capítulo 1

Noções de lógica matemática

De acordo com as Diretrizes Curriculares do MEC para Cursos de Computação e Informática, "[...] a lógica matemática é uma ferramenta fundamental na definição de conceitos computacionais." (BRASIL, [1999], p. 7). De fato, para desenvolver qualquer algoritmo e, consequentemente, qualquer software computacional, são necessários conhecimentos básicos de lógica. Ainda, resolver problemas computacionais requer o conhecimento de operadores e expressões aritméticas, operadores lógicos e relacionais, e sistemas numéricos. Neste capítulo, abordamos esses conteúdos, relacionando sua aplicação na informática em desenvolvimento de programas e algoritmos, em soluções de armazenamento de informações para otimização de memória principal e secundária e em medições e distribuição de processamento e memória em sistemas distribuídos.

Bases Científicas

- Operadores e expressões aritméticas
- Potenciação
- Operadores lógicos e relacionais
- Lógica proposicional
- Sistemas numéricos

Bases Tecnológicas

- Introdução à lógica de programação
- Construção de algoritmos: fluxogramas e pseudocódigos
  - Operadores aritméticos e expressões aritméticas
  - Operadores relacionais
  - Operadores lógicos e expressões lógicas
  - Estruturas de controle
- Conceitos de engenharia de sistemas
- Sistema operacional
- Tipos de memórias
- Armazenamento de dados
- Sistemas numéricos decimais, binário e hexadecimal
- Introdução à programação modo texto ou console
- Introdução à programação visual

Expectativas de Aprendizagem

- Reconhecer e utilizar os operadores aritméticos, lógicos e relacionais.
- Identificar os sistemas de numeração e compreender a conversão entre os sistemas.
- Usar os símbolos formais da lógica proposicional.
- Determinar o valor lógico de uma expressão em lógica proposicional.
- Construir tabelas-verdade.

Introdução

O desenvolvimento da lógica teve seu marco na Grécia Antiga, nos trabalhos desenvolvidos por Aristóteles (384 a.C.-322 a.C.). Aristóteles criou a **lógica analítica**, ou aristotélica, segundo a qual é possível chegar a certas conclusões a partir de noções preliminares sobre um assunto específico. Um exemplo clássico que resume o funcionamento da dedução na lógica aristotélica é: "Todos os homens são mortais. Sócrates é homem. Logo, Sócrates é mortal".

A lógica é utilizada na resolução de muitos problemas computacionais como a criação de algoritmos e de programas de baixa ou alta complexidade. Além disso, também serve para elaborar circuitos lógicos capazes de melhorar o desempenho do hardware dos computadores, como o ganho de velocidade de processamento ou armazenamento de dados e a diminuição dos dispositivos ou melhorias no gerenciamento de energia dos computadores.

Em computação, a lógica matemática está diretamente relacionada à **lógica de Boole** (booleana), que tem como base o 0 (zero) e o 1 (um). Essa teoria teve um papel essencial para o desenvolvimento da computação, pois definia que um sistema matemático poderia ser representado em duas quantidades: o universo (representado pelo número 1) e o nada (representado pelo número 0). Assim, um sistema matemático seria basicamente formado por dois estados para a quantificação lógica. Mais adiante, os inventores do primeiro computador entenderam que um sistema com apenas dois valores poderia compor mecanismos para refazer cálculo. Com o passar dos anos, essas teorias foram aperfeiçoadas e tais referências possibilitaram a simplificação de circuitos eletrônicos e, consequentemente, a melhora no desempenho dos computadores.

No curso técnico em informática, a lógica é parte essencial do aprendizado de computação, pois é necessária desde os primeiros passos, no desenvolvimento de modelos computacionais, diagramas, fluxogramas e algoritmos, até a resolução de problemas mais complexos como o gerenciamento de memória, armazenamento de arquivos, modelos lógicos de distribuição de informação e técnicas de segurança da informação.

» DEFINIÇÃO

Entende-se por lógica booleana ou lógica de Boole o estudo dos princípios e métodos usados para distinguir sentenças verdadeiras de falsas. George Boole (inglês, 1815-1864), matemático e filósofo britânico, foi um dos precursores do estudo da lógica.

