CAPÍTULO

# 1 Conceitos básicos

Desde o início de sua existência, o homem procurou criar máquinas que o auxiliassem em seu trabalho, diminuindo o esforço e economizando tempo. Dentre essas máquinas, o computador vem se mostrando uma das mais versáteis, rápidas e seguras.

O computador pode auxiliá-lo em qualquer tarefa. É consciente, trabalhador, possui muita energia, mas não tem iniciativa, nenhuma independência, não é criativo nem inteligente, por isso, precisa receber instruções nos mínimos detalhes.

A finalidade de um computador é receber, manipular e armazenar dados. Visto somente como um gabinete composto por circuitos eletrônicos, cabos e fontes de alimentação, certamente ele parece não ter nenhuma utilidade. O computador só consegue armazenar dados em discos, imprimir relatórios, gerar gráficos, realizar cálculos, entre outras funções, por meio de programas, portanto, sua finalidade principal é realizar a tarefa de *processamento de dados*, isto é, receber dados por um dispositivo de entrada (por exemplo, teclado, mouse, scanner etc.), realizar operações com esses dados e gerar uma resposta que será expressa em um dispositivo de saída (por exemplo, impressora, monitor de vídeo, entre outros) (ASCENCIO, 1999).

Logo, um computador possui duas partes diferentes que trabalham juntas: o hardware, composto pelas partes físicas, e o software, composto pelos programas.

Quando queremos criar ou desenvolver um software para realizar determinado tipo de processamento de dados, devemos escrever um programa ou vários programas interligados. No entanto, para que o computador compreenda e execute esse programa, devemos escrevê-lo usando uma linguagem que tanto o computador quanto o criador de software entendam. Essa linguagem é chamada *linguagem de programação*.

As etapas para o desenvolvimento de um programa são:

- **Análise**: estuda-se o enunciado do problema para definir os dados de entrada, o processamento e os dados de saída.
- **Algoritmo**: ferramentas do tipo descrição narrativa, fluxograma ou português estruturado são utilizadas para descrever o problema com suas soluções.
- **Codificação**: o algoritmo é transformado em códigos da linguagem de programação escolhida para se trabalhar.

Portanto, um programa é a codificação de um algoritmo em uma linguagem de programação (ASCENCIO, 1999).

## 1.1 Conceito de algoritmo

A seguir, apresentamos alguns conceitos de algoritmos:

"Algoritmo é uma sequência de passos que visa atingir um objetivo bem definido" (FORBELLONE, 1999).

"Algoritmo é a descrição de uma sequência de passos que deve ser seguida para a realização de uma tarefa" (ASCENCIO, 1999).

"Algoritmo é uma sequência finita de instruções ou operações cuja execução, em tempo finito, resolve um problema computacional, qualquer que seja sua instância" (SALVETTI, 1999).

"Algoritmos são regras formais para a obtenção de um resultado ou da solução de um problema, englobando fórmulas de expressões aritméticas" (MANZANO, 1997).

"Ação é um acontecimento que, a partir de um estado inicial, após um período de tempo finito, produz um estado final previsível e bem definido. Portanto, um algoritmo é a descrição de um conjunto de comandos que, obedecidos, resultam numa sucessão finita de ações" (FARRER, 1999).

Analisando as definições anteriores, podemos perceber que executamos no dia a dia vários algoritmos, como se pode observar nos exemplos a seguir.

**Algoritmo 1 — Somar três números**
- Passo 1 — Receber os três números.
- Passo 2 — Somar os três números.
- Passo 3 — Mostrar o resultado obtido.

**Algoritmo 2 — Fazer um sanduíche**
- Passo 1 — Pegar o pão.
- Passo 2 — Cortar o pão ao meio.
- Passo 3 — Pegar a maionese.
- Passo 4 — Passar a maionese no pão.
- Passo 5 — Pegar e cortar alface e tomate.
- Passo 6 — Colocar alface e tomate no pão.
- Passo 7 — Pegar o hambúrguer.
- Passo 8 — Fritar o hambúrguer.
- Passo 9 — Colocar o hambúrguer no pão.

