APOSTILA DO ALUNO

LÓGICA DE PROGRAMAÇÃO E ALGORITMOS

Aprenda, escreva, execute e corrija seus próprios algoritmos

Roteiro prático para uso com o Interpretador de Lógica

Professor: Marcos Brandão
Versão: 2026

Apresentação

Esta apostila foi preparada para acompanhar as aulas práticas de Lógica de Programação. O objetivo não é apenas decorar comandos: é aprender a pensar como um programador, transformando problemas em passos claros, testáveis e organizados.

Durante as aulas, você utilizará um Interpretador de Lógica no navegador. Nele, você escreverá algoritmos, executará o programa, fornecerá dados de entrada, observará a saída e corrigirá os erros encontrados.

Regra principal da disciplina
Não copie um algoritmo sem entendê-lo. Antes de executar, tente prever o resultado. Depois execute, compare e explique por que o computador chegou àquele resultado.

Como estudar com esta apostila

1. Leia a explicação curta do conceito.
2. Digite o exemplo no simulador — evite apenas copiar e colar.
3. Antes de executar, diga qual será a saída.
4. Execute e observe o console.
5. Altere uma coisa no programa e execute novamente.
6. Resolva os exercícios sem olhar a solução.
7. Quando errar, leia a mensagem de erro e corrija o programa.

Carga horária sugerida

Etapa	Tema	Horas
1	Pensamento lógico e algoritmos	4h
2	Variáveis, tipos e entrada/saída	4h
3	Operadores e expressões	4h
4	Decisão: se/senão	4h
5	Repetição: enquanto	4h
6	Repetição: para e repita	4h
7	Vetores	4h
8	Funções e modularização	4h
9	Projeto integrador	4h
10	Desafios, testes e revisão	4h

UNIDADE 1 — Pensar antes de programar

Objetivo: compreender problema, algoritmo, sequência lógica e representação de soluções.

1.1 O que é um problema de programação?

Em programação, um problema é uma situação para a qual precisamos definir um procedimento que produza uma resposta. Por exemplo: calcular a média de duas notas. Antes de escolher uma linguagem, precisamos saber quais dados entram, o que será calculado e qual resultado deve sair.

Modelo mental
ENTRADA → PROCESSAMENTO → SAÍDA

Exemplo: para calcular a média de duas notas:

- Entrada: nota1 e nota2.
- Processamento: media = (nota1 + nota2) / 2.
- Saída: valor da média.

1.2 Algoritmo

Algoritmo é uma sequência finita e organizada de instruções para resolver um problema. Uma receita culinária é uma boa analogia: possui ingredientes, passos e um resultado.

1.3 Primeiro algoritmo

```
programa "PrimeiroPrograma"

inicio
   escreval("Olá, mundo!")
fimalgoritmo
```

No simulador, clique em Executar. Depois altere a mensagem e execute novamente.

Pratique

8. Escreva um algoritmo que mostre seu nome.
9. Mostre seu nome, curso e uma profissão que gostaria de exercer.
10. Mostre três mensagens, cada uma em uma linha.
11. Modifique o programa para usar uma única instrução escreval com três valores.

Desafio

Crie uma apresentação de um personagem contendo nome, idade e cidade. Nesta etapa, os valores podem estar escritos diretamente no programa.

UNIDADE 2 — Variáveis, tipos de dados e entrada/saída

Objetivo: armazenar informações e fazer o programa conversar com o usuário.

2.1 Variável

Uma variável é um espaço identificado pelo nome usado para guardar um valor durante a execução. Em vez de escrever sempre o número 20, podemos armazená-lo em uma variável chamada idade.

2.2 Tipos

Tipo	Uso	Exemplo
inteiro	números sem parte decimal	idade = 20
real	números decimais	media = 7.5
cadeia	texto	nome = "Marcos"
caracter	um caractere	letra = "A"
logico	verdadeiro/falso	aprovado = verdadeiro

2.3 Declaração

```
var
   inteiro idade
   real altura
   cadeia nome
   logico ativo
```

2.4 Entrada com leia()

```
programa "Cadastro"

var
   cadeia nome
   inteiro idade

inicio
   escreva("Nome: ")
   leia(nome)
   escreva("Idade: ")
   leia(idade)
   escreval("Olá, ", nome, "! Você tem ", idade, " anos.")
fimalgoritmo
```

Prática guiada

12. Peça o nome do aluno e mostre uma saudação.
13. Leia idade e mostre a idade informada.
14. Leia duas notas e mostre as duas.
15. Leia nome, idade e curso e apresente todos os dados.

