TREINAMENTOS

Lógica de Programação

Lógica de Programação

23 de setembro de 2012

Sobre a K19

A K19 é uma empresa especializada na capacitação de desenvolvedores de software. Sua equipe é composta por profissionais formados em Ciência da Computação pela Universidade de São Paulo (USP) e que possuem vasta experiência em treinamento de profissionais para área de TI.

O principal objetivo da K19 é oferecer treinamentos de máxima qualidade e relacionados às principais tecnologias utilizadas pelas empresas. Através desses treinamentos, seus alunos se tornam capacitados para atuar no mercado de trabalho.

Visando a máxima qualidade, a K19 mantém as suas apostilas em constante renovação e melhoria, oferece instalações físicas apropriadas para o ensino e seus instrutores estão sempre atualizados didática e tecnicamente.

Seguro Treinamento

Na K19 o aluno faz o curso quantas vezes quiser!

Comprometida com o aprendizado e com a satisfação dos seus alunos, a K19 é a única que possui o Seguro Treinamento. Ao contratar um curso, o aluno poderá refazê-lo quantas vezes desejar mediante a disponibilidade de vagas e pagamento da franquia do Seguro Treinamento.

As vagas não preenchidas até um dia antes do início de uma turma da K19 serão destinadas ao alunos que desejam utilizar o Seguro Treinamento. O valor da franquia para utilizar o Seguro Treinamento é 10% do valor total do curso.

Termo de Uso

Termo de Uso

Todo o conteúdo desta apostila é propriedade da K19 Treinamentos. A apostila pode ser utilizada livremente para estudo pessoal . Além disso, este material didático pode ser utilizado como material de apoio em cursos de ensino superior desde que a instituição correspondente seja reconhecida pelo MEC (Ministério da Educação) e que a K19 seja citada explicitamente como proprietária do material.

É proibida qualquer utilização desse material que não se enquadre nas condições acima sem o prévio consentimento formal, por escrito, da K19 Treinamentos. O uso indevido está sujeito às medidas legais cabíveis.

Conheça os nossos cursos

K01- Lógica de Programação

K11 - Orientação a Objetos em Java

K12 - Desenvolvimento Web com JSF2 e JPA2

K21 - Persistência com JPA2 e Hibernate

K22 - Desenvolvimento Web Avançado com JFS2, EJB3.1 e CDI

K23 - Integração de Sistemas com Webservices, JMS e EJB

K31 - C# e Orientação a Objetos

K32 - Desenvolvimento Web com ASP.NET MVC

www.k19.com.br/cursos

INTRODUÇÃO

CAPÍTULO 1

O que é um Computador?

Hoje em dia, os computadores estão presentes no cotidiano de praticamente todas as pessoas. Você, provavelmente, está acostumado a utilizar computadores no seu dia a dia. Mas, será que você conhece o funcionamento básico de um computador? A seguir, listaremos os principais elementos de um computador e suas respectivas funções.

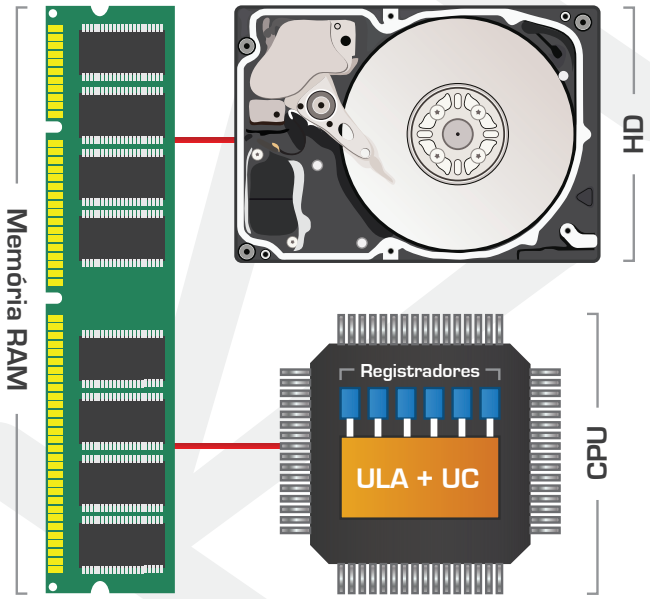

Figura 1.1: Principais elementos de um computador

CPU (Unidade Central de Processamento): A CPU é o "cérebro" que controla o funcionamento dos outros componentes do computador e realiza todo o processamento necessário. Esse processamento consiste basicamente na realização de operações matemáticas e lógicas.

Registradores: Os registradores armazenam os dados que estão sendo processados pela CPU. O acesso ao conteúdo dos registradores é extremamente rápido. Por outro lado, eles não possuem muito espaço. Dessa forma, não é possível armazenar neles uma grande quantidade de informação.

Memória RAM: Os dados utilizados pelos programas que estão sendo executados são armazenados na memória RAM. O acesso ao conteúdo da memória RAM é rápido porém mais lento do que o acesso ao conteúdo dos registradores. Por outro lado, o espaço da memória RAM é bem maior do que o espaço dos registradores.

Disco Rígido: Os dados armazenados nos registradores e na memória ram são descartados quando o computador é desligado. O conteúdo que não pode ser descartado ao desligar o computador deve ser armazenado no disco rígido. O acesso ao disco rígido é bem mais lento do que o acesso a memória ram mas, em geral, o espaço é bem maior.

Para resumir, podemos dizer que um computador é apenas uma máquina que executa comandos matemáticos e armazena dados. Os comandos que um computador executa e os dados que ele armazena são definidos em um formato não intuitivo para a maior parte das pessoas. Eles são definidos de forma binária.

Por exemplo, o número "19" pode ser armazenado da seguinte forma "00010011". O comando para realizar uma soma, na arquitetura x86 (http://en.wikipedia.org/wiki/X86), é definido através da sequência de bits "00000000".

Você deve estar se perguntando, como um computador consegue realizar tarefas tão sofisticadas se ele apenas executa comandos matemáticos ou lógicos e armazena dados? A resposta é simples. Os computadores são programados por pessoas e essas pessoas conseguem realizar tarefas sofisticadas a partir dos recursos básicos oferecidos pelos computadores. Da mesma forma que pessoas são capazes de produzir pinturas sofisticadas utilizando apenas tinta, pincel e quadro.

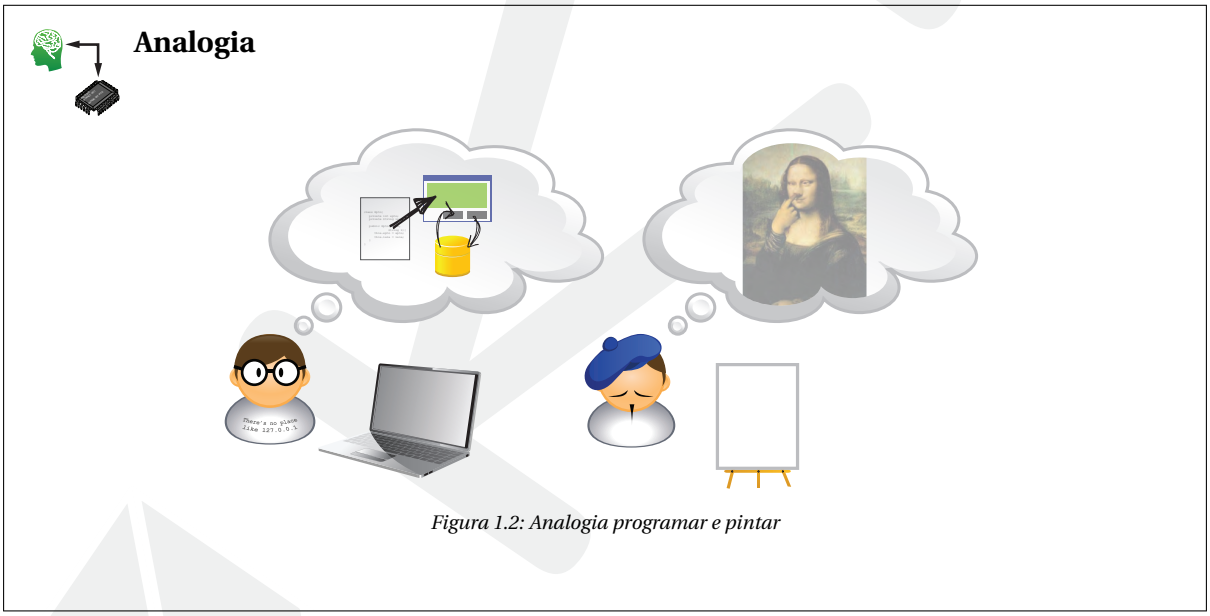

Figura 1.2: Analogia programar e pintar

O que é um Programa?

Um programa de computador é simplesmente uma sequência de comandos para um computador executar e realizar uma determinada tarefa. Esses comandos são armazenados em arquivos comumente chamados de *executáveis* e definidos em formato binário. Esse formato é extremamente difícil de ser lido por um ser humano.

Há diversos fabricantes de computadores que produzem máquinas com características diferentes. Essas diferenças vão desde a capacidade de armazenamento de dados até a estética do gabinete. Dentro de um cenário tão diversificado, é natural encontrarmos diversos tipos de proces-

sadores que não variam somente no poder de processamento, mas também em sua arquitetura. Essa variedade acontece também no mundo dos sistemas operacionais. Hoje temos o Windows, Linux e o Mac OS X entre os mais populares, mas ao longo dos últimos 20 anos tivemos também o OS/2, QNX, DOS, BeOS, entre outros.

E o que isso tem a ver com os programas que executamos em nossos computadores?

Tudo. Os comandos contidos em um programa são específicos para uma determinada arquitetura de processador e para um determinado sistema operacional. Dessa forma, um programa pode funcionar em um computador e não funcionar em outro devido a diferenças de arquitetura de processador ou de sistema operacional.

Linguagem de Máquina

O formato dos comandos que um computador pode executar depende da arquitetura do seu processador. Como existem diversas arquiteturas diferentes, um mesmo comando pode funcionar em um computador e não funcionar em outro. O formato dos comandos que funcionam em um determinado processador definem a **linguagem de máquina** ou **código de máquina** desse processador.

Comandos definidos em linguagem de máquina são perfeitos para os computadores processarem. Por outro lado, eles são extremamente complexos para as pessoas entenderem. Podemos verificar essa complexidade, observando o trecho de comandos em linguagem de máquina da figura abaixo:

```
00000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000010000000000100000000
00000000000011100001111110111010000011100000000010110100000010 0
11100110100100001101110000000000101001100110011010010000010101 01
00011010000110100101110011001000000111000001110010011011110 1100
11101110010011000010110110100100000011000110110000101101110011 0
11100110111101110100001000000110001001100101001000000111001 0011
```

Figura 1.3: Trecho do programa Chrome em binário

Linguagem de Programação

Devido a complexidade, escrever um programa em linguagem de máquina é inviável. Para tornar viável o desenvolvimento de programas, existem as **linguagens de programação** que tentam se aproximar das linguagens humanas. Confira um trecho de código escrito com a linguagem de programação Java:

```
1  while (lineMeasurer.getPosition() < paragraphEnd) {
2      TextLayout layout = lineMeasurer.nextLayout(formatWidth);
3      drawPosY += layout.getAscent();
4      float drawPosX;
5
6      if (layout.isLeftToRight()) {
7        drawPosX = 0;
8      } else {
9        drawPosX = formatWidth - layout.getAdvance();
10     }
11 }
```

Código Java 1.1: Exemplo de código em Java

Por enquanto, você não precisa se preocupar em entender o que está escrito no código acima. Apenas, observe que um programa escrito em linguagem de programação é bem mais fácil de ser lido do que um programa escrito em linguagem de máquina.

Compilador

Vimos que os computadores são capazes de processar o código escrito em linguagem de máquina. Também vimos que é inviável desenvolver um programa em linguagem de máquina. Por isso, existem as linguagens de programação. Daí surge uma pergunta: se os computadores entendem apenas comandos em linguagem de máquina, como eles podem executar código escrito em linguagem de programação?

Na verdade, os computadores não executam código escrito em linguagem de programação. Esse código que é denominado **código fonte** deve ser traduzido para código em linguagem de máquina. Essa tradução é realizada por programas especiais chamados **compiladores**.

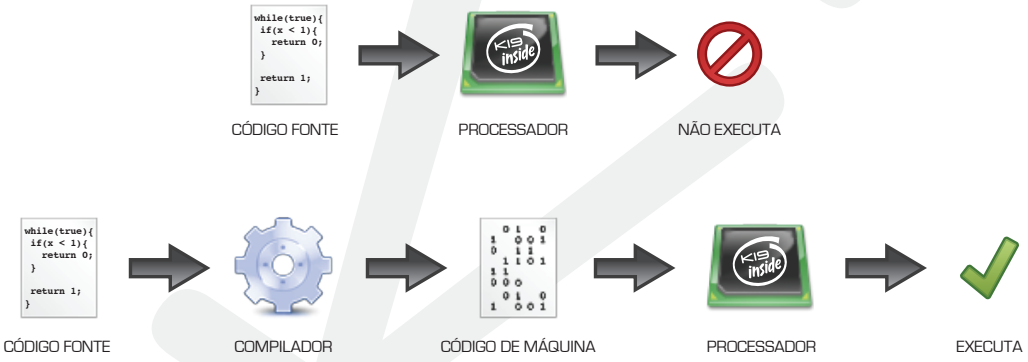

Figura 1.4: Processo de compilação e execução de um programa

Máquinas Virtuais

Como vimos anteriormente, o código fonte de um programa deve ser compilado para que esse programa possa ser executado por um computador. Além disso, vimos que os compiladores geram executáveis específicos para um determinado sistema operacional e uma determinada arquitetura de processador. Qual é o impacto disso para quem desenvolve sistemas para múltiplas plataformas?

A empresa que deseja ter uma aplicação disponível para diversos sistemas operacionais (Windows, Linux, Mac OS X, etc) e arquiteturas de processadores (Intel, ARM, PowerPC, etc) deverá desenvolver e manter um código fonte para cada plataforma (sistema operacional + arquitetura de processador). Consequentemente, os custos dessa empresa seriam altos.

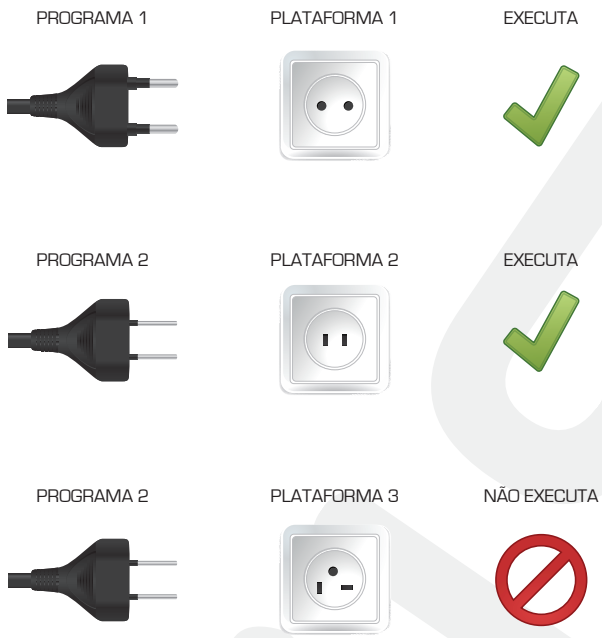

Figura 1.5: Ilustração monstrando que cada plataforma necessita de um executável específico.

Para diminuir os custos e aumentar a produtividade, podemos utilizar **máquinas virtuais**. Em um ambiente que utiliza máquina virtual, quando o código fonte é compilado, ele é traduzido para um código escrito na linguagem da máquina virtual. A linguagem da máquina virtual também pode ser considerada uma linguagem de máquina. Na execução, a máquina virtual traduz os comandos em linguagem de máquina virtual para comandos em linguagem de máquina correspondente à plataforma utilizada.

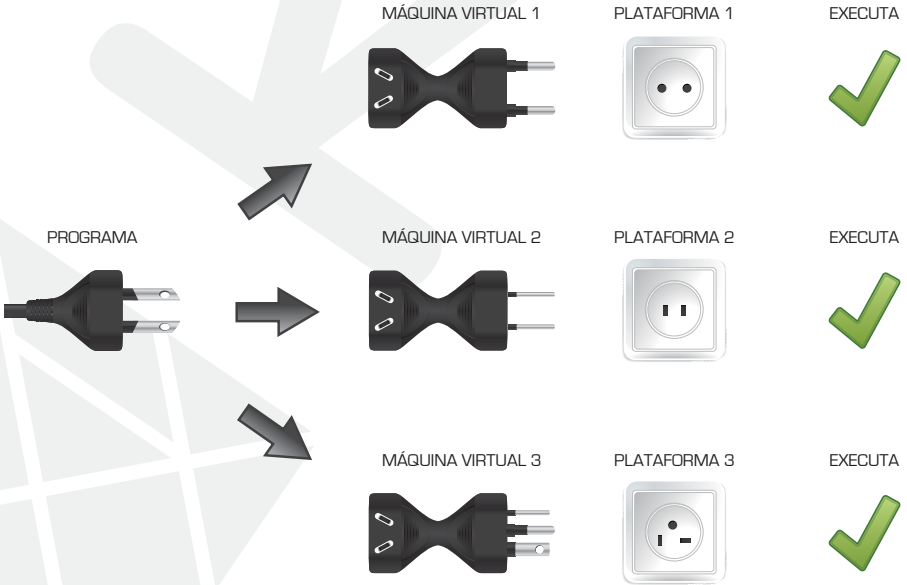

Figura 1.6: Ilustração do funcionamento da máquina virtual.

Tudo parece estar perfeito agora, porém, olhando atentamente a figura acima, percebemos que existe a necessidade de uma máquina virtual para cada plataforma. Alguém poderia dizer que, de

fato, não resolvemos o problema.

A diferença é que implementar a máquina virtual não é tarefa dos programadores que desenvolvem as aplicações que serão executados nas máquinas virtuais. A implementação da máquina virtual é responsabilidade de terceiros que geralmente são empresas bem conceituadas ou projetos de código aberto que envolvem programadores do mundo inteiro. Como maiores exemplos podemos citar a Oracle JVM (Java Virtual Machine), OpenJDK JVM, Microsoft CLR (Common Language Runtime) e Mono CLR.

A máquina virtual não funciona apenas como um mero adaptador. Ela normalmente traz recursos como o gerenciamento de memória, otimização do código em tempo de execução entre outros.

Hello World em Java

Vamos escrever o nosso primeiro programa para entendermos como funciona o processo de escrita de código fonte, compilação e execução de um programa.

Importante
Antes de compilar e executar um programa escrito em Java, é necessário que você tenha instalado e configurado em seu computador o JDK (Java Development Kit). Consulte o artigo da K19, http://www.k19.com.br/artigos/instalando-o-jdk-java-development-kit/.

Dentro de um editor de texto, escreva o seguinte código e salve o arquivo com o nome `HelloWorld.java`.

```
1 class HelloWorld {
2   public static void main(String[] args) {
3     System.out.println("Hello World");
4   }
5 }
```

Código Java 1.2: HelloWorld.java

Em seguida abra um terminal ou, no caso do Windows, o Prompt de Comando e entre na pasta em que você salvou o arquivo `HelloWorld.java`. Feito isso, digite o seguinte comando no terminal:

```
k19$ javac HelloWorld.java
```

Terminal 1.1: Compilando o arquivo HelloWorld.java

Esse comando compilará o arquivo `HelloWorld.java`. O programa `javac` é o compilador do Java. Após compilarmos o arquivo `HelloWorld.java`, nosso programa já estará pronto para ser executado. Porém, antes de executá-lo, digite no terminal o comando `ls` ou o comando `dir` no Prompt de Comando. Um arquivo chamado `HelloWorld.class` deverá aparecer na listagem de arquivos. Esse arquivo contém o código em linguagem de máquina virtual Java.

```
k19$ ls
HelloWorld.class    HelloWorld.java
```

Terminal 1.2: Listagem do diretório.

Agora vamos executar o nosso programa através do comando `java`:

```
k19$ java HelloWorld
Hello World
```

Terminal 1.3: Executando o programa HelloWorld.

Para executar o conteúdo do arquivo `HelloWorld.class`, a extensão `.class` não deve ser utilizada. Seguindo os passos acima, você terá um resultado semelhante ao mostrado abaixo:

```
k19$ cd Desktop/logica-de-programacao/
k19$ javac HelloWorld.java
k19$ ls
HelloWorld.class   HelloWorld.java
k19$ java HelloWorld
Hello World
```

Terminal 1.4: Compilação e execução do programa HelloWorld.

Hello World em C#

Agora, vamos utilizar outra linguagem de programação para criar o programa semelhante ao visto anteriormete.

Importante
Para compilar um programa escrito em C# é necessário ter o .NET Framework instalado em seu computador. As versões mais recentes do sistema operacional Windows já vêm com o framework instalado.

Se você utiliza os sistemas operacionais Linux ou Mac OS X, pode compilar e executar programas em C# utilizando a plataforma Mono (`http://www.mono-project.com/`).

Dentro de um editor de texto, escreva o seguinte código e salve o arquivo com o nome `HelloWorld.cs`.

```
1 class HelloWorld
2 {
3   static void Main()
4   {
5     System.Console.WriteLine("Hello World");
6   }
7 }
```

Código C# 1.1: HelloWorld.cs

Em seguida abra o Prompt de Comando do Windows e entre na pasta em que você salvou o arquivo `HelloWorld.cs`. Feito isso, digite o seguinte comando no Prompt de Comando:

```
C:\Users\K19\Desktop\logica-de-programacao>csc HelloWorld.cs
```

Terminal 1.5: Compilando o programa HelloWorld

Esse comando compilará o arquivo `HelloWorld.cs`. O programa `csc` é o compilador do C#. Após compilarmos o arquivo `HelloWorld.cs`, o programa estará pronto para ser executado. Porém, antes

de executá-lo, digite no Prompt de Comando o comando `dir`. Um arquivo chamado `HelloWorld.exe` deverá aparecer na listagem de arquivos. Esse arquivo é o executável gerado pelo compilador do C#.

```
C:\Users\K19\Desktop\logica-de-programacao>dir
 O volume na unidade C não tem nome.
 O Número de Série do Volume é 40EF-8653

 Pasta de C:\Users\K19\Desktop\logica-de-programacao

02/03/2011  21:07    <DIR>          .
02/03/2011  21:07    <DIR>          ..
02/03/2011  20:58                90 HelloWorld.cs
02/03/2011  21:07             3.584 HelloWorld.exe
               2 arquivo(s)          3.674 bytes
               2 pasta(s)   22.508.589.056 bytes disponíveis
```

Terminal 1.6: Listagem do diretório.