Operadores aritméticos e expressões numéricas

Praticamente todo problema computacional é desenvolvido por meio de cálculos aritméticos. Assim, é necessário saber o que são **operadores aritméticos**. Eles são utilizados para desenvolver as operações matemáticas e estão relacionados no Quadro 1.1.

Quadro 1.1 » Operadores aritméticos

Operador	Símbolo	Operação	Exemplo
Sinal mais	+	Valor do operando	A
Sinal menos	–	Negação do operando	– A
Adição	+	Adiciona operandos	A + B
Subtração	–	Subtrai operandos	A – B
Multiplicação	*	Multiplica operandos	A * B
Divisão	/	Divide operandos	A/B
Sinal igual	=	Atribui o valor do operando B ao operando A	A = B

» ATENÇÃO

Um programa, depois de escrito, passa por uma compilação antes de sua execução. Nesse processo de compilação, uma das tarefas é a interpretação das declarações do programa e o estabelecimento da sequência de operações que devem ser executadas.

Na informática, também é comum a utilização de outros dois operadores aritméticos, o DIV e o MOD, utilizados para obter o quociente inteiro da divisão de números inteiros e para obter o resto da divisão de números inteiros, respectivamente.

Quadro 1.2 » Operadores aritméticos

Operador	Símbolo	Exemplo
DIV	DIV	x DIV y
MOD	MOD	x MOD y

Uma **expressão numérica** é um conjunto de operações matemáticas que obedecem a duas propriedades: precedência e associação.

Precedência: estabelece que os operadores de maior precedência tenham seus operandos atribuídos antes daqueles de menor precedência, independentemente do lugar em que aparecem na expressão. Por exemplo, escrever 5 + 6 * 7 é o mesmo que escrever 6 * 7 + 5. Ou seja, a multiplicação tem maior precedência do que a adição. E, em ambas as expressões, o resultado é 47.

Associação: quando os operadores têm a mesma precedência, estabelece a ordem pela qual os operandos serão agrupados – da esquerda para a direita, ou da direita para a esquerda. Por exemplo, para resolver 8 – 4 + 2, agrupamos da esquerda para a direita, resolvendo primeiro a subtração e depois a adição.

» IMPORTANTE

Esta é a ordem de resolução na expressão:

1. Potenciação e radiciação (na ordem que aparecem)
2. Multiplicação ou divisão (na ordem que aparecem)
3. Adição ou subtração (na ordem que aparecem)

Também podemos utilizar parênteses na criação de expressões numéricas simples. Eles impõem uma determinada ordem de agrupamento. Resolver (8 – 4) * 2 é diferente de resolver 8 – (4 * 2). Por exemplo: (8 – 4) * 2 = 4 * 2 = 8, enquanto 8 – (4 * 2) = 8 – 8 = 0. Note que, no último caso, o uso dos parênteses é indiferente, pois a multiplicação tem maior precedência que a subtração.

Podemos, ainda, criar expressões numéricas com parênteses encaixados. Nesse caso, a ordem de agrupamento é do mais interno para o mais externo, aplicando as propriedades de associação e precedência. Veja, por exemplo, a resolução de 5 + ((7 + 3) / (8 – 3) – 8):

5 + ((7 + 3) / (8 – 3) – 8) =
5 + (10 / 5 – 8) =
5 + 2 – 8 =
7 – 8 = –1

Note que, no caso de expressões utilizadas no desenvolvimento de algoritmos ou programas, utilizamos apenas os parênteses. Chaves e colchetes, por exemplo, são símbolos utilizados com outras finalidades, de acordo com a linguagem de programação específica. Assim, para cada parêntese aberto, "(", deve haver um parêntese de fechamento, ")". Também é importante salientar que as expressões sempre devem ser executadas iniciando-se pela expressão mais interna, ou seja, de dentro para fora.

» Agora é a sua vez!