Atenção

O comando leia() interrompe a execução até que você forneça um valor no campo de entrada do simulador.

Exercícios

16. Leia dois números inteiros e mostre os valores.
17. Leia o preço de um produto e a quantidade comprada.
18. Leia nome, idade e cidade e crie uma ficha de identificação.

UNIDADE 3 — Operadores e expressões

Objetivo: realizar cálculos e construir expressões que possam ser avaliadas pelo programa.

3.1 Operadores aritméticos

Operador	Operação	Exemplo
+	adição	a + b
-	subtração	a - b
*	multiplicação	a * b
/	divisão	a / b
%	resto da divisão	a % b

3.2 Cálculo da média

```
programa "Media"

var
   real n1
   real n2
   real media

inicio
   escreva("Nota 1: ")
   leia(n1)
   escreva("Nota 2: ")
   leia(n2)

   media = (n1 + n2) / 2
   escreval("Média = ", media)
fimalgoritmo
```

3.3 Relacionais

Os operadores <, <=, >, >=, == e != produzem um resultado lógico: verdadeiro ou falso.

```
idade >= 18
media >= 7
senha == 1234
nota != 0
```

3.4 Lógicos

Use e, ou e nao para combinar condições.

```
idade >= 18 e idade <= 65
nota >= 7 ou recuperacao == verdadeiro
nao bloqueado
```

Prática

19. Calcule a área de um retângulo.
20. Calcule o perímetro de um retângulo.

21. Converta horas em minutos.
22. Calcule o preço total de uma compra.
23. Leia um número e mostre seu dobro, triplo e metade.
24. Leia um número e descubra se ele é par usando o operador %.

UNIDADE 4 — Decisão: SE e SENÃO

Objetivo: fazer o programa tomar decisões de acordo com condições.

4.1 Estrutura básica

```
se condicao entao
   comandos
fimse
```

4.2 Com alternativa

```
se idade >= 18 entao
   escreval("Maior de idade")
senao
   escreval("Menor de idade")
fimse
```

4.3 Exemplo: aprovação

```
programa "Aprovacao"

var
   real media

inicio
   escreva("Média: ")
   leia(media)

   se media >= 7 entao
      escreval("Aprovado")
   senao
      escreval("Não aprovado")
   fimse
fimalgoritmo
```

4.4 Decisões encadeadas

```
se media >= 9 entao
   escreval("Excelente")
senao
   se media >= 7 entao
      escreval("Aprovado")
   senao
      escreval("Precisa estudar mais")
   fimse
fimse
```

Exercícios

25. Leia um número e informe se é positivo, negativo ou zero.

26. Leia idade e informe: criança, adolescente, adulto ou idoso.
27. Leia três notas e informe a situação do aluno.
28. Leia um salário e calcule um desconto diferente conforme a faixa salarial.
29. Crie um sistema simples de login comparando usuário e senha.

Teste de mesa

Antes de executar, faça uma tabela com os valores das variáveis e tente descobrir a saída. Depois compare com o simulador.

UNIDADE 5 — Repetição com ENQUANTO

Objetivo: repetir instruções enquanto uma condição for verdadeira.

```
contador = 1

enquanto contador <= 5 faca
   escreval("Valor = ", contador)
   contador = contador + 1
fimenquanto
```

Observe que existem três elementos importantes: inicialização, condição e atualização. Sem atualização, o programa pode ficar preso em um loop.

Estrutura mental
INICIALIZA → TESTA → EXECUTA → ATUALIZA → TESTA NOVAMENTE

Exercícios

30. Mostre os números de 1 a 10.
31. Mostre os números de 10 até 1.
32. Mostre somente os números pares de 0 a 20.
33. Some os números de 1 a 100.
34. Leia números até que o usuário informe zero e calcule a soma.
35. Conte quantos números positivos foram digitados antes do zero.

Desafio: média de quantidade desconhecida

Leia várias notas. O programa deve parar quando o usuário digitar -1. Ao final, mostre a média das notas válidas.

