, 2004:403). A Figura 1.2 ilustra o dispositivo.

Figura 1.2: Modelo conceitual da Máquina de Turing.



O que torna uma máquina de Turing capaz de executar uma tarefa é o conjunto de regras de transição que compõem o programa da máquina (o seu objetivo) e um determinado estado inicial (Tenório, 1991:47). Possíveis instruções que poderiam compor um conjunto de regras seriam: a) Imprima 0 no quadrado, se leu 1, e mova-se para a direita; b) Mantenha 1 na fita, se leu 1, e pare.

Basicamente, o Modelo de Turing acrescenta um elemento extra à estrutura da Figura 1.1, o programa, conforme pode ser observado na Figura 1.3.

Figura 1.3: Um computador baseado no Modelo de Turing.

Denomina-se programa o conjunto de instruções dadas a um computador para a realização de determinado processo. Isso equivale, por exemplo, às regras de transição da Máquina de Turing. Ao conjunto de um ou vários programas que atuam de forma cooperada para realizar um determinado trabalho dá-se o nome de Aplicação ou Sistema. Os dados processados (através de cálculos, comparações, ordenações, etc.) transformam-se em informação útil, que apoiará a tomada de decisão a partir de algo que antes não se conhecia, ou seja, que não estava explícito nos dados brutos.

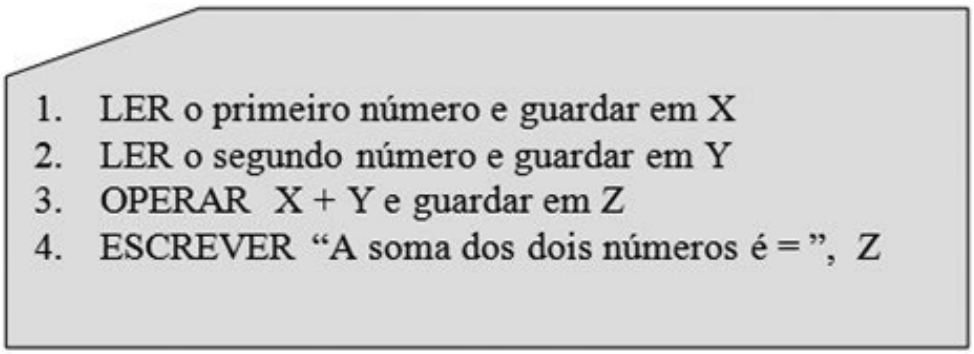
O programa capaz de ser processado pelo computador deve estar numa linguagem de programação. Existem diversas linguagens de programação, compreensíveis para o computador, tais como: C, C++, Java e Pascal. No começo da era dos computadores, as únicas linguagens de programação disponíveis eram as linguagens de máquina

que eram compostas de sequências de 0s e 1s (zeros e uns). Posteriormente, foram criadas linguagens de alto nível, mais próximas da linguagem humana (Forouzan e Mosharraf, 2011:215).

Outro termo bastante utilizado como sinônimo de programa é algoritmo, que é o conjunto de operações necessárias para transformar os dados iniciais no resultado desejado. Costuma-se adotar o termo algoritmo para a lógica da resolução do problema, ainda numa linguagem abstrata que não pode ser processada pelo computador.

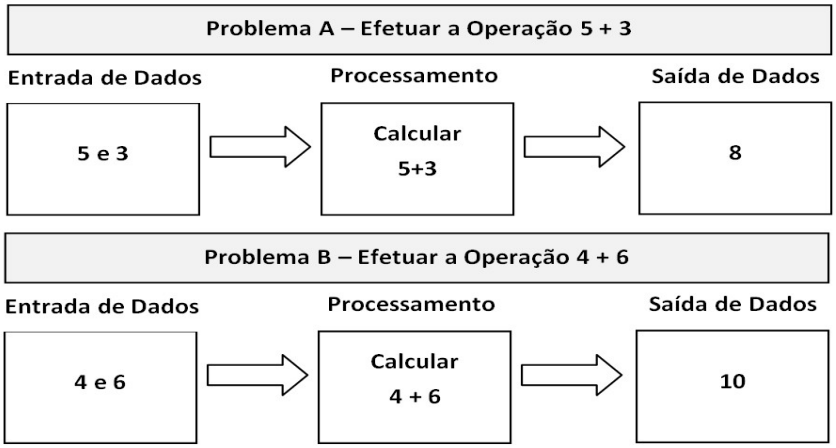
A título de ilustração a Figura 1.4 apresenta um algoritmo, escrito numa linguagem bem próxima do Português, que tem por finalidade exibir a soma de quaisquer dois números informados por um usuário.

Figura 1.4: Algoritmo para somar dois números.



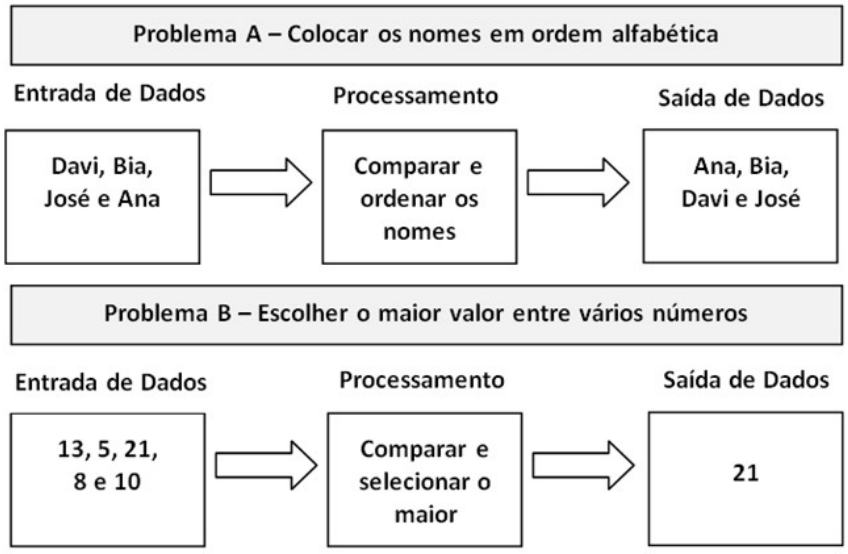
No modelo de Turing (veja Figura 1.3), os dados de saída dependem da combinação de dois fatores: os dados de entrada e o programa. No exemplo do algoritmo da Figura 1.4, se no passo 1 o usuário informar o valor 5 para `X` e no passo 2 informar o valor 3 para `Y`, o valor 8 será impresso no passo 4. Num outro processamento se forem informados respectivamente os valores 4 e 6 para `X` e `Y`, o resultado no passo 4, desta vez, será o 10, conforme ilustrado na Figura 1.5. Assim, o mesmo programa (algoritmo) produz resultados distintos para entradas de dados distintas. Por outro lado, se os dados forem os mesmos, o resultado também será o mesmo.

Figura 1.5: Exemplos de processamento de dados.



A Figura 1.6 mostra que problemas distintos requerem programas distintos. Cada programa efetua operações distintas sobre esses dados. Com base no problema A, o primeiro programa ordena alfabeticamente os nomes informados na entrada de dados. O segundo programa, orientado pelo problema B, localiza e seleciona o maior elemento do conjunto fornecido na entrada de dados.

Figura 1.6: Problemas distintos requerem programas distintos.



1.3 SISTEMAS COMPUTACIONAIS E SEUS COMPONENTES

Esta seção pretende abordar o computador como parte de um sistema computacional, que pode ser definido como um conjunto de componentes que realizam processamentos automáticos a partir de dados de entrada, e que fornecem uma saída contendo esses dados transformados. Existem vários tipos de sistemas computacionais. Os mais comuns são os computadores pessoais, dentre ele os computadores de mesa e os computadores portáteis.

Outro tipo de sistema computacional que tem se tornado cada dia mais comum são os sistemas computacionais embarcados. Esses contêm um sistema computacional embutido (Hansmann *et al*., 2003). Diferentemente dos computadores pessoais, que possuem um propósito geral, os sistemas embarcados realizam tarefas específicas como, por exemplo, falar ao telefone, emitir rotas, autenticar usuários, rastrear objetos, etc. Exemplos deles são: celulares, GPS (Global Position System), Smart Cards, Smart Labels.

O conjunto de componentes dos sistemas computacionais pode ser dividido em duas grandes categorias:

- *Hardware*; e
- *Software*.

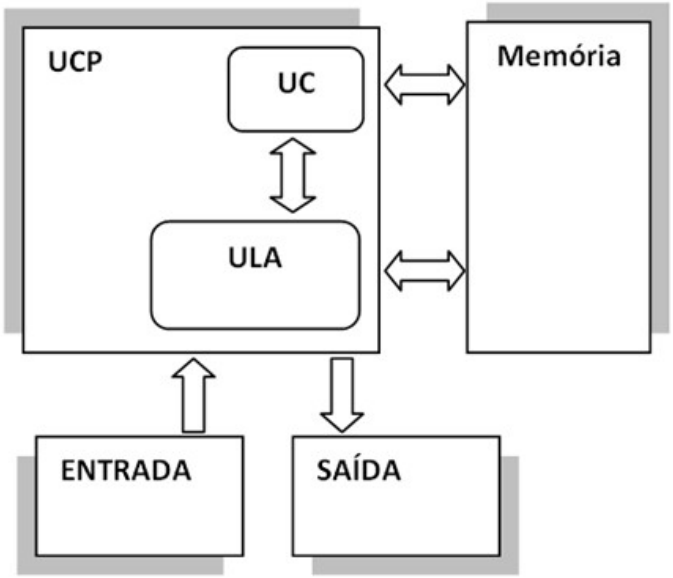
A seguir, tem-se uma descrição mais detalhada de cada uma dessas categorias.

1.3.1 *HARDWARE*

Os componentes da categoria *Hardware* são os elementos mecânicos e eletromecânicos dos sistemas computacionais e, de acordo com a arquitetura básica proposta em 1944 por John von Neumann, engenheiro e matemático húngaro, possuem quatro componentes, a saber: Unidade de Entrada e Saída de Dados; Memória; Unidade de Lógica e Aritmética (ULA); e Unidade de Controle (UC). A Unidade Central de Processamento (UCP) engloba a UC e a ULA.

A Figura 1.7 representa a arquitetura geral proposta por John von Neumann e seus componentes são detalhados a seguir.

Figura 1.7: Arquitetura geral baseada no Modelo de von Neumann.



Unidades de entrada e saída de dados

Os dispositivos de entrada são os *hardwares* responsáveis por capturar os dados de entrada do sistema. Exemplos: teclado, *mouse*, câmera de vídeo, tela de toque.
Os dispositivos de saída são os *hardwares* responsáveis por apresentar os dados transformados. Exemplos: monitores, impressoras, dispositivos de emissão de voz, etc.

Memória

Durante o processamento, dados – tanto de entrada quanto de saída – e programas encontram-se em uma área de armazenamento denominada memória. De acordo com a arquitetura de von Neumann é possível armazenar dados e programas no mesmo espaço de memória. Existem dois tipos principais de dispositivos de armazenamento (memória): memória principal e memória secundária.

A memória principal fornece aos dispositivos de processamento (ULA e UC) as informações necessárias para a transformação dos dados de entrada nos de saída. Nessa categoria encontra-se, por exemplo, a memória RAM – *Random Access Memory* (Memória de Acesso Aleatório). A memória RAM é volátil, pois seus dados são perdidos depois que o sistema é desligado. Há também a memória ROM – *Ready Only Memory* (Memória Somente de Leitura), que armazena geralmente um programa denominado BIOS (*Basic Input Output System*).

Os dispositivos de armazenamento secundário guardam dados que podem ser recuperados após o desligamento do sistema computacional. Exemplos desses dispositivos são: disco rígido (*Hard Disk* – HD) interno ou externo, pen drive, cartão de memória utilizado em câmeras digitais e celulares, entre outros. Esse tipo de memória geralmente é mais barato, no entanto, são dispositivos mais lentos que a memória RAM. A memória secundária também pode ser considerada como um dispositivo de entrada de dados, quando os dados são carregados dela para a memória RAM, assim como de saída de dados, quando são gravados nela.

Unidade de lógica e aritmética

A unidade de lógica e aritmética (ULA) é o local onde ocorrem as operações aritméticas (adição, subtração, divisão, multiplicação, dentre outras) e lógicas (comparação, classificação, dentre outras).

Unidade de controle

A Unidade de Controle (UC) é o centro nervoso de computador. Como o próprio nome diz, a UC é responsável por controlar as instruções realizadas pelo computador, bem como controlar os demais componentes de sua arquitetura. A UC tem a lógica e a estrutura necessárias para executar funções tais como controlar a entrada e saída de dados e controlar as ações da ULA. No modelo de von Neumann, a UC busca uma instrução da memória e a executa, após decodificá-la. Assim, cada uma das instruções é executada, uma após a outra. Existem computadores na atualidade que podem executar instruções simultaneamente, em paralelo.

Os componentes especificados, de acordo com a arquitetura von Neumann, podem ser agrupados em um único *hardware*. Normalmente a ULA e a UC são agrupadas, formando um componente de *hardware* denominado Unidade Central de Processamento (UCP), do inglês *Central Processing Unit* (CPU). Em alguns casos, outros componentes podem

ser agrupados à CPU, como dispositivos de entrada e saída (E/S) também conhecidos pelo termo em inglês *Input/Output* (I/O) e memória, tanto a principal quanto a secundária. Esse agrupamento é definido de acordo com a finalidade do projeto e a sua modularidade. Um exemplo de CPU que possui, além da UC e da ULA, também dispositivos de E/S e memória agrupados é o microcontrolador. A CPU está presente nos mais variados dispositivos com capacidade de processamento como é o caso, por exemplo, dos celulares e dos *smart cards*.

No caso dos computadores de mesa ou portáteis, todos os quatro componentes da arquitetura geral de von Neuman são integrados através de um *hardware* chamado de *placa mãe*. A Memória Princial (RAM e ROM) e a Unidade Central de Processamento são geralmente embutidos na placa mãe. Por outro lado, os dispositivos de entrada, saída de dados e os de armazenamento secundário são conectados à placa mãe através de cabos. Nos sistemas embarcados a integração entre os componentes varia de acordo com o sistema computacional.

1.3.2 *SOFTWARE*

Softwares são programas de computadores que permitem explorar os recursos dos *hardwares*, executar determinadas tarefas e resolver problemas de forma automática. É através do *software* que o ser humano interage com a máquina e torna operacional o sistema computacional.

Os três principais tipos de *softwares* são: i) *software* de sistema; ii) *software* de aplicação; e iii) *software* de serviço.

Software de sistema

O *software* de sistema permite interagir com os componentes de *hardware* do computador. São exemplos desses *softwares*: BIOS, *drivers*, Sistema Operacional.

O mais importante *software* de sistema é o Sistema Operacional (SO). O SO é responsável por gerenciar os recursos computacionais e fazer a comunicação (ou interface) entre os componentes de *hardware* e os aplicativos. Sem um sistema operacional, não é possível utilizar os *softwares* de aplicação instalados em um computador. No mercado, existem vários sistemas operacionais, a Tabela 1.1 mostra os mais comuns dentre eles.

Tabela 1.1:Exemplos de sistemas operacionais.

Sistema operacional	Sistema computacional
Windows	Computadores de mesa e portáteis
Linux	Computadores de mesa e portáteis
Mac OS	Computadores de mesa e portáteis
Windows Tablet Edition	Tablets
Google Andorid	Tablets, celulares
iOS	Tablets, celulares
Windows Embedded	Sistemas embarcados (GPS, eletrodomésticos, celulares, etc.)

Software de Aplicação

Os *softwares* de aplicação, ou aplicativos, são programas criados para resolver tarefas específicas como: acessar a internet, enviar e receber mensagens, navegar pelo computador, editar um texto, desenhar uma imagem, etc. A Tabela 1.2 mostra exemplos de alguns aplicativos e suas funcionalidades.

Tabela 1.2: Exemplos de aplicativos.

Aplicativo	Funcionalidade	Exemplos
Navegadores	Acessar sites na Internet	Internet Explorer, FireFox, Google Crome
Editores de texto	Editar documentos	Br Office Writer, WordPad, Microsoft Word
Planilhas eletrônicas	Realizar cálculos, plotar gráficos, analisar dados	Br Office Calc, Microsoft Excel
Processadores de imagens	Criar e editar imagens	Microsoft Paint, Adobe Photoshop

Existem também os aplicativos embarcados, que são aqueles destinados a funcionar dentro de um sistema computacional embarcado (celulares, GPS, eletrodomésticos, etc.). Exemplos desses *softwares* são: aplicativo para envio de mensagens de um celular para outro, aplicativo para emitir rotas em um GPS, aplicativos de autenticação dos smart cards.

Software de serviço

Os *softwares* de serviço, também chamados de aplicativos web, são aqueles que não precisam ser instalados em um sistema computacional, e que são utilizados diretamente na Internet através de um navegador. Exemplos: Google, Google Maps, Tidia-AE, Moodle, etc.

1.4 EVOLUÇÃO HISTÓRICA

A Computação é uma ciência nova e está associada a uma série de fatos e descobertas anteriores. Esta seção pretende destacar os principais elementos da história da computação e dos computadores. Forouzan e Mosharraf (2011:7) e Tanenbaum (2007:8) dividem essa história em três períodos: i) as máquinas mecânicas até 1930; ii) os computadores eletrônicos de 1930 a 1950; e iii) as cinco gerações dos computadores de 1950 até à época atual.

1.4.1 MÁQUINAS MECÂNICAS

Destacam-se nesse período cinco invenções de máquinas de computação que contribuíram de forma importante para a evolução dos computadores:

• Máquina de Calcular de Blaise Pascal: Em 1642, o matemático e filósofo francês, Blaise Pascal, inventou o primeiro dispositivo dotado da capacidade para processar dados. Esse aparelho, denominado de Pascaline, foi considerado a primeira máquina automática de calcular e foi utilizado durante esse período como calculadora mecânica para fazer operações de adição e subtração;

• Máquina de Calcular de Leibnitz: Em 1670, o matemático alemão Gottfried Leibnitz inventou outra calculadora mecânica, denominada Roda de Leibnitz. Essa calculadora era semelhante à máquina de Pascal, porém possuía não somente a capacidade de adicionar e subtrair, mas também a de multiplicar e dividir. Calculadoras mecânicas são consideradas precursoras dos computadores, mas não são computadores. Isto porque não possuíam memória nem podiam ser programadas;

• O Tear de Jacquard: Em 1804, Joseph-Marie Jacquard criou uma máquina que aplicou pela primeira vez o conceito de armazenamento e programação. O tear era automatizado e utilizava cartões perfurados para controlar as máquinas de tecelagem e executar operações previamente programadas;

• Máquina Analítica de Babbage: Em 1822, Charles Babbage projetou o que seria o primeiro modelo de computador. Em 1833, ele inventou uma máquina denominada Máquina das Diferenças, que podia fazer não apenas simples operações aritméticas, mas que também resolvia equações polinomiais. Nesse período, Babbage anteviu os componentes que até hoje são a base de funcionamento de um computador: i) alimentação de dados através de cartões perfurados; ii) uma unidade de memória, onde os números podiam ser armazenados e reutilizados; e iii) programação sequencial de operações. Devido a limitações tecnológicas da época, nenhuma das máquinas de Babbage chegou a ser construída totalmente neste período;

• Perfuradora de Cartões de Hollerith: Em 1890, Herman Hollerith projetou e construiu uma máquina programável que podia ler, registrar e ordenar dados armazenados em cartões perfurados. A perfuradora de Cartões de Hollerith era uma máquina que possuía o propósito específico de processar os dados do censo dos Estados Unidos da América (EUA). Somente a partir da década de 1930, tentativas sérias foram feitas para a construção de computadores de propósito geral.

1.4.2 COMPUTADORES ELETRÔNICOS

Entre 1930 e 1950, foram criados os primeiros computadores eletrônicos, dentre os quais se destacam:

• Colossus: Em 1943, Alan Turing coordenou a equipe inglesa que construiu o computador Colossus, que foi destinado a decifrar códigos de mensagens cifradas por uma máquina alemã (Enigma). O Colossus era eletrônico, mas não de propósito geral;

• Computador MARK I: Em 1944, o Prof. Aiken, da Universidade de Harvard, construiu MARK I, o primeiro computador moderno que utilizava relés eletro-mecânicos;

• Computador ENIAC: Em 1945, John Mauchly e J. Presper Eckert construíram o primeiro computador de propósito geral, que utilizava válvulas totalmente eletrônicas. Esse computador foi denominado ENIAC (*Electronic Numerical Integrator and Calculator*), media 30 metros de comprimento por 3 metros de altura e pesava 30 toneladas;

• Computador EDSAC: Em 1949, Eckert e Mauchley construíram o EDSAC, o primeiro computador que tinha um programa armazenado para resolver problemas.

1.4.3 GERAÇÕES DE COMPUTADORES

Os computadores construídos a partir de 1950 basearam-se no conceito de armazenamento de programas concebidos por John von Neumann. Os historiadores organizam esse período em cinco gerações, cada uma delas marcada por importantes transformações de *hardware* ou *software*.

• Primeira geração (1950 a 1959): Esta geração caracterizou-se por computadores que utilizavam válvulas a vácuo para controlar a passagem de corrente elétrica. As válvulas eram interligadas por fios conectados manualmente, queimavam com frequência, eram grandes e lentas. Devido ao alto custo de construção e manutenção dos computadores construídos à base de válvulas, eles eram acessíveis apenas a grandes empresas e centros de pesquisa;

• Segunda geração (1959 a 1965): Esta geração caracterizou-se por computadores que utilizavam transistores para controlar a passagem de corrente elétrica. Tecnologicamente falando, podemos dizer que os transistores são os sucessores das válvulas eletrônicas. Transistores são componentes eletrônicos construídos com materiais semicondutores e funcionam como uma chave eletrônica (liga/desliga) ou como um amplificador de corrente. Quando utilizados na construção de computadores, proporcionaram um menor consumo de energia, menos aquecimento, maior velocidade de processamento, dentre outras melhorias de performance;

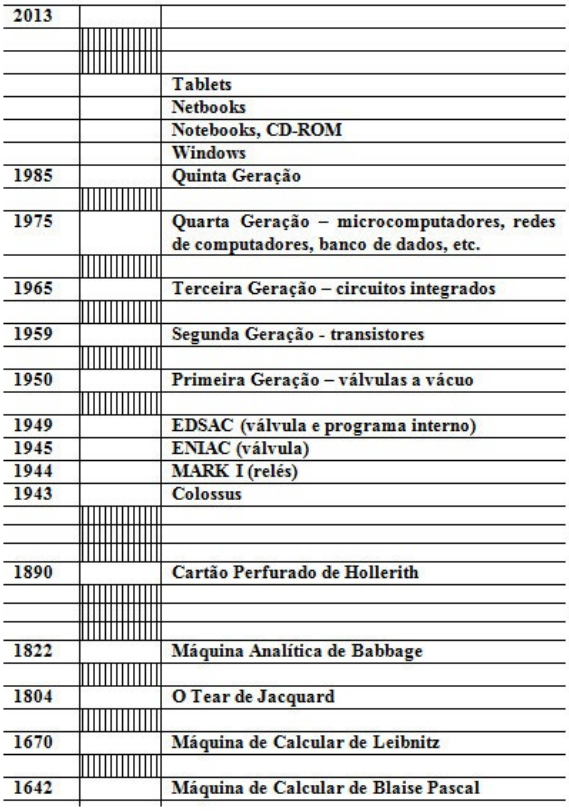
• Terceira geração (1965 a 1975): Prosseguindo na evolução tecnológica, os computadores desta geração utilizaram circuitos integrados em sua construção. Um circuito integrado é um componente eletrônico que integra elementos tais como transistores, resistores, diodos, etc., em um único chip (pastilha). A palavra chave desta tecnologia é a miniaturização, o que possibilitou, dentre outras inovações, o surgimento dos minicomputadores.

• Quarta geração (1975 a 1985): Surgimento dos microcomputadores, das redes de computadores, de banco de dados, computação distribuída, etc. (Forouzan e Mosharraf, 2011);

• Quinta geração (a partir de 1985): Surgimento dos computadores laptop e palmtop, aperfeiçoamento nos meios de armazenamento secundário (CD-ROM, DVD e assim por diante), o uso da multimídia, etc. (Forouzan e Mosharraf, 2011).

A Figura 1.8 apresenta um esquema que resume a evolução histórica dos computadores.

Figura 1.8: A Evolução histórica dos computadores (*hardware* e *software*).



1.5 ORGANIZAÇÃO DA CIÊNCIA DA COMPUTAÇÃO

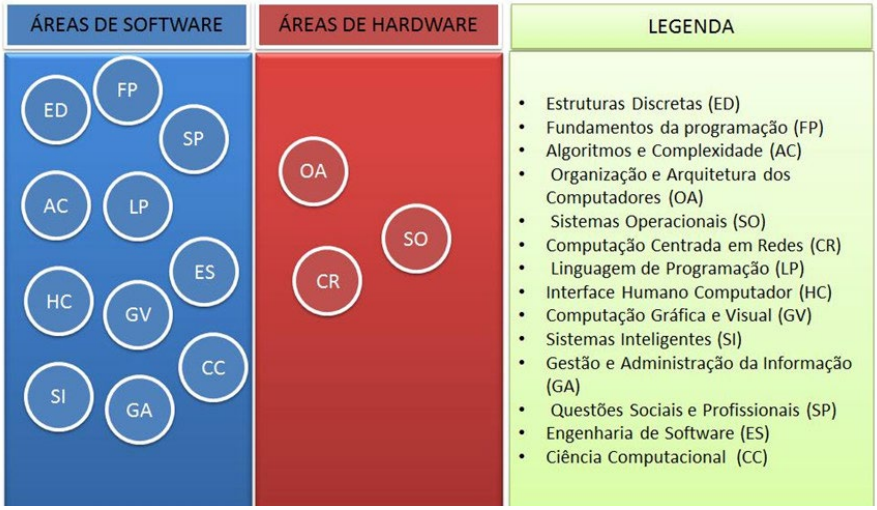
De acordo com o currículo de referência da ACM – *Association for Computer Machinery* (Cassel et al., 2008:15), a computação pode ser dividida em quatorze áreas:

1. Estruturas Discretas (ED);
2. Fundamentos da Programação (FP);
3. Algoritmos e Complexidade (AC);
4. Organização e Arquitetura dos Computadores (OA);
5. Sistemas Operacionais (SO);
6. Computação Centrada em Redes (CR);
7. Linguagem de Programação (LP);

8. Interface Humano-Computador (IH)
9. Computação Gráfica e Visual (GV);
10. Sistemas Inteligentes (SI);
11. Gestão e Administração da Informação (GI);
12. Questões Sociais e Profissionais (SP);
13. Engenharia de *Software* (ES);
14. Ciência Computacional (CC).

Nessa seção, essas quatorze áreas foram agrupadas em dois grupos: Grupo das áreas de *Softwares* e Grupo das áreas de *Hardware*. A Figura 1.9 demonstra claramente essa divisão.

Figura 1.9: Divisão das Áreas em Ciência da Computação.



Observa-se na Figura 1.9 que a Ciência da Computação enfatiza as áreas pertencentes ao grupo de *software*. Esse grupo preocupa-se basicamente com a produção de diferentes tipos de *softwares* como, por exemplo: sistemas operacionais, aplicativos convencionais, aplicativos para sistemas embarcados e aplicações web (Ver a Tabela 1.2 para recordar exemplos desses aplicativos).

Por possuir muitas áreas, o grupo de *software* foi incluído nesse documento em seis subáreas: Computação Científica, Interface Gráfica, Desenvolvimento de Sistemas, Teoria da Computação, Sistemas Inteligentes, Gestão e Administração da Informação e Questões Sociais e Profissionais.

A Figura 1.10 mostra a subdivisão do grupo de áreas de *softwares*.

Figura 1.10: Áreas de *Software* da Ciência da Computação.

1.5.1 ÁREAS DE *SOFTWARE*

Computação Científica

A área de Computação Científica reúne conceitos fundamentais da Ciência da Computação e possui forte conexão com a Matemática Discreta. Alguns conceitos dessa área de conhecimento são: funções, teoria dos conjuntos, lógica, teoria dos grafos e probabilidade discreta, análise de algoritmos, criptografia, algoritmos paralelos, modelagem e simulação e pesquisa operacional.

Exemplos de suas aplicações são:

- Simulação de reprodução de uma cultura de bactéria baseada no jogo da vida: http://code.google.com/p/lazbacterias/
- Simulador neuromuscular: http://remoto.leb.usp.br/remoto/index.html

Desenvolvimento de Sistemas

O desenvolvimento de sistemas reúne conceitos relacionados a algoritmos, implementação de sistemas computacionais e processos de desenvolvimento. Alguns conceitos dessa área de conhecimento são: algoritmos, estruturas de dados, recursividade, programação orientada a objetos, fundamentos e segurança da informação, paradigmas de programação e engenharia de *software*.

Exemplos de suas aplicações são:

- JMOL – Química (estruturas moleculares): http://jmol.sourceforge.net/
- Aplicações de Cognição (neuroimagens, neurofisiologia);
- Aplicações em Física (dinâmica de fluidos);
- Aplicações para celulares;
- Aplicações para GPS;
- Aplicações para TV Digital;
- Aplicações Internet.

Gestão e Administração da Informação

Essa área fornece o entendimento de como armazenar, organizar e buscar os dados em sistemas computacionais. Alguns conceitos abordados nessa área de conhecimento são: sistema de banco de dados, modelagem de dados, mineração de dados, hipermídia, arquitetura da informação, bibliotecas digitais.

Exemplo de aplicações são:

- Um supermercado pode analisar os dados de consumo de seus clientes para identificar quais novos produtos devem ser oferecidos a cada um deles.

Interface Gráfica

A área de Interface Gráfica reúne conceitos para o desenvolvimento de sistemas gráficos, realidade virtual, processamento de imagens, design, avaliação de usabilidade e acessibilidade. Alguns conceitos abordados nessa área de conhecimento são: interface humano-computador, processamento de imagens, animação por computador, realidade virtual, computação gráfica.

Exemplos de suas aplicações são:

- *Software* para leituras em voz de *websites*: http://webanywhere.cs.washington.edu/
- Visualização das artérias coronárias epicárdicas em contraste de microbolhas 3D em imagens ecográficas para auxiliar no diagnóstico.

Sistemas Inteligentes

Essa área apresenta técnicas para desenvolver sistemas que reproduzem a capacidade racional do ser humano para resolver problemas. Alguns conceitos abordados nessa área de conhecimento são: representação do conhecimento, aprendizagem de máquina, robótica, agentes inteligentes.

Exemplos de suas aplicações são:

- *Software* que joga xadrez;
- Futebol de robôs;
- Busca inteligente na web.

Questões sociais e profissionais (SP)

Essa área fornece conhecimento de como um profissional de Ciência da Computação deverá comportar-se eticamente no mercado, aborda questões relacionadas com crimes virtuais e questões sociais. Alguns conceitos abordados nessa área de conhecimento: histórica da computação, ética, criminalidade na computação.

Exemplos de suas aplicações são:

- Computação Verde;
- Direito de propriedade de *softwares* e materiais disponíveis na Internet;
- Computação forense.

1.5.2 ÁREAS DE *HARDWARE*

As áreas pertencentes ao grupo de *hardware* preocupam-se com o entendimento dos componentes dos sistemas computacionais, suas características, desempenho e interações. Também dão ênfase à infraestrutura, conexão e segurança de redes (Internet, redes sem fio, conexões, etc.). Alguns conceitos abordados nessa área de conhecimento são: lógica e representação dos dados, arquitetura e organização de computadores, multiprocessamento, gerenciamento de memória, sistemas de arquivos, tolerância a falhas, forense digital, redes de comunicação, segurança de redes, administração de redes, computação móvel e sem fio, sistemas paralelos e distribuídos, compreensão do modelo de programa armazenado de von Neumann e sua relação com as arquitetura e organização de computadores de uso geral.

Exemplos de suas aplicações são:

- Suporte de equipamentos;
- Redes de computadores (sem fio, internet, etc.).

1.6 ATIVIDADES EM AULA

Esta atividade tem por objetivo possibilitar ao aluno interagir com um tipo de *software* classificado como Ambiente Virtual de Aprendizagem (AVA). AVAs são sistemas computacionais que possuem diversos recursos para apoiar a educação, tais como:

- Repositórios: Locais onde o professor pode depositar arquivos com materiais instrucionais utilizados nas aulas, material de apoio, etc.;
- Escaninhos: Pastas individuais onde o estudante pode guardar materiais que podem ser visualizados pelo professor;
- Exercícios: Com essa ferramenta o estudante realiza exercícios que poderão ser corrigidos automaticamente, com *feedback* imediato;
- Fórum: Com essa ferramenta os estudantes poderão debater sobre diversos assuntos propostos pelo professor ou pelos próprios colegas.
- Mensagens: Com essa ferramenta os estudantes poderão trocar mensagens com o professor e também com os colegas. As mensagens enviadas e recebidas ficam no ambiente e podem, opcionalmente, ser enviadas para o e-mail dos destinatários.

Nessa atividade o aprendiz será guiado na exploração de algumas ferramentas do AVA utilizado pela UFABC, que é denominado TIDIA-AE. Durante a atividade o aluno deverá utilizar os diversos conceitos estudados neste capítulo.

Para realizar essa atividade, siga as instruções descritas nas próximas sub-seções.

Figura 1.11: Tela inicial (conexão) – TIDIA-AE.

1.6.1 CADASTRO E CONFIGURAÇÃO DE PERFIL NO TIDIA-AE

Para obter um cadastro no ambiente TIDIA-AE da UFABC, carregue um navegador de internet e acesse as informações sobre "Acesso ao sistema TIDIA-AE" no seguinte endereço:

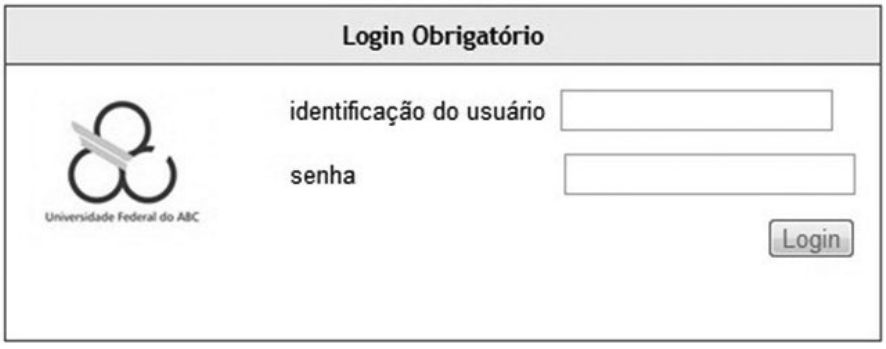
http://prograd.ufabc.edu.br/comunicados-ingressantes.

Conectando-se ao TIDIA-AE

Após o cadastro realizado, acesse o endereço: http://tidia-ae.ufabc.edu.br/portal. Clique no botão de login, conforme indica a seta vermelha na Figura 1.11, no canto superior direito da tela.

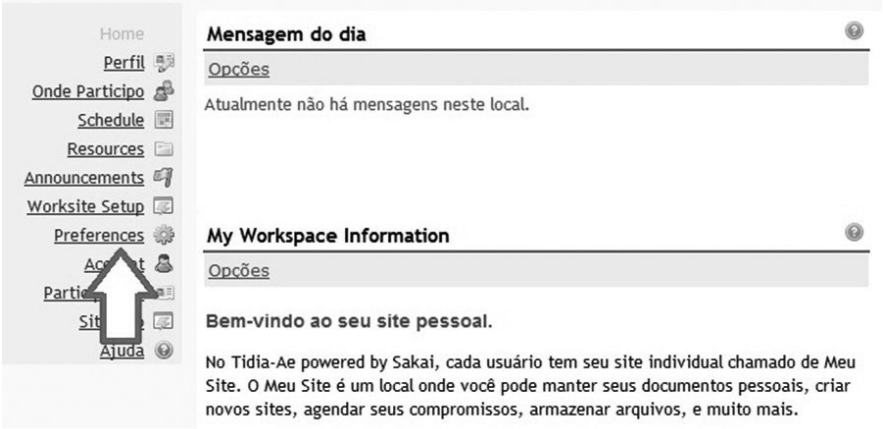
Para conectar-se ao ambiente, apenas digite o seu nome de usuário e a sua senha nos campos adequados, conforme ilustra a Figura 1.12.

Figura 1.12: Tela de login (conexão) – TIDIA-AE.



Caso tenha problemas com seu usuário e senha, acesse o seguinte link https://acesso.ufabc.edu.br/site/login e clique em "Recuperar Senha".

Figura 1.13: Tela de configuração do ambiente (preferências) – TIDIA-AE.

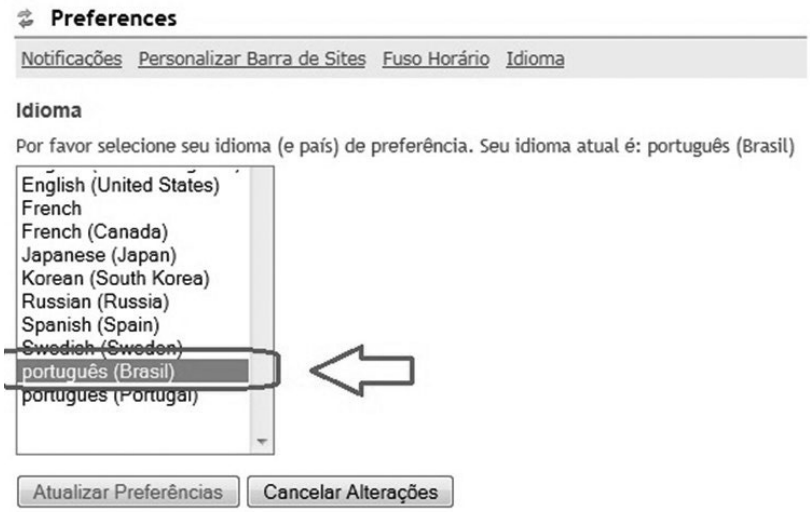


Configurando o TIDIA-AE – idioma

É possível fazer algumas configurações de preferências em relação ao ambiente TIDIA-AE, conforme ilustra a Figura 1.13. Por exemplo, pode-se configurar o idioma que se deseja utilizar no ambiente.

A Figura 1.14 exibe os passos para configurar o idioma para Português do Brasil.

Figura 1.14: Tela de configuração do idioma – TIDIA-AE.



Configurando o perfil do usuário

Recomenda-se preencher os dados do perfil do usuário, principalmente o e-mail, de forma que os outros participantes e, especialmente, o professor possam ter mais informações a respeito do usuário e possam enviar mensagens, etc. A Figura 1.15 apresenta a tela e os passos necessários para acessar o perfil.

Figura 1.15: Tela de configuração do perfil – TIDIA-AE.

1.6.2 INSCRIÇÃO EM DISCIPLINAS

Para ter acesso às atividades e aos materiais instrucionais de cada disciplina é necessário estar inscrito como participante dela. O usuário pode ser inscrito pelo professor ou pode inscrever a si próprio, caso a disciplina esteja aberta ao público para inscrição.

Para inscrever-se em uma disciplina aberta ao público, siga os seguintes passos, conforme indicados na Figura 1.16:

a) No menu lateral selecione a opção "Onde Participo" (My Worksites) –Passo 1 na figura;

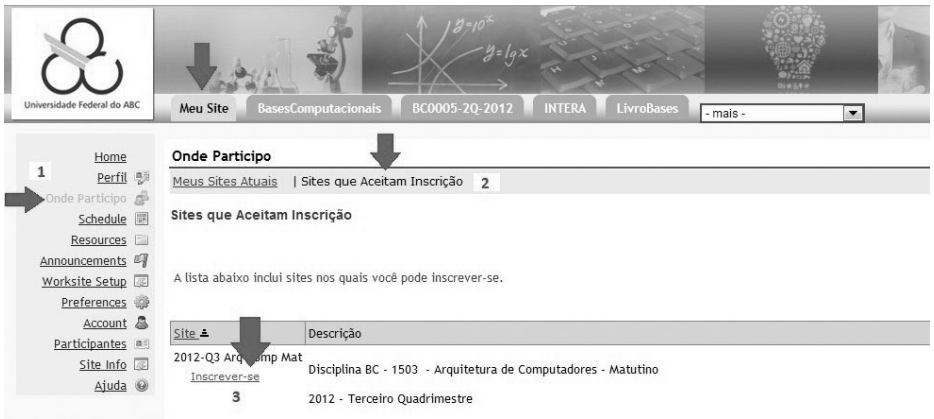
b) Selecione a opção "Sites que Aceitam inscrição" – Passo 2 na figura;

c) Localize a disciplina (site) em que deseja inscrever-se.

ATENÇÃO: Verifique com o professor qual o NOME da disciplina em que você deve se inscrever.

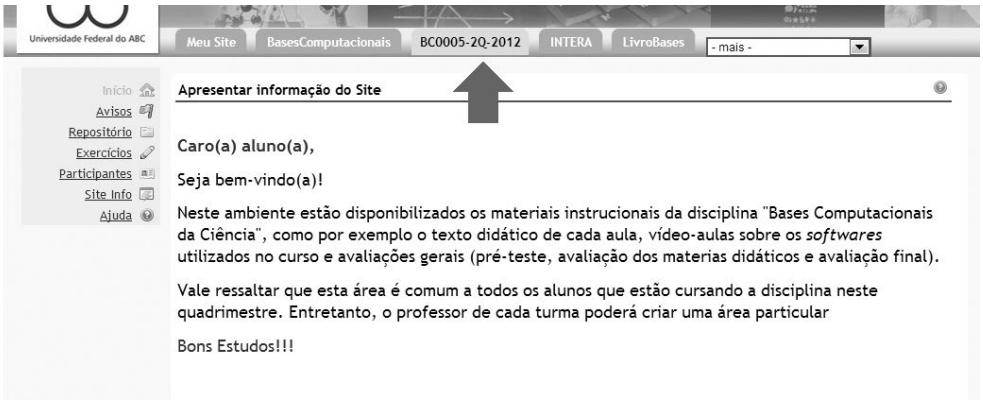
d) Selecione a opção "Inscrever-se" – Passo 3 na figura.

Figura 1.16: Tela de inscrição em disciplina (site) – TIDIA-AE.



Após concluir a inscrição, a nova *aba* da disciplina na qual se inscreveu deve aparecer, conforme mostra a Figura 1.17, a seguir. Observe também o menu lateral que existe relacionado à aba da disciplina conforme exibido na Figura 1.18, mais à frente. As diversas opções desse menu serão exploradas a seguir.

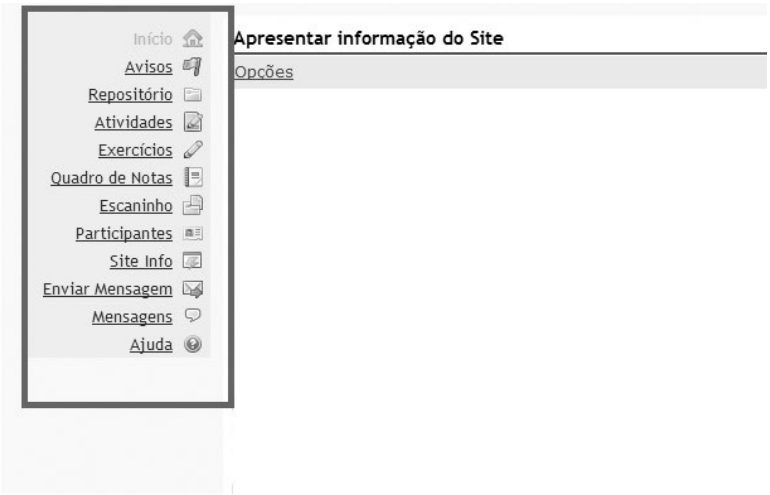
Figura 1.17: Aba da nova disciplina (inscrição) – TIDIA-AE.



1.6.3 ATIVIDADES NO TIDIA-AE

Estando inscrito corretamente na disciplina, você estará apto para realizar as atividades propostas para essa aula. Siga as instruções.

Figura 1.18: TIDIA-AE – Menu lateral.

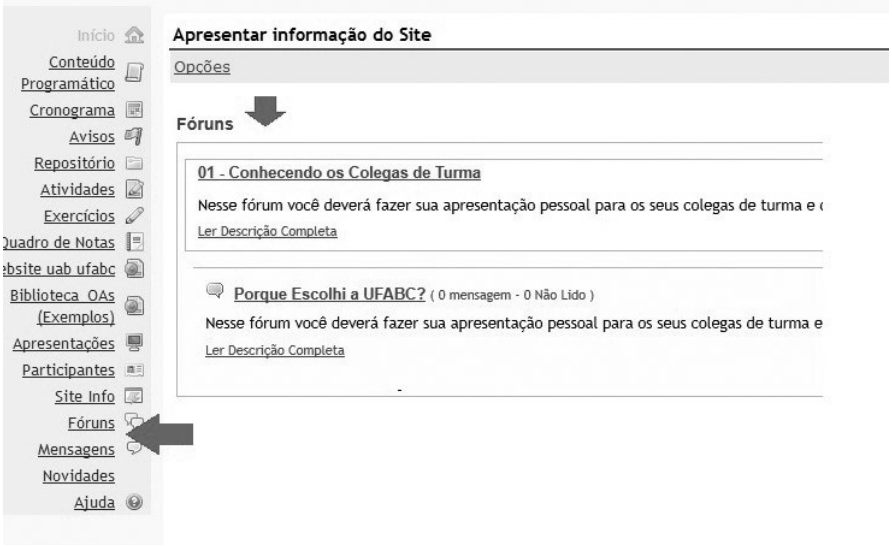


Fórum

Fórum é uma ferramenta de interação entre os participantes. Com ela, temas de discussão podem ser propostos de forma que os participantes possam expressar suas opiniões e também comentar as respostas dos demais participantes. É considerada uma ferramenta colaborativa de aprendizagem, uma vez que possibilita aprender com o outro.

A Figura 1.19 apresenta a tela de acesso ao fórum.

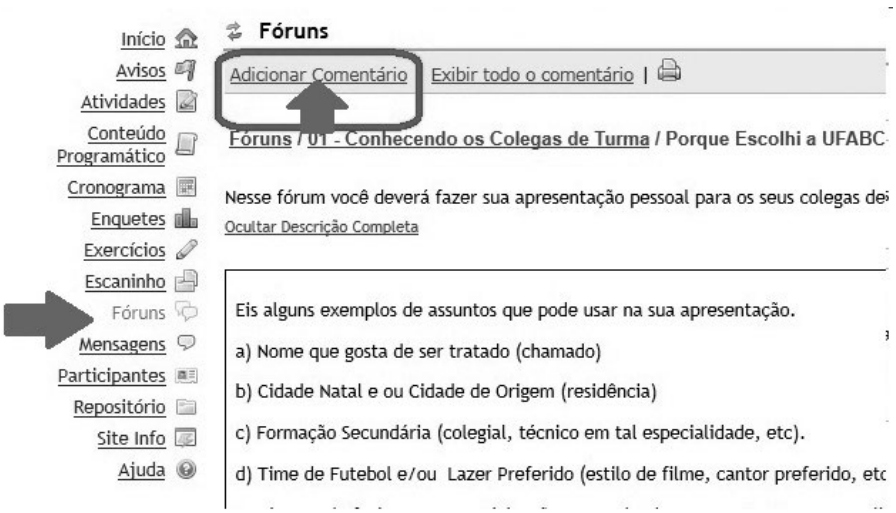
Figura 1.19: Tela de fórum – TIDIA-AE.



No fórum proposto para essa aula, cada participante deverá fazer uma breve apresentação pessoal, além de compartilhar com seus colegas as razões que o levaram a escolher estudar na UFABC.

Para acessar o fórum veja os passos indicados na Figura 1.20.

Figura 1.20: Tela do fórum "Porque Escolhi a UFABC".



Eis alguns exemplos de assuntos que você pode usar na sua apresentação.

a) Nome pelo qual gosta de ser tratado (chamado);

b) Cidade natal e ou cidade de origem (residência);

c) Formação secundária (colegial, técnico em determinada especialidade, etc.);

d) Time de futebol e/ou lazer preferido (estilo de filme, cantor preferido, etc.).

Lembre-se de fechar a sua participação respondendo a pergunta: *Porque escolhi a UFABC?*

Durante a semana, lembre-se de ler as apresentações de seus colegas no FÓRUM e fique à vontade para fazer comentários sobre as respostas dos colegas.

Algumas questões-guias para seus comentários são:

a) O que você achou do motivo pelo qual seu colega escolheu a UFABC?

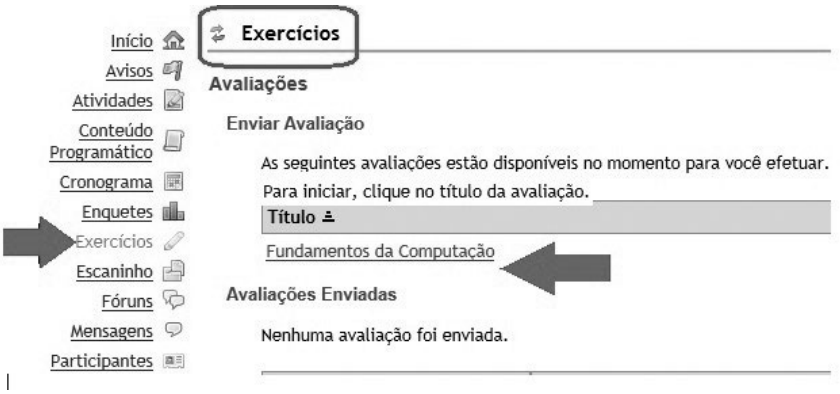
b) O que há de comum entre sua resposta e a do seu colega?

c) Você possui gostos comuns com seus colegas?

Exercícios

A ferramenta de Exercícios propicia ao estudante responder a questões disponibilizadas pelo professor. Para acessar os Exercícios siga os passos exibidos na Figura 1.21.

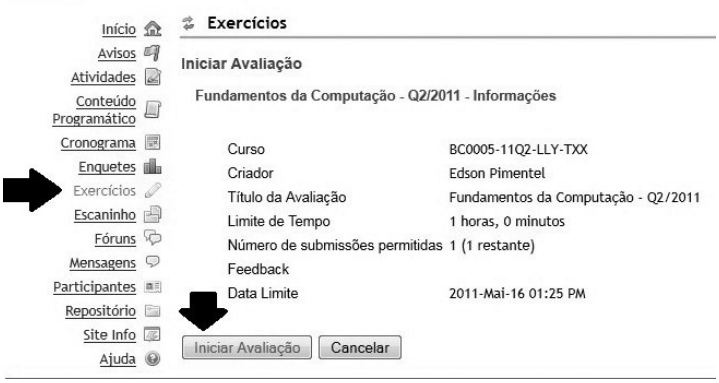
Figura 1.21: Tela de exercícios – TIDIA-AE.



Nessa atividade estão disponíveis 10 questões sobre o tema estudado na aula de determinado dia. Essa avaliação tem finalidade formativa, ou seja, a de orientar os seus estudos. Faça o seu melhor para tentar acertar cada resposta, mas fique atento ao *feedback*.

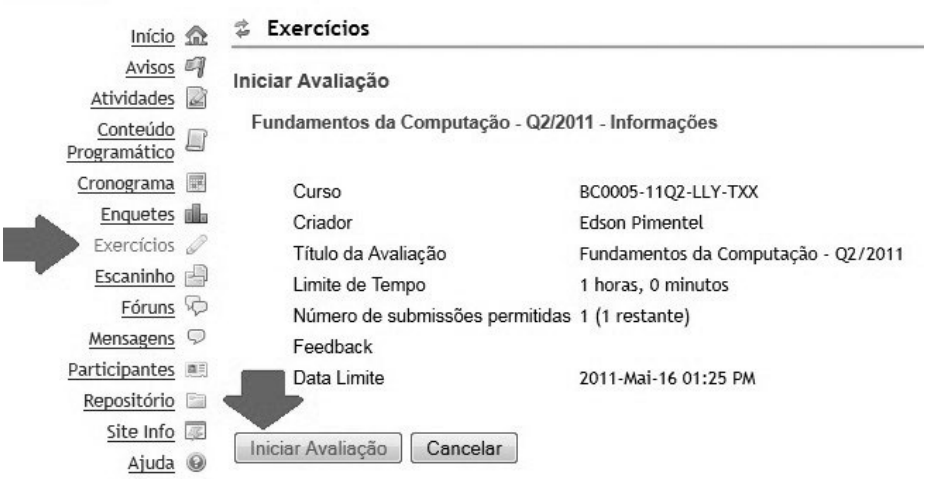
Após selecionar a avaliação a responder, dê um clique no botão "Iniciar Avaliação", conforme mostrado na Figura 1.22.

Figura 1.22: Tela de exercícios (iniciar avaliação) – TIDIA-AE.



Após responder, pode-se "Enviar para Atribuição de Nota" ou "Salvar para voltar depois" (*save for later*), conforme a Figura 1.23. Note, no entanto, que cada avaliação pode ter um tempo limite para sua conclusão.

Figura 1.23: Tela de exercícios (responder e avançar) – TIDIA-AE.



1.6.4 EXPLORANDO OUTRAS FERRAMENTAS DO TIDIA-AE

O TIDIA-AE possui diversas outras ferramentas que serão utilizadas no decorrer do curso. Algumas delas são apresentadas a seguir e você poderá explorá-las a seu tempo.

Repositório

A ferramenta de Repositório permite que o professor disponibilize recursos (apostilas, slides, arquivos em geral) que podem ser acessados por todos os participantes do curso. O TIDIA-AE permite aos participantes apenas "ler" esses arquivos e fazer download deles, mas não permite que esses participantes criem ou depositem arquivos no repositório.

A Figura 1.24 exibe como acessar o repositório.

Figura 1.24: Tela da ferramenta Repositório – TIDIA-AE.

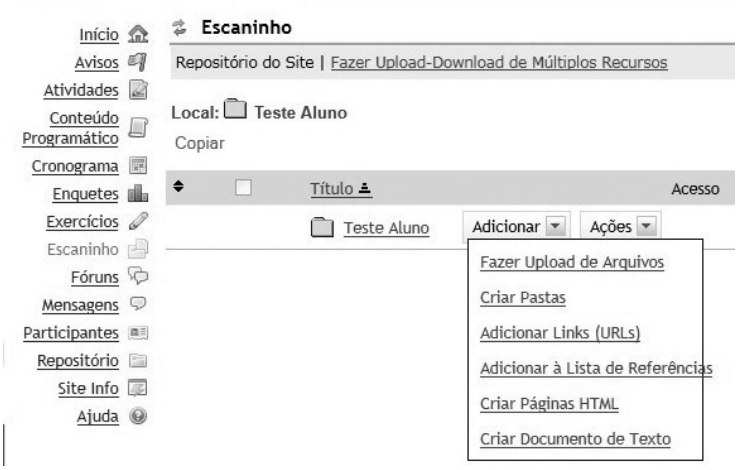
Escaninho

A ferramenta de Escaninho permite que o estudante disponibilize recursos (trabalhos, respostas de exercícios, arquivos em geral) para o professor. No escaninho, o estudante pode também guardar (em arquivos) suas "anotações" sobre a disciplina, entre outras. O TIDIA-AE não permite que outros participantes (a não ser o professor) tenham acesso a esses arquivos.

A Figura 1.25 exibe como acessar o escaninho.

ATENÇÃO: Ao disponibilizar recursos no escaninho, caso necessite que o professor veja o material, você deve avisá-lo por meio de uma mensagem. Não é usual que o professor verifique os escaninhos de todos os alunos, a não ser que seja avisado para fazê-lo ou que o próprio professor tenha solicitado ao aluno que deposite os arquivos no repositório.

Figura 1.25: Tela da ferramenta Escaninho – TIDIA-AE.

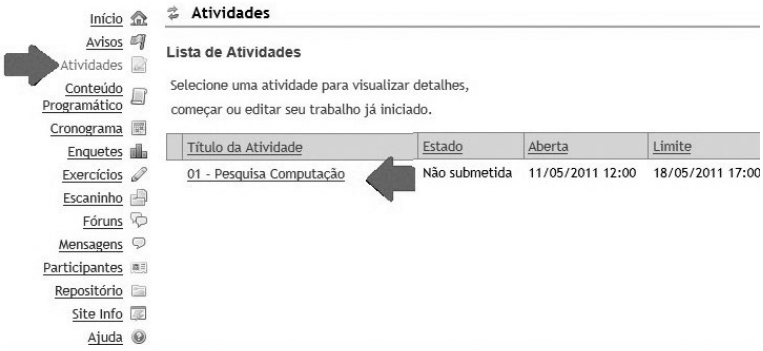


Atividades

Na ferramenta Atividades normalmente o professor disponibiliza "tarefas" que devem ser realizadas pelos alunos e submetidas para que o professor as avalie.

Acesse a ferramenta, conforme ilustrado na Figura 1.26, e note que há uma atividade disponível para você realizar no decorrer da semana.

Figura 1.26: Tela da ferramenta Atividades – TIDIA-AE.



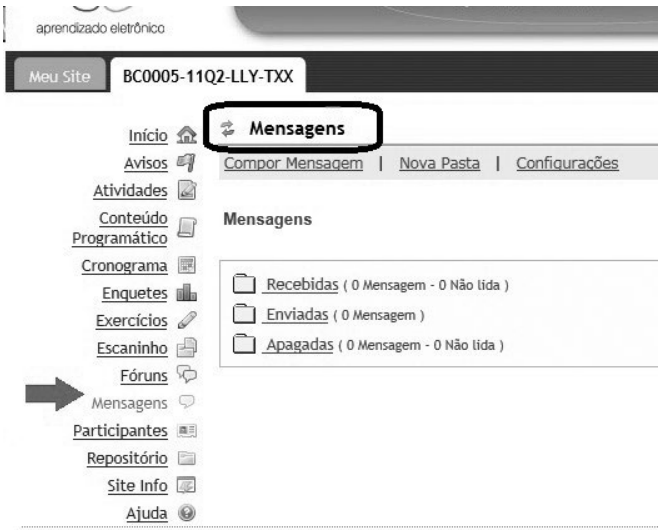
Nessa atividade você deverá realizar uma pesquisa sobre uma área da Ciência da Computação de sua escolha. Mais detalhes estão disponíveis na própria atividade no TIDIA-AE. Fique atento ao prazo para submissão.

Mensagens

Na ferramenta Mensagens é possível ler as mensagens recebidas do professor e dos colegas, bem como enviar mensagens para um ou mais participantes. Ao escrever uma mensagem pode-se indicar para a ferramenta que uma cópia dela seja enviada para o e-mail do(s) destinatário(s).

A Figura 1.27 mostra como acessar a ferramenta Mensagens.

Figura 1.27: Tela da ferramenta Mensagem – TIDIA-AE.

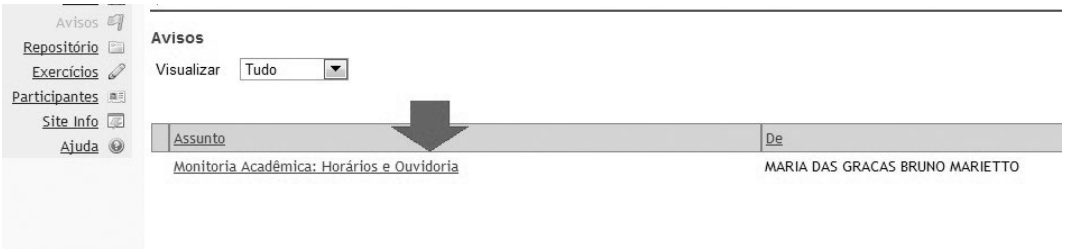


Avisos

Nessa ferramenta o participante poderá visualizar os Avisos deixados pelo professor. Normalmente, esses avisos poderão ser acessados diretamente a partir da tela inicial. Outra maneira é através do menu lateral, conforme mostra a Figura 1.28.

ATENÇÃO: Note o símbolo de "reciclagem" que existe acima do numeral (1). Esse símbolo funciona como se fosse uma opção para recarregar (reload) as opções (menu, listas, etc.) da ferramenta atual. Isso vale para qualquer ferramenta.

Figura 1.28: Tela de Avisos – TIDIA-AE.



Participantes

Na ferramenta Participantes é possível visualizar quem são os outros participantes (colegas de turma) do curso, bem como ter acesso aos seus e-mails, etc.

1.7 CONSIDERAÇÕES FINAIS

Os computadores estão por toda parte, nas mais diferentes formas. O "Pensamento Computacional" torna-se necessário nas mais diversas áreas do conhecimento. Assim, todos nós, além de precisarmos aprender a usar minimamente esses equipamentos, necessitamos compreender como funcionam os computadores e como essas máquinas "pensam". Este capítulo apresentou alguns elementos básicos dos Fundamentos da Computação para apoiar a compreensão da Computação como ciência organizada. Em suma, este capítulo demonstra que a Computação é uma ciência que possui fortes fundamentos, está evoluindo rapidamente e está se tornando ubíqua.

Os próximos capítulos irão apresentar, de forma concreta, aplicações computacionais nos mais variados cenários e com o suporte de distintas ferramentas. O próximo capítulo, por exemplo, tratará da Representação Gráfica de Funções com o suporte de ferramentas como o Scilab.

1.8 EXERCÍCIOS

1. Descreva um exemplo de funcionamento de um sistema computacional e seus componentes, considerando como sistema computacional um celular e como processamento o envio de mensagens de texto (SMS).

2. Descreva 2 outros exemplos de funcionamento de um sistema computacional e seus componentes. Procure ser original e criativo.

3. Assista ao vídeo sobre a história dos computadores e responda a questão: Como você imagina o futuro dos sistemas computacionais? O vídeo está disponível em: http://www.youtube.com/watch? v=F3qWg1JBPZg

4. O Google sites é um exemplo de *software* de serviço. Utilize esse *software* para criar uma página Web com seu currículo. Siga o tutorial disponível no link: http://www.youtube.com/watch? v=B4WmVeBxGqM

5. Responda: o que é computação ubíqua?

6. Procure na Internet exemplos de sistemas computacionais que são utilizados nas seguintes áreas do conhecimento: Base Experimental das Ciências Naturais, Bases Epistemológicas da Ciência Moderna, Bases Matemáticas, Estrutura da Matéria e Origem da Vida e Diversidade dos Seres Vivos. É necessário pelo menos um exemplo de cada área.

7. Classifique os sistemas encontrados no exercício 6 em pelo menos uma das 6 subáreas de *software*.

REFERÊNCIAS BIBLIOGRÁFICAS

BELL, T.; WHITTEN, I.; FELLOWS, M. *Computer Science Unplugged.* Universidade de Canterbury, Nova Zelândia, 2007. Disponível em http://csunplugged.org/sites/default/files/books/CSUnpluggedTeachers-portuguese-brazil-feb-2011.pdf. Acesso em 16/02/2013.

CASSEL, L.; CLEMENTS, A.; DAVIES, G.; GUZDIAL, M.; McCAULEY, R.; McGETTRICK, A.; SLOAN, B.; SNYDER, L.; TYMANN, P.; WEIDE, B. W; SEIDMAN, S.; McGETTRICK, *A. Computer Science Curriculum 2008*: An Interim Revision of CS 2001, 2008. Disponível em http://www.acm.org/education/curricula/ComputerScience2008.pdf Acesso em 16/02/2013.

FOROUZAN, B.; MOSHARRAF, F. *Fundamentos da Ciência da Computação.* [S.l.]: Editora Cengage, 2011.

GERSTING, J. L. *Fundamentos Matemáticos para a Ciência da Computação*. 5ª Edição. Rio de Janeiro:Editora Ltc., 2004. 612 p.

HAMBRUSCH, S. *et al*. A multidisciplinary approach towards computational thinking for science majors. In: *Proceedings of SIGCSE'09*. Chattanooga, Tennessee, USA: [s.n.]. 2009.

HANSMANN, U. et al. *Pervasive computing Handbook*. [S.l.]: Springer, 2003.

HENDERSON, P. B. et al. Computational thinking. In: *Proceedings of SIGCSE'07*. Covington, Kentucky, USA: [s.n.], 2007.

JOHNSON, G. *The World*: In Silica Fertilization; All Science Is Computer Science. New York Times on the Web, 2011. Disponível em http://www.nytimes.com/2001/03/25/weekinreview/the-world-in-silica-fertilization-all-science-is-computer-science.html. Acesso em 16/02/2013.

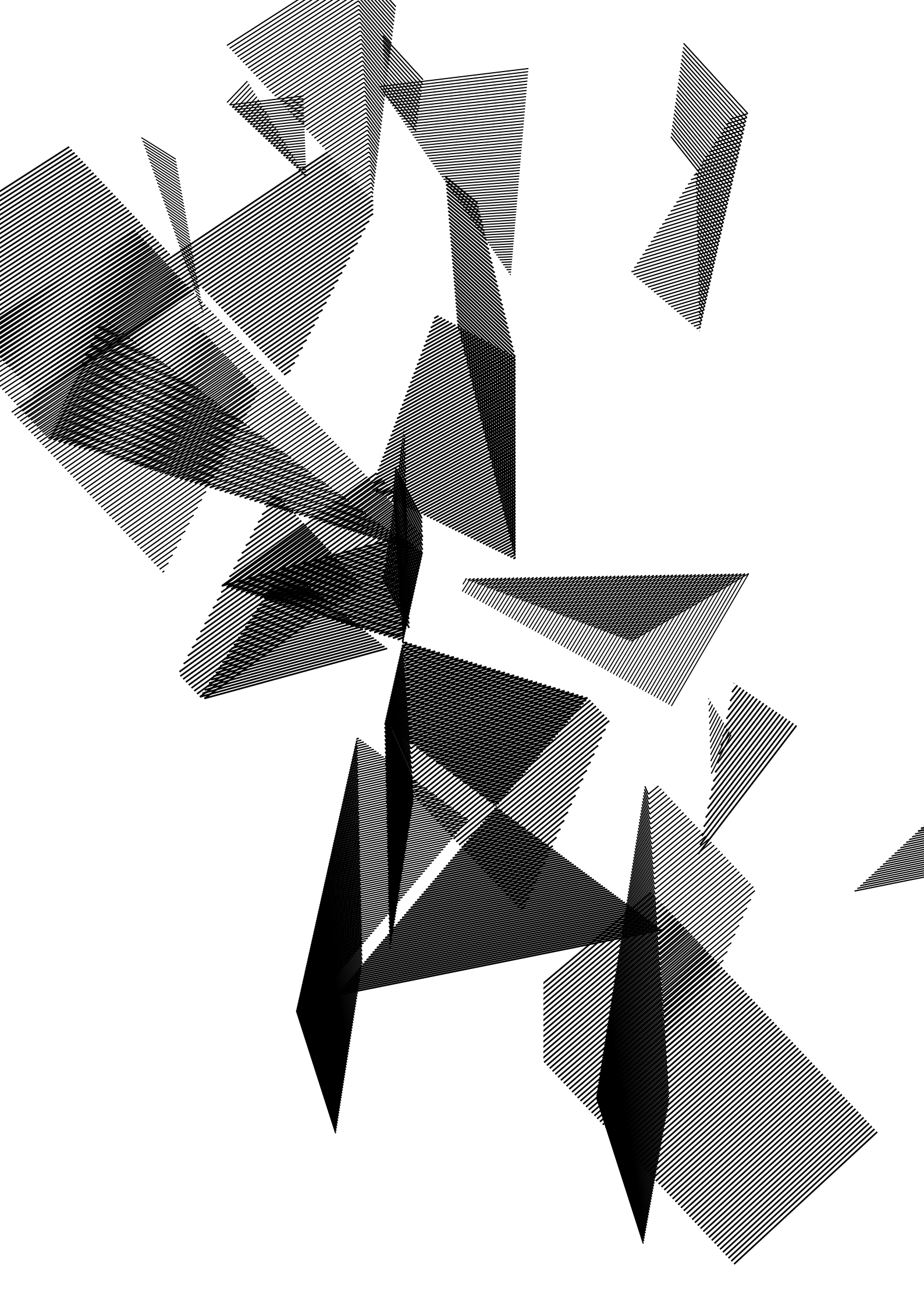
LANCHARRO, E. A.; LOPES, M. G.; FERNANDEZ, S. P. *Informática Básica*. São Paulo: Pearson, 2004. 288 p.

SEEHORN, D.; CAREY, S.; FUSCHETTO, B.; LEE, I.; MOIX, D.; O'GRADY-CUNNIFF, D.; OWENS, B. B.; STEPHENSON, C.; VERNO, A. *K–12 Computer Science Standards*. 2001. Disponível em http://csta.acm.org/Curriculum/sub/CurrFiles/CSTA_K-12_CSS.pdf. Acesso em 16/02/2013.

TANENBAUM, A. S. *Organização estruturada de computadores.* 5ª Edição. São Paulo. Editora Pearson, 2007. 464 p.

TENÓRIO, R. M. *Computadores de Papel* – Máquinas Abstratas Para um Ensino Concreto. São Paulo:Editora Cortez, 2001. 120 p.

WING, J. Computational thinking. Communications of the ACM, v. 49, n. 3, 2006.



CAPÍTULO 2

REPRESENTAÇÃO GRÁFICA DE FUNÇÕES

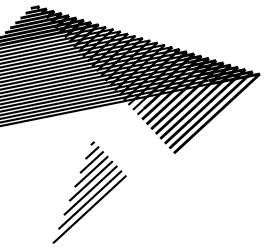
Aline Neves
Irineu Antunes Júnior
Marcio Eisencraft

2.1 INTRODUÇÃO

Neste capítulo discutimos como funções podem ser empregadas para representar fenômenos da natureza e como o uso de ferramentas computacionais voltadas a cálculos científicos pode facilitar bastante o estudo destes fenômenos pela representação gráfica de funções.

Inicialmente, na Seção 2.2, discutimos questões envolvidas na representação gráfica de funções e mostramos alguns casos em que tais gráficos podem ser úteis no auxílio do entendimento de um problema. Em seguida, na Seção 2.3, introduzimos uma ferramenta computacional que permite realizar cálculos científicos e gráficos de maneira rápida e prática. Para que você possa praticar e compreender melhor as questões apresentadas, na Seção 2.4, oferecemos atividades a serem desenvolvidas em sala de aula. Na Seção 2.5, fazemos considerações finais sobre o capítulo e, finalmente, na Seção 2.6, oferecemos alguns exercícios de fixação seguidos de exercícios mais avançados.

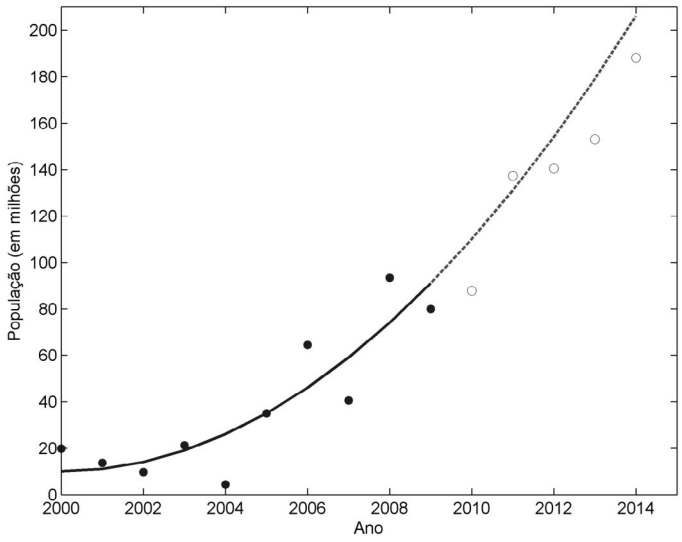
2.2 USO DE FUNÇÕES EM CIÊNCIA E ENGENHARIA



A Ciência e a Engenharia sempre buscam modelar fenômenos naturais e físicos por funções matemáticas que possam, pelo menos de maneira simplificada, reproduzir os comportamentos observados na Natureza. Neste sentido, podemos citar como exemplos leis que regem o comportamento de gases, escoamento de fluidos, propagação de ondas, movimento de corpos, crescimento de populações, além de muitos outros. Muitas vezes, dado o modelo matemático de um sistema, nos deparamos com a necessidade de visualizar o comportamento dele, ou então precisamos encontrar uma solução, mas não sabemos ao certo por onde começar a procurá-la. Nestes casos, gráficos das funções em questão podem auxiliar no entendimento, fornecendo, inclusive, uma primeira aproximação para a solução procurada.

Suponha, por exemplo, que se deseja prever a taxa de crescimento de uma faixa sócio-econômica da população, em um período de anos não abordado numa pesquisa. Seria necessário fazer um modelo matemático a partir do gráfico obtido com os dados disponíveis para se conseguir a informação desejada por meio de uma extrapolação. A Figura 2.1 ilustra tal processo.

Figura 2.1: Exemplo de extrapolação.



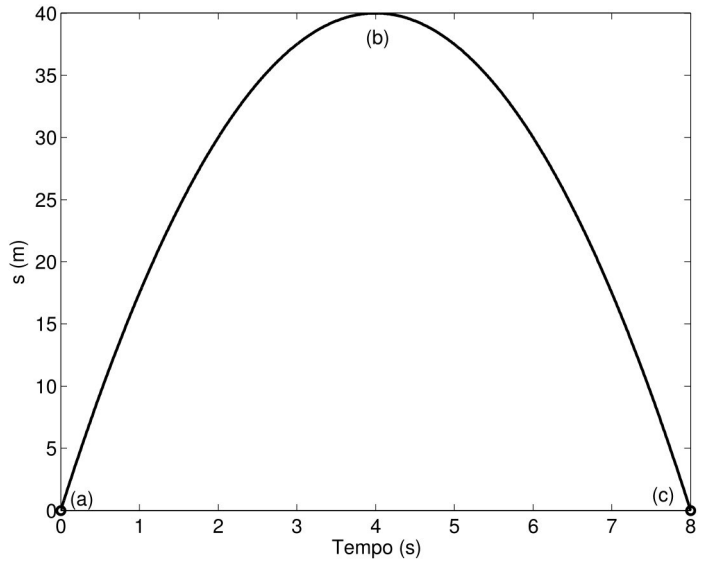
Os pontos cheios são os dados da pesquisa realizada entre os anos de 2000 e 2009; a curva cheia é uma função matemática que descreve os dados obtidos nestes anos; a curva tracejada é a função extrapolada em datas futuras; os pontos vazios são os dados futuros (desconhecidos).

Exemplo 1: Para iniciar este estudo, considere um caso bastante simples de um movimento uniformemente variado definido pela equação:

$$s = s_0 + v_0 t + \frac{1}{2} a t^2 \tag{2.1}$$

onde s é a posição atual do corpo em movimento, s_0 é a posição na qual ele começou o movimento, v_0 é a sua velocidade inicial, a é sua aceleração e t é o tempo decorrido desde o início do movimento. Como exemplo, vamos supor $s_0 = 0$, $v_0 = 20$ m/s e $a = -5$ m/s^2, ou seja, o corpo está freando. Se quisermos visualizar como a posição s irá variar em função do tempo t, podemos fazer um gráfico relacionando estas duas grandezas. A Figura 2.2 ilustra o movimento em questão. Observando o gráfico, podemos facilmente obter algumas características do movimento: como o corpo está freando, a posição máxima que ele irá atingir é $s = 40$ m e ele levará 4 segundos para atingi-la, como mostrado pelo ponto (b) na Figura 2.2. Neste ponto a sua velocidade será nula. A partir daí, a aceleração negativa pode ser vista como uma aceleração em sentido contrário e, portanto, o corpo começará a voltar, atingindo a posição $s = 0$ novamente em $t = 8$ s (ilustrado pelo ponto (c) na figura). Os dados observados graficamente podem ser facilmente conferidos através da substituição dos valores citados na equação (2.1). Em particular, os instantes em que $s = 0$ são facilmente encontrados calculando-se as raízes da equação.

Figura 2.2: Espaço em função do tempo para um movimento uniformemente variado.



Em (a) e (c) temos os pontos onde o corpo está em s=0, enquanto que (b) ilustra o valor máximo de s.

Em alguns casos, no entanto, a resolução analítica do problema pode ser bastante complicada, se não impossível.

Exemplo 2: Considere a obtenção das raízes, ou seja, dos pontos nos quais a seguinte função se anula:

$$f(x) = sen(x) + \cos(1 + x^2) - 1 \tag{2.2}$$

Neste caso, o gráfico da função pode ser útil para auxiliar a solução do problema. Observando o gráfico de $f(x)$ para $x \in [0,4]$, ilustrado na Figura 2.3, vê-se que $f(x)$ será nula para dois valores de x neste intervalo: x_1 e x_2. Alterando a escala do gráfico (Figura 2.4), vemos que a primeira raiz se encontra em $x_1 \approx 1.9$ e a segunda em $x_2 \approx 2.5$.

Figura 2.3: Gráfico de f(x) dada por (2.2).