Acesse o ambiente virtual de aprendizagem Tekne para ter acesso às respostas das questões dos quadros "Agora é a sua vez!": ***www.bookman.com.br/tekne****.*

1. Resolva a expressão x/y + a/b, onde x = 1,5; y = 1,0; a = 4 e b = 5.

Se utilizada **potenciação**, sendo dados um número real a e um número natural n, por exemplo, com $n \geq 2$, chama-se de potência de base a e expoente n o número a^n, que é o produto de n fatores iguais a a:

$$a^n = \underbrace{a \times a \times a \times \ldots \times a}_{\text{n fatores}}$$

Veja alguns casos especiais:

- $x^1 = x$
- $1^x = 1$
- $0^x = 0$
- $x^0 = 1, x <> 0$

Propriedades:

1. $a^m \times a^n = a^{m+n}$
2. $a^m / a^n = a^{m-n}$
3. $(a^m)^n = a^{m \times n}$
4. $(a^m \times b^n)^x = a^{m \times x} \times b^{n \times x}$
5. $(a^m / a^n)^x = a^{m \times x} / a^{n \times x}$
6. $a^{-m} = 1/a^m$

Por exemplo: supondo $a \times b <> 0$ para simplificar a expressão $y = (a^3b^4)^6/(a^3)^2b^8$, aplicamos as propriedades $y = a^{18}b^{24}/a^6b^8 = a^{18-6}b^{24-8} = a^{12}b^{16}$.

Agora é a sua vez!

2. Calcule o valor de $(-3)^2$ e -3^2.

Operadores lógicos e relacionais

Os operadores lógicos e relacionais são fundamentais na elaboração de programas, uma vez que as expressões lógicas e relacionais são utilizadas constantemente para solucionar problemas computacionais, desde os mais comuns aos mais complexos, como, por exemplo, a tomada de decisões em algoritmos utilizados na programação de robôs.

» IMPORTANTE

Como podemos ver, a comparação entre valores é utilizada para criar condição verdadeira ou falsa, um recurso em linguagem de programação muito utilizado e que serve para tomar decisões no fluxo do código.

» NO SITE

Acesse o ambiente virtual de aprendizagem Tekne (**www.bookman.com.br/tekne**) para ter acesso a uma apresentação animada em PowerPoint® com exemplos de operadores XOR e XAND.

Os **operadores relacionais**, como o próprio nome sugere, permitem fazer relações ou comparações entre valores e/ou expressões aritméticas. Essas relações podem ser de igualdade (x é igual a y), ou de desigualdade (maior, menor ou diferente). Veja o Quadro 1.3.

Quadro 1.3 » Operadores relacionais

Operador	Símbolo	Exemplo	Resultado
Menor	<	x < y	1 se x menor que y, senão 0
Maior	>	x > y	1 se x maior que y, senão 0
Menor ou igual	<=	x <= y	1 se x menor ou igual a y, senão 0
Maior ou igual	>=	x >= y	1 se x maior ou igual a y, senão 0
Igual	=	x = y	1 se x igual a y, senão 0
Diferente	<>	x <> y	1 se x diferente de y, senão 0

Os **operadores lógicos** são utilizados para elaborar operações relacionais compostas e possibilitam que haja, na comparação de valores ou expressões, uma resposta (retorno), que pode ser ou verdadeira (V) ou falsa (F). Veja o Quadro 1.4.

Quadro 1.4 » Operadores lógicos

Operador	Símbolo	Exemplo	Operação
AND	∧	X ∧ Y	E (conjunção lógica)
OR	∨	X ∨ Y	OU (disjunção lógica)
NOT	~	~ X	negação
Se... então...	→	X → Y	condicional
Se, e somente se,	↔	X ↔ Y	bicondicional

Na computação, é comum a utilização de outros dois operadores lógicos, o XOR e o XAND, normalmente utilizados para operações com portas lógicas. Veja o Quadro 1.5.

Quadro 1.5 » Operadores lógicos

Operador	Símbolo	Exemplo	Operação
XOR	XOR	x XOR y	OU exclusivo
XAND	XAND	x XAND y	E exclusivo

Sistemas de numeração

Nos sistemas digitais, costuma-se recorrer a diferentes sistemas de numeração para representar a informação digital. A base de um sistema de numeração é a quantidade de algarismos ou símbolos disponível na representação. Os sistemas de numeração tem seu nome derivado de sua base – o sistema decimal, por exemplo, tem base 10, enquanto que o hexadecimal tem base 16.

O sistema de numeração decimal (ou na base 10), que usa dez algarismos, é o sistema mais utilizado por seres humanos, e o sistema binário é o mais frequente no mundo da computação, mas existem outros. Veja a seguir.

