# Fundamentos da Lógica de Programação

## Apostila Completa

---

## Sumário



---

# Capítulo 1: Introdução à Lógica de Programação

## O que é Lógica?

A lógica é a base fundamental para a resolução de problemas de forma organizada e sistemática. Em nosso dia a dia, utilizamos lógica constantemente sem perceber: ao decidir qual rota tomar para chegar ao trabalho, ao organizar tarefas por prioridade ou ao seguir uma receita culinária.

Na programação, a lógica é a ferramenta que nos permite:

- Analisar problemas complexos
- Dividir problemas grandes em partes menores
- Encontrar soluções passo a passo
- Organizar instruções de forma clara e precisa

**Exemplo do cotidiano:** Quando você quer sair de casa pela manhã, mentalmente você segue uma sequência lógica:

1. Acordar
2. Tomar banho
3. Vestir roupa
4. Tomar café
5. Pegar as chaves
6. Sair de casa

Esta sequência é uma aplicação prática da lógica!

## Algoritmos

Um **algoritmo** é uma sequência finita de instruções bem definidas e organizadas para resolver um problema específico. É como uma "receita" que, quando seguida corretamente, sempre produz o resultado esperado.

### Características de um bom algoritmo:

- **Precisão:** Cada passo deve ser claro e sem ambiguidade
- **Finitude:** Deve ter um número limitado de passos
- **Eficiência:** Deve resolver o problema da melhor forma possível
- **Generalidade:** Deve funcionar para diferentes casos do mesmo problema

### Por que algoritmos são importantes?

Os algoritmos são importantes porque:

- Organizam nosso pensamento
- Permitem comunicar soluções de forma clara
- Facilitam a identificação e correção de erros
- Tornam possível a automação de tarefas

## Programas e Linguagens

Um **programa** é a implementação de um algoritmo em uma linguagem que o computador possa entender e executar. O processo de criação segue esta sequência:

**Problema → Algoritmo → Programa → Execução**

### Linguagens de Programação

As linguagens de programação são ferramentas que nos permitem "traduzir" nossos algoritmos para instruções que o computador pode executar. Existem centenas de linguagens, cada uma com suas características específicas:

- **Python:** Simples e versátil
- **Java:** Robusta e multiplataforma
- **C++:** Poderosa e rápida
- **JavaScript:** Essencial para web

**Importante:** A lógica de programação é independente da linguagem escolhida. Uma vez que você domina a lógica, pode aplicá-la em qualquer linguagem!

## Exemplo Prático: Preparando um Café

Vamos transformar o processo de preparar um café em um algoritmo detalhado:

## Algoritmo "Preparar Café"

**Início**

1. Verificar se há água suficiente no reservatório
2. **Se** não há água suficiente:
   - Adicionar água no reservatório
3. Verificar se há café em pó
4. **Se** não há café em pó:
   - Buscar café em pó no armário
   - Adicionar café no filtro
5. **Senão:**
   - Adicionar café no filtro
6. Verificar se há filtro de papel
7. **Se** não há filtro:
   - Buscar filtro de papel
   - Colocar filtro na cafeteira
8. Ligar a cafeteira
9. Aguardar o café ficar pronto
10. Verificar se o café está pronto (não está mais pingando)
11. **Se** o café está pronto:
    - Desligar a cafeteira
    - Servir o café
12. **Fim**

## Análise do Algoritmo

Note que nosso algoritmo:

- Tem instruções claras e precisas
- Considera diferentes situações (há água? há café?)
- Segue uma sequência lógica
- Tem um início e um fim bem definidos
- Sempre produz o resultado esperado (café pronto)

Este é exatamente o tipo de pensamento que precisamos desenvolver para programar!

### Exercício de Reflexão

Como você descreveria o algoritmo para:

- Trocar uma lâmpada queimada?
- Fazer uma ligação telefônica?
- Sacar dinheiro no caixa eletrônico?

Pense nos passos, nas condições que devem ser verificadas e nas diferentes situações que podem ocorrer.

---

# Capítulo 2: Tipos de Dados

## O que são Dados?

Dados são a matéria-prima da programação. Tudo que um programa manipula, processa ou exibe são dados: números, textos, imagens, sons, etc. Para que o computador possa trabalhar eficientemente com essas informações, precisamos classificá-las em tipos específicos.