**Algoritmo 3 — Trocar uma lâmpada**
- Passo 1 — Pegar uma lâmpada nova.
- Passo 2 — Pegar uma escada.
- Passo 3 — Posicionar a escada embaixo da lâmpada queimada.
- Passo 4 — Subir na escada com a lâmpada nova na mão.
- Passo 5 — Retirar a lâmpada queimada.
- Passo 6 — Colocar a lâmpada nova.
- Passo 7 — Descer da escada.
- Passo 8 — Testar o interruptor.
- Passo 9 — Guardar a escada.
- Passo 10 — Jogar a lâmpada velha no lixo.

**Algoritmo 4 — Ir para a escola**
- Passo 1 — Acordar cedo.
- Passo 2 — Ir ao banheiro.
- Passo 3 — Abrir o armário para escolher uma roupa.
- Passo 4 — Se o tempo estiver quente, pegar uma camiseta e uma calça jeans; Caso contrário, pegar um agasalho e uma calça jeans.
- Passo 5 — Vestir a roupa escolhida.
- Passo 6 — Tomar café.
- Passo 7 — Pegar uma condução.
- Passo 8 — Descer próximo à escola.

**Algoritmo 5 — Sacar dinheiro no banco 24 horas**
- Passo 1 — Ir até um banco 24 horas.
- Passo 2 — Colocar o cartão.
- Passo 3 — Digitar a senha.
- Passo 4 — Solicitar a quantia desejada.
- Passo 5 — Se o saldo for maior ou igual à quantia desejada, sacar; caso contrário, mostrar mensagem de impossibilidade de saque.
- Passo 6 — Retirar o cartão.
- Passo 7 — Sair do banco 24 horas.

**Observação**

Você pode estar pensando: "Mas eu realizo essas atividades de maneira diferente!". Esse pensamento está correto, pois, às vezes, um problema pode ser resolvido de diversas maneiras, porém, gerando a mesma resposta. Ou seja, podem existir vários algoritmos para solucionar o mesmo problema.

## 1.2 Método para a construção de algoritmos

Para a construção de qualquer tipo de algoritmo, é necessário seguir estes passos:

- Compreender completamente o problema a ser resolvido, destacando os pontos mais importantes e os objetos que o compõem.
- Definir os dados de entrada, ou seja, quais dados serão fornecidos e quais objetos fazem parte desse cenário problema.
- Definir o processamento, ou seja, quais cálculos serão efetuados e quais as restrições para esses cálculos. O processamento é responsável pela transformação dos dados de entrada em dados de saída. Além disso, deve-se verificar quais objetos são responsáveis pelas atividades.
- Definir os dados de saída, ou seja, quais dados serão gerados depois do processamento.
- Construir o algoritmo utilizando um dos tipos descritos na próxima seção.
- Testar o algoritmo realizando simulações.

## 1.3 Tipos de algoritmos

Os três tipos mais utilizados de algoritmos são: *descrição narrativa*, *fluxograma* e *pseudocódigo* ou *portugol*, que descrevemos a seguir.

### 1.3.1 Descrição narrativa

A descrição narrativa consiste em analisar o enunciado do problema e escrever, utilizando uma linguagem natural (por exemplo, a língua portuguesa), os passos a serem seguidos para sua resolução.

**Vantagem:** não é necessário aprender nenhum conceito novo, pois uma língua natural, neste ponto, já é bem conhecida.

**Desvantagem:** a língua natural abre espaço para várias interpretações, o que posteriormente dificultará a transcrição desse algoritmo para programa.

### 1.3.2 Fluxograma

O fluxograma consiste em analisar o enunciado do problema e escrever, utilizando símbolos gráficos predefinidos (Tabela 1.1), os passos a serem seguidos para sua resolução.

**Vantagem:** o entendimento de elementos gráficos é mais simples que o entendimento de textos.

**Desvantagem:** é necessário aprender a simbologia dos fluxogramas e, além disso, o algoritmo resultante não apresenta muitos detalhes, dificultando sua transcrição para um programa.