» DEFINIÇÃO

Um **sistema de numeração** é um conjunto de princípios que constitui um artifício lógico de classificação em grupos e subgrupos das unidades que formam os números.

Sistema decimal

É um sistema de numeração posicional que utiliza base 10. Nesse sistema, os dez algarismos indo-arábicos (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) servem para contar unidades, dezenas, centenas, etc., da direita para a esquerda.

Por exemplo, podemos escrever o número 473 da seguinte forma:

$473 = 4 \times 100 + 7 \times 10 + 3 = 4 \times 10^2 + 7 \times 10^1 + 3 \times 10^0$

Sistema binário

É um sistema de numeração posicional que utiliza base 2 e dispõe de duas cifras: zero e um. O sistema binário é base para a **álgebra booleana**, que permite fazer operações lógicas e aritméticas usando apenas dois dígitos ou dois estados (sim ou não, V ou F, 1 ou 0, ligado ou desligado).

A eletrônica digital e a computação estão baseadas nesse sistema binário e na lógica de Boole, que permite representar por circuitos eletrônicos digitais (portas lógicas) os números e caracteres e realizar operações lógicas e aritméticas. Os programas de computadores são codificados de forma binária e armazenados nas mídias (memórias, discos, etc.) nesse formato.

Adição de binários

Os números binários são base 2, ou seja, há apenas dois algarismos: zero e um. Na soma de 0 com 1, o total é 1. Quando se soma 1 com 1, o resultado é 2, mas, como 2 em binário é 10, o resultado é zero, e passa-se o outro 1 para "frente", ou seja, para ser somado ao próximo elemento.

Por exemplo:

$$\begin{array}{r} 1100 \\ +\ 111 \\ \hline 10011 \end{array}$$

Subtração de binários

Quando temos 0 menos 1, precisamos "emprestar" do elemento vizinho. Esse empréstimo vem valendo 2, pelo fato de ser um número binário. Então, no caso da coluna 0 − 1 = 1, porque, na verdade, a operação feita foi 2 − 1 = 1. Esse processo se repete, e o elemento que cedeu o "empréstimo" e valia 1 passa a valer 0. Note que, logicamente, quando o valor for zero, ele não pode "emprestar" para ninguém, passando-se o "pedido" para o próximo elemento.

Por exemplo:

$$\begin{array}{r} 1101110 \\ -\ 10111 \\ \hline 1010111 \end{array}$$

» IMPORTANTE
São necessários três dígitos para representarmos de 0 (000) a 7 (111) em binário.

» Sistema octal

O sistema octal (base 8) é formado por 8 (oito) símbolos ou dígitos, para representação de qualquer dígito em octal (de 0 a 7). Esse sistema foi criado com o propósito de minimizar a representação de um número binário e facilitar a manipulação humana.

» Sistema hexadecimal

O sistema hexadecimal (base 16) foi criado com o mesmo propósito do sistema octal: minimizar a representação de um número binário. Como não existem símbolos dentro do sistema arábico que possam representar os números decimais entre 10 e 15 sem repetir os símbolos anteriores, foram utilizados símbolos literais: A, B, C, D, E e F.

» IMPORTANTE
Se considerarmos quatro dígitos binários, ou seja, quatro bits, o maior número que se pode expressar com esses quatro dígitos é 1111, que é, em decimal, 15.

» Conversões de uma base numérica para outra

Binário para decimal

Começando sempre da esquerda para a direita, cria-se uma expressão aritmética cuja primeira parcela é 2^m vezes o primeiro algarismo, onde m é o número de dígitos do número a ser convertido menos 1. Na próxima parcela, m vai diminuindo de 1 em 1 até chegar a zero na última parcela. Deverá haver um número de parcelas igual ao número de algarismos do número a ser convertido.

Por exemplo:

1. O número binário 1011 representa o número 11 em decimal:
   $1011 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 8 + 0 + 2 + 1 = 11$
2. O número binário 1100 representa o número 12 em decimal:
   $1100 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 = 8 + 4 + 0 + 0 = 12$

Decimal inteiro para binário

Dado um número decimal inteiro, para convertê-lo em binário, basta dividi-lo sucessivamente por 2, anotando o resto da divisão inteira e, então, ler os números de baixo para cima.