Imagine uma biblioteca: os livros são organizados por categorias (ficção, história, ciências) para facilitar a localização e o uso. Da mesma forma, os dados em programação são organizados em tipos para que o computador saiba como manipulá-los corretamente.

## Variáveis e Constantes

### Variáveis

Uma **variável** é um espaço na memória do computador que pode armazenar um valor que pode ser modificado durante a execução do programa. É como uma caixa etiquetada onde você pode guardar diferentes objetos ao longo do tempo.

**Características das variáveis:**

- Têm um nome (identificador)
- Armazenam um valor
- Podem ter seu valor alterado
- Têm um tipo específico

**Exemplo:**

```
idade = 25
idade = 26  // O valor foi alterado
```

### Constantes

Uma **constante** é um valor que não pode ser alterado durante a execução do programa. Uma vez definida, permanece inalterada.

**Exemplos de constantes:**

```
PI = 3.14159
VELOCIDADE_DA_LUZ = 299792458
DIAS_DA_SEMANA = 7
```

## Tipos de Dados Fundamentais

### 1. Números Inteiros

Os números inteiros representam valores numéricos sem casas decimais, podendo ser positivos, negativos ou zero.

**Exemplos:**

- `42` (positivo)
- `-15` (negativo)
- `0` (zero)
- `1000` (mil)

**Usos comuns:**

- Contadores (número de cliques, iterações)
- Idades
- Quantidades de produtos
- Anos

**Exemplo prático:**

```
quantidadeAlunos = 30
anoNascimento = 1995
pontuacao = -5
```

### 2. Números Reais (Decimais)

Os números reais representam valores com casas decimais, utilizados quando precisamos de maior precisão.

**Exemplos:**

- `3.14159` (Pi)

- `2.5` (dois e meio)
- `-10.75` (negativo com decimais)
- `0.001` (decimal pequeno)

**Usos comuns:**

- Preços (R$ 15,99)
- Medidas (altura, peso, distância)
- Percentuais (15.5%)
- Temperaturas (36.5°C)

**Exemplo prático:**

```
preco = 29.99
altura = 1.75
temperatura = 23.5
```

## 3. Valores Lógicos (Booleanos)

Os valores booleanos representam apenas dois estados possíveis: **Verdadeiro** (True) ou **Falso** (False). São fundamentais para tomada de decisões.

**Exemplos:**

- `verdadeiro` ou `true`
- `falso` ou `false`

**Usos comuns:**

- Verificar condições (está logado? é maior de idade?)
- Estados de componentes (ligado/desligado)
- Resultados de comparações
- Controle de fluxo do programa

**Exemplo prático:**

```
estaLogado = verdadeiro
maiorIdade = falso
sistemaAtivo = verdadeiro
```

## 4. Caracteres e Textos (Strings)

Os caracteres e strings representam texto: letras, palavras, frases e até símbolos especiais.

**Diferença importante:**

- **Caractere:** Um único símbolo ('A', '5', '@')
- **String/Texto:** Sequência de caracteres ("João", "Rua das Flores, 123")

**Exemplos:**

- `"João Silva"` (nome completo)
- `"Rua das Palmeiras, 123"` (endereço)
- `"joao@email.com"` (email)
- `"123456"` (mesmo sendo números, tratado como texto)

**Usos comuns:**

- Nomes e sobrenomes
- Endereços
- Mensagens
- Códigos de identificação
- Senhas

**Exemplo prático:**

```
nome = "Maria Santos"
email = "maria@empresa.com"
telefone = "(11) 99999-9999"
cep = "01234-567"
```

## Exemplos Práticos de Declaração de Variáveis

### Exemplo 1: Sistema de Cadastro de Pessoa

```
// Dados pessoais
nome = "Ana Carolina Silva"       // String
idade = 28                        // Inteiro
altura = 1.65                     // Real
casada = falso                    // Booleano
email = "ana.silva@email.com"     // String
salario = 3500.50                 // Real
```

### Exemplo 2: Sistema de E-commerce

```
// Dados do produto
nomeProduto = "Smartphone XYZ"     // String
preco = 899.99                     // Real
quantidadeEstoque = 45             // Inteiro
emPromocao = verdadeiro            // Booleano
categoria = "Eletrônicos"          // String
peso = 0.185                       // Real (em kg)
```