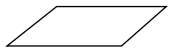
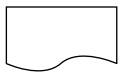
### 1.3.3 Pseudocódigo ou portugol

O pseudocódigo ou portugol consiste em analisar o enunciado do problema e escrever, por meio de regras predefinidas, os passos a serem seguidos para sua resolução.

**Vantagem:** a passagem do algoritmo para qualquer linguagem de programação é quase imediata, bastando conhecer as palavras reservadas da linguagem que será utilizada.

**Desvantagem:** é necessário aprender as regras do pseudocódigo, que serão apresentadas nos próximos capítulos.

**Tabela 1.1** Conjunto de símbolos utilizados no fluxograma.

| | |
|---|---|
|  | Símbolo utilizado para indicar o início e o fim do algoritmo. |
|  | Símbolo que permite indicar o sentido do fluxo de dados. Serve exclusivamente para conectar os símbolos ou blocos existentes. |
|  | Símbolo utilizado para indicar cálculos e atribuições de valores. |
|  | Símbolo utilizado para representar a entrada de dados. |
|  | Símbolo utilizado para representar a saída de dados. |
|  | Símbolo utilizado para indicar que deve ser tomada uma decisão, apontando a possibilidade de desvios. |

## 1.4 Exemplos de algoritmos

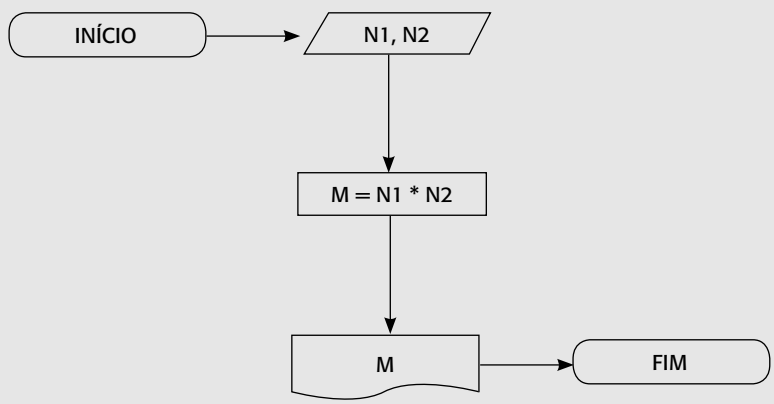
Os exemplos a seguir mostram alguns algoritmos desenvolvidos com os três tipos citados anteriormente.

a) Faça um algoritmo para mostrar o resultado da multiplicação de dois números.

*Algoritmo em descrição narrativa:*

Passo 1 — Receber dois números que serão multiplicados.
Passo 2 — Multiplicar os números.
Passo 3 — Mostrar o resultado obtido na multiplicação.

*Algoritmo em fluxograma:*




*Algoritmo em pseudocódigo:*

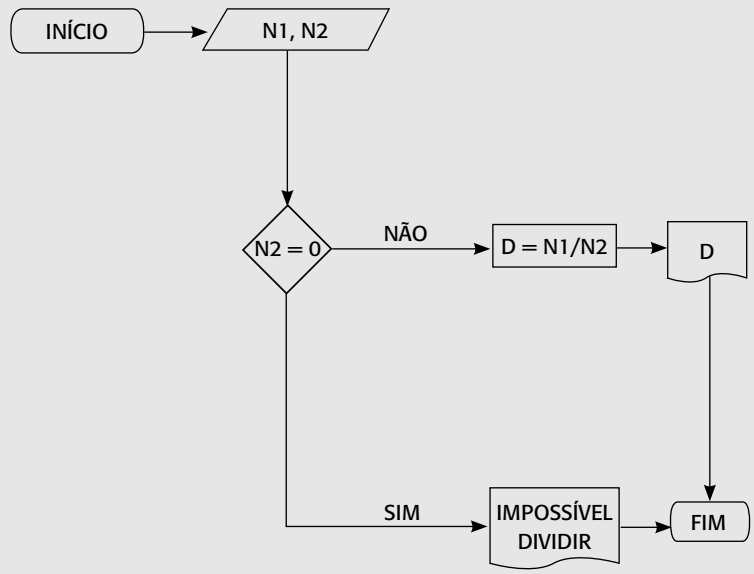
```
ALGORITMO
DECLARE N1, N2, M NUMÉRICO
ESCREVA "Digite dois números"
LEIA N1, N2
M ← N1 * N2
ESCREVA "Multiplicação = ", M
FIM_ALGORITMO.
```

b) Faça um algoritmo para mostrar o resultado da divisão de dois números.