UNIDADE 6 — Repetição com PARA e REPITA

6.1 PARA

```
para i de 1 ate 10 faca
   escreval(i)
fimpara
```

O PARA é adequado quando sabemos a quantidade ou intervalo de repetições.

6.2 PASSO

```
para i de 0 ate 20 passo 2 faca
   escreval(i)
fimpara
```

6.3 REPITA

```
repita
   escreval("Executando...")
   contador = contador + 1
} ate contador >= 5
```

No interpretador, também é possível escrever blocos com chaves, quando desejado. Para as aulas, prefira inicialmente a forma tradicional com fimpara e fimenquanto.

Exercícios

36. Faça a tabuada de um número.
37. Mostre os múltiplos de 3 entre 1 e 30.
38. Calcule o fatorial de um número.
39. Leia 10 números e mostre o maior.
40. Leia 10 números e conte quantos são pares.
41. Faça um programa que peça uma senha até que ela esteja correta.

UNIDADE 7 — Vetores

Objetivo: armazenar vários valores relacionados em uma única estrutura.

7.1 O que é um vetor?

Um vetor é uma sequência de posições. No simulador, a primeira posição é 0.

```
var
   inteiro notas[5]
   inteiro i

inicio
   para i de 0 ate 4 faca
      leia(notas[i])
   fimpara
fimalgoritmo
```

7.2 Percorrendo o vetor

```
para i de 0 ate 4 faca
   escreval("Nota ", i, " = ", notas[i])
fimpara
```

7.3 Soma e média

```
soma = 0

para i de 0 ate 4 faca
   soma = soma + notas[i]
fimpara
```

```
media = soma / 5
```

Exercícios

42. Leia 5 números e mostre todos eles.
43. Leia 5 números e mostre o maior.
44. Leia 5 notas e calcule a média.
45. Conte quantos valores são maiores que 10.
46. Pesquise se um valor existe no vetor.
47. Inverta a ordem de apresentação dos elementos.

UNIDADE 8 — Funções e modularização

Objetivo: dividir problemas maiores em partes menores, reutilizáveis e fáceis de testar.

8.1 Por que modularizar?

Um programa grande fica difícil de entender quando todas as instruções estão em um único bloco. Funções permitem separar responsabilidades. Uma função recebe dados, realiza uma tarefa e pode devolver um resultado.

```
funcao inteiro soma(inteiro a, inteiro b)
inicio
   retorne a + b
fimfuncao

inicio
   escreval(soma(10, 25))
fimalgoritmo
```

8.2 Função de média

```
funcao real media(real a, real b)
inicio
   retorne (a + b) / 2
fimfuncao
```

Exercícios

48. Crie uma função que retorne o dobro de um número.
49. Crie uma função para calcular o quadrado.
50. Crie uma função que receba três notas e retorne a média.
51. Crie uma função que informe se um número é par.
52. Crie funções para uma calculadora: somar, subtrair, multiplicar e dividir.

UNIDADE 9 — Como depurar um algoritmo

Objetivo: aprender a encontrar e corrigir erros em vez de depender de tentativa e erro.

9.1 Tipos de erro

- Erro de sintaxe: o programa não segue a forma esperada pelo interpretador.
- Erro de execução: o programa começa, mas uma operação inválida ocorre.

- Erro lógico: o programa executa, porém produz resultado incorreto.

9.2 Estratégia de depuração

53. Leia a mensagem de erro.
54. Identifique a linha indicada.
55. Observe o valor das variáveis envolvidas.
56. Faça um teste com valores simples.
57. Verifique a condição dos comandos de decisão e repetição.
58. Execute novamente.

Exemplo de erro lógico

Se você escreve media = n1 + n2 / 2, o programa pode executar, mas a fórmula está errada. Use media = (n1 + n2) / 2.

Atividade

Pegue um algoritmo que você já fez, introduza propositalmente um erro e tente descobrir o problema usando apenas o console e a mensagem do interpretador.

UNIDADE 10 — Projeto integrador

Agora você vai combinar entrada, variáveis, operadores, decisões, repetições, vetores e funções.