### Exemplo 3: Sistema Acadêmico

```
// Dados do aluno
nomeAluno = "Pedro Henrique"       // String
matricula = 202301234              // Inteiro
nota1 = 8.5                        // Real
nota2 = 7.8                        // Real
aprovado = verdadeiro              // Booleano
curso = "Ciência da Computação"    // String
```

## Dicas Importantes

### Nomenclatura de Variáveis

- Use nomes descritivos: `idade` em vez de `i`
- Evite acentos e espaços: `quantidadeItems` em vez de `quantidade itens`
- Use padrões consistentes: `nomeCompleto`, `idadeUsuario`

### Escolha do Tipo Correto

- **Para contagens:** use inteiros
- **Para valores monetários:** use reais
- **Para decisões:** use booleanos
- **Para identificação:** use strings

### Exercício Prático

Identifique o tipo de dado mais adequado para cada informação:

1. Número de páginas de um livro: _____
2. Nome de uma cidade: _____
3. Se um produto está disponível: _____
4. Preço de um produto: _____
5. CEP de um endereço: _____

6. Temperatura ambiente: _____
7. Se um usuário é administrador: _____
8. Quantidade de filhos: _____

**Respostas:** 1-Inteiro, 2-String, 3-Booleano, 4-Real, 5-String, 6-Real, 7-Booleano, 8-Inteiro

---

# Capítulo 3: Operadores Lógicos e Aritméticos

## Introdução aos Operadores

Os operadores são símbolos especiais que executam operações específicas sobre dados. Eles são as ferramentas que usamos para manipular, comparar e combinar informações em nossos programas. Imagine-os como as operações matemáticas que você aprendeu na escola, mas expandidas para trabalhar com diferentes tipos de dados.

## Operadores Aritméticos

Os operadores aritméticos realizam cálculos matemáticos com números. São fundamentais para qualquer programa que precise fazer computações.

### Operadores Básicos

| Operador | Operação | Exemplo | Resultado |
|---|---|---|---|
| `+` | Adição | `5 + 3` | `8` |
| `-` | Subtração | `10 - 4` | `6` |
| `*` | Multiplicação | `7 * 6` | `42` |
| `/` | Divisão | `15 / 3` | `5` |
| `%` | Módulo (resto) | `17 % 5` | `2` |

### Adição (+)

A adição funciona tanto com números quanto pode concatenar textos em algumas situações.

**Exemplos numéricos:**

```
resultado = 10 + 5       // resultado = 15
preco = 25.99 + 3.50     // preco = 29.49
total = -5 + 8           // total = 3
```

**Exemplo com strings (concatenação):**

```
nome = "João" + " Silva"  // nome = "João Silva"
```

## Subtração (-)

Realiza a diferença entre dois números.

**Exemplos:**

```
diferenca = 20 - 8       // diferenca = 12
saldo = 1000.50 - 250.00 // saldo = 750.50
temperatura = 15 - (-5)   // temperatura = 20
```

## Multiplicação (*)

Multiplica dois valores numéricos.

**Exemplos:**

```
area = 5 * 8           // area = 40
preco = 12.50 * 3      // preco = 37.50
negativo = -4 * 7      // negativo = -28
```

## Divisão (/)

Divide um número pelo outro. **Atenção:** divisão por zero causa erro!

**Exemplos:**

```
media = 100 / 4        // media = 25
metade = 15 / 2        // metade = 7.5
resultado = 7 / 3      // resultado = 2.333...
```

## Operador Módulo (%)

O operador módulo retorna o **resto** de uma divisão inteira. É muito útil em programação!