*Algoritmo em descrição narrativa:*

Passo 1 — Receber os dois números que serão divididos.
Passo 2 — Se o segundo número for igual a zero, não poderá ser feita a divisão, pois não existe divisão por zero; caso contrário, dividir os números e mostrar o resultado da divisão.

*Algoritmo em fluxograma:*




*Algoritmo em pseudocódigo:*

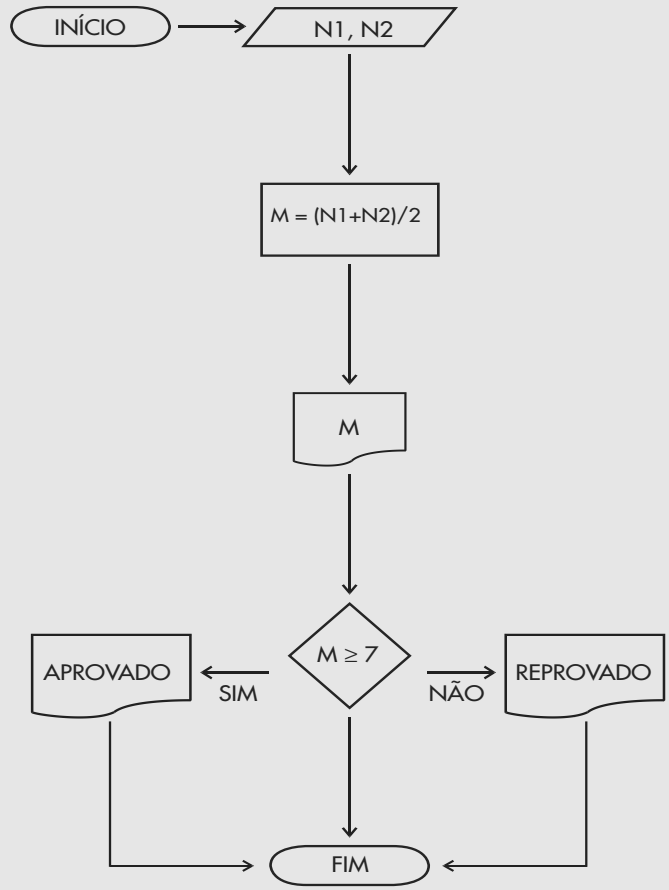
```
ALGORITMO
DECLARE N1, N2, D NUMÉRICO
ESCREVA "Digite dois números"
LEIA N1, N2
SE N2 = 0
ENTÃO ESCREVA "Impossível dividir"
SENÃO INÍCIO
       D ← N1/N2
       ESCREVA "Divisão = ", D
      FIM
FIM_ALGORITMO.
```

c) Faça um algoritmo para calcular a média aritmética entre duas notas de um aluno e mostrar sua situação, que pode ser aprovado ou reprovado.

*Algoritmo em descrição narrativa:*

Passo 1 — Receber as duas notas.
Passo 2 — Calcular a média aritmética.
Passo 3 — Mostrar a média aritmética.
Passo 4 — Se a média aritmética for maior ou igual a 7, então a situação do aluno é *aprovado*; caso contrário, a situação é *reprovado*.