Agora vamos executar o nosso programa:

```
C:\Users\K19\Desktop\logica-de-programacao>HelloWorld.exe
Hello World
```

Terminal 1.7: Executando o programa HelloWorld

Seguindo os passos acima, você terá um resultado semelhante ao mostrado abaixo:

```
Microsoft Windows [versão 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. Todos os direitos reservados.

C:\Users\K19>cd Desktop\logica-de-programacao

C:\Users\K19\Desktop\logica-de-programacao>csc HelloWorld.cs
Compilador do Microsoft (R) Visual C# 2008 versão 3.5.30729.4926
 para Microsoft (R) .NET Framework versão 3.5
Copyright (C) Microsoft Corporation. Todos os direitos reservados.


C:\Users\K19\Desktop\logica-de-programacao>dir
 O volume na unidade C não tem nome.
 O Número de Série do Volume é 40EF-8653

 Pasta de C:\Users\K19\Desktop\logica-de-programacao

03/04/2011  16:50    <DIR>          .
03/04/2011  16:50    <DIR>          ..
02/04/2011  20:58                90 HelloWorld.cs
03/04/2011  16:50             3.584 HelloWorld.exe
               2 arquivo(s)          3.674 bytes
               2 pasta(s)   22.362.529.792 bytes disponíveis

C:\Users\K19\Desktop\logica-de-programacao>HelloWorld.exe
Hello World

C:\Users\K19\Desktop\logica-de-programacao>
```

Terminal 1.8: Compilação e execução do programa HelloWorld

Exercícios de Fixação

1. Abra um terminal e crie uma pasta com o seu nome. Você deve salvar os seus exercícios nessa pasta.

```
K19$ mkdir rafael
K19$ cd rafael
K19/rafael$
```

Terminal 1.9: Criando a pasta de exercícios

2 Dentro da sua pasta de exercícios, crie uma pasta para os arquivos desenvolvidos nesse capítulo chamada **introducao**.

```
K19/rafael$ mkdir introducao
K19/rafael$ ls
introducao
```

Terminal 1.10: Criando a pasta dos exercícios desse capítulo

3 Utilize um editor de texto e implemente um programa utilizando a linguagem programação Java. Crie um arquivo chamado **HelloWorld.java** na pasta `introducao`.

```
1 class HelloWorld {
2   public static void main(String[] args) {
3     System.out.println("Hello World");
4   }
5 }
```

Código Java 1.3: HelloWorld.java

4 Através do terminal, entre na pasta `introducao`; compile o arquivo `HelloWorld.java`; execute o programa.

```
K19/rafael/introducao$ ls
HelloWorld.java

K19/rafael/introducao$ javac HelloWorld.java

K19/rafael/introducao$ ls
HelloWorld.class HelloWorld.java

K19/rafael/introducao$ java HelloWorld
Hello World
```

Terminal 1.11: Compilando e Executando

5 Utilize um editor de texto e implemente um programa utilizando a linguagem programação C#. Crie um arquivo chamado **HelloWorld.cs** na pasta `introducao`.

```
1 class HelloWorld
2 {
3   static void Main()
4   {
5     System.Console.WriteLine("Hello World");
6   }
7 }
```

Código C# 1.2: HelloWorld.cs

6 Através do terminal, entre na pasta `introducao`; compile o arquivo `HelloWorld.cs`; execute o programa.

```
K19/rafael/introducao$ ls
HelloWorld.class HelloWorld.cs HelloWorld.java

K19/rafael/introducao$ mcs HelloWorld.cs

K19/rafael/introducao$ ls
HelloWorld.class HelloWorld.cs HelloWorld.exe HelloWorld.java

K19/rafael/introducao$ mono HelloWorld.exe
Hello World
```

Terminal 1.12: Compilando e Executando

Exercícios Complementares

1. Utilize a linguagem Java para implementar um programa que imprima duas mensagens em duas linhas. Compile e execute esse programa.

2. Utilize a linguagem C# para implementar um programa que imprima duas mensagens em duas linhas. Compile e execute esse programa.

CAPÍTULO 2

ALGORITMOS

O que é um Algoritmo?

Um *algoritmo* é uma sequência finita de instruções bem definidas que levam a execução de uma tarefa por um computador, um autômato ou até mesmo por um ser humano. Muitas vezes um algoritmo é comparado a uma receita de bolo, onde cada passo da confecção do bolo seria representado pelas instruções do algoritmo.

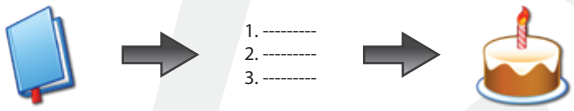

Figura 2.1: Confecção de um bolo seguindo os passos de uma receita (algoritmo).

Um algoritmo pode ter a sua corretude, assim como a quantidade de tempo necessária para a sua execução, matematicamente comprovada e, dentro da ciência da computação, a Análise de Algoritmos se dedica a estudar tais aspectos dos algoritmos.

Como um algoritmo pode ser representado?

Nós poderíamos representar um algoritmo da maneira que achássemos melhor, desde que tal representação fosse bem estruturada e organizada. Porém, muitos autores e estudiosos acabaram utilizando com muita frequência dois tipos de representação: **Fluxograma** e **Pseudocódigo**.

Fluxograma

O *fluxograma* é um dos métodos mais utilizados para se representar um algoritmo. Trata-se de uma espécie de diagrama e é utilizado para desenhar e documentar processos (simples ou complexos). Tal tipo de diagrama ajuda o leitor a visualizar um processo, compreendê-lo mais facilmente e encontrar falhas ou gargalos.

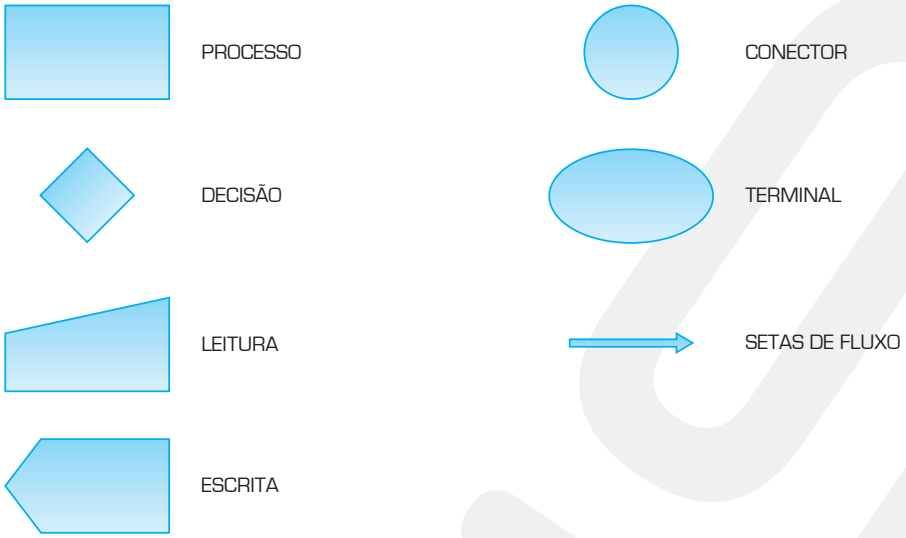

Figura 2.2: Símbolos utilizados em um fluxograma.

Vamos supor que precisemos criar um algoritmo para sacar uma determinada quantia de dinheiro de um caixa eletrônico de um banco. Como ficaria o fluxograma desse algoritmo?

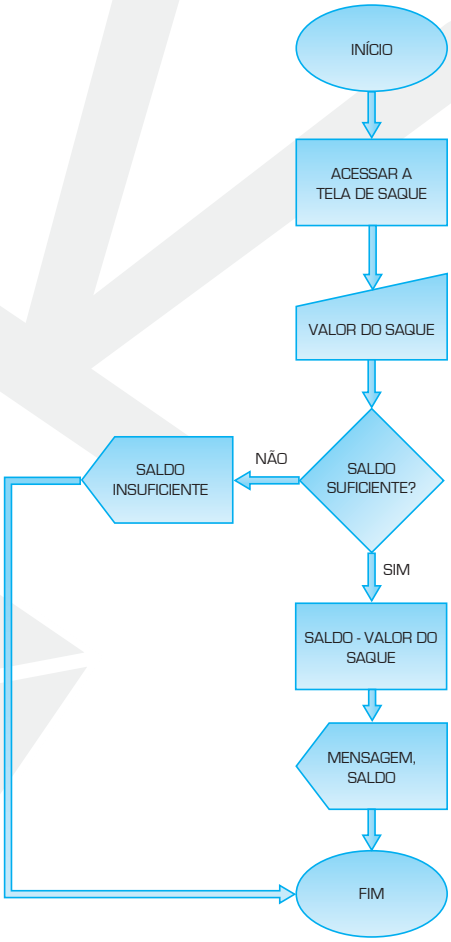

Figura 2.3: Exemplo de fluxograma para a operação de saque em um caixa eletrônico de um banco.

Podemos perceber que, uma vez que conhecemos o que cada símbolo significa, fica fácil entender o processo que o fluxograma representa.

Pseudocódigo

Escrever um algoritmo em *pseudocódigo* é outra forma muito utilizada por autores de livros que tratam de algoritmos, pois dessa forma o leitor não precisa ter o conhecimento prévio de nenhuma linguagem de programação. Nos países cujo idioma principal é o português, muitos se referem ao pseudocódigo como *portugol*. Vamos ver como ficaria o exemplo anterior escrito em pseudocódigo:

```
1  INICIO
2     LER(ValorDoSaque)
3     SE ValorDoSaque > 0 E ValorDoSaque <= Saldo ENTÃO
4        Saldo = Saldo - ValorDoSaque;
5        ESCREVER("Saque efetuado com sucesso. Saldo atual: ", Saldo);
6     SENÃO
7        ESCREVER("Saldo Insuficiente.");
8     FIM SE
9  FIM
```

Pseudocódigo 2.1: Exemplo de pseudocódigo para a operação de saque em um caixa eletrônico.

A representação em pseudocódigo é bem simples e na maioria dos casos é o suficiente para se explicar um algortimo. Existem alguns interpretadores de portugol como o *VisuAlg* e, no caso da língua inglesa, temos algumas linguagens como *Pascal* e *BASIC* cuja sintaxe se assemelha muito com a sintaxe de um pseudocódigo em inglês.

Exercícios de Fixação

1 Escreva, utilizando um fluxograma, um algoritmo para a operação de depósito em um caixa eletrônico de um banco.

2 Escreva, utilizando um fluxograma, um algoritmo para calcular o desconto obtido por um aluno da K19 através do Programa Indicação Premiada (veja as regras no site).

Dica: faça com que o aluno que está indicando receba de início 5% de desconto.

Desafios

1 Escreva, utilizando um fluxograma, um possível algoritmo para o jogo Travessia do Rio disponível online em diversos sites (ex: http://www.aulavaga.com.br/jogos/raciocinio/travessia-do-rio/).

O jogo consiste em atravessar todos os personagens de uma margem à outra do rio seguindo as seguintes regras:

1. Somente o pai, a mãe e o policial sabem pilotar o barco;
2. A mãe não pode ficar sozinha com os filhos;
3. O pai não pode ficar sozinho com as filhas;
4. O prisioneiro não pode ficar com nenhum membro da família sem o policial;
5. O barco pode transportar, no máximo, duas pessoas por vez;
6. Você pode fazer quantas viagens desejar.

CAPÍTULO 3

VARIÁVEIS

O que é uma Variável?

Em computação, uma *variável* representa um endereço da memória RAM. Nela é possível armazenar dados de vários tipos: numéricos, strings (texto), booleanos (verdadeiro ou falso), referências, entre outros.

Quando declaramos uma variável, estamos atribuindo um nome simbólico à um endereço da memória RAM. Dentro de nosso programa utilizaremos esse nome para manipular a informação contida no endereço da memória relacionado à variável.

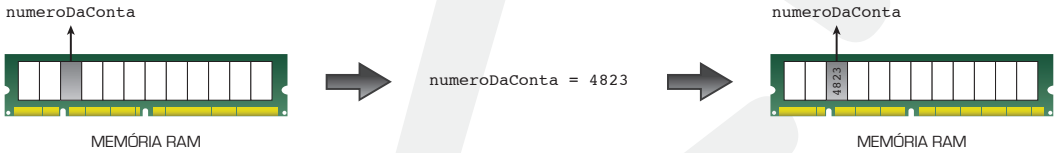

Figura 3.1: Processo de atribuição do valor numérico 4823 à variável numeroDaConta.

Declarando variáveis em Java ou C#

Para criar uma variável em Java ou C#, é necessário declará-la. Nessas duas linguagens de programação, para declarar uma variável é necessário informar o seu tipo e o seu nome (identificador).

```
1 int numeroDaConta;
2 double saldo;
3 boolean contaAtiva;
```

Código Java 3.1: Declaração de variáveis em Java.

```
1 int numeroDaConta;
2 double saldo;
3 bool contaAtiva;
```

Código C# 3.1: Declaração de variáveis em C#.

Importante

Em Java ou C#, devemos informar o tipo de dado que a variável armazenará. Isso se deve ao fato das duas linguagens serem **estaticamente tipadas**, ou seja, o tipo das variáveis deve ser definido em tempo de compilação.

Inicialização

Após declararmos uma variável e antes de utilizá-la, devemos inicializá-la para evitarmos um erro de compilação.

```
1 int numeroDaConta;
2 numeroDaConta = 3466;
3
4 boolean contaAtiva = true;
```

Código Java 3.2: Declaração e inicialização de variáveis em Java.

```
1 int numeroDaConta;
2 numeroDaConta = 3466;
3
4 bool contaAtiva = true;
```

Código C# 3.2: Declaração e inicialização de variáveis em C#.

Como podemos observar, a inicialização das variáveis se dá através do operador =. Note também que podemos, em uma única linha, declarar e inicializar uma variável.

Pare para pensar...
O que aconteceria se o compilador Java ou C# nos permitisse utilizar uma variável não inicializada?

Um programador C (não C#), responderia essa pergunta facilmente, pois em C é possível utilizar uma variável sem inicializá-la. Quando uma variável é declarada, um espaço na memória ram do computador é reservado. Esse espaço pode ter sido utilizado, anteriormente, por outro programa e pode conter dados não mais utilizados. Dessa forma, se uma variável não inicializada for utilizada, o valor que estava no espaço de memória correspondente a essa variável será utilizado.

Muitos programadores C esquecem de inicializar suas variáveis com os valores adequados. Isso provoca muitos erros de lógica. Em Java e C#, esse problema não existe pois as variáveis devem sempre serem inicializadas antes de serem utilizadas.

Tipos de variáveis

As linguagens Java e C# possuem tipos básicos de variáveis. Esses tipos são os mais utilizados e servem como base para a criação de outros tipos. A seguir, veja os tipos básicos da linguagem Java e da linguagem C#.

Tipo	Descrição	Tamanho (“peso”)
byte	Valor inteiro entre -128 e 127 (inclusivo)	1 byte
short	Valor inteiro entre -32.768 e 32.767 (inclusivo)	2 bytes
int	Valor inteiro entre -2.147.483.648 e 2.147.483.647 (inclusivo)	4 bytes
long	Valor inteiro entre -9.223.372.036.854.775.808 e 9.223.372.036.854.775.807 (inclusivo)	8 bytes

Tipo	Descrição	Tamanho (“peso”)
float	Valor com ponto flutuante entre $1,40129846432481707 \times 10^{-45}$ e $3,40282346638528860 \times 10^{38}$ (positivo ou negativo)	4 bytes
double	Valor com ponto flutuante entre $4,94065645841246544 \times 10^{-324}$ e $1,79769313486231570 \times 10^{308}$ (positivo ou negativo)	8 bytes
boolean	true ou false	1 bit
char	Um único caractere Unicode de 16 bits. Valor inteiro e positivo entre 0 (ou ‘\u0000’) e 65.535 (ou ‘\uffff’)	2 bytes

Tabela 3.1: Tipos de dados básicos em Java.

Tipo	Descrição	Tamanho (“peso”)
sbyte	Valor inteiro entre -128 e 127 (inclusivo)	1 byte
byte	Valor inteiro entre 0 e 255 (inclusivo)	1 byte
short	Valor inteiro entre -32.768 e 32.767 (inclusivo)	2 bytes
ushort	Valor inteiro entre 0 e 65.535 (inclusivo)	2 bytes
int	Valor inteiro entre -2.147.483.648 e 2.147.483.647 (inclusivo)	4 bytes
uint	Valor inteiro entre 0 e 4.294.967.295 (inclusivo)	4 bytes
long	Valor inteiro entre -9.223.372.036.854.775.808 e 9.223.372.036.854.775.807 (inclusivo)	8 bytes
ulong	Valor inteiro entre 0 e 18.446.744.073.709.551.615 (inclusivo)	8 bytes
float	Valor com ponto flutuante entre $1,40129846432481707 \times 10^{-45}$ e $3,40282346638528860 \times 10^{38}$ (positivo ou negativo)	4 bytes
double	Valor com ponto flutuante entre $4,94065645841246544 \times 10^{-324}$ e $1,79769313486231570 \times 10^{308}$ (positivo ou negativo)	8 bytes
decimal	Valor com ponto flutuante entre $1,0 \times 10^{-28}$ e $7,9 \times 10^{28}$ (positivo ou negativo)	16 bytes
bool	true ou false	1 bit
char	Um único caractere Unicode de 16 bits. Valor inteiro e positivo entre 0 (ou ‘\u0000’) e 65.535 (ou ‘\uffff’)	2 bytes

Tabela 3.2: Tipos de dados básicos em C#.

Convenções de nomenclatura

Tanto em Java como em C# existe uma convenção para a escolha dos nomes das variáveis. Ambas as linguagens utilizam o padrão *Camel Case*, que consiste em escrever o nome da variável com a primeira letra de cada palavra em maiúscula exceto a primeira.

```
1 int numeroDaConta;
2 int NumeroDaConta; // não segue a convenção
```

Código Java 3.3: Convenção para a escrita dos nomes das variáveis em Java e C#.

Também devemos nos lembrar que as duas linguagens são *Case Sensitive*, ou seja, `numeroDaConta` e `NumeroDaConta` são consideradas duas variáveis diferentes pelo fato do nome da primeira começar com letra minúscula e o da segunda maiúscula.

Regras de nomenclatura

As linguagens Java e C# possuem regras muito parecidas a respeito da nomenclatura das variáveis. O nome de uma variável:

1. Não deve começar com um dígito;
2. Não pode ser igual a uma palavra reservada;
3. Não pode conter espaço(s);
4. Pode ser uma palavra de qualquer tamanho;
5. Pode conter letras, dígitos e _ (underscore). Em Java, pode conter também o caractere $.

```
1 int numeroDaConta;
2 int 2outraVariavel; // inválido - não pode começar com um dígito
3 double double; // inválido - não pode ser igual a uma palavra reservada
4 double saldo da conta; // inválido - não pode conter espaços
5 int umaVariavelComUmNomeSuperHiperMegaUltraGigante;
6 int numeroDaContaCom8Digitos_semPontos;
7 int valorDoProdutoEmR$; // válido somente em Java
8 int #telefone; // inválido - o caractere # não é válido para a nomenclatura
```

Código Java 3.4: Exemplos de nomes de variáveis válidos e inválidos

As linguagens Java e C# permitem a criação de nomes de variáveis em qualquer idioma, pois elas aceitam qualquer caractere *Unicode*. Portanto são válidas as variáveis escritas com as acentuações do português, assim como as variáveis escritas em japonês, por exemplo.

Apesar de ser possível o uso de caracteres especiais, assim como o uso dos caracteres $ (cifrão) e _ (underscore), não é recomendável utilizá-los. Não utilizar tais caracteres é uma boa prática de programação. Essa prática facilita a leitura do código fonte em qualquer editor de texto.

Exercícios de Fixação

1 Abra um terminal; Entre na pasta dos seus exercícios e crie uma pasta chamada **variaveis** para os arquivos desenvolvidos nesse capítulo.

```
K19$ cd rafael

K19/rafael$ ls
introducao
```

```
K19/rafael$ mkdir variaveis

K19/rafael$ ls
introducao variaveis
```

Terminal 3.1: Criando a pasta `variaveis`

2 Na pasta `variaveis`, implemente um programa em Java que declare uma variável do tipo `int` chamada `idade`. Essa variável deve ser inicializada com o valor da sua idade. Utilize os comandos de impressão para mostrar o valor dessa variável.

```
1  class TestaVariavel {
2    public static void main(String[] args) {
3      int idade;
4
5      idade = 27;
6
7      System.out.println(idade);
8    }
9  }
```

Código Java 3.5: TestaVariavel.java

Compile e execute a classe `TestaVariavel`

```
K19/rafael/variaveis$ ls
TestaVariavel.java

K19/rafael/variaveis$ javac TestaVariavel.java

K19/rafael/variaveis$ ls
TestaVariavel.class TestaVariavel.java

K19/rafael/variaveis$ java TestaVariavel
27
```

Terminal 3.2: Compilando e executando a classe `TestaVariavel`

3 Na pasta `variaveis`, implemente um programa em C# que declare uma variável do tipo `int` chamada `idade`. Essa variável deve ser inicializada com o valor da sua idade. Utilize os comandos de impressão para mostrar o valor dessa variável.

```
1  class TestaVariavel
2  {
3    static void Main()
4    {
5      int idade;
6
7      idade = 27;
8
9      System.Console.WriteLine(idade);
10   }
11 }
```

Código C# 3.3: TestaVariavel.cs

```
K19/rafael/variaveis$ ls
TestaVariavel.class TestaVariavel.cs TestaVariavel.java

K19/rafael/variaveis$ mcs TestaVariavel.cs

K19/rafael/variaveis$ ls
TestaVariavel.class TestaVariavel.cs TestaVariavel.exe TestaVariavel.java

K19/rafael/variaveis$ mono TestaVariavel.exe
27
```

Terminal 3.3: Compilando e executando a classe `TestaVariavel`

Exercícios Complementares

1. Indique os tipos adequados da linguagem Java e C# para cada valor da lista abaixo.

   1. “Bom dia”
   2. 3
   3. 235.13
   4. true
   5. -135
   6. 256.23F
   7. ‘A’
   8. 6463275245745L

2. Suponha que iremos começar a desenvolver o programa de gerenciamento de mercadorias de uma loja. Escreva um código que declare variáveis para representar os seguintes dados: número do pedido, código do produto, quantidade e valor total da compra.