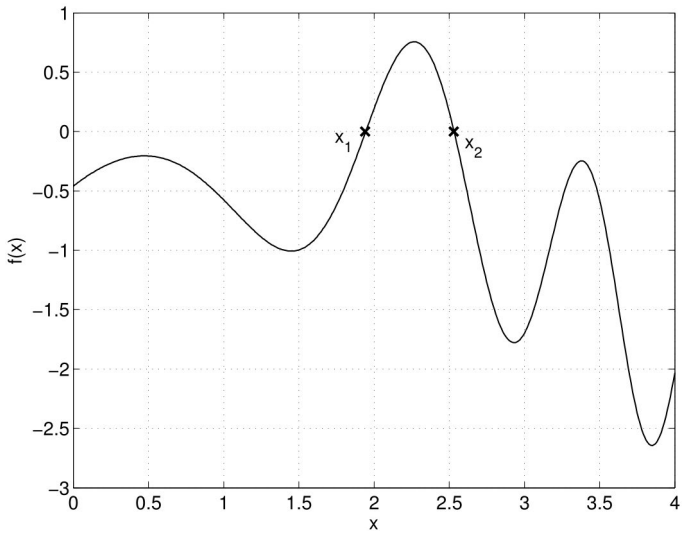
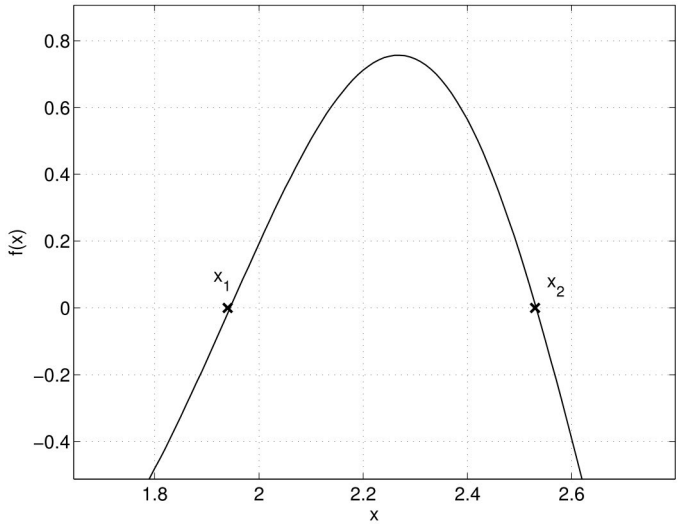


Figura 2.4: Região em torno das raízes de (3.2).

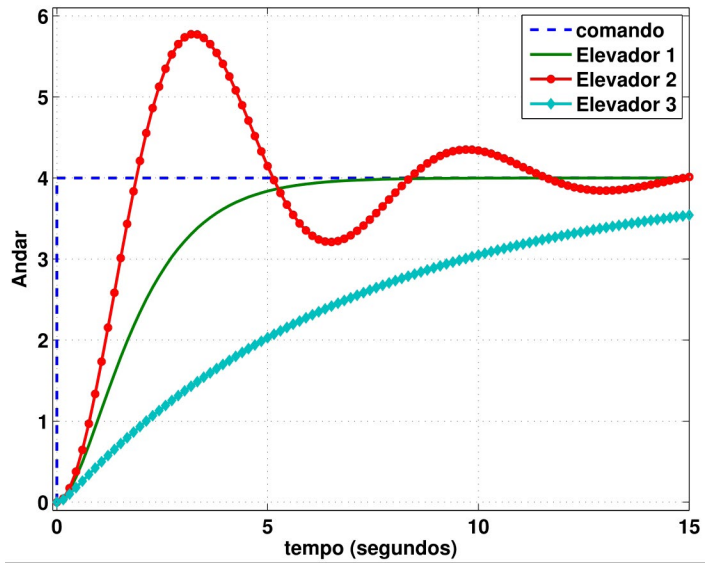


Por outro lado, nem sempre esta abordagem é possível. Técnicas mais sofisticadas de busca de raízes de funções são estudadas na disciplina de Cálculo Numérico.

Um gráfico também pode auxiliar na visualização de um problema mesmo que não se conheça seu modelo em detalhes. Observando o gráfico, tem-se uma ideia do comportamento geral de sistemas empregados na prática. Este é o caso dos exemplos a seguir.

Exemplo 3: Suponha que um elevador será usado para levar uma carga ao 4º andar de um edifício. A Figura 2.5 apresenta o comportamento de três elevadores diferentes após receberem um comando solicitando que o elevador se desloque para o 4º andar.

Figura 2.5: Resposta de três elevadores a um comando.



Imagine-se dentro de um destes elevadores. Qual deles lhe proporcionaria a viagem mais eficiente e confortável? Claramente, o Elevador 2 não é lá uma boa escolha. Veja que ele passa em muito do 4º andar, retorna e, além disso, fica chacoalhando antes de parar. O Elevador 3 vai exigir do seu ocupante uma larga dose de paciência, já que ele demora muito para atingir o andar desejado. Dentre os apresentados, o Elevador 1 é a melhor opção já que ele alcança o 4º andar num tempo razoavelmente curto e sem oscilações. Veja que todas estas informações foram obtidas do gráfico da Figura 2.5. Em cursos posteriores, você vai aprender que estas três respostas são denominadas criticamente amortecida (Elevador 1), subamortecida (Elevador 2) e superamortecida (Elevador 3).

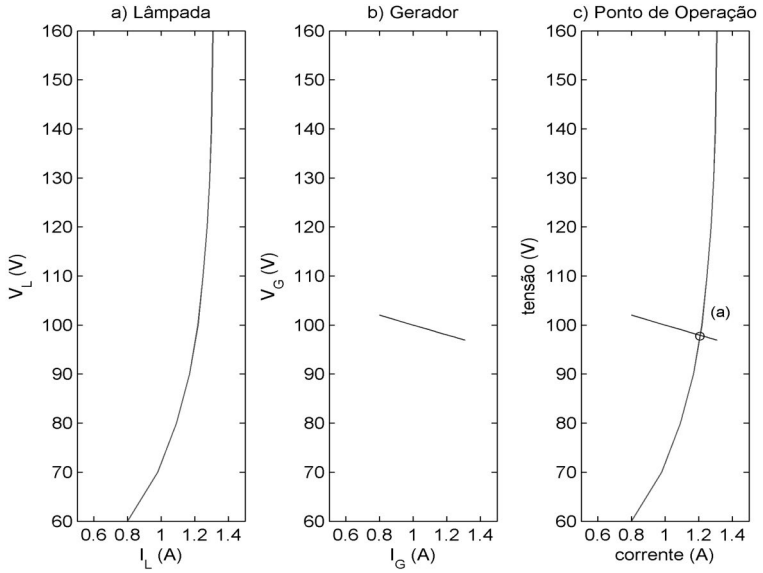
Exemplo 4: Dispositivos elétricos podem ser caracterizados pelos valores de tensão e corrente medidos nos seus terminais. Por exemplo:

- Na Figura 2.6.a, apresentamos os valores medidos de tensão (V_L) e corrente (I_L) nos terminais de uma lâmpada incandescente. Deve-se comentar que uma lâmpada incandescente possui valores nominais de operação que são impressos no seu bulbo. No entanto, na prática, o valor da corrente consumida depende da tensão que é efetivamente aplicada.
- Na Figura 2.6.b, são representados os valores de tensão (V_G) e corrente (I_G) medidos nos terminais de um gerador.

Deseja-se ligar o gerador na lâmpada. Neste caso, durante a operação do circuito, deve-se ter $V_L = V_G$ e $I_L = I_G$. Como encontrar o valor de corrente e tensão a que será submetida a lâmpada?

Este problema pode ser resolvido graficamente, conforme mostrado na Figura 2.6.c, na qual são desenhadas as curvas características dos dois dispositivos. O cruzamento das curvas fornece o ponto de operação (a) indicado neste gráfico. Lendo os valores nos eixos, obtemos que a corrente será, aproximadamente, de 1,2 A e a tensão de 98 V.

Figura 2.6: Exemplo de lâmpada ligada a um gerador.



Exemplo 5: Considere a seguinte reação química reversível:

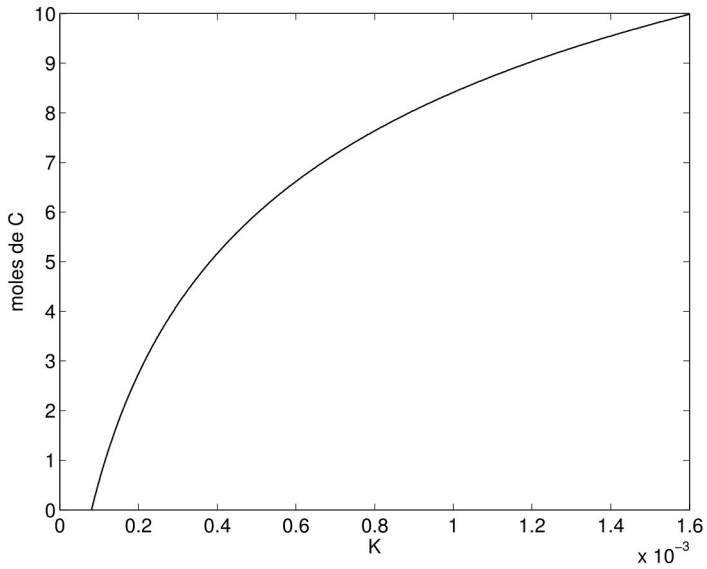
$$2A + B \leftrightarrow C \tag{2.3}$$

caracterizada pela relação de equilíbrio

$$K = \frac{c_c}{c_A^2 c_B} \tag{2.4}$$

onde c_i representa a concentração do componente *i* (Chapra, 2008). Na Figura 2.7 são mostrados quantos moles de *C* são produzidos dependendo do valor de *K*. Mesmo não conhecendo o modelo em detalhes, o gráfico permite visualizar que a quantidade de *C* aumenta rapidamente para *K* entre 0.1 e 0.6, diminuindo a taxa de crescimento após este valor.

Figura 2.7: Número de moles de C em função do valor de *K*.



Existem diversas ferramentas computacionais que podem ser usadas para produzir gráficos de funções, desde planilhas até programas bastante avançados de simulação numérica como o Matlab, o Octave e o Scilab. Neste curso, utilizamos o Scilab para estudar alguns problemas simples para os quais uma primeira aproximação da resposta pode ser obtida facilmente por meio de gráficos. Antes, no entanto, vejamos como utilizar comandos básicos no Scilab.

2.3 EMPREGO DO SCILAB PARA FAZER GRÁFICOS

O Scilab (assim como o Matlab e o Octave) é um *software* para cálculos matemáticos avançados usado em computação científica e Engenharia. Conta com bibliotecas de funções matemáticas prontas, facilidade de programação e recursos gráficos avançados, facilitando o projeto e a análise de sistemas de controle, o processamento de sinais, a análise estatística de dados e a otimização de funções.

Ao abrirmos o programa, logo visualizamos o ambiente de trabalho. O *prompt*, representado por uma seta, indica que o programa está pronto para receber uma linha de comando como se fosse uma calculadora científica. A seguir, veremos alguns comandos básicos.

2.3.1 OPERADORES BÁSICOS

A seguir, temos alguns exemplos de operações matemáticas básicas:

• Se digitarmos um comando simples como, por exemplo,

```
-->x = 2;
```

criaremos uma variável real chamada *x*, cujo valor é igual a 2. Uma variável é sempre composta por dois elementos: um identificador (ou seja, o seu nome) e um conteúdo que represente o seu valor. No caso apresentado, o identificador da variável é *x* e seu valor é igual a 2. Isso significa que foi alocado um endereço na memória RAM com o conteúdo igual a 2, o qual será referenciado pelo nome *x*. Podemos criar variáveis de vários tipos:

• Inteiro: o conteúdo da variável será dado por um número pertencente ao conjunto dos números inteiros Z como, por exemplo, -2, 0, 34.

• Real: o conteúdo da variável será dado por um número pertencente ao conjunto dos números reais como, por exemplo, -234.45, 76, 2.7.

• Caracter: o conteúdo da variável será dado por um único caracter, que pode ser um caracter numérico (de 0 a 9), alfanumérico (de A a Z, maiúsculo ou minúsculo) ou especial (como por exemplo %, @, ! etc.).

• String: o conteúdo da variável será dado por um conjunto de caracteres numéricos, alfanuméricos ou especiais como, por exemplo, *Este é um conjunto de caracteres imprimíveis*.

• Lógico: o conteúdo da variável só pode assumir os valores verdadeiro, representado pelo número 1, ou falso, representado por 0.

No caso do *software* Scilab, também é possível criar variáveis de outros tipos como, por exemplo, *string* ou caracter. Contudo, como nosso objetivo é executar cálculos matemáticos, as variáveis em geral serão reais.

O ponto-e-vírgula ao final da instrução não é obrigatório. Caso ele não seja colocado, a variável será apresentada na tela:

```
-->x = 2
 x =
   2
```

Daí em diante, cada vez que empregamos a variável *x*, estaremos utilizando o seu conteúdo.

```
-->y=x+5
 y =
  7
```

Esta operação define *y* como sendo uma variável com valor igual ao valor de *x* mais cinco, ou seja, *y* terá um valor igual a 7.

```
-->z=x*y
z =
 14
```

Neste caso, *z* será igual à multiplicação dos valores guardados em *x* e *y*, ou seja, *z* será igual a 14.

```
-->w=z/x
 w =
  7
```

Aqui, *w* será igual à divisão dos valores guardados em *z* e *x*, ou seja, *w* será igual a 7.

Além dos operadores acima, o Scilab possui várias funções matemáticas que podem ser facilmente utilizadas como:

- logaritmo: log(16) (logarítmo base e), log10(16).
- exponencial: exp(-2).
- raíz quadrada: sqrt(4).
- funções senoidais: sin(x), cos(x), tan(x), cotg(x).

2.3.2 FAZENDO O GRÁFICO DE UMA FUNÇÃO SIMPLES

Consideremos a função

$$f(x)=sen(x) \tag{2.5}$$

no intervalo $x \in [0,2\pi]$. Sempre que desejamos produzir um gráfico de uma função, precisamos, primeiramente, definir em quais pontos gostaríamos de visualizar a função, ou seja, em quais valores de *x*. No Scilab, existem duas formas para se definir estes valores:

1. Definindo diretamente os pontos *x* nos quais queremos plotar a função. Neste caso basta digitarmos, na linha de comando do Scilab, diretamente os valores de *x* que nos interessam, separados por um espaço e entre colchetes:

```
--> x=[0 0.5*%pi %pi 1.5*%pi 2*%pi]
```

No comando acima, escolhemos cinco valores de *x* entre 0 e 2π com passo 0.5. Estes são os valores para os quais desejamos calcular *f*(*x*) e, em seguida, construir o gráfico. Neste caso, *x* é chamado de vetor, pois possui mais de um elemento. É

interessante notar que o valor de π, no Scilab, é obtido precedendo-se a palavra pi do símbolo %.

2. Definindo um intervalo de valores de *x* no qual queremos plotar a função *f*(*x*). Neste caso, podemos usar diretamente o intervalo [0,2π]. Assim, para definir *x*, podemos usar o seguinte comando em Scilab:

```
--> x = 1° valor : passo : último valor do intervalo
```

Tal instrução criará um vetor x cujo primeiro valor será igual ao primeiro valor do intervalo. O segundo valor será dado pelo valor anterior somado ao valor do passo. Isso irá se repetir até que o valor da soma seja igual ou menor do que o último valor do intervalo. Por exemplo, se digitarmos o seguinte comando:

```
--> x=0:0.5*%pi:2*%pi;
```

produziremos um vetor exatamente igual ao que foi digitado no exemplo anterior. Já a instrução

```
--> x=0.5:0.25:1.5;
```

irá fornecer um vetor com os seguintes valores: x = [0.5 0.75 1 1.25 1.5]. Outro exemplo interessante seria

```
--> x=0:0.2:0.7;
```

que irá resultar num vetor x=[0 0.2 0.4 0.6]. Ou seja, o último elemento será menor do que o último valor especificado para o intervalo. Isso acontece porque, se somarmos o valor do passo mais uma vez, o valor resultante excede o intervalo especificado.

Observe que, quanto menor for o valor do passo, mais valores teremos no vetor x.

Quando o passo desejado for igual a 1, ele não precisa ser digitado. Por exemplo, a instrução

```
--> x=1:5;
```

irá gerar um vetor com os seguintes valores: x = [1 2 3 4 5].

Uma vez criado um vetor x, precisamos agora encontrar os valores de *f*(*x*) em cada ponto do vetor. É neste momento que *softwares* para cálculos matemáticos como o Scilab facilitam bastante a tarefa: eles são otimizados para trabalhar com vetores e matrizes. Assim, a simples instrução

```
--> f=sin(x);
```

irá gerar um vetor f cujos elementos são dados pelo seno dos elementos definidos em x. Finalmente, podemos fazer o gráfico desejado, utilizando o comando

```
--> plot(x,f);
```

em que o primeiro parâmetro se refere ao eixo das abscissas e o segundo, ao eixo das ordenadas.

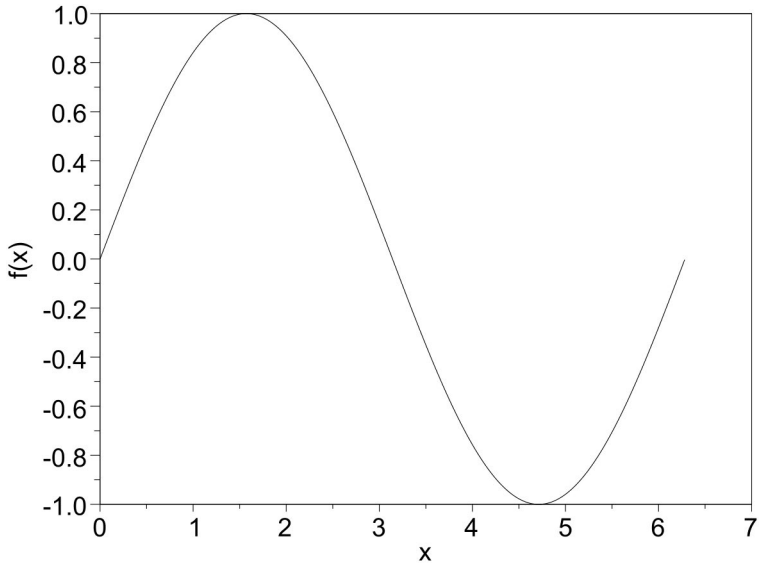
Resumindo o que foi visto até aqui, poderíamos produzir o gráfico desejado utilizando a seguinte sequência de instruções:

```
--> x=0:0.01:2*%pi;
```

```
--> f=sin(x);
--> plot(x,f);
```

O gráfico resultante é mostrado na Figura 2.8.

Figura 2.8: Gráfico da função seno.



Para colocar nomes nos eixos dos gráficos, podemos usar os comandos:

```
--> xlabel('nome do eixo x');
--> ylabel('nome do eixo y');
--> title('nome da janela do grafico');
```

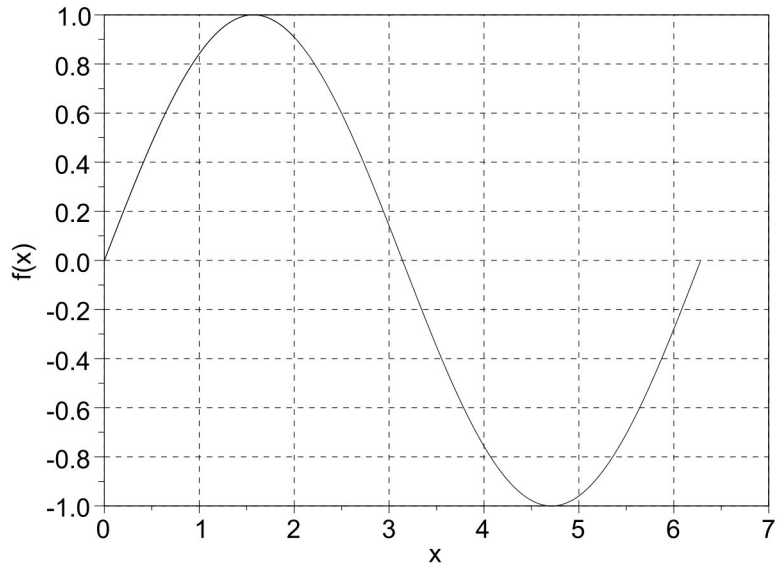
Para colocar as linhas de grade no gráfico, podemos usar o comando:

```
--> set(gca(),"grid",[1 1]);
```

Neste caso, o gráfico ficará como apresentado na Figura 2.9. O comando "set" é utilizado para ajustar o valor de alguma propriedade de um gráfico tendo, como parâmetros, o que se deseja ajustar, a propriedade em questão e o valor que esta deve assumir. Neste caso, o que se deseja ajustar são os eixos do gráfico o que é referenciado por gca, a propriedade são as linhas de grade, ou seja, "grid" e o valor que este parâmetro deve assumir é [1 1], tornando as linhas visíveis. Para retirar a grade, basta usar o comando

```
--> set(gca(),"grid",[-1 -1]);
```

Figura 2.9: Gráfico da função seno (com linhas de grade).



Para alterar a cor da curva no gráfico, podemos adicionar um último parâmetro ao comando "plot", dado pela primeira letra da cor desejada em inglês. Por exemplo,

```
-->plot(x,f,'r');
```

desenhará a curva em vermelho (*red* em inglês). Uma exceção acontece para a curva em preto, já que, em inglês, azul e preto começam com a letra b (*blue* e *black*). Somente neste caso, portanto, a última letra deve ser usada, isto é, para fazer a curva em preto devemos usar o comando plot(x,f,'k'). Para maiores informações sobre o comando "plot", digite "help plot".

2.3.3 CUIDADOS COM OUTROS TIPOS DE FUNÇÕES

Um certo cuidado precisa ser tomado quando precisamos plotar gráficos de funções que envolvam multiplicação ou divisão de vetores. Consideremos, por exemplo, a função $f(x) = xe^x$, no intervalo $x \in [0,1]$. Seguindo os passos discutidos na seção anterior, temos:

• Criação de um vetor de valores para *x*:

```
--> x=0:0.01:1;
```

• Cálculo de *f(x)*: se digitarmos a instrução exp(x), teremos um vetor com os resultados do cálculo da exponencial de cada valor existente no vetor x. Este vetor precisará ser multiplicado por x e, portanto, temos uma multiplicação de dois vetores: x e exp(x). Neste momento precisamos tomar um certo cuidado. Se digitarmos a instrução x*exp(x), interpretada pelo Scilab como o produto de dois vetores, este irá supor que desejamos realizar uma multiplicação vetorial entre os dois, o que não é o caso. O que queremos é multiplicar os dois termo a termo, isto é, queremos multiplicar o primeiro elemento de x pelo primeiro elemento de exp(x) e o segundo elemento de x pelo segundo de exp(x) e assim

por diante. Para conseguir tal resultado corretamente, precisamos utilizar a seguinte instrução:

```
--> f=x.*exp(x);
```

Ou seja, o operador multiplicação * precisa ser precedido de um ponto. O mesmo acontecerá se tivermos que calcular a divisão entre dois vetores como no caso da função $f(x) = x/(1+x^2)$. Assim, um ponto deve ser colocado antes do operador /.

Basicamente, toda vez em que tivermos vetores envolvidos e quisermos realizar as operações a cada elemento do vetor, precisaremos usar o ponto antes do operador em questão. O mesmo vale quando queremos elevar os elementos de um vetor a uma certa potência. Para calcular x^2, por exemplo, precisamos digitar a instrução x.^2.

• Por fim, o gráfico pode ser feito por meio do comando: plot(x,f).

2.4 ATIVIDADES EM AULA

A seguir, empregamos o Scilab como uma ferramenta computacional para resolver alguns problemas.

Atividade 1: A empresa COLKS é a uma indústria automobilística do pequeno país chamado Govers, onde a moeda of icial é o dubila. O lucro mensal da COLKS é função do número de carros produzidos no mês. Ela tem um custo fixo de 50 dubilas e um custo variável função do número de carros produzidos no mês. Usando N_C para definir o número de carros produzidos em um mês, o custo variável é dado por $48N_C^{0.9}$. Vamos dizer que ela venda cada carro por 50 dubilas. Assim, o seu lucro L mensal é dado por

$$L = 50N_c - 48N_c^{0.9} - 50 \tag{2.6}$$

1. Usando uma calculadora ou o computador, determine o lucro L da COLKS ao produzir $N_C = 1$, $N_C = 4$ e $N_C = 10$ carros. Interprete os resultados que você obteve.

2. Agora faça um gráfico de L em função de N_C para $0 \leq N_C \leq 20$. A partir de quantos carros mensalmente vendidos a COLKS começa a ter lucro?

3. Analisando o gráfico, quantos carros a COLKS tem que produzir no mês para ter um lucro de cerca de 100 dubilas?

Vamos começar com o Item 1. Para obter o valor de L para $N_C = 1$, podemos digitar no Scilab

```
-->N_C = 1;
-->L = 50*N_C - 48*(N_C)^(0.9)-50
  L  =
   - 48.
```

Assim, quando produz apenas 1 carro, a Colks tem "lucro negativo", ou seja, um prejuízo de 48 dubilas.

Para $N_C = 4$ e $N_C = 10$

```
-->N_C = 4;
-->L = 50*N_C - 48*(N_C)^(0.9)-50
 L  =
- 17.145708
-->N_C = 10;
-->L = 50*N_C - 48*(N_C)^(0.9)-50
 L  =
  68.722447
```

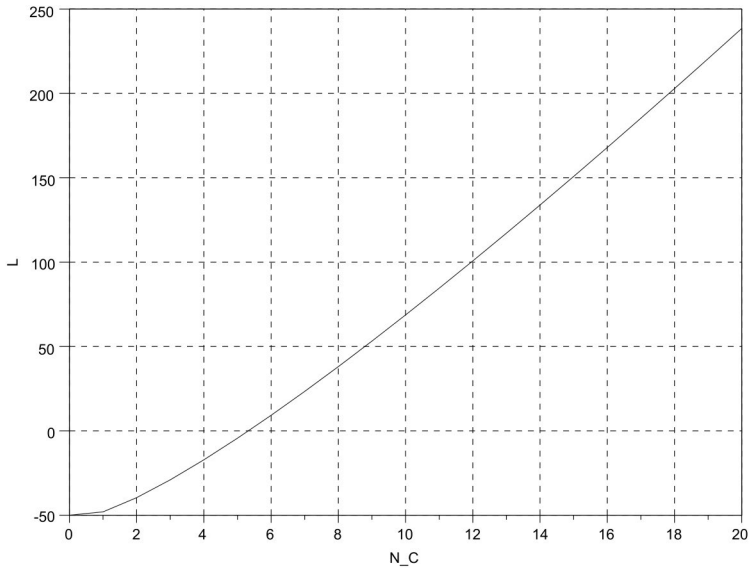
Vemos que para 4 carros vendidos a Colks ainda tem prejuízo. Porém, para $N_C = 10$, ela já tem lucro de cerca de 68 dubilas.

O Item 2 pede para fazermos um gráfico de L x N_C e obtermos a partir de quantos carros a Colks começa a ter lucro. Pelo Item 1 já sabemos que este número de carros deve estar entre 4 e 10. Façamos, então, o gráfico no Scilab. Como o número de carros vendidos N_C deve ser inteiro (afinal, ninguém vende 1/2 carro), vamos usar um passo unitário.

```
-->N_C = 0:1:20;
-->L = 50*N_C - 48*(N_C).^(0.9)-50;
-->plot(N_C,L);
-->set(gca(),"grid",[1 1])
-->xlabel('N_C');
-->ylabel('L');
```

Digitando-se estas linhas, deve-se obter o gráfico da Figura 2.10.

Figura 2.10: Gráfico obtido no item 2 da atividade 1.



Nesta figura, vemos que realmente o ponto em que L=0, raiz da função $L(N_C)$, está entre 4 e 10, mais precisamente um pouco acima de $N_C = 5$. Assim, concluímos que a COLKS começa a ter lucro quando vende pelo menos 6 carros. Caso necessário, use as ferramentas de ampliação (clique na lupa no canto esquerdo da tela).

O Item 3 fica fácil de ser resolvido com o gráfico da Figura 2.10. Para que $L \approx 100$, precisamos de $N_C = 12$ carros.

Encerramos assim esta primeira atividade.

Atividade 2: Um sistema de comunicação digital binário consiste em um transmissor e um receptor ligados por um meio físico (ar, cabo coaxial, fibra ótica, etc.) chamado de canal. O transmissor manda informação na forma de uma sequência de 0s e 1s, chamados de bits para o receptor. Devido às distorções e ao ruído inserido pelo canal, nem todo bit transmitido é recebido corretamente: na sequência de bits, alguns 0s viram 1s e alguns 1s viram 0s no receptor (Haykin, 2000).

Uma das formas de avaliar um sistema de comunicação digital é por meio da taxa de erro de bit, abreviada como BER (do inglês *Bit Error Rate*). Uma taxa de erro de 20%, por exemplo, quer dizer que, a cada 100 bits transmitidos, 20 chegam errados por causa de problemas na comunicação.

Quando se estudam sistemas de comunicação digital, *é muito comum* avaliar como a BER varia em função do ruído no canal. Esta quantidade de ruído é medida por uma grandeza denominada Relação Sinal-Ruído, abreviada por SNR (do inglês *Signal-to-Noise Ratio*). Quanto menor a SNR, mais ruído o ambiente adiciona ao sinal transmitido.

Pede-se:

1. Pelo descrito acima, espera-se que a BER seja uma função crescente ou decrescente da SNR?

2. Vamos supor que seja transmitida a seguinte sequência de bits: [0, 0, 1, 1, 1, 0, 1, 1, 0, 0]; mas que, devido aos problemas descritos acima, receba-se a seguinte sequência: [0, 0, 1, 0, 1, 0, 1, 1, 1, 0]. Quantos bits foram recebidos erroneamente? E, levando em conta estas sequências, qual a taxa de erro de bit (BER)?

Ainda nesta atividade, suponha que o sistema de comunicação binário apresente uma BER dada por (Haykin, 2000)

$$BER = \frac{1}{2} \, e^{-SNR/2} \tag{2.7}$$

3. Quanto a vale a BER para SNR=0. Interprete este valor.

4. Faça um gráfico da BER em função da SNR para valores de SNR entre 0 e 100. Comente sobre este gráfico. É possível visualizar bem os valores da BER?

5. Utilizando o gráfico que você fez no item anterior, responda: qual a SNR necessária para se chegar a uma taxa de erro de 10^{-5}?

6. Faça agora um gráfico de $log_{10}(BER)$ em função da SNR. Comente.

7. Repita o Item 5 usando este último gráfico. Comente!

Vamos começar com o Item 1. Conforme descrito no texto, conforme aumentamos a SNR, melhorarão as condições do canal. Assim, é de se esperar que o sistema tenha

menos erros (BER) nesta situação. Assim, esperamos que a BER seja uma função *decrescente* da SNR.

No Item 2 temos duas sequências de bits. Veja que elas diferem na quarta e na penúltima posição. Assim, temos dois bits recebidos erroneamente entre os 10 transmitidos e a BER será 20%.

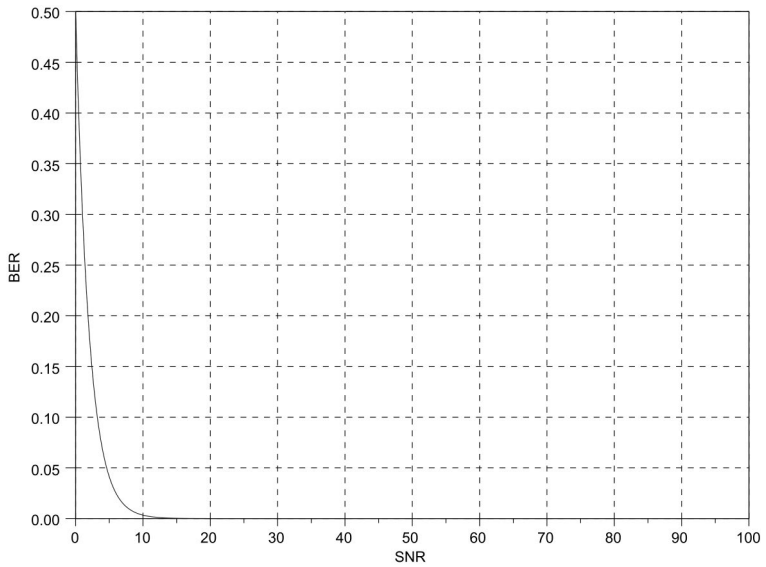
O Item 3 também é possível de ser feito diretamente. Veja que para *SNR*=0, temos $BER = \frac{1}{2} e^{0/2} = \frac{1}{2}$. Assim, quando *SNR*=0 a taxa de erros no receptor é 50%. Ou seja, ele acerta metade dos bits transmitidos, que é o pior que pode acontecer. Veja que a chance do receptor "adivinhar" qual foi o bit transmitido e acertar é exatamente 50%.

No Item 4 pede-se um gráfico da BER em função da SNR. Vamos aos comandos no Scilab:

```
SNR = 0:0.01:100;
BER = .5*exp(-SNR/2);
plot(SNR,BER);
set(gca(),"grid",[1 1])
xlabel('SNR');
ylabel('BER');
```

Você deve obter um gráfico como o da Figura 2.11.

Figura 2.11: Gráfico da BER em função da SNR usando escala linear nos dois eixos.



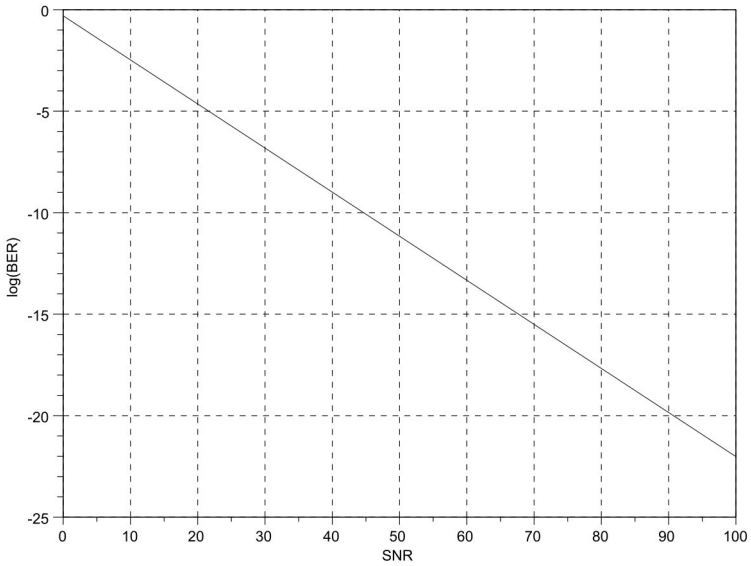
Fica difícil visualizar os valores de BER para uma *SNR*≥10. Eles ficam muito pequenos. Mesmo usando a ferramenta lupa, fica difícil analisar as taxas de erro para SNR elevadas. Assim, é impossível responder o Item 5.

Uma forma mais prática de colocar em gráfico os valores de BER e de qualquer grandeza física que assume valores muito pequenos ou muito grandes é usar a função log. Veja que $log_{10}(10^n) = n$. Assim, se $BER = 10^{-5}$, $log_{10} BER = -5$. Vamos então fazer o gráfico de $log_{10}(BER)$ em função da SNR.

```
SNR = 0:0.01:100;
BER = .5*exp(-SNR/2);
plot(SNR,log10(BER));
set(gca(),"grid",[1 1])
xlabel('SNR');
ylabel('log(BER)');
```

Obtemos o gráfico da Figura 2.12.

Figura 2.12: Gráfico do logaritmo da BER em função da SNR.



Agora fica bem mais fácil observar os valores da BER. Esta é uma técnica muito usada em todas as áreas da Ciência!

Então, para responder o Item 7, basta observar no gráfico qual valor da SNR leva a $log_{10}BER = -5$. Este valor está em torno de $SNR=22$.

Na seção 2.6 propomos vários outros exercícios para você treinar a produção computacional de gráficos e a sua interpretação.

2.5 CONSIDERAÇÕES FINAIS

Este capítulo estudou como funções podem ser usadas para representar fenômenos da natureza. Em especial, o uso de ferramentas computacionais voltadas a cálculos científicos permitiu fazer a representação gráfica de funções. Tal representação, por sua vez, constituiu uma forma valiosa para se visualizar o comportamento de fenômenos, permitindo compreendê-los melhor e extrair informações deles. Por exemplo, na Atividade 1, o gráfico possibilitou responder perguntas como: A partir de quantos carros se começa a ter lucro? Quantos carros deve-se produzir para se ter um dado lucro? Além disto, o gráfico também permitiu compreender como é o comportamento das

variáveis envolvidas. Por exemplo, ainda na Atividade 1, observando-se a expressão (2.6), a princípio, não é possível saber como é o comportamento do lucro (L) em função do número de carros produzidos (N_C). Por outro lado, a curva $L \times N_C$ evidencia que o lucro é sempre crescente, tornando-se positivo a partir de certo número de carros. Ou seja, em alguns casos, um gráfico é uma boa maneira de representar um dado fenômeno, especialmente quando se quer enfatizar o comportamento das variáveis envolvidas. A despeito disto, há casos em que o comportamento de variáveis não pode ser representado por meio de funções determinísticas. Para descrever fenômenos aleatórios, podem-se empregar métodos estatísticos, como será visto no próximo capítulo.

2.6 EXERCÍCIOS

Esta seção contém exercícios básicos (Exercícios 1 a 7) que visam à fixação do conteúdo visto em aula e exercícios mais avançados (Exercício 8 em diante) que podem exigir conhecimento adicional. Estes últimos não são essenciais para o bom aproveitamento no curso, contudo, foram incluídos como desafio para o aluno interessado.

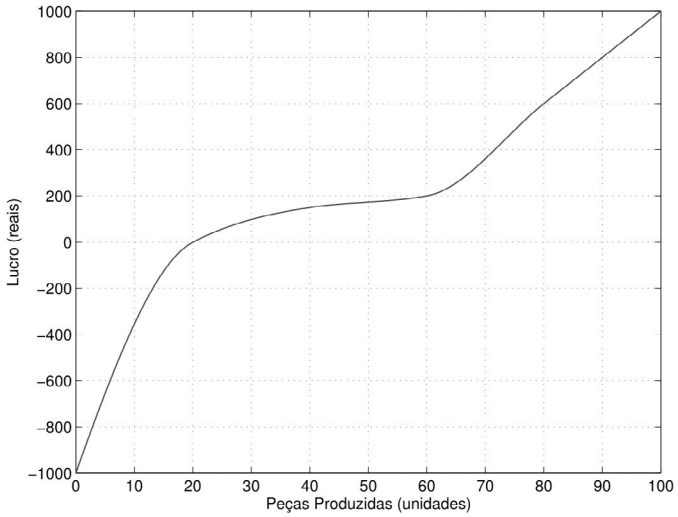
1. Um certo sistema de comunicação binário, em determinada situação, apresenta uma Taxa de Erro de Bit (*BER*) em função da Relação Sinal-Ruído (*SNR*): $BER = \frac{1}{2} e^{-2SNR}$. Seguindo o exemplo da Atividade 2, vista em aula, pede-se:

a) Faça um gráfico de $log(BER)$ em função da *SNR*.

b) Qual sistema apresenta melhor desempenho, em termos de taxa erro: o da atividade 2 ou o apresentado neste exercício?

2. O gráfico na Figura 2.13 mostra o lucro em reais, obtido por uma empresa em função do número de peças produzido por ela em um mês.

Figura 2.13: Gráfico usado no Exercício 2.



A partir deste gráfico, responda as seguintes perguntas:

a) Qual o número mínimo de peças que a empresa precisa produzir mensalmente para que não tenha prejuízo?

b) Se a empresa visa ter um lucro de R$ 10000,00 anuais, quantas peças ela precisa produzir mensalmente?

c) Qual o lucro mensal por peça produzida pela empresa quando ela produz 60 peças? E quando ela produz 20 peças? Tente representar graficamente estas taxas.

3. Usando um gráfico, localize as raízes das seguintes funções:

a) $f(x) = e^{-x} - x$

b) $f(x) = sen(10x) - cos(3x)$

c) $f(x) = x - cos(x)$

4. (Relacionado ao Exemplo 4 de uma lâmpada ligada a um gerador) Conforme visto no texto, o ponto de cruzamento do gráfico de duas funções pode ser usado para encontrar de maneira aproximada a solução simultânea de um conjunto de equações. Seja o sistema de equações

$$-2x+y = 1$$

$$3x+2y = 4$$

que pode ser expresso na forma matricial por

$$\begin{pmatrix} -2 & 1 \\ 3 & 2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 \\ 4 \end{pmatrix} \rightarrow A.sol = b$$

Usando a linha de comando no Scilab, pede-se:

a) Encontre a solução exata pelas matrizes:

```
-->A=[-2 1;3 2];b=[1; 4];
-->sol=A\b;
-->xsol=sol(1);
-->ysol=... //complete aqui com o valor
            //de y (ordenada).
```

Repita o cálculo usando a matriz inversa:

```
-->B=inv(A);
-->sol2=B*b;
```

b) Defina o intervalo de interesse para se fazer o gráfico. Por exemplo, tome 101 pontos, desde 0.8*xsol até 1.2*xsol:

```
-->0.8*xsol:(1.2-0.8)*xsol/100:1.2*xsol;
```

c) Faça o gráfico de y1 obtido pela primeira equação e de y2 obtido da segunda equação e verifique que o ponto "sol" é dado pela intersecção de duas retas. Dica:

```
-->y1=1+2*x;
```

```
-->y2=... // complete aqui
-->plot(x,y1);
-->plot(...) //complete aqui
-->ysol=... //complete aqui
-->plot(xsol,ysol,'ro'); // ponto da solucao
-->xlabel(...);ylabel(...);xtitle(...);
  // complete acima com dados do grafico.
```

d) Repita este exercício para duas funções quaisquer de sua escolha, de preferência não lineares.

5. Faça o gráfico de:

$$g(t) = \begin{cases} 0, \, t \le -2 \\ -4 - 2t, \quad -2 < t \le 0 \\ -4 - 3t, \quad 0 < t \le 4 \\ 16 - 2t, \quad 4 < t \le 8 \\ 0, \, t > 8 \end{cases}$$

6. Faça o gráfico das funções abaixo e compare os resultados obtidos:

a) $f(t) = e^{-t/2}$, para $0 < t < 2$.

b) $f(t) = e^{t/2}$, para $0 < t < 2$.

c) $f(t) = e^{(t-1)/2}$, para $1 < t < 3$.

7. Faça o gráfico de $f(x)=|x-2|+|2x+1|-x-6$. Para que valores de x, tem-se $f(x)>2x+2$?

8. Qual deve ser o valor de *f* após a execução das seguintes instruções:

a)
```
-->t=3;
-->f=sin(t);
```

b)
```
-->x=1:5;
-->f=cos(%pi*x);
```

c)
```
-->x=-1:0.5:1;
-->f=1./(x.^2+2);
```

9. Seja um sistema de coordenadas (x,y). Um círculo *C* pode ser definido analiticamente como o conjunto de pontos que obedecem a equação

$$(x-a)^2 + (x-b)^2 = r^2,$$

em que *r* é igual ao raio e (a,b) são as coordenadas do centro.

Usando gráfico de funções, um círculo pode ser desenhado em duas etapas:

1º fazer o gráfico da parte positiva: $y_+ = \sqrt{r^2 - (x-a)^2}$;

2º fazer o gráfico da parte negativa: $y\_ = -\sqrt{r^2 - (x-a)^2}$.

Nas duas etapas, a rigor, deve-se considerar x em um intervalo que produza y real, ou seja, $x \in [\, a - r,\, a + r]$. Contudo, na resolução deste exercício, você pode tomar $x \in \mathbb{R}$ num intervalo suficientemente grande e considerar apenas a parte real de y. Empregando este método, pede-se:

a) Desenhe, num mesmo gráfico, o círculo C_1 definido pela equação $x^2+y^2 = 1$ e o círculo C_2 definido por $(x-1)^2 + y^2 = 2$.

b) Encontre as coordenadas dos pontos de intersecção dos círculos. Opcional: o resultado pode ser verificado manualmente usando régua e compasso.

c) Repita os itens anteriores para C_1: $x^2+y^2 = 1$ e C_2: $(x-1)^2 + (y-2.1973)^2 = 2$.

10. a) Usando o método gráfico, determine o valor de x que anula o determinante da matriz

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & x & 5 \\ 6 & x & x \end{bmatrix}$$

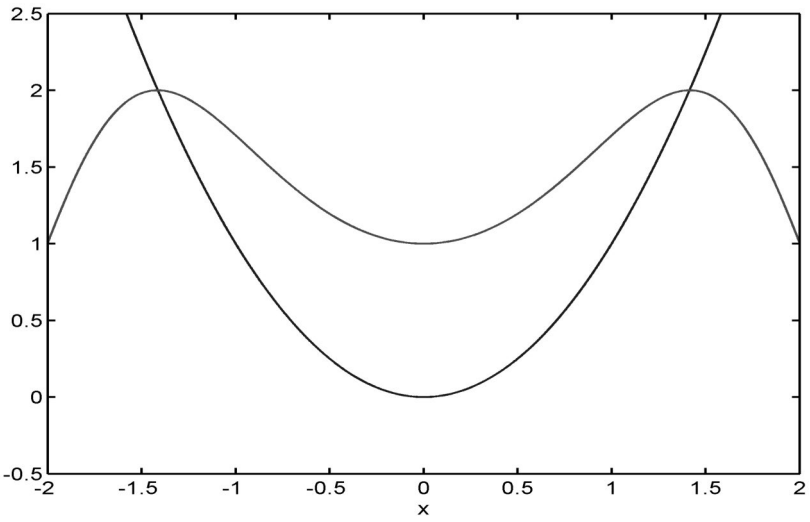
b) Verifique o resultado calculando a resposta por método algébrico.

11. As raízes de $x^2 - \sin(0.784x^2) - 1$ podem ser encontradas graficamente. Com esta finalidade, pede-se:

a) reproduza o gráfico da Figura 2.14 e

b) usando o seu gráfico, encontre as duas raízes da equação.

Figura 2.14: Gráficos usados no Exercício 11.



12. No Exercício 9, você viu como desenhar um círculo usando duas funções, sendo empregada uma função para cada metade do círculo. Uma maneira alternativa de definir o conjunto de pontos que desenham o círculo consiste em empregar o par de senoides

$$x = r \sin(t - \pi/2)$$

$$y = r \sin(t)$$

nas quais r é igual ao raio e t é um parâmetro. Para se obterem os pontos (x, y) do círculo, o parâmetro t deve ser variado no intervalo $[0,2\pi]$.

De maneira mais geral, outras figuras podem ser desenhadas usando o par de senoides

$$x = A_x \sin(\omega_x t - \theta_x)$$

$$y = A_y \sin(\omega_y t - \theta_y),$$

nas quais A_x, A_y, ω_x, ω_y, θ_x e θ_y são constantes fixadas e t é o parâmetro que deve ser variado a fim de desenhar a figura. Pede-se:

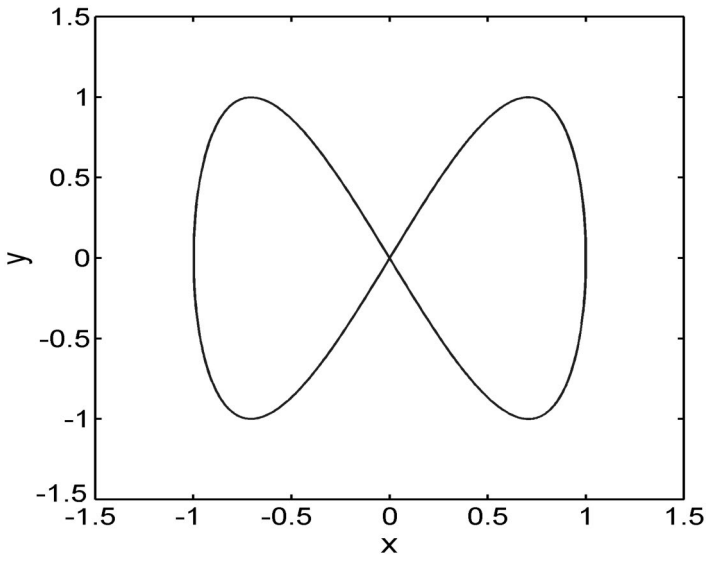
a) Usando este procedimento, desenhar um círculo de raio unitário;

b) No item anterior, use um ângulo θ no lugar de $\pi/2$. Experimente modificar θ e verifique o que ocorre com a figura. A partir desta figura é possível determinar a defasagem entre as senoides? Como?

c) Empregar $A_x = A_y = 1$, $\omega_x = 1$, $\omega_y = 2$, $\theta_x = \theta_y = 0$ e reproduzir a Figura 2.15.

d) No item anterior, experimentar $\omega_x = 1$ e, em seguida, $\omega_y = 4$. A partir da figura é possível determinar a relação harmônica entre o par de senoides?

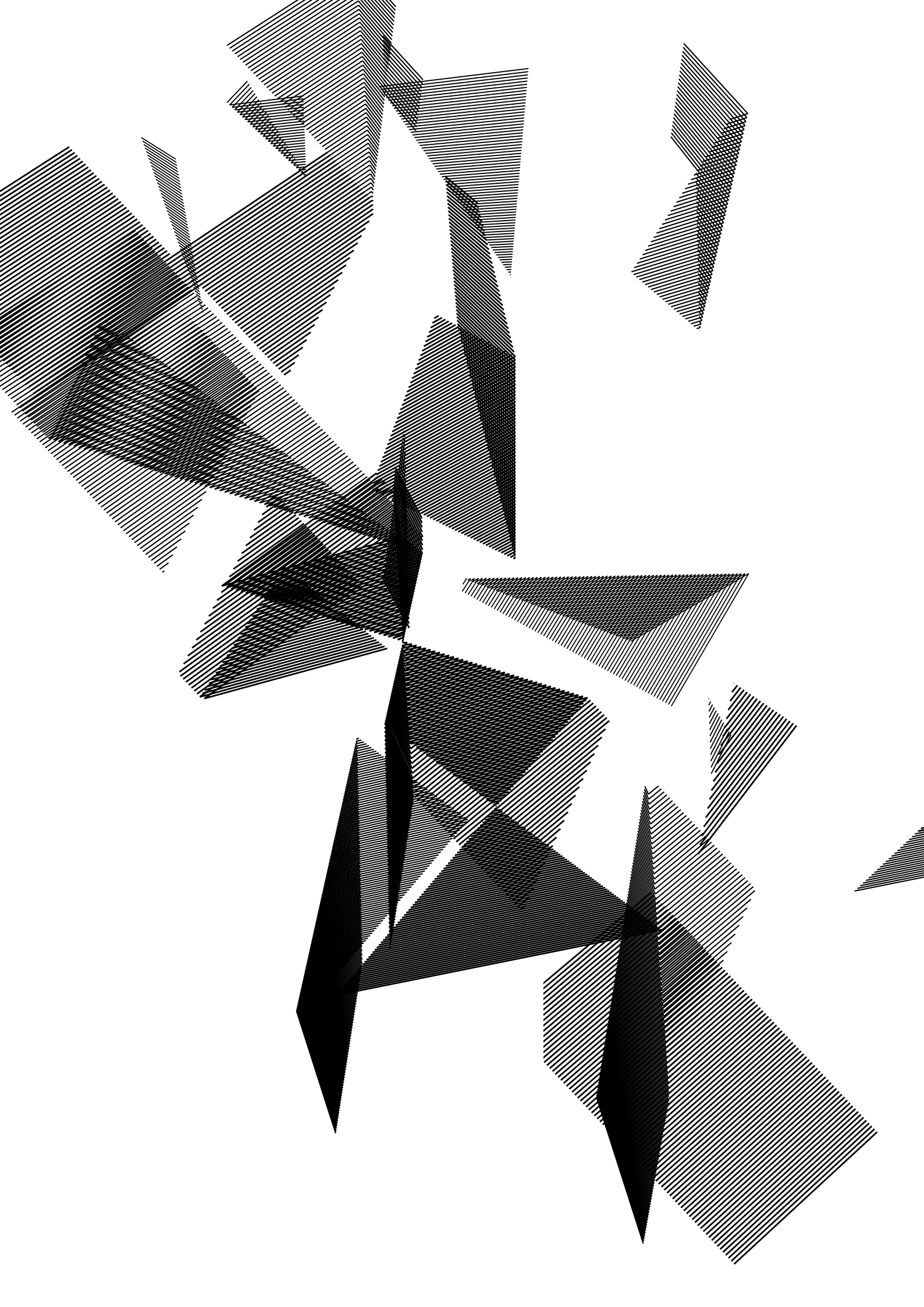
Figura 2.15: Figura do Item c, Exercício 12.



REFERÊNCIAS BIBLIOGRÁFICAS

CHAPRA, S. e CANALE, R. (2008), *Métodos Numéricos para Engenharia,* 5th ed.: McGraw Hill.

HAYKIN, S. S. (2000). *Communication systems*, 4th ed.,: Wiley.



CAPÍTULO 3

NOÇÕES DE ESTATÍSTICA, CORRELAÇÃO E REGRESSÃO

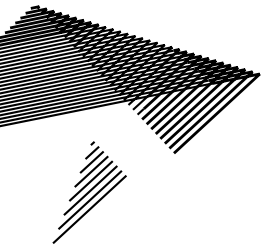
Carlos da Silva dos Santos
Cristiane Otero Reis Salum
Delmo Alves de Moura
Peter Maurice Erna Claessens

3.1 INTRODUÇÃO

Em diversos campos do conhecimento, são comuns as situações em que uma decisão deve ser tomada com base em informações parciais. Dados obtidos a partir de um número relativamente pequeno de exemplos são usados para prever comportamento em cenários ainda não observados, conforme ilustrados pelos seguintes exemplos:

• Uma montadora prepara um novo modelo de automóvel. Para determinar se o modelo é seguro, são realizados ensaios de impacto (*crash tests*) com protótipos, em que são medidos parâmetros de deformação, aceleração, força de impacto sobre passageiros, etc. Esse procedimento supõe que os protótipos iniciais são representativos do comportamento dos automóveis fabricados futuramente e, portanto, as consequências de um acidente podem ser corretamente avaliadas.

• Um paciente é internado após um ataque cardíaco; os procedimentos de cuidado envolvem estimar o risco de um novo ataque, considerando dados do prontuário do paciente como: medidas clínicas, histórico familiar de doenças cardíacas, dieta, etc. O risco é avaliado a partir de uma série histórica de internações de pacientes semelhantes.



• O objetivo de uma pesquisa eleitoral é fornecer um retrato momentâneo da intenção de voto em cada candidato. Em uma pesquisa, apenas um pequeno contingente do eleitorado é entrevistado. Por isso, é impossível dizer que a verdadeira intenção de voto em um candidato é igual ao percentual obtido por ele na pesquisa. Os números de intenção são divulgados juntamente com a *margem de erro* da pesquisa, indicando um intervalo que provavelmente contém a intenção real de voto em cada candidato.

• Estudos do efeito de novos medicamentos usualmente empregam dois grupos de pacientes: um grupo recebe o medicamento em questão, enquanto o segundo (denominado grupo de controle) recebe um composto sem ação nenhuma (placebo). O objetivo de cada estudo é determinar se o medicamento tem um efeito significativo na melhora dos pacientes, em comparação com o placebo.

O ponto comum a todos os exemplos anteriores é a *incerteza* resultante de trabalharmos com um subconjunto dos dados de interesse. Esse subconjunto é chamado de *amostra*, enquanto o conjunto total é chamado de *população*. Em geral, o processo de gerar uma amostra é aleatório, então, se gerarmos duas amostras distintas para estudar um mesmo processo, provavelmente vamos obter dois resultados diferentes. Em princípio, não podemos dizer que um desses resultados é "mais verdadeiro" que o outro. Para cumprir nosso objetivo de estudar a população por meio da amostra, precisamos reconhecer essa variabilidade e incluir um grau de *incerteza* em nossas conclusões, pois não é possível verificar se o resultado obtido com a amostra é igual àquele que obteríamos com a população inteira. A Estatística é um ramo da Matemática que estuda esse tipo de problema: como podemos usar uma amostra para tirar conclusões sobre um universo maior de objetos, levando em conta que sempre há variação e incerteza nas nossas medidas.

A Estatística está presente na base de toda a ciência experimental, pois ela fornece diretrizes para a coleta de dados, nos permite comparar diferentes hipóteses e avaliar a precisão dos resultados que obtemos experimentalmente. A Estatística moderna emprega amplamente ferramentas computacionais. Hoje em dia, ferramentas de *software* de baixo custo ou mesmo gratuitas disponibilizam métodos sofisticados de análise estatística a qualquer um que tenha um computador pessoal. Ao mesmo tempo, avanços recentes da Computação tornaram viável a análise estatística de volumes gigantescos de dados, como aqueles produzidos por experimentos de bioinformática, física de alta energia ou por sítios da Internet com grande número de usuários (como Google, Facebook, Twitter). A compreensão dos métodos estatísticos, de seus cenários de aplicação e limitações, bem como o domínio de ferramentas computacionais de análise, é fundamental para a prática da ciência.

O objetivo deste capítulo é introduzir, de maneira informal, algumas ferramentas básicas de análise estatística, que permitem visualizar e compreender características de dados experimentais e realizar formas simples de inferência. O capítulo contém uma série de exercícios realizados com uma ferramenta computacional (LibreOffice Calc) que familiarizam o leitor com uso de tal tipo de ferramenta para automatizar tarefas de análise estatística que seriam por demais tediosas ou mesmo impossíveis de se realizar manualmente. Este capítulo não pretende esgotar o assunto da análise estatística de dados. Aspectos mais técnicos dos métodos apresentados, suas limitações e aplicabilidade devem ser vistos em um curso posterior de Estatística.

O restante deste capítulo está distribuído da seguinte forma. A Seção 3.2 trata da relação entre Pesquisa Científica e Estatística e em seguida introduz alguns conceitos estatísticos básicos, como distribuição de frequência, medidas de tendência central e dispersão e visualização de dados. A Seção 3.3 introduz os conceitos de correlação e regressão, duas ferramentas para estudar a presença de relações entre medidas estatísticas. Na Seção 3.4 são apresentados exercícios para prática em sala de aula. A Seção 3.5 traz as considerações finais do capítulo. Uma série de exercícios complementares é apresentada na Seção 3.6.

3.2 CONCEITOS BÁSICOS

Para entender melhor como a Estatística influencia um estudo científico, vamos examinar os estágios que constituem esse processo. O termo *Pesquisa Científica* refere-se a um processo de aprendizagem em que o cientista determina o objetivo de uma investigação, coleta as informações relevantes, analisa os dados, tira as conclusões e decide sobre os próximos planos com base nestas conclusões. Didaticamente, podemos dividir o processo de pesquisa nas seguintes etapas:

- Planejamento: nesta etapa, levanta-se uma *hipótese testável* sobre o problema que se deseja estudar;
- Delineamento: consiste em determinar a metodologia (instrumentos) mais adequada para se obter os dados requeridos;
- Execução: trata da coleta sistemática dos dados para que se tenha controle do maior número de variáveis que influenciam o problema em questão;
- Processamento e Análise: consiste em analisar os dados com base nas hipóteses iniciais;
- Interpretação: os resultados são interpretados, dando origem às conclusões do estudo.
- Publicação: em que são divulgados os resultados e conclusões.

Anteriormente, nós usamos os termos *população* e *amostra* para ilustrar a necessidade de métodos estatísticos. Agora serão fornecidas definições mais precisas desses dois termos.

- População: é o grupo correspondente a uma coleção completa de unidades para as quais serão feitas inferências. Representa o alvo da investigação.
- Amostra: é definida como subconjunto do universo ou da população, por meio do qual se estabelecem ou se estimam as características desta população.

Uma amostra pode ser constituída, por exemplo, por 100 funcionários que fazem parte da população de 1.700 que trabalham em uma empresa. Outro exemplo de amostra pode ser dado por um determinado número de centros de saúde que compõem a rede de saúde básica estadual. O processo de gerar uma amostra para entender um processo pode ser comparado a olhar o mundo através de uma janela, que limita nosso campo de visão e pode introduzir uma distorção. Para que possamos fazer inferências válidas sobre uma população, é necessário que a amostra seja *representativa*, ou seja, escolhida de maneira aleatória e contenha um número adequado de sujeitos. Um processo descuidado de escolha pode resultar em uma amostra tendenciosa ("enviesada"). Isso aconteceria, por exemplo, caso selecionássemos os 100 funcionários de maior salário da empresa, ou escolhêssemos todos os centros de saúde em um mesmo bairro. Em cada um desses dois casos, certamente teríamos um retrato que não reflete as condições da população.

Uma vez determinada uma amostra, o passo seguinte de um estudo é a medição de certas características de interesse para cada um dos casos presentes na amostra. Essas características medidas são chamadas de *variáveis*. Por exemplo, em um estudo sobre

habitantes de uma cidade, as variáveis podem ser: altura, sexo, cor do cabelo, cor dos olhos, idade, peso, expectativa de vida, preferência por um partido político, etc. Nós dividimos as variáveis em dois tipos, em função do papel que desempenham no estudo. Chamamos de variável *dependente* a medida de interesse da pesquisa; deve variar em resposta a alguma outra variável manipulada (intervenção). Do mesmo modo, chamamos de variável *independente* aquela que sofre uma intervenção, que está sendo manipulada, e/ou que supostamente exerce uma influência sobre a variável de resposta. Por exemplo, em um estudo sobre incidência de depressão em adolescentes de diversos países, a medida do estado de depressão é a variável dependente. Outras medidas como país de origem, idade, situação familiar são possíveis variáveis independentes para este caso.

Após a coleta dos dados, quando as variáveis de interesse já foram medidas para todos os casos da amostra, inicia-se a etapa de análise estatística. Esta se caracteriza pelo cálculo de valores a partir das variáveis medidas, que nos permitem entender o comportamento dos dados e fazer previsões sobre casos futuros. Às vezes, esses valores são também chamados de estatísticas.

Podemos dividir o trabalho de análise em duas áreas, de acordo com sua finalidade. A *Estatística Descritiva* é a área da Estatística que se preocupa com a apresentação, organização e resumo dos dados. Por exemplo, o cálculo de uma média de uma amostra ou a criação de um gráfico para visualizar dados observados são atividades que fazem parte desta área. Já a *Estatística Inferencial* é a área que estuda métodos para generalizar um resultado obtido de uma amostra de dados para um grande número de sujeitos (população). A estimação de um intervalo plausível para uma média em uma população com uma certa variabilidade, por exemplo, é uma inferência estatística, conforme procedimentos desenvolvidos nesta área.

Em estudos baseados em dados amostrais, um dos papéis dos métodos estatísticos é delinear o tamanho da amostra e orientar o mecanismo de escolha dos casos, para que as observações ofereçam uma base para gerar conclusões válidas. Adicionalmente, a Estatística fornece a metodologia para se fazer inferências sobre uma população através de uma amostra de dados coletada e analisada. Por fim, os métodos estatísticos estabelecem as condições em que os resultados de um estudo com número diminuto de casos podem ser generalizados para o restante da população.

3.2.1 TIPOS DE DADOS

A identificação da natureza dos dados é de extrema importância para uma escolha correta do método estatístico de análise. Para efeito de classificação, podemos dividir os dados em dois tipos: *Categóricos* (ou qualitativos) ou *Numéricos* (ou quantitativos).

Os dados categóricos podem ainda ser subdivididos em mais tipos:

- Nominal: constituído pelas variáveis com categorias nomeadas, entre as quais não há implicação de ordem. Os dados são simplesmente rotulados por nomes ou números com o propósito de agrupar os sujeitos que possuam características semelhantes em determinadas categorias. Ex.: sexo; estado civil; grupo sanguíneo; cor dos olhos; nacionalidade.

• Ordinal: formado por variáveis cujos valores fornecem informações sobre a ordenação das categorias, mas sem indicar a magnitude das diferenças entre os valores. Em outras palavras, usando uma variável ordinal, podemos verificar para cada caso se tem valor maior, igual ou menor quando comparado a outro caso; é possível determinar a *ordem* dos casos. No entanto, não podemos dizer *o quanto* um caso é maior ou menor que outro. Exemplos: nível sócio-econômico (baixo, médio e alto); avaliação de um estudante (insuficiente, suficiente, excelente).

Já os dados numéricos se dividem em dois casos:

• Contínuo: as variáveis podem assumir qualquer valor dentro de um intervalo. Uma variável é contínua se não houver lacunas entre as observações, isto é, entre quaisquer dois valores potencialmente observáveis, há sempre outro valor potencialmente observável. Exemplo: A altura de um indivíduo pode corresponder a qualquer número entre 1,65 m e 1,78 m; como 1,65009 m ou 1,65699 m.

• Discreto: neste caso os dados podem assumir apenas determinados valores numéricos. A variável será discreta se houver lacunas entre as observações. Por exemplo, o número de crianças que apresentaram TOC (Transtorno Obsessivo Compulsivo) entre os cinco e os dez anos corresponde a um número N que pode assumir valores tais como 0, 1, 2, 3, 4... mas não pode ser 2,5 ou 4,876. Exemplos: número de nascimentos ou mortes; número de carros; etc.

Em geral, as medições dão origem a dados contínuos enquanto as contagens ou enumerações resultam em dados discretos.

Variáveis quantitativas são ainda caracterizadas pelo nível de escala no qual são expressas, o que se refere a como os números da escala se relacionam. Quando uma distância igual entre os números significa uma distância igual entre o que a variável mede, dizemos que a escala é métrica. Temos as seguintes escalas métricas:

• Escala intervalar: neste caso, os dados podem ser classificados em categorias ordenadas e a distância (ou diferença) entre elas é constante (igual). Portanto, as posições estão dispostas através de maior, igual ou menor, como também os intervalos entre os valores têm valor igual. Exemplo: Teste de inteligência (avaliação do QI). Intervalo entre QI 100 e 110 é o mesmo que entre QI 120 e 130. Porém, não se pode dizer que uma pessoa com QI 100 tem o dobro do QI de um indivíduo com QI 50. Ou: A distância em temperatura entre 15°C e 20°C é a mesma como entre 20°C e 25°C, mas é difícil sustentar que uma temperatura equivalente a 20°C é 'o dobro de' uma temperatura equivalente a 10°C: com as mesmas temperaturas expressas em Fahrenheit, o que dá 50 e 68, a razão não vale mais. A variável de escala intervalar adota um ponto zero que é convencional e arbitrário. Portanto, a multiplicação e a divisão não têm sentido.

• Escala de razão: neste caso, também temos intervalos iguais entre as categorias. No entanto, diferentemente dos casos anteriores, o ponto zero é significativo. Somente quando trabalhamos com uma variável de razão é que podemos afirmar: "O corpo A é duas vezes mais pesado do que B". Exemplos: comparação de peso corporal, notas de zero a 10.

3.2.2 DISTRIBUIÇÃO DE FREQUÊNCIAS

Após o levantamento de dados, torna-se necessária a descrição e a organização destes dados. Este processo determina o número de constituintes em cada uma das categorias que se originam ao se classificar uma população de acordo com os itens requeridos no levantamento. Para que seja mais fácil entender os dados brutos (da forma como são coletados) pode-se construir uma *distribuição de frequências*. A *frequência* é definida como o número de indivíduos pertencentes a cada categoria.

Exemplo 1: Em um estudo feito com 100 estudantes norte-americanos foi perguntado qual o curso de que eles menos gostaram durante a faculdade. A distribuição de frequências desse estudo é mostrada na Tabela 3.1.

Tabela 3.1 Número de estudantes que apontaram cada curso como aquele de que gostaram menos.

Curso	**Número de estudantes**
Economia	42
Sociologia	25
História	8
Psicologia	13
Cálculo	12

Exemplo 2: A distribuição dos alunos do curso de Estatística Aplicada à Psicologia é mostrada na Tabela 3.2.

Tabela 3.2 Distribuição dos alunos do curso de estatística aplicada à psicologia.

Sexo	**Frequência**
Masculino	1
Feminino	30
Total	31

A partir dos exemplos anteriores, poderíamos perguntar: qual das matérias foi menos apreciada pelos alunos? O curso de Economia foi realmente menos apreciado pelos alunos ou a diferença foi apenas casual? Para que os resultados se tornem mais claros e para responder essas perguntas, podemos utilizar alguns métodos para padronizar tamanhos e possibilitar a comparação de distribuições.

Com uma *proporção* comparamos o número de sujeitos de uma dada categoria com o total de sujeitos que compõem a distribuição. Portanto, para obtermos a proporção *P*, dividimos a frequência de sujeitos de uma dada categoria *f* pelo número total de sujeitos *N* :

$$P = \frac{f}{N}$$

Desta forma, para o exemplo 2, obtemos as seguintes proporções de alunos desta classe em relação ao sexo:

- sexo masculino: $P_m = 1/31 = 0{,}03$
- sexo feminino: $P_f = 30/31 = 0{,}97$

Por causa da lógica do cálculo da proporção, também é chamada de *frequência relativa*.

Em uma *porcentagem*, multiplicamos uma dada proporção por 100. Para o mesmo exemplo 2 anterior, obtemos:

- sexo masculino: 3%
- sexo feminino: 97%

Da mesma forma, podemos repetir o processo para a Tabela 3.1, obtendo a distribuição mostrada na Tabela 3.3.

Tabela 3.3 Porcentagem de alunos que apontou cada curso como seu menos preferido.

Curso	% Alunos
Economia	42%
Sociologia	25%
História	8%
Psicologia	13%
Cálculo	12%

A *taxa* é ainda uma outra maneira para relatar dados relativos, usada para comparações entre o número efetivo e o número potencial de sujeitos ou ocorrências. Exemplo: Taxa de criminalidade envolvendo adolescentes entre 10 e 17 anos. Considerando f_e o número efetivo de crimes de uma cidade e f_p o número potencial de crimes que envolveram adolescentes, teremos:

- $f_e = 300$ crimes envolvendo adolescentes
- $f_p = 1500$ crimes ocorridos

Muitas vezes, as taxas são dadas em termos de uma base de 1000 casos potenciais**,** conforme indicado na Equação 3.1:

$$\text{Taxa} = 1000\,\frac{300}{1500} = 200 \tag{3.1}$$

Desta forma, podemos dizer que em cada 1000 crimes ocorridos na cidade, 200 envolvem adolescentes. Além disso, pode ser feita a comparação entre as taxas de duas cidades. Em um outro exemplo, 25,2 em 100.000 é a taxa de homicídio registrado no Brasil: em média, 25,2 de cada 100.000 habitantes foram vítimas de homicídio intencional.

Na *razão*, comparamos diretamente o número de sujeitos que se enquadra em uma categoria com o número de sujeitos de outra categoria. A Equação 3.2 apresenta este conceito:

$$R = \frac{f_1}{f_2} \tag{3.2}$$

onde f_1 é a frequência de sujeitos da primeira categoria e f_2 é a frequência de sujeitos da segunda categoria.

Portanto, a razão entre os alunos que responderam Economia e os alunos que responderam História está indicada na Equação 3.3:

$$R = \frac{42}{8} = \frac{21}{4} \tag{3.3}$$

Simplificando, podemos dizer que, para cada 21 alunos que responderam Economia, 4 responderam História.

Frequências com dados agrupados

Em alguns casos, a formação de categorias não é imediata como, por exemplo, para variáveis como idade, peso, altura. Nesse caso, nós criamos categorias dividindo a faixa numérica ocupada pela variável, observando-se os seguintes aspectos:

- Valores reunidos em uma mesma classe passam a assumir o valor médio do intervalo de classe;
- As classes devem ser mutuamente exclusivas;
- É conveniente que as classes tenham a mesma amplitude.

Exemplo: A distribuição das idades dos alunos em uma sala de aula. Deveremos prosseguir da seguinte maneira:

1. Colocar a lista em ordem crescente ou decrescente;
2. Verificar a amplitude da lista (o valor maior menos o valor menor);
3. Determinar uma dimensão de intervalo adequada para o problema (esta deve gerar de 10 a 20 intervalos);
4. Redefinir a lista que consiste destes intervalos, da frequência relativa e do ponto médio.

Vamos introduzir a seguinte notação para intervalos:

- 18 | – 22 inclui os alunos com 18, 19, 20 e 21 anos;
- 18 | – | 22 (ou 18 – 22) inclui os alunos com 18 até 22 anos.

O *ponto médio* de um intervalo é definido como a média dos dois extremos. Portanto, o ponto médio do último intervalo é (18 + 22)/2 = 20.

Um conceito útil é o de *frequência cumulativa*, definida como o número total de sujeitos pertencentes a uma determinada categoria ou número total de sujeitos pertencentes a categorias inferiores à categoria que se está analisando. A frequência cumulativa ou acumulada é obtida através da soma da frequência daquela categoria com a frequência total de todas as categorias que estão abaixo dela. Para se obter a *porcentagem acumulada* ou *frequência cumulativa relativa*, utiliza-se a Equação 3.4,

$$c(\%) = 100\ \frac{f_a}{N} \tag{3.4}$$

onde f_a é a frequência acumulada e N é o número total de sujeitos.
A Tabela 3.4 mostra um exemplo de cálculo de frequência acumulada, dividindo-se os casos do estudo em faixas de acordo com a idade.

Tabela 3.4 Um exemplo de cálculo de frequência acumulada.

Intervalos	Frequência Simples	Ponto Médio	Frequência Relativa	Frequência Acumulada	Freq. Acum. Relativa
16 | – | 20	12	18	39%	12	39%
21 | – | 25	10	23	32%	22	71%
26 | – | 30	7	28	23%	29	94%
31 | – | 35	2	33	6%	31	100%
Total	31				

3.2.3 MEDIDAS DE TENDÊNCIA CENTRAL

É conveniente dispor de medidas que informem sobre a amostra de maneira mais resumida do que os dados brutos são capazes de fazer. As medidas de tendência central cumprem este papel, dando o valor do ponto em torno do qual os dados se distribuem. São medidas de tendência central, por exemplo, a *média*, a *mediana* e a *moda*. O objetivo destas medidas é representar um valor típico de um conjunto de dados.

A *média aritmética* ou simplesmente *média* é uma medida de tendência central utilizada para dados numéricos, dados intervalares ou do tipo razão. A média é definida como a soma de todos os valores de uma variável aleatória (função real associada aos elementos do espaço amostral considerado) em um conjunto, dividida pelo número de elementos do conjunto. O cálculo da média é expresso pela Equação 3.5,

$$\overline{X} = \frac{\sum_{i=1}^{N} X_i}{N} \tag{3.5}$$

onde **X** é a variável aleatória cuja média está sendo calculada. O símbolo $\overline{X}$ refere-se à média de **X**. O símbolo **Σ** significa somatória. Denotamos por X_i o *i*-ésimo valor de **X** encontrado no conjunto de dados. E N é o número total de sujeitos do conjunto.

Exemplo: Calcular a média da classe cujas notas são dadas pela Tabela 3.5. Nesse caso, a somatória das notas é 60 e o número total de sujeitos é 11. Logo,

$$\overline{X} = 60/11 = 5{,}45$$

Tabela 3.5 Tabela de notas de uma classe.

Sujeito	Nota
1	3,0
2	5,0
3	6,0
4	4,0
5	5,0
6	8,0
7	9,0
8	6,0
9	2,0
10	5,0
11	7,0
Total	60

A *mediana* é o valor central de um conjunto, que divide a distribuição em duas partes iguais (mesmo número de exemplos abaixo e acima do valor da mediana). É uma medida de tendência central utilizada para dados numéricos e dados ordinais. Para calcular a mediana, devemos ordenar os dados. A mediana é definida então como o elemento X_i, em que o índice i é dado por $i = (N + 1)/2$ para N ímpar. No caso de um número par de sujeitos, a mediana será a média entre os dois valores centrais.

Exemplo: Calcular a mediana dos valores da Tabela 3.6. Nesse caso, temos $i = (11 + 1)/2 = 6$. Devemos então tomar como mediana o sexto elemento do conjunto, em ordem. Logo, mediana (**X**) = 5.

Tabela 3.6 Tabela de dados para exemplo de cálculo da mediana.

Categorias	Frequência Simples	Frequência Acumulada
2	1	1
3	1	2
4	1	3
5	3	6
6	2	8
7	1	9
8	1	10
9	1	11

Exemplo: para a sequência de valores {62, 54, 82, 49, 75, 64}, calcular a mediana. Primeiro, precisamos ordenar os valores:

49, 54, 62, 64, 75, 82

Como a quantidade de valores é par, a mediana será dada pela média dos dois valores centrais. Neste caso,

$$\text{mediana}(\mathbf{X}) = (62 + 64)/2 = 63$$

A *moda* é uma medida de tendência central utilizada com dados categóricos e dados numéricos discretos. A moda é definida como a categoria que ocorre com maior frequência em um conjunto de dados. Caso um conjunto de dados tenha apenas uma categoria de maior frequência, dizemos que os dados têm distribuição *unimodal*. Caso existam duas categorias empatadas no posto de maior frequência, dizemos que a distribuição é *bimodal*. A Tabela 3.7 mostra um exemplo de distribuição unimodal, enquanto a Tabela 3.8 traz um exemplo de distribuição bimodal.

Tabela 3.7 Exemplo de distribuição unimodal. a moda é a categoria 5, que tem a maior frequência.

Categoria	Frequência
2	1
3	1
4	1
5	3
6	2
7	1
8	1

Tabela 3.8 Exemplo de distribuição bimodal. as duas modas são as categorias 3 e 7.