*Algoritmo em fluxograma:*




*Algoritmo em pseudocódigo:*

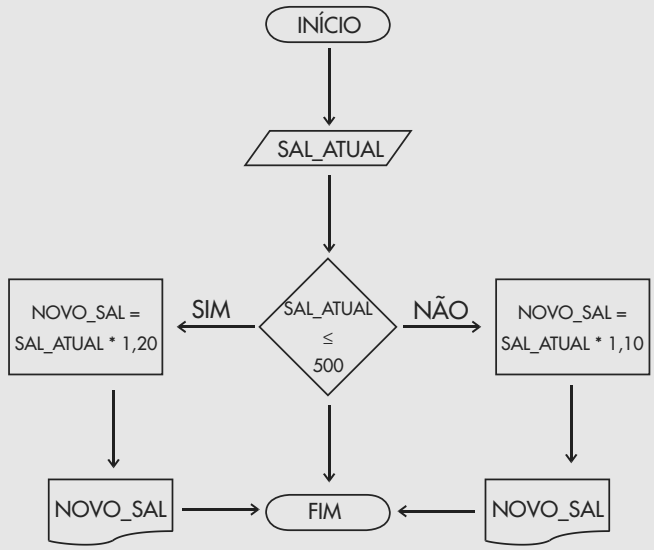
```
ALGORITMO
DECLARE N1, N2, M NUMÉRICO
ESCREVA "Digite as duas notas"
LEIA N1, N2
M ← (N1 + N2)/2
ESCREVA "Média =", M
SE M ≥ 7
ENTÃO ESCREVA "Aprovado"
SENÃO ESCREVA "Reprovado"
FIM_ALGORITMO.
```

d) Faça um algoritmo para calcular o novo salário de um funcionário. Sabe-se que os funcionários que recebem atualmente salário de até R$ 500 terão aumento de 20%; os demais terão aumento de 10%.

*Algoritmo em descrição narrativa:*

Passo 1 — Receber o salário atual do funcionário.
Passo 2 — Se o salário atual do funcionário for de até R$ 500, calcular o novo salário com percentual de aumento de 20%; caso contrário, calcular o novo salário com percentual de aumento de 10%.

*Algoritmo em fluxograma:*




*Algoritmo em pseudocódigo:*

```
ALGORITMO
DECLARE SAL_ATUAL, NOVO_SAL NUMÉRICO
ESCREVA "Digite o salário atual do funcionário"
LEIA SAL_ATUAL
SE SAL_ATUAL ≤ 500
ENTÃO NOVO_SAL ← SAL_ATUAL * 1,20
SENÃO NOVO_SAL ← SAL_ATUAL * 1,10
ESCREVA "Novo salário =", NOVO_SAL
FIM_ALGORITMO.
```

## 1.5 Conceito de variável

Duas pessoas estão conversando e precisam realizar uma conta. A primeira pessoa diz: "Vamos somar dois números". E continua: "O primeiro número é 5". A segunda pessoa guarda o primeiro número na cabeça, ou seja, na memória. A primeira pessoa diz: "O segundo número é 3". A segunda pessoa também guarda o segundo número na cabeça, sem esquecer o primeiro número, ou seja, cada número foi armazenado em posições diferentes da memória humana, sem sobreposição. A primeira pessoa pergunta: "Qual é o resultado da soma?" A segunda pessoa resgata os valores armazenados na memória, realiza a conta e responde dizendo que o resultado é 8.

Um algoritmo e, posteriormente, um programa recebem dados que precisam ser armazenados no computador para serem utilizados no processamento. Esse armazenamento é feito na memória. Todos os computadores trabalham com sistema numérico binário e, nesse sistema, os dados são transformados em 0 e 1 ('zeros' e 'uns') para, então, serem armazenados na memória. Cada dígito binário (0 ou 1) ocupa uma porção de memória chamada bit, e um conjunto de 8 bits é denominado byte. Cada byte é identificado e acessado por meio de um endereço.