3. Continuando o exercício anterior, inicialize as variáveis com valores de acordo com o tipo de variável que você escolheu em cada declaração.

4. Continuando o exercício anterior, imprima na tela o valor de cada variável.

Desafios

1. Olhando para a solução dada nos exercícios complementares, você faria alguma alteração caso estivéssemos desenvolvendo o sistema para uma loja pequena? E se fosse para uma grande rede de lojas? Quais seriam as alterações e quais as implicações?

CAPÍTULO 4

OPERADORES

Tipos de Operadores

Para manipular as variáveis de uma aplicação, devemos aplicar os operadores oferecidos pela linguagem de programação utilizada. As linguagens Java e C# possuem diversos operadores que são categorizados da seguinte forma:

- Aritmético (`+`, `-`, `*`, `/`, `%`)
- Atribuição (`=`, `+=`, `-=`, `*=`, `/=`, `%=`)
- Relacional (`==`, `!=`, `<`, `<=`, `>`, `>=`)
- Lógico (`&&`, `||`)

Operadores Aritméticos

Os operadores aritméticos funcionam de forma muito semelhante aos operadores na matemática. Os operadores aritméticos são:

- Soma `+`
- Subtração `-`
- Multiplicação `*`
- Divisão `/`
- Módulo `%`

```
1  int umMaisUm = 1 + 1;
2  // umMaisUm = 2
3
4  int tresVezesDois = 3 * 2;
5  // tresVezesDois = 6
6
7  int quatroDivididoPor2 = 4 / 2;
8  // quatroDivididoPor2 = 2
9
10 int seisModuloCinco = 6 % 5;
11 // seisModuloCinco = 1
12
13 int x = 7;
14
15 x = x + 1 * 2;
16 // x = 9
17
18 x = x - 4;
19 // x = 5
20
```

```
21 x = x / (6 - 2 + (3 * 5)/(16 - 1));
22 // x = 1
```

Código Java 4.1: Exemplo de uso dos operadores aritméticos.

O módulo de um número x, na matemática, é o valor numérico de x desconsiderando o seu sinal (valor absoluto). Na matemática, expressamos o módulo da seguinte forma: $|-2| = 2$.

Em linguagens de programação, o módulo de um número é o resto da divisão desse número por outro. No exemplo acima, o resto da divisão de 6 por 5 é igual a 1. Além disso, lemos a expressão 6%5 da seguinte forma: seis módulo cinco.

Importante
As operações aritméticas em Java e C# obedecem as mesmas regras da matemática com relação à precedência dos operadores e parênteses. Portanto, o cálculo começa com as operações definidas nos parênteses mais internos até os mais externos. As operações de multiplicação, divisão e módulo são resolvidas antes das operações de subtração e adição.

Operadores de Atribuição

Nos capítulos anteriores, vimos o principal operador de atribuição, o operador = (igual). Os operadores de atribuição são:

- Simples =
- Incremental +=
- Decremental -=
- Multiplicativa *=
- Divisória /=
- Modular %=

```
1  int valor = 1;
2  // valor = 1
3
4  valor += 2;
5  // valor = 3
6
7  valor -= 1;
8  // valor = 2
9
10 valor *= 6;
11 // valor = 12
12
13 valor /= 3;
14 // valor = 4
15
16 valor %= 3;
17 // valor = 1
```

Código Java 4.2: Exemplo de uso dos operadores de atribuição.

As instruções acima poderiam ser escritas de outra forma:

```
int valor = 1;
// valor = 1

valor = valor + 2;
// valor = 3

valor = valor - 1;
// valor = 2

valor = valor * 6;
// valor = 12

valor = valor / 3;
// valor = 4

valor = valor % 3;
// valor = 1
```

Código Java 4.3: O mesmo exemplo anterior utilizando os operadores aritméticos.

Como podemos observar, os operadores de atribuição, exceto o simples (=), reduzem a quantidade de código escrito. Podemos dizer que esses operadores funcionam como "atalhos" para as operações que utilizam os operadores aritméticos.

Operadores Relacionais

Muitas vezes precisamos determinar a precedência de uma variável ou valor em relação à outra variável ou valor. Nessas situações, utilizamos os operadores relacionais. As operações realizadas com os operadores relacionais devolvem valores do tipo boolean em Java ou bool em C#. Os operadores relacionais são:

- Igualdade ==
- Desigualdade !=
- Menor <
- Menor ou igual <=
- Maior >
- Maior ou igual >=

```
int valor = 2;
boolean t = false;
t = (valor == 2);   // t = true
t = (valor != 2);   // t = false
t = (valor < 2);    // t = false
t = (valor <= 2);   // t = true
t = (valor > 1);    // t = true
t = (valor >= 1);   // t = true
```

Código Java 4.4: Exemplo de uso dos operadores relacionais em Java.

```
int valor = 2;
bool t = false;
t = (valor == 2);   // t = true
t = (valor != 2);   // t = false
t = (valor < 2);    // t = false
t = (valor <= 2);   // t = true
```

```
7  t = (valor > 1);     // t = true
8  t = (valor >= 1);    // t = true
```

Código C# 4.1: Exemplo de uso dos operadores relacionais em C#.

Operadores Lógicos

As linguagens Java e C# permitem verificar duas ou mais condições através de operadores lógicos. Os operadores lógicos devolvem valores do tipo boolean em Java ou bool em C#. Os operadores lógicos são:

- "E" lógico &&
- "OU" lógico ||

```
1  int valor = 30;
2  boolean teste = false;
3  teste = valor < 40 && valor > 20;      // teste = true
4  teste = valor < 40 && valor > 30;      // teste = false
5  teste = valor > 30 || valor > 20;      // teste = true
6  teste = valor > 30 || valor < 20;      // teste = false
7  teste = valor < 50 && valor == 30;     // teste = true
```

Código Java 4.5: Exemplo de uso dos operadores lógicos em Java.

```
1  int valor = 30;
2  bool teste = false;
3  teste = valor < 40 && valor > 20;      // teste = true
4  teste = valor < 40 && valor > 30;      // teste = false
5  teste = valor > 30 || valor > 20;      // teste = true
6  teste = valor > 30 || valor < 20;      // teste = false
7  teste = valor < 50 && valor == 30;     // teste = true
```

Código C# 4.2: Exemplo de uso dos operadores lógicos em C#.

Tabela Verdade

A `tabela verdade` é uma tabela matemática muito utilizada na Álgebra Booleana e faremos o uso dela para compreendermos melhor os operadores lógicos.

Sendo A e B duas variáveis booleanas, confira como ficaria a tabela verdade para os operadores lógicos "E" (&&) e "OU" (||):

A	B	A e B	A ou B
V	V	V	V
V	F	F	V
F	V	F	V
F	F	F	F

V: Verdadeiro (true)

F: Falso (false)

Figura 4.1: Tabela verdade para os operadores lógicos "E" (&&) e "OU" (||).

Rapidamente notamos que a operação “E” devolve true apenas quando A e B são true. Também notamos que a operação “OU” devolve false apenas quando A e B são false.

Vamos utilizar os exemplos de operadores lógicos dados anteriormente para ilustrarmos melhor como funciona a tabela verdade.

Primeiramente, vamos ver como fica a tabela verdade completa para algumas das verificações feitas sobre a variável valor:

	A: valor < 40	B: valor > 20	C: valor > 30	A e B	A e C	C ou B
1	V	V	V	V	V	V
2	V	V	F	V	F	V
3	V	F	V	F	V	V
4	V	F	F	F	F	F
5	F	V	V	F	F	V
6	F	V	F	F	F	V
7	F	F	V	F	F	V
8	F	F	F	F	F	F

V: Verdadeiro (true)

F: Falso (false)

Figura 4.2: Tabela verdade sobre a variável valor.

No início do exemplo inicializamos a variável valor com o valor 30, portanto temos a seguinte situação com relação às verificações:

- valor < 40 é true
- valor > 20 é true
- valor > 30 é false

Agora fica fácil escolhermos a linha da tabela verdade que corresponde à essa situação:

	A: valor < 40	B: valor > 20	C: valor > 30	A e B	A e C	C ou B
1	V	V	V	V	V	V
2	V	V	F	V	F	V
3	V	F	V	F	V	V
4	V	F	F	F	F	F
5	F	V	V	F	F	V
6	F	V	F	F	F	V
7	F	F	V	F	F	V
8	F	F	F	F	F	F

V: Verdadeiro (true)

F: Falso (false)

Figura 4.3: Linha da tabela verdade que corresponde à situação passada no exercício sobre operadores lógicos.

Exercícios de Fixação

1 Abra um terminal; Entre na pasta dos seus exercícios e crie uma pasta chamada **operadores** para os arquivos desenvolvidos nesse capítulo.

```
K19$ cd rafael

K19/rafael$ ls
introducao variaveis

K19/rafael$ mkdir operadores

K19/rafael$ ls
introducao operadores variaveis
```

Terminal 4.1: Criando a pasta operadores

2 Na pasta operadores, implemente um programa em Java que utilize os operadores aritméticos.

```
1  class TestaOperadoresAritmeticos {
2    public static void main(String[] args) {
3      int a = 1 + 1;
4      int b = 10 - 2;
5      int c = 2 * 3;
6      int d = 25 / 5;
7      int e = 10 % 4;
8
9      System.out.println(a);
10     System.out.println(b);
11     System.out.println(c);
12     System.out.println(d);
13     System.out.println(e);
14   }
15 }
```

Código Java 4.6: TestaOperadoresAritmeticos.java

Compile e execute a classe TestaOperadoresAritmeticos

```
K19/rafael/operadores$ javac TestaOperadoresAritmeticos.java

K19/rafael/operadores$ java TestaOperadoresAritmeticos
2
8
6
5
2
```

Terminal 4.2: Compilando e executando a classe TestaOperadoresAritmeticos

3 Na pasta operadores, implemente um programa em C# que utilize os operadores aritméticos.

```
1  class TestaOperadoresAritmeticos
2  {
3    static void Main()
4    {
5      int a = 1 + 1;
6      int b = 10 - 2;
7      int c = 2 * 3;
8      int d = 25 / 5;
9      int e = 10 % 4;
10
11     System.Console.WriteLine(a);
12     System.Console.WriteLine(b);
13     System.Console.WriteLine(c);
14     System.Console.WriteLine(d);
15     System.Console.WriteLine(e);
```

```
16    }
17  }
```

Código C# 4.3: TestaOperadoresAritmeticos.cs

Compile e execute a classe `TestaOperadoresAritmeticos`

```
K19/rafael/operadores$ mcs TestaOperadoresAritmeticos.cs

K19/rafael/operadores$ mono TestaOperadoresAritmeticos.exe
2
8
6
5
2
```

Terminal 4.3: Compilando e executando a classe `TestaOperadoresAritmeticos`

4 Na pasta `operadores`, implemente um programa em Java que utilize os operadores de atribuição.

```
1   class TestaOperadoresDeAtribuicao {
2     public static void main(String[] args) {
3       int a = 1;
4       System.out.println(a);
5
6       a += 2;
7       System.out.println(a);
8
9       a -= 1;
10      System.out.println(a);
11
12      a *= 3;
13      System.out.println(a);
14
15      a /= 2;
16      System.out.println(a);
17
18      a %= 2;
19      System.out.println(a);
20    }
21  }
```

Código Java 4.7: TestaOperadoresDeAtribuicao.java

Compile e execute a classe `TestaOperadoresDeAtribuicao`

```
K19/rafael/operadores$ javac TestaOperadoresDeAtribuicao.java

K19/rafael/operadores$ java TestaOperadoresDeAtribuicao
1
3
2
6
3
1
```

Terminal 4.4: Compilando e executando a classe `TestaOperadoresDeAtribuicao`

5 Na pasta `operadores`, implemente um programa em C# que utilize os operadores de atribuição.

```
1   class TestaOperadoresDeAtribuicao
2   {
3     static void Main()
4     {
5       int a = 1;
6       System.Console.WriteLine(a);
```

```
7
8       a += 2;
9       System.Console.WriteLine(a);
10
11      a -= 1;
12      System.Console.WriteLine(a);
13
14      a *= 3;
15      System.Console.WriteLine(a);
16
17      a /= 2;
18      System.Console.WriteLine(a);
19
20      a %= 2;
21      System.Console.WriteLine(a);
22    }
23  }
```

Código C# 4.4: TestaOperadoresDeAtribuicao.cs

Compile e execute a classe `TestaOperadoresDeAtribuicao`

```
K19/rafael/operadores$ mcs TestaOperadoresDeAtribuicao.cs

K19/rafael/operadores$ mono TestaOperadoresDeAtribuicao.exe
1
3
2
6
3
1
```

Terminal 4.5: Compilando e executando a classe `TestaOperadoresDeAtribuicao`

6 Na pasta `operadores`, implemente um programa em Java que utilize os operadores relacionais.

```
1   class TestaOperadoresRelacionais {
2     public static void main(String[] args) {
3       int a = 1;
4       int b = 2;
5
6       System.out.println(a > b);
7       System.out.println(a >= b);
8       System.out.println(a < b);
9       System.out.println(a <= b);
10      System.out.println(a == b);
11      System.out.println(a != b);
12    }
13  }
```

Código Java 4.8: TestaOperadoresRelacionais.java

Compile e execute a classe `TestaOperadoresRelacionais`

```
K19/rafael/operadores$ javac TestaOperadoresRelacionais.java

K19/rafael/operadores$ java TestaOperadoresRelacionais
false
false
true
true
false
true
```

Terminal 4.6: Compilando e executando a classe `TestaOperadoresRelacionais`

7 Na pasta `operadores`, implemente um programa em C# que utilize os operadores relacionais.

```
1  class TestaOperadoresRelacionais
2  {
3    static void Main()
4    {
5      int a = 1;
6      int b = 2;
7
8      System.Console.WriteLine(a > b);
9      System.Console.WriteLine(a >= b);
10     System.Console.WriteLine(a < b);
11     System.Console.WriteLine(a <= b);
12     System.Console.WriteLine(a == b);
13     System.Console.WriteLine(a != b);
14   }
15 }
```

Código C# 4.5: TestaOperadoresRelacionais.cs

Compile e execute a classe `TestaOperadoresRelacionais`

```
K19/rafael/operadores$ mcs TestaOperadoresRelacionais.cs

K19/rafael/operadores$ mono TestaOperadoresRelacionais.exe
False
False
True
True
False
True
```

Terminal 4.7: Compilando e executando a classe `TestaOperadoresRelacionais`

8 Na pasta `operadores`, implemente um programa em Java que utilize os operadores lógicos.

```
1  class TestaOperadoresLogicos {
2    public static void main(String[] args) {
3      int a = 1;
4      int b = 2;
5      int c = 3;
6      int d = 4;
7
8      System.out.println(a > b || c < d);
9      System.out.println(a > b && c < d);
10   }
11 }
```

Código Java 4.9: TestaOperadoresLogicos.java

Compile e execute a classe `TestaOperadoresLogicos`

```
K19/rafael/operadores$ javac TestaOperadoresLogicos.java

K19/rafael/operadores$ java TestaOperadoresLogicos
true
false
```

Terminal 4.8: Compilando e executando a classe `TestaOperadoresLogicos`

9 Na pasta `operadores`, implemente um programa em C# que utilize os operadores lógicos.

```
1  class TestaOperadoresLogicos
2  {
3    static void Main()
4    {
5      int a = 1;
```

```
6       int b = 2;
7       int c = 3;
8       int d = 4;
9
10      System.Console.WriteLine(a > b || c < d);
11      System.Console.WriteLine(a > b && c < d);
12    }
13  }
```

Código C# 4.6: TestaOperadoresLogicos.cs

Compile e execute a classe `TestaOperadoresLogicos`

```
K19/rafael/operadores$ mcs TestaOperadoresLogicos.cs

K19/rafael/operadores$ mono TestaOperadoresLogicos.exe
True
False
```

Terminal 4.9: Compilando e executando a classe `TestaOperadoresLogicos`

Exercícios Complementares

1 Reescreva todos os exemplos sobre operadores vistos nesse capítulo e para cada operação imprima o resultado armazenado nas variáveis.

2 Imagine que temos um programa que declara uma variável de nome `valor` e a inicializa com o valor 50. Suponha seis verificações: A(valor > 40), B(valor < 60), C(valor >= 70), D(A e B), E(B ou C) e F(A ou C). Construa uma tabela verdade completa para representar essas seis verificações e escolha a linha que cujas verificações estão de acordo com o valor da variável `valor`.

CONTROLE DE FLUXO

CAPÍTULO 5

Neste capítulo, mostraremos instruções que permitem controlar o fluxo de um programa. Essas instruções aumentam a "inteligência" do código. As linguagens de programação oferecem dois tipos de instruções para controlar o fluxo de execução dos programas: instruções de **decisão** e de **repetição**.

Instruções de Decisão

Nos exemplos vistos nos capítulos anteriores, a ordem da execução das linhas de um programa é exatamente a ordem na qual elas foram definidas no código fonte. As instruções de decisão proporcionará uma forma de decidirmos se queremos executar um bloco de código ou não, ou seja, se desejamos pular um trecho de código ou não. As instruções de decisão são capazes de criar um "desvio" no fluxo de execução de um programa.

`if`

A instrução `if` (se), é utilizada quando queremos testar uma condição antes de executarmos um ou mais comandos. A estrutura ou sintaxe da instrução `if` é a seguinte:

```
1  if(condição) {
2    // comando 1
3    // comando 2
4    // comando 3
5  }
6  // comando 4
```

Código Java 5.1: A instrução `if`

Como funciona a instrução `if`? Se a condição na linha 1 for verdadeira, os comandos das linhas 2, 3 e 4 serão executadas e depois o fluxo de execução do programa segue normalmente e executa a linha 6. Por outro lado, se a condição for falsa, as linhas 2, 3 e 4 não serão executadas e o fluxo de execução do programa vai direto para a linha 6.

Pare para pensar...
O que é essa tal condição?

A condição é qualquer expressão válida em Java ou C# que devolva um valor booleano. Por exemplo, a expressão `1 < 2` é uma expressão que devolve o valor `true`. Já a expressão `8%3 == 0` devolve o valor `false`

if...else

Muitas vezes, queremos executar um bloco de comandos caso uma condição seja verdadeira e outro bloco de comandos caso essa condição seja falsa. Para isso, podemos utilizar as instruções `if` e `else`. Veja abaixo, a estrutura dessas instruções.

```
1  if(condição) {
2    // comando 1
3    // comando 2
4    // comando 3
5  } else {
6    // comando 4
7    // comando 5
8    // comando 6
9  }
10 // comando 7 ...
```

Código Java 5.2: As instruções `if` *e* `else`.

No exemplo acima, se a condição na linha 1 for verdadeira, as linhas 2, 3 e 4 serão executadas e depois o fluxo de execução do programa segue para a linha 10. Por outro lado, se a condição na linha 1 for falsa, as linhas 6, 7 e 8 serão executadas e depois o fluxo de execução do programa segue para a linha 10.

A instrução `else` não pode aparecer sozinha no código sem estar vinculada a uma instrução `if`. A instrução `else` pode ser traduzida em português para "senão". Em português, assim como em Java e C#, não podemos dizer "senão" sem antes ter dito "se". Por isso, não podemos utilizar a instrução `else` sem antes ter utilizado a instrução `if`.

if...else if...

Podemos encadear instruções `if` dentro de instruções `else` para testar outras condições caso a primeira condição seja falsa. Veja o exemplo abaixo.

```
1  if(condição 1) {
2    // comando 1
3  } else {
4    if(condição 2) {
5      // comando 2
6    }
7  }
```

Código Java 5.3: `if`*s encadeados*

Se a condição da linha 1 for verdadeira, a linha 2 será executada. Por outro lado, se a condição da linha 1 for falsa, a linha 4 será executada e a segunda condição será testada. Se essa segunda condição for verdadeira, a linha 5 será executada.