Categoria	Frequência
2	1
3	3
4	1
5	1
7	3
8	1
9	1

Os dois principais fatores que influenciam a escolha de uma medida de tendência central são o tipo do dado e a forma da distribuição. Para dados nominais, não é possível calcular média ou mediana, então a única opção é a moda. Para dados ordinais, é possível calcular mediana e moda, mas não a média. Para dados numéricos, a média representa o centro de gravidade dos dados; todos os elementos da amostra são considerados no cálculo da média e o resultado é mais afetado por valores extremos. Por exemplo, a presença de um único valor muito maior do que todos os outros vai aumentar artificialmente a média, afastando-a da região de elementos mais típicos. Quando uma distribuição é unimodal e simétrica, as três medidas, média, mediana e moda serão iguais, pois o ponto de frequência máxima (moda) é também o mais

central (mediana) e o centro de gravidade (média), como ilustrado na Figura 3.1. Em geral, em uma distribuição assimétrica, a moda está sempre próxima ao "pico". A média está mais próxima da "cauda" (sofre influência dos valores extremos) e a mediana está entre a moda e média. Para distribuições assimétricas, em geral a mediana é a medida preferida (Figura 3.2). Já para distribuições bimodais, a moda representa melhor os valores típicos da distribuição.

Figura 3.1 Localização de média, mediana e moda em uma distribuição simétrica.

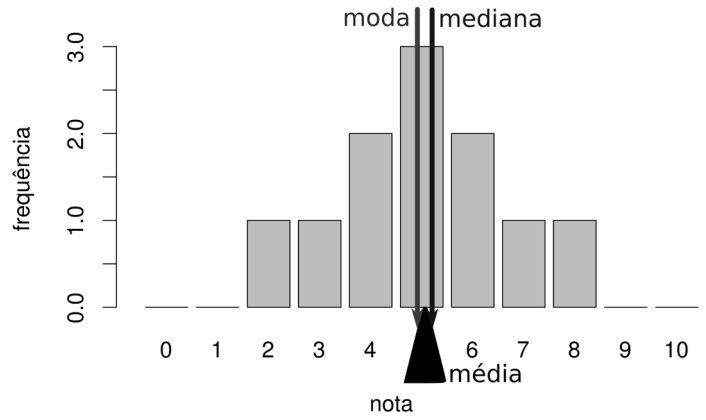
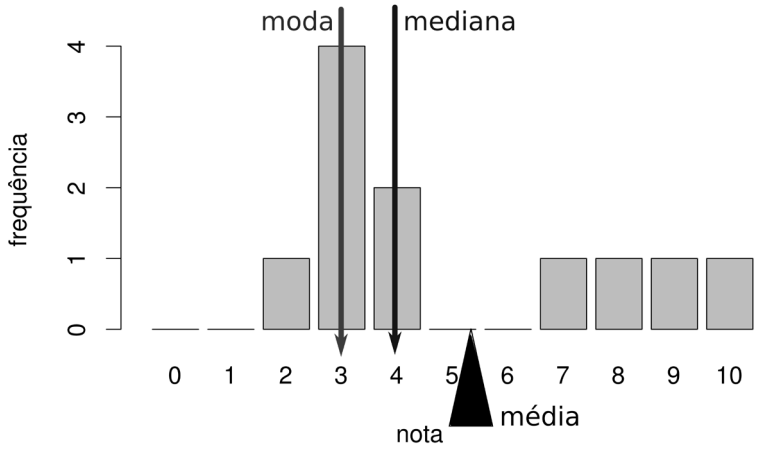


Figura 3.2 Localização de média, mediana e moda em uma distribuição assimétrica.



3.2.4 MEDIDAS DE DISPERSÃO

O processo de trabalhar com amostras introduz uma variabilidade dos resultados obtidos, pois cada amostra vai ter características ligeiramente diferentes. Essa variabilidade afeta nosso grau de confiança nos resultados. Por isso, as medidas de varia-

bilidade (ou dispersão) têm papel central na Estatística, pois são elas que permitem avaliar a precisão das conclusões que obtemos a partir dos dados experimentais.

Quando estamos interessados em uma variável aleatória contínua qualquer **X**, um indicador de variabilidade que surge naturalmente é a diferença entre cada ponto da amostra e a média: $\mathbf{X} - \overline{X}$ chamada de *desvio* ou *resíduo*. À primeira vista, poderíamos somar todas essas diferenças e obter uma estimativa da variação dos dados em torno da média. O problema é que alguns pontos estão acima da média enquanto outros estão abaixo dela, logo algumas diferenças são positivas enquanto outras são negativas. Quando somadas, diferenças de sinais diferentes compensam umas às outras e o resultado final é nulo. Isso sugere a ideia de elevar as diferenças ao quadrado antes de somar. Definimos então a *somatória dos quadrados dos desvios* de **X** como

$$\sum_{i=1}^{N} (X_i - \overline{X})^2$$

A quantidade acima representa papel importante em vários tipos de análise estatística.

Uma limitação da somatória dos quadrados dos desvios é que o resultado depende do tamanho da amostra. Quanto maior a amostra, maior o valor da somatória. Isso representa um problema quando queremos comparar amostras de tamanhos diferentes. Para resolver essa questão, podemos calcular a média do quadrado dos desvios. A princípio, pensaríamos em dividir a soma dos quadrados dos desvios pelo número N de casos na amostra. Isso gera um problema, pois a soma dos quadrados usa o valor da média, um parâmetro já estimado a partir da amostra. Uma discussão da natureza do problema está além do escopo do presente texto, mas cada parâmetro que estimamos a partir de uma amostra diminui o *grau de liberdade* daquela amostra. Para o propósito do cálculo da média dos desvios, o grau de liberdade funciona *como se fosse* o tamanho efetivo da amostra. Portanto, devemos dividir a soma dos quadrados por $N - 1$, sinalizando a perda de um grau de liberdade, em vez de dividi-la por N. A medida resultante é chamada de *variância* amostral, simbolizada por s^2 e apresentada na Equação 3.6.

$$s^2 = \frac{\sum_{i=1}^{N} (X_i - \overline{X})^2}{N - 1} \qquad (3.6)$$

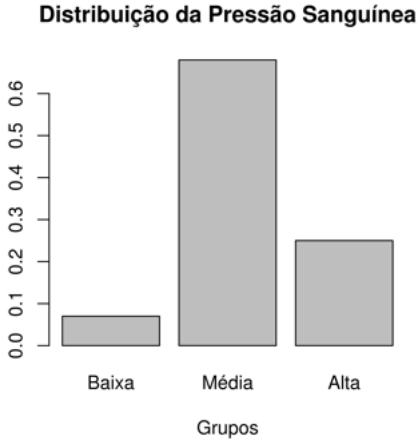
A variância é uma medida da nossa desconfiança com relação aos resultados. Quanto maior a variância, maior a variabilidade dos dados e menor a nossa confiança no resultado obtido. A variância tem a desvantagem, por causa do efeito de elevar ao quadrado, de não estar na mesma escala de unidades que a variável medida. Por exemplo, se estudamos variabilidade da altura em uma amostra, a variável é avaliada em m (metro) enquanto a variância é avaliada em m^2 (metro ao quadrado). Para facilitar a análise da variabilidade da medida, usamos a raiz quadrada da variância, denominada de *desvio padrão* amostral e representada pela letra s.

3.2.5 VISUALIZAÇÃO DE DADOS AMOSTRAIS

Depois de coletar uma amostra, um passo inicial importante da análise é obter uma representação resumida dos dados, para identificar tendências, comportamentos ou mesmo erros cometidos na aquisição dos dados. Uma maneira intuitiva de fazer isso é por meio de gráficos. Vamos examinar aqui apenas algumas alternativas simples para visualização de amostras; muitas outras podem ser encontradas em referências especializadas (Steele e Iliinsky, 2010) (Tufte, 1983).

Quando lidamos com dados categóricos, um dos parâmetros mais importantes é a *frequência relativa* de cada categoria, isto é, a proporção em que cada categoria se encontra presente na amostra. A frequência relativa pode ser prontamente visualizada em um *gráfico de barras* (ou colunas), em que cada categoria é representada por uma barra, cuja altura é proporcional à frequência respectiva. Um exemplo desse tipo de gráfico pode ser visto na Figura 3.3. Neste exemplo, foi medida a pressão sanguínea (sistólica e diastólica) de pessoas pertencentes a uma amostra de tamanho 50. As pessoas foram classificadas em três grupos, de acordo com o valor obtido: pressão normal, pressão baixa ou pressão alta. O gráfico permite ver rapidamente algumas características dos dados; por exemplo, que a ocorrência de pressão baixa é muito menor que a de pressão alta. O gráfico de barras é também utilizado para mostrar a distribuição de valores para variáveis quantitativas discretas com poucos níveis, ou a média (no eixo vertical, ou ordenada) por grupo (no eixo horizontal, a abscissa).

Figura 3.3 Gráfico de barras mostrando frequência relativa de grupos em uma amostra, de acordo com pressão sanguínea.



Para outros dados numéricos, o *histograma* é um tipo de gráfico que permite visualizar a forma da distribuição dos valores. Para construir um histograma, nós dividimos a faixa numérica da variável em intervalos. Em seguida, nós contamos quantos exemplos da amostra "caem" em cada intervalo. O histograma é simplesmente um gráfico de barras da contagem de exemplos em cada intervalo. A Figura 3.4 mostra um histograma de idade, para a mesma amostra de 50 pessoas da Figura 3.3. Pode-se ver que a maior concentração encontra-se na faixa de 30 a 35 anos (com mais de 20 pessoas), enquanto a categoria 25 a 30 anos contém menos de 15 pessoas.

Figura 3.4 Histograma mostrando distribuição da idade (em anos) em uma amostra de 50 pessoas.

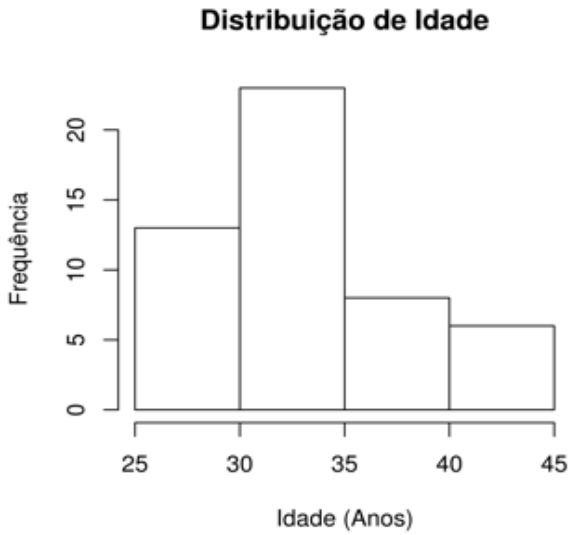
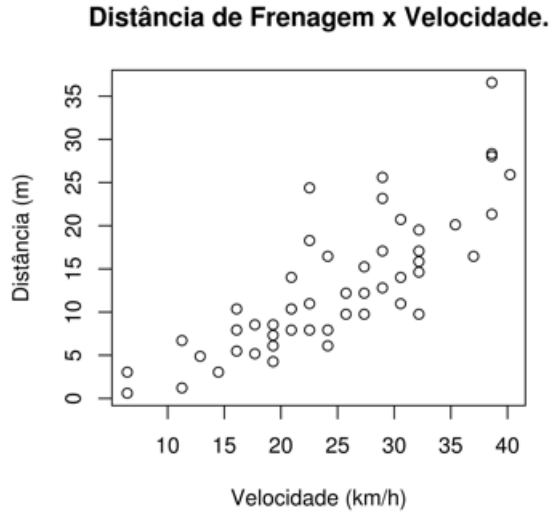


Figura 3.5 Gráfico de dispersão, mostrando a relação entre distância percorrida na frenagem e velocidade inicial para um grupo de carros.



Quando precisamos visualizar relações entre duas variáveis numéricas, a opção é usar um *gráfico de dispersão*. Nesse tipo de gráfico, simplesmente representamos os valores das duas variáveis como pares de coordenadas no plano $X - Y$, para todos os exemplos da amostra. A Figura 3.5 traz um exemplo de gráfico de dispersão, em que a distância percorrida durante frenagem é relacionada com a velocidade inicial, para um conjunto de diferentes carros. O gráfico torna possível visualizar a relação crescente entre as duas variáveis.

3.3 CORRELAÇÃO E REGRESSÃO

Correlação e regressão são duas técnicas do campo do estudo da Estatística que possuem um grau de relacionamento bem estreito e que envolvem aspectos da estimação. O foco das técnicas de correlação e regressão é a análise de dados amostrais visando descobrir como duas ou mais variáveis estão relacionadas uma com a outra numa determinada população. O cerne será o estudo de apenas duas variáveis. A análise de correlação tem como resultado um número que expressa o grau de relacionamento entre duas variáveis, enquanto a análise de regressão expressa o resultado numa equação matemática descrevendo o relacionamento. Desta forma, a equação pode ser utilizada para estimar, ou predizer, valores futuros de uma variável quando se possuem ou quando se imaginam conhecidos os valores da outra variável (Larson e Farber, 2007). Este tipo de estudo é bastante utilizado quando o trabalho/pesquisa/relatório etc. é caracterizado pela pesquisa exploratória, isto é, um analista/pesquisador busca determinar quais variáveis são relevantes e o foco está no grau do relacionamento. A correlação é uma relação entre duas variáveis. Os dados podem ser representados por pares ordenados, sendo *X* a variável independente ou explanatória e *Y* a variável dependente ou resposta. A correlação mede a força, ou grau, de relacionamento entre duas variáveis; a regressão dá uma equação que descreve o relacionamento em termos matemáticos.

A Tabela 3.9 mostra alguns exemplos de relações que poderiam ser estudadas usando algumas dessas ferramentas. Qual será o tipo de relação que existe entre as variáveis da Tabela 3.9? A correlação entre elas é significante?

Tabela 3. 9 Exemplos de pares de variáveis cujas relações podem ser estudadas usando correlação e/ou regressão.

Variável independente	**Variável dependente**
Horas de treinamento	Número de acidentes
Altura da pessoa	Número do sapato
Cigarros por dia	Capacidade pulmonar
Meses do ano	Volume de vendas
Peso da pessoa	QI

3.3.1 GRÁFICOS DE DISPERSÃO E TIPOS DE CORRELAÇÃO

Construir o gráfico de dispersão da variável dependente versus variável independente é, usualmente, o primeiro passo em uma análise de correlação. Devemos tomar cuidado com os termos *dependente* e *independente* neste contexto, pois, mesmo encontrando relação entre as variáveis, isto não necessariamente significa relação causal entre elas. A análise de correlação parte de uma *hipótese*, isto é, de um enunciado formal das relações esperadas entre pelo menos uma variável independente e uma variável dependente. Nas pesquisas exploratórias, as hipóteses podem se tornar questões de pesquisa. Estas questões, pela sua especificidade, devem dar testemunho do trabalho

conceitual efetuado pelo pesquisador e, pela sua clareza, permitir uma resposta interpretável.

A Figura 3.6 traz um exemplo de gráfico de dispersão em que a abscissa (o eixo *X* – variável independente) representa horas de treinamento numa linha de produção de automóveis e a ordenada (o eixo *Y* – variável dependente) representa o número de acidentes na fábrica. A análise gráfica do comportamento entre as variáveis mostra existência de correlação negativa, pois, à medida que X cresce, Y decresce. O gráfico mostra que a empresa, ao investir em treinamento, reduz o número de acidentes na fábrica.

Figura 3.6 Gráfico de dispersão, número de acidentes em função de tempo de treinamento.

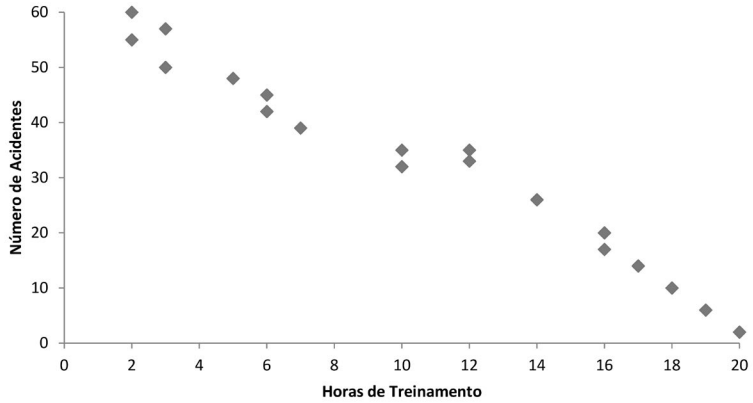
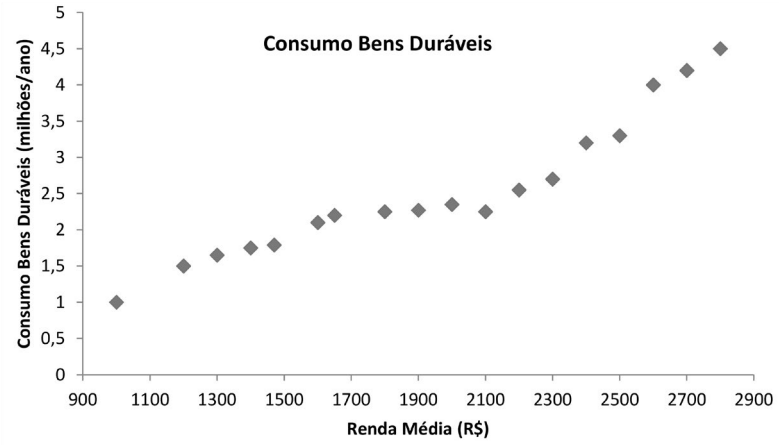


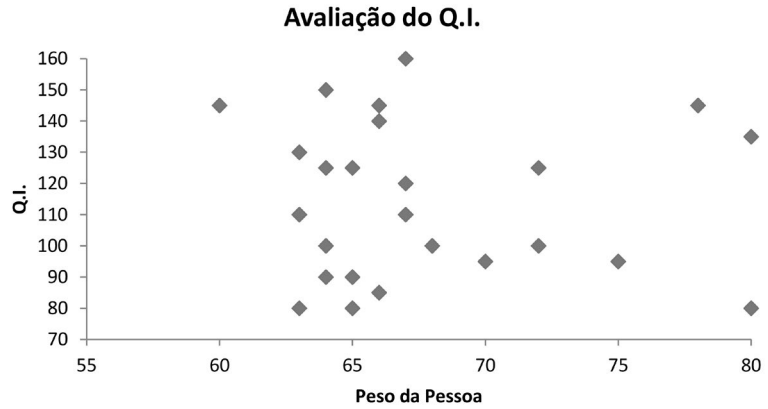
Figura 3.7 Gráfico de dispersão, consumo de bens em função do aumento da renda média.



Outro exemplo é mostrado pela Figura 3.7, em que o eixo *X* representa aumento da renda média da população e o eixo *Y* representa consumo de bens duráveis em milhões/ano. A análise mostra que há correlação positiva, pois, à medida que X cresce, Y também cresce. O gráfico mostra que, com o aumento médio da renda da população (maior poder aquisitivo), o consumo de bens duráveis aumenta.

O próximo exemplo é mostrado pela Figura 3.8, em que o peso de uma pessoa é representado no eixo *X* enquanto seu Q. I. (Quociente de Inteligência) é representado no eixo *Y*. Não há correlação linear; o gráfico mostra que não existe evidência de alguma relação entre o peso de uma pessoa com seu Q. I.

Figura 3.8 Gráfico de dispersão, Q. I. em função do peso da pessoa.



Além da correlação linear, em que a relação entre as duas variáveis é expressa adequadamente por uma reta, podemos encontrar também correlação *não-linear* entre as variáveis. Nesse caso, apesar de existir uma relação clara entre as variáveis, esta não pode ser modelada por uma reta. As Figuras 3.9 e 3.10 ilustram dois casos de correlação não-linear.

Figura 3.9 Exemplo de correlação não-linear.

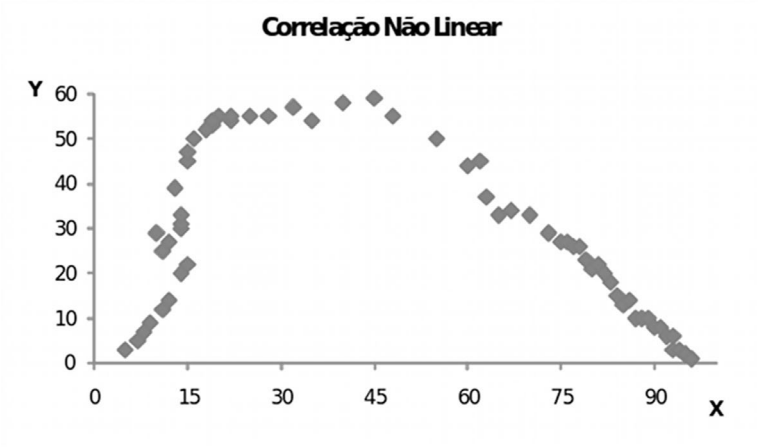
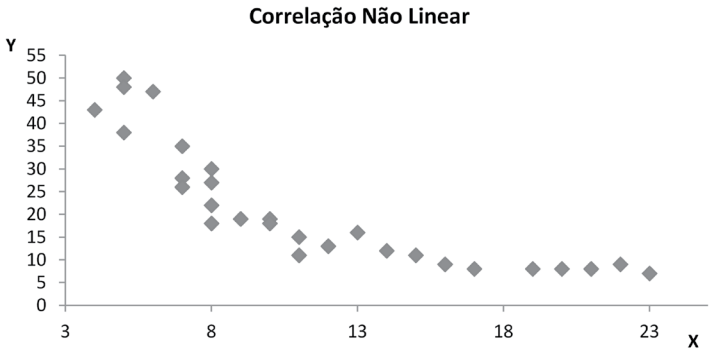


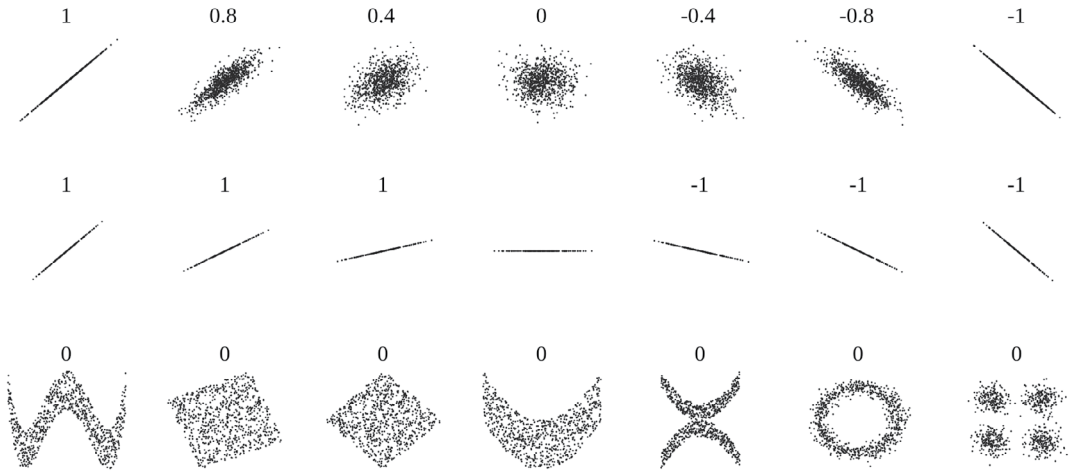
Figura 3.10 Outro exemplo de correlação não-linear.



3.3.2 COEFICIENTE DE CORRELAÇÃO

Utilizar apenas o mapa de dispersão para interpretar a existência de uma correlação pode ser uma tarefa considerada bastante subjetiva. Para isto, utiliza-se medir o grau e o tipo de uma correlação linear entre duas variáveis por meio do cálculo do coeficiente de correlação (*r*). O intervalo de variação do coeficiente de correlação *r* vai de –1 a 1. Sendo assim, se as variáveis **X** e **Y** tiverem forte correlação linear positiva, o valor de *r* está próximo de 1. Se as variáveis **X** e **Y** tiverem forte correlação linear negativa, *r* estará próximo de –1 (menos um). Contudo, se não existir correlação linear ou se a correlação linear for fraca, o valor de *r* estará próximo de zero. A Figura 3.11 ilustra vários gráficos de dispersão, com diferentes tipos de relações entre as variáveis, e os respectivos coeficientes de correlação.

Figura 3.11 Diversos exemplos de relações entre variáveis e respectivos coeficientes de correlação.



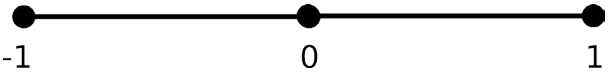
Fonte: http://en.wikipedia.org/wiki/File:Correlation_examples2.svg (domínio público).

O coeficiente de correlação pode ser calculado pela Equação 3.7.

$$r_{XY} = \frac{\sum_{i=1}^{N} (X_i - \overline{X}) \quad (Y_i - \overline{Y})}{\sqrt{\sum_{i=1}^{N} (X_i - \overline{X})^2} \quad \sqrt{\sum_{i=1}^{N} (Y_i - \overline{Y})^2}} \tag{3.7}$$

O intervalo de *r* é de -1 até 1 (ver Figura 3.12).

Figura 3.12 Representação gráfica do intervalo possível para o coeficiente de correlação.



O numerador da fração na Equação 3.7, quando dividido por *N*-1, também é chamado de covariância da amostra, anotado como *cov*(**X**,**Y**) ou s_{XY}:

$$s_{XY} = \frac{\sum_{i=1}^{N} (X_i - \overline{X}) (Y_i - \overline{Y})}{N - 1}$$

Como mostra esta fórmula, esta medida representa a média do produto dos desvios de **X** e **Y**. Com um pouco de álgebra, é fácil mostrar que a correlação r_{xy} é igual à covariância dividida pelo produto dos desvios-padrão (Equação 3.8).

$$r_{XY} = \frac{s_{XY}}{s_X s_Y} \tag{3.8}$$

Existem outras maneiras de escrever ou calcular a correlação, mas, para o cálculo à mão, a Equação 3.9 (que equivale às anteriores) é mais prática.

$$r_{XY} = \frac{N \sum_{i=1}^{N} X_i Y_i - \sum_{i=1}^{N} X_i \sum_{i=1}^{N} Y_i}{\sqrt{N \sum_{i=1}^{N} X_i^2 - (\sum_{i=1}^{N} X_i)^2} \quad \sqrt{N \sum_{i=1}^{N} Y_i^2 - (\sum_{i=1}^{N} Y_i)^2}} \tag{3.9}$$

Como exemplo, vamos analisar o coeficiente de correlação entre o número de faltas dos alunos por semestre, em relação a suas respectivas notas finais numa determinada disciplina. Os dados estão expressos na Tabela 3.10 e o respectivo gráfico de dispersão está na Figura 3.13. A Tabela 3.11 mostra os valores intermediários (XY, X^2, Y^2) usados no cálculo de *r*. Usando os dados dessa tabela, lembrando que $N = 7$ e aplicando a Equação 3.9, obtemos a Equação 3.10.

$$r_{XY} = \frac{7(3.751) - (57)(516)}{\sqrt{7(579) - (57)^2} \sqrt{7(39.898) - (516)^2}} \tag{3.10}$$

$$r_{XY} = -0.975 \tag{3.11}$$

Tabela 3.10 Tabela de notas e faltas para cada aluno.

Faltas por semestre (X)	Nota Final (Y)
8	78
2	92
5	90
12	58
15	43
9	74
6	81

Figura 3.13 Gráfico de dispersão para os dados da Tabela 3.10.

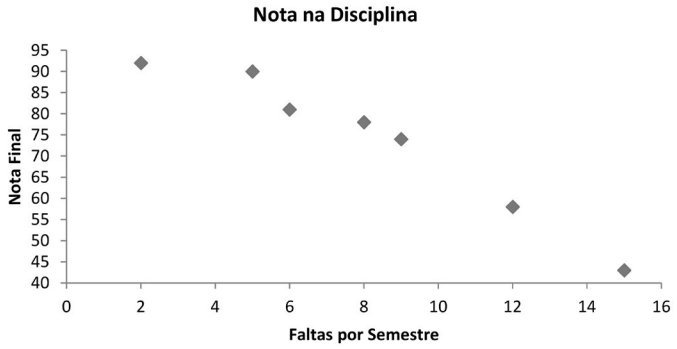


Tabela 3.11 Valores intermediários usados no cálculo do coeficiente de correlação.

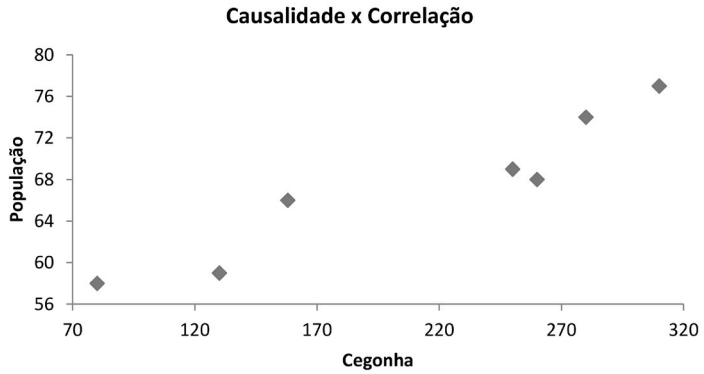
i	*X*	*Y*	*XY*	X^2	Y^2
1	8	78	624	64	6.084
2	2	92	184	4	8.464
3	5	90	450	25	8.100
4	12	58	696	144	3.364
5	15	43	645	225	1.849
6	9	74	666	81	5.476
7	6	81	486	36	6.561
Σ	57	516	3.751	579	39.898

3.3.3 CAUSALIDADE VERSUS CORRELAÇÃO

Correlação não necessariamente implica causalidade. Pesquisadores frequentemente são tentados a inferir uma relação de causa e efeito entre X e Y quando eles ajustam um modelo de regressão ou realizam uma análise de correlação. Uma associação significativa entre X e Y em ambas as situações não necessariamente implica uma relação de causa e efeito. Um exemplo pode ser visto no gráfico de dispersão da Figura 3.14, que mostra a população de Oldenberg, Alemanha, no fim de cada um dos 7 anos (Y)

contra o número de cegonhas (pássaros) naquele ano (X). O exame do gráfico pode induzir à interpretação de que existe associação entre X e Y. Frequentemente, quando duas variáveis X e Y parecem estar fortemente associadas, isso pode ser porque X e Y estão, de fato, associadas com uma terceira variável, W. Imagine que cada ponto no gráfico da Figura 3.14 represente uma contagem de cegonhas e da população de um ano. Por razões demográficas, a população na cidade cresce com os anos. O número de cegonhas também cresce, por exemplo, por causa de um programa de recuperação ambiental. Haverá uma correlação entre população de pessoas e de cegonhas por razões que não têm nada a ver com uma relação causal entre pessoas e nascimentos. Neste exemplo, X e Y aumentam com W que é a variável tempo.

Figura 3.14 Gráfico de dispersão, número de cegonhas versus população da cidade de Oldenberg.



Portanto, o fato de duas variáveis estarem correlacionadas não implica, por si só, em uma relação de causa e efeito entre as variáveis. Um estudo mais profundo e com mais detalhes é essencial para determinar se há relação causal entre as variáveis. É possível que a relação das variáveis tenha sido causada por uma terceira variável ou ainda por uma combinação de muitas outras variáveis ou é possível que a relação entre duas variáveis seja uma coincidência. Segundo Stevenson,

> É que, para estabelecer relações válidas, é preciso mais que simplesmente emparelhar qualquer tipo de dados até achar alguma correlação. Em vez disso, usam-se estudos correlacionados como pesquisas exploratórias iniciais a fim de identificar futuras áreas de pesquisa. Resultados que parecem promissores com base na lógica ou na teoria devem ser submetidos a maior análise (tal como experimentos controlados) para determinar se existe uma relação de causa e efeito. (Stevenson, 2001:389)

3.3.4 A RETA DA REGRESSÃO LINEAR ESTIMADA

Depois de constatar que existe uma correlação linear significante, você pode escrever uma equação que descreva a relação entre as variáveis *x* e *y*. Essa equação chama-se reta de regressão ou reta do ajuste ótimo. Pode-se escrever a equação de uma reta

como $y = mx + b$, onde m é a inclinação da reta e b, o intercepto. A inclinação da reta quantifica o quanto y cresce (ou diminui, se for negativa) com cada unidade de x. O intercepto é o valor de **Y** quando **X** é 0, o que em muitos gráficos é o valor onde a reta cruza a ordenada.

Assim, a reta de regressão está apresentada na Equação 3.12.

$$\hat{\mathbf{Y}} = m\mathbf{X} + b \tag{3.12}$$

A inclinação m pode ser obtida pela Equação 3.13.

$$m = \frac{s_{XY}}{s_X^2} = \frac{N\sum_{i=1}^{N} X_i Y_i - \sum_{i=1}^{N} X_i \sum_{i=1}^{N} Y_i}{N\sum_{i=1}^{N} X_i^2 - (\sum_{i=1}^{N} X_i)^2} \tag{3.13}$$

E o intercepto b é dado pela Equação 3.14.

$$b = \overline{Y} - m\overline{X} \tag{3.14}$$

Retornando ao exemplo anterior, vamos escrever a equação da reta de regressão entre o número de faltas dos alunos por semestre, em relação a suas respectivas notas finais numa determinada disciplina. Novamente, usaremos os valores da Tabela 3.11 para nos auxiliar.

Primeiro, calculamos m e b:

$$m = \frac{7(3.751) - (57)(516)}{7(579) - (57)^2} = -3.924$$

$$\overline{Y} = 516/7 = 73.714$$

$$\overline{X} = 57/7 = 8.143$$

$$b = 73.714 - (-3.924)(8.143) = 105.667$$

A reta de regressão é:

$$\hat{Y} = -3.924X + 105.667$$

Como os dados da Tabela 3.1 não possuem casas decimais, precisamos adequar a estimativa à precisão fornecida, pois uma estimativa final não deve ser mais "precisa" do que os dados fornecidos originalmente. Uma representação mais adequada para a reta de regressão pode ser com um algarismo de significância a mais que os dados iniciais, isto é, com uma casa decimal para os valores de m e b:

$$\hat{Y} = -3.9X + 105.7$$

Com a reta de regressão, é possível prever valores de **Y** correspondentes aos valores de **X** que caiam em determinado intervalo de dados.

Exemplo: Use a equação de regressão acima para prever a nota esperada de um aluno com:

a) 3 faltas

b) 12 faltas

Item (a): $\hat{Y} = -3.9(3) + 105.7 = 94$

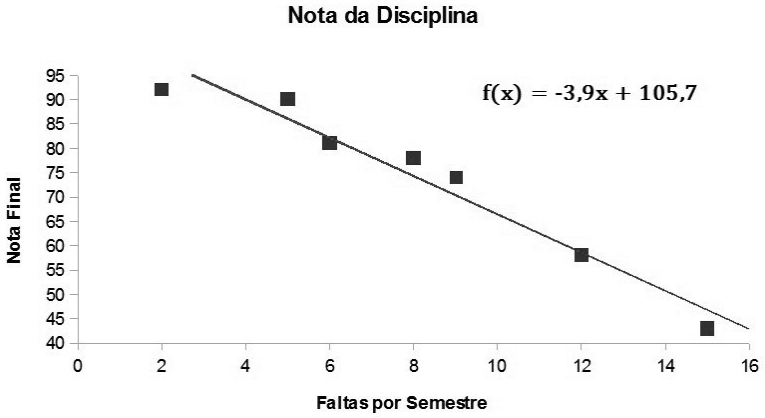
Item (b): $\hat{Y} = -3.9(12) + 105.7 = 58.9$

Novamente, como os dados da Tabela 3.10 não possuem casa decimal, uma previsão ajustada aos dados é de uma nota $\hat{Y} = 59$. Caso o resultado do item (a) fosse 94.2, também seria necessário o ajuste da previsão, para a mesma significância dos dados, ou seja, para 94.

Portanto, após ter obtido a equação de uma reta de regressão, você pode usar essa equação para prever valores **Y** sob o intervalo dos dados se a correlação entre **X** e **Y** for significante. Para prever valores **Y** , basta substituir o valor dado de **X** na equação de regressão, calculando então o valor **Y** previsto.

A reta de regressão calculada está representada graficamente na Figura 3.15.

Figura 3.15 Reta de regressão, exprimindo relação entre faltas e nota final.



3.3.5 O COEFICIENTE DE DETERMINAÇÃO

O coeficiente de determinação, R^2, é a razão entre a variação explicada em *y* e a variação total em *y*.

$$R^2 = \frac{\text{Variação Explicada}}{\text{Variação total}} \qquad (3.15)$$

O coeficiente de determinação é o quadrado do coeficiente de correlação.

Para o exemplo anterior, temos que o coeficiente de correlação entre as faltas e a nota final é de r = – 0,975. Logo, o coeficiente de determinação é dado por:

$$R^2 = (-0,975)^2 = 0,9506$$

Interpretação: Cerca de 95% da variação nas notas finais pode ser explicada pelo número de vezes que o aluno falta. Os outros 5% são inexplicados e podem estar relacionados com um erro amostral ou outras variáveis, como inteligência, tempo dedicado ao estudo etc.

3.3.6 OUTROS TIPOS DE CORRELAÇÃO

Existem outros tipos de gráficos que podem ser gerados e não necessariamente a regressão linear é sempre a mais indicada para isso. A seguir apresentamos alguns gráficos de curvas de regressão usando diferentes funções: exponencial (Figura 3.16), logarítmica (Figura 3.17), polinomial (Figura 3.18) e de potência (Figura 3.19) para os mesmos dados do exemplo anterior.

Figura 3.16 Curva de regressão exponencial.

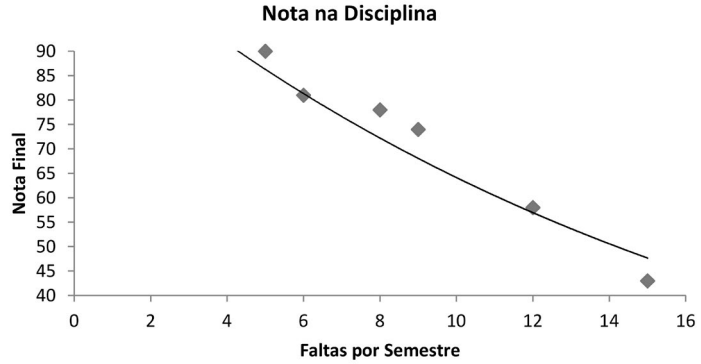


Figura 3.17 Curva de regressão logarítmica.

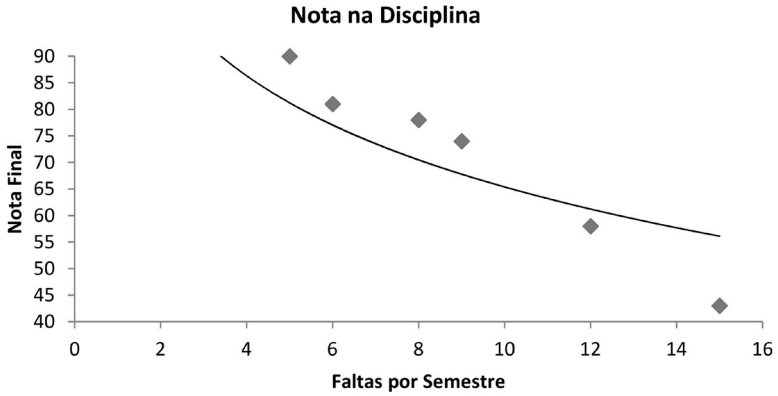


Figura 3.18 Curva de regressão polinomial.

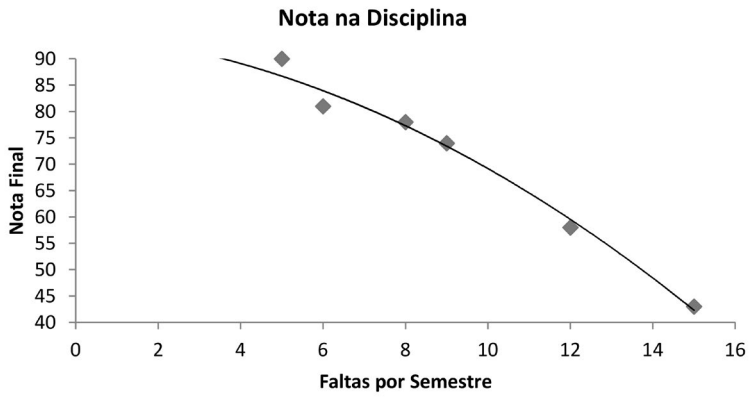
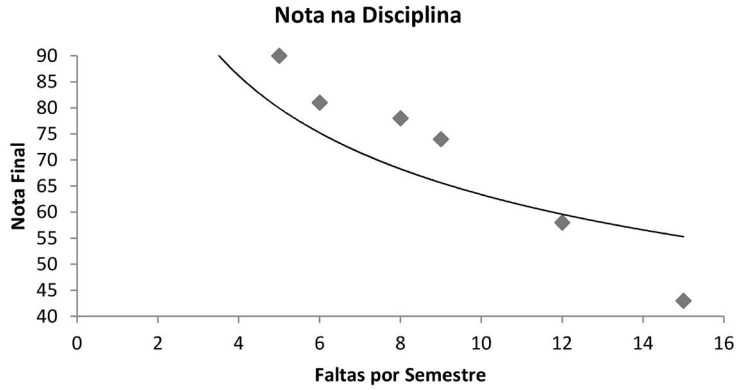


Figura 3.19 Curva de regressão de potência.

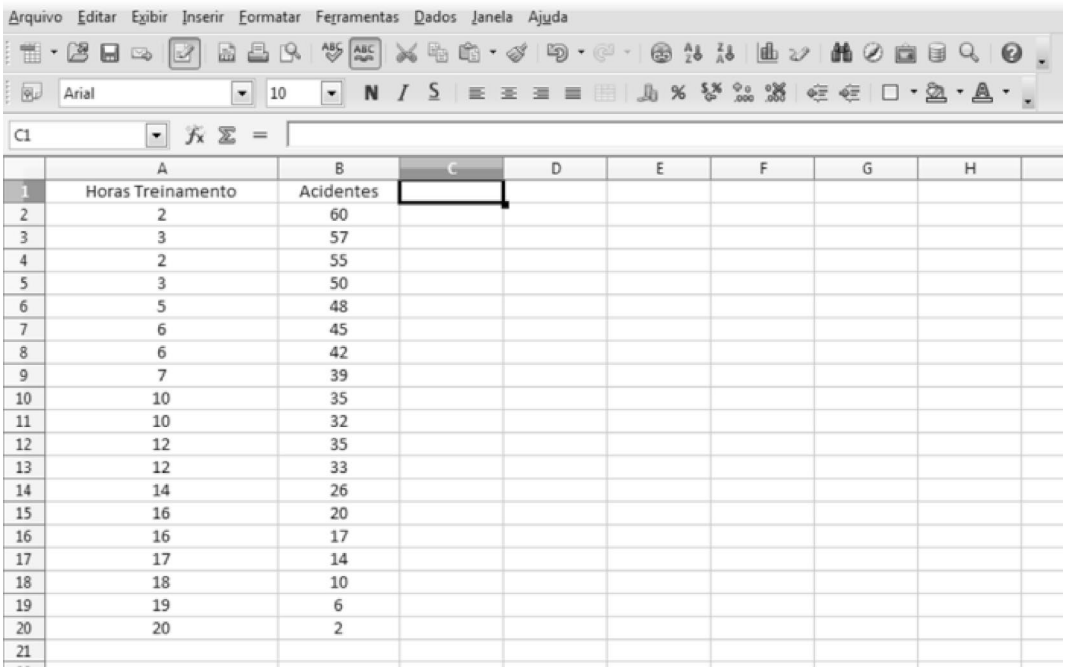


3.3.7 CRIANDO UM GRÁFICO DE DISPERSÃO COM O LIBREOFFICE CALC

A seguir detalhamos os passos necessários para a criação de um gráfico de dispersão usando o programa LibreOffice Calc:

1) Inserção dos valores das variáveis independente (*X*) e dependente (*Y*). No exemplo a seguir são as Horas de Treinamento versus Número de Acidentes (Figura 3.20);

Figura 3.20 Dados para construção do gráfico de correlação.



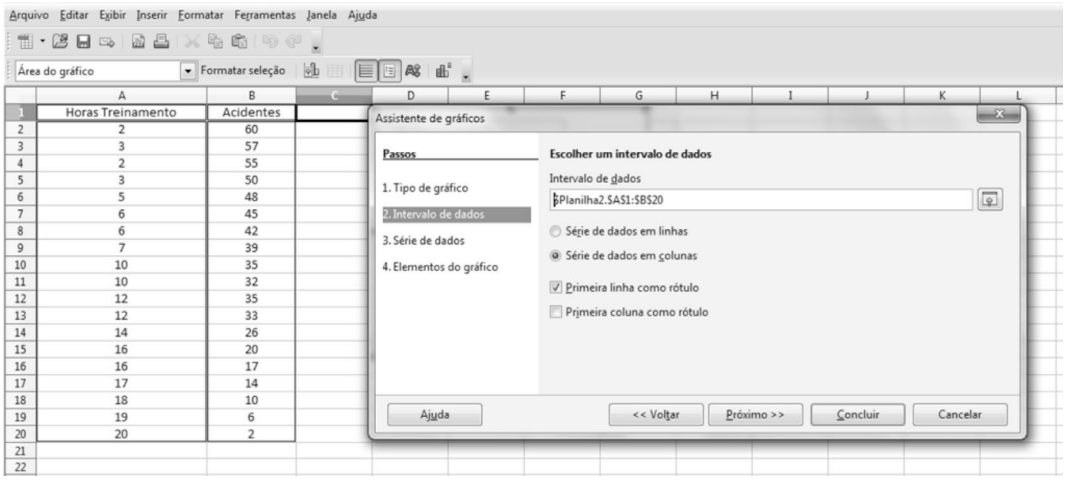
	A	B	C	D	E	F	G	H
1	Horas Treinamento	Acidentes						
2	2	60						
3	3	57						
4	2	55						
5	3	50						
6	5	48						
7	6	45						
8	6	42						
9	7	39						
10	10	35						
11	10	32						
12	12	35						
13	12	33						
14	14	26						
15	16	20						
16	16	17						
17	17	14						
18	18	10						
19	19	6						
20	20	2						
21								

2) Podemos usar diretamente o ícone `Gráfico` da barra de ferramentas ou a opção `Inserir >> Gráfico`, em seguida devemos escolher o gráfico de dispersão (Figura 3.21);

Figura 3.21 Escolhendo gráfico de dispersão.

3) Então definimos o intervalo de dados (Figura 3.22), selecionando todas as colunas de dados (Figura 3.23);

Figura 3.22 Definição do intervalo de dados.



4) Na sequência de Figuras 3.24, 3.25 e 3.26, vemos como inserir cada uma das colunas (*X* e *Y*);

5) No próximo passo, podemos escolher título do gráfico, subtítulo, nomes das variáveis X e Y (Figura 3.27);

6) Após a inserção de todas as informações podemos clicar em "Concluir" e o gráfico será gerado, conforme a Figura 3.28;

Figura 3.23 Seleção das colunas de dados.

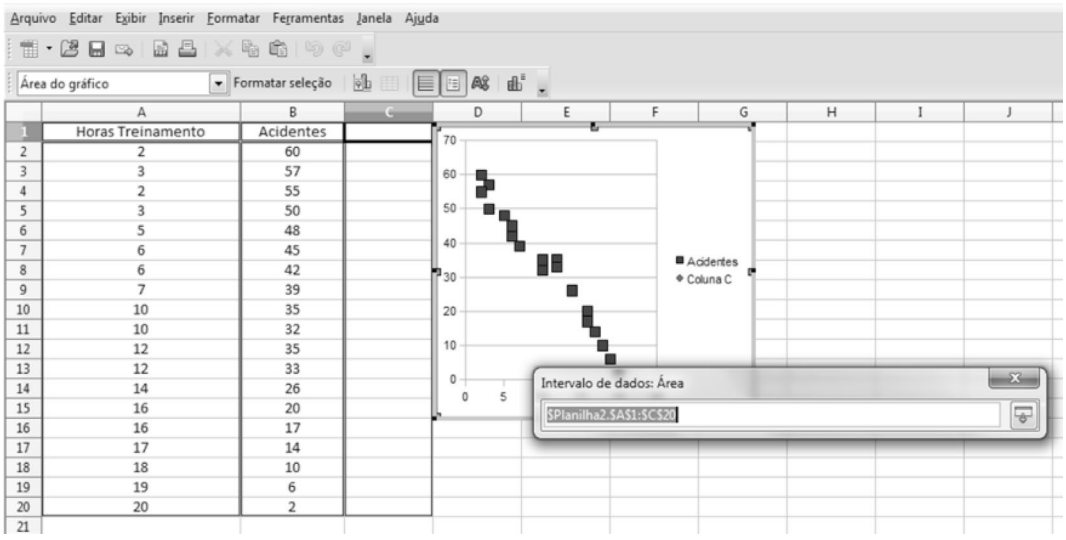


Figura 3.24 Inserindo nome.

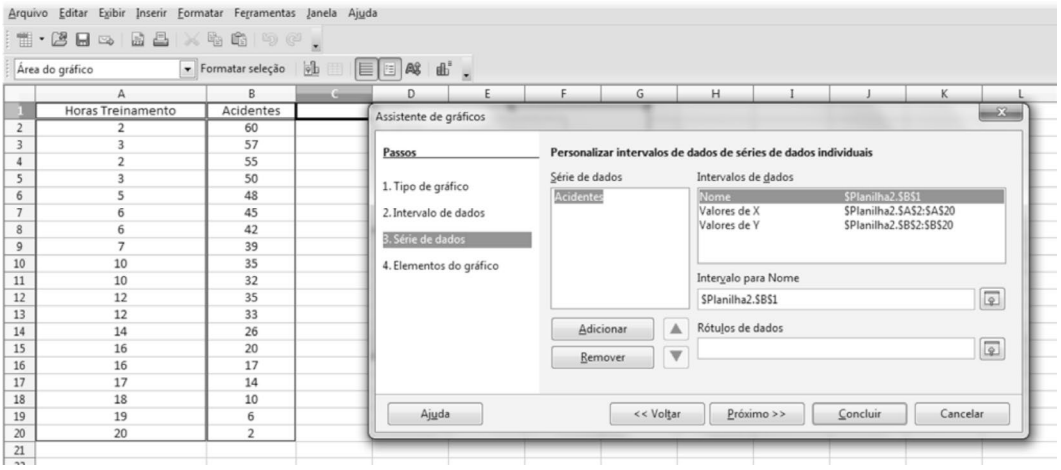


Figura 3.25 Inserindo coluna X.

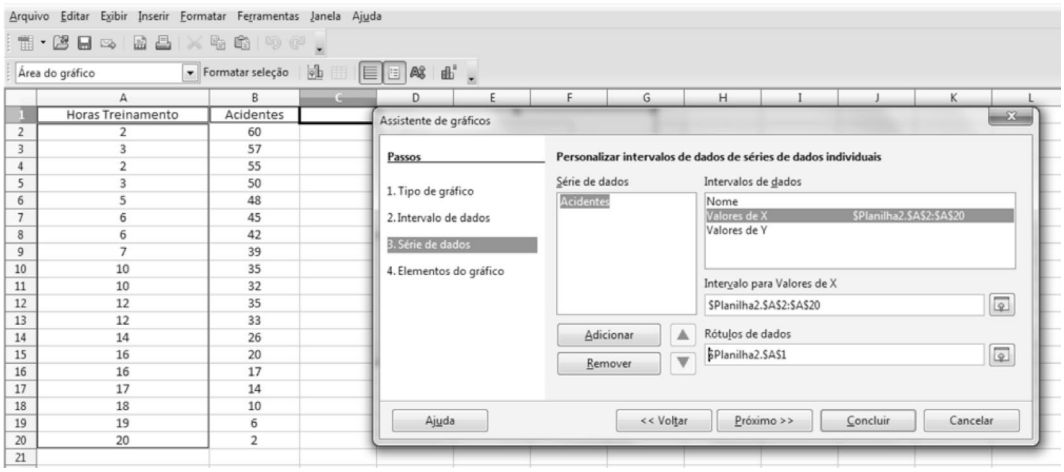


Figura 3.26 Inserindo coluna Y.

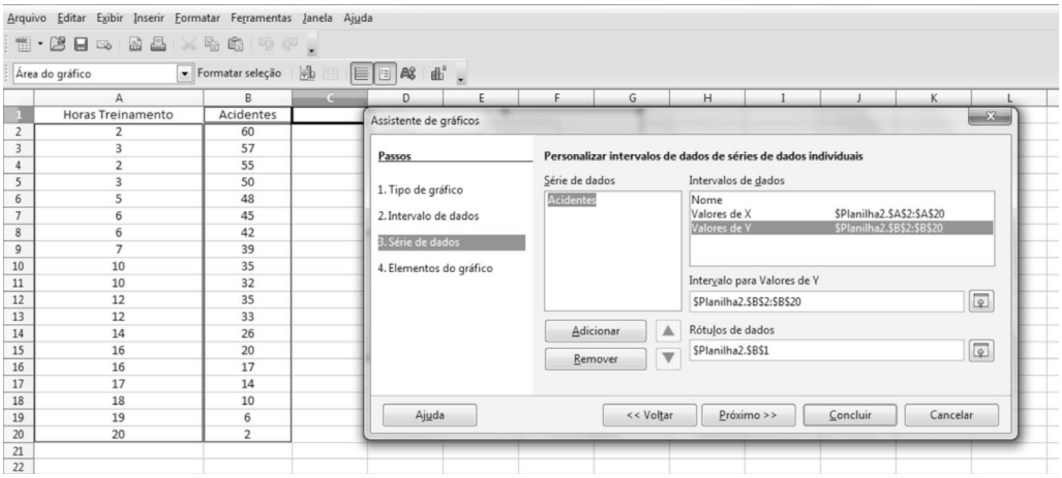


Figura 3.27 Escolha de título e nome das variáveis.

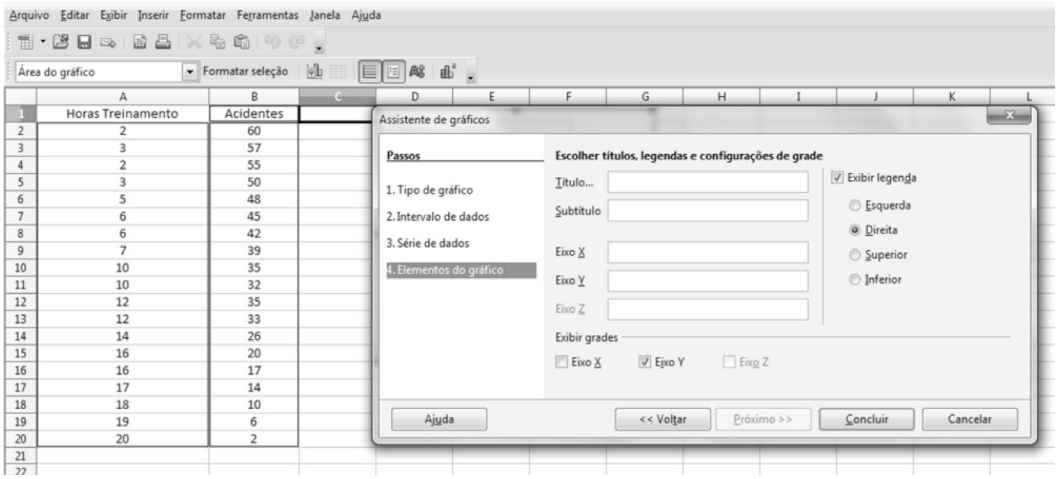
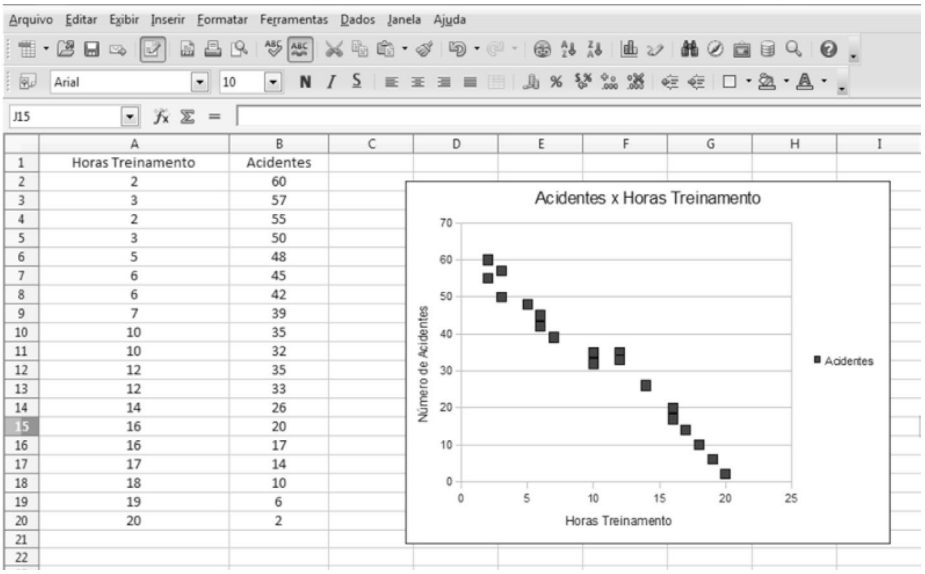
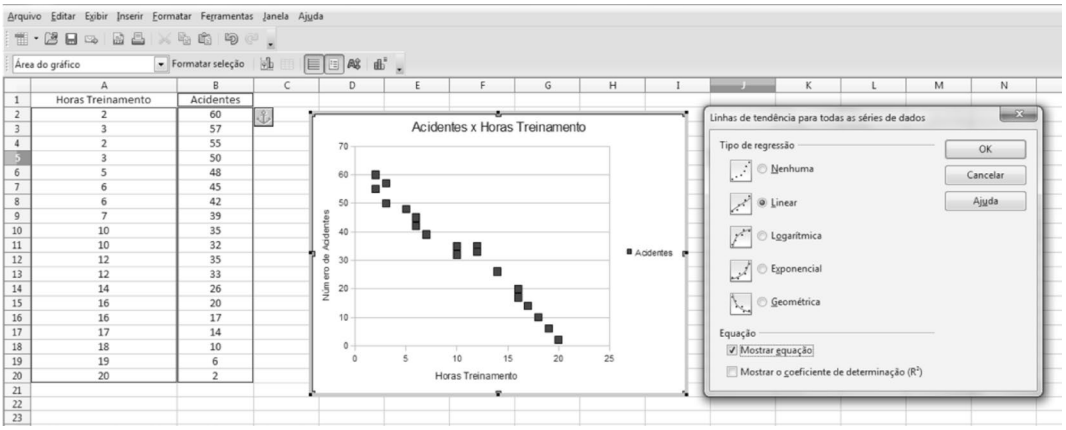


Figura 3.28 Gráfico de dispersão.



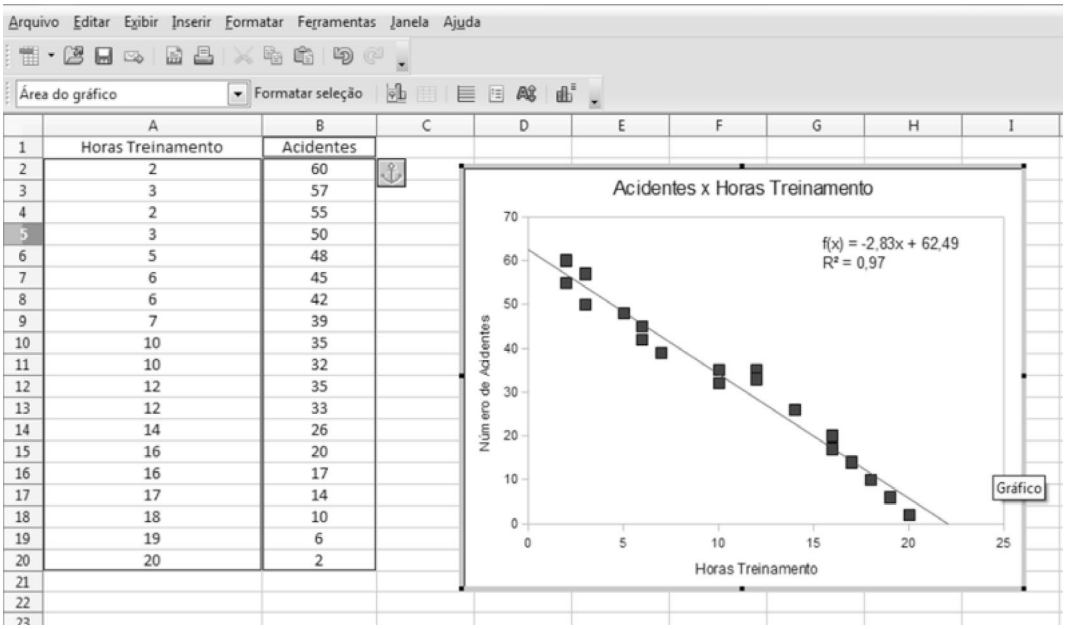
7) Clicando-se duas vezes sobre o gráfico é possível criar a reta de regressão, a equação da reta e o coeficiente de determinação. Após clicar duas vezes sobre o gráfico, vá até o campo "Inserir" e clique em "Linha de Tendência". Aparecerá a janela denominada: "Linhas de tendência". Clique em "linear". Pode-se optar também pela exibição da equação da reta e do coeficiente de determinação (R^2), conforme o quadro situado ao lado do gráfico (Figura 3.29);

Figura 3.29 Tela de seleção do tipo de regressão.



8) Concluindo esta etapa, a linha de tendência será gerada, com a equação da reta e o valor do coeficiente de determinação (Figura 3.30). Neste caso, há forte correlação negativa, como mostram o gráfico e o valor de R^2 (0,97);

Figura 3.30 Resultado da regressão.



9) O valor do coeficiente de correlação (*r*) pode ser calculado no LibreOffice Calc seguindo o mesmo procedimento para calcular outras funções. Clique no ícone do assistente de funções ou use o menu `Inserir >> Função`. Selecione o grupo de funções estatísticas e escolha a função `CORREL` (Figura 3.31). O próximo passo é selecionar as colunas de dados. Neste exemplo, para o campo "Dados_1" é selecionada a coluna "Horas Treinamento", e para o campo "Dados_2" é selecionada a coluna "Acidentes" (Figura 3.32);

Figura 3.31 Selecionando função correlação.

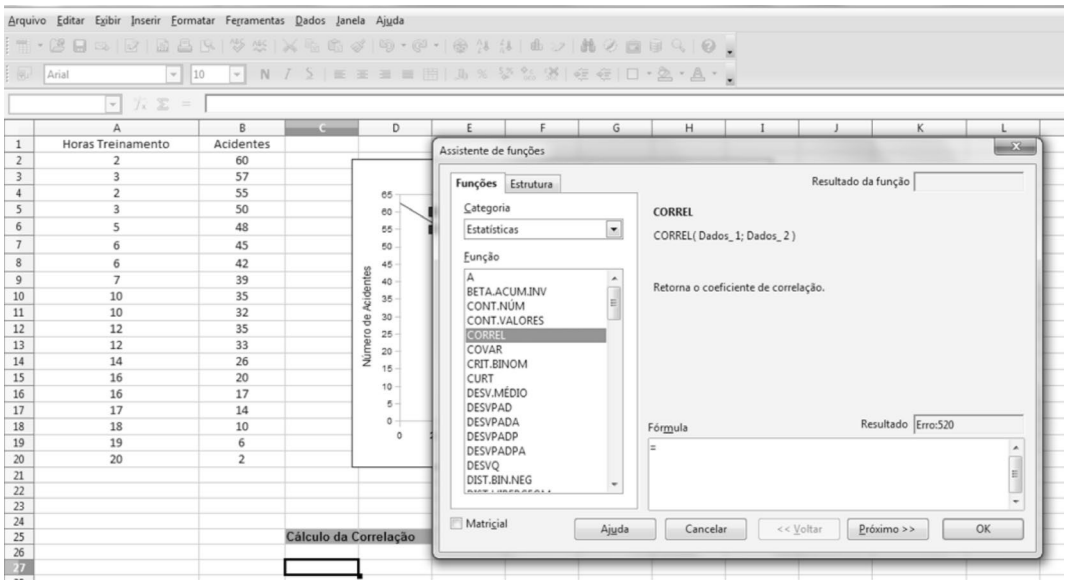
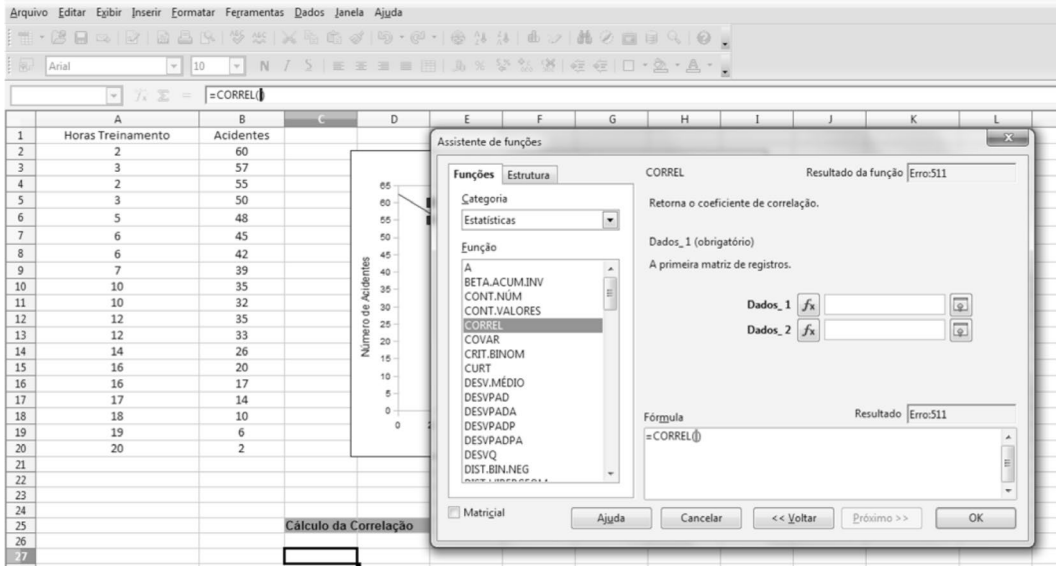
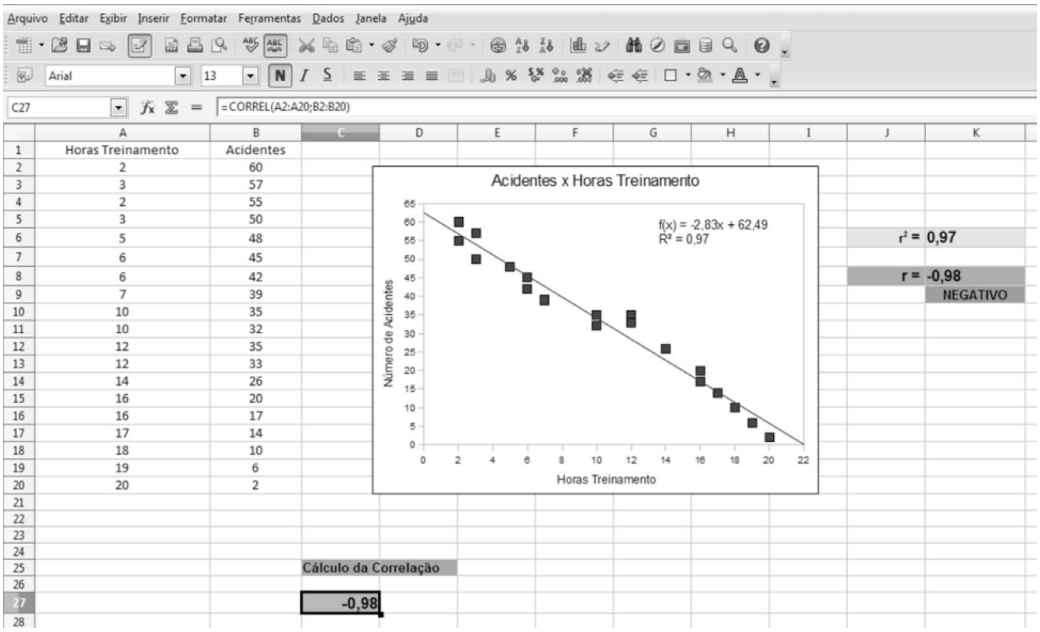


Figura 3.32 Selecionando as colunas de dados.



10) Finalmente, o valor calculado é inserido na planilha (Figura 3.33). No presente caso, o valor de r é negativo (–0,98), pois há uma forte correlação negativa!

Figura 3.33 Resultado final com o valor do coeficiente de correlação.



3.4 ATIVIDADES EM AULA

Atividade 1

1) Os tempos de reação de um indivíduo a determinados estímulos foram medidos por um psicólogo como sendo 0,53; 0,46; 0,50; 0,49; 0,52; 0,53; 0,44 e 0,55 segundos, respectivamente. Determine o tempo médio de reação do indivíduo a esses estímulos;

a) Iniciar o programa LibreOffice Calc. Entrar com os dados em uma nova planilha tendo a primeira linha como título e cada valor em uma linha. Deixar uma coluna livre antes dos dados para poder colocar os nomes das medidas calculadas (ver Figura 3.34);

b) Clicar na célula onde se deseja colocar a medida calculada (vide Figura 3.34) Clicar em f_x ou em Inserir (no menu Principal) e em seguida em Função (ver Figura 3.35);

c) Clicar na barra de rolamento de Categoria e clicar em Estatísticas (Figura 3.36). A lista de funções é filtrada para mostrar apenas as funções estatísticas, basta clicar na desejada;

d) Clicar em `MÉDIA`. Observe que existem vários tipos de média (ver Figura 3.37)! Nestas linhas deverão ser inseridos os valores para se calcular a média ou então as células onde se encontram os valores;

Figura 3.34 Entrada de dados na planilha.

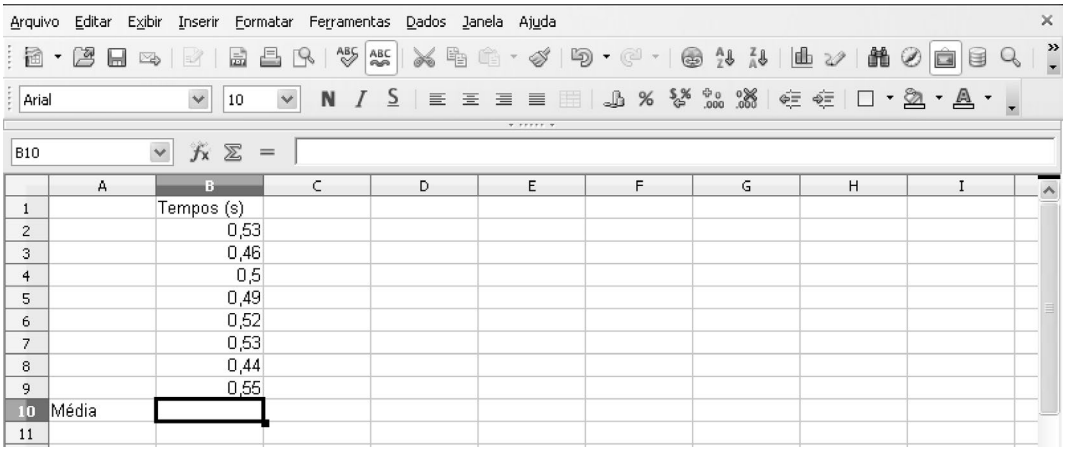
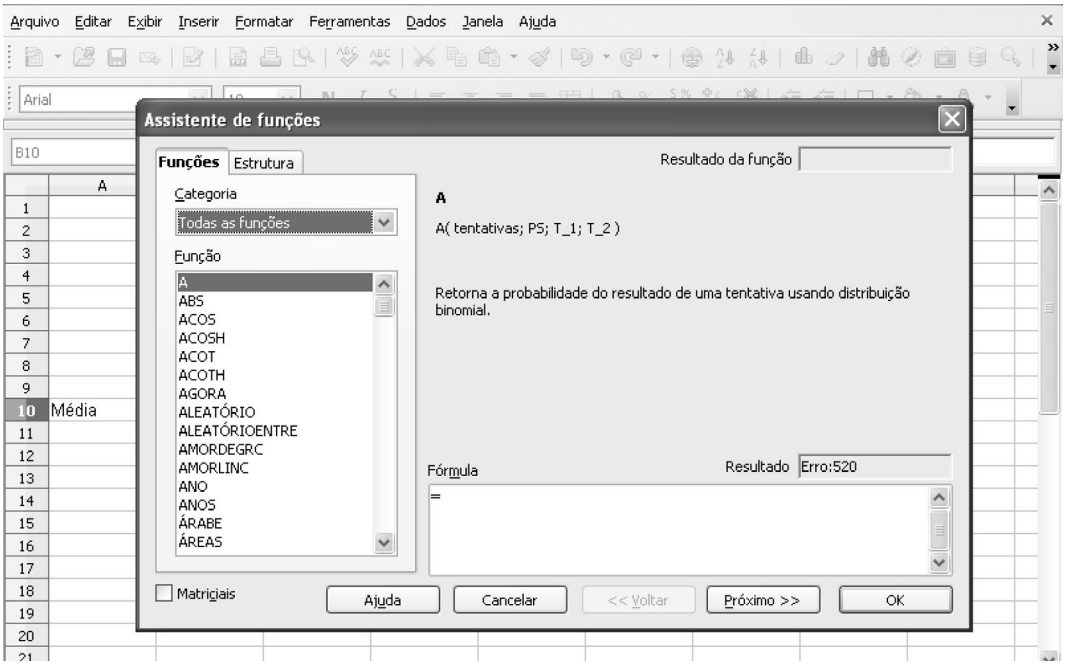


Figura 3.35 Inserindo função.



e) Para inserir as células contendo os valores basta clicar na flecha verde (Figura 3.37) em frente ao número 1 e depois selecionar os valores com o *mouse* (Figura 3.38). Observe que as referências dos dados aparecerão na barra do Assistente de Funções, no exemplo, B2:B9;

f) Voltar a clicar na flecha verde (Figura 3.38) e clicar em OK. O resultado da função escolhida, no caso a média aritmética, será colocado na célula inicial do procedimento (Figura 3.39). Observe que ao selecionar uma célula onde há uma função, a barra de funções exibirá o nome da função e as células utilizadas para o cálculo. Você pode alterar estas células nesta barra, caso necessário.

Figura 3.36 Selecionando funções estatísticas.

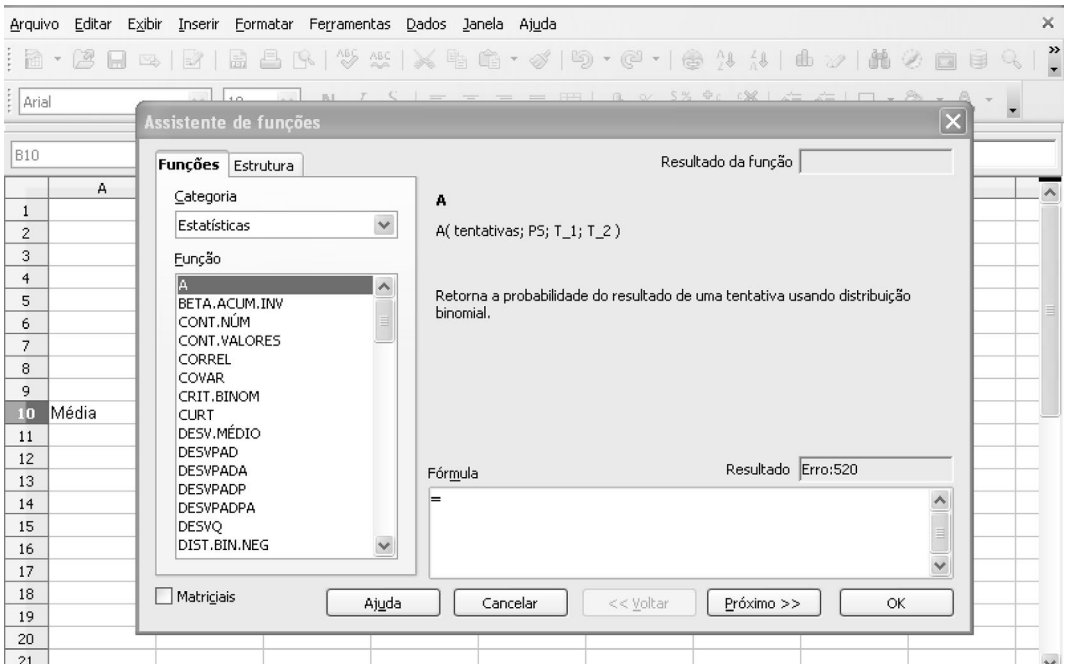


Figura 3.37 Selecionando a função média.