Projeto: Sistema de notas

Desenvolva um programa que:

59. Leia o nome do aluno.
60. Leia 4 notas.
61. Armazene as notas em um vetor.
62. Calcule a média usando uma função.
63. Informe a maior nota.
64. Informe a menor nota.
65. Informe a situação: aprovado, recuperação ou reprovado.
66. Mostre um resumo final.

Projeto: Caixa eletrônico simplificado

Desenvolva um programa que receba um valor de saque e informe a quantidade de cédulas necessárias. Comece trabalhando apenas com notas de 100, 50, 20 e 10.

Projeto livre

Escolha um problema real do cotidiano que possa ser automatizado. Exemplos: controle de estoque, cadastro simples, cálculo de salário, controle de frequência, orçamento de compras ou sistema de notas.

Checklist do projeto

- O problema está claramente definido?
- As entradas estão identificadas?
- O processamento está correto?
- As saídas são compreensíveis?

- As variáveis têm nomes claros?
- Há decisões quando necessário?
- Há repetições quando necessário?
- O código foi testado com valores diferentes?
- O programa trata situações inesperadas?

REFERÊNCIA RÁPIDA — Lógica do simulador

Programa

```
programa "Nome"

var
   inteiro x

inicio
   escreval("Olá")
fimalgoritmo
```

Entrada e saída

```
leia(x)
escreva("Valor: ", x)
escreval("Linha nova")
```

Atribuição

```
x = 10
x = x + 1
x++
x--
```

Decisão

```
se x >= 10 entao
   escreval("Maior ou igual")
senao
   escreval("Menor")
fimse
```

Repetição

```
enquanto x <= 10 faca
   escreval(x)
   x = x + 1
fimenquanto

para i de 1 ate 10 faca
   escreval(i)
fimpara
```

Vetor

```
inteiro valores[5]
valores[0] = 10
valores[1] = 20
```

Função

```
funcao inteiro soma(inteiro a, inteiro b)
inicio
   retorne a + b
fimfuncao
```

AVALIAÇÃO FORMATIVA

Ao final de cada unidade, o aluno deve conseguir explicar o conceito com suas próprias palavras e executar pelo menos um programa sem copiar o exemplo.

Competência	Iniciante	Em desenvolvimento	Consolidada
Entender algoritmo	Reconhece comandos	Explica sequência	Transforma problemas em algoritmos
Variáveis	Declara com ajuda	Escolhe tipos adequados	Modela dados corretamente
Decisão	Usa exemplo	Cria condições	Combina condições
Repetição	Executa exemplo	Cria laços	Escolhe estrutura adequada
Vetores	Acessa posições	Percorre vetor	Resolve problemas com coleções
Funções	Reconhece	Cria funções simples	Modulariza soluções
Depuração	Localiza mensagem	Corrige erros	Investiga erros lógicos

DESAFIOS EXTRAS

1. Calculadora: leia dois números e uma operação e mostre o resultado.

2. Conversor de temperatura: Celsius, Fahrenheit e Kelvin.

3. Tabuada completa: mostre as tabuadas de 1 a 10.

4. Jogo de adivinhação: o usuário tenta descobrir um número.

5. Controle de estoque: vetor de produtos/quantidades e cálculo do total.

6. Boletim: várias notas, média, maior, menor e situação.

7. Caixa eletrônico: saque e distribuição de cédulas.

8. Menu: crie um programa com opções e repetição até o usuário escolher sair.

Diário de aprendizagem

Após cada aula, registre: o que aprendi? Onde tive dificuldade? Qual erro consegui corrigir? O que consigo fazer sozinho agora?

Data

O que aprendi?

Erro/dificuldade e como resolvi?

Data

O que aprendi?

Erro/dificuldade e como resolvi?

Data

O que aprendi?

Erro/dificuldade e como resolvi?

Data

O que aprendi?

Erro/dificuldade e como resolvi?

Data

O que aprendi?

Erro/dificuldade e como resolvi?

Mensagem final

Programar é aprender a decompor problemas. No início, os erros parecem obstáculos; com a prática, eles passam a ser pistas. Use o simulador para testar ideias, faça previsões antes de executar e explique seus próprios programas. O objetivo desta disciplina é que você consiga olhar para um problema e pensar: quais são os dados, quais passos preciso executar e qual resultado quero produzir?