**Como funciona:**

- `17 % 5 = 2` (17 ÷ 5 = 3 com resto 2)
- `20 % 4 = 0` (20 ÷ 4 = 5 com resto 0)
- `13 % 3 = 1` (13 ÷ 3 = 4 com resto 1)

**Usos práticos do módulo:**

1. **Verificar se um número é par ou ímpar:**

```
numero % 2 == 0  // Se verdadeiro, é par
```

2. **Ciclos e repetições:**

```
dia % 7  // Retorna dia da semana (0-6)
```

3. **Validações:**

```
cpf % 11  // Usado em algoritmos de validação
```

## Operadores de Comparação

Os operadores de comparação comparam dois valores e retornam um resultado booleano (verdadeiro ou falso).

| Operador | Significado | Exemplo | Resultado |
|---|---|---|---|
| `==` | Igual a | `5 == 5` | `verdadeiro` |
| `!=` | Diferente de | `3 != 7` | `verdadeiro` |
| `>` | Maior que | `8 > 3` | `verdadeiro` |
| `<` | Menor que | `4 < 9` | `verdadeiro` |
| `>=` | Maior ou igual | `5 >= 5` | `verdadeiro` |
| `<=` | Menor ou igual | `3 <= 2` | `falso` |

**Exemplos práticos:**

```
idade = 18
maiorIdade = idade >= 18     // verdadeiro

nota = 7.5
aprovado = nota >= 7.0       // verdadeiro

senha = "123456"
senhaCorreta = senha == "admin"  // falso
```

## Operadores Lógicos

Os operadores lógicos combinam expressões booleanas e são essenciais para tomada de decisões complexas.

### AND (E) - Operador `and` ou `&&`

Retorna `verdadeiro` apenas quando **todas** as condições são verdadeiras.

**Tabela verdade:**

| A | B | A and B |
|---|---|---|
| V | V | V |
| V | F | F |
| F | V | F |
| F | F | F |

**Exemplos:**

```
idade = 25
temCarteira = verdadeiro
podeConduir = idade >= 18 and temCarteira  // verdadeiro

chuva = verdadeiro
temGuardaChuva = falso
sairSeco = chuva and temGuardaChuva  // falso
```

## OR (OU) - Operador `or` ou `||`

Retorna `verdadeiro` quando **pelo menos uma** condição é verdadeira.

**Tabela verdade:**

| A | B | A or B |
|---|---|---|
| V | V | V |
| V | F | V |
| F | V | V |
| F | F | F |

**Exemplos:**

```
temDinheiro = verdadeiro
temCartao = falso
podeComprar = temDinheiro or temCartao  // verdadeiro

fimSemana = falso
feriado = verdadeiro
naoTrabalha = fimSemana or feriado  // verdadeiro
```

## NOT (NÃO) - Operador `not` ou `!`

Inverte o valor lógico: transforma verdadeiro em falso e vice-versa.

**Tabela verdade:**

| A | not A |
|---|---|
| V | F |
| F | V |

**Exemplos:**

```
estaLogado = verdadeiro
precisaLogar = not estaLogado  // falso

sistemaOff = falso
sistemaOn = not sistemaOff  // verdadeiro
```

## Exemplos Práticos Complexos

### Exemplo 1: Sistema de Acesso

```
// Verificando se usuário pode acessar o sistema
idade = 20
temConta = verdadeiro
contaAtiva = verdadeiro
bloqueado = falso

podeAcessar = idade >= 18 and temConta and contaAtiva and not bloqueado
// Resultado: verdadeiro
```

### Exemplo 2: Cálculo de Desconto

```
// Sistema de e-commerce com descontos
valorCompra = 150.00
ehClienteVIP = verdadeiro
primeiraCompra = falso

// Desconto de 10% para VIP ou primeira compra acima de 100
temDesconto = ehClienteVIP or (primeiraCompra and valorCompra > 100)
// Resultado: verdadeiro (porque é VIP)

desconto = 0
se temDesconto então:
    desconto = valorCompra * 0.10  // desconto = 15.00

valorFinal = valorCompra - desconto  // valorFinal = 135.00
```

### Exemplo 3: Validação de Dados

```
// Validando formulário de cadastro
nome = "João Silva"
idade = 25
email = "joao@email.com"
senha = "minhasenha123"

nomeValido = nome != ""
idadeValida = idade >= 18 and idade <= 120
emailValido = email != "" and (email contém "@")
senhaValida = senha != "" and (tamanho da senha >= 8)

formularioValido = nomeValido and idadeValida and emailValido and senhaValida
// Resultado: verdadeiro (todos os campos são válidos)
```

### Exemplo 4: Situações do Cotidiano

```
// Decidindo se deve levar guarda-chuva
previsaoChuva = 80  // porcentagem
temGuardaChuva = verdadeiro
saiDeCarro = falso

deveLevar = (previsaoChuva > 50 and temGuardaChuva) and not saiDeCarro
// Resultado: verdadeiro
```