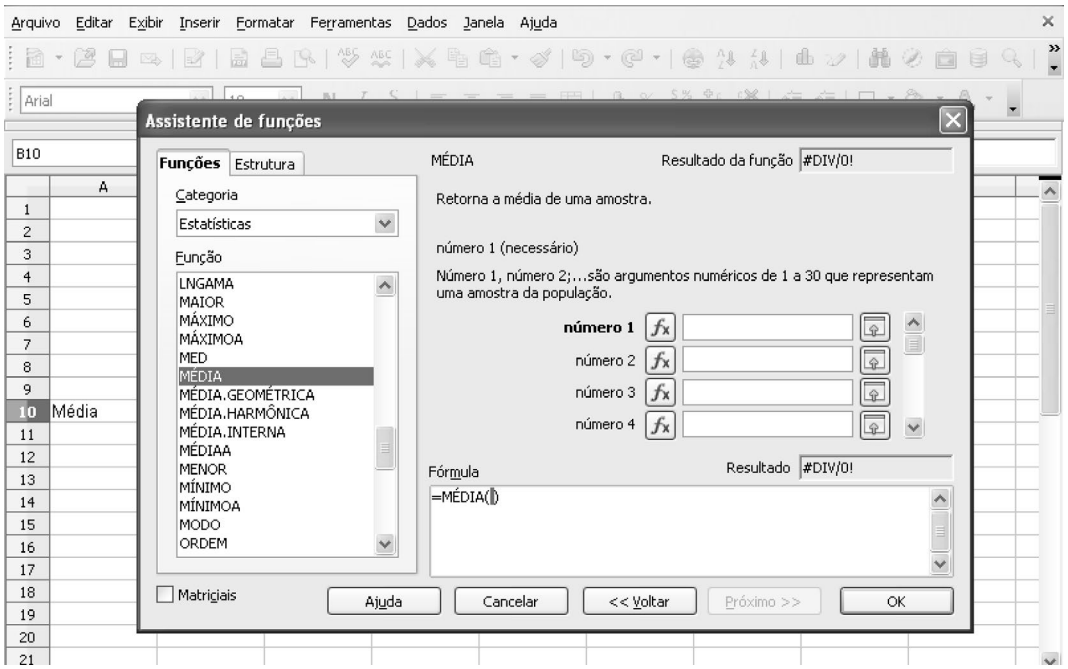


Figura 3.38 Selecionando coluna de dados.

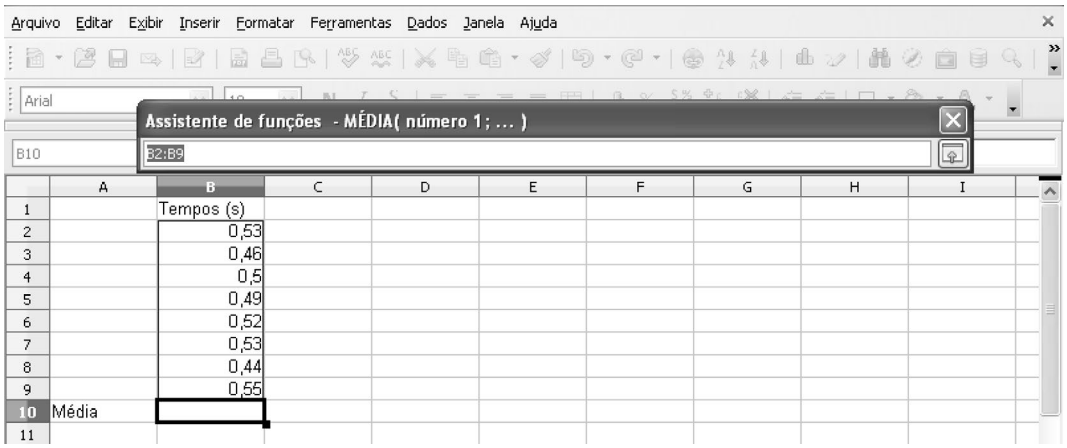
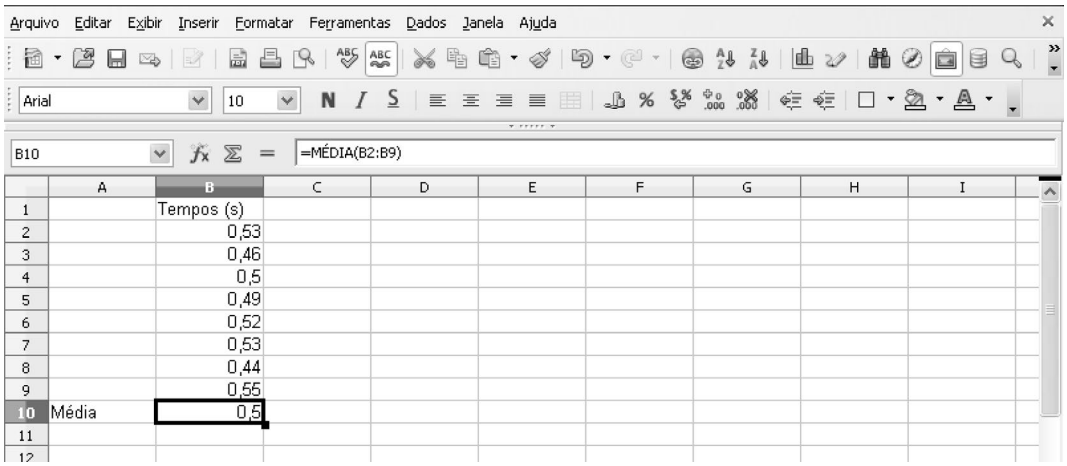


Figura 3.39 Resultado da média.



2) Calcule a mediana para os dados acima;

Posicione o cursor na célula onde deseja inserir a mediana, execute os mesmos passos acima descritos, porém escolha a função `MED` (Figura 3.40).

3) Calcule a moda para os dados acima;

Posicione o cursor na célula onde se deseja inserir a moda, execute os mesmos passos acima descritos, porém escolha a função `MODO` (Figura 3.41). Ao fim desse processo, todas as células das medidas de tendência central devem estar preenchidas (Figura 3.42).

Figura 3.40 Escolhendo função mediana.

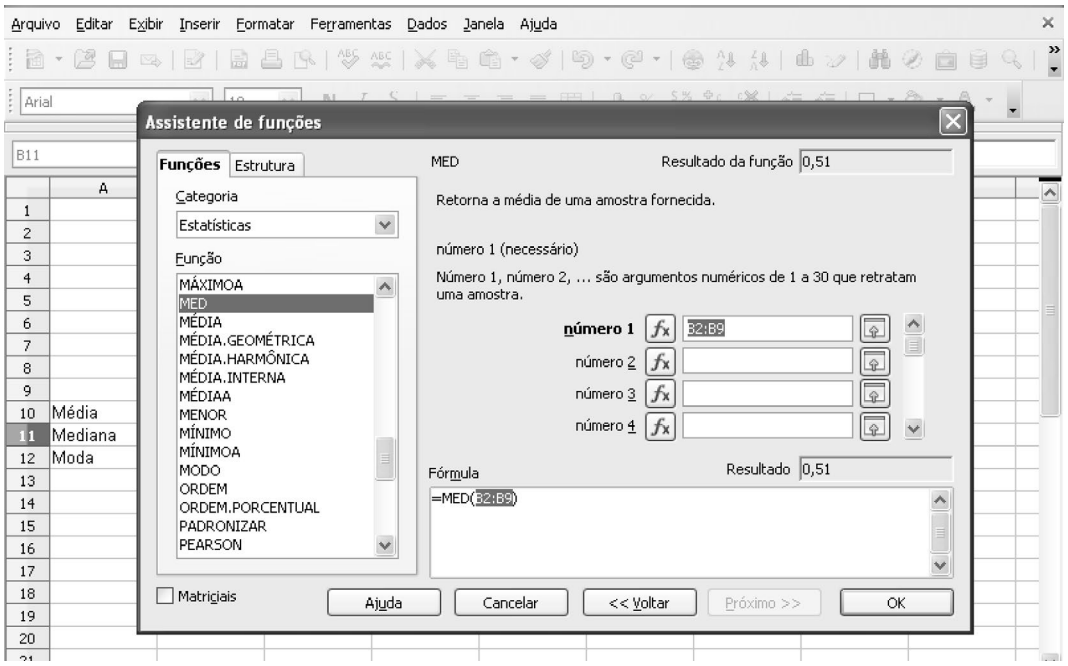


Figura 3.41 Escolhendo a função moda.

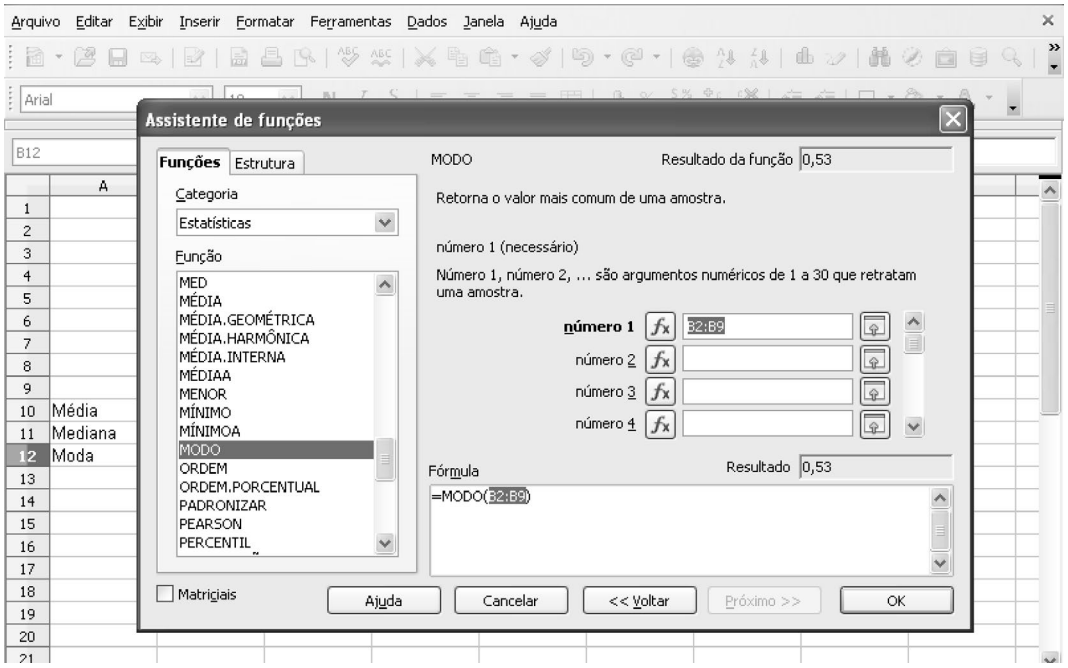
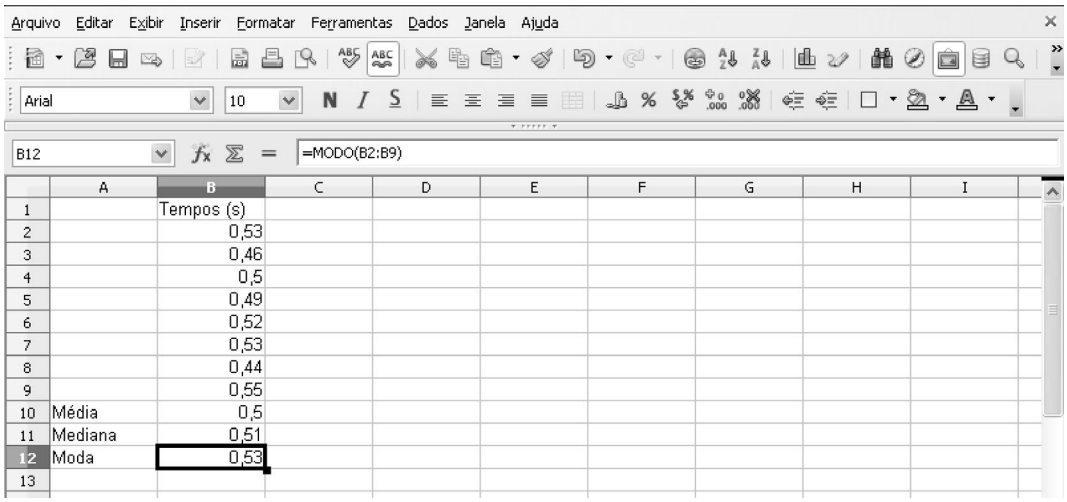


Figura 3.42 Resultados das medidas de tendência central.



	A	B	C	D	E	F	G	H	I
1		Tempos (s)							
2		0,53							
3		0,46							
4		0,5							
5		0,49							
6		0,52							
7		0,53							
8		0,44							
9		0,55							
10	Média	0,5							
11	Mediana	0,51							
12	Moda	0,53							
13									

Atividade 2

Ao longo dos próximos exercícios nós trabalharemos com uma planilha de dados a ser fornecida durante a aula. A planilha contém dados de um estudo sobre fatores de risco para o desenvolvimento de doença arterial coronariana (uma condição em que o fornecimento de sangue para o músculo cardíaco é prejudicado pelo estreitamento dos vasos sanguíneos). Nesse estudo, pacientes do sexo masculino foram diagnosticados quanto à presença de doença coronariana, sendo registrados também alguns parâmetros clínicos. A planilha com a qual trabalharemos contém os seguintes parâmetros:

- id: identificador numérico do paciente;
- idade em anos;
- pressão sistólica;
- tabaco: medida acumulada do uso;
- histórico familiar de doença coronariana;
- álcool: consumo do paciente à época do estudo;
- doença coronariana: diagnóstico (0: não apresenta doença, 1: apresenta doença).

Abra o arquivo da planilha usando o aplicativo LibreOffice Calc. O aspecto dos dados carregados deve ser semelhante ao visto na Figura 3.43. Agora nós vamos calcular medidas de tendência central e de espalhamento para alguns dos parâmetros contínuos presentes na planilha. Essas medidas serão inseridas na própria planilha, abaixo da coluna do parâmetro respectivo. Antes de calcular, vamos rotular algumas linhas da planilha para identificar cada medida. Na primeira coluna da planilha, a partir de alguma linha abaixo dos dados amostrais, insira os seguintes rótulos, um em cada linha (ver Figura 3.44 para ilustração): *média*, *variância*, *desvio padrão* e *mediana*.

Figura 3.43 Aparência da planilha de dados logo após ser carregada.

	A	B	C	D	E	F	G	H
1		id	idade	pressão sistólica	tabaco	histórico familiar	álcool	doença coronariana
2		1	52	160	12	Presente	97,2	1
3		2	63	144	0,01	Ausente	2,06	1
4		3	46	118	0,08	Presente	3,81	0
5		4	58	170	7,5	Presente	24,26	1
6		5	49	134	13,6	Presente	57,34	1
7		6	45	132	6,2	Presente	14,14	0
8		7	38	142	4,05	Ausente	2,62	0
9		8	58	114	4,08	Presente	6,72	1
10		9	29	114	0	Presente	2,49	0
11		10	53	132	0	Presente	0	1
12		11	60	206	6	Ausente	56,06	1
13		12	40	134	14,1	Presente	0	1
14		13	17	118	0	Ausente	0	0
15		14	15	132	0	Ausente	0,97	0

Figura 3.44 Atribuindo rótulos a linhas onde as medidas serão inseridas.

	A	B	C	D	E	F	G	H
1		id	idade	pressão sistólica	tabaco	histórico familiar	álcool	doença coronariana
37		36	28	122	4,26	Ausente	48,99	1
38		37	32	140	3,9	Ausente	36,77	0
39		38	46	110	4,64	Ausente	15,22	0
40		39	29	130	0	Presente	0	0
41		40	58	136	11,2	Presente	22,94	1
42	média							
43	variância							
44	desvio padrão							
45	mediana							
46								

Agora nós calcularemos as medidas estatísticas para o parâmetro *pressão sistólica*, começando com a média, seguindo os mesmos procedimentos da Atividade 1. Selecione a célula da tabela no cruzamento da linha rotulada "média" e da coluna "pressão sistólica". Agora clique sobre o botão do assistente de funções, localizado na barra acima dos dados da planilha (representado por *f*(x)). Um outro meio de acionar o assistente de funções é pelo menu `Inserir >> Função`. A tela do assistente será exibida; selecione a categoria de funções estatísticas e, em seguida, selecione a função `MÉDIA` dentro da lista de funções; pressione então o botão "Próximo".

Na próxima tela, nós vamos selecionar a faixa de valores usados para o cálculo da média. Clique sobre a planilha e selecione todos os dados numéricos da coluna "pressão sistólica", a faixa de valores selecionada aparece no campo denominado "número 1" na tela do assistente. Note que em vez de selecionar a faixa de valores com o mouse, poderíamos digitar diretamente a faixa pretendida, dando o nome da primeira e da última célula, separado por dois pontos. Por exemplo, D2:D41. Para finalizar, pressione o botão "Ok" na tela do assistente e observe que o valor da média é inserido na célula desejada. Você pode agora seguir o mesmo procedimento para calcular as outras medidas do parâmetro "pressão sistólica", alterando o nome da função usada. A seguir, associamos cada medida com a função correspondente do LibreOffice Calc:

- média: função `MÉDIA`;
- variância: função `VAR`;
- desvio padrão: função `DESVPAD`;
- mediana: função `MED`.

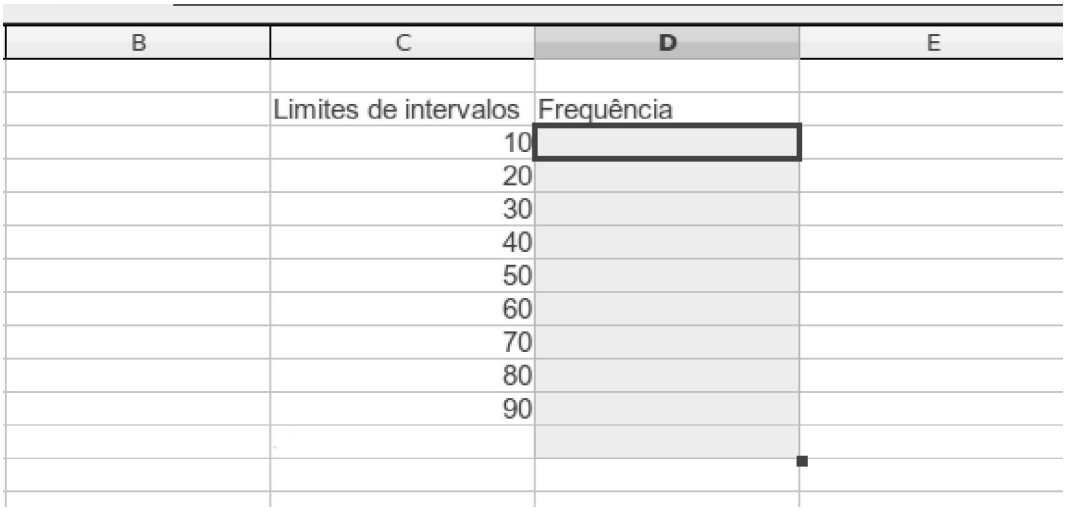
Compare o valor da média obtida com o valor da mediana, eles são semelhantes?

Repita o cálculo das medidas média, desvio padrão e mediana para o parâmetro "álcool". Compare os resultados da média e da mediana. É possível deduzir algo sobre a distribuição deste parâmetro a partir da comparação destes valores?

Atividade 3

Para entender melhor a distribuição do parâmetro "álcool", nós vamos construir um histograma. Para isso, é preciso dividir a faixa de valores exibida pelo parâmetro em um número razoável de intervalos. Como o menor valor exibido é 0 e o maior é 97,2, vamos adotar, como primeira tentativa, a divisão em 10 intervalos de largura 10. Ou seja, nossos intervalos serão: <10, 10 |– 20, 20 |– 30, 30 |– 40, 40 |– 50, 50 |– 60, 60 |– 70, 70 |– 80, 80 |– 90 e ≥90. No LibreOffice Calc, antes de se construir o histograma é preciso determinar quantos pontos pertencem a cada intervalo usando a função `FREQUÊNCIA`. Essa função precisa de dois argumentos: uma coluna de dados a ser analisados e uma coluna contendo os valores que delimitam os intervalos. Antes de prosseguir, vamos inserir uma nova folha em nossa planilha de trabalho, evitando que os dados do histograma interfiram com os dados do estudo. Para isso, use a opção de menu `Inserir >> Planilha`. Na nova folha de planilha, vamos criar uma coluna composta pelos valores superiores de cada intervalo, em ordem: 10,20,30,40,50,60, 70, 80, 90. Agora, devemos selecionar a coluna (ainda vazia) à direita da coluna dos intervalos (Figura 3.45).

Figura 3.45 Selecionando a coluna onde os valores de frequência serão inseridos.



Devemos agora abrir o assistente de funções, selecionar o grupo de funções *Matriciais* e escolher a função `FREQUÊNCIA` (Figura 3.46), pressionado o botão "Próximo" a seguir. Na tela seguinte, vamos selecionar os dados. Clique sobre o campo "dados" e em seguida retorne para a planilha 1, onde estão os dados do estudo. Selecione a coluna do parâmetro "álcool" (Figura 3.47). Ainda na mesma tela, clique agora sobre o campo "classes", selecionando-o. Volte para a folha da planilha que contém os limites dos intervalos e selecione os valores de 10 até 90 (Figura 3.48). O LibreOffice Calc automaticamente cria uma contagem para a faixa de valores >*N*, em que *N* é o último valor de "classe" selecionado neste passo. Agora finalizamos o cálculo das

frequências, clicando o botão "Ok". O programa deve então preencher a coluna de frequências selecionada no início.

Figura 3.46 Selecionando a função frequência.

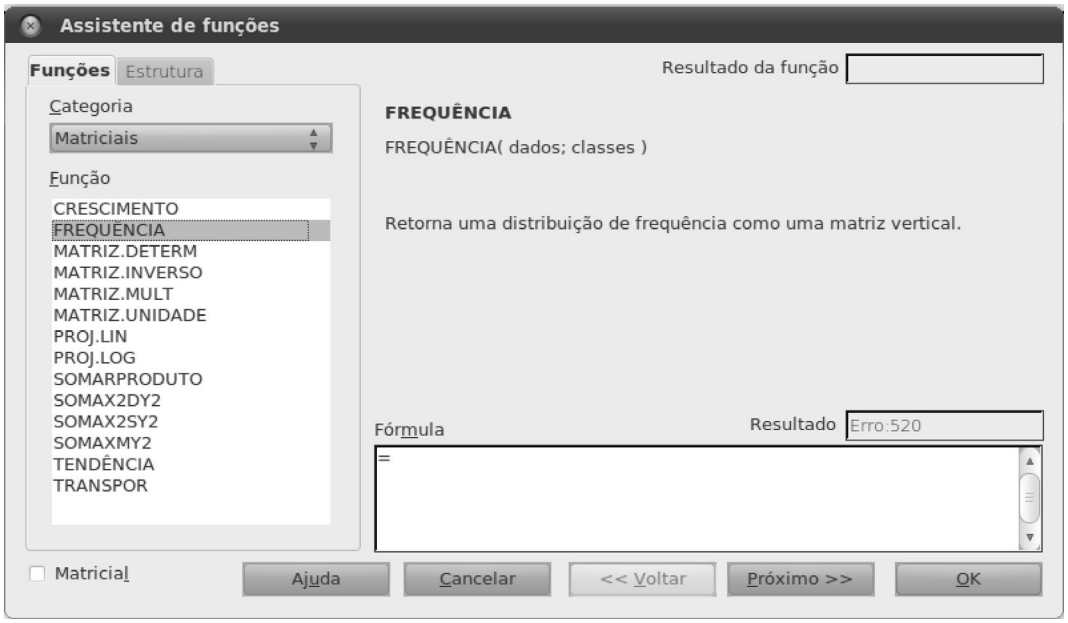
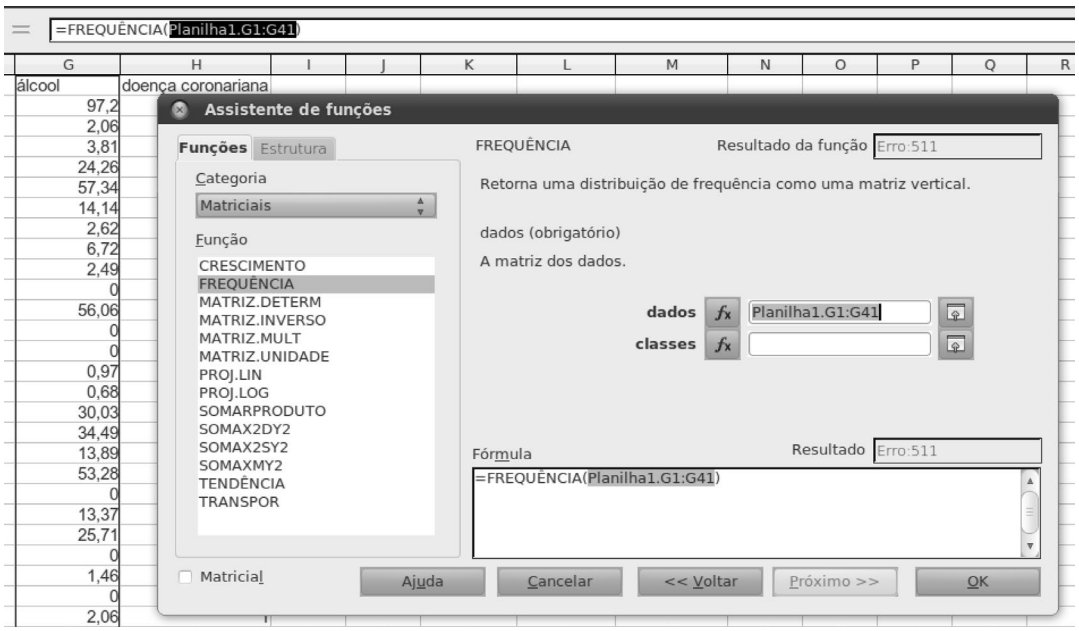


Figura 3.47 Selecionando dados para a construção do histograma.



Finalmente, vamos construir um gráfico de barras das frequências, representando o histograma. Para tanto, o Calc precisa de uma série de rótulos para atribuir a cada barra. Nós poderíamos usar os próprios limitantes dos intervalos, mas isso não expressaria com clareza que estamos tratando de grandezas intervalares. Em vez disso, vamos usar uma coluna extra de rótulos, usando a notação de intervalos introduzida anteriormente (Figura 3.49).

Figura 3.48 Selecionando limites dos intervalos (classes) do histograma.

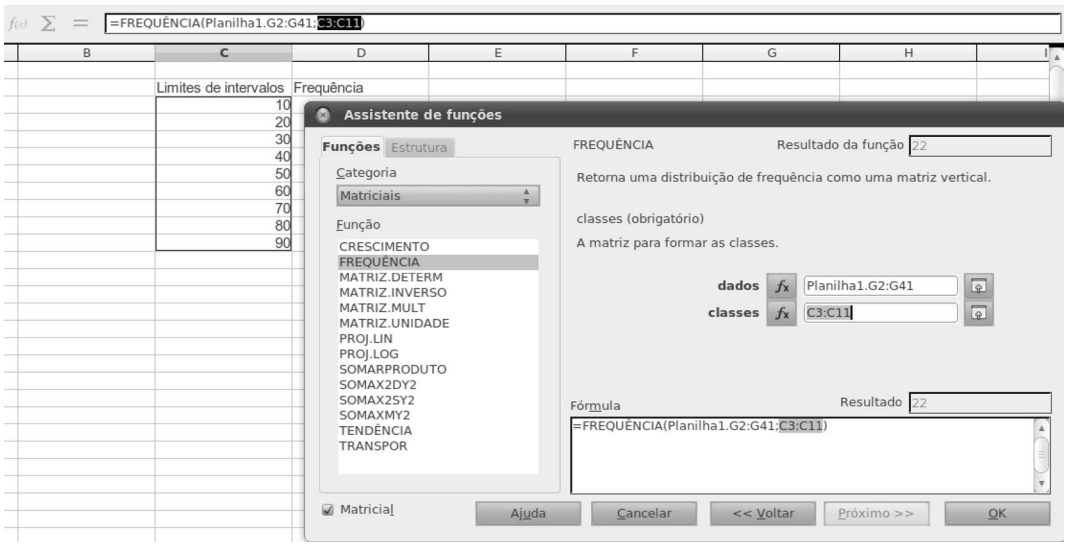


Figura 3.49 Inserindo coluna de rótulos.

	Intervalos	Limites de intervalos	Frequência
	0 |- 10	10	22
	10 |- 20	20	5
	20 |- 30	30	5
	30 |- 40	40	3
	40 |- 50	50	1
	50 |- 60	60	3
	60 |- 70	70	0
	70 |- 80	80	0
	80 |- 90	90	0
	90 |- 100		1

Agora selecione a coluna das frequências e clique em `Inserir >> Gráfico`; selecione o gráfico do tipo coluna (Figura 3.50). Clique em "Próximo", na tela do segundo passo ("Intervalo de dados") não há nada a ser alterado. Clique em "Próximo" novamente. Na tela de "Série de dados", clique sobre o campo "Categorias" e selecione a coluna de rótulos (Figura 3.51). Finalmente, clique em "Próximo", insira um título adequado para o histograma e clique em "Concluir" (Figura 3.52). Deve ser inserido na planilha um gráfico como o da Figura 3.53. Note que o programa *Calc* desenha as barras com um espaço vazio entre elas, mas, conceitualmente, tal espaço não existe, os intervalos que formam o histograma são contíguos.

Figura 3.50 Selecionando tipo de gráfico.

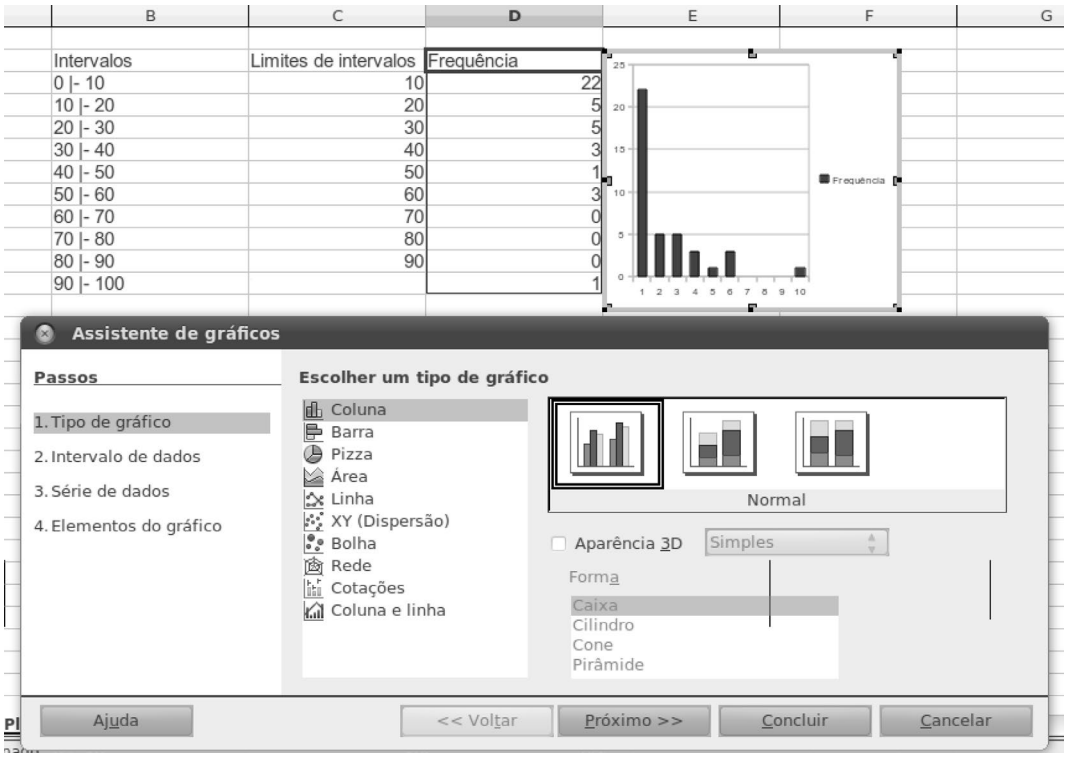


Figura 3.51 Selecionando coluna de rótulos.

Figura 3.52 Inserindo título.

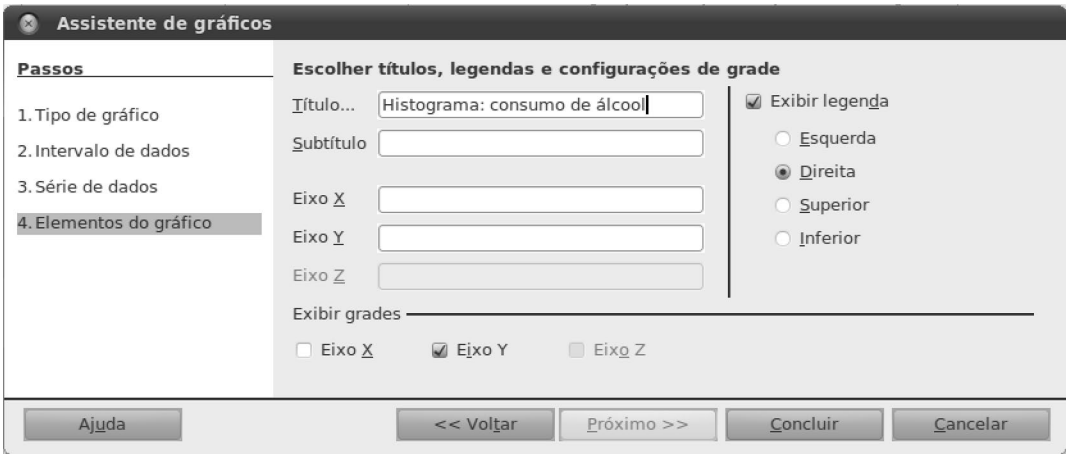
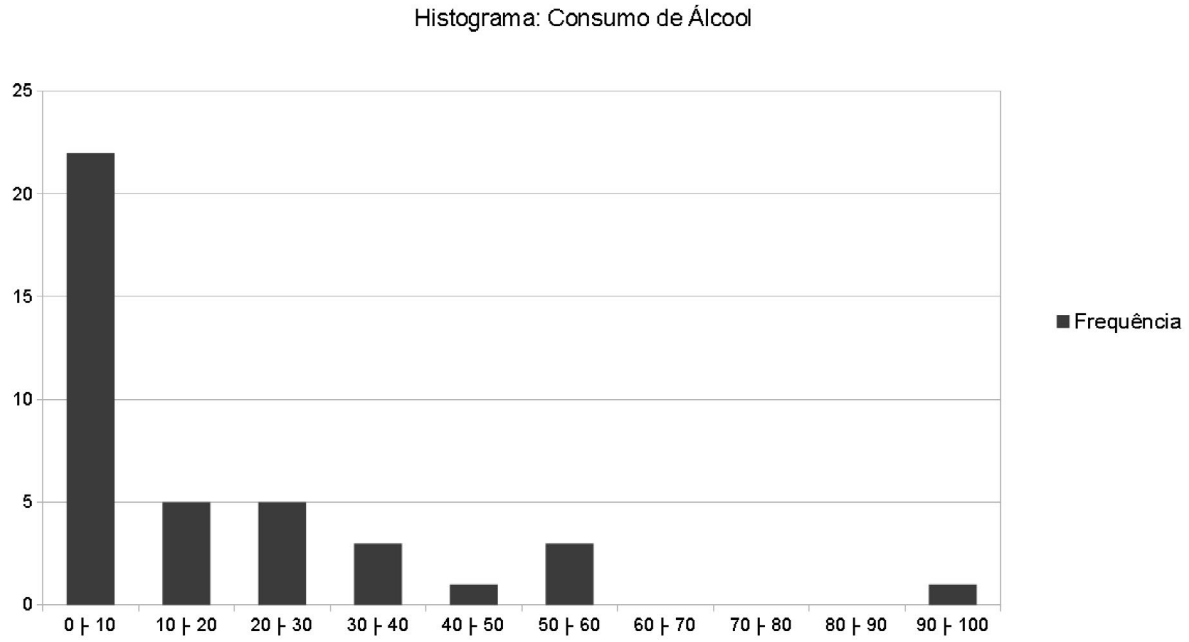


Figura 3.53 Histograma final.



3.5 CONSIDERAÇÕES FINAIS

Neste capítulo, conceitos básicos de Estatística foram apresentados, mostrando como a variabilidade dos dados com os quais trabalhamos gera uma incerteza sobre os resultados obtidos. Os métodos estatísticos nos permitem tirar conclusões e fazer inferências úteis, apesar da variabilidade e da incerteza. O uso desses métodos envolve muitas outras questões que não foram abordadas neste capítulo. Por exemplo, a Estatística nos fornece métodos para dimensionar amostras a fim de obter resultados com determinada precisão, para quantificar a incerteza dos resultados, para comparar resultados obtidos com diferentes amostras, etc. Essas questões devem ser abordadas em um curso específico de Estatística. Um dos pontos cruciais para um bom trabalho de análise estatística é a qualidade dos dados utilizados. Nenhum truque de manipulação matemática é capaz de corrigir uma coleta ruim de dados e, por isso, devemos assegurar que os dados medidos sejam relevantes para o problema a ser estudado, que existam em quantidade suficiente para tirarmos conclusões significativas e que estejam bem organizados. No próximo capítulo, trataremos da estruturação e organização de dados em *bases de dados*, apresentando ferramentas gerais para gerenciar os dados coletados e facilitar o trabalho de raciocinar sobre eles.

3.6 EXERCÍCIOS

1. Em uma pesquisa, uma amostra de pessoas respondeu a um questionário em que as seguintes variáveis foram medidas:

Ano de nascimento, cidade de origem, sexo, renda familiar, escolaridade (número de anos de estudo), número de filhos, peso, altura, cor dos olhos.

Classifique as variáveis acima nos seguintes grupos: Categóricas (nominal ou ordinal) ou Numéricas (discreta ou contínua).

2. Para uma determinada localidade, a Tabela 3.12 mostra a distribuição dos indivíduos segundo o grau de escolaridade e o local de residência, em uma amostra adquirida.

Tabela 3.12 Dados para o Exercício 2. Distribuição de escolaridade segundo tipo de residência.

Grau de Escolaridade	**Local de Residência**	
	Urbano	**Rural**
Primário	5025	2580
Secundário	3155	285
Superior	1720	20

Para cada um dos tipos de residência, calcule:

a) Proporção de indivíduos em cada grau de escolaridade.

b) Porcentagem de indivíduos em cada grau de escolaridade.

c) Razões entre números de indivíduos: Primário/Secundário, Primário/Superior e Secundário/Superior.

Com base nos valores calculados, você diria que o tipo de residência é um fator que afeta o grau de escolaridade?

3. A Tabela 3.13 mostra a distribuição das pessoas a bordo do navio *Titanic* em sua viagem inaugural, divididos de acordo com 3 categorias (Criança, Mulher e Homem) e o tipo de cabine (Primeira/Segunda/Terceira classe ou Tripulação). Para cada combinação de (Categoria, Tipo de cabine), a tabela mostra o número de pessoas que conseguiram (ou não) sobreviver ao naufrágio. Por meio do cálculo de taxas de sobrevivência, investigue as seguintes hipóteses:

a) As crianças e mulheres tiveram preferência no resgate.

b) O tipo de cabine não influenciou decisivamente a chance de sobrevivência dos ocupantes do navio.

Tabela 3.13 Dados para o Exercício 3. Distribuição dos sobreviventes do *Titanic* de acordo com categoria e tipo de acomodação.

Categoria	**Tipo de cabine**	**Sobrevivência**	
		Não	***Sim***
Criança	Primeira Classe	1	5
	Segunda Classe	0	24
	Terceira Classe	52	27
Mulher	Primeira Classe	4	140
	Segunda Classe	13	80
	Terceira Classe	89	76
	Tripulação	3	20
Homem	Primeira Classe	118	57
	Segunda Classe	154	14
	Terceira Classe	387	75
	Tripulação	693	192

4. A Tabela 3.14 traz as distâncias (em metros) obtidas pelos vencedores da prova de salto triplo masculino das Olimpíadas entre os anos de 1936 e 2004 (não houve Jogos Olímpicos em 1940 e 1944 devido à II Guerra Mundial). Para esses dados:

a) Faça o gráfico de dispersão da distância do salto em função do ano. Você acha que a relação entre as duas grandezas é aproximadamente linear?

b) Calcule o coeficiente de correlação entre distância e ano.

c) Ajuste uma reta de regressão para os dados apresentados e calcule o coeficiente de determinação. Desenhe a reta de regressão sobre o gráfico de dispersão anterior.

d) Em 1932, o vencedor do salto triplo atingiu a marca de 15,72m. Em 2008, o vencedor obteve 17,67m. Use a reta ajustada anteriormente para estimar a distância para cada um desses anos. Compare os valores estimados e os realmente obtidos.

Tabela 3.14 Dados para o Exercício 4. Distâncias obtidas pelos vencedores do salto triplo masculino em Olimpíadas.

Ano	Distância (m)	Ano	Distância (m)
1936	16,00	1976	17,29
1948	15,40	1980	17,35
1952	16,22	1984	17,25
1956	16,35	1988	17,61
1960	16,81	1992	17,17
1964	16,85	1996	17,58
1968	17,39	2000	17,71
1972	17,35	2004	17,79

5. Em um sítio da Internet para comparação entre produtos, os consumidores podem atribuir notas entre 1 e 5 para itens que tenham adquirido. A Tabela 3.15 mostra a distribuição das notas para dois modelos de câmeras fotográficas.

a) Para cada modelo de câmera, calcule a média e a variância das notas atribuídas. Com base nesses dois valores, você diria que as duas câmeras têm avaliações semelhantes?

b) Calcule a mediana das notas atribuídas para cada modelo de câmera.

c) Faça um histograma das distribuições das notas para cada modelo de câmera.

Tabela 3.15 Dados para o Exercício 5. Para cada valor de nota, a tabela registra o número de consumidores que atribuíram aquela nota ao produto.

Nota	Câmera A	Câmera B
1	3	9
2	8	3
3	12	1
4	6	8
5	4	7

6. Para os dados da Tabela 3.16, faça os seguintes itens:

a) Calcule média e desvio padrão para os dados contidos na coluna y_1. Repita para y_2, y_3 e y_4.

b) Construa o gráfico de dispersão de y_1 em função de x_a. Calcule a reta de regressão para y_1 em função de x_a. Repita esse procedimento para: y_2 em função de x_a; y_3 em função de x_a; y_4 em função de x_b. Calcule o coeficiente de regressão e o coeficiente de determinação (R^2) em cada um dos casos.

c) Examinando as retas de regressão calculadas no item anterior, em quais casos você acredita que a reta estimada é adequada?

Tabela 3.16 Dados para o Exercício 6. Análise de regressão.

x_a	x_b	y_1	y_2	y_3	y_4
10	8	8,04	9,14	7,46	6,58
8	8	6,95	8,14	6,77	5,76
13	8	7,58	8,74	12,74	7,71
9	8	8,81	8,77	7,11	8,84
11	8	8,33	9,26	7,81	8,47
14	8	9,96	8,10	8,84	7,04
6	8	7,24	6,13	6,08	5,25
4	19	4,26	3,10	5,39	12,50
12	8	10,84	9,13	8,15	5,56
7	8	4,82	7,26	6,42	7,91
5	8	5,68	4,74	5,73	6,89

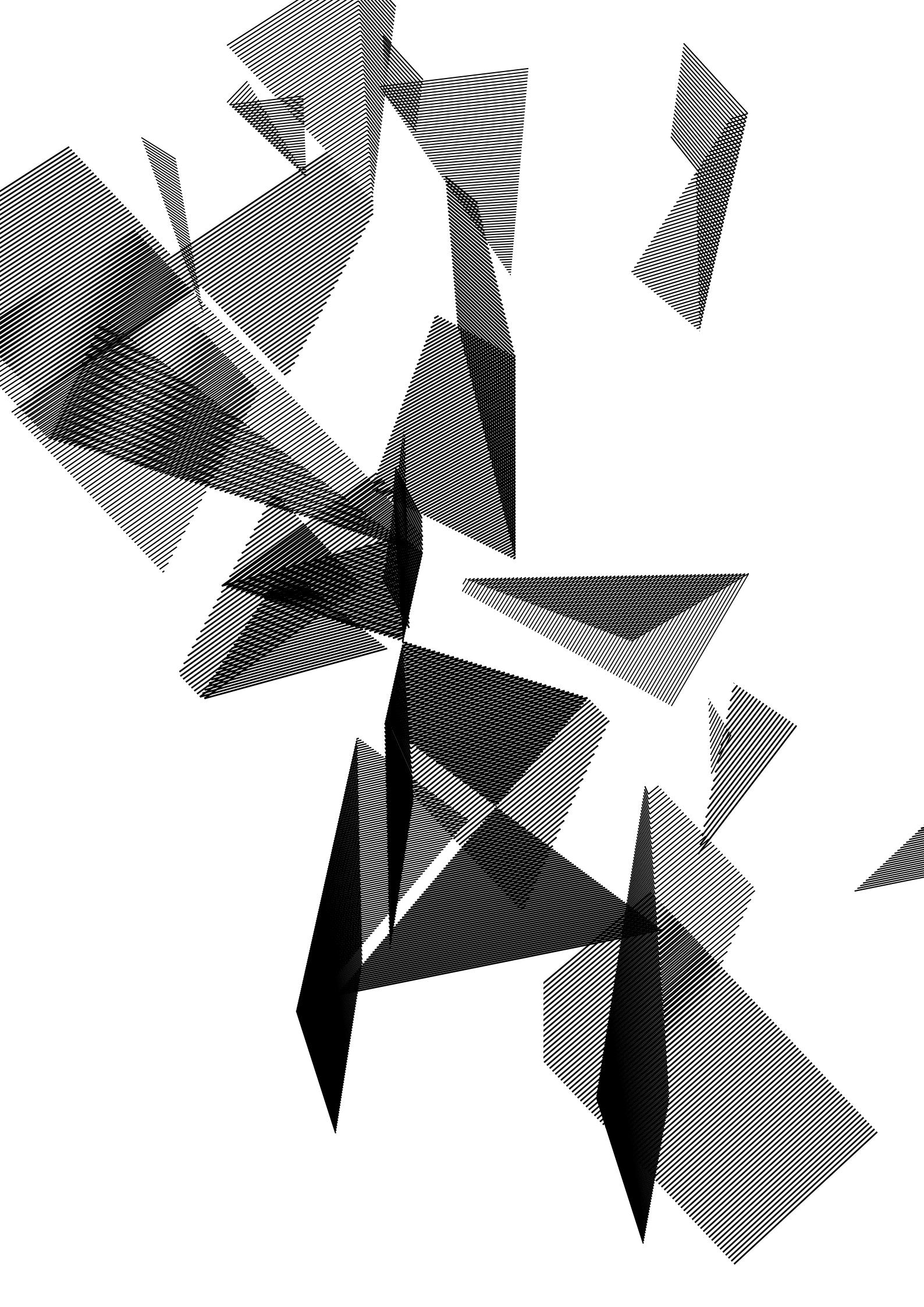
REFERÊNCIAS BIBLIOGRÁFICAS

LARSON, R. e FARBER, B. 2ª edição. *Estatística aplicada*. São Paulo: Pearson Prentice Hall, 2007.

STEELE, J. e ILIINSKY, N. (ed.). *Beautiful Visualization*. Sebastopol, California, EUA: O'Reilly, 2010.

STEVENSON, W. & STEVENSON, J. *Estatística aplicada à administração*. São Paulo: Habra, 2001.

TUFTE, E. R. *The Visual Display of Quantitative Information*. Cheshire, Connecticut, EUA: Graphics Press, 1983.



CAPÍTULO 4

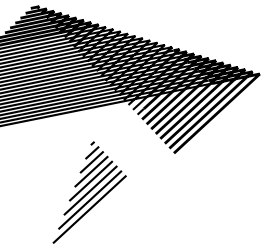
BASE DE DADOS

Irineu Antunes Jr.
Márcio Katsumi Oikawa
Humberto Luiz Razente
Maria Camila Nardini Barioni

4.1 INTRODUÇÃO

Nos tempos atuais, convivemos com a necessidade constante de interação e manipulação de dados. Muitas atividades rotineiras estão invariavelmente ligadas a operações diversas de gerenciamento de dados, tais como tratamento, armazenamento, apresentação, interpretação, geração ou recuperação. Após a revolução causada pela Internet e pelos recursos de rede, o acesso à informação tornou-se mais fácil, amplo e democrático. Isso também provocou um sério e crescente problema de organização de dados, evidenciando a necessidade de oferecer melhores maneiras de tratá-los e potencializar o seu aproveitamento por usuários comuns, grupos ou mesmo grandes instituições.

A utilização correta e eficiente de dados nem sempre é tarefa simples. Em muitas situações, constitui-se numa operação mais delicada e difícil do que inicialmente aparenta. Esse raciocínio se aplica tanto a grandes quanto a pequenos conjuntos de dados. A falta de planejamento e organização pode expor a base de dados a vários problemas, entre os quais a coleta ineficiente de dados e a presença de inconsistências sobre o conjunto de dados. Tais problemas normalmente são causados quando não há uma séria preocupação com o detalhamento dos dados que serão usados para algum fim posterior. Sendo assim, normalmente não é uma boa política guardar dados de forma desorganizada, sem saber exatamente como eles serão usados.



Outro aspecto diz respeito ao tratamento propriamente dito dos dados. Uma vez definidas as características dos dados, como construir uma estrutura organizada capaz de aproveitá-los ao máximo? De que modo usar os dados armazenados em operações futuras? De que forma controlar e garantir a qualidade desses dados?

Neste capítulo, procuraremos entender melhor a relação existente entre dados e informações, e apresentaremos conceitos importantes para a construção de bases de dados simples. Durante o capítulo, serão apresentadas características importantes de bases de dados que as tornam muito úteis do ponto de vista computacional: organização, armazenamento, recuperação e visualização. *O objetivo deste capítulo é, assim, introduzir a ideia de bases de dados, associando-a com o ambiente de uso de computadores e sistemas de software.* Muitos dos conceitos abordados neste capítulo permitirão compreender a importância da escolha das formas de representação de

dados, do controle de qualidade da base de dados, da organização dos dados e do planejamento voltado para uma correta utilização futura.

É importante ressaltar que o tratamento eficiente de grandes conjuntos de dados não se restringe aos assuntos abordados neste capítulo. Para grandes bases de dados comerciais ou governamentais, existe uma decisiva preocupação com outras características, que vão além da simples organização de dados. Nesses casos, também se buscam modelos de dados mais eficientes, linguagens de integração de bases de dados com sistemas computacionais, compartilhamento de dados por vários grupos de usuários, controle de usuários, segurança de acesso, ferramentas de recuperação de falhas, etc. *O objetivo deste capítulo NÃO é discutir formas mais avançadas de modelagem de dados (como a modelagem relacional, por exemplo) e nem entrar em detalhes sobre qualquer dos temas citados no início deste parágrafo, diretamente ligados a operações de Sistemas de Gerenciamento de Bancos de Dados (SGBD).* Esses temas, apesar de muito importantes, só serão tratados em disciplinas específicas, em estágios mais avançados dos cursos da área de Computação. O desenvolvimento desses temas depende de uma experiência com assuntos específicos da área, que fogem aos objetivos desta disciplina. Para os interessados, entretanto, há uma vasta coleção de livros que aborda o assunto com alto nível de detalhes, entre os quais podemos destacar Elmasri e Navathe (2006); Silberschatz *et al.* (2006) e Gillenson *et al.* (2009).

O restante deste capítulo está distribuído conforme a seguinte organização. A Seção 4.2 aborda uma introdução aos conceitos básicos de bases de dados, que são reforçados na Seção 4.3, por meio de um exemplo de organização dos dados em uma base de dados. A Seção 4.4 apresenta noções de arquivos estruturados, muito usados para trocar dados entre diferentes bases de dados. Na Seção 4.5 são discutidos conceitos relacionados a consultas em bases de dados. Os exercícios da aula prática são apresentados na Seção 4.6, e exercícios complementares estão disponíveis na Seção 4.8. Por fim, a Seção 4.7 apresenta as considerações finais do capítulo.

4.2 ENTENDENDO A BASE DE DADOS

4.2.1 DADOS E INFORMAÇÕES

Embora rotineiramente tratados como sinônimos, os *dados* e as *informações* possuem significados bem diferentes do ponto de vista de processamento computacional. Em linhas gerais, chamamos de *dado* (ou *dado bruto*) qualquer elemento que possa ser processado por um computador, tal como um conjunto de números, cadeias de caracteres, imagens, códigos de barras, senhas criptografadas, etc. Chamamos de *informação* a interpretação dada a um conjunto de dados, tornando-os significativos para algum contexto do mundo real. A compreensão dessa diferença, embora simples, permitirá a potencialização do uso dos dados, pensando-os e planejando seu uso de forma mais organizada.

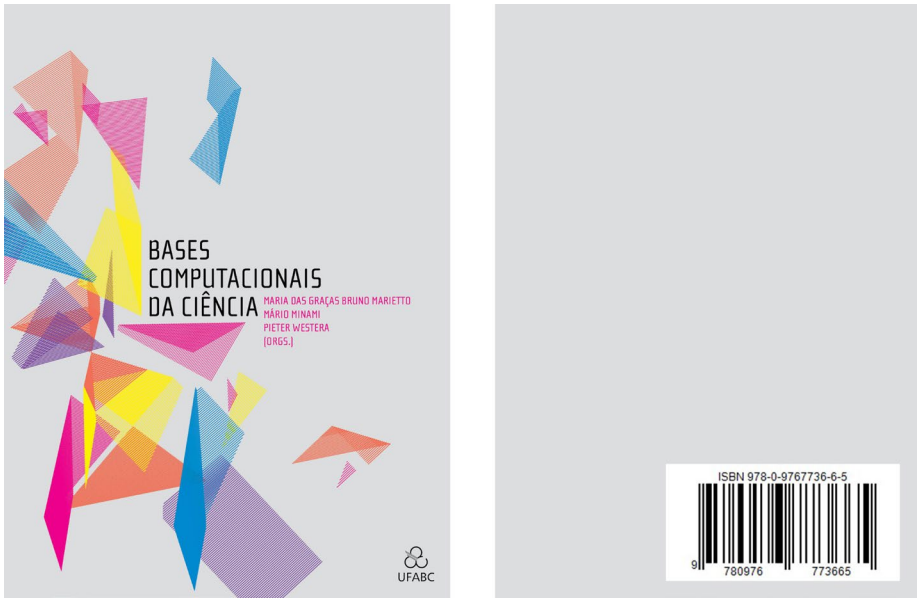
Para tornar essa definição mais clara, vamos imaginar o seguinte cenário. Em muitas livrarias o processo de cobrança dos caixas é automatizado, realizado por meio de

códigos de barras. O código de barras é uma sequência de barras verticais claras e escuras de largura (possivelmente) variada. A leitura do código de barras é realizada por um tipo especial de *scanner* que o interpreta e mapeia para números ou letras, que serão tratados pelo sistema de computador (veja a Figura 4.1). Como os códigos de barras podem ser manipulados computacionalmente, podemos chamá-los de *dados*.

Figura 4.1: Exemplo de código de barras.

Apesar da forma válida de representação, um código de barras, isoladamente, tem pouco (ou nenhum) significado quando dissociado do objeto físico que representa. O código da Figura 4.1 pertence a um livro específico das prateleiras de uma livraria. Dessa forma, a *informação* representada pelo código é, nesse caso, o livro na Figura 4.2. Todas as vezes que o computador processar alguma operação relacionada ao código de barras da Figura 4.1, estará na realidade, processando a operação para o livro da Figura 4.2!

Figura 4.2: Livro representado pelo código de barras.



Com esse simples exemplo, podemos verificar que a *informação* é o elemento essencial e significativo que desejamos utilizar e manipular dentro de algum contexto. O *dado* é a representação computacional de alguma informação. Sendo assim, em termos gerais, o computador só é capaz de manipular *dados*, que se transformam em *informações* quando interpretados por seus usuários. Usa-se o termo *base de dados* para indicar

um aglomerado, ou conjunto, de dados que deve ser manipulado pelo computador. Em muitas referências, você encontrará o termo *Banco de Dados* como definição para os conjuntos de dados que discutimos em sala de aula. Evitaremos utilizar esse termo por entender que, apesar de amplamente difundido, está informalmente associado com *Bancos de Dados Relacionais*, que não serão discutidos neste documento.

4.2.2 O QUE É UMA BASE DE DADOS?

Uma *base de dados* pode ser definida como qualquer conjunto de dados no qual é possível realizar três operações sobre seus elementos: armazenamento, recuperação e visualização. Essa definição é válida não apenas para o contexto de sistemas computacionais, mas para qualquer ambiente do mundo real. Baseados nisso, vamos construir um exemplo de fácil analogia para fixar esses conceitos. Vamos explorar a ideia de uma *lista de supermercado*, ilustrada na Figura 4.3.

Figura 4.3: Lista de supermercado.

Primeiro, vamos responder à seguinte pergunta:

Por que as pessoas fazem listas de supermercado?

A resposta mais natural seria algo como:

Para se lembrar de todos os itens que devem ser comprados.

Apesar da simplicidade do exemplo, ele nos permite identificar as principais características que toda base de dados deve possuir. Vamos a elas:

- *Armazenamento* (Persistência): As listas de supermercado podem ser armazenadas tanto em mídias simples (como papel) quanto em mídias digitais

gerenciadas por dispositivos como *palms*, *tablets*, telefones celulares, entre outros. O objetivo de todas essas mídias é preservar de forma confiável as informações em um local seguro para futura utilização. Em outras palavras, dados armazenados não devem ser perdidos! Dados só devem ser removidos da base quando propositalmente excluídos. Dizemos que, nessa situação, o armazenamento é *persistente*, indicando que os dados armazenados não são voláteis e sua guarda é controlada pelo usuário da base de dados. Mais à frente, veremos alguns conceitos e mecanismos interessantes usados para melhorar a qualidade do processo de armazenamento de dados (Seções 4.3 e 4.4).

- *Recuperação*: No supermercado, a lista cumpre a sua principal função: lembrar os itens que devem ser comprados. Um dos principais objetivos de qualquer base de dados é recuperar informações previamente cadastradas de forma rápida e precisa, de modo que o usuário tenha completa segurança de que tudo o que fora cadastrado não foi perdido e está à disposição para consulta a qualquer momento (Seção 4.5.1).

- *Visualização*: A visualização trabalha a forma como os dados recuperados são exibidos. Imagine, por exemplo, que a lista seja mostrada em ordem alfabética. Se os itens da lista forem procurados nessa ordem, o usuário provavelmente andará em círculos pelo supermercado. Isso ocorre porque a lista ignora a organização dos itens em setores dentro do supermercado (setores de alimentos congelados, vestuário, bebidas, doces, etc.). Uma lista que exiba os itens agrupados nos setores do supermercado poderia melhorar sensivelmente a sua busca nas prateleiras. Esse exemplo ilustra como a visualização pode facilitar ou dificultar o trabalho de quem usa a informação, de forma que sempre deve ser considerado no planejamento da base de dados (Seção 4.5.2).

Normalmente, as bases de dados são usadas para problemas muito maiores e mais complexos do que o armazenamento de listas de supermercado. Não é difícil imaginar alguns deles, como controlar a lista de clientes de uma empresa, de correntistas de um banco, de alunos de uma universidade, de livros de uma biblioteca, de contribuintes de uma unidade governamental, etc. Para todos esses exemplos, a base de dados exerce um papel fundamental na condução e no planejamento das atividades, pois é ela que informa o estado atual da informação, acompanha o desenvolvimento das tarefas e fornece embasamento para a tomada de decisões.

Para complexos conjuntos de dados, que envolvem um grande volume de informação e uma grande quantidade de usuários, também é desejável que a base possa apresentar outras importantes características:

- *Compartilhamento*: Capacidade de troca de informações com outros usuários ou outros sistemas computacionais. É comum o uso de bases de dados com informações relevantes a diferentes grupos de usuários, que podem trocar informações direta ou indiretamente por meio dos dados presentes na base. Voltando ao exemplo da lista de supermercado, considere que, para agilizar a compra do mês, cada membro da família será responsável por comprar parte dos itens da lista, atualizando na lista (removendo) os itens já comprados;

- *Segurança*: Capacidade de fornecer um ambiente seguro para o funcionamento da base de dados. A segurança de dados pode ser interpretada de duas formas distintas: segurança de *infraestrutura* e segurança de *acesso*. A segurança de infraestrutura indica procedimentos relevantes que procuram garantir o funcionamento do *hardware* (local físico) que hospeda a base de dados. A segurança de acesso procura controlar as permissões de acesso de usuários ou grupos de usuários à base de dados (completa ou parcialmente). No exemplo, considere que apenas os adultos da família podem autorizar a inclusão de guloseimas na lista;

- *Regras de consistência*: Capacidade de incorporar regras de validação de dados que auxiliem na garantia de qualidade e coerência da informação armazenada. As regras, em geral, procuram verificar se a correlação de dados é feita de forma correta e prevenir a inclusão de valores inválidos na base. Por exemplo, não permitir a inclusão de itens que já estão na lista (garantia de valores únicos).

Nem sempre é trivial administrar funções de compartilhamento, segurança e consistência em uma base de dados. Em alguns casos, é preciso utilizar sistemas de *software* específicos, ou construir (por meio de programação) regras próprias para cada caso. A necessidade e a viabilidade de cada um desses itens, bem como sua relação de custo e benefício, devem ser avaliadas por uma equipe específica, que poderá ou não indicar sua aplicação. Por constituírem conceitos avançados de administração de bases de dados, apenas apresentamos a sua definição por sua relevância, mas não discutiremos detalhes neste texto. Para o gerenciamento e compartilhamento de grandes volumes de dados, muitas organizações utilizam *softwares* especiais que implementam essas características, chamados Sistemas de Gerenciamento de Bases de Dados (SGBDs).

4.3 ORGANIZAÇÃO DE DADOS

Existem vários modos de organizar uma base de dados. Quando representada em uma planilha, a base de dados pode ser vista como uma *matriz de dados*. O formato matricial possui duas grandes vantagens:

- Oferece uma forma fácil de visualização global dos dados;
- Permite identificar precisamente a posição de cada valor armazenado na base.

É muito comum chamar essa matriz de *tabela*. Essa terminologia é amplamente difundida e aceita no contexto de bases de dados, de modo que também a adotaremos neste texto. Por convenção, as linhas da tabela armazenam itens de dados, enquanto colunas armazenam atributos (propriedades ou características) vinculados a cada item de dado armazenado nas linhas.

Aproveitando o exemplo anterior, imagine que, numa lista de supermercado maior e mais elaborada, deseja-se anotar mais informações sobre os produtos a fim de permitir comparações de preços entre vários supermercados diferentes. O usuário dessa lista poderia ter interesse em anotar as seguintes características de cada item de supermercado:

- *Nome do produto*: Nome usado para identificar o produto pesquisado;

- *Fabricante*: Nome do fabricante do produto pesquisado;
- *Quantidade*: Valor numérico para quantificar o produto pesquisado;
- *Medida*: Unidade de medida usada para classificar o produto, normalmente indicado em sua embalagem. Neste exemplo, assuma as seguintes unidades de medida: *kg* (quilogramas), *l* (litros), *g* (gramas), *u* (unidades) e *p* (pacotes);
- *Preço*: Valor de compra do produto;
- *Supermercado pesquisado*: Nome do supermercado onde foi realizada a pesquisa;
- *Data da pesquisa*: Data de realização da pesquisa.

Dessa maneira, a base de dados poderia ser organizada com as seguintes colunas (atributos), como mostrado na Tabela 4.1.

Tabela 4.1: Atributos da lista de supermercado.

Produto	**Fabricante**	**Qtd.**	**Med.**	**Preço**	**Supermercado**	**Data**

Após uma hipotética visita a vários supermercados, considere que a base de dados foi preenchida com os valores mostrados na Tabela 4.2.

Tabela 4.2: Lista de supermercado preenchida.

Produto	**Fabricante**	**Qtd.**	**Med.**	**Preço**	**Supermercado**	**Data**
Suco	ValeSuco	1	l	3,00	Arpoador	12/05/2011
Suco	Flash	1	l	4,50	Arpoador	12/05/2011
Tomate	-	1	kg	3,50	Noite	14/05/2011
Arroz	Tio José	5	kg	8,64	Noite	14/05/2011
Arroz	Sem Broto	5	kg	9,99	Arpoador	12/05/2011
Arroz	Da TV	1	kg	1,99	Noite	14/05/2011
Feijão	Sem Broto	1	kg	4,00	Arpoador	12/05/2011
Tomate	-	1	kg	2,99	Noite	14/05/2011
Ovo	A Granja	12	u	3,19	Arpoador	12/05/2011
Ovo	Caseiro	6	u	1,45	Noite	14/05/2011
Suco	Flash	1	l	3,99	Noite	12/05/2011
…	…	…	…	…	…	…

O exemplo nos permite perceber características que conferem à base de dados uma boa e desejável organização:

1. Linhas contêm dados relacionados aos produtos (*instâncias* ou *itens de dados*);
2. Colunas contêm os principais atributos dos produtos que se deseja armazenar;
3. Na mesma coluna, só estão presentes dados relacionados ao mesmo atributo, sem exceção!

4. Na mesma linha, só estão presentes dados relacionados ao mesmo item de supermercado, sem exceção!

5. Cada atributo de uma linha contém um único valor ao invés de poder conter uma coleção de valores (atributo *monovalorado*);

6. A base de dados pode crescer em seu número de linhas indefinidamente;

7. Para cada produto pesquisado, o usuário da base de dados tem consigo uma referência de quais informações são relevantes e devem ser armazenadas na base, minimizando problemas de coleta de dados.

4.3.1 TIPOS DE DADOS E DOMÍNIOS

Toda base de dados deve reservar uma especial atenção para a escolha dos tipos dos valores armazenados (*tipos de dados*). A escolha correta dos tipos de dados permite elevar o nível de qualidade da base, na medida em que torna viável controlar possíveis erros de cadastramento de valores. Esse controle busca prevenir inconsistências, tais como o armazenamento de letras em locais destinados exclusivamente a valores numéricos ou datas escritas de forma incorreta.

Apesar de não ter uma rígida exigência sobre o tipo dos dados que armazena, ainda assim a planilha fornece ferramentas para especificação de tipos de dados simples. Entre os mais comuns, podemos destacar:

- Número: Representa valores numéricos de forma geral. Em muitos aplicativos, é possível configurar seus parâmetros a fim de restringir os valores a conjuntos mais específicos, como números inteiros ou fracionários. Ex.: -12,3; 4; +345, 1234...

- Texto: Representa cadeias de caracteres (sequências contínuas de caracteres). Normalmente apresentadas entre aspas duplas (""), as cadeias de caracteres permitem representar informações textuais dentro da base de dados. Ex.: "Joaquim José da Silva Xavier", "Avenida dos Estados, 5001", "7 de setembro" ...

- Moeda: Uma variação do tipo que representa valores numéricos, na qual existe uma parte inteira e uma fracionária com, no máximo, dois dígitos decimais. É bastante útil para trabalhar com operações financeiras. Ex.: -12,34; 56,99; 0,00...

- Data: Representação de uma data no calendário Gregoriano (adotado como referência em muitos países do Ocidente, inclusive o Brasil), com informações sobre dia, mês e ano. Apesar de padronizada em termos de medida de tempo, datas não apresentam padrão de formato, possuindo muitas formas de apresentação em diferentes países. Normalmente, o *software* usado para construir a base de dados possui flexibilidade na escolha desses formatos. No Brasil, é comum representar datas no formato DD/MM/AAAA, com dois dígitos para o dia (DD), dois dígitos para o mês (MM) e quatro dígitos para o ano (AAAA). Ex.: 12/12/2012; 29/02/2000...

- Hora: Representa a divisão de tempo no período de 1 (um) dia. Normalmente, adota-se o formato hh:mm:ss, com dois dígitos para horas (hh), dois para minutos (mm) e dois para segundos (ss). Muitas ferramentas também oferecem alguma

flexibilidade adicional, considerando milissegundos, e indicação de intervalos de 12h ou 24h. Ex.: 12:45:00; 00:00:01...

- Lógico: Representa os valores lógicos VERDADEIRO e FALSO, os únicos valores válidos para esse tipo. Muito usado quando se deseja projetar alguma operação condicional sobre os dados.

Os tipos de dados básicos impõem um controle inicial sobre o comportamento dos valores na base de dados, auxiliando na garantia de sua qualidade. No exemplo da lista de supermercado, a correta indicação dos tipos de dados poderia prevenir a inclusão de valores não-numéricos para o atributo *Qtd.* (claramente numérica por sua definição), ou valores inválidos de datas (fora de formato ou inexistentes) para o atributo *Data*.

Além do tipo de dados, normalmente é possível personalizar regras ainda mais rígidas sobre o comportamento dos valores da base. Essas regras são representadas pela definição de um *domínio*. O *domínio* é o conjunto de valores considerados válidos para um determinado atributo da tabela. Vamos usar novamente o exemplo e indicar os domínios dos atributos *Qtd.*, *Med.* e *Preço*.

Para o atributo *Qtd.*, construímos o seguinte domínio:

$$dom(Qtd.)=\{x \in \mathbb{Z} \; e \; x > 0\}$$

Pela definição, só são considerados válidos para o atributo *Qtd.* valores inteiros positivos. Perceba que a garantia de que a coluna armazenará valores inteiros já é dada pelo tipo de dados, mas o domínio restringe um pouco mais a regra, exigindo que, além de inteiros, sejam positivos. No exemplo, a definição desse domínio procede, uma vez que no contexto da lista de supermercado, não parece razoável que um produto tenha quantidades negativas ou nulas.

Para o atributo *Med.*:

$$dom(Med.) = \{kg, l, g, u, p\}$$

O atributo *Med.* possui somente cinco valores válidos: *kg* (quilograma), *l* (litro), *g* (grama), *u* (unidade) e *p* (pacote). O objetivo desse atributo é indicar a unidade de medida referência do produto pesquisado. Note que, apesar de configurado como uma coluna de texto, só serão considerados válidos esses valores, de modo que qualquer outra medida será inválida. Este exemplo é particularmente interessante porque permite ilustrar uma forte relação entre diferentes atributos da tabela. As colunas *Qtd.*, *Med.* e *Preço* devem ser usadas juntas para representar o valor de um produto (empacotado em uma embalagem padronizada por alguma unidade de medida). Um valor errado que for cadastrado em *Med.* torna inviável calcular a relação de preço obtida por *Qtd.* e *Preço* e provavelmente inutilizará a linha inteira para futuras consultas, justificando mais ainda o zelo pela qualidade dos dados.

Quanto ao *Preço*:

$$dom(Preço) = \{x \in Preço: x \in \mathbb{R} \; e \; x \geq 0\}$$

O atributo *Preço* reconhece somente valores reais maiores ou iguais a zero (ou valores não-negativos). Os possíveis valores do tipo de dado *Moeda* podem ser vistos como subconjuntos de números reais, sempre representados com duas casas decimais.

Embora valores negativos possam ser válidos em algumas aplicações (como planilhas contábeis, por exemplo), não é esperado que apareçam preços negativos em um supermercado.

Como vimos, a definição de tipos e domínios presta um importante serviço para a manutenção da qualidade e confiabilidade dos dados contidos em uma base de dados. Tipos de dados e domínios são conceitos correlatos e atuam de forma complementar. De alguma maneira, o tipo de dados define um domínio inicial, que pode ser refinado de acordo com a contextualização da informação que será guardada.

Nem sempre é vantajoso criar domínios. Analisemos, por exemplo, a coluna *Produto*. Claramente, parece um exagero aceitar qualquer possibilidade de texto para descrever um produto do supermercado. Por outro lado, também não é trivial catalogar todas as possibilidades possíveis de produtos em um supermercado que poderiam entrar nessa lista. Nessa situação, cabe ao responsável pela base analisar as vantagens e desvantagens envolvidas na definição dos domínios. Para algumas situações, apesar de mais abrangente, pode ser mais adequado aceitar qualquer possibilidade de construção de texto do que restringi-las a um conjunto grande de dados e de difícil manutenção.

Por que é importante definir tipos?

A discussão sobre a importância da escolha correta dos tipos pode ser ainda justificada, além de todos os aspectos já citados, pela interpretação dada pelo computador aos valores guardados na base. Para ilustrar essa situação, vamos acompanhar a comparação de duas datas, respondendo à seguinte pergunta:

Qual desses *valores* é *menor*: 12/12/2010 ou 22/12/2009?

A melhor resposta seria:

Não sei. Preciso saber o tipo de dados para poder responder!

Nesse exemplo, a interpretação do tipo de dados faz toda a diferença! A interpretação natural de qualquer pessoa ao ver os valores *12/12/2010* e *22/12/2009* seria tratá-los como datas. Dessa forma, considerando a menor data aquela que ocorreria primeiro na escala crescente de tempo, *22/12/2009* é menor que *12/12/2010*. Decisão tomada sem maiores problemas.

Para o computador, entretanto, prevalece o tipo de dado da posição na qual foram armazenados esses valores. Se armazenados como cadeias de caracteres (texto), a comparação de ambos ocorrerá levando em conta o critério de comparação de texto, ou seja, a classificação lexicográfica (a mesma usada pelo dicionário). Sendo assim, a cadeia de caracteres "12/12/2010" seria menor que "22/12/2009", pois o primeiro caractere de ambas indicaria que o texto iniciado em "1" é menor que o texto iniciado em '2'.

Nesse caso, nota-se como é importante indicar claramente os tipos de dados para ter sucesso em determinadas operações, principalmente operações de comparação e, consequentemente, classificação (ordenação), que exploraremos nas próximas seções.

Algumas observações sobre planilhas

Planilhas eletrônicas são sistemas de *software* que servem para organizar dados, proporcionando visualização e tratamento matricial, e integrando com um número razoável de funções matemáticas e sistemas geradores de gráficos. É possível organizar dados no formato de tabelas de bases de dados, mas muitas funcionalidades, principalmente as relacionadas ao controle de consistência são menos rígidas e flexíveis do que o necessário para muitas aplicações.

Embora permita a definição de tipos de dados para um conjunto de posições de linha e coluna, planilhas eletrônicas usualmente permitem a inclusão de dados fora de seu domínio e, por vezes, incompatíveis com os tipos de cada posição da tabela. Dados incorretos, embora possíveis, causam muito transtorno e problemas quando utilizados em operações matemáticas, de modo que é sempre importante verificar se atendem plenamente à especificação de seus tipos e domínios.

Outra importante observação refere-se à preservação da estrutura matricial. Quando um sistema de planilhas é usado como referência para a manipulação de uma base de dados, deve-se tomar especial cuidado para garantir que todas as suas células possuam informações corretas e consistentes. A consistência dos dados só pode ser preservada se a estrutura da tabela for respeitada. Segundo essa restrição, os dados dispostos nas linhas devem ter exatamente a mesma quantidade de colunas (mesmo que algumas colunas tenham valor vazio). Caso essa regra não seja respeitada, a base de dados passa a não ser mais confiável e, consequentemente, perde sua utilidade.

A estrutura matricial não deve ser quebrada, tanto na linha quanto na coluna, pois isso provoca uma corrupção dos dados e assim da base inteira. Além disso, operações de visualização ou recuperação devem preservar a integridade da base, nas linhas e nas colunas, como veremos na Seção 4.5.

Em sistemas de *software* específicos para o tratamento de bases de dados, o controle de tipos e domínios é extremamente rigoroso, obrigando a adequação dos valores a seus respectivos tipos e domínios.

4.4 ARQUIVOS ESTRUTURADOS

Arquivos estruturados (ou *flat files*) são arquivos de dados organizados sob uma estrutura rígida e pré-definida. Os arquivos estruturados são muito usados para a troca de dados entre sistemas computacionais diferentes que não mantêm entre si um canal direto de comunicação. Informalmente, dizemos que o arquivo estruturado é um "espelho" da tabela, pois sua disposição física facilita a organização dos valores na estrutura de linhas e colunas, tal qual a tabela.

Esses arquivos também são muito difundidos na comunidade científica, para distribuição de bases de dados, preservando a sua estrutura e seu conteúdo. Há muitos exemplos de bases abertas distribuídas gratuitamente que utilizam arquivos estruturados para facilitar a importação de dados por parte de pesquisadores.

Em um arquivo estruturado, é fácil identificar todos os elementos presentes na base de dados. Definem-se caracteres (ou sequências de caracteres) especiais que exercerão

o papel de delimitadores de linha e de coluna. Os separadores de coluna servem para indicar a separação entre diferentes colunas na mesma linha. Enquanto isso, os separadores de linha servem para separar as linhas da tabela entre si. Considerando os dados de nosso exemplo, e assumindo o caractere ";" como separador de colunas e o caractere "quebra de linha" como separador de linhas, teríamos o arquivo estruturado mostrado na Figura 4.4.

Esse formato é uma variação do conhecido CSV (*Comma-Separated Values*), em que as vírgulas (",") são usadas como separadores de colunas. Como nosso exemplo usa *vírgulas* na coluna *Preço* para compor o valor (fracionário), não podemos usá-las diretamente como separador. Veja que o separador deve ser sempre um caractere (ou uma sequência deles) que não se confunde com nenhum outro valor presente na tabela, a fim de evitar ambiguidades de interpretação de valores. Nesse exemplo, o separador escolhido foi o caractere ponto-e-vírgula (";"). Note que, obrigatoriamente, todas as linhas possuem cinco (5) separadores de coluna, evidenciando que a tabela possui seis (6) colunas.