No exemplo acima, as chaves do `else` podem ser omitidas.

```
1  if(condição 1) {
2    // comando 1
3  } else if(condição 2) {
4    // comando 2
```

```
5 }
```

Código Java 5.4: `if`s encadeados

Exercícios de Fixação

1 Abra um terminal; Entre na pasta dos seus exercícios e crie uma pasta chamada **controle-de-fluxo** para os arquivos desenvolvidos nesse capítulo.

```
K19$ cd rafael

K19/rafael$ ls
introducao operadores variaveis

K19/rafael$ mkdir controle-de-fluxo

K19/rafael$ ls
controle-de-fluxo introducao operadores variaveis
```

Terminal 5.1: Criando a pasta `controle-de-fluxo`

2 Na pasta `controle-de-fluxo`, implemente um programa em Java com duas variáveis: `precoDoProduto1` e `precoDoProduto2`. Esse programa deve imprimir a mensagem "O produto 1 é o mais caro." quando o valor da variável `precoDoProduto1` for maior que o valor da variável `precoDoProduto2` ou a mensagem "O produto 2 é o mais caro." quando o valor da variável `precoDoProduto2` for maior que o valor da variável `precoDoProduto1`.

```
1  class PrecoProduto {
2    public static void main(String[] args) {
3      double precoDoProduto1 = 5325.12;
4      double precoDoProduto2 = 4366.34;
5
6      if (precoDoProduto1 < precoDoProduto2) {
7        System.out.println("O produto 2 é o mais caro.");
8      } else if(precoDoProduto2 < precoDoProduto1){
9        System.out.println("O produto 1 é o mais caro.");
10     }
11   }
12 }
```

Código Java 5.5: PrecoProduto.java

```
K19/rafael/controle-de-fluxo$ javac PrecoProduto.java

K19/rafael/controle-de-fluxo$ java PrecoProduto
O produto 1 é o mais caro
```

Terminal 5.2: Compilando e executando a classe `PrecoProduto`

3 Implemente o mesmo programa do exercício anterior em C#.

```
1  class PrecoProduto
2  {
3    static void Main()
4    {
5      double precoDoProduto1 = 5325.12;
6      double precoDoProduto2 = 4366.34;
7
8      if (precoDoProduto1 < precoDoProduto2)
9      {
```

```
10        System.Console.WriteLine("O produto 2 é o mais caro.");
11      }
12      else if(precoDoProduto2 < precoDoProduto1)
13      {
14        System.Console.WriteLine("O produto 1 é o mais caro.");
15      }
16    }
17  }
```

Código C# 5.1: PrecoProduto.cs

```
K19/rafael/controle-de-fluxo$ mcs PrecoProduto.java

K19/rafael/controle-de-fluxo$ mono PrecoProduto.exe
O produto 1 é o mais caro
```

Terminal 5.3: Compilando e executando a classe `PrecoProduto`

4 Escreva um programa em Java que guarde dois valores numéricos: a e b. Imprima na tela a mensagem "É divisível" quando a for divisível por b ou a mensagem "Não é divisível", caso contrário.

```
1   class ADivisivelPorB {
2     public static void main(String[] args) {
3       int a = 246;
4       int b = 3;
5
6       if (a % b == 0) {
7         System.out.println("É divisível");
8       } else {
9         System.out.println("Não é divisível");
10      }
11    }
12  }
```

Código Java 5.6: ADivisivelPorB.java

```
K19/rafael/controle-de-fluxo$ javac ADivisivelPorB.java

K19/rafael/controle-de-fluxo$ java ADivisivelPorB
É divisível
```

Terminal 5.4: Compilando e executando a classe `ADivisivelPorB`

5 Implemente o mesmo programa do exercício anterior em C#.

```
1   class ADivisivelPorB
2   {
3     static void Main()
4     {
5       int a = 246;
6       int b = 3;
7
8       if (a % b == 0)
9       {
10        System.Console.WriteLine("É divisível");
11      }
12      else
13      {
14        System.Console.WriteLine("Não é divisível");
15      }
16    }
17  }
```

Código C# 5.2: ADivisivelPorB.cs

```
K19/rafael/controle-de-fluxo$ mcs ADivisivelPorB.java

K19/rafael/controle-de-fluxo$ mono ADivisivelPorB.exe
É divisível
```

Terminal 5.5: Compilando e executando a classe `ADivisivelPorB`

6 Escreva um programa em Java que contenha uma variável chamada `horaDoDia`. Essa variável deve armazenar a hora do dia. Esse programa deve imprimir a mensagem "Bom dia" se a hora estiver no intervalo [0, 11], "Boa tarde" se a hora estiver no intervalo [12, 17] ou "Boa noite" se a hora estiver no intervalo [18, 23].

```
1  class Saudacao {
2    public static void main(String[] args) {
3      int hora = 22;
4
5      if (hora >= 0 && hora < 12) {
6        System.out.println("Bom dia");
7      } else if (hora >= 12 && hora < 18) {
8        System.out.println("Boa tarde");
9      } else if (hora >= 18 && hora < 24) {
10       System.out.println("Boa noite");
11     } else {
12       System.out.println("Unibanco 30 horas :P");
13     }
14   }
15 }
```

Código Java 5.7: Saudacao.java

```
K19/rafael/controle-de-fluxo$ javac Saudacao.java

K19/rafael/controle-de-fluxo$ java Saudacao
Boa noite
```

Terminal 5.6: Compilando e executando a classe `Saudacao`

7 Implemente o mesmo programa do exercício anterior em C#.

```
1  class Saudacao
2  {
3    static void Main()
4    {
5      int hora = 22;
6
7      if(hora >= 0 && hora < 12)
8      {
9        System.Console.WriteLine("Bom dia");
10     }
11     else if(hora >= 12 && hora < 18)
12     {
13       System.Console.WriteLine("Boa tarde");
14     }
15     else if(hora >= 18 && hora < 24)
16     {
17       System.Console.WriteLine("Boa noite");
18     }
19     else
20     {
21       System.Console.WriteLine("Unibanco 30 horas :P");
22     }
23   }
24 }
```

Código C# 5.3: Saudacao.cs

```
K19/rafael/controle-de-fluxo$ mcs Saudacao.java

K19/rafael/controle-de-fluxo$ mono Saudacao.exe
Boa noite
```

Terminal 5.7: Compilando e executando a classe Saudacao

Exercícios Complementares

1 Para tornar os exercícios anteriores mais interessantes, podemos gerar números aleatórios ao invés de definir esses valores diretamente no código fonte.

Em java, um valor do tipo `double` entre 0 e 1000 pode ser gerado aleatoriamente da seguinte forma:

```
1 java.util.Random gerador = new java.util.Random();
2 double numero = gerador.nextDouble() * 1000;
```

Código Java 5.8: Gerando um valor `double` *entre 0 e 1000 aleatoriamente*

Para gerar um valor do tipo `int` entre 0 e 23, podemos utilizar o seguinte código:

```
1 java.util.Random gerador = new java.util.Random();
2 int numero = gerador.nextInt(24);
```

Código Java 5.9: Gerando um valor `int` *entre 0 e 23 aleatoriamente*

Em C#, um valor do tipo `double` entre 0 e 1000 pode ser gerado aleatoriamente da seguinte forma:

```
1 System.Random gerador = new System.Random();
2 double numero = gerador.NextDouble() * 1000;
```

Código C# 5.4: Gerando um valor `double` *entre 0 e 1000 aleatoriamente*

Para gerar um valor do tipo `int` entre 0 e 23, podemos utilizar o seguinte código:

```
1 System.Random gerador = new System.Random();
2 int numero = gerador.Next(24);
```

Código C# 5.5: Gerando um valor `int` *entre 0 e 23 aleatoriamente*

Modifique os exercícios anteriores para utilizar números aleatórios.

Instruções de Repetição

Basicamente, as instruções de de decisão permitem que um determinado trecho de código seja executado ou não. Em algumas situações, é necessário repetir a execução de um determinado trecho de código. Nessas situações, devemos utilizar as instruções de repetição.

Nos exercícios do Capítulo 2, utilizamos a idéia das instruções de repetição através de fluxogramas. Vamos rever o fluxograma um exercício desse capítulo.

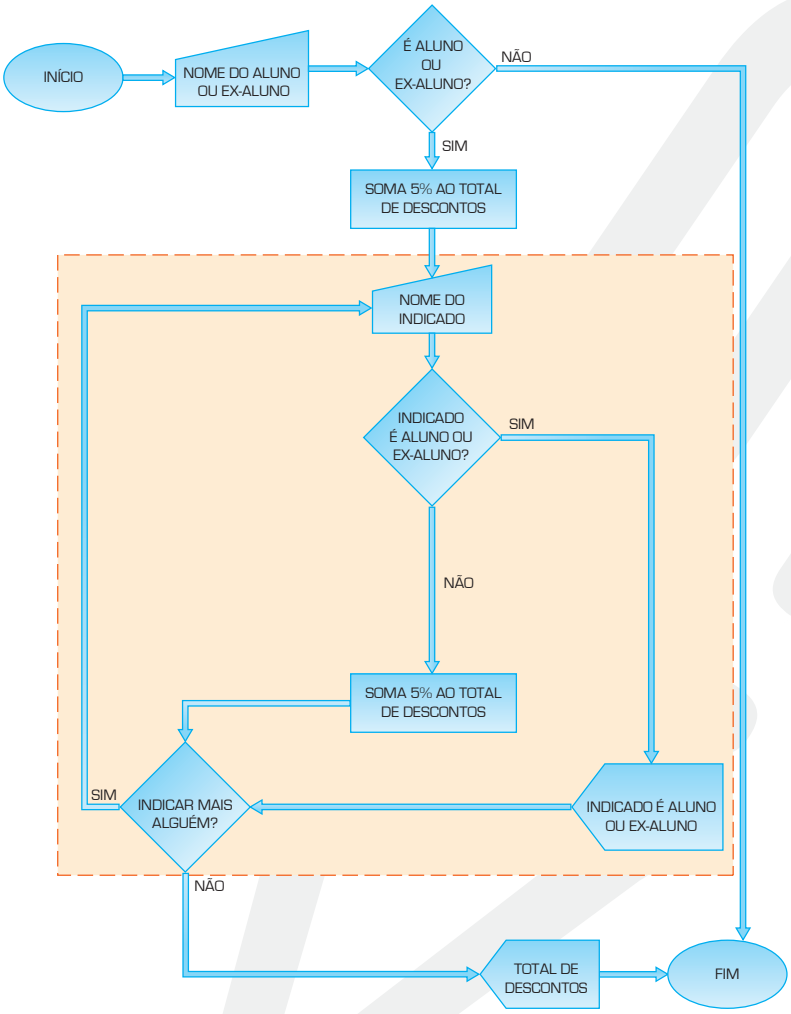

Figura 5.1: Fluxograma do segundo exercício do Capítulo 2. Loop em destaque.

A área em destaque no fluxograma acima está indicando a porção do nosso algoritmo em que ocorre repetições. Em programação chamamos essas repetições de *loop* ou *laço*. Perceba que enquanto a condição "Indicar mais alguém?" devolver `true` a instrução que captura o nome de um aluno indicado é executada. Caso contrário, nosso algoritmo continua até o seu fim.

Instrução `while`

A instrução `while` indica o início de um laço e recebe como parâmetro uma condição. Essa condição é chamada de **condição de parada**, pois quando ela for falsa, o laço é encerrado. A estrutura ou sintaxe da instrução `while` é a seguinte:

```
1  while(condição de parada){
2    // comando 1
3    // comando 2
4    // comando 3
5  }
```

Código Java 5.13: A instrução `while`

Se traduzirmos para o português a instrução `while` como **enquanto**, fica mais fácil entender o seu funcionamento. O código acima poderia ser lido da seguinte forma:

"Enquanto a condição de parada for verdadeira, execute comando 1, comando 2 e comando 3."

Considere um programa que imprime na tela cem mensagens de acordo com o seguinte padrão:

```
Mensagem número 1
Mensagem número 2
Mensagem número 3
...
```

Terminal 5.8: Programa que imprime mensagens

Esse programa poderia ser implementado em Java ou C# de uma forma não prática. Veja os exemplos abaixo.

```
1  System.out.println("Mensagem número 1");
2  System.out.println("Mensagem número 2");
3  System.out.println("Mensagem número 3");
4  ...
5  System.out.println("Mensagem número 100");
```

Código Java 5.14: Imprimindo a frase "Mensagem número x".

```
1  System.Console.WriteLine("Mensagem número 1");
2  System.Console.WriteLine("Mensagem número 2");
3  System.Console.WriteLine("Mensagem número 3");
4  ...
5  System.Console.WriteLine("Mensagem número 100");
```

Código C# 5.9: Imprimindo a frase "Mensagem número x".

Contudo, utilizando a instrução `while` o código fica bem mais simples. Observe.

```
1  // Contador de vezes que a mensagem foi impressa.
2  int i = 1;
3
4  while(i <= 100){
5    System.out.println("Mensagem número " + i);
6    i++;
7  }
```

Código Java 5.15: Imprimindo a frase "Mensagem número x".

```
1  // Contador de vezes que a mensagem foi impressa.
2  int i = 1;
3
4  while(i <= 100){
5    System.Console.WriteLine("Mensagem número " + i);
6    i++;
7  }
```

Código C# 5.10: Imprimindo a frase "Mensagem número x".

Lembre-se

Repare que nas linhas em destaque, utilizamos o operador `+` entre uma `string` e uma variável do tipo `int`. Esse operador pode ser utilizado tanto em Java como em C#, para concatenar uma *string* com qualquer outro valor ou variável de tipos primitivos.

Até agora, o uso da instrução `while` parece ser mais uma conveniência do que uma necessidade. Vamos mudar um pouco o exemplo anterior para verificar a importância das instruções de repetição. Considere que a frase "Mensagem número x" tenha que ser impressa um número aleatório de vezes. Dessa forma, durante a codificação, não sabemos quantas vezes a frase deverá ser impressa.

Um possível código para solucionar esse novo problema seria:

```
class ExemploWhile {
  public static void main(String[] args) {
    int i = 1;
    java.util.Random geradorDeNumeroAleatorio = new java.util.Random();

    // Gera um número aleatório entre 0 e 99
    int numeroAleatorio = geradorDeNumeroAleatorio.nextInt(100);

    // Pega o numero aleatório gerado e soma 1 para que o valor obtido esteja entre 1 ↩
        e 100
    numeroAleatorio++;

    while(i <= numeroAleatorio) {
      System.out.println("Mensagem número " + i);
      i++;
    }
  }
}
```

Código Java 5.16: Imprimindo a frase "Mensagem número x" um número aleatório de vezes.

```
class ExemploWhile
{
  static void Main()
  {
    int i = 1;
    System.Random geradorDeNumeroAleatorio = new System.Random();

    // Gera um número aleatório entre 1 e 100
    int numeroAleatorio = geradorDeNumeroAleatorio.Next(1, 101);

    while(i <= numeroAleatorio)
    {
      System.Console.WriteLine("Mensagem número " + i);
      i++;
    }
  }
}
```

Código C# 5.11: Imprimindo a frase "Mensagem número x" um número aleatório de vezes.

A cada vez que é executado, o programa acima pode imprimir uma quantidade diferente de mensagens. Esse comportamento seria impossível sem a utilização de uma instrução de repetição?

Instrução `for`

A instrução `for` é uma outra instrução de repetição e tem a mesma finalidade da instrução `while`. Na maioria dos casos, podemos resolver questões que envolvem repetições com `while` ou `for`. A diferença é que, geralmente, utilizamos a instrução `for` nos casos em que precisamos de um contador em nossa condição de parada. Para ficar mais claro, veja a estrutura ou sintaxe da instrução `for`:

```
for(expressão 1; condição de parada; expressão 2){
  // comando 1
  // comando 2
  // comando 3
}
```

Código Java 5.17: A instrução `for`

No lugar da *expressão 1*, devemos inserir comandos que deverão ser executados antes do início do laço. No lugar da *expressão 2*, devemos inserir comandos que deverão ser executadas ao final de cada iteração(repetição).

Lembre-se
O termo iteração é utilizado quando nos referimos à repetição de uma ou mais ações. Portanto, quando dizemos que "algo deve ser executado a cada iteração de um laço" estamos querendo dizer que "a cada rodada desse laço algo deve ser executado".

Vamos reescrever um dos exemplos visto anteriormente com a instrução `while` para utilizar a instrução `for`:

```
for(int i = 1; i <= 100; i++) {
  System.out.println("Mensagem número " + i);
}
```

Código Java 5.18: Imprimindo a frase "Mensagem número x" utilizando a instrução `for`.

```
for(int i = 1; i <= 100; i++)
{
  System.Console.WriteLine("Mensagem número " + i);
}
```

Código C# 5.12: Imprimindo a frase "Mensagem número x" utilizando a instrução `for`.

Perceba que o código ficou mais compacto sem prejudicar a compreensão. Na linha em destaque, declaramos e inicializamos a variável `i` (`int i = 1`), definimos a condição de parada (`i <= 100`) e definimos que ao final de cada iteração devemos atualizar a variável `i` (`i++`). Utilizando a instrução `for`, fizemos em apenas uma linha aquilo que foi feito em três linhas utilizando a instrução `while`.

Instrução `break`

A instrução `break` não é uma instrução de repetição, mas está fortemente ligada às instruções `while` e `for`. Ela é utilizada para forçar a parada de um laço. Veja o exemplo abaixo:

```
class ExemploBreak {
  public static void main(String[] args) {
    java.util.Random geradorDeNumeroAleatorio = new java.util.Random();

    // Gera um número aleatório entre 0 e 99
    int numeroAleatorio = geradorDeNumeroAleatorio.nextInt(100);

    // Pega o numero aleatório gerado e soma 1 para que o valor obtido
    // esteja entre 1 e 100
    numeroAleatorio++;
```

```
11
12      System.out.println("Número aleatório: " + numeroAleatorio);
13
14      for (int i = 0; i <= numeroAleatorio; i++) {
15        int j = i * 2 + numeroAleatorio;
16
17        System.out.println("Iteração " + i + ": " + j);
18
19        if (j % 4 == 0) {
20          System.out.println("Fim por módulo.");
21          break;
22        }
23      }
24    }
25  }
```

Código Java 5.19: Exemplo de uso da instrução break

```
1   class ExemploBreak
2   {
3     static void Main()
4     {
5       System.Random geradorDeNumeroAleatorio = new System.Random();
6
7       // Gera um número aleatório entre 1 e 100
8       int numeroAleatorio = geradorDeNumeroAleatorio.Next(1, 101);
9
10      Console.WriteLine("Número aleatório: " + numeroAleatorio);
11
12      for(int i = 0; i <= numeroAleatorio; i++)
13      {
14        int j = i*2 + numeroAleatorio;
15
16        System.Console.WriteLine("Iteração " + i + ": " + j);
17
18        if(j%4 == 0)
19        {
20          System.Console.WriteLine("Fim por módulo.");
21          break;
22        }
23      }
24    }
25  }
```

Código C# 5.13: Exemplo de uso da instrução break

A cada iteração, declaramos a variável `j` e calculamos o seu valor. Além disso, verificamos se o resto da divisão de `j` por 4 é igual a 0. Caso a verificação seja verdadeira, imprimimos uma mensagem e finalizamos o laço com a instrução `break`.

Exercícios de Fixação

8 Escreva um programa em Java que imprime na tela cinco vezes a mensagem "Lorem ipsum dolor sit amet".

```
1   class LoremIpsum {
2     public static void main(String[] args) {
3       for (int i = 0; i < 10; i++) {
4         System.out.println("Lorem ipsum dolor sit amet");
5       }
6     }
7   }
```

Código Java 5.20: LoremIpsum.java

```
K19/rafael/controle-de-fluxo$ javac LoremIpsum.java

K19/rafael/controle-de-fluxo$ java LoremIpsum
Lorem ipsum dolor sit amet
Lorem ipsum dolor sit amet
Lorem ipsum dolor sit amet
Lorem ipsum dolor sit amet
Lorem ipsum dolor sit amet
```

Terminal 5.9: Compilando e executando a classe `LoremIpsum`

9 Implemente o mesmo programa do exercício anterior em C#.

```
1  class LoremIpsum
2  {
3    static void Main()
4    {
5      for(int i = 0; i < 10; i++)
6      {
7        System.Console.WriteLine("Lorem ipsum dolor sit amet");
8      }
9    }
10 }
```

Código C# 5.14: LoremIpsum.cs

```
K19/rafael/controle-de-fluxo$ mcs LoremIpsum.java

K19/rafael/controle-de-fluxo$ mono LoremIpsum.exe
Lorem ipsum dolor sit amet
Lorem ipsum dolor sit amet
Lorem ipsum dolor sit amet
Lorem ipsum dolor sit amet
Lorem ipsum dolor sit amet
```

Terminal 5.10: Compilando e executando a classe `LoremIpsum`

10 Escreva um programa em Java que imprime na tela os números de 1 a 100.

```
1  class Imprime100 {
2    public static void main(String[] args) {
3      for (int i = 1; i <= 100; i++) {
4        System.out.println(i);
5      }
6    }
7  }
```

Código Java 5.21: Imprime100.java

```
K19/rafael/controle-de-fluxo$ javac Imprime100.java

K19/rafael/controle-de-fluxo$ java Imprime100
1
2
3
...
100
```

Terminal 5.11: Compilando e executando a classe `Imprime100`

11 Implemente o mesmo programa do exercício anterior em C#.

```
class Imprime100
{
  static void Main()
  {
    for(int i = 1; i <= 100; i++)
    {
      System.Console.WriteLine(i);
    }
  }
}
```

Código C# 5.15: Imprime100.cs

```
K19/rafael/controle-de-fluxo$ mcs Imprime100.java

K19/rafael/controle-de-fluxo$ mono Imprime100.exe
1
2
3
...
100
```

Terminal 5.12: Compilando e executando a classe `Imprime100`

12 Escreva um programa em Java que imprime na tela os números de 1 a 100 exceto os números múltiplos de 3.

```
class Imprime100ExcetoMultiplo3 {
  public static void main(String[] args) {
    for (int i = 1; i <= 100; i++) {
      if (i % 3 != 0) {
        System.out.println(i);
      }
    }
  }
}
```

Código Java 5.22: Imprime100ExcetoMultiplo3.java

```
K19/rafael/controle-de-fluxo$ javac Imprime100ExcetoMultiplo3.java

K19/rafael/controle-de-fluxo$ java Imprime100ExcetoMultiplo3
1
2
4
5
...
100
```

Terminal 5.13: Compilando e executando a classe `Imprime100ExcetoMultiplo3`

13 Implemente o mesmo programa do exercício anterior em C#.

```
class Imprime100ExcetoMultiplo3
{
  static void Main()
  {
    for(int i = 1; i <= 100; i++)
    {
      if(i % 3 != 0)
      {
        System.Console.WriteLine(i);
      }
    }
  }
}
```

Código C# 5.16: Imprime100ExcetoMultiplo3.cs

```
K19/rafael/controle-de-fluxo$ mcs Imprime100ExcetoMultiplo3.java

K19/rafael/controle-de-fluxo$ mono Imprime100ExcetoMultiplo3.exe
1
2
4
5
...
100
```

Terminal 5.14: Compilando e executando a classe `Imprime100ExcetoMultiplo3`

14 Escreva um programa em Java que declare e inicialize uma variável que receberá o maior número possível do tipo `int`. Divida o valor dessa variável por 2 até que o resultado obtido seja inferior a 100 (não inclusivo). A cada iteração imprima o resultado.

```
1  class DivideMaiorInteiro {
2    public static void main(String[] args) {
3      int numero = 2147483647;
4
5      while (numero >= 100) {
6        numero /= 2;
7        System.out.println(numero);
8      }
9    }
10 }
```

Código Java 5.23: DivideMaiorInteiro.java

```
K19/rafael/controle-de-fluxo$ javac DivideMaiorInteiro.java

K19/rafael/controle-de-fluxo$ java DivideMaiorInteiro
1073741823
536870911
268435455
...
```

Terminal 5.15: Compilando e executando a classe `DivideMaiorInteiro`

15 Implemente o mesmo programa do exercício anterior em C#.

```
1  class DivideMaiorInteiro
2  {
3    static void Main()
4    {
5      int numero = 2147483647;
6
7      while(numero >= 100)
8      {
9        numero /= 2;
10       System.Console.WriteLine(numero);
11     }
12   }
13 }
```

Código C# 5.17: DivideMaiorInteiro.cs

```
K19/rafael/controle-de-fluxo$ mcs DivideMaiorInteiro.java

K19/rafael/controle-de-fluxo$ mono DivideMaiorInteiro.exe
1073741823
536870911
```

```
268435455
...
```

Terminal 5.16: Compilando e executando a classe `DivideMaiorInteiro`

Exercícios Complementares

2 Escreva um programa que imprime a tabuada dos números de 1 a 10 de acordo com o padrão abaixo:

```
1x1 = 1
1x2 = 2
1x3 = 3
...
10x8 = 80
10x9 = 90
10x10 = 100
```

3 Escreva um programa que desenhe uma pirâmide de asteriscos (`*`). A saída do seu programa deve seguir o padrão abaixo:

```
   *
  ***
 *****
*******
```

4 Seguindo a idéia do desafio anterior, escreva um programa que imprime uma árvore de natal de acordo com o padrão abaixo:

```
       *
      ***
     *****
    *******
     *****
    *******
   *********
  ***********
   *********
  ***********
 *************
***************
```

CAPÍTULO 6

ARRAY

O que é um Array?