## Precedência de Operadores

A ordem de execução dos operadores segue regras específicas:

1. **Parênteses** `()`
2. **Operadores aritméticos:** `*`, `/`, `%` (depois `+`, `-`)
3. **Operadores de comparação:** `>`, `<`, `>=`, `<=`, `==`, `!=`
4. **Operadores lógicos:** `not`, depois `and`, depois `or`

**Exemplo:**

```
resultado = 5 + 3 * 2 > 10 and not falso or 15 % 4 == 3

// Passo a passo:
// 1. 3 * 2 = 6
// 2. 5 + 6 = 11
// 3. 11 > 10 = verdadeiro
// 4. not falso = verdadeiro
// 5. 15 % 4 = 3
// 6. 3 == 3 = verdadeiro
// 7. verdadeiro and verdadeiro = verdadeiro
// 8. verdadeiro or verdadeiro = verdadeiro
// Resultado final: verdadeiro
```

## Exercícios de Fixação

**Calcule os resultados das expressões:**

1. `15 + 3 * 2` = ____
2. `20 / 4 + 10` = ____
3. `17 % 5` = ____
4. `10 > 5 and 3 < 8` = ____
5. `not (5 == 5)` = ____
6. `7 >= 7 or 2 > 3` = ____
7. `(10 + 5) / 3` = ____
8. `4 * 3 + 2 * 5` = ____

**Respostas:** 1-21, 2-15, 3-2, 4-verdadeiro, 5-falso, 6-verdadeiro, 7-5, 8-22

---

# Capítulo 4: Entrada e Saída de Dados (I/O)

## Introdução à Interação com o Usuário

Input/Output (I/O) ou Entrada e Saída de dados é a forma como nossos programas se comunicam com o mundo exterior. Sem I/O, um programa seria como uma calculadora sem tela e sem botões - completamente inútil! A interação com o usuário é o que torna nossos programas vivos e úteis.

**Por que I/O é fundamental?**

- Permite que o programa receba informações do usuário
- Possibilita mostrar resultados e mensagens
- Torna o programa interativo e dinâmico

- Permite validação e feedback em tempo real

## Entrada de Dados - Input

A **entrada de dados** é o processo pelo qual o programa recebe informações do usuário. É como fazer uma pergunta e aguardar a resposta.

### Como funciona o Input

Quando um programa executa uma operação de entrada:

1. O programa para e aguarda
2. O usuário digita a informação
3. O usuário pressiona Enter
4. O programa recebe e armazena a informação
5. A execução continua

### Sintaxe Básica (Pseudocódigo)

```
variavel = input("Mensagem para o usuário")
```

### Exemplos Básicos

#### Exemplo 1: Lendo o nome

```
nome = input("Digite seu nome: ")
// O programa para aqui e aguarda o usuário digitar
// Se o usuário digitar "Maria", nome receberá "Maria"
```

#### Exemplo 2: Lendo idade

```
texto_idade = input("Digite sua idade: ")
idade = converter_para_inteiro(texto_idade)
// Input sempre retorna texto, precisamos converter se necessário
```

#### Exemplo 3: Múltiplas entradas

```
nome = input("Nome: ")
sobrenome = input("Sobrenome: ")
cidade = input("Cidade: ")
```

### Conversão de Tipos

**Importante:** A entrada de dados geralmente retorna texto (string). Se precisarmos de números, devemos converter:

```
// Para números inteiros
texto = input("Digite um número: ")
numero = converter_para_inteiro(texto)

// Para números reais
texto = input("Digite um valor: ")
valor = converter_para_real(texto)

// Para booleanos (mais complexo)
texto = input("Digite sim ou não: ")
se texto == "sim" então:
    resposta = verdadeiro
senão:
    resposta = falso
```

## Saída de Dados - Output

A **saída de dados** é como o programa exibe informações para o usuário. É a forma de mostrar resultados, mensagens e feedback.