Figura 4.4: Exemplo de arquivo estruturado csv.

```
"Suco";"ValeSuco";1;l;3.00;"Arpoador";12/05/2011
"Suco";"Flash";1;l;4,50;"Arpoador";12/05/2011
"Tomate";" - ";1;kg;3,50;"Noite";14/05/2011
"Arroz";"Tio José";5;kg;8,64;"Noite";14/05/2011
"Arroz";"Sem Broto";5;kg;9,99;"Arpoador";12/05/2011
"Arroz";"Da TV";1;kg;1,99;"Noite";14/05/2011
"Feijão";"Sem Broto";1;kg;4,00;"Arpoador";12/05/2011
"Tomate";" - ";1;kg;2,99;"Noite";14/05/2011
"Ovo";"A Granja";12;u;3,19;"Arpoador";12/05/2011
"Ovo";"Caseiro";6;u;1,45;"Noite";14/05/2011
"Suco";"Flash";1;l;3,99;"Noite";12/05/2011
```

A maioria das ferramentas de bases de dados (e também de planilhas eletrônicas) possui funcionalidades de importação (leitura) e exportação (gravação) de arquivos em formato estruturado.

4.5 MANIPULAÇÃO DE DADOS

Como vimos anteriormente, toda base de dados deve apresentar três funcionalidades principais que, juntas, permitem explorar plenamente o potencial dos dados que estão sendo manipulados: *armazenamento*, *recuperação* e *visualização*. Nas seções 4.3 e 4.4, tratamos o armazenamento, e agora abordaremos em detalhes as operações de recuperação (Filtros) e visualização (Ordenação).

4.5.1 FILTROS

A filtragem é uma forma de recuperação de dados que consiste em realizar *consultas* sobre a base de dados, verificando valores e propriedades dos dados armazenados.

Consultas são extremamente úteis, pois permitem *selecionar* dados, exibindo somente aqueles que obedecem a determinadas propriedades. Isso fornece um mecanismo para que o usuário possa encontrar informações facilmente (na maioria dos casos) em sua base de dados.

As consultas em bases de dados são organizadas como conjuntos de predicados lógicos sobre os atributos. Um *predicado lógico simples* (também chamado *predicado simples* ou *predicado*) é uma expressão que devolve um valor VERDADEIRO ou FALSO. O formato de um predicado é normalmente dado por:

<*atributo*> <*operador*> <*valor*>

onde:

<*atributo*> é a característica usada como referência na consulta;

<*operador*> é um operador lógico que devolve um resultado VERDADEIRO ou FALSO. Os principais operadores lógicos são mostrados na Tabela 4.3.

Tabela 4.3: Operadores lógicos para consultas.

<	Menor que
≤	Menor ou igual a
>	Maior que
≥	Maior ou igual a
=	Igual a
≠	Diferente de

<*valor*> estabelece uma referência de valor que atuará como filtro sobre o <*atributo*>.

Para ilustrar melhor esse conceito, vamos usar um exemplo. Imagine que, após muito esforço feito ao visitar vários supermercados, alguém tenha conseguido montar a base de dados de produtos mostrada na Tabela 4.2. Uma vez montada, alguém deseja consultar todas as opções de compra de arroz que foram catalogadas na pesquisa de supermercados. Essa informação pode ser obtida aplicando-se sobre a base uma consulta sobre o nome do *produto* com valor igual a "arroz", que poderia apresentar um formato semelhante a:

Produto ='Arroz'

Ao aplicar essa consulta na base de dados (e assumindo-se o formato de tabela como armazenamento), todos os itens do supermercado (linhas) são testados, aplicando-se o predicado anterior sobre o atributo (coluna) *Produto*. Após essa operação, farão parte da resposta SOMENTE os produtos (linhas) nas quais o predicado é VERDADEIRO, ou seja, todos os *produtos* com valor igual a 'arroz'. O resultado da consulta anterior pode ser visto na Tabela 4.4:

Tabela 4.4: Lista de supermercados que vendem o produto 'arroz'.

Produto	Fabricante	Qtd.	Med.	Preço	Supermercado	Data
Arroz	Tio José	5	kg	8,64	Noite	14/05/2011
Arroz	Sem Broto	5	kg	9,99	Arpoador	12/05/2011
Arroz	Da TV	1	kg	1,99	Noite	14/05/2011

Nesse exemplo, é fácil perceber que a consulta devolveu somente os produtos (linhas) que obedecem ao critério inicial, permitindo ao usuário se concentrar somente nos produtos que são de seu interesse no momento.

OBS.: Convém salientar que a consulta não altera o estado da base de dados, ou seja, nenhum dado é removido ou alterado durante a realização de uma consulta. Ela apenas manipula a base de dados, selecionando e exibindo os dados de acordo com o predicado que lhe é informado.

As consultas mais elaboradas também podem ser usadas, construindo-se um predicado por meio da combinação de outros predicados. Essa construção ocorre por meio dos operadores lógicos relacionais *E* e *OU*. Chamamos de *predicado composto* todo predicado formado a partir de outros predicados (que, por sua vez, podem ser simples ou compostos).

O operador relacional *E*, também chamado *operador relacional conjuntivo*, é um operador binário que constrói uma expressão com valor lógico (VERDADEIRO ou FALSO), combinando conjuntivamente dois predicados. Vamos usar a Tabela 4.5 para mostrar o comportamento do operador *E*. Nesse caso, assuma que *E* será aplicado sobre dois predicados, que chamaremos P_1 e P_2.

Tabela 4.5: Combinação conjuntiva de dois predicados, P_1 e P_2.

P_1	P_2	$P_E = P_1 \; E \; P_2$
V	V	V
V	F	F
F	V	F
F	F	F

Note que, P_1 e P_2, como predicados, possuem algum valor lógico. O predicado composto P_E (definido como $P_1 \; E \; P_2$) será verdadeiro somente se P_1 e P_2 forem verdadeiros. Caso algum deles seja falso, P_E também será falso.

Voltando ao exemplo da lista de compras, vamos expressar uma consulta na qual desejamos verificar todas as opções de compra de arroz disponíveis no supermercado "Noite":

Produto = 'Arroz' *E* Supermercado = 'Noite'

O resultado dessa consulta pode ser visto na Tabela 4.6.

Tabela 4.6: Supermercados que vendem 'arroz' *e* chamam-se 'noite'.

Produto	Fabricante	Qtd.	Med.	Preço	Supermercado	Data
Arroz	Tio José	5	kg	8,64	Noite	14/05/2011
Arroz	Da TV	1	kg	1,99	Noite	14/05/2011

Como você pode notar, o resultado da consulta restringiu-se somente aos produtos nos quais TODOS os predicados simples são verdadeiros, ou seja, *Produto = "Arroz"* e *Supermercado = "Noite"*.

O operador relacional *OU*, também chamado *operador relacional disjuntivo*, é igualmente um operador binário que constrói uma expressão com valor lógico, combinando dois outros predicados. Diferentemente do operador *E*, o operador *OU* apresenta o comportamento indicado na Tabela 4.7. Novamente, assuma a existência dos predicados P_1 e P_2.

Tabela 4.7: Combinação disjuntiva de dois predicados, P_1 ou P_2.

P_1	P_2	$P_{OU} = P_1$ *OU* P_2
V	V	V
V	F	V
F	V	V
F	F	F

O predicado composto P_{OU} (definido como P_1 *OU* P_2) será falso somente se P_1 e P_2 forem falsos. Caso algum deles seja verdadeiro, P_{OU} também será verdadeiro.

Voltando ao exemplo anterior, vamos substituir o operador *E* por *OU*. Veja o resultado:

Produto = 'Arroz' *OU* Supermercado = 'Noite'

O resultado dessa consulta pode ser visto na Tabela 4.8:

Tabela 4.8: Supermercados que, *ou* vendem 'arroz,' *ou* chamam-se 'Noite'.

Produto	Fabricante	Qtd.	Med.	Preço	Supermercado	Data
Tomate	-	1	kg	3,50	Noite	14/05/2011
Arroz	Tio José	5	kg	8,64	Noite	14/05/2011
Arroz	Sem Broto	5	kg	9,99	Arpoador	12/05/2011
Arroz	Da TV	1	kg	1,99	Noite	14/05/2011
Tomate	-	1	kg	2,99	Noite	14/05/2011
Ovo	Caseiro	6	u	1,45	Noite	14/05/2011
Suco	Flash	1	l	3,99	Noite	12/05/2011

Como você pode perceber, o resultado da consulta é bem diferente. Isso ocorre porque basta que um dos predicados simples (*Produto = "Arroz"* ou *Supermercado = "Noite"*) seja verdadeiro para que toda a expressão seja verdadeira. A interpretação dessa consulta seria:

Recuperar todos os produtos nos quais, pelo menos, uma das condições é verdadeira:

(1) O produto chama-se 'arroz' (, nesse caso, independentemente do nome do supermercado que o vendo);

(2) O produto é vendido pelo supermercado 'Noite' (nesse caso, independentemente do nome do produto).

Consultas também podem envolver operações de agregação. Uma agregação calcula um valor resultante de uma coleção de valores, como média, soma, mínimo, máximo e contagem. Essas operações podem ser utilizadas para responder consultas como: qual a média de preços dos produtos de um determinado fabricante? Ou quantos produtos de uma determinada marca são vendidos por um supermercado específico? Perguntas semelhantes são frequentes em muitas situações da vida real. Essas funções não recuperam valores diretamente dos dados armazenados, sendo resultado de alguma operação aplicada sobre um subconjunto dos dados. Com isso, quando se usam planilhas, a obtenção de operações de agregação precisa de células auxiliares e/ou de funções pré-definidas.

4.5.2 ORDENAÇÃO

Ordenação é uma operação comum quando se trabalha com dados. Dados ordenados facilitam a localização de informação e identificação de padrões. Além disso, a ordenação de dados pode ser considerada uma operação básica e presente em praticamente todos os sistemas que trabalham com visualização de dados. Tal como a consulta, a ordenação não muda as características da base de dados e consiste apenas em uma operação aplicada sobre um subconjunto de dados, produzindo uma resposta na qual os dados são exibidos segundo alguma ordem.

Grosseiramente, a ordenação é uma operação que organiza os dados obedecendo a algum critério pré-definido. Quando falamos em ordenação, invariavelmente estamos também nos referindo a formas de enxergar e interpretar tipos de dados. Apesar de bastante intuitiva, a ordenação nem sempre é uma operação fácil de aplicar.

Para tornar mais explicativo o tema, vamos analisar algumas situações. Provavelmente, você não teria problemas para ordenar um conjunto de números inteiros, ou uma lista de palavras (como um dicionário). Mas como você ordenaria seus principais objetivos neste ano? Como você ordenaria as melhores opções de transporte para sair da UFABC e chegar à sua residência? Como você colocaria em ordem seus melhores amigos? Aparentemente, nem tudo é tão fácil de ordenar.

Para entender um pouco melhor a natureza dessa operação, vamos tentar analisá-la um pouco mais. Ordenação é uma forma de *comparação*. Comparar significa fazer

escolhas, de modo que uma comparação só pode ser feita se você tiver duas ou mais opções de escolha. Para o tratamento convencional de dados, ordenar significa organizar um conjunto de dados, escolhendo cuidadosamente qual aparecerá antes e qual aparecerá depois na resposta a uma consulta. Do ponto de vista computacional, essa escolha se baseia no resultado de uma comparação usando os operadores $>$, $\geq$, $<$ e $\leq$.

Há conjuntos em que o resultado dessa comparação não levanta dúvidas. No conjunto dos números reais, por exemplo, sabe-se facilmente quais são o maior e o menor entre dois números distintos; sabe-se identificar, também facilmente, dois números iguais. Assim sendo, é fácil ordenar números reais, pois seu conjunto é dito *parcialmente ordenado*. Os conceitos formais de ordem parcial e correlatos serão explorados em disciplinas posteriores de Álgebra; por hora, você pode reconhecer um conjunto parcialmente ordenado se retirar quaisquer dois elementos dele e conseguir identificar claramente qual deles é maior (ou menor) ou identificar que são iguais.

Quando temos dados com maior complexidade, cuja ordenação não é clara, precisamos estabelecer critérios (ou fatores) de ordenação, que nos ajudarão a organizar melhor os dados. Por exemplo, considere que você deseja avaliar as melhores opções de deslocamento da UFABC para sua residência. Após algumas tentativas, você consegue catalogar as opções de transporte (listadas na Tabela 4.9), considerando algumas informações como trajeto (distância de deslocamento do veículo), preço (total gasto com o transporte), tempo (quantidade de minutos média necessária para cumprir o trajeto), quantidade de horários (disponibilidade da opção de transporte) e distância de passeio (quantidade de metros que deverão ser percorridos a pé).

Tabela 4.9: Lista original de opções de transporte.

Opção	Trajeto	Preço	Tempo	Qtd. Horários	Passeio
1 táxi	15 km	R$ 40,00	10 min.	vários	0 m.
1 ônibus	20 km	R$ 5,00	25 min.	10 por dia	500 m.
2 ônibus	25 km	R$ 5,00	35 min.	20 por dia	200 m.
1 ônibus + 1 trem	21,5 km	R$ 6,00	23 min.	18 por dia	50 m.
3 ônibus	21,5 km	R$ 5,00	25 min.	18 por dia	1000 m.
2 trens	30 km	R$ 6,50	45 min.	40 por dia	300 m.
1 ônibus + 2 trens	35 km	R$ 7,00	50 min.	40 por dia	100 m.
2 ônibus + 1 trem	30 km	R$ 6,80	55 min.	18 por dia	500 m.

As melhores opções de transporte não são claras, pois cada opção de transporte possui suas próprias características e pode apresentar pontos positivos e negativos em relação às demais. Como podemos, então, ordenar esse conjunto de dados?

Só é possível ordenar um conjunto de dados de forma justa se forem definidos claramente quais serão os critérios de ordenação. Suponha que o critério escolhido seja o preço total do deslocamento. Sendo assim, uma ordenação dos dados, considerando o preço total, resultaria na Tabela 4.10:

Tabela 4.10: Opções de transporte ordenadas por preço.

Opção	Trajeto	Preço	Tempo	Qtd. Horários	Passeio
1 ônibus	20 km	R$ 5,00	25 min.	10 por dia	500 m.
2 ônibus	25 km	R$ 5,00	35 min.	20 por dia	200 m.
3 ônibus	21,5 km	R$ 5,00	25 min.	18 por dia	1000 m.
1 ônibus + 1 trem	21,5 km	R$ 6,00	23 min.	18 por dia	50 m.
2 trens	30 km	R$ 6,50	45 min.	40 por dia	300 m.
2 ônibus + 1 trem	30 km	R$ 6,80	55 min.	18 por dia	500 m.
1 ônibus + 2 trens	35 km	R$ 7,00	50 min.	40 por dia	100 m.
1 táxi	15 km	R$ 40,00	10 min.	vários	0 m.

Nessa situação, note que os dados são exatamente os mesmos e foram ordenados de forma *crescente* sobre o preço total do deslocamento. Na ordenação crescente, valores menores são mostrados antes de valores maiores. Na ordenação *decrescente*, valores maiores são mostrados antes de valores menores. Após serem ordenadas, fica fácil verificar quais são as opções de transporte mais baratas e, portanto, melhores (quando nos referimos a preço total).

É possível, ainda, definir critérios secundários de ordenação, que servem para ordenar dados que apresentam valores iguais para o critério principal de ordenação. No exemplo, note que, para as viagens mais baratas (R$ 5,00), temos três opções: 1 ônibus, 2 ônibus ou 3 ônibus. Para escolher a melhor delas, podemos definir um novo critério de ordenação, que será aplicado sempre que o preço for igual. Considere, então, como critério secundário o tempo gasto no deslocamento. Se aplicarmos esse raciocínio, as opções de transporte serão ordenadas crescentemente por seu preço, exibindo primeiro as que demoram menos tempo quando os preços forem iguais. O resultado pode ser visto na Tabela 4.11:

Tabela 4.11: Opções de transporte ordenadas por preço e tempo.

Opção	Trajeto	Preço	Tempo	Qtd. Horários	Passeio
1 ônibus	20 km	R$ 5,00	25 min.	10 por dia	500 m.
3 ônibus	21,5 km	R$ 5,00	25 min.	18 por dia	1000 m.
2 ônibus	25 km	R$ 5,00	35 min.	20 por dia	200 m.
1 ônibus + 1 trem	21,5 km	R$ 6,00	23 min.	18 por dia	50 m.
2 trens	30 km	R$ 6,50	45 min.	40 por dia	300 m.
2 ônibus + 1 trem	30 km	R$ 6,80	55 min.	18 por dia	500 m.
1 ônibus + 2 trens	35 km	R$ 7,00	50 min.	40 por dia	100 m.
1 táxi	15 km	R$ 40,00	10 min.	Vários	0 m.

Note que, usando esses critérios, parece mais vantajosa a opção de um deslocamento usando três ônibus, em vez de apenas dois. Você pode ainda estabelecer critérios de ordenação terciários, quaternários, etc.

Outra importante observação deve ser feita sobre os tipos de dados. Só foi possível ordenar a base de dados porque os tipos de dados das colunas envolvidas na ordenação recebem valores que mantêm entre si uma relação de ordem parcial. Quando se usam planilhas, de forma geral, todos os tipos de dados podem ser ordenados, mesmo campos textuais, que obedecem à classificação lexicográfica.

Em bancos de dados mais elaborados, quando são adotados sistemas gerenciadores de bancos de dados, entretanto, nem sempre os atributos são representados com tipos de dados com ordem parcial, tais como tipos de dados que armazenam imagens e objetos binários. Esses tipos são adotados normalmente para garantir o armazenamento seguro dos dados complexos, cujas características fogem da maioria dos dados com os quais estamos acostumados a trabalhar. Nesses casos, a prioridade é dada ao sistema de armazenamento, e operações de busca são realizadas por meio de outros atributos, modelados com tipos de dados mais simples.

IMPORTANTE*:* Quando se deseja ordenar dados, tenha cuidado para garantir a integridade das linhas! Numa representação de planilha, as linhas representam um conjunto de informações que se refere ao mesmo objeto, portanto a linha inteira deve ser movimentada em caso de uma ordenação. Quando se trabalha com conjuntos de dados em uma planilha, a ordenação isolada de uma coluna pode NÃO FAZER SENTIDO, sendo uma fonte considerável de erros.

É fácil notar isso no exemplo a seguir. Considere a seguinte tabela com dados de alunos (primeiro nome, data de nascimento e idade):

Primeiro Nome	Data de Nascimento	Idade
Vanessa	03/01/1994	19
Antônio	02/01/1992	21
Maria	01/01/1998	15

Considerando que gostaríamos de ordenar crescentemente os dados a partir da idade, imagine que alguém equivocadamente realiza a ordenação SOMENTE sobre a coluna *Idade*, ignorando as demais. O resultado seria:

PERDA DE CONSISTÊNCIA

Primeiro Nome	Data de Nascimento	Idade
Vanessa	03/01/1994	15
Antônio	02/01/1992	19
Maria	01/01/1998	21

Como é possível perceber, o resultado é completamente incoerente, dado que as idades e as datas de nascimento não são mais compatíveis entre si. Veja, por exemplo, que Maria (a mais nova), passou a ser a mais velha após a ordenação equivocada. Dizemos, nesse caso, que os dados são *inconsistentes*, pois foram manipulados de forma incorreta e já não estão coerentes com os dados originais.

Ao realizar a ordenação, embora a Idade seja o critério desejado, a linha inteira deve ser deslocada para produzir o resultado da visualização. Sendo assim, o resultado correto da operação seria:

Primeiro Nome	Data de Nascimento	Idade
Maria	01/01/1998	15
Vanessa	03/01/1994	19
Antônio	02/01/1992	21

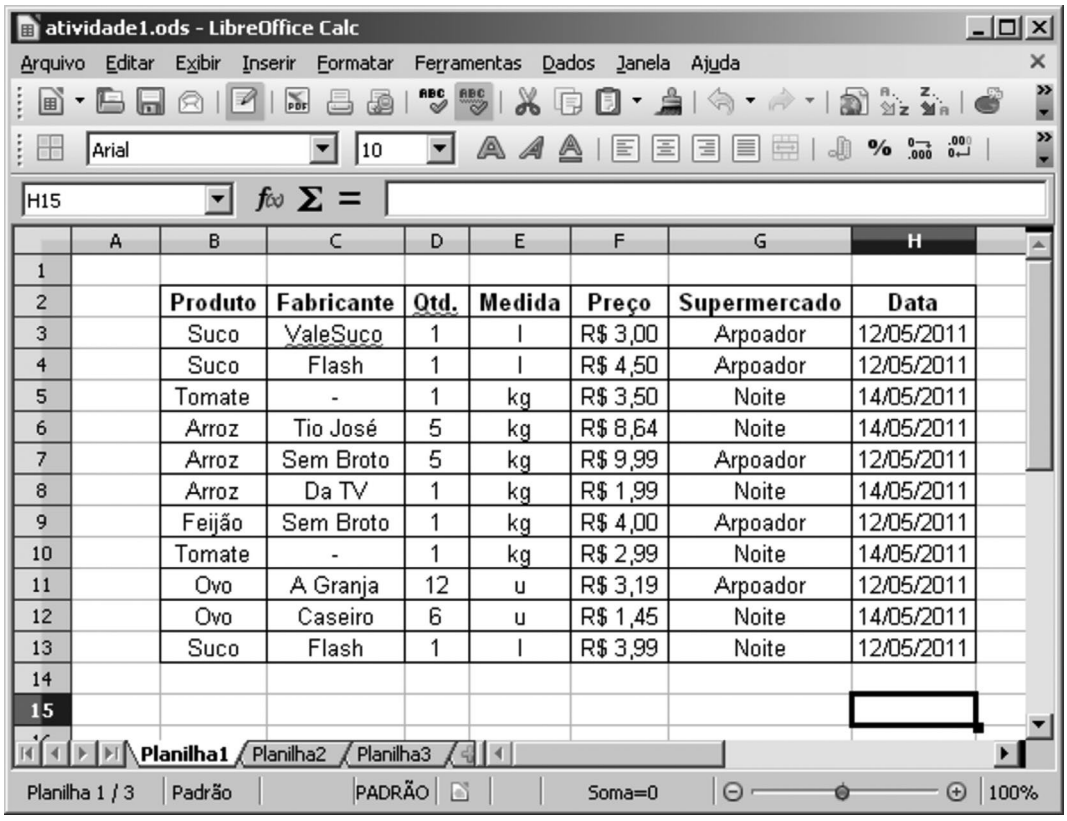
4.6 ATIVIDADES EM AULA

1. (Armazenamento: Definição de tipos e domínio) Vamos criar uma planilha no Calc contendo a tabela com a lista de supermercado exemplificada no texto.

Abra o Calc. Aparecerá uma área de trabalho com uma planilha em branco. Como primeira operação, salve esta planilha usando as teclas de atalho "Ctrl + Shift + s" ou o menu "Arquivo >> Salvar como". Digite em "Nome do arquivo": "atividade1". (*Dica:* Crie uma pasta separada para guardar os seus arquivos, por exemplo, em "Meus documentos\SeuNomeSobrenome")

Agora, você pode digitar uma tabela como a apresentada na Figura 4.5. Contudo, há um método mais prático para importar os dados e que pode poupar o trabalho de digitação.

Figura 4.5: Base de dados: Atividade 1.



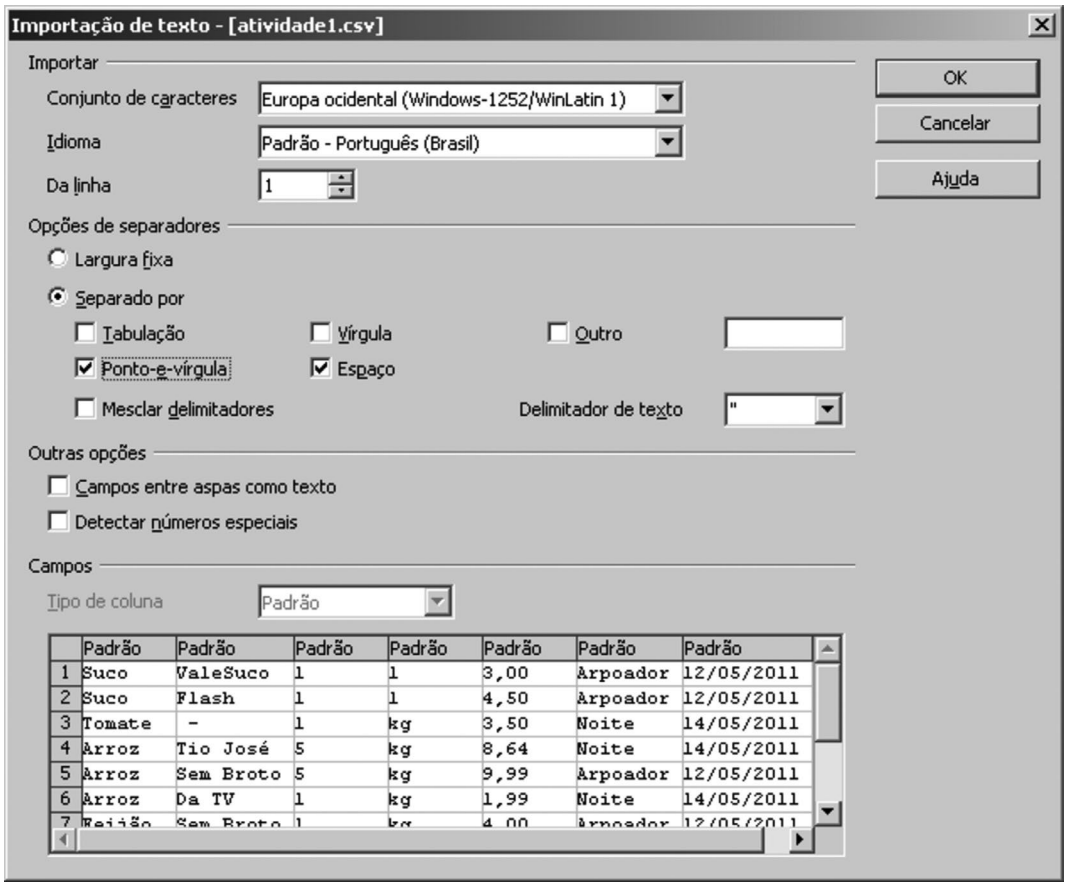
	A	B	C	D	E	F	G	H
1								
2		Produto	Fabricante	Qtd.	Medida	Preço	Supermercado	Data
3		Suco	ValeSuco	1	l	R$ 3,00	Arpoador	12/05/2011
4		Suco	Flash	1	l	R$ 4,50	Arpoador	12/05/2011
5		Tomate	-	1	kg	R$ 3,50	Noite	14/05/2011
6		Arroz	Tio José	5	kg	R$ 8,64	Noite	14/05/2011
7		Arroz	Sem Broto	5	kg	R$ 9,99	Arpoador	12/05/2011
8		Arroz	Da TV	1	kg	R$ 1,99	Noite	14/05/2011
9		Feijão	Sem Broto	1	kg	R$ 4,00	Arpoador	12/05/2011
10		Tomate	-	1	kg	R$ 2,99	Noite	14/05/2011
11		Ovo	A Granja	12	u	R$ 3,19	Arpoador	12/05/2011
12		Ovo	Caseiro	6	u	R$ 1,45	Noite	14/05/2011
13		Suco	Flash	1	l	R$ 3,99	Noite	12/05/2011
14								
15								

Clique no menu "Inserir >> Planilha do arquivo" e escolha o arquivo "atividade1.csv". Trata-se do arquivo texto com os dados separados por vírgula. Na janela que aparece na Figura 4.6, em "Opções de separadores" selecionar "Outros" e preencher o tipo de separador usado: "**;**" (ponto-e-vírgula). Observe que, na parte inferior dessa caixa, aparece como os dados devem ser exibidos na planilha (note que é importante desmarcar a caixa "Vírgula", uma vez que estamos usando a vírgula para denotar o ponto decimal). Se estiver tudo certo, clique em **"**OK".

Os dados aparecem sem formatação e sem os nomes das colunas (rótulos da tabela). Com o *mouse*, selecione todo o bloco de dados, recorte e cole em uma célula de sua preferência (por exemplo, a célula "*B3"*). Agora, você pode digitar os rótulos de cada coluna (Figura 4.5), acertando a largura delas, caso seja necessário.

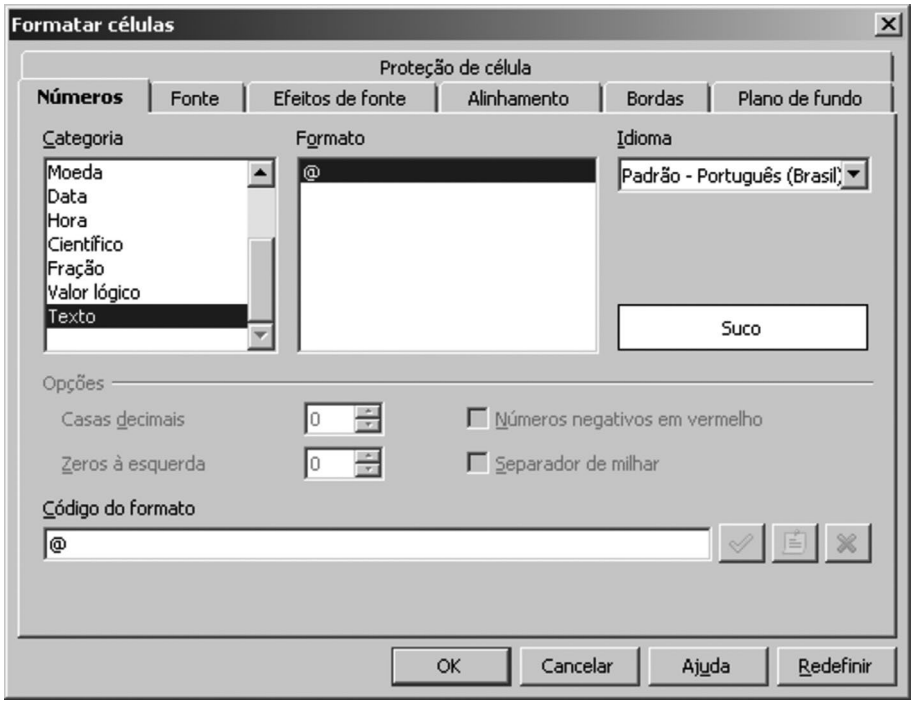
Selecione simultaneamente os dados das colunas *Produto*, *Fabricante*, *Medida* e *Supermercado*. Para isto use o *mouse* e mantenha pressionada a tecla *Ctrl* para selecionar colunas não adjacentes. Clique com o botão direito do *mouse* sobre a área selecionada e escolha *Formatar células* (ou, no *menu* "Formatar >> Formatar células"). Em *Categoria*, especifique o tipo de dado como *Texto* (Figura 4.7), em seguida, confirme no botão "OK".

Figura 4.6: Base de dados: Atividade 1. Janela para importação de arquivos ".csv".



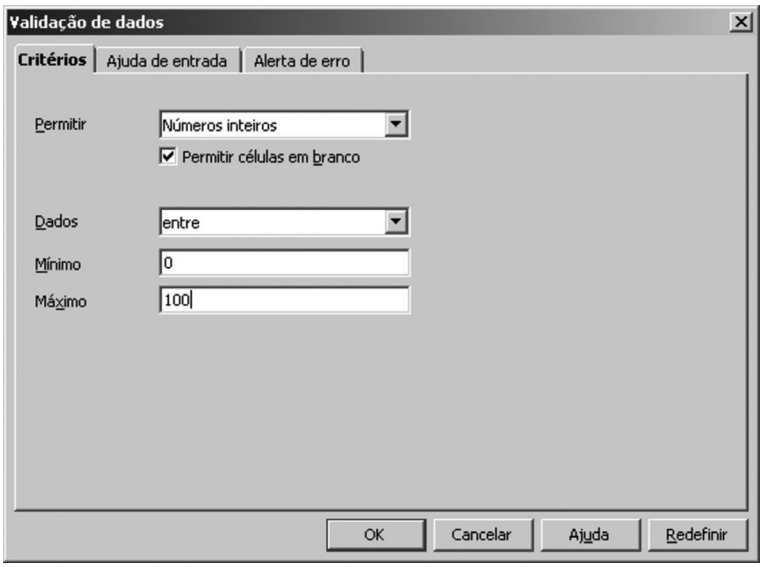
Repita para as colunas *Qtd.*, *Preço* e *Data*, respectivamente atribuindo os tipos *Número* (Geral), *Moeda* (Português Brasil) e *Data*.

Figura 4.7: Base de dados: Atividade 1. Janela para formatação de células.



Para evitar que sejam introduzidos dados inválidos, ainda é possível definir um domínio para os dados. Por exemplo, selecione os dados da coluna *Qtd.* e clique no *menu* "Dados >> Validação". Aparecerá a janela da Figura 4.8, na qual se escolheu: *Permitir Números Inteiros*, *Permitir Células em Branco*, *Dados entre*, no *Mínimo 0* e, no *Máximo 100*. Faça o mesmo e, em seguida, clique na aba "Alerta de Erro".

Figura 4.8: Base de dados: Atividade 1. Janela para definição de valores válidos de dados.



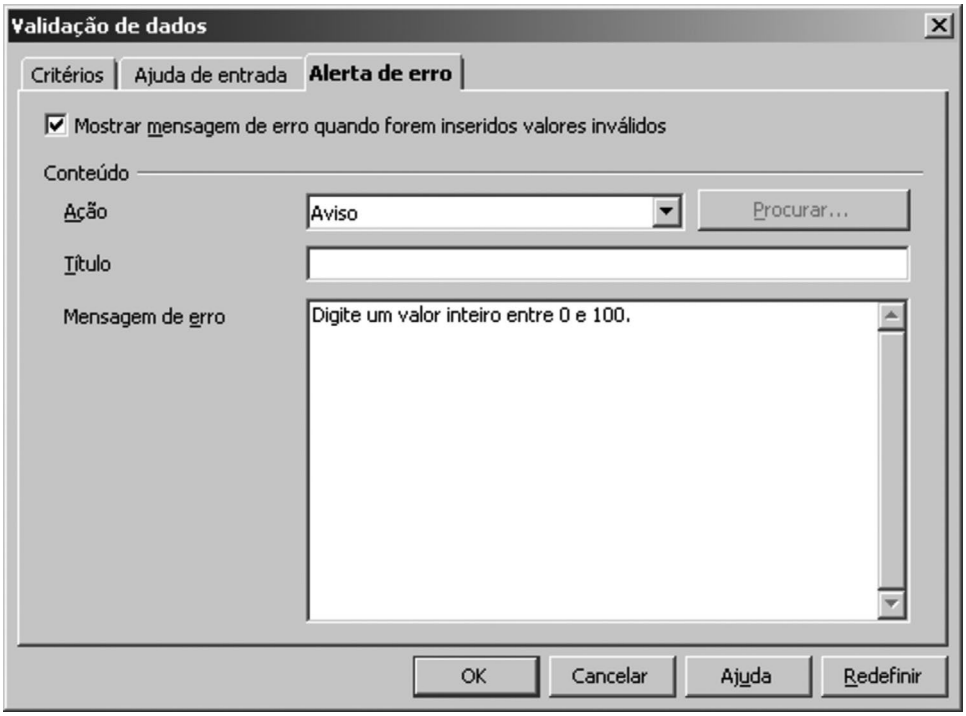
Na mesma janela, na aba "Alerta de erro", marcar "Mostrar mensagem", definir como *Ação* a emissão de um *Aviso* e digite uma *Mensagem de erro* como na Figura 4.9.

Agora, experimente digitar algum valor inválido e veja o que ocorre (para fins didáticos, restringiu-se a quantidade a números inteiros. Na prática, isso pode ser inconveniente caso se queira especificar 0,5 kg de algum produto, por exemplo).

Opcional – (Formatação da tabela). Uma tabela é uma maneira organizada de armazenar e apresentar os dados, sendo a sua aparência muito importante. Para formatar uma ou mais células, basta selecioná-las, clicar com o botão direito do *mouse* e escolher "Formatar células", ou na barra de *menus*, clicar em "Formatar" e, em seguida, escolher o item "Células".

Selecione toda a sua tabela e abra a janela "Formatar células". Clique na aba "Bordas" e faça os seguintes ajustes: "Padrão (Definir a borda externa e todas as linhas internas**)**"; "Estilo (0,50 pt)"; "Cor (Preto)**"**. Confirme os ajustes no botão "OK".

Figura 4.9: Base de dados: Atividade 1. Janela para definição de alertas de erros para dados inválidos.



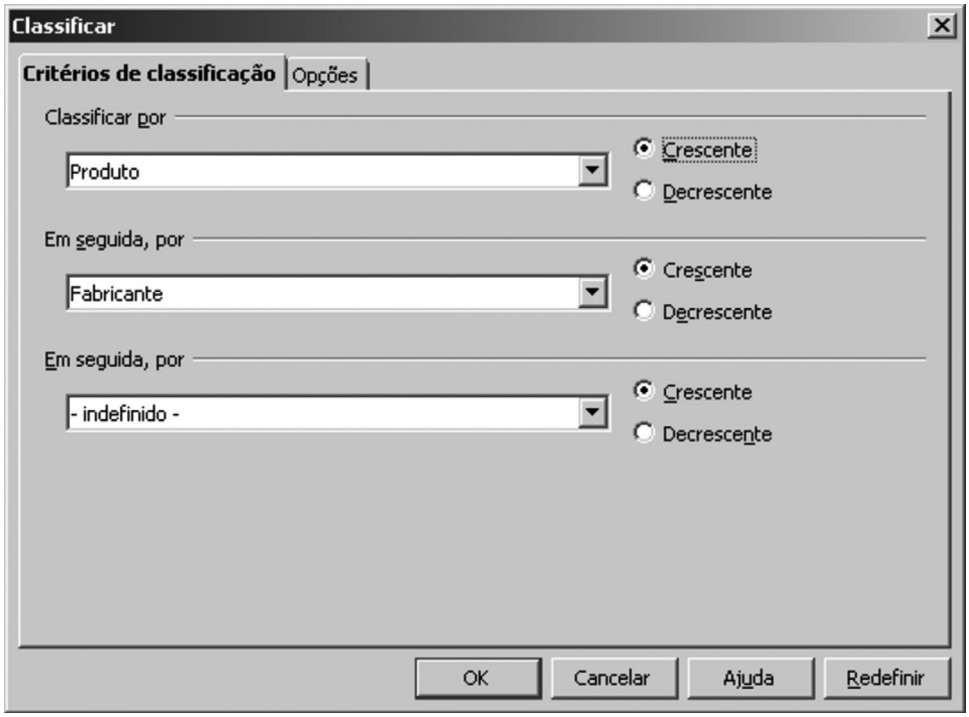
É conveniente destacar os rótulos da tabela. Para fazer isto, selecione a primeira linha e clique no botão "Negrito (N)" na barra de ferramentas. Este e muitos outros ajustes também estão disponíveis na caixa de diálogo "Formatar células". Experimente formatar a sua tabela para que fique como a apresentada na Figura 4.5.

2. (Visualização: Ordenação de dados usando critérios) Para facilitar a localização de dados na sua tabela, pode-se organizá-la usando critérios estabelecidos pelo usuário. Como exemplo, considere a sua planilha, que, no momento, encontra-se como a da Figura 4.5 e que poderia ser usada pelo caixa do supermercado para conferir o preço dos produtos. Neste caso, a consulta é muito fácil, pois digitamos apenas 11 itens de dados, contudo, imagine procurar um dado numa tabela com milhares de itens.

É conveniente que o operador de caixa disponha de uma lista impressa com os itens do campo *Nome* ordenados em ordem alfabética. E, como um mesmo produto (*Nome*) pode ter vários fabricantes, a lista impressa pode se tornar ainda mais organizada se também tiver os itens do campo *Fabricante* ordenados. Vejamos como isto pode ser feito.

Selecione toda a tabela e clique no *menu* "Dados >> Classificar". Abre-se a janela da Figura 4.10. Como primeiro critério, escolha "Classificar por Produto". E, em seguida por *Fabricante*. Ambos em ordem *Crescente*. Clique no botão *OK* e veja o resultado. Na barra de ferramentas, há um botão (*"AZ↓"*) que permite fazer classificações simples. O uso deste botão não é recomendado quando se quer classificar segundo vários rótulos e critérios mais gerais.

Figura 4.10: Base de dados: atividade 1. Janela de classificação de dados.



3. (Recuperação: Consulta de dados usando "AutoFiltro") Existem recursos que facilitam bastante a consulta de informações numa tabela com muitos itens de dados, chamados *filtros*. Nesta atividade, trabalharemos com um deles, chamado *AutoFiltro*, normalmente disponível em sistemas de planilhas.

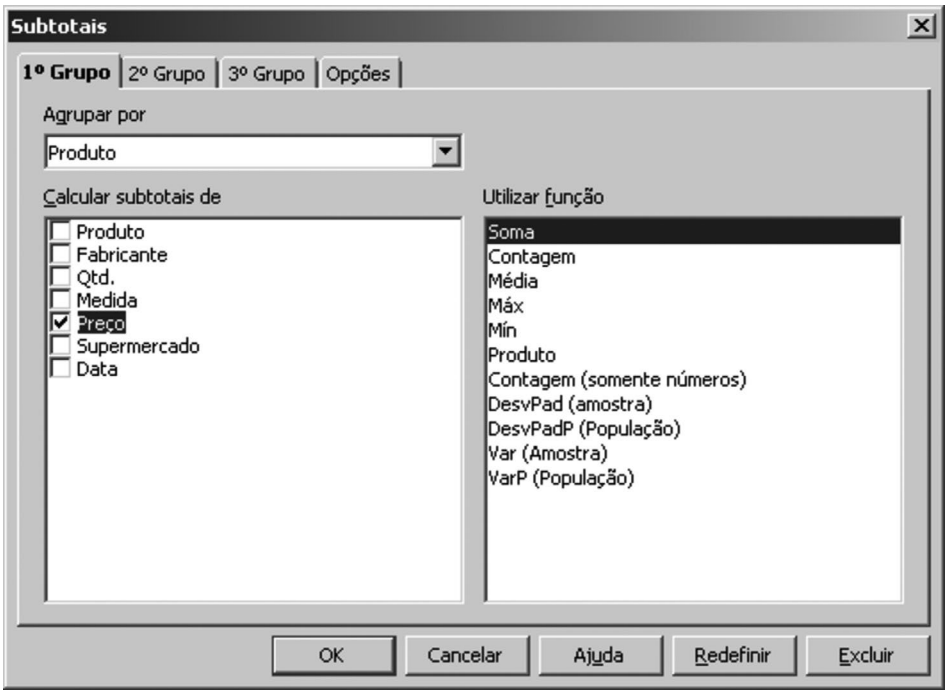
Para exibir este recurso, selecione uma das células da tabela e clique no *menu* "Dados >> Filtro >> AutoFiltro". Irão aparecer setas ao lado dos rótulos. Clique na seta do rótulo *Produto* e escolha *Suco*. Serão exibidas apenas as entradas referentes a *Suco*. Observe que a seta desta coluna muda de cor, indicando que está sendo usada como filtro.

Agora, você pode refinar um pouco mais a busca. Clique na seta de *Fabricante* e escolha *Flash*. Serão apresentadas apenas as linhas desse fabricante.

Uma dada classificação pode ser desfeita clicando-se na seta (no momento com a cor azul) e escolhendo-se *Todas*. E todo o recurso de *AutoFiltro* pode ser desabilitado em "Dados >> Filtro >> Ocultar autofiltro" .

4. (Criando subtotais) O levantamento de informações a partir de uma grande quantidade de dados tabelados pode ser facilitado por meio de ferramentas que automatizam várias operações e produzem um relatório. Mais especificamente, no Calc isto pode ser feito empregando o recurso *Subtotais*, que organiza os dados (classificando segundo algum critério) e, em seguida, calcula subtotais de algum campo utilizando uma dada função. Vejamos como isto pode ser feito por meio do exemplo da Figura 4.11.

Figura 4.11: Base de dados: Atividade 1. Janela para cálculo de subtotais.



Selecione toda a sua tabela e clique no *menu* "Dados >> Subtotais". É apresentada a janela *Subtotais* da Figura 4.11. Nessa janela, escolha "Agrupar por Produto", "Calcular subtotais de Preço" e "Utilizar função Soma". Clique no botão "OK". Agora, você tem um relatório com o subtotal gasto em cada produto e, no final, o total geral.

Os subtotais podem ser removidos abrindo novamente a janela *Subtotais* e acionando o botão "Excluir". De maneira alternativa, pode-se empregar o atalho "Ctrl + z" para desfazer a operação.

5. (Consultas usando predicados) Em uma tabela com uma grande quantidade de dados, a consulta de informações também po+de ser feita por um recurso chamado "`padrão`" e que permite definir padrões (predicados) para a busca de informações. Por exemplo, na lista de supermercado, imagine que você queira saber quais são os produtos com preços maiores do que R$ 4,00. Vejamos como isto pode ser feito no Calc:

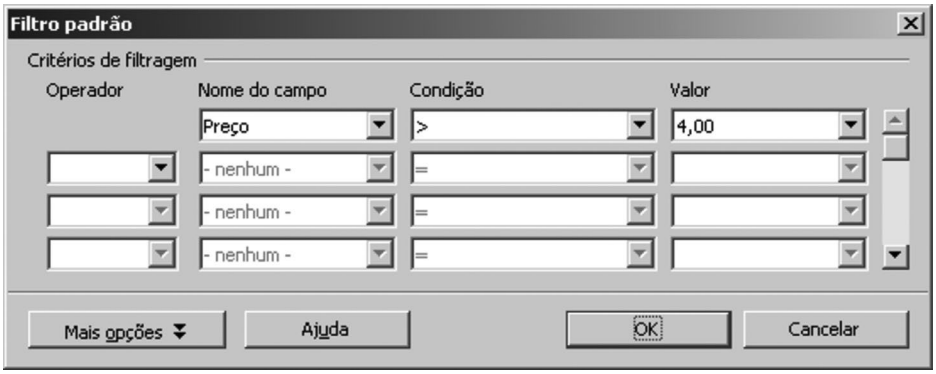
a. Selecione toda a tabela e acione o "AutoFiltro" por meio do menu "Dados >> Filtro >> AutoFiltro".

b. Clique na seta que aparece ao lado do rótulo *Preço* e escolha "Filtro padrão".

c. Aparece a janela Filtro padrão. Faça os ajustes indicados na Figura 4.12 e clique em "OK".

Para desfazer a consulta, basta desativar o *AutoFiltro*.

Figura 4.12: Base de dados: Atividade 1. Janela para definição de filtros.



4.7 CONSIDERAÇÕES FINAIS

Este capítulo introduziu alguns conceitos básicos de organização de informação, apresentando algumas das principais características de bases de dados. Como indicado no início deste capítulo, não é objetivo deste capítulo introduzir processos de modelagem avançada de dados ou detalhes de sistemas de *software* específicos para o tratamento e gerenciamento de dados, mas apresentar conceitos básicos importantes sobre organização de dados. Os conceitos apresentados neste capítulo se aplicam a qualquer base de dados e auxiliam na compreensão de mecanismos que garantem maior qualidade à criação e ao tratamento de dados em qualquer contexto de manipulação de informação.

A compreensão dos principais conceitos de bases de dados permite aproveitar melhor o potencial de cálculo das planilhas eletrônicas vistas neste curso. Uma boa organização dos dados facilita a aplicação de funções matemáticas e oferece uma boa compreensão visual da informação armazenada. A importância dos tipos de dados e domínios é percebida em especial quando a base de dados trabalha com outras ferramentas computacionais. Nos próximos capítulos, quando serão apresentadas as primeiras noções de programação, conceitos como tipos de dados, valores e domínios serão direta ou indiretamente importantes para realizar muitas operações.

Bases de Dados representam uma área fundamental da Computação, com ampla inserção no mercado de trabalho empresarial e nas universidades. Para os interessados, o conteúdo deste capítulo será novamente explorado e estendido em disciplinas mais avançadas da área de Computação, nas quais será possível explorar temas mais complexos da área, tais como Modelagem de Dados, Otimização de Bases de Dados, Linguagens de Recuperação e Segurança de Dados, entre muitos outros.

4.8 EXERCÍCIOS

1. O que é uma base de dados? Em sua resposta, procure indicar qual a sua importância, citar características e avaliar em quais situações o seu uso é recomendado.

2. Explique claramente qual a diferença essencial entre *dado* e *informação*. Ilustre sua explicação utilizando exemplos.

3. O que são *flat files*? O que diferencia *flat files* de arquivos texto convencionais? Como podem ser usados para apoiar operações com bases de dados?

4. Defina *tipos de dados* e *domínios*. Explique como a escolha dos tipos de dados e domínios pode influenciar o comportamento de operações básicas de bases de dados, como ordenações e operações matemáticas.

5. Explique o que são consultas (filtros). Como consultas podem ajudar a utilizar informações armazenadas em uma base de dados?

6. Em uma base de dados, você acha que é *sempre* possível ordenar seus dados? Em caso positivo, explique o motivo que torna a ordenação possível em todas as situações. Em caso negativo, explique quais regras devem ser obedecidas para que dados possam ser ordenados.

7. Considere a base de dados exibida na Tabela 4.12, baseada na tabela apresentada na Seção 4.5.2, mas com alterações no preenchimento de seus dados:

Tabela 4.12: Lista de opções de transporte para o Exercício 7.

Opção	Trajeto	Preço	Tempo	Qtd. Horários	Passeio
1 táxi	15000 m	R$ 40,00	10 min.	vários	0 m.
1 ônibus	20 km	R$ 5,00	25 min.	2 por hora	0,5 km.
2 ônibus	25 km	R$ 5,00	0,5 h.	20 por dia	200 m.
1 ônibus + 1 trem	21,5 km	R$ 6,00	23 min.	18 por dia	50 m.
3 ônibus	21,5 km	R$ 5,00	25 min.	3 por hora	1000 m.
2 trens	30 km	R$ 6,50	45 min.	40 por dia	300 m.
1 ônibus + 2 trens	35 km	R$ 7,00	50 min.	40 por dia	100 m.
2 ônibus + 1 trem	30 km	R$ 6,80	1 h.	18 por dia	0,5 km.

Nesse exemplo simples, é possível notar alguns problemas:

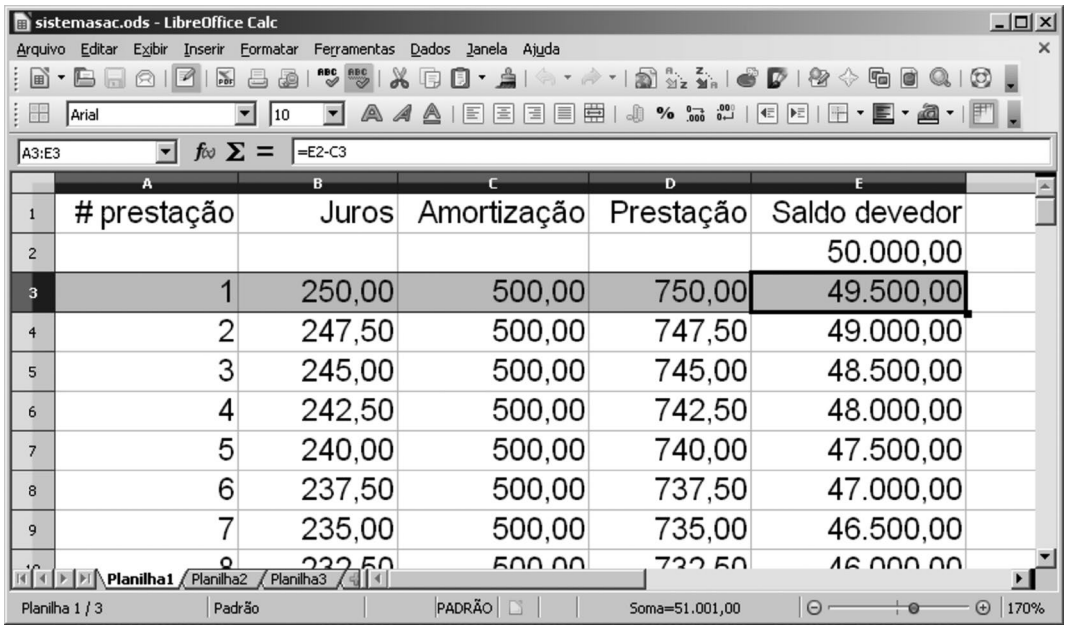
- *Domínios*: Os atributos *Trajeto*, *Tempo*, *Qtd. Horários* e *Passeio* apresentam um grave problema em relação à interpretação de seus domínios e os dados armazenados. Explique que problema é esse, indicando como ele pode prejudicar a manipulação da base de dados. Refaça a tabela, corrigindo o problema.

- *Organização*: Problemas de organização não são considerados erros conceituais, mas podem causar limitações indesejáveis na base de dados. Por exemplo, considere o atributo *Opção*. Da forma como está, é impossível ordenar os dados considerando critérios como menor quantidade de ônibus, ou menor variedade

de veículos (viagens usando somente ônibus, ou somente metrô, etc.). Remodele a tabela, de modo que seja possível corrigir essas limitações.

8. Um cidadão pretende fazer um financiamento para compra da casa própria pelo Sistema de Amortização Constante (SAC). Nesse sistema, cada prestação é resultado da soma da amortização com os juros mensais. A amortização mensal é dada pelo valor do financiamento dividido pelo número de parcelas do financiamento. Os juros que compõem cada prestação podem ser calculados com aplicação direta do percentual contratado em relação ao saldo devedor no mês anterior. Crie uma planilha que simule um financiamento de R$ 50.000,00 e taxa de juros de 0,5% ao mês com prazo de 100 meses. A planilha deve apresentar o número da prestação, os juros, a amortização, o valor da prestação e o saldo devedor. No final, devem ser computados somatórios das colunas de juros, de amortização e do valor da prestação. Considere o exemplo da Figura 4.13, onde foram criadas funções para cálculo dos valores das colunas, de modo que seja possível copiar as células desta linha para as próximas 99 linhas que correspondem às prestações restantes.

Figura 4.13: Sistema de amortização constante.



	A	B	C	D	E
1	# prestação	Juros	Amortização	Prestação	Saldo devedor
2					50.000,00
3	1	250,00	500,00	750,00	49.500,00
4	2	247,50	500,00	747,50	49.000,00
5	3	245,00	500,00	745,00	48.500,00
6	4	242,50	500,00	742,50	48.000,00
7	5	240,00	500,00	740,00	47.500,00
8	6	237,50	500,00	737,50	47.000,00
9	7	235,00	500,00	735,00	46.500,00

9. Considere o arquivo "idh.csv" com dados obtidos do sítio do Instituto de Pesquisa Econômica Aplicada (IPEA) e cujo dicionário de dados encontra-se na Tabela 4.13. Abra o arquivo no Calc, altere as opções de importação, atribuindo *ponto e vírgula* (";") como separador e desmarcando a opção *vírgula*. Salve o arquivo no formato nativo do Calc denominado *Planilha ODF (.ods)* e responda às seguintes questões, criando uma nova planilha para cada uma delas:

a. Criar um autofiltro para a planilha. Selecione as linhas referentes ao Estado do Acre em 1991 e que tenham IDH-M Longevidade maior que zero.

b. Classificar por Estado, em seguida por Município, e calcular subtotais (média dos atributos IDH-M Longevidade, IDH-M Educação, IDH-M Renda,..., Renda familiar *per capita* média) agrupando por Código do Município.

c. Classificar por Região e Estado, apenas para a pesquisa realizada em 1991, com valores de IDH-M Longevidade diferentes de zero, calculando subtotais agrupados primeiramente por Região e, em seguida, por Estado, calculando as médias dos atributos IDH-M Longevidade, IDH-M Educação, IDH-M Renda..., Renda familiar per capita média.

Tabela 4.13: Dicionário de dados do arquivo idh.csv.

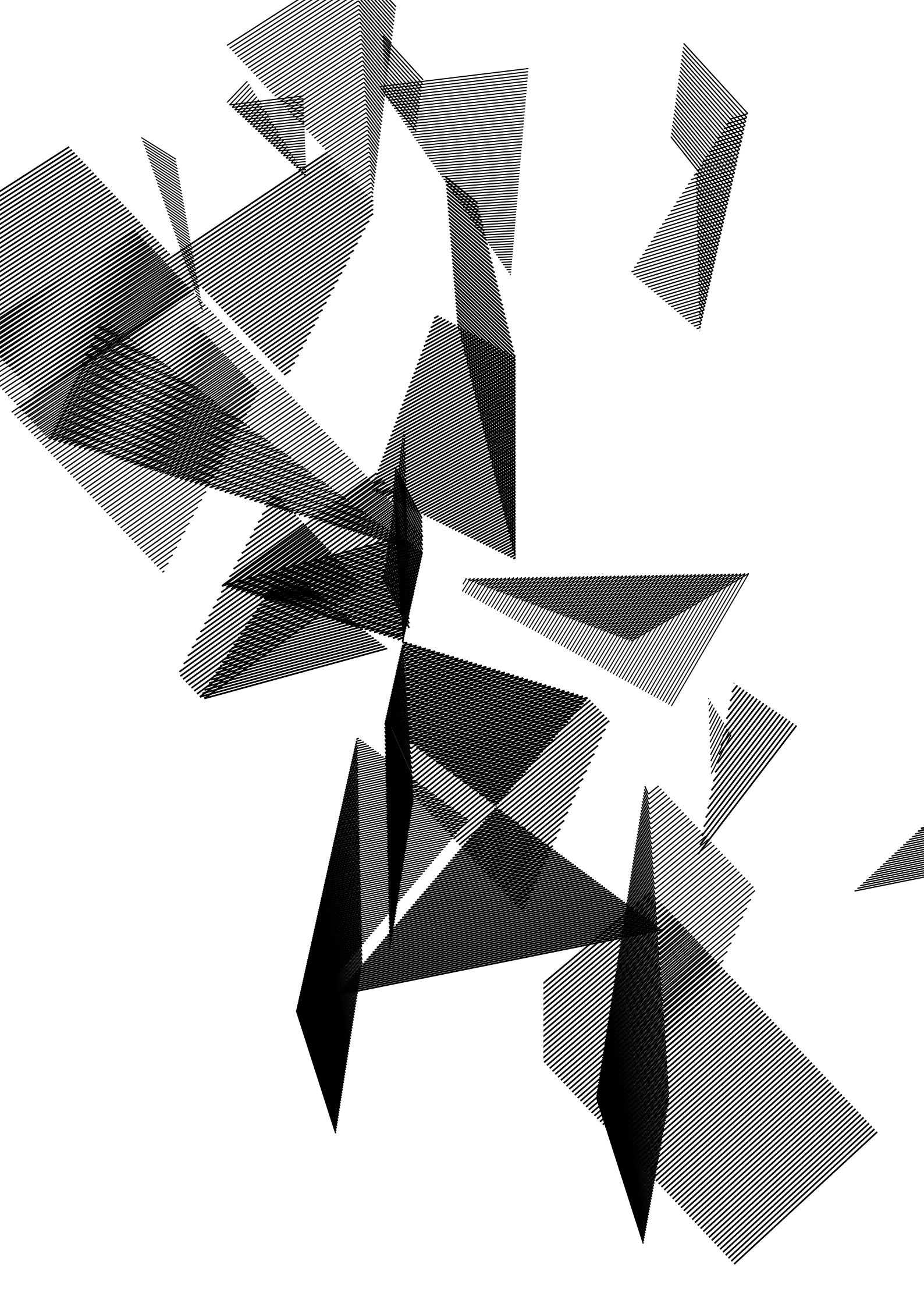
Atributo	Descrição
REALIZACAO	Ano de realização da pesquisa.
CODIGO	Código do município.
UF	Unidade federativa.
REGIAO	Sigla da região.
MUNICIPIO	Nome do município.
LONGVD	Índice IDH-M Longevidade.
EDUC	Índice IDH-M Educação.
RENDA	Índice IDH-M Renda.
ESP_VD_NC	Esperança de vida ao nascer.
POP_URB	% População que vive em área urbana.
IDHM	Índice Desenvolvimento Humano IDH-M.
POP_25AN	População de 25 ou + anos de idade.
POP_15AN	População de 15 ou + anos de idade.
POP_10AN	População de 10 a 14 anos.
POP_7AN	População de 7 a 14 anos.
TX_MRT	Taxa de mortalidade infantil, por mil nascidos vivos.
RD_FM	Renda familiar per capita média.

REFERÊNCIAS BIBLIOGRÁFICAS

ELMASRI, R. & NAVATHE, S.. *Sistemas de banco de dados*. São Paulo, Brasil: Pearson-Addison Wesley, 2006.

GILLENSON, M., PONNIAH, P., KRIEGEL, A., TRUNKHNOV, B., TAYOR, A. & POWELL, G. *Introdução a gerência de banco de dados*. Rio de Janeiro, Brasil: LTC, 2009.

SILBERSCHATZ, A., KORTH, H e SUDARSHAN, S. *Sistema de banco de dados*. Rio de Janeiro: Campus, 2006.



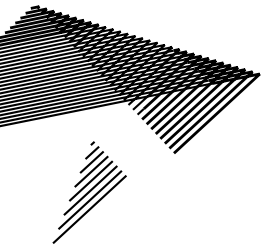
CAPÍTULO 5

LÓGICA DE PROGRAMAÇÃO: VARIÁVEIS E ESTRUTURAS SEQUENCIAIS

Harlen Costa Batagelo
João Paulo Gois
Letícia Rodrigues Bueno
Luiz Carlos da Silva Rozante
Ronaldo Cristiano Prati

5.1 INTRODUÇÃO

Nesta etapa do curso, apresentamos conceitos básicos de desenvolvimento de programas para serem executados em dispositivos computacionais, como o computador. O desenvolvimento desses programas envolve uma maneira específica de pensar e planejar estratégias para que esses dispositivos possam ser usados para resolver problemas. Resolução de problemas é a habilidade de formular problemas, pensar criativamente em soluções e expressá-las de maneira clara e precisa, a fim de que, se e quando nos depararmos novamente com o mesmo problema, poderemos seguir a descrição da solução como uma receita ou, neste caso, delegar a sua execução a um dispositivo (normalmente o computador) que irá executar as instruções.



Essa maneira de pensar e resolver problemas computacionalmente combina e integra algumas características de diferentes áreas como Matemática, Engenharia e Ciências Naturais. Da Matemática, usamos algum formalismo para representar ideias (particularmente quando envolvem computação). Da Engenharia, a maneira de projetar coisas, juntar componentes em sistemas maiores e avaliar e decidir quando temos que ponderar diferentes alternativas. Das Ciências Naturais, a habilidade de observar o comportamento de sistemas complexos, formular modelos e hipóteses e testar predições. Aprender a desenvolver programas é uma excelente oportunidade para praticar habilidades de resolução de problemas e, consequentemente, de criatividade. *O objetivo deste capítulo é* mostrar como atribuir instruções a um computador, de maneira que ele possa nos auxiliar a resolver problemas. Para isto precisamos de dois ingredientes: linguagem e lógica. Apresentamos, através de situações do dia a dia, como intuitivamente delegamos tarefas prescrevendo instruções ou realizamos tarefas seguindo instruções em uma ordem lógica, de modo a resolver problemas. Utilizamos estes exemplos como motivação para introduzir os conceitos básicos de lógica de programação. Com o objetivo de auxiliar na ilustração dos conceitos apresentados, utilizamos ao longo dos capítulos 5, 6 e 7 uma ferramenta chamada RoboMind, disponível em http://www.robomind.net/. Neste aplicativo, o objetivo é controlar um robô virtual através da prescrição de instruções, usando uma linguagem

que contém vários elementos de lógica de programação. Ao final de cada capítulo, exemplificamos os mesmos conceitos em uma linguagem de programação de propósito mais geral nativa de um *software* chamado Scilab. Para maiores informações sobre estas ferramentas, indicamos as referências RoboMind (2011) e Scilab (2011).

O objetivo deste capítulo NÃO é o estudo e o desenvolvimento de programas complexos. Embora muito importante, esse assunto é trabalhado em disciplinas específicas dos cursos da área de Computação. O desenvolvimento desse tema depende de aprendizado e experiência progressivos, que fogem aos objetivos desta disciplina. No entanto, embora seja possível aprofundar-se muito no assunto, programas um pouco mais elaborados são estudados no curso de Processamento na Informação. Como uma bibliografia complementar para a disciplina de Bases Computacionais da Ciência, e aos interessados em aprofundar-se no estudo de algoritmos, indicamos os livros de Forbellone *et al.* (2005) e Puga e Rissetti (2009).

O restante deste capítulo está distribuído conforme a seguinte organização. As seções 5.2 e 5.3 fazem uma analogia entre as tarefas que executamos diariamente e as tarefas que o computador pode executar. Os conceitos são ilustrados através do RoboMind, sendo reforçados pelas atividades em aula na Seção 5.4 e por exemplos com o Scilab na Seção 5.5. A Seção 5.6 apresenta as considerações finais do capítulo e exercícios complementares são fornecidos na Seção 5.7.

5.2 DELEGANDO TAREFAS: O CONCEITO DE INSTRUÇÃO

Em nosso cotidiano, a todo momento, costumamos delegar tarefas em forma de instruções:

> (*i*) Você chega faminto a uma lanchonete e faz um pedido de uma refeição: um lanche, um suco e uma sobremesa. Também solicita que seja preparada "com urgência", pois está faminto. Se a refeição for preparada como você gostaria, a instrução foi bem dada e a lanchonete (a executora da instrução) foi eficiente;
>
> (*ii*) Para você não perder o horário, você já instruiu algum equipamento que possui a função de despertador (relógio cuco, telefone celular, rádio-relógio). Se você ajustou corretamente o horário e o volume sonoro alto o suficiente para você acordar, você despertará.

Estes são exemplos de como costumamos delegar tarefas por prescrever instruções. No primeiro caso, as instruções foram dadas através de comunicação oral na Língua Portuguesa, enquanto no segundo foram instruções específicas para o equipamento adotado.

As instruções diferem de acordo com as funcionalidades que o equipamento é capaz de oferecer. Por exemplo, uma campainha tem a única função de ser tocada. Em uma calculadora simples, as instruções correspondem às operações aritméticas de soma, subtração, multiplicação e divisão. Já em um aparelho de televisão, dentre as diversas instruções que pode haver, normalmente estão os comandos de avançar/retroceder o canal, aumentar/diminuir o volume do som e a função liga/desliga. Em todos os casos,

temos um conjunto de instruções específicas básicas que são as ações fundamentais que o equipamento é capaz de realizar.

5.3 SEQUÊNCIA DE INSTRUÇÕES: ESTRUTURAS SEQUENCIAIS

Nem sempre as instruções básicas são capazes de resolver um problema. Nesse caso, instruções podem ser compostas de modo a produzir resultados que não poderiam ser obtidos com apenas uma única instrução. Por exemplo, você pode combinar com algum morador da sua casa que, se a campainha for tocada três vezes seguidas, com um intervalo de aproximadamente dois segundos entre cada toque, é você que está na porta e ela pode ser aberta por quem estiver dentro da casa.

Vamos agora ver um exemplo um pouco mais "computacional". Suponha que desejamos calcular o resto da divisão de 9 por 2 usando uma calculadora que possui as operações de soma, subtração, multiplicação e divisão, mas não possui a operação de resto da divisão. Você, sendo um aluno astuto, pensou na seguinte solução:

1. Dividir 9, o dividendo, por 2, o divisor, obtendo o resultado 4,5;
2. Considerar apenas a parte inteira do resultado, ou seja, 4;
3. Multiplicar a parte inteira do resultado pelo divisor, obtendo $4 \times 2 = 8$;
4. Subtrair o dividendo pelo resultado da multiplicação anterior, isto é, $9 - 8 = 1$, que é o resultado esperado.

Assim, conseguimos obter o resto da divisão entre dois números usando uma composição das operações de multiplicação, subtração e divisão. De fato, não apenas a operação de resto da divisão, mas uma infinidade de operações mais complexas pode ser obtida através da composição de operações simples.

A sequência de instruções importa?

A ordem em que uma sequência de instruções é executada pode alterar o resultado obtido. Por exemplo, quando você pede a refeição na lanchonete, você não espera que o garçom traga a sobremesa antes do lanche. Outro exemplo em que a ordem é importante é no caso de, antes de executar a ação de vestir seu calçado, primeiro você executa a ação de vestir a meia.

Mas nem sempre a ordem das instruções importa. Por exemplo, usando ainda o exemplo da calculadora, se desejamos calcular o resultado da expressão $1+1-2$, podemos calcular primeiramente $1+1$ e então subtrair 2 do resultado. Mas também podemos calcular $1-2$ em primeiro lugar e só então somar 1 ao resultado. O resultado será o mesmo neste caso, não importa a ordem das operações. Entretanto, as expressões $(1+1)\times 2$ e $1+(1\times 2)$, na qual apenas os parênteses foram modificados, não são equivalentes, já que $(1+1)\times 2 = 4$ e $1+(1\times 2) = 3$. Assim, em geral, além de saber quais instruções devemos utilizar, devemos saber exatamente qual a ordem em que essas instruções serão executadas.

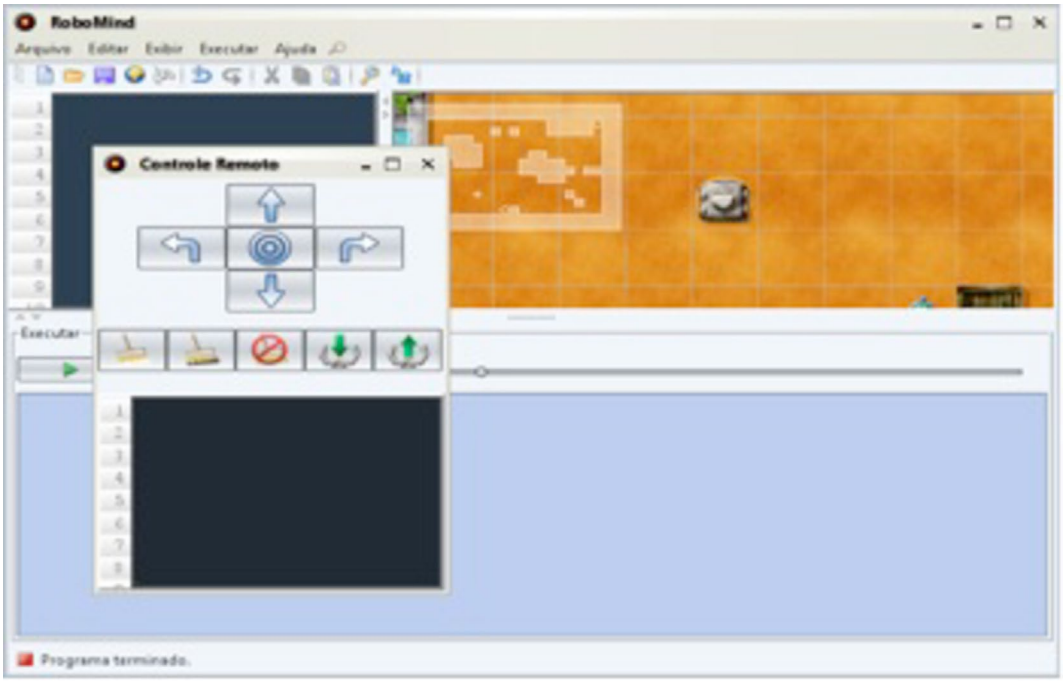
Ilustrando instrução e sequência de instruções com RoboMind

Vamos exemplificar o conceito de instrução e sequência de instruções usando o *software* RoboMind. O objetivo do RoboMind é criar um programa para que o robô execute alguma tarefa. O robô tem algumas instruções básicas, tais como:

- mover-se;
- marcar (pintar) o chão;
- observar o ambiente; e
- pegar um objeto.

As instruções para o robô podem ser dadas via controle remoto ou escrevendo um programa com comandos. Vamos primeiro experimentar como controlar o robô usando o controle remoto. No programa RoboMind, abra o controle remoto (`Executar`⟶ `Controle Remoto`) e experimente movimentar o robô. Na Figura 5.1 é apresentada uma janela do RoboMind com o controle remoto aberto.

Figura 5.1: Janela do RoboMind com o controle remoto aberto.



Observe que na parte de cima do controle remoto há alguns botões com setas para fazer o robô se mover para cima, para baixo, para a direita e para a esquerda. O desenho simbolizando um alvo faz o robô voltar para a posição inicial no cenário. Experimente usar os botões direcionais para movimentar o robô no ambiente. Observe que, logo abaixo dos botões, à medida que você move o robô, vão aparecendo os comandos associados aos movimentos que você mandou o robô executar através do controle remoto.

Observe bem os comandos que são gerados à medida que o robô se movimenta. Antes, porém, observe que se a interface do RoboMind não estiver em Português, você pode

trocar a língua em `File⟶Settings` e selecionar o português como idioma. Quando se pressiona seta para cima (↑), aparece o comando `andarFrente(1)`, o que faz com que o robô ande um passo para frente. Do mesmo modo, quando se pressiona seta para baixo (↓), aparece o comando `andarTrás(1)`, que faz o robô andar um passo para trás. As setas para a direita (↱) e para a esquerda (↰) fazem com que o robô vire 90^0 para a direita ou esquerda, exibindo os comandos `virarDireita()` e `virarEsquerda()`, respectivamente.

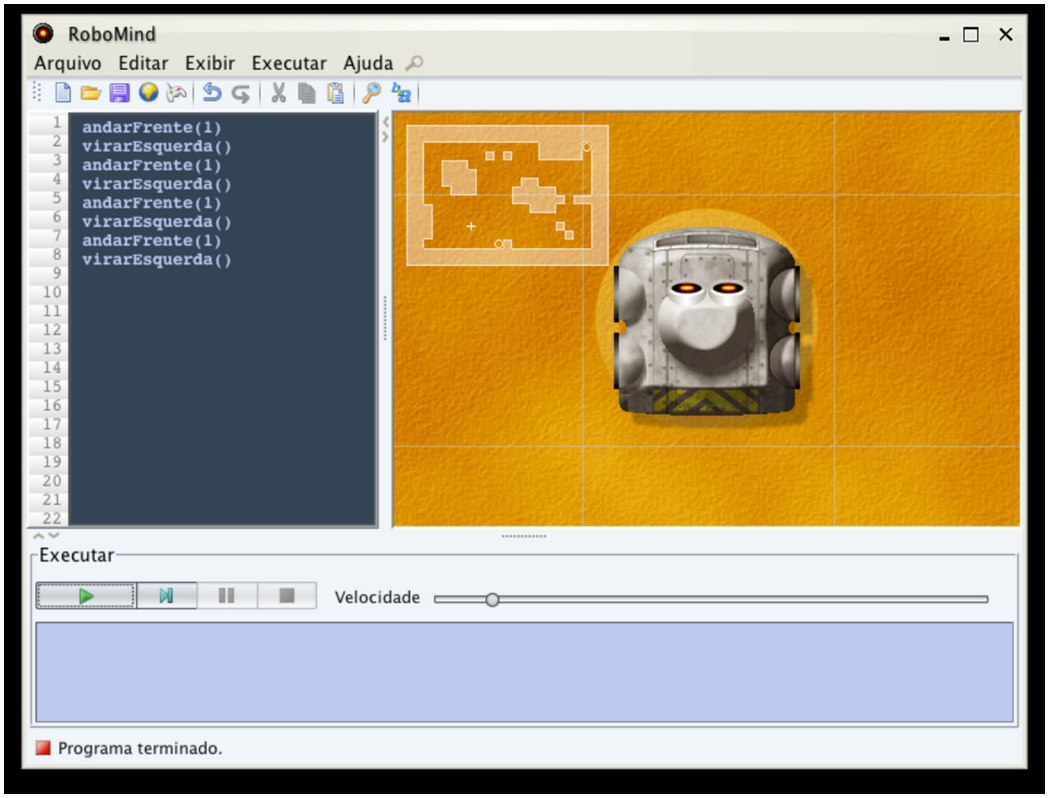
Usando o controle remoto, tente fazer com que o robô percorra um quadrado andando um passo a frente e virando à esquerda, até voltar para sua posição inicial. Para fazer isso, você provavelmente deve ter usado a sequência de comandos ↑,↰,↑,↰,↑,↰,↑,↰, o que gerou o programa da Listagem 5.1.

Listagem 5.1: Programa para fazer um quadrado de lado 1.

```
1  andarFrente (1)
2  virarEsquerda ()
3  andarFrente (1)
4  virarEsquerda ()
5  andarFrente (1)
6  virarEsquerda ()
7  andarFrente (1)
8  virarEsquerda ()
```

Em vez de usar o controle remoto, vamos agora instruir o robô a percorrer o mesmo quadrado usando comandos. Para fazer isso, você pode usar a mesma sequência de comandos que foi gerada quando você usou o controle remoto. Feche o controle remoto e coloque essa sequência de comandos na janela de comandos, do lado esquerdo do cenário, como mostrado na Figura 5.2, e execute o programa clicando em `Executar`, na parte inferior da janela. Se tudo estiver certo, o robô deve percorrer um quadrado e voltar à posição inicial, como aconteceu anteriormente quando usamos o controle remoto.

Figura 5.2: Janela do RoboMind com comandos para o robô percorrer um quadrado.



Agora, imagine que, em vez de percorrer o quadrado andando um único passo para a frente, você quer que o robô dê dois passos antes de cada virada à esquerda. Utilizando o controle remoto, isso pode ser feito clicando em ↑,↑,↰,↑,↑,↰,↑,↑,↰,↑,↑,↰, o que levaria ao programa da Listagem 5.2.