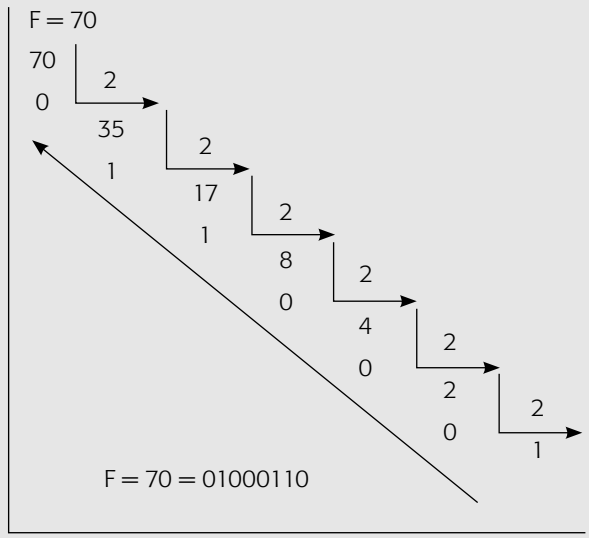
Todos os caracteres existentes possuem um correspondente numérico na tabela ASCII, transformado em caractere binário pelo método da divisão para, então, ser armazenado na memória. Dessa maneira, uma variável representa uma posição de memória, que possui nome e tipo e seu conteúdo pode variar ao longo do tempo, durante a execução de um programa. Embora uma variável possa assumir diferentes valores, ela só pode armazenar um valor a cada instante.

A seguir, um pedaço da tabela ASCII:

**Tabela 1.2** Amostra da tabela ASCII.

| Caractere | Valor decimal na tabela ASCII | Valor binário |
|---|---|---|
| A | 65 | 01000001 |
| B | 66 | 01000010 |
| C | 67 | 01000011 |

Exemplo de transformação em binário:




Todo computador possui uma tabela de alocação que contém o nome da variável, seu tipo (para saber quantos bytes ocupará) e seu endereço inicial de armazenamento. Dessa maneira, quando queremos buscar algum dado na memória, basta sabermos o nome da variável, que o computador, por meio da tabela de alocação, busca automaticamente.

## 1.6 Tipos de dados

Os tipos de dados mais utilizados são: *numéricos*, *lógicos* e *literais* ou *caracteres*, que descreveremos a seguir.

### 1.6.1 Numéricos

Os dados numéricos dividem-se em dois grupos: *inteiros* e *reais*.

Os números inteiros podem ser positivos ou negativos e *não* possuem parte fracionária. Exemplos de dados numéricos inteiros:

```
–23
98
0
–357
237
–2
```

Os números reais podem ser positivos ou negativos e possuem parte fracionária. Exemplos de dados numéricos reais:

```
23.45
346.89
–34.88
0.0
–247.0
```

**Observação**

Os números reais seguem a notação da língua inglesa, ou seja, a parte decimal é separada da parte inteira por um ponto, e não por uma vírgula.

### 1.6.2 Lógicos

São também chamados dados booleanos (oriundos da álgebra de Boole) e podem assumir os valores *verdadeiro* ou *falso*.

### 1.6.3 Literais ou caracteres

São dados formados por um único caractere ou por uma cadeia de caracteres. Esses caracteres podem ser as letras maiúsculas, as letras minúsculas, os números (não podem ser usados para cálculos) e os caracteres especiais (&, #, @, ?, +).

Exemplos de dados literais:

```
"aluno"
"1234"
"@ internet"
"0.34"
"1 + 2"
'A'
'3'
```

**Observação**

Um caractere é representado entre apóstrofos e um conjunto de caracteres é representado entre aspas.

## 1.7 Formação de identificadores

Os identificadores são os nomes das variáveis, dos programas, das constantes, das rotinas, das unidades etc. As regras básicas para a formação dos identificadores são:

- Os caracteres permitidos são: os números, as letras maiúsculas, as letras minúsculas e o caractere sublinhado.
- O primeiro caractere deve ser sempre uma letra ou o caractere sublinhado.
- Não são permitidos espaços em branco e caracteres especiais (@, $, +, –, %, !).
- Não podemos usar as palavras reservadas nos identificadores, ou seja, palavras que pertençam à linguagem de programação.