### Sintaxe Básica (Pseudocódigo)

```
print("Mensagem para o usuário")
print(variavel)
print("Texto", variavel, "mais texto")
```

### Exemplos de Output

#### Exemplo 1: Mensagem simples

```
print("Bem-vindo ao nosso sistema!")
print("Por favor, faça seu login.")
```

#### Exemplo 2: Exibindo variáveis

```
nome = "João"
idade = 25
print(nome)        // Exibe: João
print(idade)       // Exibe: 25
```

#### Exemplo 3: Combinando texto e variáveis

```
nome = "Maria"
idade = 30
print("Olá,", nome, "! Você tem", idade, "anos.")
// Exibe: Olá, Maria ! Você tem 30 anos.
```

**Exemplo 4: Formatação mais elegante**

```
nome = "Carlos"
salario = 3500.50
print("Funcionário:", nome)
print("Salário: R$", salario)
// Exibe:
// Funcionário: Carlos
// Salário: R$ 3500.5
```

## Exemplo Prático Completo: Sistema de Cadastro

Vamos criar um programa que lê informações do usuário e exibe uma mensagem personalizada:

```
// PROGRAMA: Cadastro Pessoal

// Mensagem de boas-vindas
print("=== SISTEMA DE CADASTRO PESSOAL ===")
print("Por favor, forneça suas informações:")
print()  // Linha em branco

// Coletando dados do usuário
nome = input("Nome completo: ")
texto_idade = input("Idade: ")
cidade = input("Cidade onde mora: ")
texto_salario = input("Salário (R$): ")
estado_civil = input("Estado civil: ")

// Convertendo dados numéricos
idade = converter_para_inteiro(texto_idade)
salario = converter_para_real(texto_salario)

// Processando informações
ano_atual = 2024
ano_nascimento = ano_atual - idade

// Classificando por faixa etária
se idade < 18 então:
    faixa_etaria = "menor de idade"
senão se idade <= 65 então:
    faixa_etaria = "adulto"
senão:
    faixa_etaria = "idoso"

// Exibindo resultado formatado
print()  // Linha em branco
print("=== RESUMO DO CADASTRO ===")
print("Nome:", nome)
print("Idade:", idade, "anos")
print("Ano de nascimento aproximado:", ano_nascimento)
print("Faixa etária:", faixa_etaria)
print("Cidade:", cidade)
print("Estado civil:", estado_civil)
print("Salário: R$", salario)
print()
print("Cadastro realizado com sucesso!")
print("Obrigado por usar nosso sistema,", nome, "!")
```

## Exemplo de Execução:

```
=== SISTEMA DE CADASTRO PESSOAL ===
Por favor, forneça suas informações:

Nome completo: Ana Silva Santos
Idade: 28
Cidade onde mora: São Paulo
Salário (R$): 4500.00
Estado civil: Solteira

=== RESUMO DO CADASTRO ===
Nome: Ana Silva Santos
Idade: 28 anos
Ano de nascimento aproximado: 1996
Faixa etária: adulto
Cidade: São Paulo
Estado civil: Solteira
Salário: R$ 4500.0

Cadastro realizado com sucesso!
Obrigado por usar nosso sistema, Ana Silva Santos !
```

## Exemplo Prático: Calculadora de Média

```
// PROGRAMA: Calculadora de Média de Notas

print("=== CALCULADORA DE MÉDIA ===")
print("Digite suas quatro notas bimestrais:")

// Coletando as notas
nota1_texto = input("1ª nota: ")
nota2_texto = input("2ª nota: ")
nota3_texto = input("3ª nota: ")
nota4_texto = input("4ª nota: ")

// Convertendo para números reais
nota1 = converter_para_real(nota1_texto)
nota2 = converter_para_real(nota2_texto)
nota3 = converter_para_real(nota3_texto)
nota4 = converter_para_real(nota4_texto)

// Calculando a média
soma = nota1 + nota2 + nota3 + nota4
media = soma / 4

// Determinando situação do aluno
se media >= 7.0 então:
    situacao = "APROVADO"
senão se media >= 5.0 então:
    situacao = "RECUPERAÇÃO"
senão:
    situacao = "REPROVADO"

// Exibindo resultados
print()
print("=== RESULTADO ===")
print("Nota 1:", nota1)
print("Nota 2:", nota2)
print("Nota 3:", nota3)
print("Nota 4:", nota4)
print("Soma das notas:", soma)
print("Média final
```