Listagem 5.2: Programa para fazer um quadrado de lado 2: primeira versão.

```
1   andarFrente (1)
2   andarFrente (1)
3   virarEsquerda ()
4   andarFrente (1)
5   andarFrente (1)
6   virarEsquerda ()
7   andarFrente (1)
8   andarFrente (1)
9   virarEsquerda ()
10  andarFrente (1)
11  andarFrente (1)
12  virarEsquerda ()
```

Você já deve ter notado que existe uma diferença entre as instruções de andar e virar. Nos comandos `andarFrente` e `andarTrás`, entre os parênteses existe o número 1, o que não ocorre nos comandos `virarEsquerda` e `virarDireita`. Se você intuiu que esse número está relacionado ao tamanho do passo que o robô anda, você está certo! Para fazer o robô andar um quadrado no qual ele dá dois passos antes de

virar à esquerda, ao invés de repetir duas vezes o comando `andarFrente(1)`, você pode usar uma única vez o comando `andarFrente(2)`. O número entre parênteses é chamado argumento ou parâmetro da instrução `andarFrente(n)`, no qual `n` é uma *variável* que representa o valor que deve ser atribuído ao parâmetro da instrução. Dessa maneira, para fazer com que o robô percorra um quadrado de lado dois, você pode usar o programa da Listagem 5.3.

Listagem 5.3: Programa para fazer um quadrado de lado 2: segunda versão.

```
1  andarFrente (2)
2  virarEsquerda ()
3  andarFrente (2)
4  virarEsquerda ()
5  andarFrente (2)
6  virarEsquerda ()
7  andarFrente (2)
8  virarEsquerda ()
```

Veja que, em termos práticos, esse programa tem o mesmo funcionamento do programa anterior, mas usa menos instruções. Escrever instruções parece, a princípio, mais trabalhoso do que usar o controle remoto. Entretanto, utilizando o controle remoto, você está limitado aos comandos embutidos nos botões exibidos na interface do programa RoboMind, enquanto a linguagem de comandos proporciona muito mais flexibilidade! Além disso, você pode armazenar o programa em um arquivo, e executá-lo quantas vezes você quiser, ao passo que com o controle remoto, se quiser percorrer um novo quadrado, você teria que repetir toda a sequência de botões. Além desses comandos de movimento, o RoboMind tem também os comandos `andarNorte(n)`, `andarSul(n)`, `andarLeste(n)` e `andarOeste(n)`, nos quais *n* é um parâmetro que pode ser usado da mesma maneira que `andarFrente(n)`. Experimente usá-los para percorrer o ambiente.

Além dos comandos para movimentar o robô no ambiente, o RoboMind provê comandos para o robô interagir com o ambiente com ações como pintar o chão de branco ou preto e pegar um objeto do chão. Após executar a instrução `pintarBranco()` (ou `pintarPreto()`), para cada movimento do robô, o caminho pelo qual o robô passou é pintado de branco (e preto, respectivamente). Para parar de pintar, é preciso dar a instrução `pararPintar()`. Na Listagem 5.4 temos o código de um programa que desenha um quadrado branco no chão.

Listagem 5.4: Programa para fazer um quadrado branco.

```
1  pintarBranco ()
2  andarFrente (1)
3  virarEsquerda ()
4  andarFrente (1)
5  virarEsquerda ()
6  andarFrente (1)
7  virarEsquerda ()
8  andarFrente (1)
9  virarEsquerda ()
10 pararPintar ()
```

Os comandos `pegar()` e `soltar()` servem para pegar e soltar um objeto, respectivamente. O comando `pegar()` só funciona se o objeto estiver na célula imediatamente à frente e o robô não estiver carregando nenhum outro objeto. O comando `soltar()` coloca o objeto na célula imediatamente à frente, caso não tenha nenhum obstáculo nessa célula.

5.4 ATIVIDADES EM AULA

Exercício resolvido

Vamos agora realizar um exercício completo passo a passo, para que você possa praticar, resolvendo depois outros problemas, e treinar lógica de programação. Para realização deste exercício, considere que:

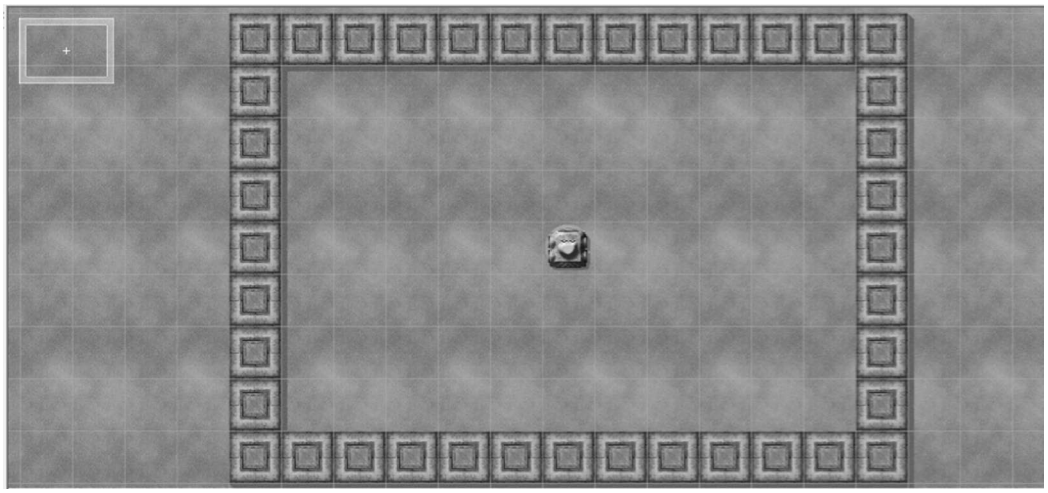
- O sentido norte–sul equivale ao eixo Y e o sentido leste–oeste equivale ao eixo X.
- O ponto (0,0) é a origem.

Vamos primeiro criar um mapa, que será o cenário para o nosso robô. Abra um programa de edição de texto, como o Notepad, e insira as linhas apresentadas na Figura 5.3. Observe que as linhas que começam com "#" são os comentários, e não precisam ser inseridas; logo a seguir, a partir da linha de comando `map`, estão representados os caracteres para desenhar o mapa.

Figura 5.3: Criação de um cenário para o RoboMind.

Quando você terminar de definir o seu mapa você deve salvá-lo com um nome que escolher, mas com a extensão "`.map`" em vez da extensão "`.txt`". Agora, inicie o RoboMind para testar o seu mapa. É só clicar em `Arquivo`⟶`Abrir mapa`. Na Figura 5.4 é mostrada a visualização do mapa dentro do RoboMind.

Figura 5.4: Visualização do mapa criado dentro do RoboMind.



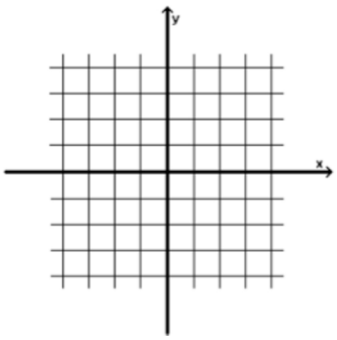
O objetivo desse exercício é programar o robô para que ele passe pelas coordenadas: (3,-2), (2, 3), (-3, 1), considerando que cada uma das coordenadas em que o robô parar, será a nova origem. Antes do programar o robô, vamos tentar entender melhor o problema para planejar como iremos resolvê-lo, analisando as seguintes perguntas:

a) Represente as coordenadas em um plano cartesiano, como mostrado na Figura 5.5;

b) Qual a coordenada final em relação à origem?

c) Que rota o robô realizou?

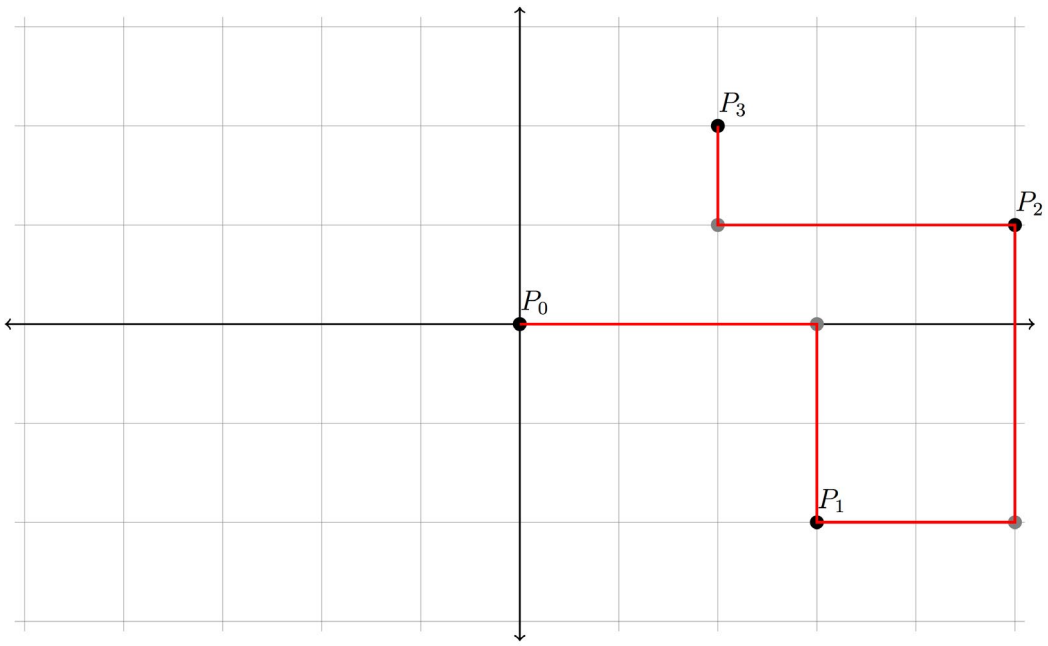
Figura 5.5: Representação de um plano cartesiano.



Primeiro passo

Fazer o mapa: um tabuleiro com 10 por 6 casas, conforme a Figura 5.6. Colocar o `Robô` na posição inicial P_0 para fazê-lo percorrer os pontos P_1, P_2 e P_3.

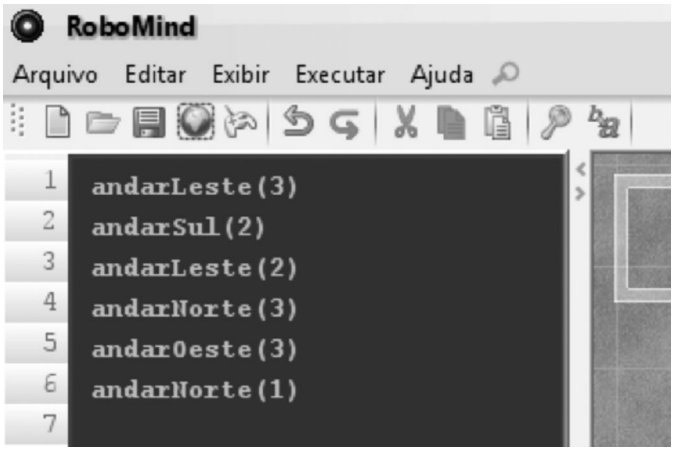
Figura 5.6: Um tabuleiro e pontos para o robô percorrer.



Segundo passo

Inserir os comandos no RoboMind, utilizando os comandos disponíveis, para que o robô percorra os pontos solicitados (um possível caminho está destacado em vermelho na Figura 5.6). Como não é possível andar na diagonal, você deve passar por pontos intermediários antes de chegar às coordenadas solicitadas no problema (esses pontos estão destacados em cinza na Figura 5.6). Trace um possível caminho que passe por esses pontos e tente imaginar os comandos para percorrer esse caminho. Uma possível resposta é mostrada na Figura 5.7.

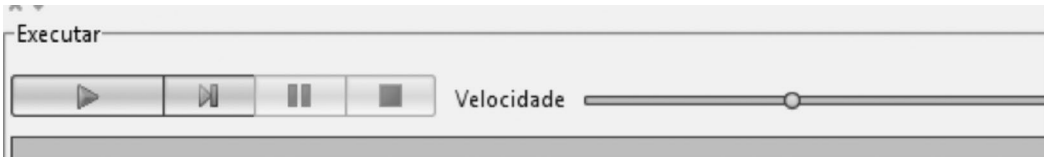
Figura 5.7: Uma possível resposta para o exercício de percorrer os três pontos.



Terceiro passo

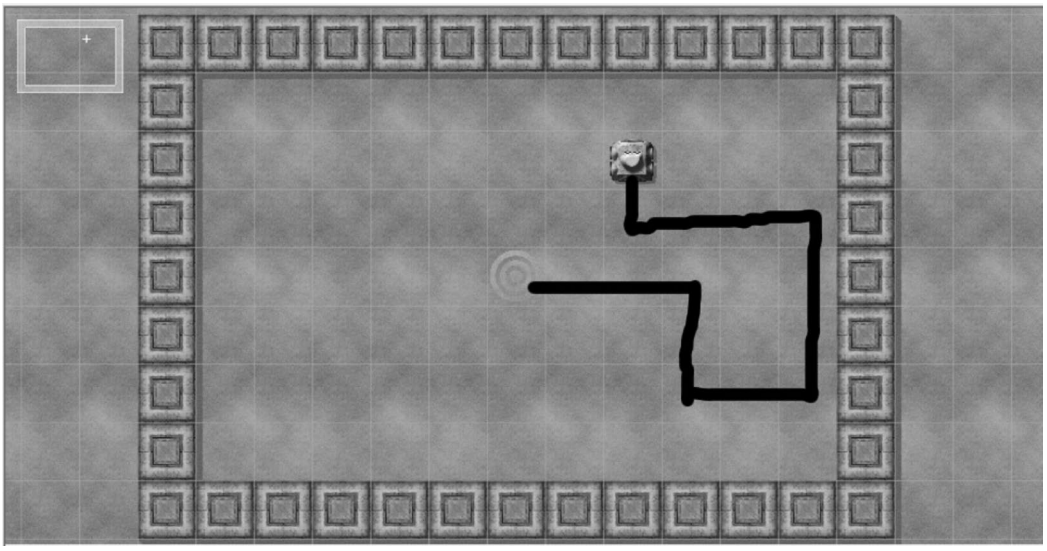
Executar o programa. Para executar o programa, você deve clicar no botão `Executar`, mostrado na Figura 5.8.

Figura 5.8: Controle de execução do programa.



O robô deve percorrer o caminho indicado na Figura 5.9, e parar na posição final.

Figura 5.9: Um possível caminho para o exercício de percorrer os três pontos.



Exercícios propostos

Caro aluno, para que você possa desenvolver a lógica de programação é importante que você tente primeiro construir os programas sem o auxílio da ferramenta `Controle Remoto` do RoboMind.

1. Acesse o RoboMind e teste os programas vistos na seção anterior;

2. Escreva um programa para que o robô pinte a letra "`E`" no chão;

3. É possível construir um programa diferente daquele que você fez no exercício anterior e que faça a mesma coisa? Se sim, então o faça; se não, então justifique. Obs.: vamos assumir que dois programas são diferentes se suas sequências de instruções não são idênticas;

4. Escreva um programa para que o robô desenhe um "alvo", isto é, pinte "círculos" (aqui, na verdade, quadrados) concêntricos em preto e branco alternadamente. Faça de tal modo que a "mosca" (centro) seja um ponto preto e que haja seis quadrados à sua volta;

5. Escreva um programa que faça com que o robô escreva o seu nome no chão;

6. Similarmente ao exercício anterior, escreva um programa para que o robô escreva o seu primeiro nome no chão, mas agora alternando as cores das letras com preto e branco. Comece com a primeira letra sendo branca;

7. No ambiente dado pelo mapa padrão (`default.map`), observe o ambiente em que o robô está e escreva um programa em que o robô busque o objeto mais próximo e o coloque no ponto de partida;

8. Faça um programa para que o robô, no ambiente dado pelo mapa padrão (`default.map`), pegue o objeto mais próximo e o coloque junto do mais distante. Seu programa deve ter no máximo 7 instruções.

5.5 ILUSTRANDO OS CONCEITOS APRENDIDOS COM O SCILAB

Neste capítulo, vimos vários exemplos de como prescrevemos instruções para executar determinadas tarefas. Nesta seção, veremos como o computador pode nos auxiliar na resolução de problemas usando o Scilab.

O primeiro problema a resolver é o cálculo da área `A`, do perímetro `P` e diagonal `D` de um quadrado de lado `x=2` unidades de medida (u.m.). Como sabemos, a área é $A=x^2$, o perímetro é `P=4x` e a diagonal é $D=\sqrt{2}*x$. Agora já temos todas as informações necessárias para resolver o problema no computador. Resta conhecer a linguagem que o Scilab utiliza para resolver este problema para nós. A esta "conversa" escrita, daremos o nome de *código-fonte*. Na Listagem 5.5 tem-se o nosso primeiro código-fonte em Scilab para resolver o problema descrito.

Listagem 5.5: Primeiro programa no *software* Scilab.

```
1  x = 2;
2  A = x*x
3  P = 4*x
4  D = sqrt(2)*x
```

Este código-fonte deve ser digitado em um arquivo em formato texto usando um editor tal como o Bloco de Notas do Microsoft Windows, gEdit do Gnome-Linux ou o próprio editor presente no Scilab (SciNotes), que pode ser acessado pelo primeiro ícone da janela do Scilab (Figura 5.10). O código da Listagem 5.5, por exemplo, foi digitado no SciNotes (Figura 5.11) e salvo com o nome `quadrado.sce`, em que `sce` é a extensão padrão do Scilab.

Figura 5.10: Console do Scilab: ícone para abrir o SciNotes (Figura 5.11).

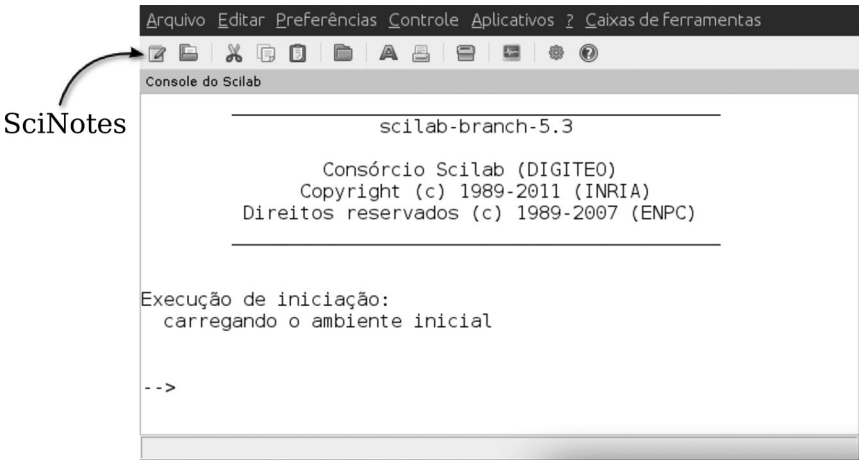
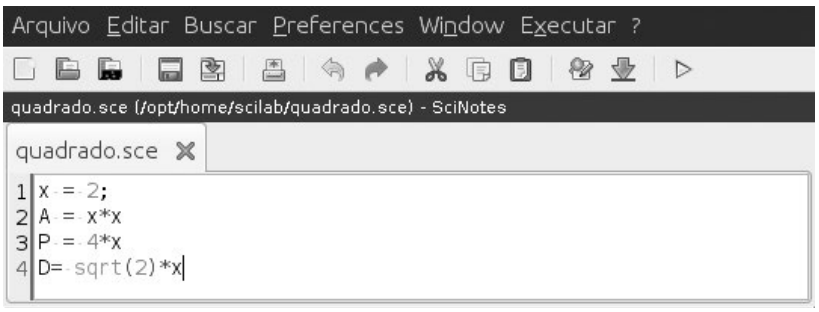


Figura 5.11: Janela do SciNotes para editar o código-fonte.



Para executarmos as instruções contidas no nosso código-fonte, devemos usar o comando `exec` no console do Scilab, do seguinte modo:

```
exec("quadrado.sce");
```

Outra forma de executar o programa é através do menu `Arquivo/Executar`, selecionando o arquivo `quadrado.sce`. A execução do programa anterior gera como resposta:

```
 A   =
    4.
 P   =
    8.
 D   =
    2.8284271
```

Em cada uma das linhas do código da Listagem 5.5 existe ao menos uma das letras `X`, `A`, `P` e `D`, que são as *variáveis* do nosso problema. Por exemplo, na Linha 1 (`x=2;`), "atribuímos" o valor 2 à variável `x` ou, em outras palavras, armazenamos o valor 2 na variável `x`. Isto é, estamos considerando que o lado de nosso quadrado é de comprimento de 2 u.m. Podemos ler a Linha 1 por "o valor 2 é atribuído a `x`", ou "`x` recebe 2".

Na Linha 2 (`A=x*x`), realizamos a operação x^2 para, em seguida, atribuir este resultado à variável `A`. A Linha 3 é executada de forma análoga. Observe que na Linha 4 aparece a instrução `sqrt(2)`, que significa $\sqrt{2}$. Na verdade, poderíamos calcular a raiz quadrada de qualquer número real positivo `n` no lugar de 2.

5.5.1 ENTRADA E SAÍDA

Agora queremos executar novamente o programa anterior, só que para um quadrado de lado `x=5`. A forma mais imediata de se fazer esta tarefa é modificando a Linha 1, substituindo o 2 por 5. E que tal deixar que o usuário do seu programa decida o comprimento a ser inserido? Experimente substituir a Linha 1 pela seguinte instrução:

```
x = input("Entre com o comprimento do lado do quadrado:");
```

Salve o programa e execute-o. O programa aguardará que o usuário entre com o tamanho do lado. Experimente entrar com o valor 5, seguido da tecla *enter*. Você terá algo como:

```
Entre com o comprimento do lado do quadrado: 5
 A  =
    25.
 P  =
    20.
 D  =
    7.0710678
```

Dizemos então que seu programa tem como *entrada*, via teclado, o valor da variável `x` e, como *saída* no monitor, as variáveis `A`, `P` e `D`. Vamos tornar a saída mais amigável. As variáveis `A`, `P` e `D` estão sendo mostradas porque omitimos o ponto-e-vírgula no final das linhas 2, 3 e 4. Vamos inserir um ponto-e-vírgula no final de cada uma destas linhas e acrescentar uma nova instrução, conforme apresentado na Listagem 5.6.

Listagem 5.6: Programa com variáveis de entrada e variáveis de saída.

```
1  x = input ("Entre com o comprimento do lado do quadrado:");
2  A = x*x;
3  P = 4*x;
4  D = sqrt (2)*x;
5  printf ("A área é %f\n O perímetro é %f\n A diagonal é %f
   ",A,P,D);
```

A saída deste programa, para a entrada `x=5`, será:

```
Entre com o comprimento do lado do quadrado: 5
A área é 25.000000
O perímetro é 20.000000
A diagonal é 7.071068
```

O comando `printf` é uma instrução de *saída formatada*. Por exemplo, os caracteres `\n` formatam a saída de modo que ela "pule" para a próxima linha. Experimente tirar os `\n` e confira o que acontece. Já os caracteres `%f` formatam a saída de variáveis para representação em *ponto flutuante* (um subconjunto dos números reais). Observe que existem três ocorrências dos caracteres `%f`, uma para cada uma das variáveis `A`, `P` e `D`, respectivamente. Substituindo o `%f` por `%d`, a saída das variáveis será formatada para os números inteiros:

```
Entre com o comprimento do lado do quadrado: 5
A área é 25
O perímetro é 20
A diagonal é 7
```

No exemplo da Listagem 5.7, mostramos como desenhar um gráfico de uma função $f(x)=ax^2+bx+c$ no intervalo $[x_i, x_f]$, em que os valores de `a, b, c`, x_i e x_f são dados pelo usuário e $x_i < x_f$.

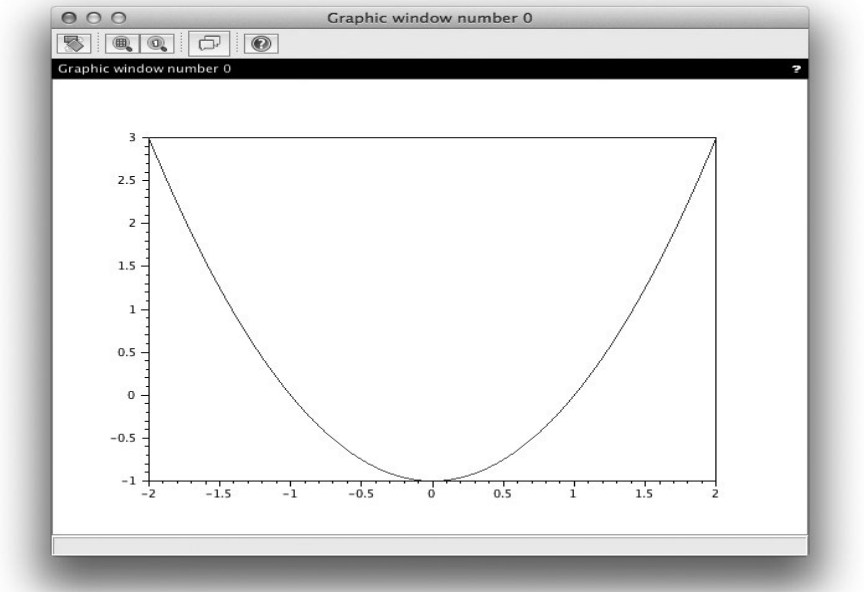
Listagem 5.7: Programa para o gráfico de uma função.

```
1  a = input ("Digite o valor de a: ");
2  b = input ("Digite o valor de b: ");
3  c = input ('Digite o valor de c: ");
4  xi = input ("Digite o valor de xi: ");
5  xf = input ("Digite o valor de xf: ");
6  x = xi :0.01: xf;
7  f = a*x^2 + b*x + c;
8  plot (x,f);
```

A seguir, mostramos um exemplo de execução cujo resultado é apresentado na Figura 5.12.

```
Digite o valor de a: 1
Digite o valor de b: 0
Digite o valor de c: -1
Digite o valor de xi: -2
Digite o valor de xf: 2
```

Figura 5.12: Gráfico da equação $f(x)=x^2-1$ no intervalo [–2,2].



5.6 CONSIDERAÇÕES FINAIS

Este capítulo introduziu alguns conceitos básicos de lógica de programação, em especial, a atribuição de tarefas ao computador através de instruções. Como indicamos no início do capítulo, o objetivo não é introduzir programas complexos, detalhes de implementação ou técnicas elaboradas de resolução computacional de problemas, mas apresentar conceitos básicos para a resolução de problemas simples através de computadores.

A compreensão dos principais conceitos de lógica de programação permite aproveitar melhor o potencial dos computadores disponíveis, atribuindo a eles tarefas relativamente simples, porém muito específicas, que podem ser automatizadas e para as quais ainda não dispomos de aplicações prontas no mercado de *software*. No próximo capítulo, apresentamos uma estrutura que concede maior poder computacional ao fornecer uma forma de instruir os computadores para tomar algumas decisões: as estruturas condicionais.

Atendendo a necessidades de diversos setores do mercado e da pesquisa científica, a programação representa uma área fundamental da Computação, que é vital para o desenvolvimento tecnológico de uma nação. Para os interessados, o conteúdo deste capítulo será aprofundado em disciplinas mais avançadas da área de Computação, nas quais será possível explorar temas mais complexos tais como Técnicas de Resolução de Problemas, Análise de Algoritmos, Programação Orientada a Objetos, Programação para Web, entre outros.

5.7 EXERCÍCIOS

5.7.1 EXERCÍCIOS COM O ROBOMIND

1. Faça um programa para que o robô, no ambiente dado pelo mapa `copyLine1.map`, agrupe todos os objetos junto àquele que está mais distante da posição inicial do robô;

2. Similarmente ao exercício anterior, escreva um programa para que o robô escreva o seu nome inteiro no chão, mas agora alternando as cores das letras em preto e branco. Comece com a primeira letra sendo preta;

3. Faça um programa para que o robô, no ambiente dado pelo mapa `passBeacons.map`, saia da posição inicial (salão à esquerda) e vá para o ponto marcado em branco, no salão à direita. Para realizar este trajeto o robô precisa mover alguns objetos. Entretanto, o robô deverá recolocar os objetos em seus locais de origem, antes de ir em direção à posição final. Sendo assim, ao final os objetos devem estar na mesma posição em que estavam no começo;

4. Usando os recursos que você já apreendeu do RoboMind, é possível construir um programa que faça com que o robô pinte de preto a célula à sua esquerda, caso ela esteja pintada de branco? Se sim, faça esse programa; se não, justifique porque não é possível: quais recursos seriam necessários para isso?

5.7.2 EXERCÍCIOS COM O SCILAB

1. Acesse o Scilab e teste os exemplos vistos na Seção 5.5;
2. Faça um programa que:

a) Leia do teclado 5 números inteiros naturais em ordem crescente, representando cada um a idade de um indivíduo;

b) Calcule e escreva: média, mediana, desvio padrão e variância destas 5 idades.

3. A conversão de graus Fahrenheit para Celsius é dada pela expressão

$$C = \frac{F - 32}{1.8}$$

e a conversão de graus Kelvin para graus Celsius é dada por

$$C = k - 273.15$$

Faça um programa que calcule e escreva duas tabelas: uma de graus Celsius em função de graus Fahrenheit e outra de Celsius em função de graus Kelvin. Ambos, graus Fahrenheit e Kelvin, variam de 50 a 60 de 1 em 1.

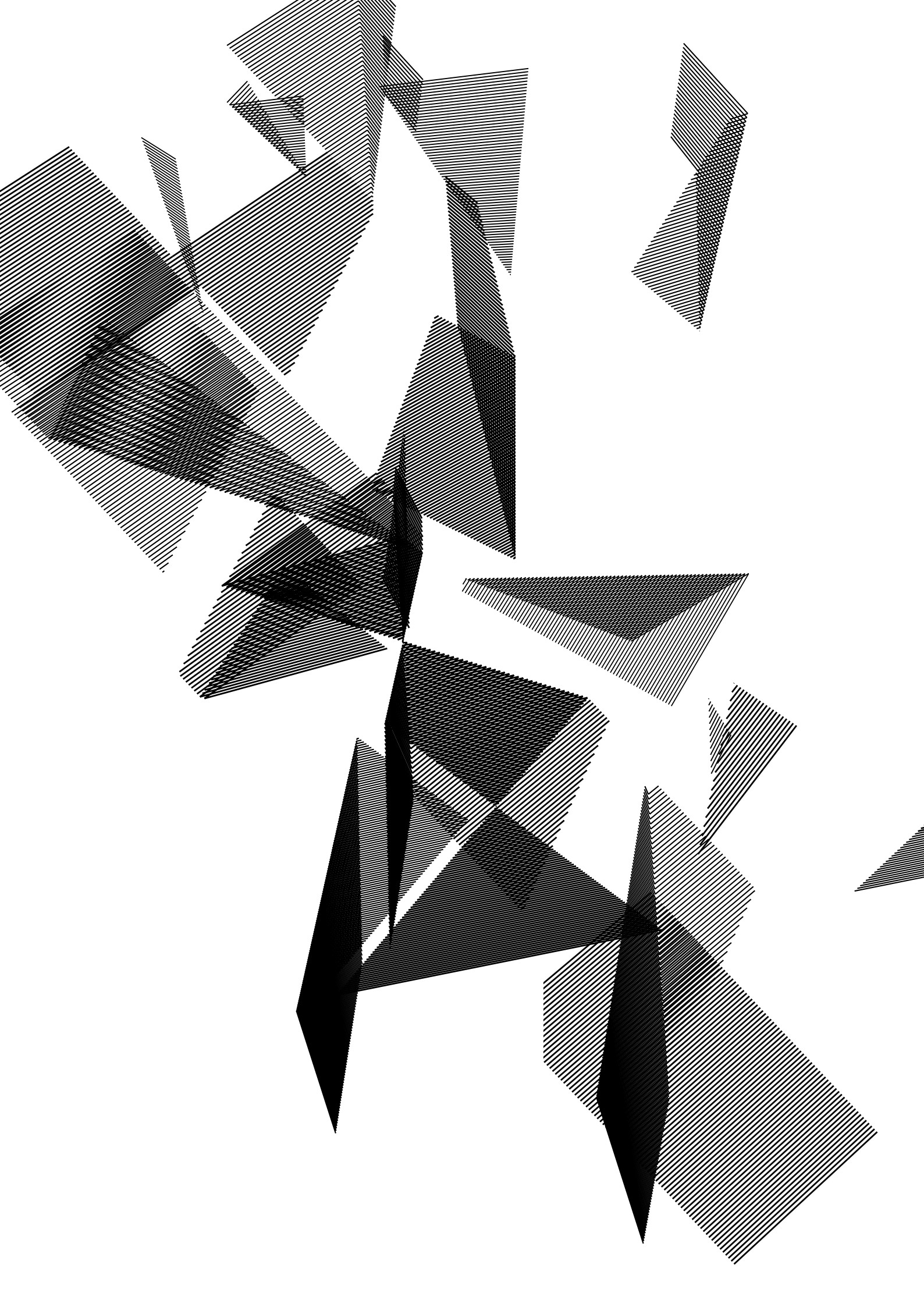
REFERÊNCIAS BIBLIOGRÁFICAS

FORBELLONE, A. L. V.; EBERSPACHER, H. F. *Lógica de programação: a construção de algoritmos e estruturas de dados*. 3.ed. São Paulo: Prentice Hall, 2005.

PUGA, S.; RISSETTI, G. *Lógica de programação e estruturas de dados, com aplicação em java*. São Paulo: Pearson Prentice, 2009.

RoboMind *RoboMind.net – Documentation Overview*. 2011. Disponível em http://www.robomind.net/en/docOverview.htm.

Scilab *Scilab Tutorials*. 2011. Disponível em http://www.scilab.org/support/documentation/tutorials.



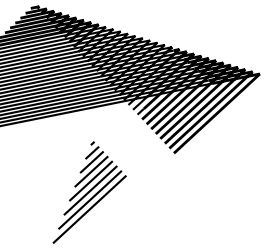
CAPÍTULO 6

LÓGICA DE PROGRAMAÇÃO: ESTRUTURAS CONDICIONAIS

Harlen Costa Batagelo
João Paulo Gois
Letícia Rodrigues Bueno
Luiz Carlos da Silva Rozante
Ronaldo Cristiano Prati

6.1 INTRODUÇÃO

Em um computador, o conjunto de instruções pode compreender desde as operações aritméticas que encontramos em uma calculadora simples, até instruções utilizadas para tomar decisões, repetir a execução de sequências de outras instruções, ler dados a partir do teclado, gravar esses dados em um dispositivo de armazenamento não volátil (como o disco rígido do computador ou um *pendrive*), enviá-los pela rede ou exibi-los no monitor, produzir sons, entre muitas outras. Aprender a programar não é uma tarefa trivial, pois envolve manipular uma representação abstrata do mundo real, criada dentro do computador; aprender as questões relativas à linguagem que está sendo usada para criar os programas e a lógica de programação em si. No capítulo anterior, vimos os primeiros conceitos de lógica de programação: instrução e sequência de instruções. *O objetivo deste capítulo é* apresentar um conceito mais elaborado de programação: as estruturas condicionais. Essas estruturas fornecem uma maneira de instruirmos os computadores na tomada de algumas decisões, aumentando assim o poder computacional dos programas que vimos até agora. *O objetivo deste capítulo NÃO é* estudar o assunto em profundidade, assim não serão discutidas estruturas condicionais muito elaboradas, nem tampouco questões como desempenho e melhores práticas de programação.



O restante deste capítulo está distribuído conforme a seguinte organização. A Seção 6.2 faz uma analogia entre as decisões que tomamos diariamente e as decisões que podemos instruir o computador a fazer. Como no capítulo anterior, os conceitos são ilustrados através do RoboMind, sendo reforçados pelas atividades em aula na Seção 6.3 e por exemplos com o Scilab na Seção 6.4. A Seção 6.5 apresenta as considerações finais do capítulo e exercícios complementares são fornecidos na Seção 6.6.

6.2 TOMANDO DECISÕES: ESTRUTURAS CONDICIONAIS

O nosso cotidiano é repleto de tomadas de decisões. Você deve decidir entre ir à universidade de carro ou de bicicleta, ou entre estudar para a disciplina de Bases Computacionais da Ciência ou ir à festa. Para cada decisão, há consequências específicas.

Você pode estabelecer as tomadas de decisões com base em regras. Por exemplo, você está jogando "par-ou-ímpar" com seu amigo. Se a soma dos dedos apresentados por você e seu amigo for par, você ganha, caso contrário, seu amigo ganha. Um outro exemplo de tomada de decisões baseada em regras pode ser aplicado na definição do seu conceito final da disciplina de Bases Computacionais da Ciência. Seja `x` a porcentagem final de aproveitamento que você conseguiu no curso:

- Se $x<45\%$, seu conceito é F;
- Caso contrário, se $x \geq 45\%$ e $x<50\%$, seu conceito é D;
- Caso contrário, se $x \geq 50\%$ e $x<70\%$, seu conceito é C;
- Caso contrário, se $x \geq 70\%$ e $x<85\%$, seu conceito é B;
- Caso contrário, seu conceito é A.

Ilustrando estruturas condicionais com RoboMind

O RoboMind tem comandos tanto para tomada de decisões quanto para observar o ambiente, assim os seus programas podem tomar decisões baseando-se no estado do ambiente. O robô pode observar as células da esquerda, da direita e à frente. Essas observações podem ser de cinco tipos diferentes:

- existe um obstáculo na célula?
- a célula está vazia?
- tem um objeto na célula?
- a célula está pintada de branco?
- a célula está pintada de preto?

Um resumo dos comandos para testar essas observações está na Tabela 6.1.

Tabela 6.1: Comandos de observação do ambiente.

Esquerda	Frente	Direita
`temObstáculoEsquerda()`	`temObstáculoFrente()`	`temObstáculoDireita()`
`vazioEsquerda()`	`vazioFrente()`	`vazioDireita()`
`temObjetoEsquerda()`	`temObjetoFrente()`	`temObjetoDireita()`
`brancoEsquerda()`	`brancoFrente()`	`brancoDireita()`
`pretoEsquerda()`	`pretoFrente()`	`pretoDireita()`

Esses comandos de observação do ambiente podem ser usados em combinação com estruturas de decisão para que o robô tome decisões baseando-se no estado do mundo. Uma dessas estruturas é o comando `se (condição) {comandos}`, que executa a sequência de comandos somente se a condição for verdadeira. O programa da Listagem 6.1, por exemplo, observa a célula da esquerda e, se ela estiver pintada de branco, repinta a célula de preto, voltando à sua posição.

Listagem 6.1: Primeiro programa com estrutura condicional.

```
1  se( brancoEsquerda ()){
2     virarEsquerda ()
3     andarFrente (1)
4     pintarPreto ()
5     pararPintar ()
6     andarTrás (1)
7     virarDireita ()
8  }
```

Se você tentar executar o programa da Listagem 6.1, e a célula da esquerda não estiver pintada de branco, nada acontece pois o bloco de código será executado apenas se aquela célula estiver pintada de branco. Você pode usar uma variação dessa estrutura de controle para instruir o robô a realizar uma atividade alternativa, caso a condição não seja verdadeira: `se (condição) {comandos} senão {comandos}`. Compare o programa da Listagem 6.1 com o programa da Listagem 6.2.

Listagem 6.2: Segundo programa com estrutura condicional.

```
1  se( brancoEsquerda ()){
2     virarEsquerda ()
3     andarFrente (1)
4     pintarPreto ()
5     pararPintar ()
6     andarTrás (1)
7     virarDireita ()
8  }
9  senão{
10    andarFrente(3)
11 }
```

Nesse caso, caso a célula esteja pintada de branco, o robô irá pintá-la de preto e voltar à sua posição anterior. Caso contrário, o robô irá andar três posições para a frente e parar. Observe que apenas um dos blocos de comandos é executado. Se a célula da direita estiver pintada de branco, o robô pinta de preto e não irá avançar as três casas. Caso contrário, ele irá avançar as três casas, mas não irá pintar a célula da esquerda de preto.

Expressões Lógicas

A condição no comando `se` é, na verdade, uma expressão lógica que pode assumir os valores `verdadeiro` ou `falso`, dependendo do estado do cenário. Uma expressão lógica pode ser uma ação de percepção, como aquelas listadas na Tabela 6.1, ou o resultado da instrução `sortear()`, que retorna um valor verdadeiro ou falso, de maneira aleatória.

Expressões lógicas também podem envolver combinação de expressões simples através do uso de conectivos lógicos. No RoboMind dispomos de tês conectivos lógicos: `não (~)`, e `(&)` e ou `(|)`. O conectivo "`não`" modifica o valor verdade da expressão lógica posterior a ele. Por exemplo, se `brancoFrente()` é verdadeiro, `não brancoFrente()` será falso, e vice-versa. O conectivo "`e`" compara duas expressões lógicas e o resultado é verdadeiro apenas se as duas forem verdadeiras. Para qualquer outra combinação, o resultado da expressão é falso. Já o conectivo "`ou`"

compara duas expressões e o resultado da expressão é verdadeiro se qualquer uma das duas, ou ambas, forem verdadeiras. Somente a combinação `falso ou falso` tem valor verdade falso. Para referência, um resumo dos conectivos lógicos e seu significado é mostrado na Tabela 6.2.

Tabela 6.2: Conectores lógicos e a sua interpretação.

Conector	Funcionamento
`Não`	`não verdadeiro = falso` `não falso = verdadeiro`

`E`	`verdadeiro e verdadeiro = verdadeiro`
	`verdadeiro e falso = falso`
	`falso e verdadeiro = falso`
	`falso e falso = falso`

`Ou`	`verdadeiro ou verdadeiro = verdadeiro`
	`verdadeiro ou falso = verdadeiro`
	`falso ou verdadeiro = verdadeiro`
	`falso ou falso = falso`

É possível combinar as expressões lógicas usando vários conectivos, mas a ordem em que os conectivos aparecem é importante. O conectivo "`não`" é o que tem maior prioridade, seguido pelo "`e`" e, por fim, o "`ou`". Parênteses podem ser usados para mudar a prioridade, caso necessário. O programa da Listagem 6.3 apresenta alguns exemplos de expressões lógicas.

Listagem 6.3: Programa com exemplos de expressões lógicas.

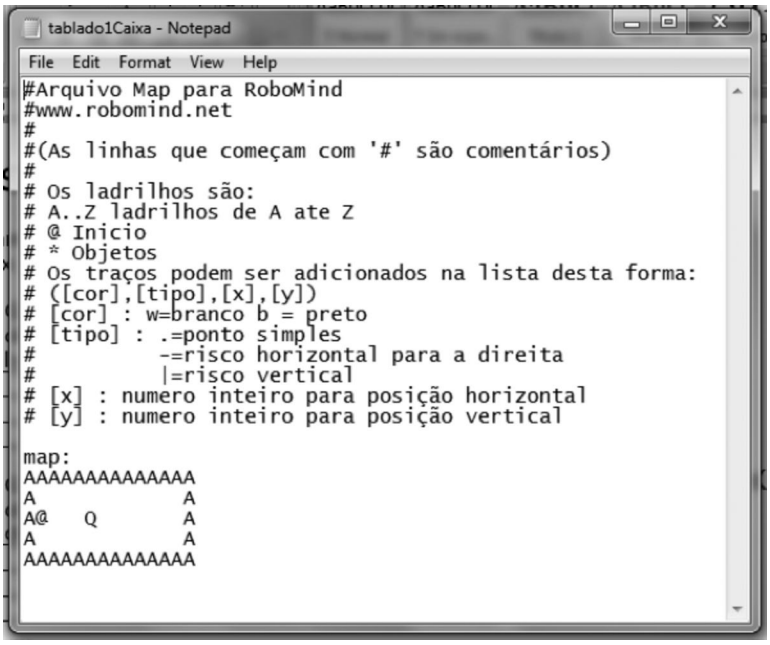
```
1  se( sortear () e não brancoDireita ())
2  {
3     virarDireita ()
4     andarTrás (1)
5  }
6
7  se( verdadeiro e falso ){
8     # essa instrução nunca é executada
9     andarFrente (1)
10 }
```

6.3 ATIVIDADES EM AULA

Exercício resolvido

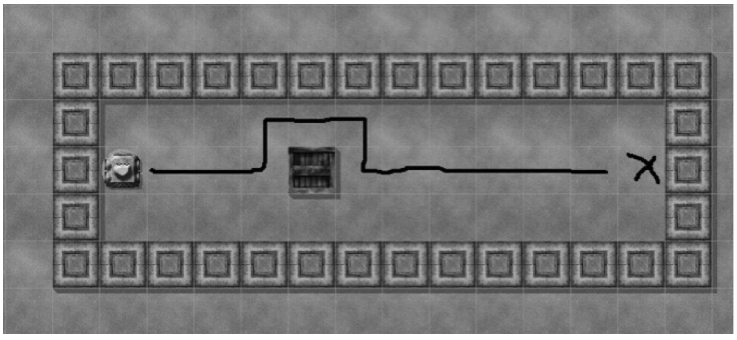
Neste exercício iremos fazer com que o robô tome decisões para contornar os obstáculos à sua frente. Vamos criar um cenário em que o robô deve se mover em linha reta, da esquerda para direita, mas com um obstáculo à frente. O arquivo para a criação do mapa para esse cenário pode ser visto na Figura 6.1.

Figura 6.1: Descrição do mapa com uma caixa.



O caminho a ser percorrido pelo robô para esse mapa pode ser visto na Figura 6.2. A princípio poderíamos usar uma estratégia parecida com a do capítulo anterior, e programar o caminho "na mão" passando pelos pontos que contornam a caixa. Entretanto, essa abordagem não é muito flexível, pois, se mudarmos a caixa de posição, o programa perde a utilidade.

Figura 6.2: Cenário com uma caixa.



O programa da Listagem 6.4 faz com que o robô tente atravessar o cenário da esquerda para a direita. Não se preocupe por enquanto com o comando `repetir(11)`. Ele será explicado e explorado melhor na próxima aula, mas o seu propósito é repetir o comando `andarLeste(1)` onze vezes. Esse programa funcionaria se não houvesse a caixa no cenário, mas, se você executar esse programa nesse cenário, o seu programa ficará parado em frente à caixa.

Listagem 6.4: Programa para o robô andar da esquerda para a direita.

```
1  repetir (11) {
2     andarLeste (1)
3  }
```

Vamos alterá-lo para contornar a caixa, conforme ilustrado no código da Listagem 6.5.

Listagem 6.5: Programa para o robô contornar uma caixa.

```
1  repetir (10){
2     se( temObstáculoFrente () ){
3        andarNorte (1)
4        andarLeste (2)
5        andarSul (1)
6     } senão {
7        andarLeste (1)
8     }
9  }
```

No programa da Listagem 6.5, a lógica do robô é de dar um passo de cada vez e, se houver um obstáculo à frente, ele contorna a caixa, dando um passo para o norte, dois para o leste e um para o sul. Observe que o número de vezes em que a sequência é repetida foi diminuído para dez, pois, para poder contornar a caixa, o robô dá dois passos para frente (`andarLeste(2)`) depois de dar um passo para o norte e antes de dar um passo para o sul, em vez de dar um único passo para a frente. Caso a caixa não estiver ali, para percorrer todo o percurso, a instrução `andarLeste(1)` seria executada duas vezes, mas, como em (`andarLeste(2)`) é dado um passo para o leste a mais do que (`andarLeste(1)`), podemos diminuir o número de repetições para dez, descontando o passo a mais da instrução `andarLeste(2)` executada ao contornar a caixa em vez de `andarLeste(1)`. Esse programa funciona para qualquer lugar em que você colocar a caixa no caminho do robô (experimente mudar a posição da caixa no cenário para se certificar disso), uma vez que a posição da caixa não está codificada diretamente no código, mas é "encontrada" testando-se se há um obstáculo à frente.

A mesma estrutura do programa funciona ainda que existam duas (ou mais) caixas separadas no cenário. Vamos testá-lo agora num cenário com duas caixas, que pode ser construído usando a descrição do mapa mostrada na Figura 6.3.

Figura 6.3: Descrição do mapa com duas caixas.

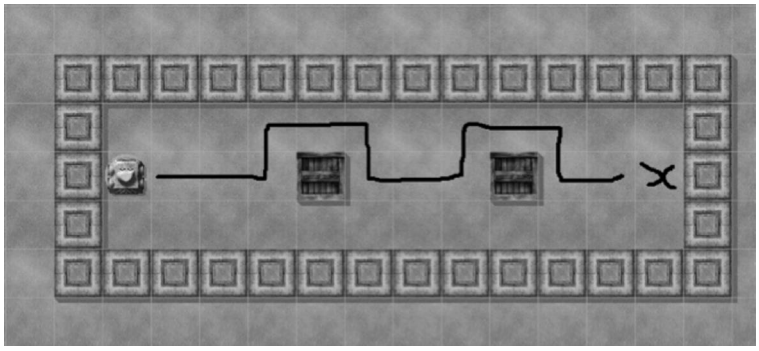
Para percorrer esse cenário você pode usar o programa da Listagem 6.6.

Listagem 6.6: Programa para o robô contornar duas caixas.

```
1  repetir (9){
2     se( temObstáculoFrente () ){
3        andarNorte (1)
4        andarLeste (2)
5        andarSul (1)
6     } senão {
7        andarLeste (1)
8     }
9  }
```

Observe que a estrutura é a mesma, exceto pela única diferença de que, como temos duas caixas, o número de repetições da sequência de comandos deve ser apenas nove, e não dez, pois ao contornar as duas caixas a instrução `andarLeste (2)` é executada duas vezes, uma para cada caixa. Ao executar esse programa, o robô percorrerá o percurso mostrado na Figura 6.4. Novamente, não importa a posição das caixas no caminho do robô (desde que elas não estejam juntas), pois o robô irá verificar a posição delas testando se há um obstáculo à frente.

Figura 6.4: Cenário com duas caixas.



6.4 ILUSTRANDO OS CONCEITOS APRENDIDOS USANDO SCILAB

Retomando o problema exemplificado em Scilab no capítulo anterior, lembre-se de que o problema consiste em calcular a área `A`, o perímetro `P` e a diagonal `D` de um quadrado de lado `x` unidades de medida (u.m.), onde `x` é um valor de entrada fornecido pelo usuário. E se o usuário entrar com um número menor ou igual a zero? O programa funcionará normalmente, mas os resultados não farão sentido para este problema. O modo de resolver isso é fazendo um teste antes de realizar os cálculos, ou seja, verificamos se o comprimento é positivo (se `x>0`) utilizando uma estrutura condicional. Alteramos nosso programa, obtendo o código da Listagem 6.7.

Listagem 6.7: Programa com um comando condicional, para identificar números negativos.

```
1  x = input (" Entre com o comprimento do lado do quadrado:");
2  if x > 0
3     A = x*x;
4     P = 4*x;
5     D = sqrt (2)*x;
6     printf ("A área é %f\nO perímetro é %f\nA diagonal é
              %f",A,P,D);
7  end
```

Se, por exemplo, `x=5`, então teremos a solução já conhecida do capítulo anterior. Porém, se entrarmos com 0, –1 ou qualquer outro número negativo, nenhuma saída é apresentada. As novas instruções utilizadas na Listagem 6.7 estão nas linhas 2 e 7. Na Linha 2 (`if x > 0`), a instrução `if` (do inglês, "se") é utilizada para que o programa tome uma decisão sobre a condição `x>0`. Isto é, se `x>0` for verdadeiro, o programa executará todos os comandos seguintes até encontrar a instrução `end` (Linha 7). Observe que acrescentamos espaços no início das Linhas 3 a 6. Fizemos isto para facilitar a leitura do código-fonte, uma vez que estas instruções serão executadas apenas quando `x>0`. Para esta boa prática de programação damos o nome de *identação*. Estes espaços não interferem no resultado final do programa e facilitam a identificação do bloco de comandos que será executado caso a condição seja verdadeira. Experimente acrescentar a seguinte instrução abaixo da Linha 7, na Listagem 6.7:

```
printf("\nFim do programa");
```

Execute novamente com os valores `x=5` e `x=-1` e veja o que acontece. Seria desejável que o programa informasse ao usuário que uma entrada $x \leq 0$ não é válida, em vez de simplesmente não apresentar nenhuma saída. Na Listagem 6.8 está a implementação de tal comportamento.

Listagem 6.8: Programa que apresenta uma mensagem ao usuário.

```
1  x = input (" Entre com o comprimento do lado do quadrado:");
2  if x > 0
3     A = x*x;
4     P = 4*x;
5     D = sqrt (2)*x;
6     printf ("A área é %f\nO perímetro é %f\nA diagonal é
            %f",A,P,D);
7  else
8     printf (" Entrada inválida \n");
9  end
```

Na Linha 7 do código da Listagem 6.8 acrescentamos a instrução `else` (do inglês, "senão"). Se `x`>0, todas as instruções contidas entre as instruções `if-else` serão executadas. Senão ($x \leq 0$), todas as instruções contidas entre `else-end` serão executadas. Agora, para qualquer valor de `x`, o programa apresentará algum tipo de mensagem ao usuário.

No exemplo da Listagem 6.8, a comparação foi realizada com o símbolo de maior. De fato, podemos realizar comparações com qualquer um dos símbolos matemáticos utilizados para comparação de valores. Na Tabela 6.3 apresentamos as sintaxes para as operações de comparação entre números no Scilab.

Tabela 6.3: Sintaxe para as operações de comparação entre números no Scilab.

igual	==
diferente	~=
menor	<
maior	>
menor-ou-igual	<=
maior-ou-igual	>=

O Scilab também possui conectores lógicos. Diferentemente do RoboMind que utiliza palavras da Língua Portuguesa para definir os conectores lógicos, no Scilab são utilizados símbolos. A Tabela 6.4 apresenta uma comparação entre os conectores utilizados pelo RoboMind e os correspondentes no Scilab.

Tabela 6.4: Comparação entre conectores lógicos do RoboMind e do Scilab.

RoboMind	**Scilab**
não	~
e	&
ou	|

As listagens 6.9 e 6.10 mostram programas que recebem como entrada um número natural, armazenam na variável `a` e verificam se este valor está contido no intervalo [5,10].

Listagem 6.9: Programa com um teste condicional "`ou`".

```
1  a = input (" Digite um número natural : ");
2  if (a < 5 | a > 10)
3     printf ("O número não está no intervalo [5 ,10]");
4  else
5     printf ("O número está no intervalo [5 ,10]");
6  end
```

Listagem 6.10: Programa com um teste condicional "`e`".

```
1  a = input (" Digite um número natural : ");
2  if (a >= 5 & a <= 10)
3     printf ("O número está no intervalo [5 ,10]");
4  else
5     printf ("O número não está no intervalo [5 ,10]");
6  end
```

6.5 CONSIDERAÇÕES FINAIS

Este capítulo apresentou uma estrutura que concede maior poder computacional ao determinar como instruir os computadores na tomada de decisões. Como indicamos no início do capítulo, o objetivo não é explorar todo o poder computacional destas estruturas, nem tampouco discutir detalhes de implementação ou melhores práticas de programação, mas apresentar conceitos básicos para a resolução de problemas simples através da tomada automatizada de decisões.

No próximo capítulo, apresentamos, finalmente, uma última estrutura fundamental que concede ainda mais recursos para nossos programas, que é a capacidade de repetir um conjunto de instruções: as estruturas de repetição. Estas estruturas, juntamente com os conceitos apresentados no capítulo anterior e neste capítulo, consistem na base na qual a lógica de programação subsiste. A partir dessa base, outros recursos e tecnologias avançadas – discutidos em disciplinas de cursos da área de Computação – são utilizados em todo o mundo para desenvolver programas complexos e vitais a toda hora.

6.6 EXERCÍCIOS

6.6.1 EXERCÍCIOS COM O ROBOMIND

Agora tente resolver os exercícios a seguir:

1. Acesse o RoboMind e teste os programas vistos na Seção 6.2;

2. Faça um programa para que o robô, no ambiente dado pelo mapa `openArea.map`, escreva de preto seu nome ou sobrenome. A escolha se será o nome ou sobrenome deve ser aleatória;

3. Faça um programa para que o robô, no ambiente dado pelo mapa `openArea.map`, pinte de branco um quadrado do seguinte modo: o robô anda para frente três posições e escolhe aleatoriamente uma direção (esquerda ou direita), depois conclui a pintura segundo esta escolha;

4. Faça um programa para que o robô pinte um quadrado de modo similar ao exercício anterior, mas agora escolhendo as cores dos lados de modo aleatório. Use o mesmo ambiente dado pelo mapa `openArea.map`;

5. Faça um programa para que o robô, no ambiente dado pelo mapa `copyLine1.map`, escolha aleatoriamente uma de duas direções (para frente ou para trás) e:

- se escolheu ir para frente, pinte de preto o restante da linha já existente no mapa até o limite definido pelo "muro" ao norte;
- se escolheu ir para trás, pinte de preto o restante da linha já existente no mapa até o limite definido pelo "muro" ao sul.

6. Faça um programa que permita o robô escolher aleatoriamente uma das quatro direções possíveis: direita, esquerda, frente e trás;

7. Faça um programa para que o robô, no ambiente dado pelo mapa `findSpot1.map`, encontre o vaso com planta ou vaso com água;

8. Escreva um programa para que o robô desenhe um "alvo", isto é, pinte "círculos" (aqui na verdade quadrados) concêntricos em preto e branco alternadamente. Faça de tal modo que a cor da "mosca" (centro) seja escolhida aleatoriamente e que haja seis quadrados à sua volta;

9. Escreva um programa para que o robô pinte o seu primeiro nome no chão, sendo que as cores de cada letra devem ser escolhidas aleatoriamente;

10. Similarmente ao exercício anterior, escreva um programa para que o robô pinte o seu primeiro nome no chão, sendo que as cores de cada letra devem ser escolhidas aleatoriamente; entretanto, agora com uma tarefa adicional: se a última letra for preta, o robô deve pintar também o sobrenome (com qualquer cor);

11. Faça um programa para que o robô, no ambiente dado pelo mapa `copyLine1.map`, faça uma das seguintes tarefas:

- pinte de branco a faixa já pintada de preto presente no mapa;
- pegue o objeto (baliza) que está mais distante do robô;
- pegue o objeto que está mais próximo do robô;
- pegue o objeto que não é o mais distante nem o mais próximo do robô.

12. Faça um programa para que o robô, no ambiente dado pelo mapa `copyLine1.map`, escolha aleatoriamente um dos três objetos presentes no mapa e o coloque na posição inicial do robô;

13. De forma parecida com o exercício anterior, faça um programa para que o robô, também no ambiente dado pelo mapa `copyLine1.map`, escolha aleatoriamente um dos três objetos presentes no mapa e o coloque na posição inicial do robô. Adicionalmente são tarefas do robô:

- se o objeto escolhido foi o mais próximo, a faixa preta no mapa deverá ser pintada de branco;
- se o objeto escolhido foi o mais distante, a faixa preta no mapa deverá ser duplicada (ele deve pintar outra faixa preta paralela à já existente);
- se o objeto escolhido não foi nem o mais distante nem o mais próximo, então nada será feito.

14. Faça um programa para que o robô, no ambiente dado pelo mapa `copyLine1.map`, escolha aleatoriamente um dos três objetos presentes no mapa e agrupe os demais objetos em torno dele;

15. Faça os exercícios indicados na Seção 6.6.2.

6.6.2 EXERCÍCIOS COM O SCILAB

1. Acesse o Scilab e teste os exemplos vistos na Seção 6.4;

2. Faça um programa que leia do teclado dois números `x` e `y` e atribua o menor desses valores em `x` e o maior em `y`, mostrando o resultado na saída;

3. Escreva um programa que leia do teclado um número natural `x` e verifique se `x` é par ou ímpar. Para este exercício você terá que utilizar o comando do Scilab `modulo(x,y)` que retorna o resto da divisão de `x` por `y`;

4. Faça um programa em Scilab que leia três números `a`, `b` e `c` e verifique se o polinômio ax^2+bx+c possui uma raiz real, duas raízes reais distintas ou raízes complexas. Verifique também se a parábola definida por este polinômio possui concavidade para cima ou para baixo;

5. Escreva um programa que leia três valores, armazenando-os nas variáveis `x`, `y` e `z`, e ordene esses valores de modo que, ao final, o menor valor esteja armazenado na variável `x`, o valor intermediário esteja armazenado na variável `y` e o maior valor esteja armazenado na variável `z`;

6. Escreva um programa que leia do teclado o seu percentual `x` de aproveitamento de uma disciplina da UFABC e calcule seu conceito final usando os seguintes critérios:

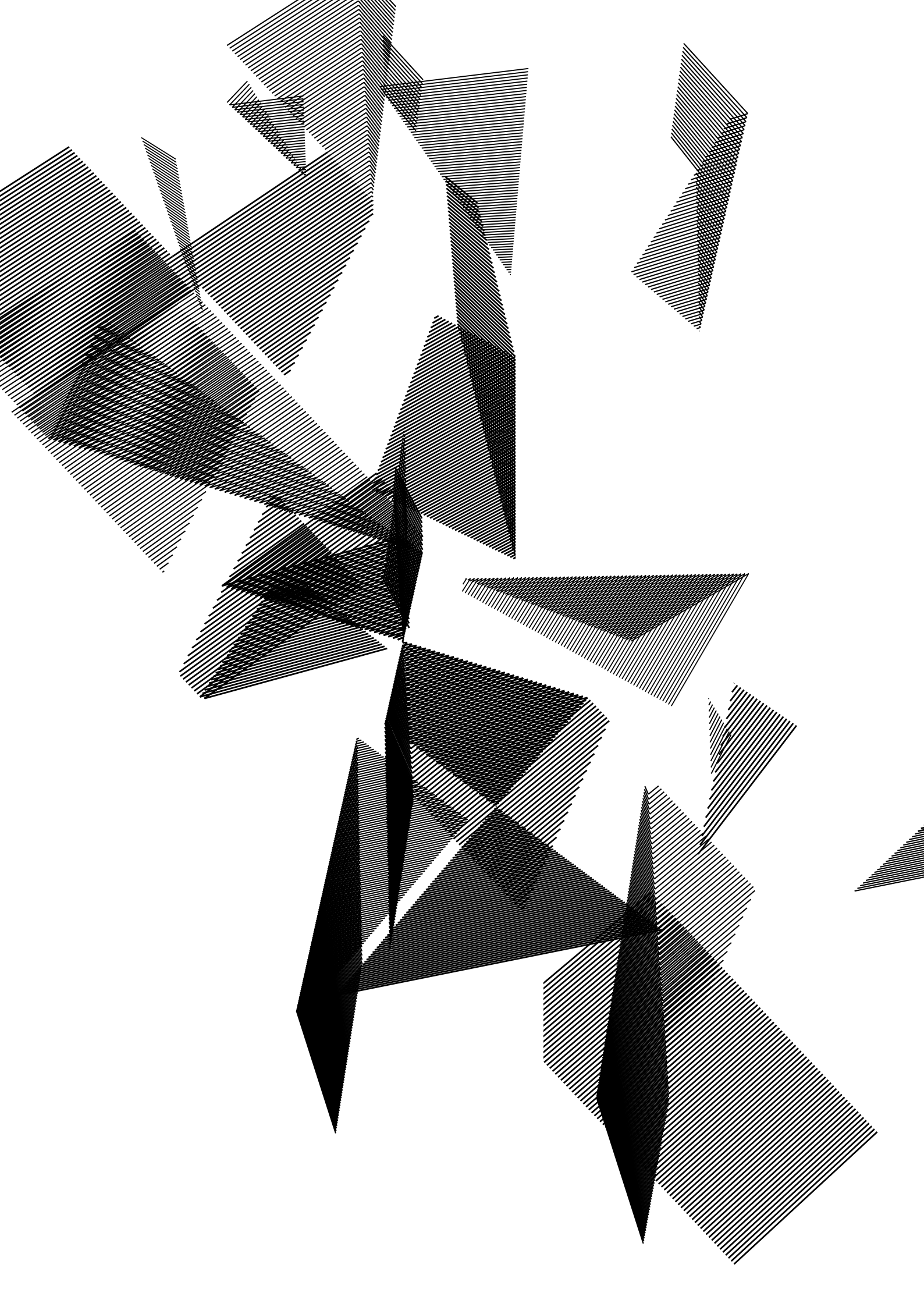
- Se $x < 45\%$, seu conceito é F;
- Caso contrário, se $x \geq 45\%$ e $x < 50\%$, seu conceito é D;
- Caso contrário, se $x \geq 50\%$ e $x < 70\%$, seu conceito é C;
- Caso contrário, se $x \geq 70\%$ e $x < 85\%$, seu conceito é B;
- Caso contrário, seu conceito é A.

O programa também deve mostrar na saída se você foi aprovado, ou reprovado, e com qual conceito;

7. Faça um programa que leia três valores. Esses valores representarão o comprimento dos lados de um triângulo. Verifique se esses valores formam um triângulo e classifique-o em:

- equilátero: três lados iguais;
- isósceles: dois lados iguais;
- escaleno: três lados diferentes.

Lembre-se de que, para formar um triângulo, nenhum dos lados pode ter comprimento menor ou igual a zero e que cada lado deve ser menor do que a soma dos outros dois lados.



CAPÍTULO 7

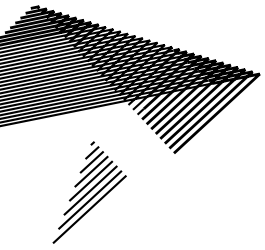
LÓGICA DE PROGRAMAÇÃO: ESTRUTURAS DE REPETIÇÃO

Harlen Costa Batagelo
João Paulo Gois
Letícia Rodrigues Bueno
Luiz Carlos da Silva Rozante
Ronaldo Cristiano Prati

7.1 INTRODUÇÃO

No Capítulo 5 estudamos os primeiros conceitos de lógica de programação: instrução e sequência de instruções. No capítulo anterior, vimos as estruturas condicionais, uma maneira de instruir o computador na tomada de decisões. *O objetivo deste capítulo é* apresentar um conceito mais elaborado de programação: as estruturas de repetição. Essas estruturas fornecem uma maneira de instruir os computadores na repetição de um conjunto de instruções, o que nos fornece mais um recurso computacional importante. *O objetivo deste capítulo NÃO é* esgotar o assunto, nem tampouco discutir estruturas de repetição muito elaboradas, diferentes formas de implementação, questões de desempenho ou melhores práticas de programação.

O restante deste capítulo está distribuído conforme a seguinte organização. A Seção 7.2 faz uma analogia entre as ações que necessitamos repetir em nossa vida diária e as ações que podemos instruir o computador a repetir. A Seção 7.3 destaca rapidamente a importância dos conceitos vistos no desenvolvimento de programas complexos. Como nos dois capítulos anteriores, os conceitos são ilustrados através do RoboMind, sendo reforçados pelas atividades em aula na Seção 7.4 e por exemplos com o Scilab na Seção 7.5. A Seção 7.6 apresenta as considerações finais do capítulo e exercícios complementares são fornecidos na Seção 7.7.



7.2 REPETINDO AS MESMAS INSTRUÇÕES

Em diversas situações, desejamos realizar uma determinada sequência de ações um número repetido de vezes. Por exemplo, você pode querer usar o despertador para acordar todo dia no mesmo horário. Se você usa o despertador do celular, já deve ter notado que ele tem a opção de tocar uma única vez ou repetir o toque diariamente. A opção de repetir o toque diariamente foi adicionada para que você não precise ficar programando todo dia o despertador do celular.

Suponha que desejamos calcular a expressão 2^{10}. Sabemos que:

$$2^{10} = 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 = 1024$$

A expressão acima contém nove vezes a instrução de multiplicação. Usando uma calculadora, você teria que repetir a multiplicação nove vezes. Agora, se quisermos calcular 2^{100}? Por favor, não faça isto usando a técnica acima. Assim como celulares tem a opção de repetir o toque diariamente, linguagens de programação fornecem mecanismos para facilitar a execução de tarefas repetidas. Mostramos agora como um computador pode nos auxiliar a realizar facilmente este cálculo através de instruções específicas para repetir outras instruções. Em particular, podemos passar ao computador a instrução 2×2 e instruí-lo a usar o resultado desta operação para novamente multiplicar por dois, e assim sucessivamente até que esta operação seja repetida 99 vezes.

Ilustrando estruturas de repetição com RoboMind

Vamos ilustrar a ideia usando RoboMind. No programa que desenha um quadrado no Capítulo 5, as instruções `andarFrente(n)` e `virarEsquerda()` são repetidas quatro vezes seguidas. A linguagem utilizada para criar programas para o RoboMind fornece uma estrutura de controle que pode ser usada para situações desse tipo, de maneira que você não precisa repetir as instruções para executar ações repetidas que seguem um padrão. Essa estrutura de controle é o `repetir(n) {comandos}` e é usada para repetir os comandos entre parênteses `n` vezes, no qual `n` é um parâmetro que controla o número de vezes que a sequência de comandos deve ser repetida. Por exemplo, para percorrer o quadrado, você poderia usar o programa da Listagem 7.1.

Listagem 7.1: Programa para percorrer um quadrado, com um comando de repetição.

```
1  repetir (4){
2     andarFrente (1)
3     virarEsquerda ()
4  }
```

Ao executar esse programa, os comandos que estão entre chaves serão executados quatro vezes, o que é equivalente aos programas para se percorrer um quadrado que fizemos anteriormente nos Capítulos 5 e 6. Você pode colocar qualquer sequência de comandos entre as chaves, incluindo uma outra estrutura de repetição. O programa da Listagem 7.2 percorre o quadrado duas vezes, usando duas estruturas de repetição, uma dentro da outra.

Listagem 7.2: Programa para percorrer um quadrado, com dois comandos de repetição.

```
1  repetir (2){
2     repetir (4){
3        andarFrente (1)
4        virarEsquerda ()
5     }
6  }
```

A primeira estrutura de repetição controla o número de vezes que o percurso do quadrado será feito, e a segunda estrutura controla a repetição de comandos necessária para percorrer o quadrado. Observe que a sequência `andarFrente(1)` e `virarEsquerda()` será executada oito vezes. Se você quiser deixar o robô percorrendo esse quadrado indefinidamente, você pode usar o comando `repetir()` sem argumentos. O programa da Listagem 7.3, por exemplo, faz com que o robô fique percorrendo o quadrado até que a execução do programa seja abortado.

Listagem 7.3: Programa para fazer um quadrado, com o comando `repetir()` sem argumento.

```
1  repetir (){
2     repetir (4){
3        andarFrente (1)
4        virarEsquerda ()
5     }
6  }
```

O RoboMind também tem uma estrutura condicional que envolve a repetição dos comandos enquanto a condição for verdadeira. Essa estrutura é `repetirEnquanto(condição) {comandos}`. Observe a diferença entre as estruturas `repetirEnquanto` e `repetir`. Na estrutura `repetir`, a sequência de comandos é repetida um número de vezes `n` pré-definido, independentemente do estado do ambiente. Na estrutura `repetirEnquanto`, o robô executa a ação enquanto sua percepção do ambiente, avaliada pela condição, for verdadeira. O programa da Listagem 7.4 faz com que o robô siga uma linha em branco pintada no chão.

Listagem 7.4: Programa para seguir uma linha em branco, usando o comando `repetir()`.

```
1  repetirEnquanto ( brancoFrente ()){
2     andarFrente (1)
3  }
```

7.3 CRIANDO PROGRAMAS COMPLEXOS

Computadores são usados para resolver diferentes problemas, muitos deles extremamente complexos como manter atualizado o saldo bancário de centenas de milhares de correntistas de grandes bancos, que envolve o processamento de milhões de transações diárias; controlar equipamentos complexos como os diversos componentes eletrônicos de um avião, que contêm centenas de sensores e atuadores para que o avião funcione corretamente; e realizar simulações computacionais de modelos matemáticos do universo, que pode gerar quantidades enormes de dados a respeito dessas simulações, dentre muitos outros.

Apesar da complexidade, na grande maioria dos casos, esses sistemas são criados usando sequências de comandos, repetições e desvios de fluxo devido à tomada de decisão, combinados de maneira apropriada.

7.4 ATIVIDADES EM AULA

1. Acesse o RoboMind e teste os programas vistos na Seção 7.2;

2. Faça um programa para que o robô, no ambiente dado pelo mapa `default.map`, se desloque, a partir da posição inicial, indo para leste, até encontrar a primeira posição pintada de branco. A partir daí o robô deverá rastrear a linha branca pintada no chão, isto é, segui-la até encontrar o muro ao norte;

3. Faça um programa para que o robô, no ambiente dado pelo mapa `default.map`, a partir da posição inicial faça: escolha uma direção aleatória, siga na direção escolhida e pare somente quando encontrar algum obstáculo;

4. Faça um programa para que o robô, no ambiente dado pelo mapa `passBeacons.map`, saia da posição inicial (salão à esquerda) e vá para o ponto marcado em branco no salão à direita.

7.5 ILUSTRANDO OS CONCEITOS APRENDIDOS USANDO SCILAB

Retomando o problema exemplificado em Scilab nos Capítulos 5 e 6, que consiste em calcular a área `A`, o perímetro `P` e a diagonal `D` de um quadrado de lado `x` unidades de medida (u.m.), onde `x` é um valor de entrada fornecido pelo usuário. Suponha, agora, que queiramos realizar várias vezes os cálculos de diferentes quadrados. Para isto, utilizamos a estrutura de repetição `while`, conforme o código da Listagem 7.5.

Listagem 7.5: Programa utilizando o comando `while`.

```
1  x = 0;
2  while x ~= -1
3     x=input ("Entre com o comprimento do lado do quadrado:")
4     if x > 0
5        A = x*x;
6        P = 4*x;
7        D = sqrt (2)*x;
8        printf ("A área é %f\nO perímetro é %f\nA diagonal é
                 %f",A,P,D);
9     else
10       printf (" Entrada inválida \n");
11       printf ("Não foi possível realizar os cálculos \n");
12    end
13 end
14 printf ("\ nFim do programa ");
```

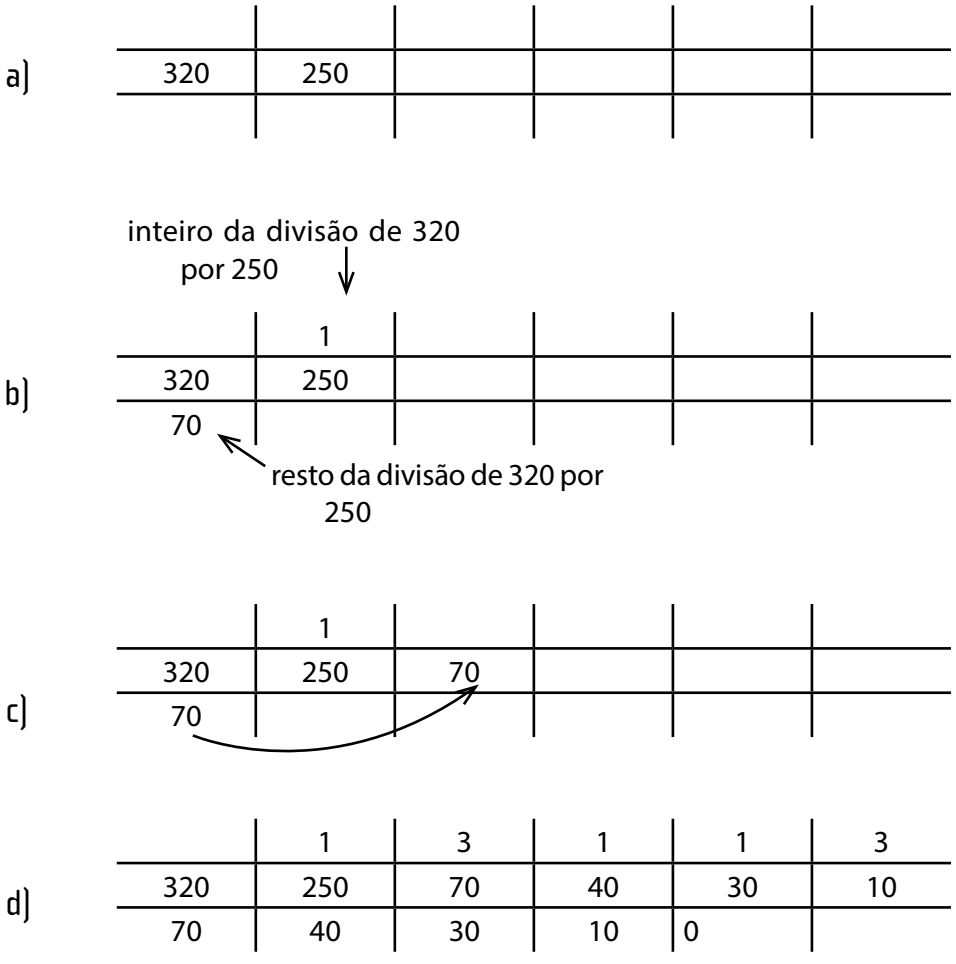
No código da Listagem 7.5 as linhas 3–12 são executadas repetidas vezes, enquanto (`while`) o valor de `x` (variável que representa o lado do quadrado) é diferente (~=) de -1. Podemos observar que o comportamento do comando `while` no Scilab é idêntico ao comando `repetirEnquanto` no RoboMind.

Vamos apresentar um outro exemplo de uso de instrução de repetição. Provavelmente você já ouviu falar do cálculo do Máximo Divisor Comum (MDC) de dois números

inteiros pelo algoritmo de Euclides. Na Figura 7.1 apresentamos um exemplo do cálculo do MDC dos números 320 e 250. Os passos para realizar o cálculo do MDC são:

1. Insira os dois números (em ordem crescente) nas duas primeiras colunas da segunda linha;
2. Divida o primeiro número (320) pelo segundo (250);
3. Coloque a parte inteira da divisão sobre o número 250;
4. Coloque o resto da divisão (70) abaixo do número 320;
5. Copie o resto da divisão para a terceira coluna da segunda linha;
6. Repita o processo a partir de 2, considerando o resto como segundo número e o primeiro como o antigo segundo número;
7. O processo para quando o resto da divisão for igual a zero e, neste caso, o MDC é dado pelo número da última coluna na segunda linha.