Um array ou vetor é uma estrutura de dados utilizada para armazenar uma coleção de itens. Cada item é identificado através de um índice. Podemos imaginar um array como sendo um armário com um determinado número de gavetas e cada gaveta possui um rótulo com um número de identificação.

Figura 6.1: Abstração de array como um armário com gavetas

Quando criamos um array, devemos informar qual o tipo de dado pretendemos armazenar em cada posição. Na analogia com armário, seria como se tivéssemos que definir o que o é permitido guardar em cada gaveta. Por exemplo, se definirmos que um armário deve guardar livros, então somente livros podem ser armazenados nas gavetas desse armário. Não poderemos guardar revistas ou jornais.

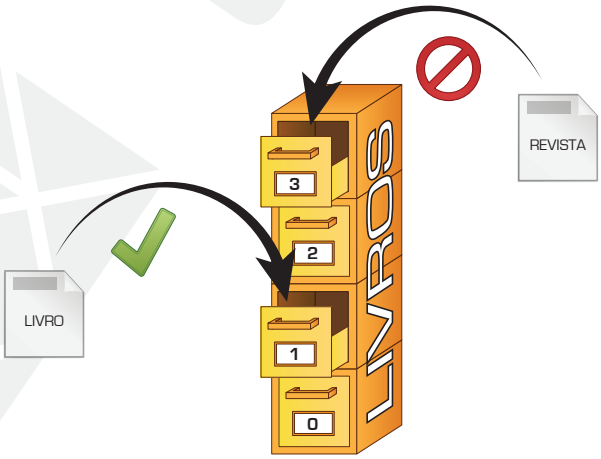

Figura 6.2: Um armário de livros não pode guardar revistas

Como declarar e inicializar um array?

Para utilizarmos um array, devemos criar uma variável para guardar a referência desse array. A declaração dessa variável é semelhante à declaracão das variáveis que vimos até agora.

```
1  int[] nomeDoArray;
```

Código Java 6.1: Declaração de um array

Lembre-se que sempre devemos inicializar as variáveis para não ocorrer um erro de compilação. Portanto, vamos inicializar o nosso array:

```
1  int[] nomeDoArray = new int[10];
```

Código Java 6.2: Declaração e inicialização de um array

A inicialização de um array se dá através da instrução `new` tanto em Java quanto em C#. No exemplo acima, criamos um array de tamanho 10, ou seja, teremos 10 posições para armazenar valores do tipo `int`. A instrução `new` é abordada com mais detalhes nos cursos K11 - Orientação a Objetos em Java e K31 - C# e Orientação a Objetos.

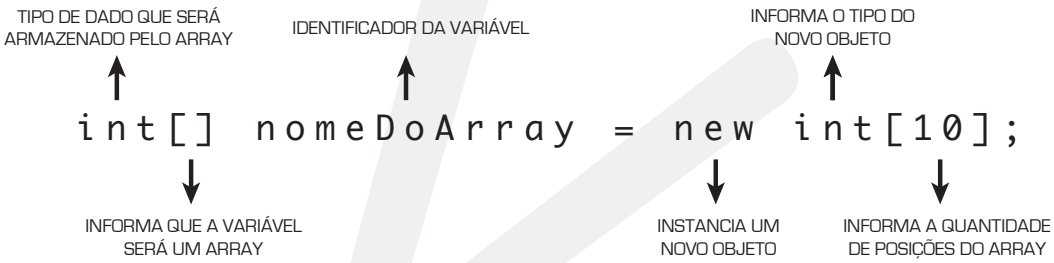

Figura 6.3: Declaração e inicialização de um array

Inserindo valores de um array

Existem diversas formas de inserirmos valores em um array. A forma mais comum é a seguinte:

```
1  int[] a = new int[3];
2  a[0] = 124;
3  a[1] = 43;
4  a[2] = 1023;
```

Código Java 6.3: Inserindo valores em um array

Na linha 1 declaramos e inicializamos um array do tipo `int` com três posições. Nas linhas 2, 3 e 4 inserimos no array os valores 124, 43 e 1023 nas posições 0, 1 e 2, respectivamente. Repare que a numeração dos índices de um array começa pelo número zero.

As outras formas de se inserir valores em um array fazem muito mais do que simplesmente inserir tais valores. Na verdade essas formas declaram, inicializam e inserem os valores, tudo em apenas uma linha de código.

```
1 int[] b = new int[] {1, 62, 923, 15};
2 int[] c = {125, 76432, 23};
```

Código Java 6.4: Outras formas de se inserir valores em um array

```
1 int[] b = new int[] {1, 62, 923, 15};
2 int[] c = {125, 76432, 23};
3 int[] d = new int[2] {634, 5};
```

Código C# 6.1: Outras formas de se inserir valores em um array

Repare que, no momento da criação dos arrays acima, os valores de cada posição devem ser definidos. Já na primeira forma apresentada, esses valores poderiam ser definidos depois.

Acessando os valores de um array

Para acessarmos o valor armazenado em uma das posições de um array, basta conhecermos o índice de tal posição. Veja o exemplo abaixo:

```
1 int[] a = new int[] {3215, 754, 23};
2
3 System.out.println("Valor na posição de índice 0: " + a[0]);
4 System.out.println("Valor na posição de índice 2: " + a[2]);
```

Código Java 6.5: Acessando os valores de um array

```
1 int[] a = new int[] {3215, 754, 23};
2
3 System.Console.WriteLine("Valor na posição de índice 0: " + a[0]);
4 System.Console.WriteLine("Valor na posição de índice 2: " + a[2]);
```

Código C# 6.2: Acessando os valores de um array.

Lembre-se

No exemplo acima criamos um array de três posições. O que aconteceria se tentássemos acessar a posição `a[3]`, por exemplo?

A tentativa de acessar uma posição que não existe causaria um erro de execução, ou seja, esse erro só seria identificado no momento em que seu programa estivesse em execução. Esse tipo de erro não é detectado no momento da compilação. Em Java seria lançada uma exceção do tipo `java.lang.ArrayIndexOutOfBoundsException` e em C# seria lançada uma exceção do tipo `System.IndexOutOfRangeException`.

Percorrendo um array

Quando trabalhamos com arrays, uma das tarefas mais comuns é acessarmos todas ou algumas de suas posições sistematicamente. Geralmente fazemos isso para resgatar todos ou alguns dos valores armazenados e realizar algum processamento sobre tais valores.

Para percorrermos um array utilizaremos a instrução de repetição `for`. Podemos utilizar a instrução `while` também, porém logo perceberemos que a sintaxe da instrução `for` é mais apropriada quando estamos trabalhando com arrays.

```
1  int[] numeros = new int[100];
2
3  for(int i = 0; i < 100; i++){
4    numeros[i] = i*2;
5  }
6
7  for(int i = 0; i < 100; i++){
8    System.out.println(numeros[i]);
9  }
```

Código Java 6.6: Percorrendo um array para inserir e acessar valores

```
1  int[] numeros = new int[100];
2
3  for(int i = 0; i < 100; i++){
4    numeros[i] = i*2;
5  }
6
7  for(int i = 0; i < 100; i++){
8    System.Console.WriteLine(numeros[i]);
9  }
```

Código C# 6.3: Percorrendo um array para inserir e acessar valores

Imagine que exista uma grande quantidade de linhas de código entre as linhas destacadas no exemplo acima, ou seja, entre a declaração e inicialização do array `numeros` e o `for` que o percorre. Além disso, imagine também que o código teve que ser modificado, mais especificamente na inicialização do array `numeros`. Agora o array passará a ter 1000 posições. Seria muito fácil esquecermos de atualizar a instrução `for` e informá-la de que o contador `i` deverá percorrer o intervalo de 0 a 1000 e não de 0 a 100.

Para evitar esse tipo de problema, uma boa prática é utilizar o atributo `length` dos arrays da linguagem Java ou a propriedade `Length` dos arrays da linguagem C# para descobrirmos qual a sua dimensão (tamanho). Veja como ficaria o exemplo acima utilizando o atributo `length`:

```
1  int[] numeros = new int[100];
2
3  for(int i = 0; i < numeros.length; i++){
4    numeros[i] = i*2;
5  }
6
7  for(int i = 0; i < numeros.length; i++){
8    System.out.println(numeros[i]);
9  }
```

Código Java 6.7: Utilizando o atributo `length` do array

```
1  int[] numeros = new int[100];
2
3  for(int i = 0; i < numeros.Length; i++){
4    numeros[i] = i*2;
5  }
6
7  for(int i = 0; i < numeros.Length; i++){
8    System.Console.WriteLine(numeros[i]);
9  }
```

Código C# 6.4: Utilizando a propriedade `Length` do array

Array de arrays

Até agora trabalhamos com arrays de uma dimensão. Porém, tanto me Java como em C#, podemos criar arrays com mais de uma dimensão (arrays multidimensionais). Isso nos permite trabalhar com arrays para representar tabelas, matrizes ou até um tabuleiro de batalha naval. Voltando à analogia que fizemos com um armário cheio de gavetas, seria como se pudéssemos guardar dentro da gaveta de um armário um outro armário com gavetas. Veja a figura abaixo:

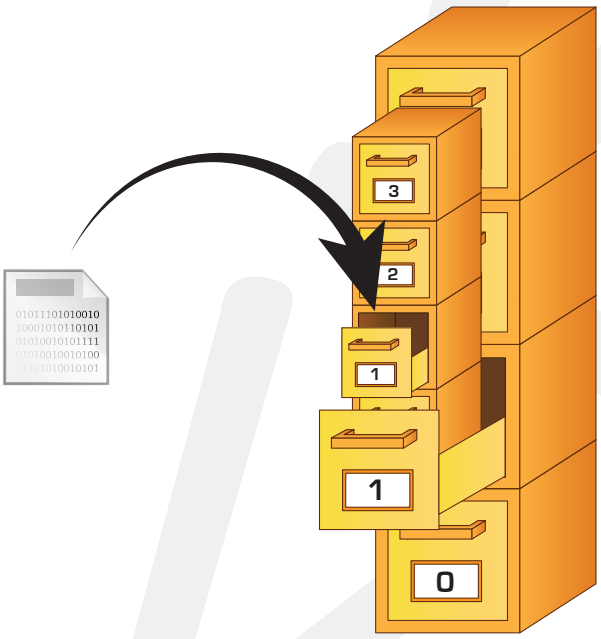

Figura 6.4: Abstração de um array multidimensional

Lembre-se

Em Java, array multidimensional é sinônimo de array de arrays. Em C#, jagged arrays é sinônimo de array de arrays. Em C# a denominação array multidimensional refere-se a um outro tipo de array. Um array multidimensional em C# é assunto do curso K31 - C# e Orientação a Objetos.

Neste curso, para evitarmos confusões, sempre utilizaremos a denominação array de arrays.

A declaração de um array de arrays é muito semelhante à declaração e inicialização de um array simples.

```
1 int[][] arrays = new int[4][];
2
3 arrays[0] = new int[1];
4 arrays[1] = new int[3];
5 arrays[2] = new int[2];
6 arrays[3] = new int[7];
```

Código Java 6.8: Declarando um array de arrays

Em cada posição do nosso array de arrays, devemos criar um novo array. Por esse motivo, ele recebe o nome array de arrays. Além disso, repare que podemos criar arrays de diferentes tamanhos em cada posição.

Assim como nos arrays unidimensionais, para inserir ou acessar valores de um array de arrays, devemos utilizar os índices de cada posição. Podemos pensar nos índices como um esquema de coordenadas. Por exemplo, se quiséssemos representar um gráfico no sistema cartesiano de eixos xy através de um array de arrays, a coordenada de cada ponto do gráfico seria equivalente ao par de índices do nosso array de arrays (supondo que no gráfico seja permitido apenas coordenadas inteiras).

```
boolean[][] pontosDoGrafico = new boolean[10][];

for(int i = 0; i < pontosDoGrafico.length; i++){
  pontosDoGrafico[i] = new boolean[10];
}

pontosDoGrafico[0][0] = true;
pontosDoGrafico[1][1] = true;
pontosDoGrafico[2][1] = true;
pontosDoGrafico[2][2] = true;
pontosDoGrafico[3][2] = true;
pontosDoGrafico[4][1] = true;
```

Código Java 6.9: Conjunto de pontos de um gráfico de eixos xy armazenados em um array de arrays

```
bool[][] pontosDoGrafico = new bool[10][];

for(int i = 0; i < pontosDoGrafico.Length; i++)
{
  pontosDoGrafico[i] = new bool[10];
}

pontosDoGrafico[0][0] = true;
pontosDoGrafico[1][1] = true;
pontosDoGrafico[2][1] = true;
pontosDoGrafico[2][2] = true;
pontosDoGrafico[3][2] = true;
pontosDoGrafico[4][1] = true;
```

Código C# 6.5: Conjunto de pontos de um gráfico de eixos xy armazenados em um array de arrays

Percorrendo um array de arrays

Para percorrer um array de arrays, utilizaremos novamente as instruções de repetição `while` e `for`. Porém, como estamos trabalhando com arrays com mais de uma dimensão, teremos uma ou mais instruções `for` aninhadas.

```
int[][] tabelaDeNumeros = new int[5][];

for(int i = 0; i < tabelaDeNumeros.length; i++) {
  tabelaDeNumeros[i] = new int[5];
}

for(int i = 0; i < tabelaDeNumeros.length; i++) {
  for(int j = 0; j < tabelaDeNumeros[i].length; j++) {
    tabelaDeNumeros[i][j] = i*j;
  }
}
```

Código Java 6.10: Percorrendo um array de arrays com instruções `for` *aninhadas*

Ordenando um array

Uma tarefa muito comum quando estamos trabalhando com arrays é ordená-los seguindo um determinado critério. Um array de `int` pode ser ordenado do menor para o maior número.

Na computação, a análise de algoritmos estuda a complexidade de diversos algoritmos calculando-se quanto tempo um algoritmo leva para completar uma determinada tarefa, assim como a quantidade de memória necessária para execução de tal tarefa. Por exemplo, quando estamos aprendendo a analisar um algoritmo os exemplos mais frequentes são os de ordenação de arrays. Existem diversos algoritmos de ordenação amplamente estudados e cada um com suas vantagens e desvantagens. Neste curso abordaremos dois desses algoritmos, o **Selection Sort** e o **Bubble Sort**.

Selection Sort

Começaremos por este algoritmo, pois logo perceberemos que, provavelmente, trata-se da abordagem mais intuitiva. A idéia do *Selection Sort* é procurar o menor valor e colocá-lo na primeira posição do array. Em seguida, procurar o segundo menor valor e colocá-lo na segunda posição. Devemos repetir a operação até que o n-ésimo menor valor seja colocado na n-ésima posição.

```
1  int[] array; // suponha que temos este array inicializado e com todas as posições ↩
       preenchidas
2  int auxiliar = 0;
3  int indiceDoMenor = 0;
4
5  for (int i = 0; i < array.length; i++) {
6      indiceDoMenor = i;
7
8      for (int j = i + 1; j < array.length; j++) {
9      if (array[j] < array[indiceDoMenor]) {
10       indiceDoMenor = j;
11     }
12     }
13
14     if (indiceDoMenor != i) {
15     auxiliar = array[indiceDoMenor];
16     array[indiceDoMenor] = array[i];
17     array[i] = auxiliar;
18     }
19 }
```

Código Java 6.11: Algoritmo de ordenação - Selection Sort

```
1  int[] array; // suponha que temos este array inicializado e com as posições ↩
       preenchidas
2  int auxiliar = 0;
3  int indiceDoMenor = 0;
4
5  for (int i = 0; i < array.Length; i++)
6  {
7      indiceDoMenor = i;
8
9      for (int j = i + 1; j < array.Length; j++)
```

```
10      {
11      if (array[j] < array[indiceDoMenor])
12      {
13        indiceDoMenor = j;
14      }
15      }
16
17      if (indiceDoMenor != i)
18      {
19      auxiliar = array[indiceDoMenor];
20      array[indiceDoMenor] = array[i];
21      array[i] = auxiliar;
22      }
23  }
```

Código C# 6.6: Algoritmo de ordenação - Selection Sort

Bubble Sort

A idéia do *Bubble Sort* é percorrermos o array diversas vezes "empurrando" o maior valor para uma posição maior até que ele chegue na última. Depois, o segundo maior valor até que ele chegue na penúltima posição e assim por diante. O nome *Bubble Sort* foi dado devido ao fato dos maiores valores de cada posição do array irem "subindo" de posição, lembrando bolhas na água.

```
1   int[] array; // suponha que temos este array inicializado e com todas as posições ↩
        preenchidas
2
3   for (int i = array.length; i >= 1; i--) {
4     for (int j = 1; j < i; j++) {
5       if (array[j - 1] > array[j]) {
6         int auxiliar = array[j];
7
8         array[j] = array[j - 1];
9         array[j - 1] = auxiliar;
10      }
11    }
12  }
```

Código Java 6.12: Algoritmo de ordenação: Bubble Sort

```
1   int[] array; // suponha que temos este array inicializado e com todas as posições ↩
        preenchidas
2
3   for (int i = array.Length; i >= 1; i--)
4   {
5     for (int j = 1; j < i; j++)
6     {
7       if (array[j - 1] > array[j])
8       {
9         int auxiliar = array[j];
10
11        array[j] = array[j - 1];
12        array[j - 1] = auxiliar;
13      }
14    }
15  }
```

Código C# 6.7: Algoritmo de ordenação: Bubble Sort

Exercícios de Fixação

1 Abra um terminal; Entre na pasta dos seus exercícios e crie uma pasta chamada **arrays** para os arquivos desenvolvidos nesse capítulo.

```
K19$ cd rafael

K19/rafael$ ls
controle-de-fluxo introducao operadores variaveis

K19/rafael$ mkdir controle-de-fluxo

K19/rafael$ ls
arrays controle-de-fluxo introducao operadores variaveis
```

Terminal 6.1: Criando a pasta `arrays`

2 Crie um programa em Java que armazene 10 números inteiros em um array. Todas as posições do array devem ser preenchidas e o valor armazenado fica à sua escolha. Após preencher o array, imprima os seus valores na tela.

```
1  class TestaArray {
2    public static void main(String[] args) {
3      int[] array = new int[10];
4
5      array[0] = 57;
6      array[1] = 436;
7      array[2] = 724;
8      array[3] = 564;
9      array[4] = 245;
10     array[5] = 47;
11     array[6] = 34;
12     array[7] = 1;
13     array[8] = 347735;
14     array[9] = 83;
15
16     for(int i = 0; i < array.length; i++) {
17       System.out.println(array[i]);
18     }
19   }
20 }
```

Código Java 6.13: TestaArray.java

```
K19/rafael/arrays$ javac TestaArray.java

K19/rafael/arrays$ java TestaArray
57
436
724
564
245
47
34
1
347735
83
```

Terminal 6.2: Compilando e executando a classe `TestaArray`

3 Implemente o mesmo programa do exercício anterior utilizando a linguagem C#.

```
1  class TestaArray
2  {
3    static void Main()
4    {
5      int[] array = new int[10];
6
```

```
7       array[0] = 57;
8       array[1] = 436;
9       array[2] = 724;
10      array[3] = 564;
11      array[4] = 245;
12      array[5] = 47;
13      array[6] = 34;
14      array[7] = 1;
15      array[8] = 347735;
16      array[9] = 83;
17
18      for(int i = 0; i < array.Length; i++)
19      {
20        System.Console.WriteLine(array[i]);
21      }
22    }
23  }
```

Código C# 6.8: TestaArray.cs

```
K19/rafael/arrays$ mcs TestaArray.cs

K19/rafael/arrays$ mono TestaArray.exe
57
436
724
564
245
47
34
1
347735
83
```

Terminal 6.3: Compilando e executando a classe `TestaArray`

4 Crie um programa em Java que armazene 10 números inteiros em um array. Preencha todas as posições do array com valores sequenciais e em seguida imprima-os na tela. Depois, escolha duas posições aleatoriamente e troque os valores de uma posição pelo da outra. Repita essa operação 10 vezes. Ao final, imprima o array novamente.