## 1.8 Exemplos de identificadores

Exemplos de identificadores válidos:

```
A
a
nota
NOTA
X5
A32
NOTA1
MATRICULA
nota_1
dia
IDADE
```

Exemplos de identificadores inválidos:

`5b` — *por começar com número*;
`e 12` — *por conter espaço em branco*;
`x-y` — *por conter o caractere especial –*;
`prova 2n` — *por conter espaço em branco*;
`nota(2)` — *por conter os caracteres especiais ()*;
`case` — *por ser palavra reservada*;
`SET` — *por ser palavra reservada*.

## 1.9 Linguagem PASCAL

A linguagem PASCAL foi desenvolvida em 1968 por Niklaus Wirth, na Suíça, destinada principalmente à programação científica, mas sua grande evolução permitiu que, nos dias de hoje, seja utilizada para qualquer fim. Essa linguagem possui um ambiente integrado de desenvolvimento chamado Turbo Pascal com as seguintes características:

- Apresenta um editor que permite ao desenvolvedor digitar, salvar e modificar o código de seus programas.
- Possui um compilador que converte os códigos dos programas em instruções de máquina e verifica a existência de erros de sintaxe.
- Dispõe de um depurador que permite inspecionar um programa durante sua execução, facilitando a localização de erros.
- Conta com um sistema de ajuda ativo que oferece diferentes níveis de informação.
- Possui ainda o ambiente de execução propriamente dito, que permite executar os programas sem sair do Turbo Pascal (arquivos de extensão PAS) ou, se preferir, permite gerar arquivos a serem executados fora do ambiente do Turbo Pascal (arquivos de extensão EXE).

## 1.10 Linguagem C/C++

Segundo Schildt (1996), Dennis Ritchie inventou a linguagem C e foi o primeiro a implementá-la usando um computador DEC PDP-11, que utilizava o sistema operacional Unix. Essa linguagem é resultante de um processo evolutivo de linguagens, cujo marco inicial foi uma linguagem chamada BCPL, desenvolvida por Martin Richards, que teve forte influência em uma linguagem denominada B, inventada por Ken Thompson. Na década de 1970, B levou ao desenvolvimento de C.

Durante alguns anos, o padrão da linguagem C foi aquele fornecido com a versão 5 do sistema operacional Unix, mas, com a popularização dos microcomputadores, várias implementações de C foram criadas,

gerando, assim, muitas discrepâncias. Para resolver tal situação, o American National Standards Institute (Ansi) estabeleceu, em 1983, um comitê para definir um padrão que guiasse todas as implementações da linguagem C.

A linguagem C++ é uma extensão da linguagem C, e as instruções que fazem parte desta última representam um subconjunto da primeira. Os incrementos encontrados na linguagem C++ foram feitos para dar suporte à programação orientada a objetos, e a sintaxe dessa linguagem é basicamente a mesma da linguagem C.

## 1.11 Linguagem JAVA

A tecnologia JAVA é composta pela linguagem de programação JAVA e pela plataforma de desenvolvimento JAVA. Essa linguagem de programação possui como principais características: simplicidade, orientação a objetos, portabilidade, alta performance e segurança.

Nessa linguagem, os programas são escritos em arquivos texto com a extensão .java e, ao serem compilados com o compilador javac, são gerados os arquivos .class. Um arquivo .class é constituído por bytecodes, código interpretado pela Máquina Virtual Java (*Java Virtual Machine*).

Uma plataforma é um ambiente composto por hardware e software, ou seja, um sistema operacional e o hardware com o qual se comunica. A plataforma JAVA, entretanto, é composta apenas por software, uma vez que é a Máquina Virtual Java que faz a interface entre os programas e o sistema operacional. A plataforma JAVA é composta:

- pela Máquina Virtual Java, responsável por fazer a interface entre seu programa e o sistema operacional, transformando os bytecodes (comuns a qualquer ambiente) em código nativo reconhecido pelo hardware; e
- pela *Application Programming Interface* (API) JAVA, composta por amplo conjunto de classes já implementadas e testadas que fornecem variados recursos aos desenvolvedores.

**Figura 1.1** Processo de execução de um programa em JAVA.