Figura 7.1: Algoritmo de Euclides do Máximo Divisor Comum de dois números inteiros positivos: MDC(320,250) = 10.



Como pode ser observado na Figura 7.1 o processo é iterativo, repetido, enquanto o resto da divisão é diferente de zero. No programa em Scilab da Listagem 7.6 mostramos como se calcular o MDC entre dois números.

Listagem 7.6: Programa para o cálculo do MDC.

```
1  primeiro = input ('Digite o primeiro numero: ');
2  segundo = input ('Digite o segundo numero: ');
3  cprimeiro = primeiro;
4  csegundo = segundo;
5  resto = 1;
6  while resto ~= 0
7     resto = modulo ( primeiro , segundo );
8     primeiro = segundo;
9     segundo = resto;
10 end
11 printf ('\ nMDC (%d ,%d)=%d ',cprimeiro , csegundo ,
          primeiro );
```

No início da Seção 7.2 apresentamos o problema de calcular iterativamente a expressão:

$$2^{10} = 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 = 1024$$

Podemos resolver este problema repetindo a operação de multiplicação 10 vezes, conforme o programa em Scilab da Listagem 7.7.

Listagem 7.7: Programa para realizar a operação de multiplicação.

```
1  contador = 1;
2  resultado = 1;
3  while contador <= 10
4     resultado = resultado * 2;
5     contador = contador + 1;
6  end
7  printf ('%d ', resultado );
```

Observe que no programa da Listagem 7.7 utilizamos duas variáveis: `resultado`, que acumula as multiplicações, e `contador`, que tem a exclusiva função de contar até dez. Este tipo de construção de repetição utilizando contador é muito comum. Por exemplo, para somarmos todos os números pares de 2 a 51, podemos escrever um programa em Scilab, conforme indicado na Listagem 7.8.

Listagem 7.8: Programa para realizar a soma de números pares.

```
1  contador = 2;
2  resultado = 0;
3  while contador <= 51
4     resultado = resultado + contador ;
5     contador = contador + 2;
6  end
7  printf ("%d",resultado );
```

Ambos os programas das listagens 7.7 e 7.8 utilizam a variável `contador`: no primeiro exemplo ela aumenta de um em um e, no segundo, de dois em dois. Uma forma mais compacta de se escrever laços de repetição com contadores é a utilização da instrução `for`, cuja sintaxe é:

```
for contador=inicio:fim
    <<instruções a serem repetidas>>
end
```

onde `inicio` é o valor inicial da variável `contador` e `fim` é o seu valor final. A variável `contador` é incrementada automaticamente em uma unidade a cada iteração. Podemos também especificar o passo do incremento utilizando a seguinte sintaxe:

```
for contador=inicio:passo:fim
          <<instruções a serem repetidas>>
end
```

Deste modo, os dois exemplos podem ser reescritos utilizando laços `for`, conforme apresentado nas listagens 7.9 e 7.10.

Listagem 7.9: Programa para realizar a operação de multiplicação, usando o comando `for`.

```
1  resultado = 1;
2  for contador = 1:10
3     resultado = resultado *2;
4  end
5  printf ('%d ', resultado );
```

Listagem 7.10: Programa para realizar a soma de números pares, usando o comando `for`.

```
1  resultado = 0;
2  for contador = 2:2:51
3     resultado = resultado + contador ;
4  end
5  printf ("%d",resultado );
```

7.6 CONSIDERAÇÕES FINAIS

Este capítulo apresentou uma estrutura poderosa que nos habilita a instruir os computadores na repetição de um conjunto de instruções, um certo número de vezes, que pode variar de acordo com as condições estabelecidas por nós, programadores. Finalizamos, assim, o estudo dos fundamentos da lógica de programação, que consistem apenas na "ponta do iceberg". A partir dos conceitos estudados nestes capítulos, muitos outros recursos e tecnologias mais avançados e poderosos continuam surgindo, alavancando o desenvolvimento tecnológico progressivamente e obtendo resultados que antes eram aparentemente impossíveis. Aos interessados, esses assuntos são estudados em disciplinas de cursos da área de Computação. No próximo capítulo, serão estudadas algumas técnicas computacionais de ampla aplicação, utilizadas na análise, compreensão e estudo de problemas complexos e multidisciplinares, e largamente usadas em inovação tecnológica e na pesquisa científica.

7.7 EXERCÍCIOS

7.7.1 EXERCÍCIOS COM O ROBOMIND

1. Faça um programa para que o robô, no ambiente dado pelo mapa `openArea.map`, pinte quadrados (de lado tamanho 3) em posições aleatórias do mapa e em cores aleatórias. Ele deve fazer isso até que o programa seja abortado pelo usuário;

2. Faça um programa para que o robô, no ambiente dado pelo mapa `maze1.map`, encontre sozinho o objeto no labirinto. A palavra "sozinho" aqui significa que você não pode programá-lo de modo a explorar a visão global que você tem desse labirinto específico. Você deverá programá-lo de modo que ele possa encontrar o objeto mesmo que a configuração do labirinto seja trocada;

3. Faça um programa para que o robô, no ambiente dado pelo mapa `findSpot1.map`, se encaixe na "garagem" definida pela posição pintada de branco. Aqui também o robô deverá fazer a tarefa sozinho;

4. Faça um programa para que o robô, no ambiente dado pelo mapa `copyLine1.map`, coloque os três objetos existentes em alguma das posições pintadas de preto do mapa. Mais uma vez, aqui também o robô deverá fazer a tarefa sozinho.

7.7.2 EXERCÍCIOS COM O SCILAB

1. Acesse o Scilab e teste os exemplos vistos na Seção 7.5;

2. Faça um programa que mostre a sequência de números naturais de 1 a 100;

3. Faça os exercícios enunciados abaixo empregando apenas estruturas de repetição, de decisão e os operadores aritméticos (+ - * /), isto é, sem usar funções matemáticas pré-definidas ou os demais operadores (por exemplo o operador ^) do Scilab;

a) Leia do teclado um número natural `n` e calcule `n!` (fatorial de `n`);

b) Leia do teclado um número natural `x` e um número natural `n` e calcule

$$S = \frac{1}{x^n}$$

4. Leia do teclado um número natural `n`. Em seguida leia uma sequência de `n` números inteiros (um por vez) e determine a soma dos números inteiros positivos e a soma dos números não-positivos desta sequência.

5. Leia do teclado um número natural `n`. Em seguida leia `n` notas pertencendo ao intervalo de 0 a 10 e verifique quantas notas são maiores ou iguais a 5 (aprovada) e quantas notas são menores do que 5 (reprovada);

6. Faça um programa que leia um número natural maior do que 1 e determine se ele é um número primo ou não. Neste exercício não é permitido utilizar a função `primes` do Scilab;

7. Faça programas que calculem o valor de `S` para os seguintes casos. Não é permitido utilizar condicional (`if`) nestes exercícios:

a. $S = \frac{1}{1} + \frac{3}{2} + \frac{5}{3} + ... + \frac{99}{50}$

b. $S = \frac{5}{1} - \frac{6}{4} + \frac{7}{9} - \frac{8}{16} + \frac{9}{25} - \frac{10}{36} + \frac{11}{49} - \frac{12}{64} + \frac{13}{81} - \frac{14}{100}$

8. Faça um programa que calcule a soma dos 50 primeiros termos da seguinte soma. Não é permitido utilizar condicional (`if`) nestes exercícios:

$$\frac{1000}{1} - \frac{997}{2} + \frac{994}{3} - \frac{991}{4} + ...$$

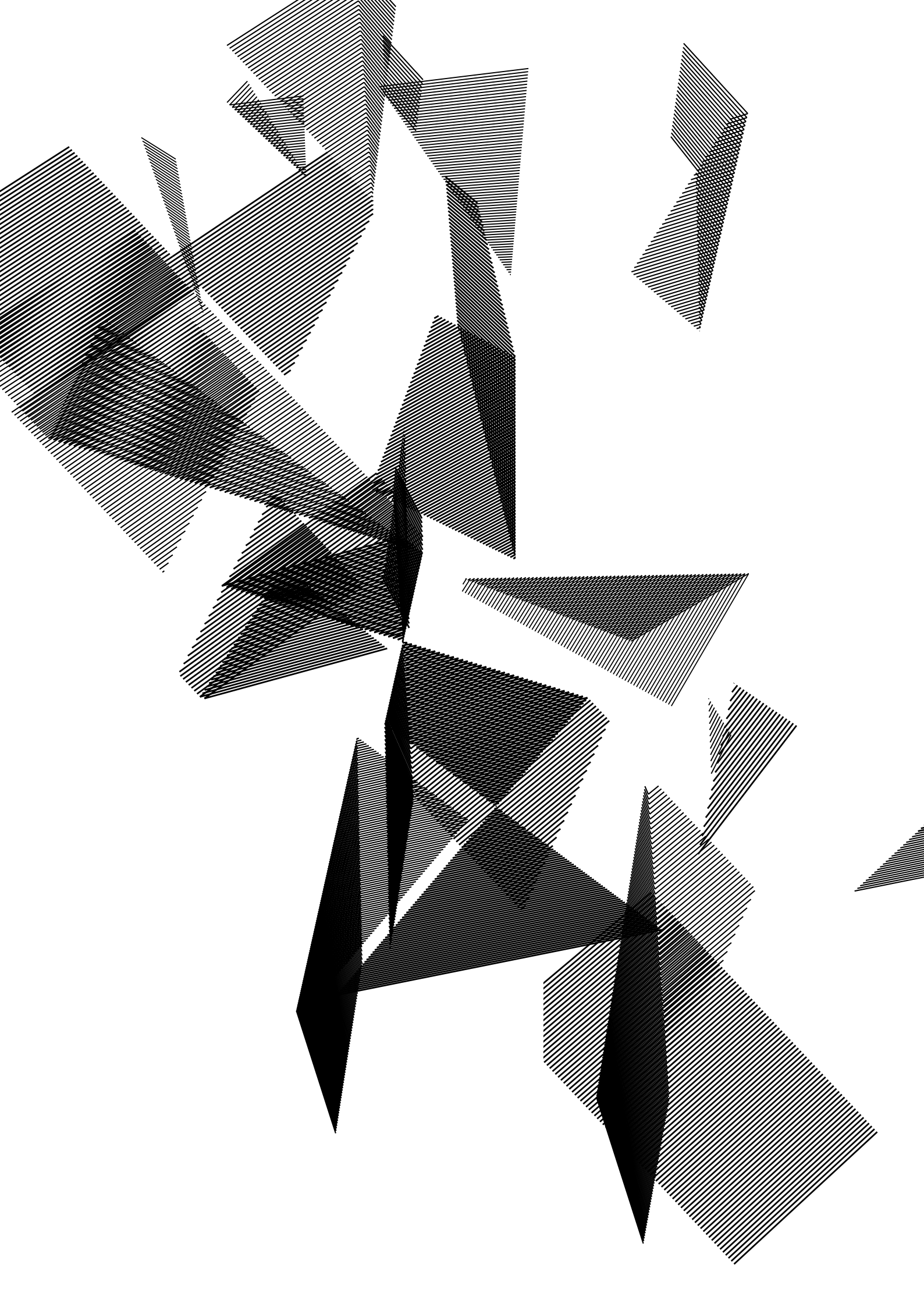
9. Leia do teclado um número natural `n` e uma sequência de `n` números naturais e verifique quantos são múltiplos de 3 e 5;

10. Leia do teclado um número natural `n` e uma sequência de `n` números naturais e verifique quantos são múltiplos de 11 ou 7;

11. A seguinte relação de recorrência nos fornece uma aproximação para $\sqrt{2}$

$$x_{k+1} = \frac{x_k^2 + 2}{2x_k}$$

considerando $x_0 > 0, k \geq 0$. Faça um programa que leia um $x_0 > 0$ (*chute inicial*) e um número natural `n` e faça `n` iterações desta relação de recorrência.



CAPÍTULO 8

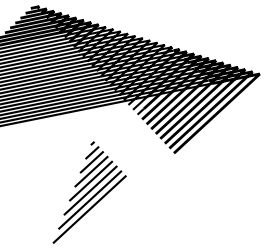
MODELAGEM E SIMULAÇÃO COMPUTACIONAL: CONCEITOS FUNDAMENTAIS

Alessandro S. Nascimento
Maria das Graças Bruno Marietto
Ricardo Suyama
Wagner Tanaka Botelho

8.1 INTRODUÇÃO

De maneira geral, quando falamos sobre Modelagem e Simulação, nos referimos a duas atividades distintas, porém intimamente relacionadas, que podem fornecer uma melhor compreensão acerca de um sistema, fenômeno ou processo observado. Elas se referem ao processo de obtenção de um *modelo*, uma representação, em geral simplificada, de aspectos físicos e operacionais de um *Sistema*, e à realização de *Simulações Computacionais*, isto é, executar ou operar um modelo em um ambiente computacional, com o propósito de entender o comportamento do sistema ou de avaliar estratégias para a operação do sistema.

De fato, a área de Modelagem e Simulação Computacional já é amplamente utilizada nos mais diferentes campos do saber, e o grande interesse nessa abordagem advém da grande facilidade e flexibilidade conferidas pelas simulações computacionais. O uso de modelos e simulações computacionais permite, por exemplo, identificar problemas e propor soluções para um sistema já existente; tentar predizer comportamentos e ações futuras; ou ainda analisar um sistema antes mesmo de que ele seja construído. Isto abre perspectivas para o desenvolvimento de novas tecnologias e o avanço científico, sendo necessário que futuros profissionais e cientistas tenham conhecimento sobre os fundamentos dessa área.



Dessa forma, neste capítulo abordaremos diferentes conceitos básicos ligados à área de Modelagem e Simulação Computacional, de maneira a preparar o aluno para que seja capaz de:

- Definir o que é um sistema e suas formas de estudo;
- Definir um modelo, bem como os passos principais para sua construção;
- Diferenciar modelos gerais dos modelos de simulação;
- Definir os conceitos de validação e verificação, no contexto de simulação computacional;

- Enumerar e descrever os passos para o desenvolvimento de simulações computacionais (metodologia);
- Trabalhar com simulações simples nos *softwares* Scilab e RoboMind;
- Testar hipóteses e analisar os resultados, em simulações computacionais.

Tendo isso em mente, o capítulo foi estruturado da seguinte forma. Nas Seções 8.2, 8.3 e 8.4 apresentamos e discutimos os conceitos de Sistema, Modelo e Simulação, incluindo uma metodologia geral para o desenvolvimento de simulações computacionais, respectivamente. Na Seção 8.5 apresentamos duas simulações computacionais, uma envolvendo a utilização do método de Monte Carlo para a solução de problemas, e outra implementada no ambiente RoboMind, emulando o comportamento de um robô que realiza comandos e reage ao ambiente de acordo com a programação feita pelo usuário. Na Seção 8.6 concluímos com algumas considerações, e deixamos na Seção 8.7 alguns exercícios a respeito do conteúdo estudado.

8.2 DEFINIÇÃO DE SISTEMA

O termo sistema vem do grego *snistánai* e significa "fazer ficar junto". Um sistema é um conjunto de elementos interligados e que interagem entre si, objetivando compor uma estrutura com um determinado nível de organização. Um sistema e seus elementos estão inseridos em um ambiente, havendo uma fronteira entre ambos. Assim, o que não pertencer ao sistema irá pertencer ao seu ambiente e vice-versa. Sistemas podem ser classificados como sistemas abertos ou sistemas fechados. Sistemas abertos interagem com seu ambiente e, consequentemente, o sistema e o ambiente influenciam-se mutuamente. Nos sistemas fechados não há interação dos elementos do sistema com elementos do ambiente, e por isto tais sistemas são considerados autossuficientes. Como exemplos de sistemas citam-se sistema computacional, sistema econômico, Sistema Solar, sistema nervoso, uma universidade e uma colônia de formigas.

Considerando esses exemplos, uma empresa é um sistema, pois (i) é formada por um conjunto de elementos (funcionários e proprietários), (ii) seus elementos possuem um objetivo em comum, que é manter a empresa no mercado, (iii) há relacionamentos entre os funcionários e proprietários, (iv) os elementos procuram estabelecer certa estrutura de organização e (v) a empresa está em um ambiente, que neste caso é o mercado econômico.

Este exemplo pode suscitar a seguinte pergunta: os clientes da empresa fazem parte do sistema ou do ambiente da empresa? Para responder a esta questão é preciso verificar se a empresa (o sistema em questão) tem o poder de controlar o cliente, obrigando-o a realizar determinadas ações. Caso um sistema tenha este poder sobre um elemento, então este elemento faz parte do sistema. Caso contrário, ele faz parte do ambiente do sistema. No caso do cliente, ele faz parte do ambiente da empresa, pois a empresa pode tentar induzi-lo a um tipo de comportamento, mas a decisão final de como utilizar os serviços da empresa é do cliente.

8.2.1 FORMAS DE SE ESTUDAR UM SISTEMA

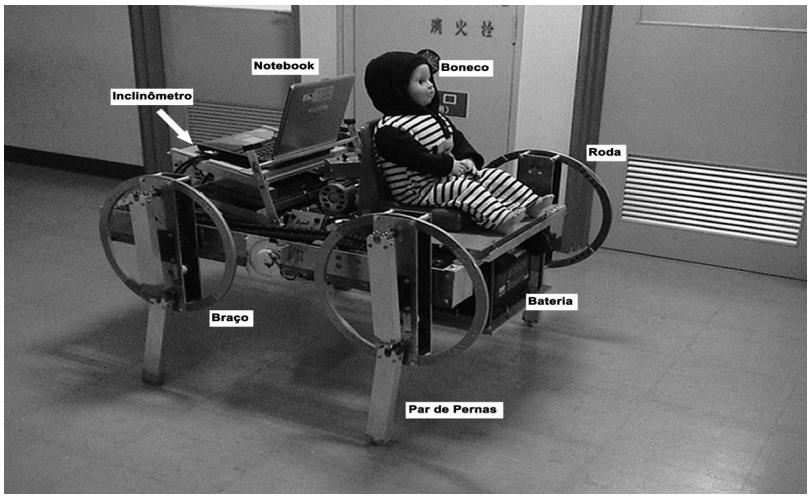
Nas próximas subseções são apresentadas algumas formas de se estudar sistemas, apresentadas em Averill e Kelton (1991), quais sejam: experimentos com o sistema real, experimento com modelos físicos e experimentos com modelos matemáticos.

Experimentos com o sistema real

Neste tipo de experimento trabalha-se diretamente com o sistema real, atuando em seus elementos e/ou alterando sua configuração para fazê-lo operar sob estas novas condições propostas. Por exemplo, um dos temas tratados na área de Robótica Móvel é a navegabilidade de robôs por ambientes desconhecidos, exigindo assim o reconhecimento dos objetos e dos limites de tais ambientes. Nesta situação os robôs podem interagir diretamente com o ambiente físico real, até mesmo alterando algumas de suas estruturas.

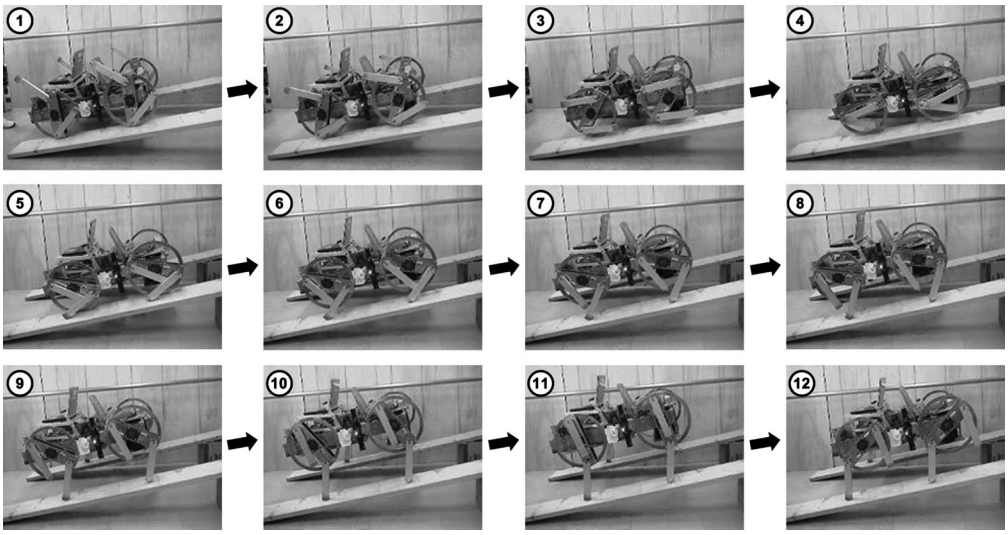
Um exemplo disso é o trabalho de Okada e colaboradores (2010), que tem como objetivo principal descrever a mudança no modo de locomoção entre pernas e rodas do robô híbrido PEOPLER-II (*Perpendicularly Oriented Planetary Legged Robot*), ilustrado na Figura 8.1. O robô pode realizar cinco tarefas: caminhar, locomover-se por rodas, realizar mudanças entre os modos de locomoção com pernas e rodas, virar para esquerda ou direita e rotacionar no sentido horário ou anti-horário.

Figura 8.1: Protótipo do robô PEOPLER-II.



Considere a situação na qual o robô locomove-se, utilizando rodas, e depara-se com uma rampa, conforme ilustrado na Figura 8.2. Neste caso, a tarefa do robô é realizar o chaveamento de roda para perna, como ilustrado nas Subfiguras de 1 à 12. Após a configuração ilustrada na Subfigura 10, o robô passa a utilizar a perna como meio de locomoção.

Figura 8.2: Resultado experimental no chaveamento de roda para perna.



Entretanto, lidar diretamente com o sistema real muitas vezes não é recomendado, pois:

- O experimento pode ser muito caro ou perigoso, até mesmo desestabilizando o sistema real. Por exemplo, analisar pessoas em uma situação de incêndio visando estudar seus comportamentos não é viável, tanto pela segurança física dos indivíduos, por questões éticas, bem como por uma possível inviabilidade econômica e logística;
- Em algumas situações é impossível tratar diretamente com sistemas reais como, por exemplo, no caso da análise direta dos buracos negros descritos pela astrofísica;
- Há também situações nas quais não há evidências da existência do sistema. Este tipo de estudo permite ao pesquisador abstrair *a priori* quaisquer relações físicas, sociais, psicológicas, econômicas, etc., usualmente conhecidas e adotadas.

Sendo assim, pode-se perceber que em muitas situações é necessário construir um modelo que represente parcialmente o sistema, e realizar experimentos com este modelo. Desta forma, é possível estudar o sistema real de maneira indireta, deixando-o inalterado. Para este momento é importante saber que um modelo é uma representação parcial de um sistema. Na Seção 8.3 tem-se um detalhamento maior deste conceito.

De acordo com Law e Kelton (1991), há duas formas de construir o modelo de um sistema: ou se faz um modelo físico ou se faz um modelo matemático.

Experimentos com modelos físicos

Os modelos físicos consideram experimentos com objetos reais, e tais objetos atuam como representações parciais do sistema que se deseja estudar. Como exemplo de modelos físicos pode-se citar mapas e maquetes de aviões.

Experimento com modelos matemáticos

Modelos matemáticos usam símbolos no lugar de componentes físicos, procurando representar as principais características e comportamentos do sistema-alvo que se

deseja analisar. A estrutura de um modelo matemático é formada por um conjunto de equações e/ou representações lógicas, que representam variáveis e relacionamentos qualitativos e quantitativos do referido sistema.

As equações e representações lógicas são tratadas e resolvidas visando analisar como o modelo reage sob determinadas condições, definidas pela atribuição de valores aos parâmetros do modelo. Os resultados obtidos da execução do modelo podem, posteriormente, ser comparados com dados obtidos no próprio sistema alvo (Berends e Romme, 1999).

De acordo com Law e Kelton (1991), há duas formas de solução de modelos matemáticos: a solução analítica e a solução numérica, *via* simulação.

Na solução analítica o problema matemático é resolvido encontrando-se a solução exata para a(s) equação(ções) que descreve(m) o modelo. Em casos como estes as teorias geralmente utilizadas são de Pesquisa Operacional, Teoria das Filas, Equações Diferenciais, dentre outras. Por exemplo, imagine que queremos achar a solução de um sistema que pode ser dada por uma equação de segundo grau, do tipo ax^2+bx+c, e que a solução do nosso problema consista em encontrar quais são as raízes deste polinômio. Neste cenário, podemos empregar a solução analítica para encontrar as raízes da equação, ou ainda uma solução numérica, via simulação.

Vamos começar encontrando uma solução analítica utilizando a fórmula de Bhaskara. Tomemos o polinômio da Equação 8.1.

$$2x^2 - 8x + 2 \tag{8.1}$$

Utilizando a fórmula de Bhaskara, nas equações 8.2 e 8.3, teremos como soluções as raízes 3,732 e 0,2679.

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{8.2}$$

$$x = \frac{8 \pm \sqrt{64 - 16}}{4} = \frac{8 \pm \sqrt{48}}{4} \tag{8.3}$$

Entretanto, há sistemas complexos cujos modelos não apresentam respostas precisas e definitivas, sendo necessário uma abordagem mais holística e sistêmica. Em tais casos uma solução exata não é possível. Nestes casos, simplificações podem ser feitas para viabilizar a construção de um modelo através de métodos empregados em simulações. Por exemplo, em Raczynski (2004) tem-se um estudo sobre como se organizam e interagem estruturas terroristas e anti-terroristas. Para tanto, o autor modelou elementos tais como: terroristas, agentes anti-terror, pessoas neutras, agentes anti-terror infiltrados em ambientes terroristas, etc. A execução do modelo deve atuar como um laboratório para o teste de hipóteses e um melhor entendimento do sistema alvo. Sendo assim, a possibilidade de executar o modelo várias vezes, simular diversas situações, elaborar e explorar hipóteses, etc., será o foco do uso do modelo matemático (não apenas uma solução analítica exata). Os modelos que apresentam este objetivo são denominados *modelos de simulação* e resolvidos via solução numérica, que são o foco de estudo deste capítulo.

Vamos retomar ao problema do cálculo das raízes de um polinômio. Caso uma solução analítica não fosse possível, poderíamos ainda estimar os valores das raízes de um modo relativamente preciso utilizando um método numérico *via* simulação. Neste exemplo, empregaremos o método de Newton-Raphson para estimar estas raízes. O método consiste em "chutar" um valor inicial para a raiz e melhorar a estimativa deste valor inicial através da Equação 8.4.

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{8.4}$$

Aqui, `f(x_n)` e `f'(x_n)` representam o valor do polinômio para os valores de `x` calculados e o valor da derivada do polinômio, respectivamente. A derivada de uma função `y=f(x)`, com relação à variável `x`, é usualmente escrita na forma $\frac{d_y}{d_x}$ e mede a taxa com a qual o valor de `y` varia em função de uma variação de `x`. Assim, podemos verificar a taxa de variação (derivada) de `y` com relação a `x` através da Equação 8.5.

$$\frac{\Delta y}{\Delta x} = \frac{y(x + \Delta x) - y(x)}{\Delta x} \tag{8.5}$$

Isto é, para um dado valor de `x`, calculamos o valor de `y` correspondente. Em seguida, calculamos um novo valor de `y`, mas, desta vez, somamos uma pequena diferença (`Δx`) no valor de `x`. Desta forma, teremos dois valores de `y`, calculados para `x` e para `x+Δx`. Podemos, finalmente, avaliar a taxa de variação da função `y` em razão da alteração que fizemos em `x` ao somar um Δx, através da relação mostrada na Equação 8.5. A derivada também pode ser vista, graficamente, como a inclinação da reta tangente à curva da função `y=f(x)`.

Se repetirmos a operação da Equação 8.5 para valores muito pequenos (infinitésimos) de `Δx`, trocamos a notação da derivada para a Equação 8.6.

$$\frac{dy}{dx} = \frac{y(x + \Delta x) - y(x)}{\Delta x} \Big|\lim_{\Delta x \to 0} \tag{8.6}$$

Nesta situação limite, na qual `Δx` tende a zero, algumas regras de derivação podem ser desenvolvidas para cada tipo de função.

A relação de Newton-Raphson mostra que, para uma solução inicial x_n, uma nova solução x_{n+1} pode ser calculada, refinando a estimativa inicial da raiz do polinômio. Após alguns ciclos de refinamento, uma solução precisa pode ser atingida através deste método. Por outro lado, vale dizer também que somente as raízes reais de um polinômio podem ser obtidas através deste método.

8.3 DEFINIÇÃO DE MODELO

De acordo com Shannon (1975), um modelo é uma representação parcial de um objeto, sistema ou ideia. A construção de um modelo é útil quando um sistema pode

ser simplificado/reduzido em um nível tratável, tornando clara a estrutura essencial do sistema e como seus componentes interagem entre si.

Um ponto importante na construção de um modelo é a definição de quais e como simplificações são introduzidas. Tais simplificações envolvem, por exemplo, a escolha dos elementos do sistema alvo a serem considerados, quais relacionamentos serão modelados, como delimitar o ambiente do sistema, etc. Esta escolha deve levar em consideração um ponto de equilíbrio, pois, muito embora a exatidão do modelo aumente em uma relação direta com as informações consideradas, sabe-se que uma parcela destas informações consegue explicar adequadamente o sistema de estudo. Assim, o ponto-chave do processo de modelagem é descobrir as variáveis principais e a relação entre elas.

Dentre as variáveis a serem inseridas em um modelo têm-se as variáveis de entrada e de saída. Espera-se que o modelo consiga estabelecer relações entre estas variáveis. Para tanto, é necessário entender quais aspectos do sistema alvo o modelo procura descrever, e quais são as limitações do modelo decorrentes das simplificações. A modelagem deve manter, com o maior grau de aderência e fidedignidade possível, o relacionamento entre o sistema alvo e o sistema resultante da modelagem.

8.3.1 CONSTRUÇÃO DE MODELOS

Os passos principais para a construção de um modelo estão descritos a seguir. Para ilustrar cada passo da construção de modelos, será apresentado o exemplo da modelagem de um sistema de robôs móveis exploradores.

Detectar o problema de interesse: Uma das principais etapas consiste em detectar o problema a ser modelado.

Para a modelagem dos robôs exploradores, o problema pode ser definido em como programar robôs móveis capazes de se deslocarem em um ambiente dinâmico e desconhecido, reconhecerem e planejarem seus movimentos. Na solução deste problema a infraestrutura teórico-técnica da área de Inteligência Artificial (IA) pode ser utilizada, mais especificamente no que se refere aos algoritmos de busca. Segundo Russell e Norvig (2004), o processo da busca por diferentes sequências de ações, e depois a escolha da melhor sequência, é conhecido como *busca*. Assim, ao final da execução de um algoritmo de busca tem-se a definição de uma sequência de ações possíveis de serem executadas.

Definição dos objetivos de estudo do sistema: Os objetivos indicam o que se pretende alcançar. Os objetivos direcionarão a construção do modelo, e geralmente são elaborados na forma de hipóteses a serem testadas, efeitos a serem analisados, etc.

Como objetivos do modelo dos robôs exploradores, tem-se que, em um ambiente desconhecido, um grupo de robôs procura mapear o ambiente na intenção de encontrar a saída. Para o mapeamento do ambiente os robôs compartilham suas informações, criando uma memória coletiva. Além de detectar e mapear

os objetos do ambiente, os robôs devem evitar trafegar em locais já conhecidos, evitar choques, bem como determinar o melhor trajeto até a saída.

Delimitação do escopo do modelo: O escopo limita a abrangência do estudo. Como exemplo de delimitação do escopo tem-se: estudar o tema (i) criminalidade, (ii) a criminalidade em termos de homicídio, (iii) os homicídios no Estado de São Paulo, (iv) qual a relação que existe entre o perfil dos assassinos e as condições sócio-econômicas do Estado, (v) no período de 1980 à 2011.

Como delimitação do escopo do exemplo dos robôs móveis, os robôs andarão em um labirinto.

Configurações do sistema a ser modelado: As configurações estabelecem para o modelo seus limites, restrições, etc. Dentre as principais configurações e premissas do sistema de navegação dos robôs exploradores tem-se:

- A modelagem do robô deve considerar suas características e limitações físicas;
- O robô possui quatro sensores que podem identificar obstáculos nas seguintes direções: para frente, para trás, à direita e à esquerda;
- As paredes são detectadas por estes sensores e adicionadas ao mapa;
- O mapa é compartilhado por todos os robôs, e todos contribuem para a formação dele;
- O mapa é confiável, ou seja, a informação fornecida por todos os robôs é real;
- Caminhos já percorridos devem ser evitados. Caminhos desconhecidos são priorizados;
- Quando há informação suficiente para encontrar a saída, algoritmos mais eficientes devem ser utilizados.

Construção do modelo propriamente dito: Aqui se tem a elaboração dos elementos principais do modelo, bem como seus relacionamentos. A seguir são apresentados os principais elementos do sistema de navegação dos robôs exploradores em um labirinto.

O agente robô virtual

Seguindo as especificações do modelo conceitual, o robô é capaz de movimentar-se nas quatro direções e detectar os obstáculos. A cada movimento o robô avalia a informação de seus sensores, bem como o seu último movimento, para determinar o próximo passo. Em primeiro lugar o robô verifica em quais direções é possível movimentar-se, detectando paredes e outros robôs. Estas informações são incluídas no mapa, informando onde há paredes e onde há espaços livres. Com estas informações, o robô faz as seguintes inferências:

- A posição que possui uma parede é descartada da lista de possíveis movimentos;
- Se há outro robô em alguma posição da lista, esta também é descartada.
- A escolha da nova posição, entre as disponíveis, considera nesta ordem:
- Saída do ambiente;

- Continuidade do movimento. Por exemplo, se o robô está indo para direção sul, ele prefere continuar nesta direção, considerando custoso o processo de mudança desta;
- Locais nunca visitados;
- Locais menos visitados.

No momento que o robô ocupa uma nova posição, este ponto no mapa é incrementado para identificar que uma nova visita foi feita neste ponto. Na próxima inferência, por este ou outro robô, esta posição terá um peso menor.

Métricas de desempenho: Estas métricas são utilizadas para medir a eficácia de diferentes configurações do sistema.

Para o caso da exploração de labirintos, uma das métricas a ser utilizada para analisar a eficácia do modelo é o tempo que os robôs demoram para sair do labirinto. Define-se que, quanto mais rápido os robôs encontrarem a saída, mais eficiente será o sistema.

Levantamento da infraestrutura necessária: Nesta etapa tem-se o levantamento dos equipamentos, ferramentas e programas computacionais a serem utilizados na construção do modelo.

Para a implementação do modelo dos robôs navegadores, considera-se necessário o uso de uma plataforma de simulação. Dentre as opções de plataformas disponíveis cita-se a plataforma Swarm Group (Swarm GD, 2013).

8.3.2 MODELOS DE SIMULAÇÃO

Um modelo de simulação é um tipo particular de modelo que também procura representar um determinado sistema, real ou abstrato. Entretanto, difere dos demais modelos na medida em que permite:

- Estudar como o sistema modelado comporta-se sob determinadas condições;
- Examinar, em variados graus de detalhamento, as consequências de alterações internas no comportamento geral do sistema.

Um dos primeiros modelos de simulação, desenvolvido por John Von Neumann e Stanislaw Ulam em 1940, durante a Segunda Guerra Mundial, foi utilizado para o estudo da blindagem de reatores nucleares. Este estudo seria muito caro em uma solução experimental real, e até mesmo muito perigoso em termos de segurança às pessoas envolvidas. Von Neumann e Ulam utilizaram ferramentas estatísticas com amostragens aleatórias como método para obter uma aproximação numérica de problemas complexos. Assim, conseguiram simular um comportamento complexo através de um modelo estatístico. Tal técnica foi denominada Método de Monte Carlo e será utilizada em uma simulação na Seção 8.5.1 para a estimativa do valor de π.

Em Gilbert e Troitzsch (1999) e Ferber (1996) são apresentados alguns objetivos de uma simulação:

- Elaborar e formalizar novas teorias, bem como auxiliar em um melhor entendimento de teorias já existentes;
- Testar hipóteses do sistema modelado;
- Obter um melhor entendimento de algumas características do sistema;
- Predizer comportamentos e ações futuras;
- Analisar e detectar elementos críticos do sistema.

A infraestrutura técnico-teórica da área de simulação permite capturar elementos essenciais de um sistema alvo, sem trabalhar diretamente com este. Esta característica é importante, por exemplo, em situações em que:

- É proibitivo, por ser arriscado ou caro, realizar experiências em situações reais;
- Deseja-se saber os limites de um sistema, sem o risco de destruí-lo.

8.4 SIMULAÇÃO COMPUTACIONAL

A área de Simulação Computacional propicia uma infraestrutura teórico-técnica adequada para a manipulação e geração de conhecimento de forma sistêmica e integrada. Isto porque propicia a criação e execução de modelos, em ambientes computacionais, que funcionam como laboratórios que podem ser usados para analisar e testar teorias, hipóteses, limites, previsões, etc. Assim, ambientes computacionais de simulação podem ser vistos como laboratórios, e uma simulação como um experimento realizado a partir de um modelo.

As técnicas e ferramentas computacionais de simulação têm sido usadas em diversas áreas como, por exemplo:

- Pesquisadores da área médica usam ambientes simulados para treinar técnicas de cirurgia, antes de testá-las em pacientes reais;
- Na indústria projetos de peças, máquinas, processos e produtos podem ser inicialmente desenvolvidos em ambientes controlados de simulação;
- Na simulação de catástrofes e no posterior trabalho de resgate, em queimadas, terremotos, situações de pânico coletivo, etc. Como exemplo de ambiente de simulação, veja o projeto RoboCup Rescue (RoboCupRescue, 2013).

8.4.1 VERIFICAÇÃO E VALIDAÇÃO

A credibilidade, e a consequente aceitação dos resultados de uma simulação, é atestada através de dois processos: verificação e validação.

O processo de verificação objetiva comprovar se o modelo computacional foi implementado corretamente em um sistema computacional. A comprovação se dá através da comparação entre o modelo conceitual e o modelo computacional (Sargent, 2010).

Aqui, a pergunta a ser respondida é: o modelo está implementado corretamente no computador?

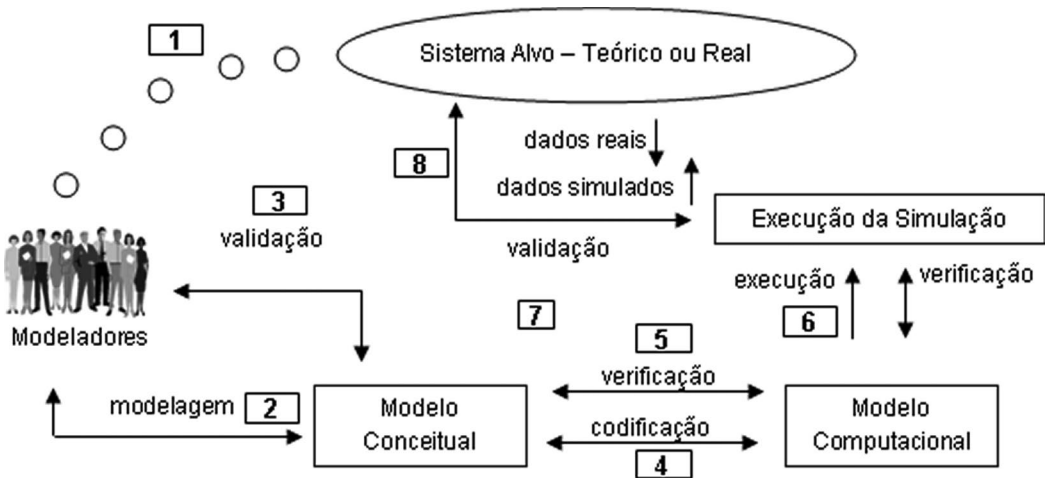
Para verificar a implementação de uma simulação computacional, técnicas tradicionais de Engenharia de *Software* podem ser utilizadas, bem como técnicas mais específicas para programas de simulação. O crescente número de variáveis presentes nas simulações pode representar um obstáculo no processo de verificação, aumentando o esforço necessário nesta fase de desenvolvimento. Neste caso, uma abordagem comum é a escolha de variáveis-chave no processo de verificação.

Por sua vez, validar um modelo significa comprovar que as considerações feitas para simplificar o comportamento do sistema real são aceitáveis. A questão a ser respondida no processo de validação é: o modelo conceitual foi construído de forma correta? Ou seja, como determinar que uma simulação é um modelo correto da realidade? A validação é um processo iterativo de comparação entre os comportamentos do modelo conceitual e do sistema alvo. Correções são feitas até que se atinja a precisão desejada, obtendo-se confiança de que a análise dos resultados da simulação leva a inferências válidas (Sargent, 2010).

8.4.2 METODOLOGIA PARA O DESENVOLVIMENTO DE SIMULAÇÕES COMPUTACIONAIS

A Figura 8.3 ilustra um ciclo completo para o desenvolvimento de uma simulação computacional.

Figura 8.3: Ciclo de vida de uma simulação computacional.



Passo 1. Análise do sistema alvo: O processo de desenvolvimento de uma simulação começa na análise do sistema alvo, objeto de estudo. Nesta etapa os modeladores formulam o problema, incluindo a definição de um objetivo. O objetivo da simulação guiará a escolha das partes do sistema alvo a serem modeladas. Como exemplo de objetivos tem-se a estimativa do tempo médio de espera de clientes em uma fila, a estimativa da quantidade de colisões de pacotes em uma rede, análise do comportamento coletivo em situações de pânico

em multidão, análise da evolução de normas sociais, etc. Neste passo também ocorre a coleta de dados para auxiliar na construção do modelo conceitual.

Passo 2. Construção do Modelo Conceitual: Tendo como base o sistema alvo e o objetivo da simulação, os modeladores constroem o modelo conceitual.

Passo 3. Validação Estática/Estrutural: As ações de modelagem devem ser revisadas na medida em que o desenvolvimento do modelo avança. Antes de ser codificado em um sistema computacional, o modelo precisa ser analisado para determinar se sua estrutura e seu funcionamento correspondem ao sistema alvo. Esta etapa é denominada validação estática/estrutural. A validação estática procura analisar se a representação lógica, matemática, e as relações de causa e efeito, retratam de forma adequada as características do sistema alvo.

Passo 4. Construção do modelo computacional: Após a validação estrutural do modelo conceitual, passa-se à modelagem computacional e sua posterior implementação. Ao final desta etapa tem-se um programa implementado, baseado no modelo computacional.

Passo 5. Verificação estática: Nesta etapa o modelo computacional e a execução do programa são analisados procurando assegurar que, primeiramente, as técnicas computacionais utilizadas estão corretas. Em um segundo momento procura-se analisar se as técnicas computacionais transcrevem corretamente o modelo conceitual para o ambiente computacional. Para tanto, inúmeros testes devem ser realizados como, por exemplo, colocar comprovações no programa para detectar certos tipos de erros. Uma comprovação usual é verificar se uma variável supera um valor impossível (por exemplo, uma probabilidade maior que 1), ou se a diferença entre indivíduos criados e destruídos é incoerente.

Passo 6. Execução do modelo computacional em testes piloto: Uma vez verificado o correto funcionamento técnico do modelo computacional, experimentos piloto são projetados tendo como base situações reais do sistema alvo. Após isto, os experimentos são executados no simulador. Estes experimentos piloto são simplificados, por exemplo, analisando-se os resultados da execução do programa tendo como dados de entrada uma quantidade reduzida de clientes, diminuindo o número de servidores de uma rede, etc.

Passo 7. Verificação dinâmica: Na etapa da verificação dinâmica o modelo computacional é executado com os dados dos experimentos piloto, procurando-se encontrar respostas previsíveis e comprovar se coincidem, ou não, com os resultados esperados da simulação. Isto não assegura que a simulação funcione de forma adequada para o caso geral, mas muitos erros podem ser encontrados/identificados em simulações simples (Sargent, 2010).

Passo 8. Validação dinâmica/comportamental: Nesta etapa o programa que representa o modelo computacional é executado diversas vezes, visando construir uma série de experimentos que permitirão extrair resultados e conclusões. Nesta etapa o comportamento do programa é conhecido como simulação computacional.

8.5 ATIVIDADES EM AULA

Conforme visto até o momento, o uso da modelagem e simulação pode trazer grandes benefícios ao estudo de sistemas e fenômenos observados. Nesta seção, colocaremos em prática algumas das ideias discutidas anteriormente, buscando ilustrar por meio de exemplos simples a utilidade das simulações na solução de problemas. No primeiro exemplo, abordaremos o método de Monte Carlo, um algoritmo computacional que se baseia em propriedades de números aleatórios para resolver problemas físicos e matemáticos para os quais não seria possível obter uma solução analítica ou não seria possível utilizar um algoritmo determinístico. Em outro exemplo, apresentamos um modelo computacional que emula o comportamento de um robô, que segue comandos programados pelo usuário.

8.5.1 ESTIMANDO O VALOR DE π NO SCILAB COM O MÉTODO DE MONTE CARLO

No colégio, todos aprendemos a usar a constante π em Geometria para calcular a área de figuras geométricas, por exemplo. Aprendemos, também, que o valor da constante é de 3,1415926535... Contudo, nem sempre nos ensinam como este valor pode ser calculado a partir de relações mais simples. Nesta prática, nosso objetivo é empregar um modelo matemático simples para calcular o valor de π, utilizando um método clássico de simulação computacional.

Há diversos métodos que nos permitem calcular o valor de π de maneira relativamente simples. Um destes métodos é o método de Monte Carlo, que se baseia nas propriedades dos números aleatórios (Metropolis e Ulam, 1949; Metropolis *et al.*, 1953). O método de Monte Carlo foi desenvolvido por Nicholas Metropolis, John von Neumann e Stanislaw Ulam na década de 1940, durante o Projeto Manhattan, no laboratório norte-americano de Los Alamos, onde se desenvolvia o projeto da bomba atômica. Como o projeto era secreto, os nomes dos estudos não podiam ser alusivos à bomba. Neste caso, Monte Carlo foi o nome dado em homenagem ao bairro do principado de Mônaco, famoso pelos cassinos.

O método de Monte Carlo usa números aleatórios para calcular propriedades de interesse. Uma distribuição de números aleatórios pode ser definida, de maneira simples, como um conjunto de números no qual nenhum número tem maior chance de aparecer do que outro. Por incrível que pareça, é algo bem difícil sortear números aleatórios (mesmo com um computador!). Alguns pesquisadores já notaram, por exemplo, que, se pedirmos a um conjunto de pessoas que digam um número qualquer de maneira "aleatória", os números que obteremos dificilmente passarão num teste estatístico que comprove que o conjunto realmente trata-se de um conjunto de números aleatórios. Contudo, existem rotinas computacionais bastante eficientes e já estabelecidas para gerar sequências de números aleatórios.

As aplicações do método de Monte Carlo são vastas. Alguns exemplos incluem:

- Ciências físicas: simulações de difusão de calor, simulações de dinâmica molecular, problemas de muitos corpos (*many-body*) em sistemas quânticos, etc.;

- Engenharia: Engenharia microeletrônica (análise de variações correlacionadas e não-correlacionadas em circuitos digitais e analógicos), localização de robôs autônomos, etc.;
- Biologia Computacional: Dinâmica de proteínas e membranas, inferências baesianas na construção de árvores filogenéticas, etc.;
- Matemática: Métodos de integração, otimização, etc.

Para compreender o uso do método de Monte Carlo, vamos primeiro desenvolver algumas relações geométricas que nos levarão a uma equação para o valor da constante π. Sabemos que a área de uma circunferência de raio `r` é dada pela Equação 8.7.

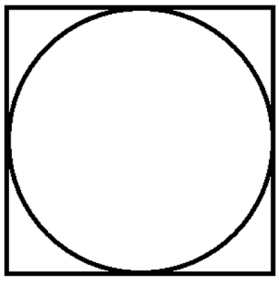
$$A_{circ} = \pi r^2 \tag{8.7}$$

Sabemos, ainda, que a área de um quadrado de lado `l` é dada pela Equação 8.8.

$$A_{quad} = l^2 \tag{8.8}$$

Agora, considere uma situação na qual temos um quadrado com uma circunferência inscrita no quadrado, como mostrado na Figura 8.4.

Figura 8.4: Circunferência inscrita em um quadrado.



Se a circunferência tem um diâmetro de `2r`, sabemos que a área do quadrado será dada pela Equação 8.9.

$$A_{quad} = l^2 = (2r)^2 = 4r^2 \tag{8.9}$$

Neste ponto, se calcularmos a razão entre a área da circunferência e a área do quadrado, teremos a Equação 8.10.

$$\frac{A_{circ}}{A_{quad}} = \frac{\pi r^2}{4r^2} \tag{8.10}$$

Rearranjando os termos e isolando π do lado esquerdo da equação teremos, finalmente, a Equação 8.11, que usaremos para calcular o valor da constante.

$$\pi = 4\,\frac{A_{circ}}{A_{quad}} \tag{8.11}$$

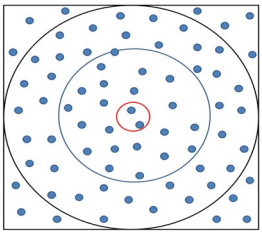
A Equação 8.11 nos mostra que é possível saber precisamente o valor de π, bastando para isso calcular a razão entre a área da circunferência e a área do quadrado, e multiplicar esta razão por 4. Contudo, para sabermos a área da circunferência, precisamos

da constante π, como nos mostrou a Equação 8.7. Como resolver, então, este círculo vicioso?

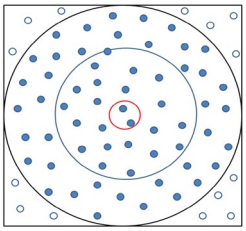
A resposta para a solução do problema está no método de Monte Carlo, que tem como fundamento o uso de números aleatórios para a estimativa de parâmetros de interesse. Aqui, usaremos o método e as propriedades dos números aleatórios para estimar a relação entre as áreas da circunferência e do quadrado.

A ideia do método de Monte Carlo está representada na Figura 8.5. A Figura 8.5(a) mostra um conjunto de pontos azuis, distribuídos de maneira aleatória dentro da área de um quadrado. No mesmo quadrado temos três circunferências inscritas de raios diferentes. A Figura 8.5(b) mostra em azul os pontos que estão somente dentro da esfera maior, enquanto as figuras 8.5(c) e 8.5(d) mostram, respectivamente, os pontos que estão dentro das esferas média e menor, somente. A observação nos faz perceber que, em uma distribuição aleatória de pontos, a quantidade de pontos que vai cair dentro da esfera inscrita deve ser proporcional à área da esfera!

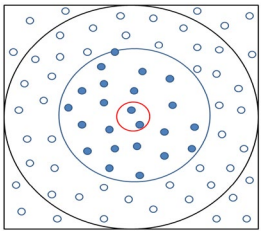
Figura 8.5: Pontos aleatórios no quadrante e na circunferência.



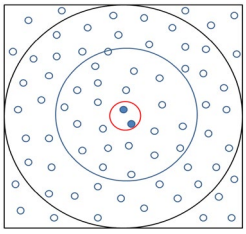
(a) Pontos aleatórios em um quadrado.



(b) Pontos aleatórios na circunferência maior.



(c) Pontos aleatórios na circunferência média.



(d) Pontos aleatórios na circunferência menor.

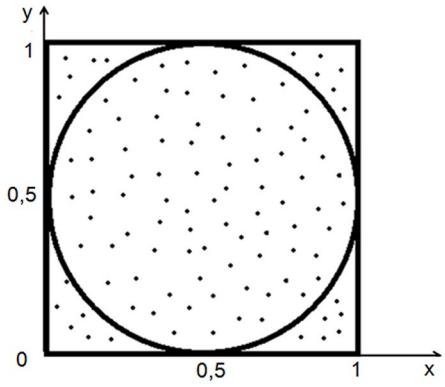
Podemos tentar escrever esta mesma conclusão na forma da Equação 8.12. Um número maior de pontos vai cair dentro de qualquer circunferência (N_{circ}), tanto maior for a área da circunferência (A_{circ}) em relação à área do quadrado (A_{quad}).

$$\frac{N_{circ}}{N_{quad}} = \frac{A_{circ}}{A_{quad}} \tag{8.12}$$

A Equação 8.12 também nos mostra que os números aleatórios nos permitem estimar a razão $\frac{A_{circ}}{A_{quad}}$. Fazendo da leitura da equação da direita para a esquerda, vemos que a razão entre a área da circunferência e a área do quadrado é dada pela razão do número de pontos aleatórios que caem dentro da circunferência em relação ao número de pontos aleatórios que caem dentro do quadrado. É esta a relação que usaremos para estimar o valor de π.

Para tornar os cálculos mais simples, vamos assumir que o quadrado tenha um lado de tamanho `l=2r=1`. Logo, teremos que o raio da circunferência será $r = \frac{1}{2}$. Desta forma, podemos imaginar a base do quadrado como um eixo `x` (abscissa) de um sistema de coordenadas cartesianas e a altura do quadrado como um eixo `y` (veja Figura 8.6). Utilizando um computador, nós sortearemos agora alguns pares de números aleatórios no intervalo [0,1]. Cada par de números representará as coordenadas `x` e `y` de um ponto que pertence à área do quadrado, como mostra a Figura 8.6.

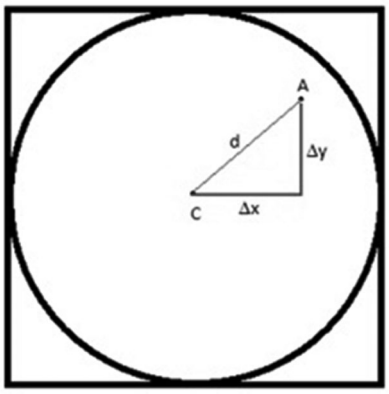
Figura 8.6: Circunferência inscrita em um quadrado e com uma série de pontos aleatoriamente distribuídos sobre a área do quadrado.



Se os números sorteados forem realmente aleatórios, a chance de um ponto cair no quadrado ou no círculo será basicamente proporcional às áreas do quadrado e do círculo. Desta forma, podemos estimar as áreas do quadrado e do círculo simplesmente contando quantos pontos caem sobre cada umas das figuras. Ainda melhor, podemos calcular a razão $\frac{A_{circ}}{A_{quad}}$ simplesmente através da razão entre o número de pontos que cai na circunferência em relação ao número total de pontos. Lembre-se de que, uma vez que os pontos estão no intervalo [0,1], todos os pontos cairão no quadrado.

Dadas as coordenadas **(**`x`**,** `y`**)** de um ponto `A` qualquer, oriundas de um sorteio aleatório, podemos saber se o ponto está dentro ou fora da circunferência calculando a distância entre `A` e o ponto central da circunferência `C`, com coordenadas `x=0,5` e `y=0,5`, como mostrado na Figura 8.7.

Figura 8.7: Distância *d* entre qualquer ponto *a* e o centro da circunferência pode ser calculada através do cálculo de uma hipotenusa.



Para estar dentro da circunferência, a distância entre o ponto `A` e o centro da circunferência deve ser menor que o raio. Este processo está descrito nas equações 8.13, 8.14 e 8.15.

$$\Delta x = x_A - x_C \tag{8.13}$$

$$\Delta y = y_A - y_C \tag{8.14}$$

$$d = \sqrt{(\Delta x)^2 + (\Delta y)^2} \tag{8.15}$$

Desta forma, se sortearmos as coordenadas `x=0,8` e `y=0,75`, por exemplo, teremos `Δx=0,8-0,5=0,3` e `Δy=0,75-0,5=0,25` sendo $d = \sqrt{0{,}3^2 + 0{,}25^2} = 0{,}3905$. Como o valor obtido é menor do que o valor do raio da circunferência, é possível dizer, mesmo sem a necessidade de uma análise gráfica, que o ponto está dentro da circunferência. Como o ponto está dentro do limite [0,1], também está dentro do quadrado. Caso o valor da distância `d` obtida seja maior que o raio, o ponto estará fora da circunferência, porém dentro do quadrado.

Assim, cumprimos as duas tarefas: i) sortear muitos números aleatórios de forma a gerar uma amostragem suficiente de pontos dispersos randomicamente no quadrado e ii) decidir se cada ponto encontra-se dentro ou fora da circunferência. Com isto, podemos contar o número `N` de pontos sorteados e o número de pontos que estão dentro da circunferência. Com estas duas informações, finalmente calculamos o valor de π através da Equação 8.11.

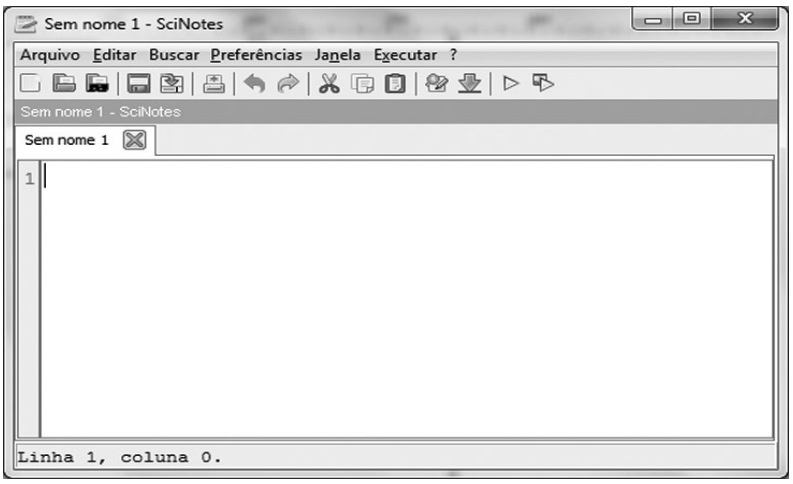
O *script* da Listagem 8.1 traz uma série de comandos que podem ser utilizados no Scilab, a fim de se realizar o cálculo do valor de π numericamente através do método de Monte Carlo. Um *script* nada mais é do que um conjunto de comandos em uma ordem lógica, e tais comandos podem ser interpretados por uma linguagem de programação ou por um ambiente computacional.

Listagem 8.1: Script para o cálculo de π.

```
1  circ=0;
2  pi_calc=0;
3  N = 1000000;
4  for i=1:N
5     x=rand();
6     y=rand();
7     d=sqrt(((x-0.5)*(x-0.5))+((y-0.5)*(y-0.5)));
8     if d < 0.5 then
9       circ = circ+1;
10    end
11 end
12 pi_calc = 4*(circ/N);
13 difer = (pi_calc - %pi);
14 difer
15 pi_calc
```

Para usar este *script* digite o texto da Listagem 8.1 no ambiente Scinotes do Scilab, como apresentado na Figura 8.8. Em seguida, execute os comandos do *script* selecionando `Executar`⟶`Arquivo com Eco`, ou simplesmente pressionando `Ctrl`+`L`.

Figura 8.8: Ambiente Scilab e SciNotes.



Observe que o nosso *script* não faz nenhuma checagem quanto ao sorteio repetido de variáveis aleatórias. Isto quer dizer que, eventualmente, o mesmo ponto pode, em princípio, ser sorteado mais de uma vez. Como baseamos nossos cálculos em números que são aleatoriamente sorteados pelo computador, os resultados obtidos em diferentes simulações no mesmo computador serão ligeiramente diferentes. Além disso, o seu resultado e o resultado dos seus colegas poderão ser também diferentes. Vale comparar e conferir. Coloque os comandos da Listagem 8.1 no seu Scilab e observe o que acontece.

Neste *script* definimos as variáveis `CIRC` e `PI_CALC` nas linhas 1 e 2. Estas variáveis guardarão os valores do número de pontos que caem dentro do círculo e o valor de π que está sendo calculado pelo método de Monte Carlo, respectivamente. Na Linha 3 definimos o número de pontos que serão aleatoriamente distribuídos na nossa si-

mulação de Monte Carlo. Altere este número para observar o efeito da amostragem sobre o método, por exemplo. A Linha 4 introduz uma estrutura de repetição denominada *laço* ou *loop*. Ela permite que o Scilab execute um determinado número de vezes um comando, ou uma sequência de comandos. No nosso caso temos a estrutura a seguir:

```
for i=1:N
...
end
```

Esta estrutura diz ao Scilab que todos os comandos compreendidos entre as linhas 5 e 10 deverão ser repetidos `N` vezes, ou seja, um milhão de vezes (Linha 3). Para sabermos em que passo da repetição o Scilab está, uma variável interna ao laço é definida também na Linha 4, como a variável `I`. Por exemplo, os comandos a seguir:

```
N=10;
for i=1:N
disp("Estou no passo", i);
end
```

Este trecho de código mostra a mensagem "Estou no passo", seguida do número do passo em que o laço está. Em outras palavras, a variável `i` é atualizada a cada repetição do laço.

Os comandos `RAND()`, usados nas linhas 5 e 6, são empregados para sortear números aleatórios. Por padrão, o Scilab sorteia números entre 0 e 1, embora este padrão possa ser alterado para trabalhar com outros universos de números aleatórios. A Linha 7 faz o cálculo da distância entre o ponto aleatório sorteado, com coordenadas `x` e `y`, e o centro do círculo (coordenadas `x=0,5` e `y=0,5`).

A Linha 8 emprega um *comando de seleção* do tipo `IF` (`SE`, em Português). Se a distância `D` calculada na Linha 7 for menor que 0,5 (raio da circunferência), o ponto está dentro da circunferência e uma unidade é somada à variável `CIRC`. Do contrário, nada é feito.

A Linha 12 calcula, finalmente, o valor de π como 4 vezes o valor `CIRC` dividido por `N`. A Linha 13 compara o valor do "nosso" π, calculado pelo método de Monte Carlo, e o valor da constante π guardada pelo Scilab. As linhas 14 e 15 mostram estes valores.

Exercícios propostos

A seguir, têm-se algumas atividades a serem realizadas com relação ao cálculo do valor de π através do método de Monte Carlo:

1. Qual foi o valor de π calculado?
2. O que acontece se você repetir a simulação 5 vezes?
3. Experimente, agora, diminuir a amostragem de dados, diminuindo o valor de `N` para 100.000, por exemplo. Qual foi o valor de π calculado com a amostragem menor?

8.5.2 SIMULANDO O RESGATE DE VÍTIMAS NO ROBOMIND

Em 1999 foi proposto um novo domínio para o estudo de agentes autônomos, conhecido como RoboCup Rescue (RoboCupRescue, 2013), que tem como intenção promover apoio em decisões emergenciais. Tal apoio ocorre pela integração das informações de desastres, previsões, planejamento e interface humana, utilizando robôs.

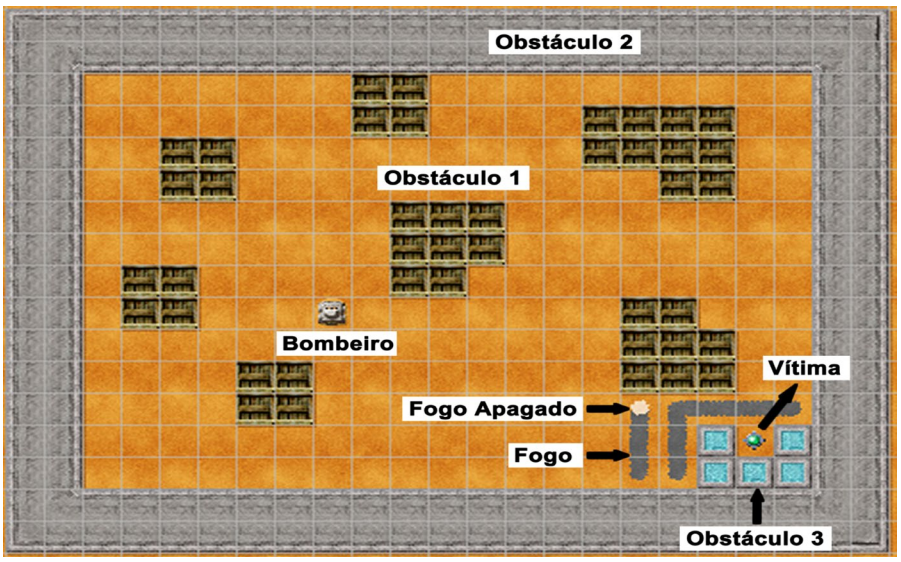
Nesta seção iremos apresentar uma simulação no RoboMind que tem como objetivo o resgate de pessoas após um desastre ambiental em larga escala, como um terremoto ou uma inundação.

Objetivo do ambiente de resgate

Os robôs de resgate são projetados com o objetivo de ajudar na busca e no salvamento de pessoas em atentados, como o que ocorreu no World Trade Center, em Nova Iorque, além de ser utilizado em desastres naturais como terremotos e *tsunamis*.

No ambiente de resgate, ilustrado na Figura 8.9, um robô que representa um bombeiro deverá encontrar a vítima. Mas, antes, será necessário apagar todo o fogo que encontrar pelo caminho e desviar dos obstáculos.

Figura 8.9: Ambiente de resgate no RoboMind.



Modelagem de um ambiente de resgate

Para a criação do cenário de um resgate da Figura 8.9, no RoboMind será necessário criar um mapa. Para que isso seja possível, você pode abrir o Bloco de Notas, ou qualquer editor de texto, e salvar o arquivo com extensão `.map` com os comandos definidos na Figura 8.10. O mapa criado deve ser aberto no menu `Arquivo`⟶`Abrir Mapa` no RoboMind.

Figura 8.10: Mapa do cenário do resgate da Figura 8.9.

```
paint:
{(b,|,16,12),(b,|,16,13),(b,|,17,12),(b,|,17,13),(b,-,17,12),(b,-,18,12),
(b,-,19,12)}
map:
CHHHHHHHHHHHHHHHHHHHD
GMFFFFFFFFFFFFFFFFFJI
GI       QQ          GI
GI       QQ    QQQQ  GI
GI  QQ         QQQQ  GI
GI  QQ           QQ  GI
GI        QQQ        GI
GI        QQQ        GI
GI QQ     QQ         GI
GI QQ   @        QQ  GI
GI                QQQ GI
GI    QQ          QQQ GI
GI    QQ              GI
GI                 O*OGI
GI                 OOOGI
GLHHHHHHHHHHHHHHHHHHKI
BFFFFFFFFFFFFFFFFFFFFE
```

Para implementar uma simulação utilizando o RoboMind é necessário definir o cenário. Para tanto, vamos modelar o sistema considerando os seguintes elementos:

• O Obstáculo 1 na Figura 8.9 é representado, no mapa da Figura 8.10, pela letra (Q);

• O Obstáculo 2 é representado pelas letras B, C, D, E, F, G, H, I, J, K, L, M;

• O Obstáculo 3 é representado pela letra O;

• A Vítima é a baliza, sendo representada pelo símbolo *;

• A posição inicial do Bombeiro é indicada pelo símbolo @;

• O Fogo em chamas é representado pela linha preta, e o Fogo apagado pelo ponto branco.

Se o Bombeiro encontrar uma linha preta, o ponto branco deve ser inserido em cima da linha para indicar que as chamas foram controladas. A Baliza é o objeto que o robô pode "capturar". O Bombeiro só consegue salvar a Vítima se ela estiver na célula imediatamente à frente. Da mesma forma, a Vítima só será solta em algum lugar seguro na célula imediatamente à frente do Bombeiro, caso não tenha nenhum obstáculo nessa célula.

A seguir tem-se a descrição das funcionalidades de cada elemento do modelo:

• Bombeiro: anda aleatoriamente em busca de focos de incêndio e da Vítima. Quando percebe Fogo em qualquer ponto do ambiente, deve ir até ele e apagá-lo. Além disso, deve desviar dos Obstáculos e resgatar a Vítima;

• Vítima: tem um comportamento estático e deve ser resgatada pelo Bombeiro. Este resgate ocorre quando a Vítima é retirada do local onde está e levada para um lugar seguro. No código da Listagem 8.2 o Bombeiro tem o comportamento de chegar até à Vítima e levá-la a um lugar seguro;

• Fogo: é representado pelas linhas pretas e deve ser controlado pelo Bombeiro;

• Obstáculos: representam as construções, destruídas ou não, por algum acidente natural.

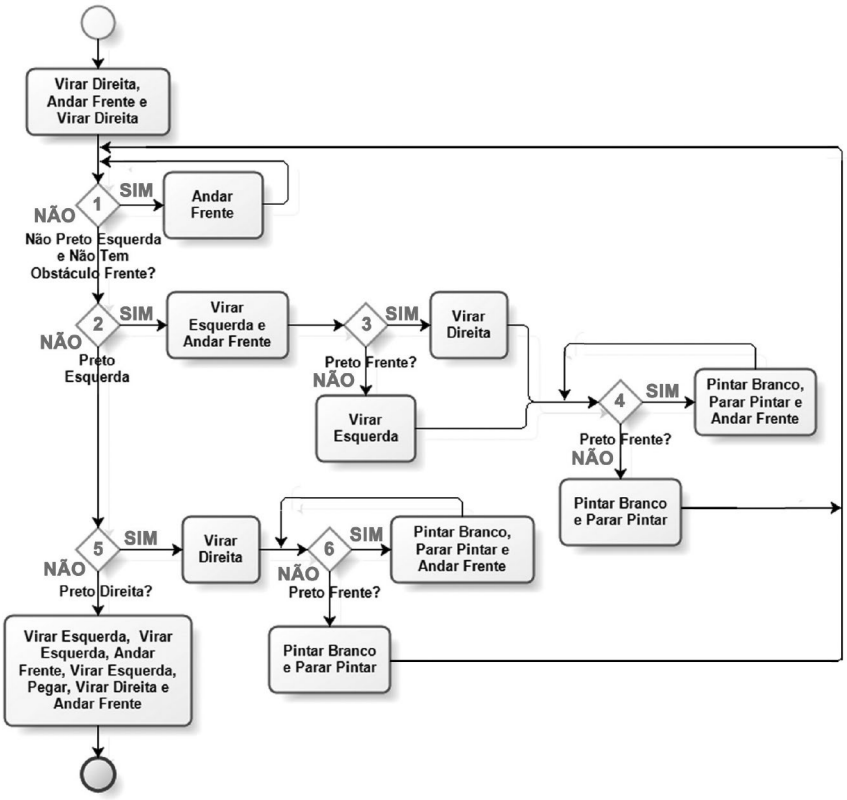
O comportamento do robô BOMBEIRO, que realiza o resgate, está implementado no código da Listagem 8.2. Esse arquivo deve ser salvo no RoboMind com a extensão .IROBO. O fluxograma do algoritmo pode ser visualizado na Figura 8.11.

Listagem 8.2: Implementação da simulação de resgate.

```
1  virarDireita()
2  andarFrente(7)
3  virarDireita()
4
5  repetir()
6  {
7     repetirEnquanto(não pretoEsquerda() e
         não temObstáculoFrente())
8     {
9        andarFrente(1)
10    }
11
13    se(pretoEsquerda())
12    {
14       virarEsquerda()
15       andarFrente(1)
16
17       se(pretoFrente())
18       {
19          virarDireita()
20       }
21       senão
22       {
23          virarEsquerda()
24       }
25
26       repetirEnquanto(pretoFrente())
27       {
28          pintarBranco()
29          pararPintar()
30          andarFrente(1)
31       }
32
33       pintarBranco()
34       pararPintar()
35    }
36    senão se(pretoDireita())
37    {
38       virarDireita()
39
```

```
40        repetirEnquanto(pretoFrente())
41         {
42            pintarBranco()
43            pararPintar()
44            andarFrente(1)
45         }
46
47         pintarBranco()
48         pararPintar()
49      }
50      senão
51      {
52         virarEsquerda()
53         virarEsquerda()
54         andarFrente(1)
55         virarEsquerda()
56         pegar()
57         virarDireita()
58         andarFrente(9)
59         fim
60      }
61   }
```

Figura 8.11: Fluxograma da simulação de resgate.



Nas linhas 1-3 o Bombeiro irá andar até encontrar o primeiro Obstáculo. O primeiro laço de repetição, definido no Losango 1 da Figura 8.11, está relacionado com as linhas 7-10. O objetivo é andar para frente até encontrar o primeiro sinal de Fogo.

Se tiver Fogo no lado esquerdo do Bombeiro (Linha 12, Losango 2 da Figura 8.11) o movimento de virar para a esquerda e andar para frente tem como objetivo ir na direção do Fogo. Se o Bombeiro encontrar mais focos de Fogo na sua frente (Linha 17 e Losango 3) ele deverá apagar o Fogo que foi primeiro encontrado. Para isso o movimento do Bombeiro será virar para a direita (Linha 19) e depois o Fogo será apagado nas linhas 26-31, que também está representado na condição definida no Losango 4. Após terminar (linhas 33-34) de apagar o primeiro foco de incêndio, o Bombeiro deverá ir em direção ao segundo foco (linhas 12-35, Losango 2), que está localizado à sua esquerda.

Nas linhas 36-49 e nos losangos 5 e 6, o Bombeiro irá apagar o Fogo que está localizado à sua direita. Caso não haja Fogo na esquerda (Linha 12, Losango 1) nem na direita (Linha 36, Losango 2) do Bombeiro, isso significa que o incêndio já foi controlado e a Vítima deve ser resgatada. Essa última tarefa está implementada nas linhas 50-60 e na quinta condição do losango.

Exercícios propostos

Com base no comportamento do robô implementado no código da Listagem 8.2, você deve entender a lógica de seu funcionamento para:

- Mudar o comportamento do Bombeiro para que a Vítima seja resgatada para a posição no canto superior esquerdo, conforme ilustrado na Figura 8.12;
- Definir novos focos de incêndio e analisar a necessidade de implementar novos comportamentos para o Bombeiro. Lembre-se de que, antes do Bombeiro capturar a Vítima, é necessário apagar todo o Fogo;
- Definir mais de uma Vítima em outras posições, e verificar o comportamento do Bombeiro. Lembre-se de que o Bombeiro só poderá resgatar uma Vítima por vez;
- Elaborar o fluxograma para os novos algoritmos gerados.

Figura 8.12: Nova posição da *Vítima* após ser resgatada.

8.6 CONSIDERAÇÕES FINAIS

O campo da Modelagem e Simulação Computacional é bastante amplo e se configura como uma abordagem multidisciplinar para estudo de sistemas e solução de problemas. Nesse capítulo foram apresentados os princípios básicos relacionados à área, bem como uma metodologia geral para o desenvolvimento de simulações computacionais. Os exemplos didáticos desenvolvidos ilustram apenas alguns dos potenciais usos de modelos computacionais para a pesquisa científica e desenvolvimento tecnológico. No capítulo seguinte discutiremos alguns outros modelos, a fim de ilustrar como simulações computacionais podem ser utilizadas como laboratórios para auxiliar na compreensão dos sistemas estudados.

8.7 EXERCÍCIOS

1. Identifique as principais características dos seguintes sistemas: um computador, uma universidade, o sistema solar, uma colônia de bactérias, uma família e um país. As características são as seguintes: (i) os elementos que constituem o sistema, (ii) os relacionamentos entre os elementos, (iii) o objetivo em comum e (iv) o ambiente onde o sistema está inserido.

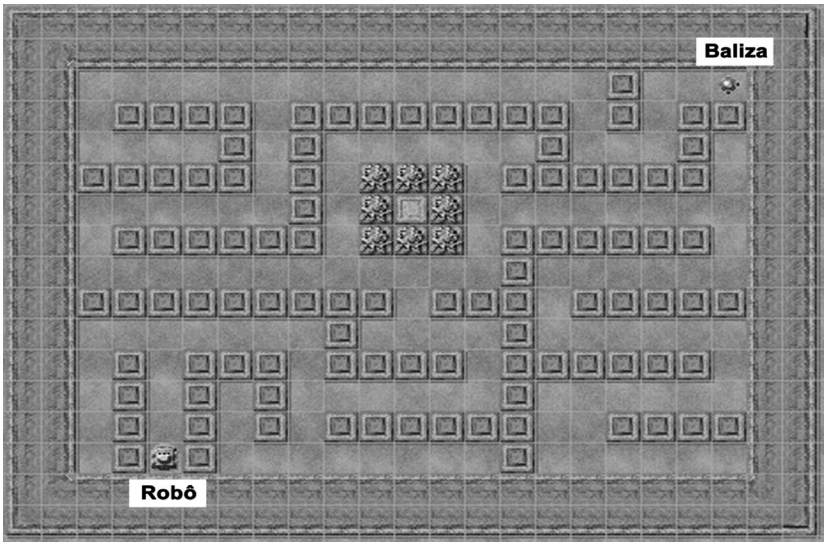
2. De forma equivalente ao exercício anterior, escolha três sistemas de seu interesse e identifique suas principais características.

3. Considerando a simulação para o cálculo do valor de π pelo método de Monte Carlo, modifique o *script* dado na Listagem 8.1 para, ao final de cada iteração, calcular a diferença entre o valor de π calculado e o valor de π real. Ao final, plote os dados no Scilab, tendo o número da iteração como abscissa e o valor da diferença como ordenada.

8.7.1 LABIRINTO NO ROBOMIND

Um labirinto é constituído por vários percursos, criados para confundir quem os percorre. A Figura 8.13 apresenta o cenário de um labirinto no RoboMind.

Figura 8.13: Ambiente do labirinto no RoboMind.