Dica: para visualizar melhor o resultado, imprima uma linha entre a primeira e a segunda impressão do array.

```
1   class EmbaralhaArray {
2     public static void main(String[] args) {
3       int[] array = new int[10];
4       java.util.Random geradorDeNumeroAleatorio = new java.util.Random();
5
6       for (int i = 0; i < array.length; i++) {
7         array[i] = i;
8       }
9
10      for (int i = 0; i < array.length; i++) {
11        System.out.println(array[i]);
12      }
13
14      for (int i = 0; i < 10; i++) {
15        int posicao1 = geradorDeNumeroAleatorio.nextInt(10);
16        int posicao2 = geradorDeNumeroAleatorio.nextInt(10);
17        int auxiliar = array[posicao1];
18
19        array[posicao1] = array[posicao2];
20        array[posicao2] = auxiliar;
21      }
22
23      System.out.println("--------------------");
```

```
24
25      for (int i = 0; i < array.length; i++) {
26        System.out.println(array[i]);
27      }
28    }
29  }
```

Código Java 6.14: EmbaralhaArray.java

```
K19/rafael/arrays$ javac EmbaralhaArray.java

K19/rafael/arrays$ java EmbaralhaArray
```

Terminal 6.4: Compilando e executando a classe `EmbaralhaArray`

5 Implemente o mesmo programa do exercício anterior utilizando a linguagem C#.

```
1   class EmbaralhaArray
2   {
3     static void Main()
4     {
5       int[] array = new int[10];
6       System.Random geradorDeNumeroAleatorio = new System.Random();
7
8       for(int i = 0; i < array.Length; i++)
9       {
10        array[i] = i;
11      }
12
13      for(int i = 0; i < array.Length; i++)
14      {
15        System.Console.WriteLine(array[i]);
16      }
17
18      for(int i = 0; i < 10; i++)
19      {
20        int posicao1 = geradorDeNumeroAleatorio.Next(0, 11);
21        int posicao2 = geradorDeNumeroAleatorio.Next(0, 11);
22        int auxiliar = array[posicao1];
23
24        array[posicao1] = array[posicao2];
25        array[posicao2] = auxiliar;
26      }
27
28      System.Console.WriteLine("--------------------");
29
30      for(int i = 0; i < array.Length; i++)
31      {
32        System.Console.WriteLine(array[i]);
33      }
34    }
35  }
```

Código Java 6.15: EmbaralhaArray.cs

```
K19/rafael/arrays$ mcs EmbaralhaArray.cs

K19/rafael/arrays$ mono EmbaralhaArray
```

Terminal 6.5: Compilando e executando a classe `EmbaralhaArray`

6 Crie um programa em Java que armazene 10 números inteiros em um array. Preencha todas as posições do array com valores aleatórios e em seguida imprima-os na tela. Após imprimir o array, ordene o array do menor valor para o maior. Ao final, imprima o array ordenado.

```java
class OrdenaArray {
  public static void main(String[] args) {
    int[] array = new int[10];
    java.util.Random geradorDeNumeroAleatorio = new java.util.Random();

    for (int i = 0; i < array.length; i++) {
      array[i] = geradorDeNumeroAleatorio.nextInt(100);
    }

    for (int i = 0; i < array.length; i++) {
      System.out.println(array[i]);
    }

    for (int i = array.length; i >= 1; i--) {
      for (int j = 1; j < i; j++) {
        if (array[j - 1] > array[j]) {
          int auxiliar = array[j];

          array[j] = array[j - 1];
          array[j - 1] = auxiliar;
        }
      }
    }

    System.out.println("--------------------");

    for (int i = 0; i < array.length; i++) {
      System.out.println(array[i]);
    }
  }
}
```

Código Java 6.16: OrdenaArray.java

```
K19/rafael/arrays$ javac OrdenaArray.java

K19/rafael/arrays$ java OrdenaArray
```

Terminal 6.6: Compilando e executando a classe `OrdenaArray`

7 Implemente o mesmo programa do exercício anterior utilizando a linguagem C#.

```csharp
class OrdenaArray
{
  static void Main()
  {
    int[] array = new int[10];
    System.Random geradorDeNumeroAleatorio = new System.Random();

    for(int i = 0; i < array.Length; i++)
    {
      array[i] = geradorDeNumeroAleatorio.Next(0, 100);
    }

    for(int i = 0; i < array.Length; i++)
    {
      Console.WriteLine(array[i]);
    }

    for (int i = array.Length; i >= 1; i--)
    {
      for (int j = 1; j < i; j++)
      {
        if (array[j - 1] > array[j])
        {
          int auxiliar = array[j];

```

```
26            array[j] = array[j - 1];
27            array[j - 1] = auxiliar;
28          }
29        }
30      }
31
32      System.Console.WriteLine("--------------------");
33
34      for(int i = 0; i < array.Length; i++)
35      {
36        Console.WriteLine(array[i]);
37      }
38    }
39  }
```

Código Java 6.17: OrdenaArray.cs

```
K19/rafael/arrays$ mcs OrdenaArray.cs

K19/rafael/arrays$ mono OrdenaArray
```

Terminal 6.7: Compilando e executando a classe `OrdenaArray`

8 Crie um programa em Java que utilize arrays para representar o estado inicial de um tabuleiro do jogo *Damas*. O tabuleiro possui 64 casas dispostas no formato de um quadrado de 8x8 casas.

Observe a figura abaixo para ajudá-lo a abstrair o tabuleiro.

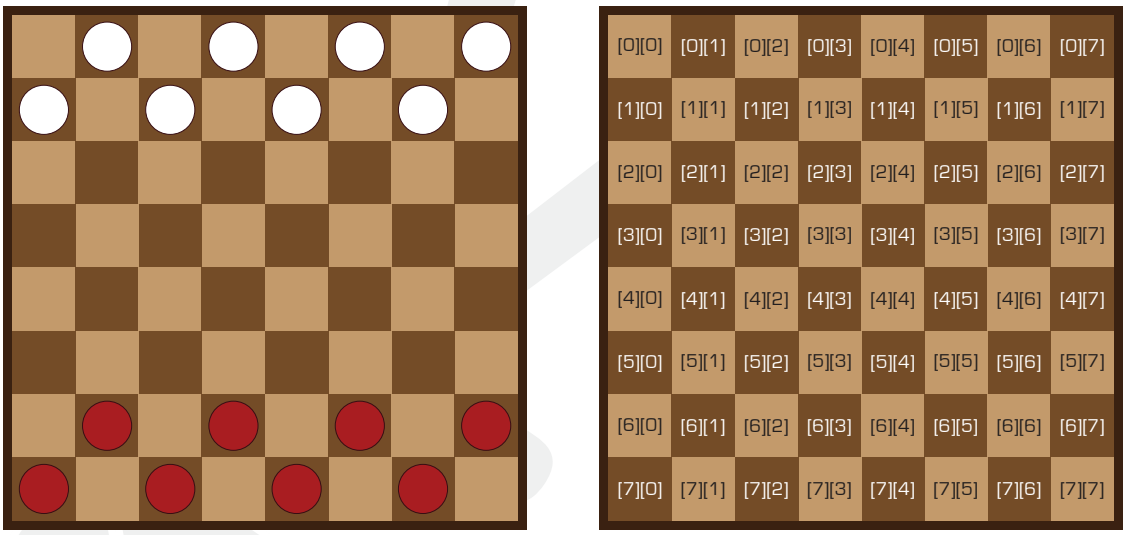

Figura 6.5: Tabuleiro de Damas e sua abstração em um array de arrays

No array utilize os caracteres * (asterisco) para representar as pedras vermelhas e 0 (zero) para representar as brancas. Imprima o tabuleiro conforme o exemplo abaixo:

```
---------------------------------
|   | 0 |   | 0 |   | 0 |   | 0 |
---------------------------------
| 0 |   | 0 |   | 0 |   | 0 |   |
---------------------------------
|   |   |   |   |   |   |   |   |
```

```
---------------------------------
|   |   |   |   |   |   |   |   |
---------------------------------
|   |   |   |   |   |   |   |   |
---------------------------------
|   |   |   |   |   |   |   |   |
---------------------------------
|   | * |   | * |   | * |   | * |
---------------------------------
| * |   | * |   | * |   | * |   |
---------------------------------
```

```
1  class Damas {
2    public static void main(String[] args) {
3      char[][] tabuleiro = new char[8][];
4
5      for (int i = 0; i < tabuleiro.length; i++) {
6        tabuleiro[i] = new char[8];
7      }
8
9      for (int i = 0; i < tabuleiro.length; i++) {
10       for (int j = 0; j < tabuleiro[i].length; j++) {
11         if ((i == 0 && j % 2 == 1) || (i == 1 && j % 2 == 0)) {
12           tabuleiro[i][j] = '0';
13         } else if ((i == 6 && j % 2 == 1) || (i == 7 && j % 2 == 0)) {
14           tabuleiro[i][j] = '*';
15         } else {
16           tabuleiro[i][j] = ' ';
17         }
18       }
19     }
20
21     System.out.println("---------------------------------");
22
23     for (int i = 0; i < tabuleiro.length; i++) {
24       String linha = "|";
25
26       for (int j = 0; j < tabuleiro[i].length; j++) {
27         linha += " " + tabuleiro[i][j] + " |";
28       }
29
30       System.out.println(linha);
31       System.out.println("---------------------------------");
32     }
33   }
34 }
```

Código Java 6.18: Damas.java

```
K19/rafael/arrays$ javac Damas.java

K19/rafael/arrays$ java Damas
```

Terminal 6.8: Compilando e executando a classe `Damas`

9 Implemente o mesmo programa do exercício anterior utilizando a linguagem C#.

```
1  class Damas
2  {
3    public void Main()
4    {
5      char[][] tabuleiro = new char[8][];
```

```
6
7       for(int i = 0; i < tabuleiro.Length; i++)
8       {
9         tabuleiro[i] = new char[8];
10      }
11
12      for(int i = 0; i < tabuleiro.Length; i++)
13      {
14        for(int j = 0; j < tabuleiro[i].Length; j++)
15        {
16          if((i == 0 && j%2 == 1) || (i == 1 && j%2 == 0))
17          {
18            tabuleiro[i][j] = '0';
19          }
20          else if((i == 6 && j%2 == 1) || (i == 7 && j%2 == 0))
21          {
22            tabuleiro[i][j] = '*';
23          }
24          else
25          {
26            tabuleiro[i][j] = ' ';
27          }
28        }
29      }
30
31      System.ConsoleWriteLine("---------------------------------");
32
33      for(int i = 0; i < tabuleiro.Length; i++)
34      {
35        string linha = "|";
36
37        for(int j = 0; j < tabuleiro[i].Length; j++)
38        {
39          linha += " " + tabuleiro[i][j] + " |";
40        }
41
42        System.ConsoleWriteLine(linha);
43        System.ConsoleWriteLine("---------------------------------");
44      }
45    }
46  }
```

Código Java 6.19: Damas.cs

```
K19/rafael/arrays$ mcs Damas.cs

K19/rafael/arrays$ mono Damas.exe
```

Terminal 6.9: Compilando e executando a classe `Damas`

Desafios

1 Neste desafio iremos escrever o jogo "Batalha Naval". Abaixo segue a definição do jogo no site *Wikipedia*:

> "Batalha naval é um jogo de tabuleiro de dois jogadores, no qual os jogadores têm de adivinhar em que quadrados estão os navios do oponente. Embora o primeiro jogo em tabuleiro comercializado e publicado pela Milton Bradley Company em 1931, o jogo foi originalmente jogado com lápis e papel.
>
> O jogo original é jogado em duas grelhas para cada jogador - uma que representa a disposição dos barcos do jogador, e outra que representa a do oponente. As grelhas são

tipicamente quadradas, estando identificadas na horizontal por números e na vertical por letras. Em cada grelha o jogador coloca os seus navios e registra os tiros do oponente.

Antes do início do jogo, cada jogador coloca os seus navios nos quadros, alinhados horizontalmente ou verticalmente. O número de navios permitidos é igual para ambos jogadores e os navios não podem se sobrepor.

Após os navios terem sido posicionados o jogo continua numa série de turnos, em cada turno um jogador diz um quadrado na grelha do oponente, se houver um navio nesse quadrado, é colocada uma marca vermelha, senão houver é colocada uma marca branca.

Os tipos de navios são: porta-aviões (5 quadrados adjacentes em forma de T), os submarinos (1 quadrado apenas), barcos de dois, três e quatro canos. Numa das variações deste jogo, as grelhas são de dimensão 10x10, e o número de navios são: 1, 4, 3, 2, 1, respectivamente."

Para facilitar um pouco a implementação do nosso jogo, vamos alterar levemente as especificações do jogo. Em nossa versão, teremos apenas navios do tipo submarino e cada jogador poderá posicionar em sua grelha dez submarinos e o programa aceitará apenas um jogador, pois o oponente será o computador. Além disso, para não precisarmos utilizar cores no terminal, quando um navio é acertado (marca vermelha), marcaremos a posição com o caractere * (asterisco). Caso contrário (marca branca), marcaremos com o caractere - (traço). As posições que possuem um navio devem ser marcadas com o caractere "N".

Em cada turno, a situação atual do tabuleiro do jogador deverá ser impressa na tela de acordo com o modelo abaixo:

```
-------------------------------------------
                JOGADOR
-------------------------------------------
|   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
-------------------------------------------
| A | N |   |   |   |   |   |   |   |   |   |
-------------------------------------------
| B |   |   |   | - |   | * |   |   |   | N |
-------------------------------------------
| C |   |   | N |   |   |   |   |   | - |   |
-------------------------------------------
| D |   |   |   |   | - |   |   |   |   |   |
-------------------------------------------
| E |   |   |   |   |   |   | N |   |   |   |
-------------------------------------------
| F |   |   | * |   |   |   |   |   |   |   |
-------------------------------------------
| G |   |   |   |   |   |   |   |   | N |   |
-------------------------------------------
| H |   |   |   |   |   |   | N |   |   |   |
-------------------------------------------
| I | N |   |   | - |   |   |   |   |   |   |
-------------------------------------------
| J |   |   |   |   |   |   | N |   |   |   |
```

Ao final do jogo, devemos indicar o vencedor e imprimir os dois tabuleiros.

Dica 1: leia o apêndice "Leitura do Teclado" para saber como receber dados digitados no teclado.

Dica 2: para forçar o encerramento do programa, pressione as teclas "CTRL" e "C".

LEITURA DO TECLADO

APÊNDICE A

Ao longo do curso, todos os programas desenvolvidos nos exemplos e exercícios agiram de forma "passiva", ou seja, não havia interação com o usuário que estava executando o programa. Existem diversas formas de um programa interagir com o mundo externo e as formas mais comuns são através do mouse e teclado de um computador. Neste apêndice iremos mostrar como receber informações do teclado nas linguagens Java e C#.

Leitura do teclado em Java

A linguagem Java possui uma classe que pode receber informações de diversas fontes diferentes, inclusive do teclado. Essa classe chama-se `InputStream`. Para que um programa possa utilizar essa classe, devemos importá-la através da instrução `import`:

```
1  import java.io.InputStream;
2
3  class ExemploTeclado{
4    public static void main(String[] args){
5      InputStream entrada;
6      // ...
7      // ...
8      // ...
9    }
10 }
```

Código Java A.1: Importando a classe `InputStream`.

Como a classe InputStream sabe trabalhar com diversas fontes de entrada de dados, antes de utilizá-la devemos informar qual será a fonte de entrada de dados. No nosso caso, estamos interessados em receber dados do teclado, portanto devemos utilizar o atributo estático `in` da classe `System`.

```
1  import java.io.InputStream;
2
3  class ExemploTeclado{
4    public static void main(String[] args){
5      InputStream entrada = System.in;
6      // ...
7      // ...
8      // ...
9    }
10 }
```

Código Java A.2: Passando o `System.in` *como fonte de dados de entrada de um* `InputStream`.

Com isso já seríamos capazes de trabalhar com a leitura de dados do teclado, porém em Java podemos utilizar a classe `Scanner` para facilitar a leitura de dados de uma fonte. Antes de utilizarmos a classe, devemos importá-la. Vamos ver como utilizar a classe `Scanner` para a leitura do teclado:

```
1  import java.io.InputStream;
```

```
2  import java.util.Scanner;
3
4  class ExemploTeclado{
5    public static void main(String[] args){
6      InputStream entrada = System.in;
7      Scanner scanner = new Scanner(entrada);
8      // ...
9      // ...
10     // ...
11   }
12 }
```

Código Java A.3: Utilizando a classe `Scanner` *para a leitura de dados do teclado.*

Repare que na inicialização da variável `scanner`, instanciamos um novo objeto da classe `Scanner` e passamos como parâmetro do construtor a variável `entrada`. Nesse momento informamos ao programa que queremos que o `Scanner` trabalhe com dados provenientes do teclado.

A classe scanner provê alguns métodos para acessar os dados obtidos de uma determinada fonte. Um desses métodos é o `nextLine` que, no caso da leitura do teclado, aguarda o usuário digitar algo e pressionar a tecla *Enter*. Ao pressionar a tecla *Enter*, o método nos devolve uma `String` com o valor daquilo que foi digitado. Veja um exemplo:

```
1  import java.io.InputStream;
2  import java.util.Scanner;
3
4  class ExemploTeclado{
5    public static void main(String[] args){
6      InputStream entrada = System.in;
7      Scanner scanner = new Scanner(entrada);
8
9      String teste = scanner.nextLine();
10     System.out.println(teste);
11     // ...
12     // ...
13     // ...
14   }
15 }
```

Código Java A.4: Exemplo de utilização do método `nextLine` *da classe* `Scanner`.

Na linha em destaque o programa ficará aguardando até que o usuário digite algo no teclado e pressione a tecla *Enter*. Ainda nessa linha, o programa irá atribuir à variável `teste` o valor devolvido pelo método `nextLine` (aquilo que o usuário digitou). Vamos supor que o usuário tenha digitado "123 testando". Ao executarmos o comando `System.out.println(teste)`, a mensagem `123 testando` deverá aparecer na tela do usuário.

Leitura do teclado em C#

Assim como em Java, a linguagem C# possui uma classe que permite receber dados do teclado. A classe chama-se `Console` e possui um método chamado `ReadLine` que aguarda o usuário digitar algo e pressionar a tecla *Enter*. Ao pressionar a tecla *Enter*, o método nos devolve uma `string` com o valor daquilo que foi digitado. Veja um exemplo:

```
1  using System;
2
3  class ExemploTeclado{
4    static void Main(){
5      string teste = Console.ReadLine();
6      Console.WriteLine(teste);
7      // ...
8      // ...
9      // ...
10   }
11 }
```

Código C# A.1: Exemplo de utilização do método `ReadLine` *da classe* `Console`.

Perceba que em C# a classe `Console` faz a tarefa das classes `InputStream` e `Scanner` do Java. A classe `Console` trabalha especificamente com a *entrada padrão*, que em nosso caso é o teclado. Já as classes `InputStream` e `Scanner` do Java, são mais genéricas.

RESPOSTAS

APÊNDICE B

Resposta do Complementar 1.1

Crie um arquivo chamado **HelloWorld2.java** na pasta `introducao`. Depois utilize o terminal para compilar e executar.

```
1  class HelloWorld2 {
2    public static void main(String[] args) {
3      System.out.println("Hello World 1");
4      System.out.println("Hello World 2");
5    }
6  }
```

Código Java 1.4: HelloWorld2.java

```
K19/rafael/introducao$ ls
HelloWorld.class HelloWorld.cs HelloWorld.exe HelloWorld.java HelloWorld2.java

K19/rafael/introducao$ javac HelloWorld2.java

K19/rafael/introducao$ ls
HelloWorld.class HelloWorld2.class HelloWorld.cs HelloWorld.exe HelloWorld.java HelloWorld2.java

K19/rafael/introducao$ java HelloWorld2
Hello World 1
Hello World 2
```

Terminal 1.13: Compilando e Executando

Resposta do Complementar 1.2

Crie um arquivo chamado **HelloWorld2.cs** na pasta `introducao`. Depois utilize o terminal para compilar e executar.

```
1  class HelloWorld2
2  {
3    static void Main()
4    {
5      System.Console.WriteLine("Hello World 1");
6      System.Console.WriteLine("Hello World 2");
7    }
8  }
```

Código C# 1.3: HelloWorld2.cs

```
K19/rafael/introducao$ ls
HelloWorld.class HelloWorld.cs HelloWorld.exe HelloWorld2.java
HelloWorld2.class HelloWorld2.cs HelloWorld.java

K19/rafael/introducao$ mcs HelloWorld2.java

K19/rafael/introducao$ ls
HelloWorld.class HelloWorld.cs HelloWorld.exe HelloWorld.java
HelloWorld2.class HelloWorld2.cs HelloWorld2.exe HelloWorld2.java
```

```
K19/rafael/introducao$ mono HelloWorld2
Hello World 1
Hello World 2
```

Terminal 1.14: Compilando e Executando

Resposta do Exercício 2.1

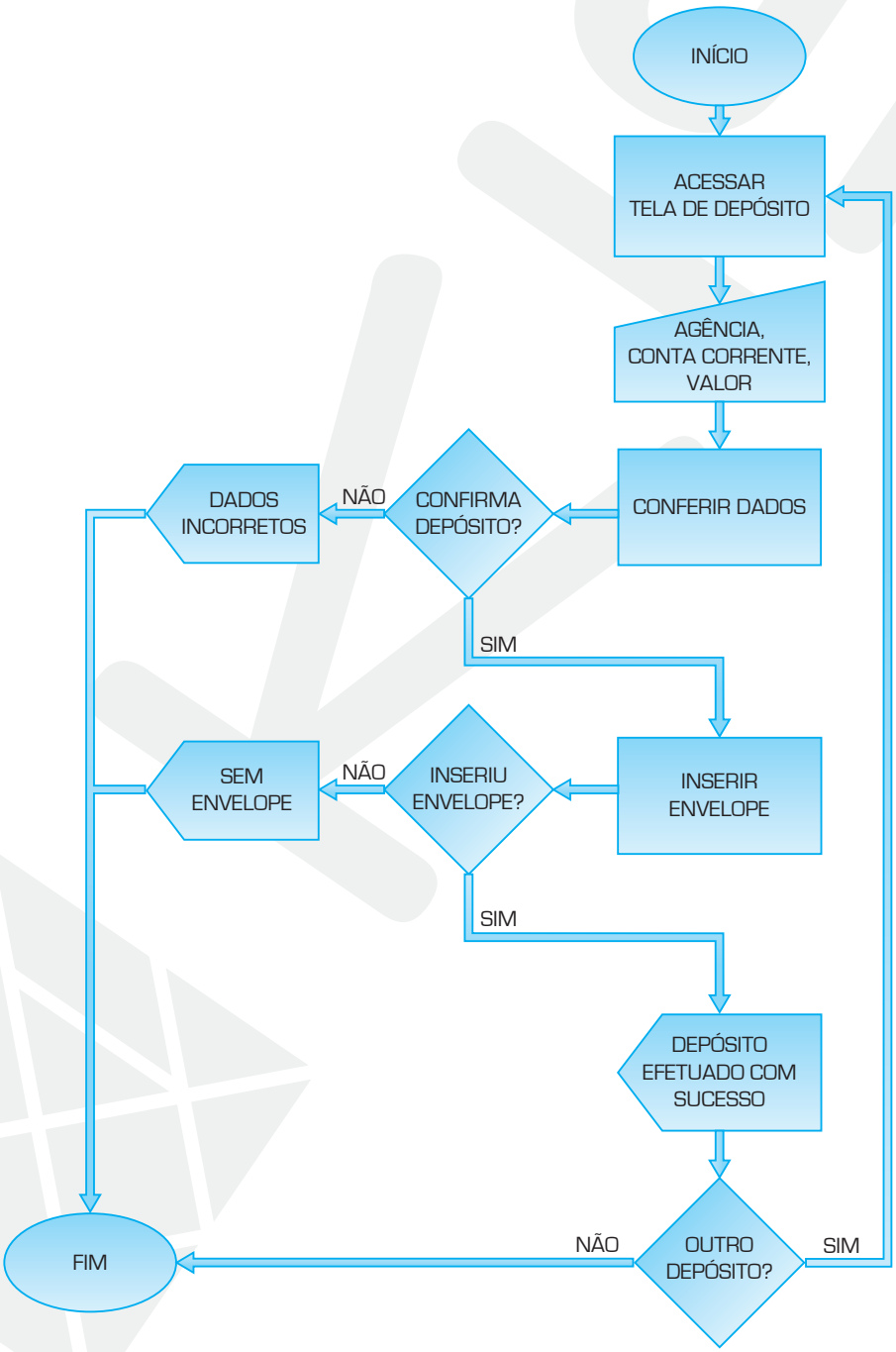

Figura 2.4: Resolução do exercício.

Resposta do Exercício 2.2

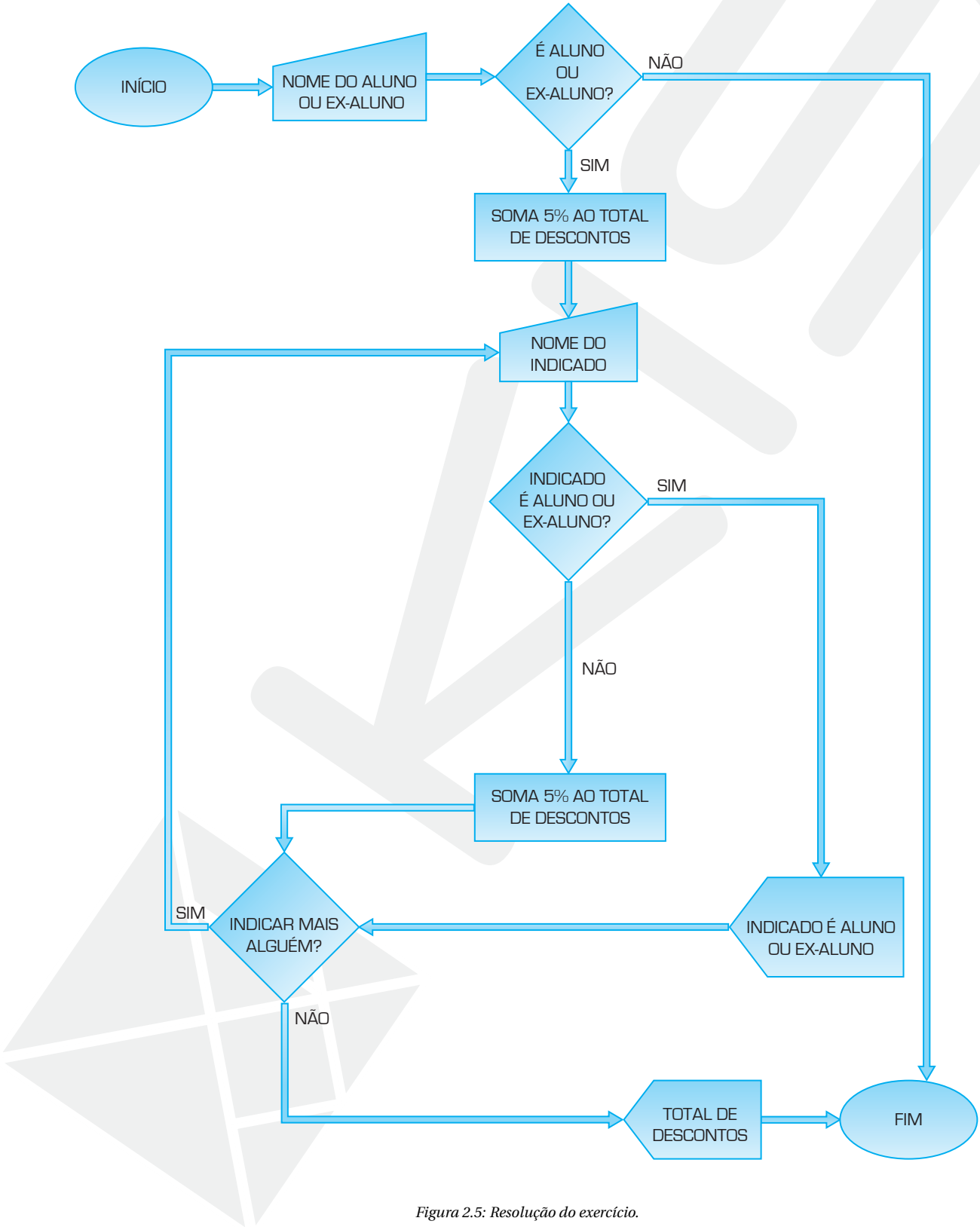

Figura 2.5: Resolução do exercício.

Resposta do Desafio 2.1

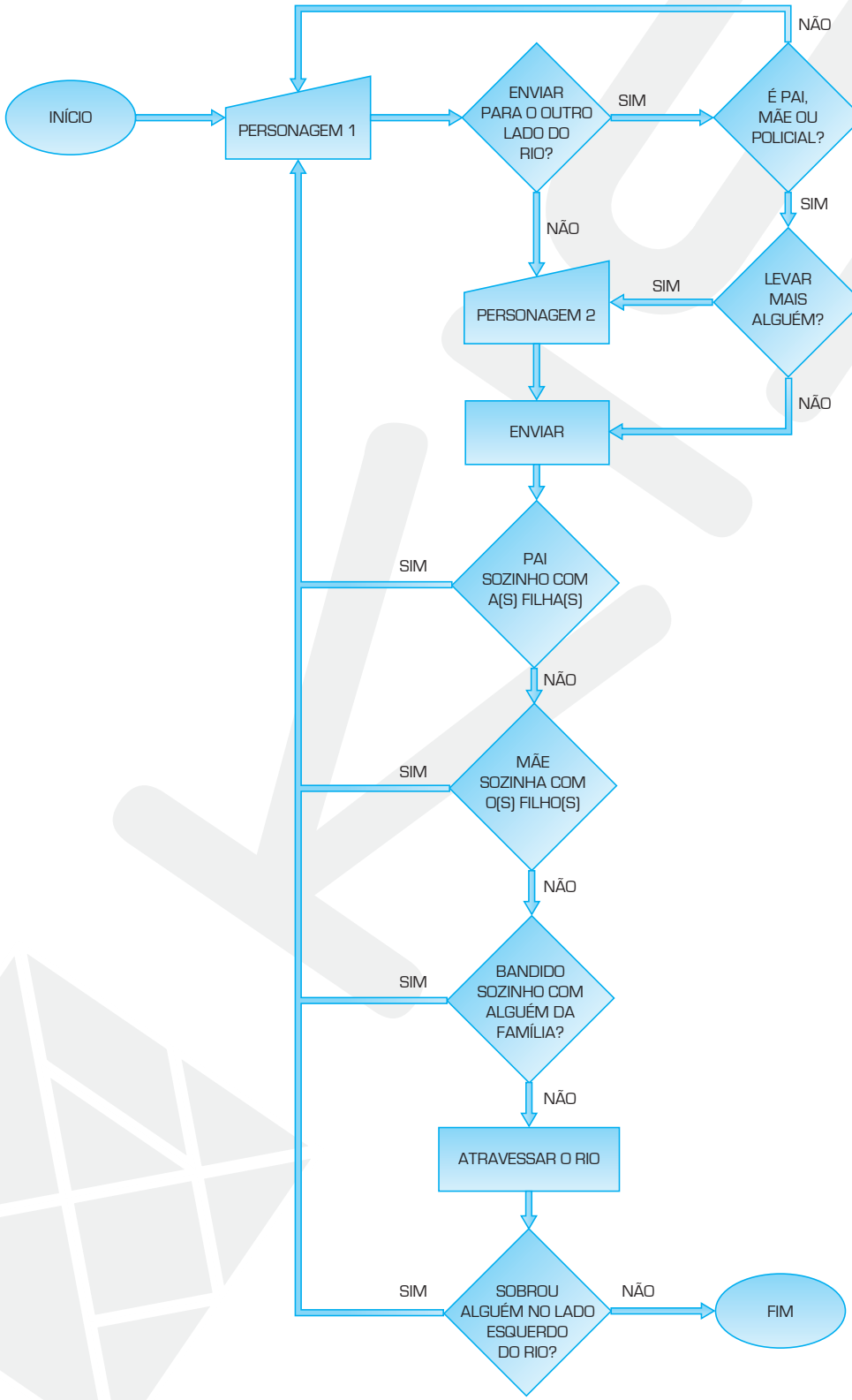

Figura 2.6: Resolução do desafio.

Resposta do Complementar 3.1

Em Java:

1. “Bom dia” -> `String`
2. 3 -> `byte`, `short`, `int` ou `long`
3. 235.13 -> `double`
4. true -> `boolean`
5. -135 -> `short`, `int` ou `long`
6. 256.23F -> `float`
7. ‘A’ -> `char`
8. 6463275245745L -> `long`

Em C#:

1. “Bom dia” -> `string`
2. 3 -> `sbyte`, `byte`, `short`, `ushort`, `int`, `uint`, `long` ou `ulong`
3. 235.13 -> `double`
4. true -> `bool`
5. -135 -> `short`, `int` ou `long`
6. 256.23F -> `float`
7. ‘A’ -> `char`
8. 6463275245745L -> `long`

Resposta do Complementar 3.2

Em Java:

```
1  class TestaVariavel2 {
2    public static void main(String[] args) {
3      int numeroDoPedido;
4      int codigoDoProduto;
5      short quantidade;
6      double valorTotalDaCompra;
7    }
8  }
```

Código Java 3.6: TestaVariaveis2.java

Em C#:

```
class TestaVariavel2
{
  static void Main()
  {
    int numeroDoPedido;
    int codigoDoProduto;
    short quantidade;
    double valorTotalDaCompra;
  }
}
```

Código C# 3.4: TestaVariaveis2.cs

Resposta do Complementar 3.3

Em Java:

```
class TestaVariavel2 {
  public static void main(String[] args) {
    int numeroDoPedido = 1523;
    int codigoDoProduto = 845732;
    short quantidade = 200;
    double valorTotalDaCompra = 62373.5;
  }
}
```

Código Java 3.7: TestaVariaveis2.java

Em C#:

```
class TestaVariavel2
{
  static void Main()
  {
    int numeroDoPedido = 1523;
    int codigoDoProduto = 845732;
    short quantidade = 200;
    double valorTotalDaCompra = 62373.5;
  }
}
```

Código C# 3.5: TestaVariaveis2.cs

Resposta do Complementar 3.4

Em Java:

```
class TestaVariavel2 {
  public static void main(String[] args) {
    int numeroDoPedido = 1523;
    int codigoDoProduto = 845732;
    short quantidade = 200;
    double valorTotalDaCompra = 62373.5;

    System.out.println(numeroDoPedido);
    System.out.println(codigoDoProduto);
    System.out.println(quantidade);
    System.out.println(valorTotalDaCompra);
  }
}
```

Código Java 3.8: TestaVariaveis2.java

Em C#:

```
class TestaVariavel2
{
  static void Main()
  {
    int numeroDoPedido = 1523;
    int codigoDoProduto = 845732;
    short quantidade = 200;
    double valorTotalDaCompra = 62373.5;

    System.Console.WriteLine(numeroDoPedido);
    System.Console.WriteLine(codigoDoProduto);
    System.Console.WriteLine(quantidade);
    System.Console.WriteLine(valorTotalDaCompra);
  }
}
```

Código C# 3.6: TestaVariaveis2.cs

Resposta do Desafio 3.1

Se estivéssemos trabalhando com uma loja bem pequena, com um baixo volume de vendas, assim como uma pequena variedade de produtos, poderíamos alterar as variáveis `numeroDoPedido` e `codigoDoProduto` para o tipo `short`. Dessa forma reduziríamos em 50% a quantidade de memória necessária para armazenarmos essas variáveis.

Caso estivéssemos trabalhando com uma grande rede de lojas, o tipo mais apropriado seria `long`. Consequentemente estaríamos aumentando em 50% a quantidade de memória necessária para armazenarmos essas variáveis.

Resposta do Complementar 4.1

```
int umMaisUm = 1 + 1;
System.out.println(umMaisUm);

int tresVezesDois = 3 * 2;
System.out.println(tresVezesDois);

int quatroDivididoPor2 = 4 / 2;
System.out.println(quatroDivididoPor2);

int seisModuloCinco = 6 % 5;
System.out.println(seisModuloCinco);

int x = 7;
System.out.println(x);

x = x + 1 * 2;
System.out.println(x);

x = x - 4;
System.out.println(x);

x = x / (6 - 2 + (3*5)/(16-1));
System.out.println(x);
```

Código Java 4.10: Resolução do exercício.

```
int umMaisUm = 1 + 1;
System.Console.WriteLine(umMaisUm);

int tresVezesDois = 3 * 2;
System.Console.WriteLine(tresVezesDois);

int quatroDivididoPor2 = 4 / 2;
System.Console.WriteLine(quatroDivididoPor2);

int seisModuloCinco = 6 % 5;
System.Console.WriteLine(seisModuloCinco);

int x = 7;
System.Console.WriteLine(x);

x = x + 1 * 2;
System.Console.WriteLine(x);

x = x - 4;
System.Console.WriteLine(x);

x = x / (6 - 2 + (3*5)/(16-1));
System.Console.WriteLine(x);
```

Código C# 4.7: Resolução do exercício.

```
int valor = 1;
System.out.println(valor);

valor += 2;
System.out.println(valor);

valor -= 1;
System.out.println(valor);

valor *= 6;
System.out.println(valor);

valor /= 3;
System.out.println(valor);

valor %= 3;
System.out.println(valor);
```

Código Java 4.11: Resolução do exercício.

```
int valor = 1;
System.Console.WriteLine(valor);

valor += 2;
System.Console.WriteLine(valor);

valor -= 1;
System.Console.WriteLine(valor);

valor *= 6;
System.Console.WriteLine(valor);

valor /= 3;
System.Console.WriteLine(valor);

valor %= 3;
System.Console.WriteLine(valor);
```

Código C# 4.8: Resolução do exercício.

```
int valor = 2;
```

```
2  boolean t = false;
3  System.out.println(t);
4
5  t = (valor == 2);
6  System.out.println(t);
7
8  t = (valor != 2);
9  System.out.println(t);
10
11 t = (valor < 2);
12 System.out.println(t);
13
14 t = (valor <= 2);
15 System.out.println(t);
16
17 t = (valor > 1);
18 System.out.println(t);
19
20 t = (valor >= 1);
21 System.out.println(t);
```

Código Java 4.12: Resolução do exercício.

```
1  int valor = 2;
2  bool t = false;
3  System.Console.WriteLine(t);
4
5  t = (valor == 2);
6  System.Console.WriteLine(t);
7
8  t = (valor != 2);
9  System.Console.WriteLine(t);
10
11 t = (valor < 2);
12 System.Console.WriteLine(t);
13
14 t = (valor <= 2);
15 System.Console.WriteLine(t);
16
17 t = (valor > 1);
18 System.Console.WriteLine(t);
19
20 t = (valor >= 1);
21 System.Console.WriteLine(t);
```

Código C# 4.9: Resolução do exercício.

```
1  int valor = 30;
2  boolean teste = false;
3  System.out.println(teste);
4
5  teste = valor < 40 && valor > 20;
6  System.out.println(teste);
7
8  teste = valor < 40 && valor > 30;
9  System.out.println(teste);
10
11 teste = valor > 30 || valor > 20;
12 System.out.println(teste);
13
14 teste = valor > 30 || valor < 20;
15 System.out.println(teste);
16
17 teste = valor < 50 && valor == 30;
18 System.out.println(teste);
```

Código Java 4.13: Resolução do exercício.

```
int valor = 30;
bool teste = false;
System.Console.WriteLine(teste);

teste = valor < 40 && valor > 20;
System.Console.WriteLine(teste);

teste = valor < 40 && valor > 30;
System.Console.WriteLine(teste);

teste = valor > 30 || valor > 20;
System.Console.WriteLine(teste);

teste = valor > 30 || valor < 20;
System.Console.WriteLine(teste);

teste = valor < 50 && valor == 30;
System.Console.WriteLine(teste);
```

Código C# 4.10: Resolução do exercício.

Resposta do Complementar 4.2

	A: valor > 40	B: valor < 60	C: valor >= 70	D(A e B)	E(B ou C)	F(A ou C)
1	V	V	V	V	V	V
2	V	V	F	V	V	V
3	V	F	V	F	V	V
4	V	F	F	F	F	V
5	F	V	V	F	V	V
6	F	V	F	F	V	F
7	F	F	V	F	V	V
8	F	F	F	F	F	F

V: Verdadeiro (true)

F: Falso (false)

Figura 4.4: Resolução do exercício.

Resposta do Complementar 5.1

Altere os arquivos como mostra os códigos abaixo. Compile e execute novamente o código de cada exercício.

```
class PrecoProduto {
  public static void main(String[] args) {
    java.util.Random gerador = new java.util.Random();

    double precoDoProduto1 = gerador.nextDouble() * 1000;
    double precoDoProduto2 = gerador.nextDouble() * 1000;

    System.out.println("precoDoProduto1 = " + precoDoProduto1);
    System.out.println("precoDoProduto2 = " + precoDoProduto2);

    if (precoDoProduto1 < precoDoProduto2) {
      System.out.println("O produto 2 é o mais caro.");
    } else if(precoDoProduto2 < precoDoProduto1){
```

```
14        System.out.println("O produto 1 é o mais caro.");
15      }
16    }
17  }
```

Código Java 5.10: PrecoProduto.java

```
1   class PrecoProduto
2   {
3     static void Main()
4     {
5       System.Random gerador = new System.Random();
6
7       double precoDoProduto1 = gerador.NextDouble() * 1000;
8       double precoDoProduto2 = gerador.NextDouble() * 1000;
9
10      System.Console.WriteLine("precoDoProduto1 = " + precoDoProduto1);
11      System.Console.WriteLine("precoDoProduto2 = " + precoDoProduto2);
12
13      if (precoDoProduto1 < precoDoProduto2)
14      {
15        System.Console.WriteLine("O produto 2 é o mais caro.");
16      }
17      else if(precoDoProduto2 < precoDoProduto1)
18      {
19        System.Console.WriteLine("O produto 1 é o mais caro.");
20      }
21    }
22  }
```

Código C# 5.6: PrecoProduto.cs

```
1   class ADivisivelPorB {
2     public static void main(String[] args) {
3       java.util.Random gerador = new java.util.Random();
4
5       int a = gerador.nextInt(1000);
6       int b = gerador.nextInt(5) + 1;
7
8       System.out.println("a = " + a);
9       System.out.println("b = " + b);
10
11      if (a % b == 0) {
12        System.out.println("É divisível");
13      } else {
14        System.out.println("Não é divisível");
15      }
16    }
17  }
```

Código Java 5.11: ADivisivelPorB.java

```
1   class ADivisivelPorB
2   {
3     static void Main()
4     {
5       System.Random gerador = new System.Random();
6
7       int a = gerador.Next(1000);
8       int b = gerador.Next(5) + 1;
9
10      System.Console.WriteLine("a = " + a);
11      System.Console.WriteLine("b = " + b);
12
13      if (a % b == 0)
14      {
15        System.Console.WriteLine("É divisível");
```

```
16      }
17      else
18      {
19        System.Console.WriteLine("Não é divisível");
20      }
21    }
22  }
```

Código C# 5.7: ADivisivelPorB.cs

```
1   class Saudacao {
2     public static void main(String[] args) {
3       java.util.Random gerador = new java.util.Random();
4
5       int hora = gerador.nextInt(24);
6
7       System.out.println("hora = " + hora);
8
9       if (hora >= 0 && hora < 12) {
10        System.out.println("Bom dia");
11      } else if (hora >= 12 && hora < 18) {
12        System.out.println("Boa tarde");
13      } else if (hora >= 18 && hora < 24) {
14        System.out.println("Boa noite");
15      } else {
16        System.out.println("Unibanco 30 horas :P");
17      }
18    }
19  }
```

Código Java 5.12: Saudacao.java

```
1   class Saudacao
2   {
3     static void Main()
4     {
5       System.Random gerador = new System.Random();
6
7       int hora = gerador.Next(24);
8
9       System.Console.WriteLine("hora = " + hora);
10
11      if(hora >= 0 && hora < 12)
12      {
13        System.Console.WriteLine("Bom dia");
14      }
15      else if(hora >= 12 && hora < 18)
16      {
17        System.Console.WriteLine("Boa tarde");
18      }
19      else if(hora >= 18 && hora < 24)
20      {
21        System.Console.WriteLine("Boa noite");
22      }
23      else
24      {
25        System.Console.WriteLine("Unibanco 30 horas :P");
26      }
27    }
28  }
```

Código C# 5.8: Saudacao.cs

Resposta do Complementar 5.2

```java
class Tabuada {
  public static void main(String[] args) {
    for (int i = 1; i <= 10; i++) {
      for (int j = 1; j <= 10; j++) {
        System.out.println(i + "x" + j + " = " + i * j);
      }
    }
  }
}
```

Código Java 5.24: Resolução do exercício.

```csharp
class Tabuada
{
  static void Main()
  {
    for(int i = 1; i <= 10; i++)
    {
      for(int j = 1; j <= 10; j++)
      {
        System.Console.WriteLine(i + "x" + j + " = " + i * j);
      }
    }
  }
}
```

Código C# 5.18: Resolução do exercício.