O código do mapa no RoboMind, relacionado ao ambiente da Figura 8.13, está apresentado na Figura 8.14.

Figura 8.14: Mapa do cenário do labirinto da Figura 8.13.

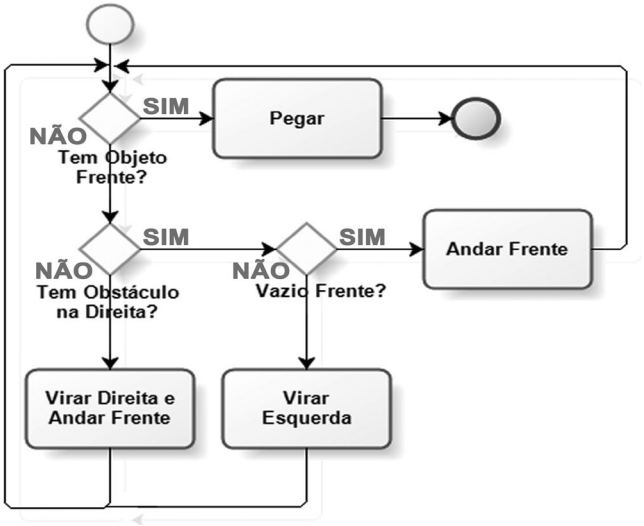
```
map:
CHHHHHHHHHHHHHHHHHHHHD
GMFFFFFFFFFFFFFFFFFFJI
GI                  A  *GI
GI AAAA AAAAAAAA A AAGI
GI    A A       A   A GI
GIAAAAA A PPP AAAAAA GI
GI      A POP        GI
GI AAAAAA PPP AAAAAA GI
GI            A      GI
GIAAAAAAAAA AAA AAAAAGI
GI        A    A      GI
GI A AAA AAAA AAAAAA GI
GI A A A      A      GI
GI A A A AAAAAA  AAAAGI
GI A@A        A      GI
GLHHHHHHHHHHHHHHHHHHHKI
BFFFFFFFFFFFFFFFFFFFFE
```

O objetivo do robô é encontrar a baliza, que representa a saída do labirinto. O algoritmo desenvolvido está no código da Listagem 8.3 e o fluxograma está ilustrado na Figura 8.15. Pode-se observar que o algoritmo é bem simples para um problema que pode parecer complexo.

Listagem 8.3: Implementação da simulação de um labirinto.

```
1  repetir()
2  {
3     se(temObjetoFrente())
4     {
5        pegar()
6        fim
7     }
8     senão
9     {
10       se(temObstáculoDireita())
11       {
12          se(vazioFrente())
13          {
14             andarFrente(1)
15          }
16          senão
17          {
18             virarEsquerda()
19          }
20       }
21       senão
22       {
23          virarDireita()
24          andarFrente(1)
25       }
26    }
27 }
```

Figura 8.15: Fluxograma do algoritmo do labirinto.



Faça as seguintes atividades, tendo como base o algoritmo da Listagem 8.3 e o fluxograma da Figura 8.15:

1. Para um melhor entendimento da solução apresentada, faça uma relação entre o algoritmo e o fluxograma. Por exemplo, o código das linhas 3-7 está relacionado com o primeiro losango "Tem objeto frente"?

2. Após executar a simulação e entender o algoritmo implementado, que tipo de comportamento o robô tem para encontrar a baliza? Justifique sua resposta;

3. Modifique a posição da baliza e do robô para o canto superior esquerdo e inferior direito, respectivamente. É possível melhorar o algoritmo para o novo cenário? Justifique a sua resposta;

4. Crie outros cenários de labirinto e defina outras posições para o robô e também a baliza. Verifique o comportamento do sistema.

8.7.2 MÉTODO DE NEWTON-RAPHSON

O método de Newton ou Newton-Raphson é um dos métodos mais empregados para a resolução de equações algébricas não lineares. O método evoluiu a partir de proposições de Isaac Newton e Joseph Raphson (Ypma, 1995). Nesta seção usaremos o método de Newton para calcular, numericamente (via simulação), as raízes de polinômios de qualquer ordem maior ou igual a 2.

No método de Newton-Raphson, a raiz de um polinômio é inicialmente estimada a partir de um "chute" inicial. A partir daí este valor inicial é refinado em iterações sucessivas, a partir da Equação 8.16.

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{8.16}$$

Aqui, $f(x_n)$ é o valor do próprio polinômio calculado com o valor de x estimado, e $f'(x_n)$ é o valor da derivada do polinômio. A cada passo, um novo valor de x é obtido e usado no passo posterior, em uma nova iteração de refinamento do valor de x como raiz do polinômio. O processo pode ser repetido por um número determinado de vezes, ou ainda pode ser repetido até que haja uma convergência entre os valores de x_n e x_{n+1}, por exemplo.

Neste exemplo, tomaremos como parâmetros o número máximo de ciclos como 500 e um critério de convergência de $1E^{-5}$, ou seja, 0,00001. Isto quer dizer que o programa vai repetir 500 vezes a iteração, mostrada na Equação 8.16, ou até que a Equação 8.17 seja satisfeita.

$$|x_{n+1} - x_n| < 0{,}00001 \tag{8.17}$$

Quando um dos critérios é atingido, o programa finaliza sua execução e mostra o valor final da raiz obtida (x_{n+1}), bem como o número de ciclos que foram necessários para atingir aquela raiz.

A Listagem 8.4 mostra os comandos necessários para implementar o método de Newton-Raphson no Scilab. Para esta finalidade, abra o ambiente SciNotes no Scilab e digite os comandos da Listagem 8.4.

Listagem 8.4: Cálculo de raízes utilizando o método de Newton-Raphson.

```
function y=fx(coeffs, x)
   y=0;
   maior_termo=length(coeffs)-1;
   for i=1:length(coeffs)
      y= y + (coeffs(i))*(x^maior_termo);
      maior_termo = maior_termo - 1;
   end
endfunction

function y=dfdx(coeffs, x)
   y=0;
   maior_termo=length(coeffs)-1;
   for i=1:length(coeffs)
      y = y + (maior_termo) * (coeffs(i)) *
          (x^(maior_termo-1));
      maior_termo = maior_termo - 1;
   end
endfunction

count=0;
termos = input("Entre com a ordem do polinomio >= 2): ");
termos = termos+1;
coeffs = []
for i=1:termos
   printf("Entre com o termo %d: ", i);
   t = input("");
   coeffs = [coeffs t];
end

for i=1:termos-1
   printf("%+dx^%d ", coeffs(i), termos-i);
end
printf("%+d\n", coeffs(termos));

old_x = input("Entre com um valor estimado da raiz: ");
func = fx(coeffs, old_x);
deriv = dfdx(coeffs, old_x);

if deriv ~= 0.00
   new_x = (double(old_x) - double(func/deriv));
end
delta = abs(new_x-old_x);
```

```
42
43 while (count < 500) & (delta > 1E-5)
44    func = fx(coeffs, old_x);
45    deriv = dfdx(coeffs, old_x);
46    if deriv ~= 0.00
47       new_x = (double(old_x) - double(func/deriv));
48       delta = abs(new_x-old_x);
49       old_x=new_x;
50       count = count + 1;
51    else
52       printf("A derivada calculada foi zero. \nPor favor,
            tente novamente com outro valor inicial de x\n");
53       count=501;
54    end
55 end
56
57 printf("Raiz: %.5f\n", new_x);
58 printf("Convergencia em %d ciclos\n", count);
```

Finalmente, vamos empregar o método de Newton que acabamos de implementar para calcular qual é a raiz real do polinômio $2x^3+5x^2-3x+25$**.** A entrada dos dados é realizada de forma interativa, e o resultado obtido deve ser similar ao mostrado a seguir:

```
-->exec('C:\Users\UFABC\Desktop\newton.sci', -1)
Entre com a ordem do polinomio >= 2): 3
Entre com o termo 1:
2
Entre com o termo 2:
5
Entre com o termo 3:
-3
Entre com o termo 4:
25
+2x^3 +5x^2 -3x^1 +25
Entre com um valor estimado da raiz: 100
Raiz: -3.77469
Convergencia em 24 ciclos
```

É possível verificar que o método de Newton funcionou corretamente de duas formas:

- Calculando as raízes com o Scilab:

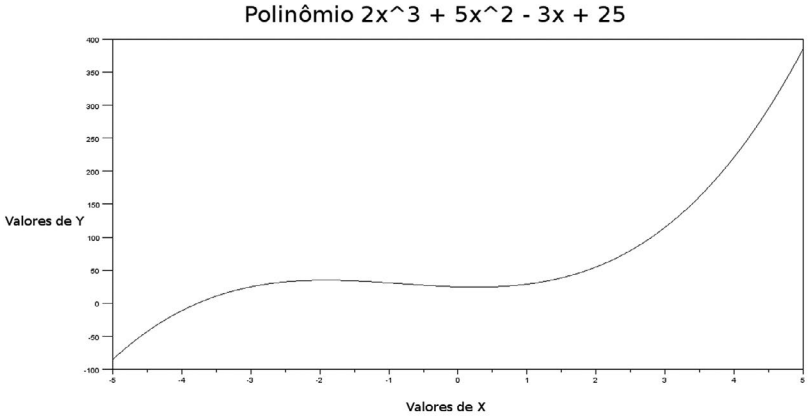
```
-->p = poly([25, -3, 5, 2], "x", "coeff");
-->roots(p)
```

```
ans  =
  - 3.774685
    0.6373425 + 1.7045026i
    0.6373425 - 1.7045026i
```

O resultado mostrado pelo Scilab revela uma raiz real (–3.774685) e duas raízes imaginárias;

- Ou fazendo uma análise de forma gráfica, como ilustrado no gráfico da Figura 8.16.

Figura 8.16: Análise gráfica do polinômio.



O *script* usa muitos conceitos já trabalhados nesta disciplina e outros que com estudos adicionais podem ser entendidos com facilidade. Vale destacar alguns destes itens:

- Funções: As linhas 1 e 10 mostram que é possível definir funções no Scilab. No caso da Linha 1, definimos uma função que chamamos de `fx` e que tem como argumentos um vetor `COEFFS` e um número real `x`. A partir daí, o valor do polinômio é calculado como um valor `y`. O uso desta função facilita a implementação do algoritmo, uma vez que permite separar partes do código com objetivos diferentes em regiões (ou até arquivos) diferentes;

- Entrada de dados: Em algumas porções do código o comando `INPUT` é usado para fazer a entrada de dados. Este comando permite que o usuário, ao executar o código, possa entrar com dados de maneira interativa.

Com relação ao *script* do método de Newton-Raphson, faça as seguintes atividades:

- Experimente alterar os parâmetros de convergência e executar novamente o *script*. O que acontece?

- Altere o *script* utilizado para o cálculo da raiz real de um polinômio na Listagem 8.4, modificando os critérios de parada. Utilize $1E^{-10}$ em vez de $1E^{-5}$ e um número máximo de ciclos como 5000, em vez de 500. O que acontece? Como estes parâmetros alteram a execução do programa?

- Qual a finalidade do comando `abs` no comando `delta=abs(new_x--old_x)`? Faça uma pesquisa deste comando.

REFERÊNCIAS BIBLIOGRÁFICAS

BERENDS, P. & Romme, G. "*Simulation as a Research Tool in Management Studies*", *EUROPEAN Management Journal*, v. 17, n, 6, pp.576–583, 1999.

FERBER, J. *Reactive Distributed Artificial Intelligence: Principles and Applications*. In: G. O´Hare e N. R. Jennings (ed.). *Foundations of Distributed Artificial Intelligence*. New York, NY, USA: John Wiley & Sons, 1996. pp. 287–314.

GILBERT, N. & TROITZSCH, K. G. *Simulation for the Social Scientist*. Buckingham – Philadelphia, USA: Open University Press, 1999.

LAW, A. M. & KELTON, W. D. *Simulation Modelling and Analysis*. 2nd edition. Industrial Engineering Series. New York: McGraw-Hill International Editions, 1991.

METROPOLIS, N.; ROSENBLUTH, A.; ROSENBLUTH, M., TELLER, A. & TELLER, E. "Equations of State Calculations by Fast Computing Machines", *Journal of Chemical Physics*, v. 21, n. 6, pp.1087–1092, 1953.

METROPOLIS, N. & ULAM, S. "The Monte Carlo Method", *Journal of the American Statistical Association*, v. 44, n. 27, pp.335–341, 1949.

OKADA, T.; BOTELHO, W. T. & SHIMIZU, T. "Motion Analysis with Experimental Verification of the Hybrid Robot Peopler-II for Reversible Switch Between Walk and Roll on Demand", *The International Journal of Robotics Research*, v. 29, n. 9, pp.1199-1221, 2010.

RACZYNSKI, S. "Simulation of the Dynamic Interactions between Terror and Anti-terror Organizational Structures", *Journal of Artificial Societies and Social Simulation*, v. 7, n.2, 2004. Disponível em http://jasss.soc.surrey.ac.uk/7/2/8.html, Última visita em 28/02/2013.

ROBOCUPRESCUE. *Robocup Rescue Home Page*. http://www.robocuprescue.org. 2013. Última visita em 28/02/2013.

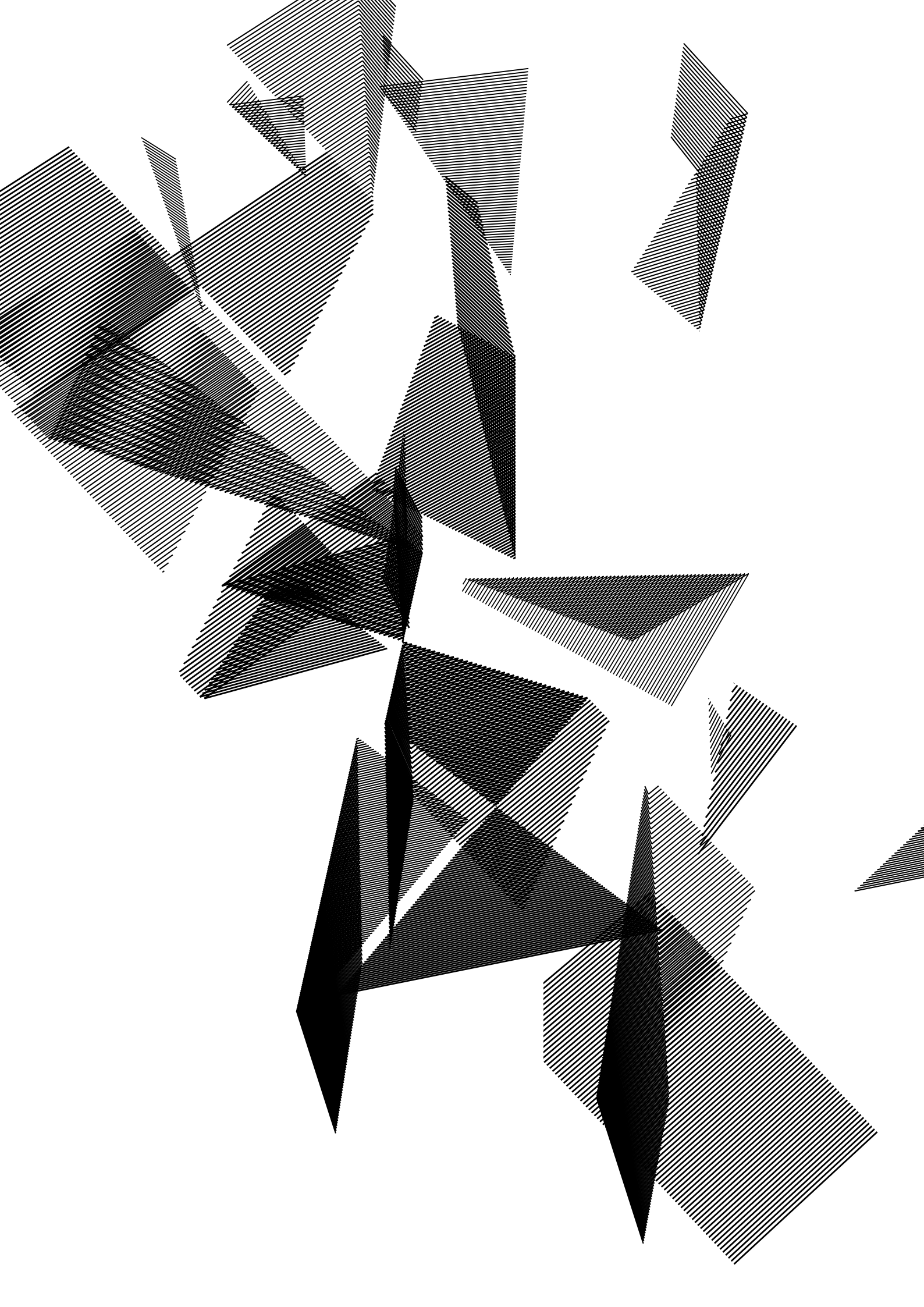
RUSSELL, S. & NORVIG, P. I*nteligência Artificial*. São Paulo: Editora Campus, 2004.

SARGENT, R. G. Validation and Verification of Simulation Models. In: *Proceedings of the Winter Simulation Conference*, 2010, pp. 166-183.

SHANNON, R. E. *Systems Simulation: The Art and Science*. Prentice-Hall, Inc., 1975.

SWARM G. D. *Swarm Development Group Home Page*, 2013. http://www.swarm.org. Última visita em 28/02/2013.

YPMA, T. J. "Historical Development of the Newton-Raphson Method", *SIAM Review*, v. 37, n. 4, pp.531–551, 1995.



CAPÍTULO 9

MODELAGEM E SIMULAÇÃO COMPUTACIONAL: A CIÊNCIA NA PRÁTICA

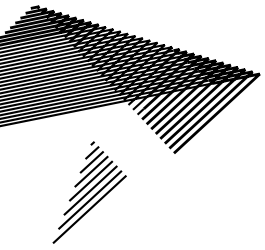
Alessandro S. Nascimento
Maria das Graças Bruno Marietto
Ricardo Suyama
Wagner Tanaka Botelho

9.1 INTRODUÇÃO

No Capítulo 8 foi apresentado um panorama geral dos principais conceitos relacionados à área de Modelagem e Simulação Computacional. No presente capítulo, daremos continuidade ao estudo de sistemas por meio de simulação computacional, procurando utilizar os resultados das simulações na análise de hipótese e testes empíricos.

Dessa forma, abordaremos outros dois exemplos de simulação, desenvolvidos nos *softwares* Scilab e RoboMind, com o intuito de ilustrar de que forma os modelos computacionais podem trazer benefícios para o avanço científico e tecnológico. Ao final do capítulo, espera-se que o aluno seja capaz de:

- Utilizar simulações computacionais como laboratórios para o estudo de teorias e hipóteses;
- Associar os conceitos de modelagem e simulação nos exemplos apresentados;
- Analisar os resultados dos sistemas implementados no Scilab e RoboMind.



Para esse fim, este capítulo está estruturado da seguinte forma. Na Seção 9.2 discutimos como simulações computacionais podem ser vistas como laboratórios, apresentando para isso um exemplo de simulação do comportamento social de pessoas vivendo em uma sociedade indígena. Na Seção 9.3 desenvolvemos duas simulações computacionais, uma baseada no *software* RoboMind, e outra implementada no *software* Scilab, a fim de ilustrar como é possível utilizar o modelo computacional para extrair conclusões relevantes sobre o sistema estudado. Por fim, finalizamos o capítulo apresentando na Seção 9.4 as considerações finais, e na Seção 9.5 apresentamos alguns exercícios computacionais para fixação dos conceitos discutidos.

9.2 SIMULAÇÃO COMPUTACIONAL COMO UM LABORATÓRIO

Simulações computacionais podem ser consideradas como laboratórios onde teorias e hipóteses são exploradas ou desenvolvidas, possibilitando um melhor entendimento dos sistemas analisados (Ruas *et al.*, 2011). Para tanto, inicialmente os modelos são construídos a partir de um arcabouço teórico formado por um rigoroso conjunto de regras, estruturas e conceitos relacionados ao sistema em questão.

Posteriormente, o modelo é implementado em um sistema computacional, possibilitando análises empíricas em um ambiente digital seguro e controlado. A execução da simulação e seus resultados podem auxiliar a identificar situações não previstas, gerar novas hipóteses, entender melhor o sistema em estudo, aperfeiçoar e/ou rever as teorias usadas na construção do modelo, etc.

Neste contexto, o uso de simulações computacionais deve ser visto como parte de um processo exploratório que inclui ações tais como (Parker *et al.*, 2003): (i) testar o conjunto de regras, estruturas e conceitos utilizados no modelo (teoria utilizada), através da demonstração de que este arcabouço teórico pode levar a resultados coerentes; (ii) analisar outros possíveis motivos que levam a um mesmo resultado, explorando assim a robustez das explicações das relações causais.

Pelo o que foi visto até o momento, simulações computacionais são utilizadas para estudar um sistema através de experimentos. Embora esta afirmação pareça simples, suas consequências não são tão óbvias. Por isto, como exemplo de como uma simulação pode de fato ser utilizada como um laboratório para o teste de teorias e hipóteses, apresentamos na Subseção 9.2.1 a simulação Cyber-Anasazi.

9.2.1 SIMULAÇÃO COMPUTACIONAL DA SOCIEDADE ANASAZI

A civilização Anasazi foi uma sociedade indígena que habitou o que hoje é o Four Corners, no Estado do Arizona (EUA), entre 1200 a.C. até o final do século XIII. Têm-se muitos dados sobre esta civilização, porém os motivos que levaram à sua extinção são muito discutidos. Hipotetiza-se que esta sociedade foi extinta devido a sua cultura e guerras. Há aqueles que acreditam em epidemias ou saqueadores, outros defendem uma possível migração. Entretanto, até hoje não há uma explicação efetiva para a extinção dessa civilização (Janssen, 2009).

O modelo computacional de simulação Cyber-Anasazi foi desenvolvido inicialmente por Axtell e colaboradores (2002), pesquisadores do Santa Fe Institute. Este modelo visa reproduzir o comportamento da cultura e o declínio da sociedade Anasazi até o seu desaparecimento, objetivando encontrar os motivos de sua extinção. Há uma grande quantidade de dados levantados em estudos e pesquisas sobre a sociedade, que permitem definir tanto o comportamento de seus indivíduos como regras para a agricultura, movimento, crescimento populacional, quanto os números ligados à produção agrícola, dados meteorológicos daquela região no período em questão, padrões de assentamento, etc.

A simulação consiste em criar uma réplica computadorizada do Long Valley House e simular o período 800 d.C. a 1350 d.C. Um agente neste modelo representa uma família e possui informações como idade, tempo de vida, habilidade para locomoção, necessidade nutricional, consumo, etc. O conjunto de regras que regem o comportamento dos agentes permite que interajam tanto com outros agentes quanto com o ambiente em que estão inseridos (Axtell *et al.*, 2002).

Resumidamente, o comportamento de um agente família se dá da seguinte forma. Todos os anos cada família realiza colheitas de milho em suas terras para se alimentar, e armazena o que excede para que possa consumir e garantir sua sobrevivência no inverno. Se a produção é suficiente para a alimentação neste ciclo, então a família permanece nas terras para o plantio do ano seguinte. Se o rendimento for insuficiente, a família se muda para o próximo lote disponível que pareça promissor, para então realizar o mesmo ciclo. Se, mesmo assim, a família ainda não puder se sustentar, então ela é retirada da simulação, como se tivesse morrido. Há regras que permitem a formação de novas famílias, o nascimento de filhos, etc. (Axtell *et al.*, 2002).

As simulações realizadas, baseadas no modelo Anasazi, auxiliaram em um melhor entendimento da evolução desta civilização, mas não conseguiram explicar por que os Anasazi desapareceram. Mesmo com esta limitação no resultado dos experimentos, tais simulações têm demonstrado que as sociedades artificiais são uma ferramenta importante na reprodução da gênese, evolução e extinção de sociedades reais, podendo também predizer algumas tendências.

9.3 ATIVIDADES EM AULA

Nesta seção tem-se a apresentação de exemplos de como trabalhar com simulações, considerando-as como laboratórios computacionais.

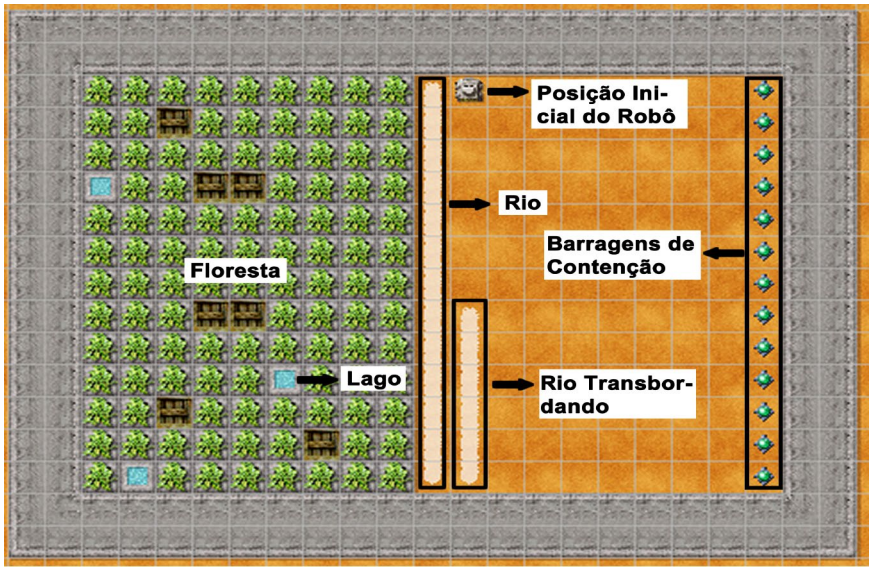
9.3.1 SIMULAÇÃO DO TRANSBORDAMENTO DE UM RIO NO ROBOMIND

A cada dia os robôs estão mais presentes no dia a dia de nossa sociedade, podendo ser desenvolvidos com o objetivo de prestar assistência em diversas situações. Por exemplo, em casa, na indústria, na medicina, na construção, no trabalho, etc. Uma das aplicações que atrai a atenção dos pesquisadores na área da Robótica é a utilização de robôs na prevenção de desastres. Por esse motivo, nesta seção propomos a utilização de um robô para proteger uma população do transbordamento de um rio, que pode ser ocasionado, por exemplo, por fortes chuvas.

Objetivo do ambiente de transbordamento

O cenário da Figura 9.1 apresenta um sistema de proteção de uma população com relação ao transbordamento de um rio, utilizando um robô posicionado próximo ao rio e encarregado de: (i) construir barragens de contenção ao lado do rio, através de balizas que foram espalhadas na lateral direita do cenário e (ii) jogar terra na parte do rio que estiver transbordando. O cenário é uma matriz com 23 por 17 células, simétrica e dividida ao meio pelo rio, que é definido pela linha branca.

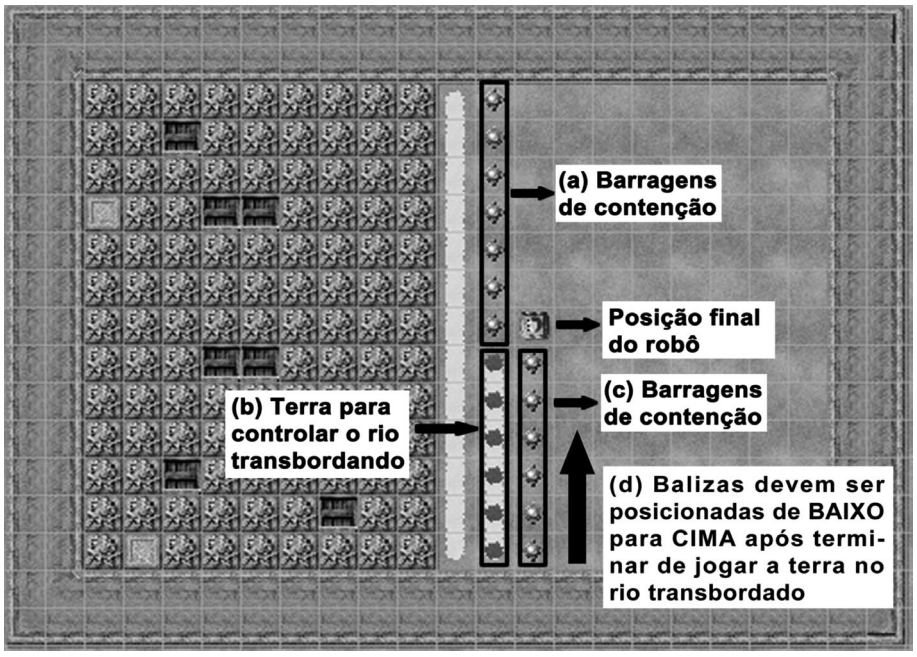
Figura 9.1: Ambiente inicial da simulação de transbordamento de um rio.



Para a construção das barragens, o robô deve posicionar as balizas nas células vizinhas ao rio, como pode ser visto no item (a) na Figura 9.2. O resultado da ação do robô, após jogar terra para controlar o rio transbordando, pode ser visto no item (b) na Figura 9.2. Por último, no item (c) desta figura observa-se o resultado final da ação do robô, após ter colocado balizas para fortalecer o controle do transbordamento. As balizas devem ser colocadas de baixo para cima, após o robô ter terminado de colocar terra na parte do rio que estiver transbordando (veja item (d) da Figura 9.2).

A construção das barragens tem como objetivo dar início à tarefa de prevenção. Com a alocação das barragens (balizas) sobre as células vizinhas às margens do rio, procura-se evitar que novos transbordamentos ocorram.

Figura 9.2: Ambiente final da simulação de transbordamento de um rio.



Modelagem de um ambiente de transbordamento

Para a criação do cenário de um resgate da Figura 9.1, no RoboMind será necessário criar um mapa. Para que isso seja possível, deve-se abrir o Bloco de Notas, ou qualquer editor de texto, e salvar o arquivo com extensão `.map` com os comandos definidos na Figura 9.3. O mapa criado pode ser aberto no menu `Arquivo⟶Abrir Mapa` no RoboMind.

Figura 9.3: Mapa do cenário do transbordamento da Figura 9.1.

```
paint:
{(w,|,11,2),(w,|,11,3),(w,|,11,4),(w,|,11,5),(w,|,11,6),(w,|,11,7),(w,|,11,8),
(w,|,11,9),(w,|,11,10),(w,|,11,11),(w,|,11,12),(w,|,11,13),(w,|,12,9),
(w,|,12,10),(w,|,12,11),(w,|,12,12),(w,|,12,13)}

map:
CHHHHHHHHHHHHHHHHHHHHD
GMFFFFFFFFFFFFFFFFFFJI
GIPPPPPPPPP @        *GI
GIPPQPPPPPP          *GI
GIPPPPPPPPP          *GI
GIOPPQQPPPP          *GI
GIPPPPPPPPP          *GI
GIPPPPPPPPP          *GI
GIPPPPPPPPP          *GI
GIPPPQQPPPP          *GI
GIPPPPPPPPP          *GI
GIPPPPPOPPP          *GI
GIPPQPPPPPP          *GI
GIPPPPPPQPP          *GI
GIPOPPPPPPP          *GI
GLHHHHHHHHHHHHHHHHHHKI
```

Na implementação de uma simulação utilizando o RoboMind é necessário definir o cenário. Para tanto, vamos modelar o sistema considerando os seguintes elementos:

1. A Vegetação da Floresta na Figura 9.1 é representada, no mapa da Figura 9.3, pela letra X;

2. Os Lagos na Figura 9.1 são representados pela letra O;

3. As Montanhas na Figura 9.1 são representadas pela letra Q;

4. As Barragens são indicadas pelo símbolo *;

5. A posição inicial do Robô é definida pelo símbolo @;

6. O Rio é representado pela linha em branco maior, e o transbordamento pela linha em branco menor;

7. A Terra é ilustrada pelo ponto preto.

O robô consegue identificar um transbordamento quando, no momento que ele estiver colocando as barragens de proteção, a sua direita tiver uma linha branca. Neste caso ele deve controlar o transbordamento com a barragem de proteção indicado no item (c) da Figura 9.2. Além disso, se o robô encontrar na sua esquerda o rio transbordado e controlado por terra, o mesmo deve ser reforçado com as barragens de proteção. Este cenário é indicado pelo item (d) da Figura 9.2.

A seguir tem-se a descrição das funcionalidades de cada elemento do modelo:

- Robô: percorre o cenário na busca da identificação do rio e do transbordamento. Quando ele encontra um rio, balizas são utilizadas para evitar o transbordamento. Caso necessário, o robô também usará terra para conter o transbordamento;

- Balizas: têm um comportamento estático e devem ser utilizadas pelo robô para evitar e controlar o transbordamento;

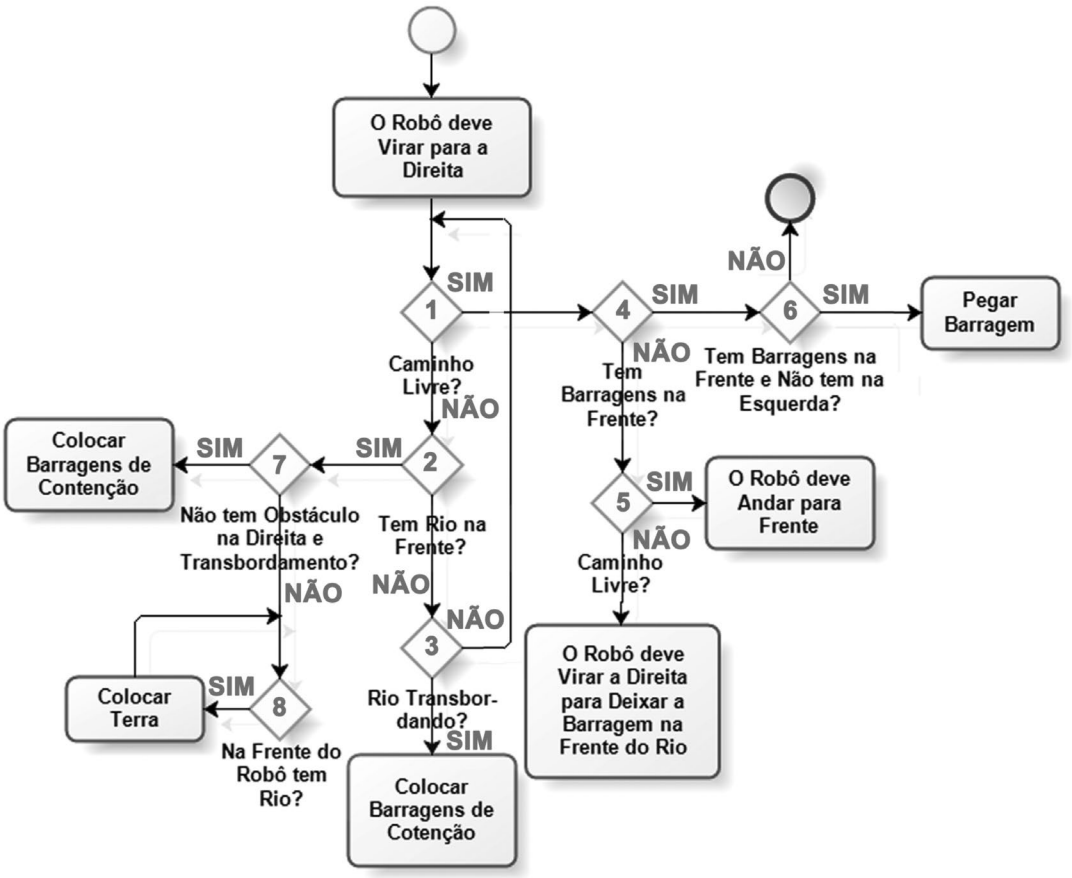
- Rio: representado pela linha branca maior, também apresenta um comportamento estático;

- Transbordamento: tem um comportamento estático e é representado pela linha branca menor.

- Floresta: é representada pelas árvores e também por lagos e montanhas.

O comportamento do robô que implementa o controle do transbordamento do rio pode ser observado na Listagem 9.1. Esse arquivo deve ser salvo no RoboMind com extensão `.irobo`. O fluxograma do algoritmo implementado pode ser visualizado na Figura 9.4.

Figura 9.4: Fluxograma da simulação do transbordamento de um rio.



Listagem 9.1: Implementação da simulação do transbordamento de um rio.

```
1  virarDireita()
2  repetir()
3  {
4     se(não brancoFrente() e  não brancoEsquerda()
         e não brancoDireita() e não pretoFrente())
5     {
6        se(temObjetoFrente())
7        {
8           se(temObjetoFrente() e não temObjetoEsquerda())
9           {
10             pegar()
11          }
12          senão
13          {
14             fim
15          }
16       }
17       senão se(não temObstáculoFrente())
18       {
19          andarFrente(1)
20       }
21       senão
22       {
23          virarDireita()
24          virarDireita()
25       }
26    }
27    senão se(brancoFrente())
28    {
29       se(temObstáculoDireita() e não brancoEsquerda())
30       {
31          andarTrás(1)
32          soltar()
33          virarEsquerda()
34          andarFrente(1)
35          virarEsquerda()
36       }
37       senão
38       {
39          andarTrás(1)
40          soltar()
41          virarEsquerda()
42          andarFrente(1)
43          virarDireita()
44          andarFrente(1)
45          virarEsquerda()
46
47          repetirEnquanto(brancoFrente())
48          {
```

```
49                pintarPreto()
50                pararPintar()
51                andarFrente(1)
52             }
53
54             pintarPreto()
55             pararPintar()
56             virarEsquerda()
57             andarFrente(1)
58          }
59       }
60       senão
61      {
62             andarTrás(1)
63             soltar()
64             virarDireita()
65             andarFrente(1)
66             virarDireita()
67             andarFrente(1)
68       }
69   }
```

Na Linha 1 o robô deve virar para a direita com o objetivo de pegar a primeira barragem de contenção. Na Linha 2 as instruções, representadas pelos losangos 1, 2 e 3 da Figura 9.4, são repetidas continuamente até o Robô evitar que o rio transborde e até conter o transbordamento.

O Robô deve andar para frente quando o caminho estiver livre, caso não tenha obstáculos como um rio na sua frente, à esquerda, à direita e em uma situação na qual o transbordamento foi controlado por terra. Entretanto, ele deve pegar uma barragem se estiver na sua frente e depois será necessário virar para a direita no sentido de ir em direção ao rio. Caso contrário, o transbordamento já foi evitado e a simulação é finalizada. Esse cenário é executado nas linhas 4-25 e representado pelos losangos 1, 4, 5 e 6.

Nas linhas 29-35, o Robô coloca as barragens no rio até encontrar o transbordamento. Nas linhas 39-45 o Robô deixa a última barragem no rio antes de começar a colocar terra para conter o transbordamento (linhas 47-52). Após essa tarefa ele é direcionado para pegar a barragem nas linhas 54-57. Finalmente as barragens são colocadas para realmente conter o transbordamento (linhas 60-69). Os losangos 2, 3, 7 e 8 representam essa situação.

Exercícios propostos

Com base no comportamento do robô implementado para o controle do transbordamento de um rio, responda como será o comportamento do robô para cada item a seguir:

1. Caso haja um aumento do tamanho do rio, do transbordamento e a quantidade de balizas, como o robô se comportará? Conseguirá conter o transbordamento? Justifique sua resposta;

2. Caso haja um aumento na distância entre o rio e as balizas, como o robô se comportará? Conseguirá conter o transbordamento? Justifique sua resposta;

3. E se tirarmos o transbordamento, como o robô se comportará? Justifique sua resposta;

4. A floresta tem um papel importante no cenário? Justifique sua resposta;

5. É possível melhorar o algoritmo? Justifique sua resposta.

9.3.2 SIMULAÇÃO DA TRAJETÓRIA DE UM PROJÉTIL

Você provavelmente já ouviu falar do famoso jogo de computador chamado *The Angry Birds*. É um jogo no qual o objetivo é acertar um conjunto de alvos usando um pássaro como projétil. Para esta finalidade, o jogador deve se valer de uma boa pontaria usando apenas o *mouse*. É possível ajustar a velocidade inicial do pássaro e o ângulo de lançamento, usando o *mouse* ou mesmo as mãos em telas sensíveis ao toque (*touch screen*). Nesta prática vamos simular o lançamento de um projétil usando conceitos muito simples de Física, implementando o nosso modelo no Scilab.

O lançamento de um projétil pode ser feito (desprezando a resistência do ar) a partir da decomposição do vetor velocidade inicial $\vec{v}_0$ em uma componente vertical e uma componente horizontal. Se o vetor $\vec{v}_0$ faz um ângulo θ com o semi-eixo horizontal positivo, teremos um componente vertical de módulo $v_v = v_0 \text{sen}(\theta)$ e um componente horizontal de módulo $v_h = v_0 \cos(\theta)$, onde v_0 é o módulo do vetor velocidade inicial $\vec{v}_0$.

A decomposição do vetor inicial, em componentes horizontal e vertical, é útil para nos permitir fazer uma análise independentemente do deslocamento horizontal e vertical do projétil. Na prática, podemos calcular separadamente as posições horizontal e vertical do projétil em cada instante do voo. A posição horizontal do projétil no instante `t` é dada pela Equação 9.1.

$$x - x_0 = (v_0 \cos\theta)t \tag{9.1}$$

Já a posição vertical é dada pela Equação 9.2.

$$y - y_0 = (v_0 \,\text{sen}\theta)t - \frac{1}{2} gt^2 \tag{9.2}$$

Analisando a Equação 9.1, vemos que a posição horizontal depende somente da velocidade horizontal inicial. Por exemplo, se você anda a 10m/s, depois de 1 segundo você terá andando 10 metros. É exatamente isto o que nos diz a Equação 9.1, onde `v0cos(θ)` é o componente horizontal da velocidade inicial e `T` é o tempo decorrido. O produto dos dois nos dá diretamente a distância horizontal percorrida pelo projétil.

A Equação 9.2 apresenta um segundo termo além do termo equivalente aquele da Equação 9.1. O movimento vertical do projétil está submetido à aceleração da gravidade (aproximadamente *9,81m/s²*). O segundo termo da Equação 9.2 subtrai da velocidade inicial uma velocidade referente à (des)aceleração da gravidade. Reescrevendo a Equação 9.2 temos a Equação 9.3.

$$y - y_0 = t\left[(v_0\, sen\theta) - \left(\frac{1}{2}gt\right)\right] \quad (9.3)$$

A Equação 9.3 evidencia o fato que $v_0 sen(\theta)$ é uma velocidade vertical e *½·gt* também é uma velocidade vertical, mas no sentido contrário ao sentido da velocidade inicial.

Rearranjando as equações podemos calcular o alcance horizontal R, através da Equação 9.4.

$$R = \frac{v_0^2}{g} sen2\theta \quad (9.4)$$

A Equação 9.5 calcula o tempo t de voo até que o projétil atinja a altura de lançamento.

$$t = \frac{2v_0 sen\theta}{g} \quad (9.5)$$

Com essas equações, estamos prontos para iniciar a construção do nosso modelo.

Construção do modelo computacional

Os comandos da Listagem 9.2 devem ser digitados no Scilab.

Listagem 9.2: Modelo computacional para o lançamento de um projétil.

```
1  teta=30;
2  tf=10;
3  g=9.81;
4  v0=35;
5  passos=50;
6
7  R = (v0*v0/g)*sind(2*teta);
8  printf("Alcance horizontal: %.2f m", R);
9  t = 2*v0*sind(teta)/g;
10 printf("Tempo de voo: %.2f s", t);
11 dt=t/passos;
12
13 x=zeros(passos);
14 y=zeros(passos);
15
16 cont=1;
17 for t=0 : dt : t
18    x(cont) = (v0*cosd(teta))*t;
19    y(cont) = ((v0*sind(teta))*t) - (0.5*g*t*t);
20    cont = cont+1;
21 end
22
23 plot(x,y, 'bo');
```

Nestes comandos, as linhas 1-5 fazem as declarações de variáveis: ângulo de lançamento (θ, em graus), aceleração da gravidade (g em m/s^2), velocidade inicial (v_0 em m/s) e o número de passos em que a trajetória do projétil vai ser calculada.

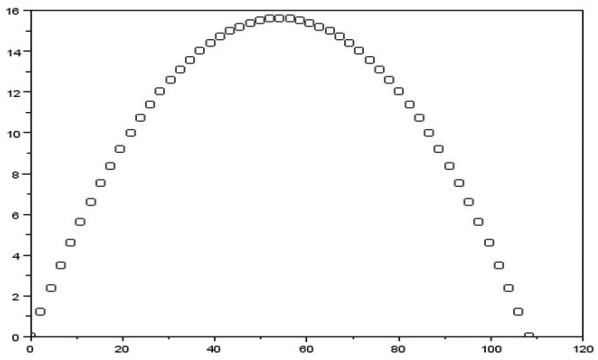
As linhas 7 e 9 calculam, respectivamente, o alcance horizontal do projétil e o tempo total de voo. O tempo de voo é dividido pelo número de passos, visando definir o intervalo no qual a análise de tempo vai ser discretizada (`Δt`, Linha 11).

As linhas 13 e 14 declaram dois vetores, `x` e `y`, de tamanho igual ao número de passos. A Linha 17 inicia uma estrutura de repetição que vai do instante de tempo inicial (`T=0`) até o instante de tempo final `T` igual ao tempo de voo, andando em passos de `Δt`. Para cada um destes passos, as coordenadas `x` e `y` do projétil são calculadas com as equações 9.1 e 9.2, respectivamente, e guardadas nos respectivos vetores. Finalmente, a Linha 23 mostra a trajetória (bidimensional) descrita pelo trajeto.

Simulando o lançamento de um projétil

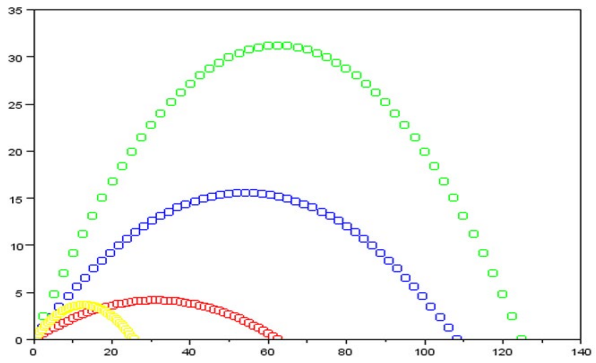
Uma vez implementado o nosso modelo computacional para o lançamento de um projétil na ausência de resistência do ar, vamos simular alguns lançamentos. A Figura 9.5 mostra a simulação com a trajetória do projétil disparado, considerando um movimento bidimensional. Observe que o tempo que o projétil leva para atingir a altura do lançamento também é calculado e mostrado nos nossos comandos. Outra observação importante é que, de acordo com o modelo físico, a trajetória do projétil é independente da sua massa!

Figura 9.5: Lançamento de um projétil.



Simulando o lançamento com uma velocidade inicial de 35 m/s (mais de 120km/h!!) temos que ângulos iniciais de 30^0, 15^0 e 45^0 resultam nas curvas mostradas em azul, vermelho e verde, respectivamente, da Figura 9.6. Isto é consistente com a Equação 9.1, onde o deslocamento depende do cosseno do ângulo de lançamento (cosseno de 90^0 é zero).

Figura 9.6: Lançamento de projéteis em diferentes condições iniciais.



A velocidade inicial também é importante para determinarmos o alcance horizontal. Sabemos, também intuitivamente, que quanto mais força utilizamos no lançamento, maior o alcance do projétil lançado. Isto está demonstrado na curva em amarelo da Figura 9.6. Nesta simulação, o ângulo de lançamento foi mantido em 30^0, mas a velocidade inicial foi reduzida de 35 m/s para 17 m/s (≈61km/h, mais realista!). Vemos que o alcance é reduzido de ≈110 m para algo em torno de 27 m.

Acertando o projétil no alvo com o Scilab

Com algumas pequenas modificações é possível criar um jogo no qual o objetivo do jogador é acertar o alvo (um círculo mostrado em cor preta), ajustando apenas o ângulo de lançamento e a velocidade inicial do projétil. Os comandos estão mostrados na Listagem 9.3.

Listagem 9.3: Comandos para acertar o alvo a partir do lançamento do projétil.

```
1  xdel;
2  alvox=rand()*100;
3  alvoy=0;
4  erro=0.1;
5  plot(alvox, alvoy, 'ko');
6  printf("Seu objetivo e acertar o circulo preto.\n");
7  printf("A aventura comeca agora!\n");
8  g=9.81;
9  passos=50;
10
11 acertou=%f;
12 tentativa=0;
13
14 while acertou == %f,
15     tentativa=tentativa+1;
16     teta= input("Entre com o angulo de ataque: ");
17     v0=input("Entre com a velocidade inicial (km/h):");
18     v0=v0/3.6; // converte para m/s
19     printf("Lancando projetil a %.2f m/s a %d graus.\n",
               v0, teta);
```

```
20      R = (v0*v0/g)*sind(2*teta);
21      tt = 2*v0*sind(teta)/g;
22      dt=tt/passos;
23      x=zeros(passos);
24      y=zeros(passos);
25      cont=1;
26      for t=0 : dt : tt
27          x(cont) = (v0*cosd(teta))*t;
28          y(cont) = ((v0*sind(teta))*t) - (0.5*g*t*t);
29          cont = cont+1;
30      end
31      plot(x,y, 'yo');
32     if (R>(alvox*(1.0-erro))) & (R<(alvox*(1.0+erro))) then
33          printf("Voce acertou o alvo com %d tentativas!\n",
                    tentativa);
34        printf("O tempo de voo do projetil foi de %.2s\n",tt);
35          acertou=%t;
36      else
37          printf("Voce errou o alvo.");
38          repete=input("Pressione r para repetir ou s
                    para sair: ", "string");
39          if repete == "s"w then
40              acertou=%t
41          end
42          xdel;
43          plot(alvox, alvoy, 'ko');
44      end
45  end
```

Estes comandos têm poucas diferenças em relação aos comandos da Listagem 9.2. No início, as variáveis ALVOX e ALVOY são definidas. Estas variáveis contêm as coordenadas x e y do alvo. Observe que a variável y é sempre zero (o alvo está sempre sobre o eixo x). Já a variável x tem um valor que varia entre 0 e 100, dado por um número aleatório sorteado pelo Scilab (comando RAND()). A variável ERRO define uma certa tolerância em relação ao acerto. Assim, se o alvo está na posição x=100 metros e o erro está definido como 0,1 (10%), lançamentos que acertem coordenadas entre 90 e 110 metros serão consideradas acerto. As linhas 16 e 17 fazem uma entrada de dados interativa, isto é, o usuário entra com os valores de θ e v_0 durante a execução do programa. Note que a entrada de v_0 é feita em km/h e posteriormente (Linha 18) convertida para m/s.

A Linha 32 estabelece o critério de acerto, de acordo com a variável ERRO definida anteriormente. Se o usuário acerta o alvo, o número de tentativas necessárias para o acerto é mostrado (Linha 33). Se o lançamento não acertou o alvo, o usuário tem a oportunidade de repetir ou abandonar. Acertando ou errando, a trajetória do projétil é mostrada juntamente com o alvo na janela de gráficos do Scilab.

Exercícios propostos

Digite os comandos da Listagem 9.3 no Scinotes e salve o arquivo. Em seguida, execute-o. Quantos lançamentos foram necessários até que você acertasse o alvo? Lembre-se: a prática (e, neste caso, a Física) leva à perfeição!

9.4 CONSIDERAÇÕES FINAIS

Conforme visto nos exemplos discutidos nesse capítulo, hoje a simulação computacional pode ser considerada fundamental para o desenvolvimento científico e tecnológico em geral. As simulações executadas, embora representem modelos simples, já demonstram o seu potencial como ferramenta para auxiliar no entendimento de diversos aspectos dos sistemas estudados.

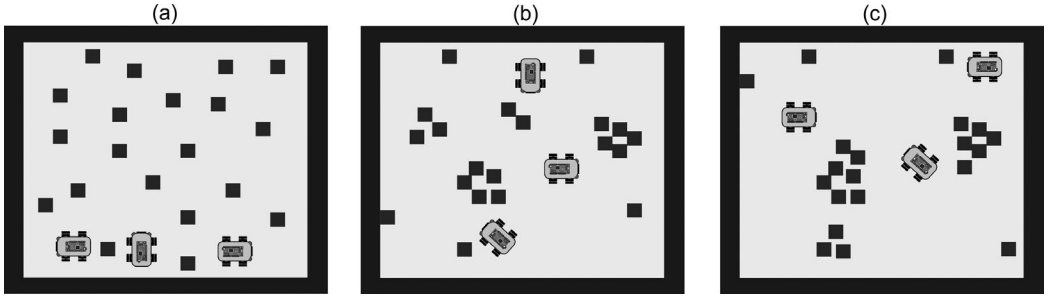
Por esse motivo, consideramos essencial que os futuros profissionais tenham contato com esse novo paradigma, e saibam como utilizar essa ferramenta poderosa e altamente versátil.

9.5 EXERCÍCIOS

9.5.1 SIMULANDO AGRUPAMENTO DE CUBOS NO ROBOMIND

Imagine que você tenha que desenvolver o sistema multi-robótico ilustrado na Figura 9.7, capaz de agrupar vários cubos espalhados em uma arena. Como será esse sistema? Será necessário utilizar processamento de imagens para que o robô possa identificar cada cubo, e também localizar-se no ambiente? Algum algoritmo muito complexo deve ser implementado para que o robô possa navegar pelo ambiente? E com relação à parte física do robô? Será necessário desenvolver um robô com vários sensores, atuadores, câmeras, etc.? O desafio parece ser complicado, não é mesmo? Entretanto, para Maris e te Boekhorst (1996) a resposta é NÃO e a solução é simples, mas muito eficiente. Esses pesquisadores exploraram a estrutura física do robô, chamado de DidaBot, e a sua capacidade de auto-organização em um trabalho em grupo.

Figura 9.7: Experimento do agrupamento de cubos.



O cenário proposto para o experimento contém cubos distribuídos de forma aleatória em uma arena com três DidaBots (normalmente são de 3 à 5 robôs), inicialmente definidos em qualquer posição, como ilustra a Figura 9.7. A ideia principal dos experimentos é a de que comportamentos aparentemente complexos podem ser gerados a partir de um conjunto limitado, mas eficiente de regras simples. Tais regras orientam as iterações entre os robôs e o ambiente onde estão inseridos.

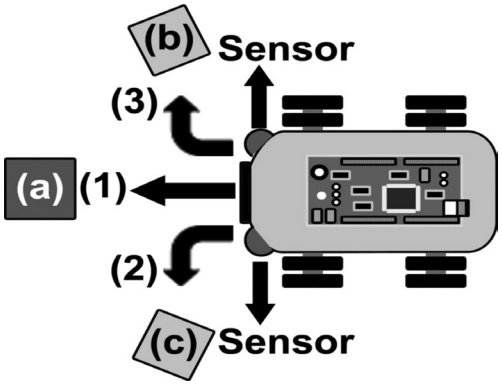
Após a definição da quantidade de cubos, robôs e também a posição inicial de cada elemento do cenário, o experimento pode ser inicializado. Observa-se que na medida em que os três DidaBots estão se movimentando no ambiente, três grupos de cubos estão sendo formados e podem ser visualizados no item (b) da Figura 9.7. No item (c) da Figura 9.7 pode-se observar que somente dois grupos foram formados, e que alguns cubos estão separadamente espalhados na arena.

Com relação à estrutura do DidaBot, ele possui um sensor infravermelho localizado no seu lado direito e o segundo no lado esquerdo, conforme indicado na Figura 9.8. A partir dos sensores é possível medir a proximidade de objetos e outros robôs no ambiente. Este tipo de sensor tem um alcance relativamente curto, na ordem de 5 cm, e tem como principal função verificar a proximidade de algum objeto. Com essas informações pode-se concluir que, cada vez que o robô estiver próximo de um objeto, o valor de retorno do sensor infravermelho será alto, caso contrário será baixo ou zero (Maris e te Boekhorst, 1996).

Com a estrutura definida, pode-se observar que o robô não tem a função de perceber objetos que podem aparecer à sua frente. Sendo assim, o objetivo principal é desviar dos obstáculos (que serão detectados pelos sensores) que podem surgir do seu lado direito e/ou esquerdo. Se o robô encontrar o objeto (a) da Figura 9.8, o seu movimento será de acordo com a seta (1). Entretanto, se o objeto for o indicado por (b), a seta que define o seu movimento é a (2). A mesma ideia pode ser observada caso o objeto (c) seja detectado pelo sensor do lado esquerdo do robô. Neste caso o seu movimento será definido pela seta (3). O algoritmo de controle do robô pode ser definido por dois passos simples, descritos a seguir:

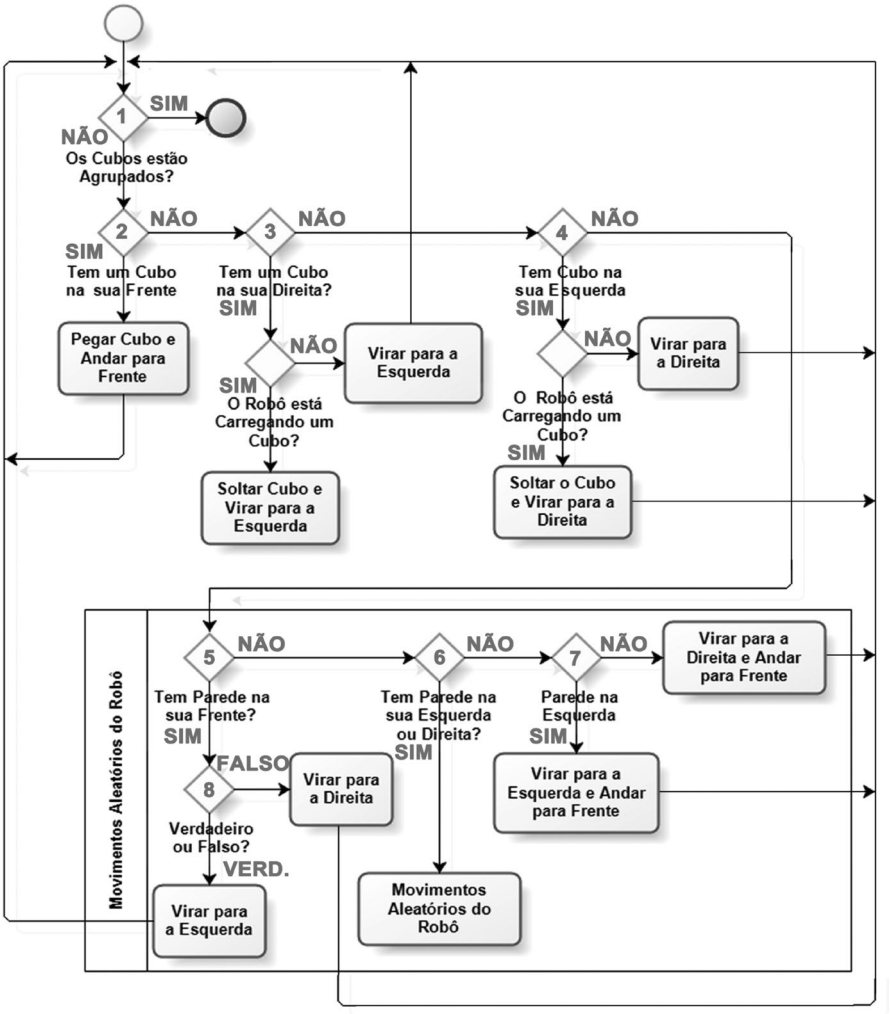
- O sensor do lado esquerdo percebeu um objeto: o robô deve virar para o lado direito;
- O sensor do lado direito percebeu um objeto: o robô deve virar para o lado esquerdo.

Figura 9.8: Estrutura e as possíveis movimentações do DidaBot.



A Figura 9.9 apresenta o fluxograma do algoritmo do experimento DidaBots.

Figura 9.9: Fluxograma da simulação do agrupamento de cubos DidaBots.



O código fonte implementado no RoboMind pode ser observado na Listagem 9.4.

Listagem 9.4: Comandos da simulação do DidaBots no RoboMind.

```
repetir()
{
   se(temObjetoFrente())
   {
      pegar()

      se(sortear())
      {
         virarDireita()
      }
      senão
      {
         virarEsquerda()
      }
   }
  senão se (temObjetoDireita() e não  temObstáculoEsquerda())
   {
      soltar()
      virarEsquerda()
      andarFrente(1)
   }
  senão se(temObjetoEsquerda() e não temObstáculoDireita())
   {
      soltar()
      virarDireita()
      andarFrente(1)
   }
   senão
   {
      se(temObstáculoFrente() e não temObjetoFrente())
      {
         se(sortear())
         {
            virarDireita()
         }
         senão
         {
            virarEsquerda()
         }
      }
      senão se((temObstáculoEsquerda() ou
          temObstáculoDireita()) e (não temObjetoEsquerda()
             e não temObjetoDireita()))
      {
         se(temObstáculoEsquerda())
         {
            virarDireita()
```

```
46              andarFrente(1)
47           }
48           senão
49           {
50              virarEsquerda()
51              andarFrente(1)
52           }
53        }
54        senão
55        {
56           se(sortear())
57           {
58              andarFrente(1)
59           }
60           senão
61           {
62              se(sortear())
63              {
64                 virarDireita()
65                 andarFrente(1)
66              }
67              senão
68              {
69                 virarEsquerda()
70                 andarFrente(1)
71              }
72           }
73        }
74     }
75  }
```

O ambiente inicial da simulação DidaBots pode ser visualizado na Figura 9.10. O código fonte no RoboMind, do mapa referente a este ambiente, está ilustrado na Figura 9.11.

Figura 9.10: Ambiente inicial da simulação do DidaBot.

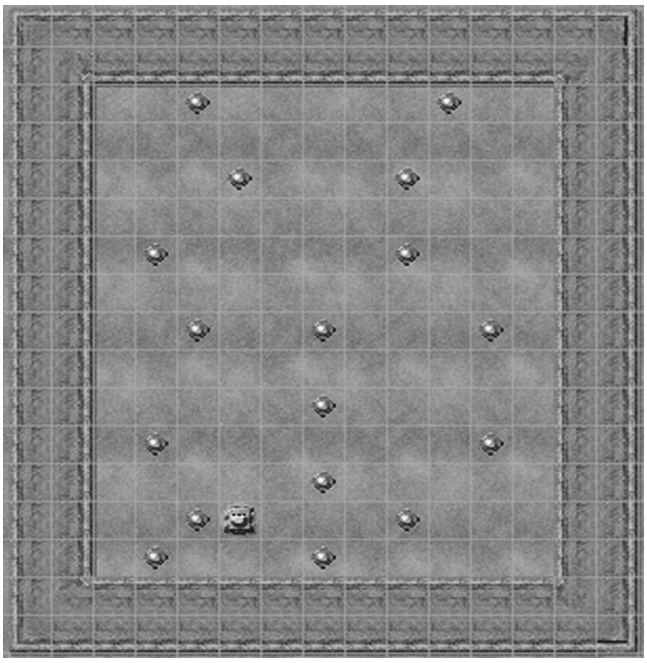


Figura 9.11: Mapa do cenário do DidaBots da Figura 9.10.

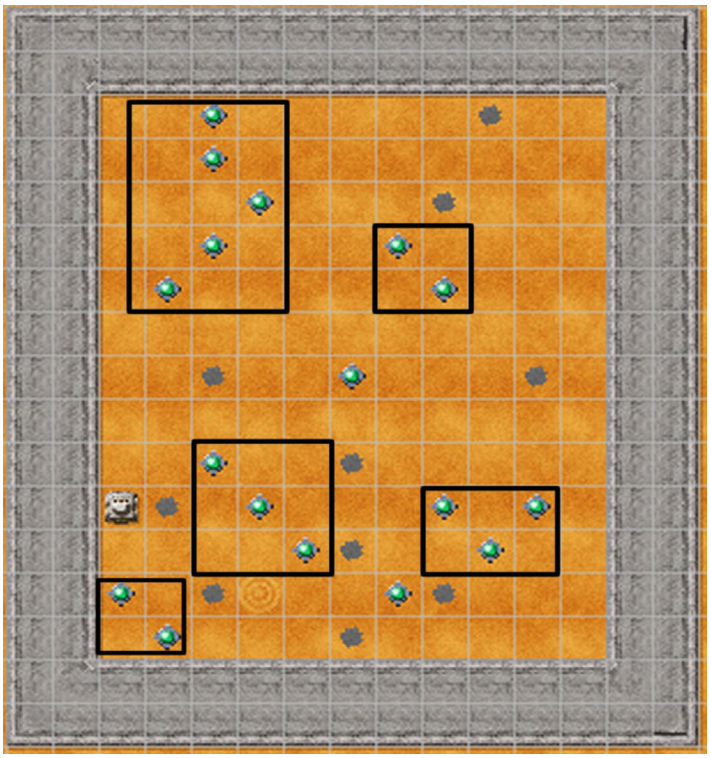
```
paint:
{(b,.,4,2)(b,.,10,2)(b,.,5,4)(b,.,9,4)(b,.,3,6)(b,.,9,6)(b,.,4,8)(b,.,7,8)
(b,.,11,8)(b,.,7,10)(b,.,3,11)(b,.,11,11)(b,.,7,12)(b,.,4,13)(b,.,9,13)
(b,.,3,14)(b,.,7,14)}

map:
CHHHHHHHHHHHHD
GMFFFFFFFFFFJI
GI  *     *  GI
GI           GI
GI   *   *   GI
GI           GI
GI *     *   GI
GI           GI
GI  *  *   * GI
GI           GI
GI     *     GI
GI *       * GI
GI     *     GI
GI  *@   *   GI
GI *   *     GI
GLHHHHHHHHHHHKI
BFFFFFFFFFFFFFE
```

É importante ressaltar que não há um comando para parar a simulação do agrupamento de cubos. A execução do programa deve ser interrompida pelo usuário, quando este considerar que os grupos de cubos já estão criados. Esta condição está definida em (1) na Figura 9.9. Esta participação direta do usuário na simulação é uma técnica utilizada na área de Simulação Computacional. Isto ocorre quando usuários especialistas do domínio da simulação (denominados *stakeholders*) participam no processo de validação para avaliarem a proximidade dos resultados obtidos pela execução do modelo conceitual com o comportamento real do sistema. Sendo assim, como o término da simulação é definido pelo usuário, um possível resultado final da

simulação com os agrupamentos está ilustrado na Figura 9.12. Neste ambiente final os pontos pretos no cenário representam as posições iniciais das balizas.

Figura 9.12: Um possível ambiente final da simulação do DidaBots.



As condições definidas para o robô perceber um cubo do seu lado direito, ou esquerdo, estão definidas em (3) e (4) da Figura 9.9, e nas linhas 16-22 e 23-27 da Listagem 9.4, respectivamente. Entretanto, para evitar que o robô fique preso na parede do cenário, é necessário implementar movimentos aleatórios e definir algumas condições para que isso não aconteça. Para estes casos, as condições (5), (6), (7) e (8) foram definidas e implementadas nas linhas 28-74.

A seguir têm-se atividades a serem realizadas, relacionadas à simulação DidaBots de agrupamentos de cubos:

1. É possível melhorar a ideia implementada nas linhas 28-74? Se sim, implemente um novo algoritmo, faça o teste no RoboMind e justifique a sua resposta;

2. Após executar e entender o algoritmo implementado na Listagem 9.4, modifique o cenário para verificar e explicar o comportamento da simulação para os seguintes casos:

- Aumentando o tamanho da arena, mantendo a mesma quantidade de cubos e a posição inicial do robô;
- Aumentar a quantidade de cubos, mantendo o tamanho da arena e alterar a posição inicial do robô;
- O que você pode concluir se relacionarmos o tamanho da arena, a quantidade de cubos e a posição inicial do robô?

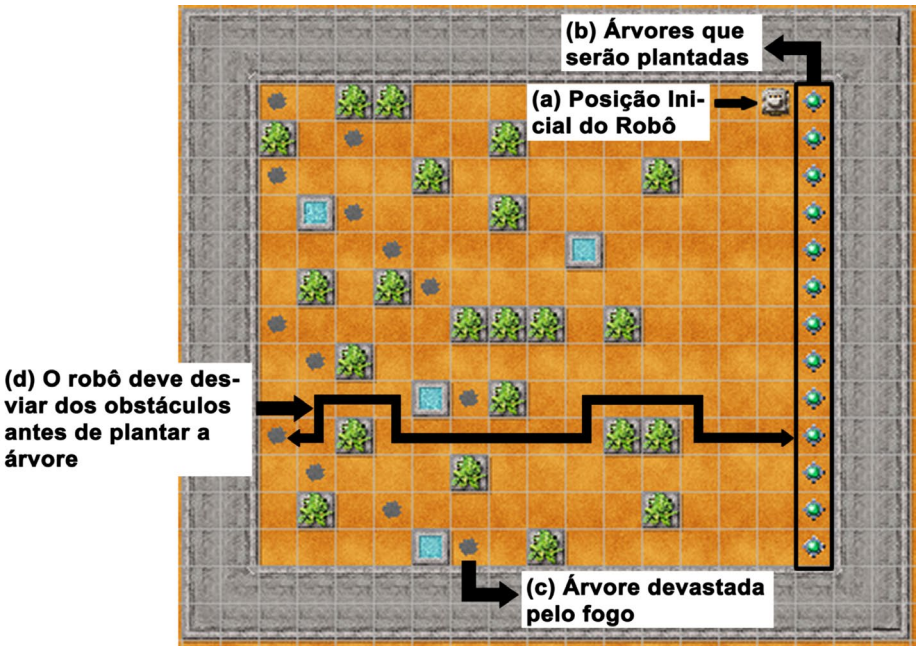
• O posicionamento inicial dos cubos e do robô pode influenciar no resultado final? Justifique a sua resposta;

• Se você executar o mesmo experimento várias vezes com o mesmo algoritmo, a quantidade de grupos formados será igual? A posição final de cada grupo será a mesma? A quantidade de grupos formados será a mesma? O tempo de agrupar será o mesmo? Justifique as suas respostas.

9.5.2 SIMULAÇÃO DE UM REFLORESTAMENTO

O cenário da Figura 9.13 é uma matriz com 19 por 17 células, e propõe o reflorestamento de um local em que havia uma floresta que foi devastada por queimadas ilegais. Para que o reflorestamento seja possível, é necessário desenvolver um robô encarregado de plantar árvores nos lugares onde houve a queimada. O robô deve pegar as árvores localizadas na lateral direita do cenário e levá-las até as árvores destruídas pelo fogo. Após o reflorestamento de todas as árvores, o robô deve terminar seu trabalho dirigindo-se ao ponto branco, localizado no canto inferior direito do cenário, e permanecer parado (terminando, assim, a simulação).

Figura 9.13: Ambiente inicial da simulação do reflorestamento de uma floresta.



Com base no cenário da Figura 9.13 pede-se que sejam realizadas as seguintes atividades:

1. Construir o mapa da Figura 9.15 (adaptar o modelo do mapa `OPENAREA.MAP`, disponível no diretório `map` no RoboMind);

2. Implementar no RoboMind a simulação do reflorestamento de um local onde houve uma floresta, utilizando estruturas sequenciais, condicionais, de repetição,

bem como demais conhecimentos de programação e de modelagem e simulação aprendidos neste livro. As seguintes informações devem ser consideradas:

(a) Deve-se começar a programar utilizando o código da Listagem 9.5;

(b) A posição inicial do robô está indicada no item (a) da Figura 9.13;

(c) Cada árvore (baliza), indicada no item (b) da Figura 9.13, deve ser plantada pelo robô nos locais onde as árvores foram devastadas pelo fogo. Os locais que indicam que uma árvore foi queimada estão definidos por pontos pretos, conforme ilustrado no item (c) da Figura 9.13;

(d) O robô deve carregar as árvores (balizas) até cada ponto preto, conforme indica o item (d) da Figura 9.13;

(e) Após o robô realizar as tarefas descritas anteriormente, o sistema deve parar utilizando o comando `FIM`. A posição final do robô é indicada no item (a) da Figura 9.14.

3. Faça o fluxograma da simulação que implementou, relacionada ao reflorestamento da floresta;

4. Responda às seguintes perguntas, após a implementação:

(a) O que acontece se colocar mais obstáculos no cenário? Justifique sua resposta;

(b) O programa funciona se aumentar a quantidade de árvores a serem plantadas, e as árvores devastadas?

OBS: O robô deve ser autônomo, no sentido de que deve conseguir chegar à sua meta mesmo SEM comandos específicos como, por exemplo, `andarNorte(12)`.

Figura 9.14: Ambiente final da simulação do reflorestamento de uma floresta.

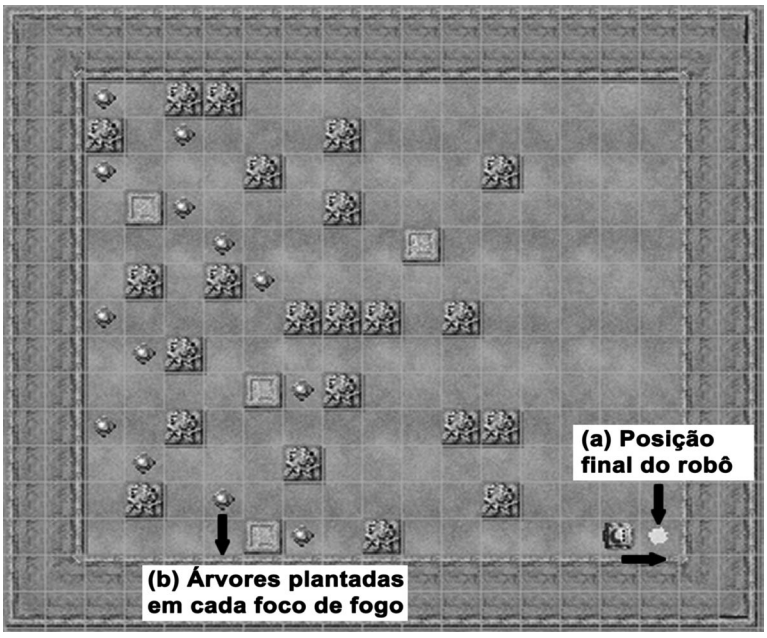


Figura 9.15: Mapa do cenário do reflorestamento da Figura 9.13.

```
paint:
{(b,.,2,2),(b,.,4,3),(b,.,2,4),(b,.,4,5),(b,.,5,6),(b,.,6,7),(b,.,2,8),(b,.,3,9),
(b,.,7,10)),(b,.,2,11),(b,.,3,12),(b,.,5,13),(b,.,7,14),(w,.,16,14)}

map:
CHHHHHHHHHHHHHHHHD
GMFFFFFFFFFFFFFFJI
GI  PP         @*GI
GIP     P       *GI
GI    P     P   *GI
GI O    P       *GI
GI        O     *GI
GI P P          *GI
GI     PPP P    *GI
GI  P           *GI
GI    O P       *GI
GI  P       PP  *GI
GI     P        *GI
GI P          P  *GI
GI    O  P      *GI
GLHHHHHHHHHHHHHHHKI
BFFFFFFFFFFFFFFFFE
```

Listagem 9.5: Estrutura geral da implementação da simulação do reflorestamento no local em que havia uma floresta.

```
1  virarDireita()
2  repetir ()
3  {
4     #Inserir o código
5  }
```

REFERÊNCIAS BIBLIOGRÁFICAS

AXTELL, R. L.; EPSTEIN, J. M.; DEAN, J. S.; GUMERMAN, G. J.; SWEDLUND, A. C.; HARBURGER, J; CHAKRAVARTY, B.; HAMMOND, R.; PARKER, J. e PARKER, M. Population Growth and Collapse in a Multi-agent Model of the Kayenta Anasazi in Long House Valley. In: *Proceedings of the National Academy of Sciences of the United States of America*, v. 99, 2002, pp.7275–7279.

JANSSEN, M. A. "Understanding Artificial Anasazi", *Journal of Artificial Societies and Social Simulation*, v. 12, n. 4, 2009, Disponível em http://jasss.soc.surrey.ac.uk/12/4/13.html. Última visita em 28/02/2013.

MARIS, M. & TE BOEKHORST, R. Exploiting Physical Constraints: Heap Formation through Behavioral Error in a Group of Robots. In: *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems* IROS-96, 1996, pp.1655–1660.

PARKER, D. C.; MANSON, S. M.; JANSSEN, M. A.; HOFFMANN, M. J. & DEADMAN, P. Multi-agent Systems for the Simulation of Land-Use and Land-Cover Change: A Review. *Annals of the Association of American Geographers*, v. 93, n. 2, pp. 314-337, 2003.

RUAS, T. L.; MARIETTO, M. G. B.; MORAES BATISTA, A. F. de; FRANÇA, R. S.; HEIDEKER, A.; NORONHA, E. A. e DA SILVA, F. A. *Modeling Artificial Life Through Multi-Agent Based Simulation*. In: ALKHATEEB, F. MAGHAYREH, E. A. & DOUSH, I. A. (ed.).*Multi-Agent Systems* – Modeling, Control, Programming, Simulations and Applications. Rijeka, Croatia:InTech. pp 4166.

Memória Produção Editorial

1ª edição: Junho, 2013
formato: 21,0cm x 29,7cm | 242 p.
tipologia: Myriad Pro e CholllaSans
papel da capa: Supremo 250 g/m²
papel do miolo: Offset 75g/m²

produtora editorial: Maíra Nassif
capa & projeto gráfico: Ana C. Bahia
diagramação: Jamile Faller
Revisão: Erick Ramalho
Revisão de provas: Cláudia Rajão