Resposta do Complementar 5.3

```java
class Piramide {
  public static void main(String[] args) {
    int baseMaior = 7;

    for (int i = 1; i <= baseMaior; i += 2) {
      int espacos = (baseMaior - i) / 2;
      String linha = "";

      for (int j = 0; j < espacos; j++) {
        linha += " ";
      }

      for (int k = 0; k < i; k++) {
        linha += "*";
      }

      System.out.println(linha);
    }
  }
}
```

Código Java 5.25: Resolução do desafio.

```csharp
class Piramide
{
  static void Main()
  {
    int baseMaior = 7;

    for(int i = 1; i <= baseMaior; i += 2)
```

```
8       {
9         int espacos = (baseMaior - i)/2;
10        string linha = "";
11
12        for(int j = 0; j < espacos; j++)
13        {
14          linha += " ";
15        }
16
17        for(int k = 0; k < i; k++)
18        {
19          linha += "*";
20        }
21
22        System.Console.WriteLine(linha);
23      }
24    }
25  }
```

Código C# 5.19: Resolução do desafio.

Resposta do Complementar 5.4

```
1   class ArvoreNatal {
2     public static void main(String[] args) {
3       int baseMaior = 15;
4
5       for (int m = 7; m <= 15; m += 4) {
6
7         for (int i = m - 6; i <= m; i += 2) {
8           int espacos = (baseMaior - i) / 2;
9           String linha = "";
10
11          for (int j = 0; j < espacos; j++) {
12            linha += " ";
13          }
14
15          for (int k = 0; k < i; k++) {
16            linha += "*";
17          }
18
19          System.out.println(linha);
20        }
21      }
22    }
23  }
```

Código Java 5.26: Resolução do desafio.

```
1   class ArvoreNatal
2   {
3     static void Main()
4     {
5       int baseMaior = 15;
6
7       for(int m = 7; m <= 15; m += 4)
8       {
9
10        for(int i = m - 6; i <= m; i += 2)
11        {
12          int espacos = (baseMaior - i)/2;
13          string linha = "";
14
15          for(int j = 0; j < espacos; j++)
```

```
16          {
17            linha += " ";
18          }
19
20          for(int k = 0; k < i; k++)
21          {
22            linha += "*";
23          }
24
25          System.Console.WriteLine(linha);
26        }
27      }
28    }
29  }
```

Código C# 5.20: Resolução do desafio.

Resposta do Desafio 6.1

```
1   import java.io.InputStream;
2   import java.util.Scanner;
3   import java.util.Random;
4
5   class BatalhaNaval{
6     public static void main(String[] args){
7       InputStream entrada = System.in;
8       Scanner scanner = new Scanner(entrada);
9       Random geradorDeNumeroAleatorio = new Random();
10      int numeroDeNaviosPosicionados = 0;
11      int ganhador = 0;
12      int x = 0;
13      int y = 0;
14      String strX = "";
15      String strY = "";
16      int pontosJogador = 0;
17      int pontosComputador = 0;
18      char[][] tabuleiroJogador = new char[10][];
19      char[][] tabuleiroComputador = new char[10][];
20
21      for(int i = 0; i < 10; i++){
22        tabuleiroJogador[i] = new char[10];
23        tabuleiroComputador[i] = new char[10];
24      }
25
26      for(int i = 0; i < 10; i++){
27        for(int j = 0; j < 10; j++){
28          tabuleiroJogador[i][j] = ' ';
29          tabuleiroComputador[i][j] = ' ';
30        }
31      }
32
33      while(numeroDeNaviosPosicionados < 10){
34
35
36        System.out.println("Navios posicionados: " + numeroDeNaviosPosicionados);
37        System.out.println("Escolha um número entre 0 e 9 (inclusivo):");
38        strX = scanner.nextLine();
39        System.out.println("Escolha uma letra entre A e J (inclusivo):");
40        strY = scanner.nextLine();
41
42        x = Integer.parseInt(strX);
43
44        if(strY.equalsIgnoreCase("A")){
45          y = 0;
46        }
```

```
47        else if(strY.equalsIgnoreCase("B")){
48          y = 1;
49        }
50        else if(strY.equalsIgnoreCase("C")){
51          y = 2;
52        }
53        else if(strY.equalsIgnoreCase("D")){
54          y = 3;
55        }
56        else if(strY.equalsIgnoreCase("E")){
57          y = 4;
58        }
59        else if(strY.equalsIgnoreCase("F")){
60          y = 5;
61        }
62        else if(strY.equalsIgnoreCase("G")){
63          y = 6;
64        }
65        else if(strY.equalsIgnoreCase("H")){
66          y = 7;
67        }
68        else if(strY.equalsIgnoreCase("I")){
69          y = 8;
70        }
71        else{
72          y = 9;
73        }
74
75        if(tabuleiroJogador[y][x] == ' '){
76          tabuleiroJogador[y][x] = 'N';
77          numeroDeNaviosPosicionados++;
78        }
79        else{
80          System.out.println("Posição já selecionada. Escolha outra.");
81        }
82      }
83
84      numeroDeNaviosPosicionados = 0;
85
86      while(numeroDeNaviosPosicionados < 10){
87        x = geradorDeNumeroAleatorio.nextInt(10);
88        y = geradorDeNumeroAleatorio.nextInt(10);
89
90        if(tabuleiroComputador[y][x] == ' '){
91          tabuleiroComputador[y][x] = 'N';
92          numeroDeNaviosPosicionados++;
93        }
94      }
95
96      System.out.println("");
97      System.out.println("Começando a partida...");
98      System.out.println("");
99
100     while(ganhador == 0){
101       System.out.println("-------------------------------------------------");
102       System.out.println("------------------ JOGADOR ----------------------");
103       System.out.println("-------------------------------------------------");
104       System.out.println("|   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |");
105       System.out.println("-------------------------------------------------");
106
107       for(int i = 0; i < 10; i++){
108         String linha = "| ";
109
110         if(i == 0){
111           linha += "A |";
112         }
113         else if(i == 1){
114           linha += "B |";
115         }
116         else if(i == 2){
```

```
117             linha += "C |";
118           }
119           else if(i == 3){
120             linha += "D |";
121           }
122           else if(i == 4){
123             linha += "E |";
124           }
125           else if(i == 5){
126             linha += "F |";
127           }
128           else if(i == 6){
129             linha += "G |";
130           }
131           else if(i == 7){
132             linha += "H |";
133           }
134           else if(i == 8){
135             linha += "I |";
136           }
137           else{
138             linha += "J |";
139           }
140
141           for(int j = 0; j < 10; j++){
142             linha += " " + tabuleiroJogador[i][j] + " |";
143           }
144
145           System.out.println(linha);
146           System.out.println("-----------------------------------------------");
147         }
148
149         System.out.println("Sua vez de atacar...");
150         System.out.println("Escolha um número entre 0 e 9 (inclusivo):");
151         strX = scanner.nextLine();
152         System.out.println("Escolha uma letra entre A e J (inclusivo):");
153         strY = scanner.nextLine();
154         System.out.println("Atacando na coordenada " + strX + " " + strY);
155
156         x = Integer.parseInt(strX);
157
158         if(strY.equalsIgnoreCase("A")){
159           y = 0;
160         }
161         else if(strY.equalsIgnoreCase("B")){
162           y = 1;
163         }
164         else if(strY.equalsIgnoreCase("C")){
165           y = 2;
166         }
167         else if(strY.equalsIgnoreCase("D")){
168           y = 3;
169         }
170         else if(strY.equalsIgnoreCase("E")){
171           y = 4;
172         }
173         else if(strY.equalsIgnoreCase("F")){
174           y = 5;
175         }
176         else if(strY.equalsIgnoreCase("G")){
177           y = 6;
178         }
179         else if(strY.equalsIgnoreCase("H")){
180           y = 7;
181         }
182         else if(strY.equalsIgnoreCase("I")){
183           y = 8;
184         }
185         else{
186           y = 9;
```

```
187       }
188
189       if(tabuleiroComputador[y][x] == ' '){
190         tabuleiroComputador[y][x] = '-';
191         System.out.println("Não acertou nada.");
192       }
193       else if(tabuleiroComputador[y][x] == 'N'){
194         tabuleiroComputador[y][x] = '*';
195         pontosJogador++;
196         System.out.println("Acertou um navio do oponente!");
197       }
198       else{
199         System.out.println("Você já havia tentado essa posição.");
200       }
201
202       if(pontosJogador == 10){
203         ganhador = 1;
204         break;
205       }
206
207       System.out.println("Vez do computador...");
208
209       x = geradorDeNumeroAleatorio.nextInt(10);
210       y = geradorDeNumeroAleatorio.nextInt(10);
211
212       if(tabuleiroJogador[y][x] == ' '){
213         tabuleiroJogador[y][x] = '-';
214         System.out.println("Não acertou nada.");
215       }
216       else if(tabuleiroJogador[y][x] == 'N'){
217         tabuleiroJogador[y][x] = '*';
218         pontosComputador++;
219         System.out.println("Acertou um navio seu!");
220       }
221
222       if(pontosComputador == 10){
223         ganhador = 2;
224       }
225     }
226
227     if(ganhador == 1){
228       System.out.println("Parabéns! Você ganhou o jogo!");
229     }
230     else{
231       System.out.println("O computador venceu. Tente novamente.");
232     }
233
234     System.out.println("-----------------------------------------------");
235     System.out.println("------------------ JOGADOR ------------------");
236     System.out.println("-----------------------------------------------");
237     System.out.println("|   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |");
238     System.out.println("-----------------------------------------------");
239
240     for(int i = 0; i < 10; i++){
241       String linha = "| ";
242
243       if(i == 0){
244         linha += "A |";
245       }
246       else if(i == 1){
247         linha += "B |";
248       }
249       else if(i == 2){
250         linha += "C |";
251       }
252       else if(i == 3){
253         linha += "D |";
254       }
255       else if(i == 4){
256         linha += "E |";
```

```
257         }
258         else if(i == 5){
259           linha += "F |";
260         }
261         else if(i == 6){
262           linha += "G |";
263         }
264         else if(i == 7){
265           linha += "H |";
266         }
267         else if(i == 8){
268           linha += "I |";
269         }
270         else{
271           linha += "J |";
272         }
273 
274         for(int j = 0; j < 10; j++){
275           linha += " " + tabuleiroJogador[i][j] + " |";
276         }
277 
278         System.out.println(linha);
279         System.out.println("-----------------------------------------------");
280       }
281 
282       System.out.println("-----------------------------------------------");
283       System.out.println("---------------- COMPUTADOR -----------------");
284       System.out.println("-----------------------------------------------");
285       System.out.println("|   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |");
286       System.out.println("-----------------------------------------------");
287 
288       for(int i = 0; i < 10; i++){
289         String linha = "| ";
290 
291         if(i == 0){
292           linha += "A |";
293         }
294         else if(i == 1){
295           linha += "B |";
296         }
297         else if(i == 2){
298           linha += "C |";
299         }
300         else if(i == 3){
301           linha += "D |";
302         }
303         else if(i == 4){
304           linha += "E |";
305         }
306         else if(i == 5){
307           linha += "F |";
308         }
309         else if(i == 6){
310           linha += "G |";
311         }
312         else if(i == 7){
313           linha += "H |";
314         }
315         else if(i == 8){
316           linha += "I |";
317         }
318         else{
319           linha += "J |";
320         }
321 
322         for(int j = 0; j < 10; j++){
323           linha += " " + tabuleiroComputador[i][j] + " |";
324         }
325 
326         System.out.println(linha);
```

```
327       System.out.println("-----------------------------------------");
328     }
329   }
330 }
```

Código Java 6.20: Resolução do desafio.

```
1  using System;
2
3  class BatalhaNaval{
4    static void Main(){
5      Random geradorDeNumeroAleatorio = new Random();
6      int numeroDeNaviosPosicionados = 0;
7      int ganhador = 0;
8      int x = 0;
9      int y = 0;
10     string strX = "";
11     string strY = "";
12     int pontosJogador = 0;
13     int pontosComputador = 0;
14     char[][] tabuleiroJogador = new char[10][];
15     char[][] tabuleiroComputador = new char[10][];
16
17     for(int i = 0; i < 10; i++){
18       tabuleiroJogador[i] = new char[10];
19       tabuleiroComputador[i] = new char[10];
20     }
21
22     for(int i = 0; i < 10; i++){
23       for(int j = 0; j < 10; j++){
24         tabuleiroJogador[i][j] = ' ';
25         tabuleiroComputador[i][j] = ' ';
26       }
27     }
28
29     while(numeroDeNaviosPosicionados < 10){
30
31
32       Console.WriteLine("Navios posicionados: " + numeroDeNaviosPosicionados);
33       Console.WriteLine("Escolha um número entre 0 e 9 (inclusivo):");
34       strX = Console.ReadLine();
35       Console.WriteLine("Escolha uma letra entre A e J (inclusivo):");
36       strY = Console.ReadLine();
37
38       x = Convert.ToInt32(strX);
39
40       if(strY.Equals("A", StringComparison.OrdinalIgnoreCase)){
41         y = 0;
42       }
43       else if(strY.Equals("B", StringComparison.OrdinalIgnoreCase)){
44         y = 1;
45       }
46       else if(strY.Equals("C", StringComparison.OrdinalIgnoreCase)){
47         y = 2;
48       }
49       else if(strY.Equals("D", StringComparison.OrdinalIgnoreCase)){
50         y = 3;
51       }
52       else if(strY.Equals("E", StringComparison.OrdinalIgnoreCase)){
53         y = 4;
54       }
55       else if(strY.Equals("F", StringComparison.OrdinalIgnoreCase)){
56         y = 5;
57       }
58       else if(strY.Equals("G", StringComparison.OrdinalIgnoreCase)){
59         y = 6;
60       }
61       else if(strY.Equals("H", StringComparison.OrdinalIgnoreCase)){
62         y = 7;
```

```
63        }
64        else if(strY.Equals("I", StringComparison.OrdinalIgnoreCase)){
65          y = 8;
66        }
67        else{
68          y = 9;
69        }
70
71        if(tabuleiroJogador[y][x] == ' '){
72          tabuleiroJogador[y][x] = 'N';
73          numeroDeNaviosPosicionados++;
74        }
75        else{
76          Console.WriteLine("Posição já selecionada. Escolha outra.");
77        }
78      }
79
80      numeroDeNaviosPosicionados = 0;
81
82      while(numeroDeNaviosPosicionados < 10){
83        x = geradorDeNumeroAleatorio.Next(0, 10);
84        y = geradorDeNumeroAleatorio.Next(0, 10);
85
86        if(tabuleiroComputador[y][x] == ' '){
87          tabuleiroComputador[y][x] = 'N';
88          numeroDeNaviosPosicionados++;
89        }
90      }
91
92      Console.WriteLine("");
93      Console.WriteLine("Começando a partida...");
94      Console.WriteLine("");
95
96      while(ganhador == 0){
97        Console.WriteLine("-----------------------------------------------");
98        Console.WriteLine("------------------ JOGADOR ------------------");
99        Console.WriteLine("-----------------------------------------------");
100       Console.WriteLine("|   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |");
101       Console.WriteLine("-----------------------------------------------");
102
103       for(int i = 0; i < 10; i++){
104         string linha = "| ";
105
106         if(i == 0){
107           linha += "A |";
108         }
109         else if(i == 1){
110           linha += "B |";
111         }
112         else if(i == 2){
113           linha += "C |";
114         }
115         else if(i == 3){
116           linha += "D |";
117         }
118         else if(i == 4){
119           linha += "E |";
120         }
121         else if(i == 5){
122           linha += "F |";
123         }
124         else if(i == 6){
125           linha += "G |";
126         }
127         else if(i == 7){
128           linha += "H |";
129         }
130         else if(i == 8){
131           linha += "I |";
132         }
```

```
133         else{
134           linha += "J |";
135         }
136
137         for(int j = 0; j < 10; j++){
138           linha += " " + tabuleiroJogador[i][j] + " |";
139         }
140
141         Console.WriteLine(linha);
142         Console.WriteLine("-----------------------------------------");
143       }
144
145       Console.WriteLine("Sua vez de atacar...");
146       Console.WriteLine("Escolha um número entre 0 e 9 (inclusivo):");
147       strX = Console.ReadLine();
148       Console.WriteLine("Escolha uma letra entre A e J (inclusivo):");
149       strY = Console.ReadLine();
150       Console.WriteLine("Atacando na coordenada " + strX + " " + strY);
151
152       x = Convert.ToInt32(strX);
153
154       if(strY.Equals("A", StringComparison.OrdinalIgnoreCase)){
155         y = 0;
156       }
157       else if(strY.Equals("B", StringComparison.OrdinalIgnoreCase)){
158         y = 1;
159       }
160       else if(strY.Equals("C", StringComparison.OrdinalIgnoreCase)){
161         y = 2;
162       }
163       else if(strY.Equals("D", StringComparison.OrdinalIgnoreCase)){
164         y = 3;
165       }
166       else if(strY.Equals("E", StringComparison.OrdinalIgnoreCase)){
167         y = 4;
168       }
169       else if(strY.Equals("F", StringComparison.OrdinalIgnoreCase)){
170         y = 5;
171       }
172       else if(strY.Equals("G", StringComparison.OrdinalIgnoreCase)){
173         y = 6;
174       }
175       else if(strY.Equals("H", StringComparison.OrdinalIgnoreCase)){
176         y = 7;
177       }
178       else if(strY.Equals("I", StringComparison.OrdinalIgnoreCase)){
179         y = 8;
180       }
181       else{
182         y = 9;
183       }
184
185       if(tabuleiroComputador[y][x] == ' '){
186         tabuleiroComputador[y][x] = '-';
187         Console.WriteLine("Não acertou nada.");
188       }
189       else if(tabuleiroComputador[y][x] == 'N'){
190         tabuleiroComputador[y][x] = '*';
191         pontosJogador++;
192         Console.WriteLine("Acertou um navio do oponente!");
193       }
194       else{
195         Console.WriteLine("Você já havia tentado essa posição.");
196       }
197
198       if(pontosJogador == 10){
199         ganhador = 1;
200         break;
201       }
202
```

```
203       Console.WriteLine("Vez do computador...");
204
205       x = geradorDeNumeroAleatorio.Next(0, 10);
206       y = geradorDeNumeroAleatorio.Next(0, 10);
207
208       if(tabuleiroJogador[y][x] == ' '){
209         tabuleiroJogador[y][x] = '-';
210         Console.WriteLine("Não acertou nada.");
211       }
212       else if(tabuleiroJogador[y][x] == 'N'){
213         tabuleiroJogador[y][x] = '*';
214         pontosComputador++;
215         Console.WriteLine("Acertou um navio seu!");
216       }
217
218       if(pontosComputador == 10){
219         ganhador = 2;
220       }
221     }
222
223     if(ganhador == 1){
224       Console.WriteLine("Parabéns! Você ganhou o jogo!");
225     }
226     else{
227       Console.WriteLine("O computador venceu. Tente novamente.");
228     }
229
230     Console.WriteLine("-----------------------------------------------");
231     Console.WriteLine("------------------ JOGADOR ------------------");
232     Console.WriteLine("-----------------------------------------------");
233     Console.WriteLine("|   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |");
234     Console.WriteLine("-----------------------------------------------");
235
236     for(int i = 0; i < 10; i++){
237       string linha = "| ";
238
239       if(i == 0){
240         linha += "A |";
241       }
242       else if(i == 1){
243         linha += "B |";
244       }
245       else if(i == 2){
246         linha += "C |";
247       }
248       else if(i == 3){
249         linha += "D |";
250       }
251       else if(i == 4){
252         linha += "E |";
253       }
254       else if(i == 5){
255         linha += "F |";
256       }
257       else if(i == 6){
258         linha += "G |";
259       }
260       else if(i == 7){
261         linha += "H |";
262       }
263       else if(i == 8){
264         linha += "I |";
265       }
266       else{
267         linha += "J |";
268       }
269
270       for(int j = 0; j < 10; j++){
271         linha += " " + tabuleiroJogador[i][j] + " |";
272       }
```

```
273
274         Console.WriteLine(linha);
275         Console.WriteLine("-----------------------------------------------");
276       }
277
278       Console.WriteLine("-----------------------------------------------");
279       Console.WriteLine("----------------- COMPUTADOR ------------------");
280       Console.WriteLine("-----------------------------------------------");
281       Console.WriteLine("|    | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |");
282       Console.WriteLine("-----------------------------------------------");
283
284       for(int i = 0; i < 10; i++){
285         string linha = "| ";
286
287         if(i == 0){
288           linha += "A |";
289         }
290         else if(i == 1){
291           linha += "B |";
292         }
293         else if(i == 2){
294           linha += "C |";
295         }
296         else if(i == 3){
297           linha += "D |";
298         }
299         else if(i == 4){
300           linha += "E |";
301         }
302         else if(i == 5){
303           linha += "F |";
304         }
305         else if(i == 6){
306           linha += "G |";
307         }
308         else if(i == 7){
309           linha += "H |";
310         }
311         else if(i == 8){
312           linha += "I |";
313         }
314         else{
315           linha += "J |";
316         }
317
318         for(int j = 0; j < 10; j++){
319           linha += " " + tabuleiroComputador[i][j] + " |";
320         }
321
322         Console.WriteLine(linha);
323         Console.WriteLine("-----------------------------------------------");
324       }
325     }
326   }
```

Código C# 6.9: Resolução do desafio.