Por exemplo:

1. Veja como converter o número $(24)_{10}$ em binário.

24 : 2	=	12 +	0	
12 : 2	=	6 +	0	
6 : 2	=	3 +	0	
3 : 2	=	1 +	1	
1 : 2	=	0 +	1	↑ *LER*

Assim, $(24)_{10}$ corresponde a $(11000)_2$.

2. Veja como converter $(35)_{10}$ em binário.

35 : 2	=	17 +	1	
17 : 2	=	8 +	1	
8 : 2	=	4 +	0	
4 : 2	=	2 +	0	
2 : 2	=	1 +	0	
1 : 2	=	0 +	1	↑ *LER*

Assim, $(35)_{10}$ corresponde a $(100011)_2$.

Octal para decimal

Dado um número em octal, para convertê-lo em decimal, começamos sempre da esquerda para a direita. Criamos uma expressão aritmética cuja primeira parcela é 8^m vezes o primeiro algarismo, onde m é o número de dígitos do número a ser convertido menos 1. Na próxima parcela, m vai diminuindo de 1 em 1 até chegar a zero na última parcela. Deverá haver um número de parcelas igual ao número de algarismos do número a ser convertido.

Por exemplo:

$(24)_8 = 2 \times 8^1 + 4 \times 8^0 = 16 + 4 = (20)_{10}$

$(16)_8 = 1 \times 8^1 + 6 \times 8^0 = 8 + 6 = (14)_{10}$

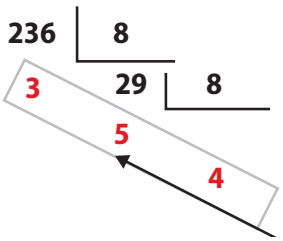
Decimal para octal e para hexadecimal

Para esses tipos de conversão, é necessário dividir sucessivamente pela base o número decimal e os quocientes que vão sendo obtidos, até que o quociente de uma

das divisões seja menor que a base. O resultado é a sequência de baixo para cima do último quociente mais todos os restos obtidos.

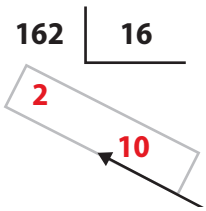
Por exemplo:

1. Conversão de base decimal para base octal:



$(236)_{10} = (354)_8$

2. Conversão de base decimal para base hexadecimal:



$(162)_{10} = (A2)_{16}$

» IMPORTANTE
Em hexadecimal, utilizamos a letra A para "10". Não se esqueça de converter os valores numéricos, de 10 a 15, para letras: A = 10, B = 11, C = 12, D = 13, E = 14 e F = 15.

Hexadecimal para decimal

Para esse tipo de conversão, é necessário primeiramente transformar cada dígito alfabético em número. Assim, utilizando o exemplo da conversão anterior, $(A2)_{16}$, a letra A será convertida para 10 e teremos os números 10 e 2.

Cria-se uma expressão aritmética cuja primeira parcela é 16^m vezes o primeiro coeficiente, onde m é o número de dígitos do número a ser convertido menos 1. Na próxima parcela, m vai diminuindo de 1 até chegar a zero na última parcela. Deverá haver um número de parcelas igual ao número de algarismos do número a ser convertido.

Por exemplo:

$(A2)_{16} = A \times 16^1 + 2 \times 16^0 = 10 \times 16^1 + 2 \times 16^0 = (162)_{10}$

Binário para octal e hexadecimal

Base binária para base octal e vice-versa

É preciso dividir o número binário de 3 em 3 bits, contando sempre da direita para esquerda, e trocar pelos valores da coluna "octal" da Tabela 1.1.

Tabela 1.1 » Tabela de conversão

Decimal	Binário	Octal
0	000	0
1	001	1
2	010	2
3	011	3
4	100	4
5	101	5
6	110	6
7	111	7

Por exemplo:

1. Veja como converter $(1111000111)_2$ na base octal:

   $(1111000111)_2 = (001|111|000|111)_2 = (1707)_8$

2. Veja como fazer o retorno do resultado obtido:

   $(1707)_8 = (001|111|000|111)_2 = (1111000111)_2$

3. Veja como converter $(010100110000)_2$ na base octal:

   $(010100110000)_2 = (010|100|110|000)_2 = (2460)_8$

Base binária para base hexadecimal e vice-versa

Para essa conversão, é necessário dividir o número binário de 4 em 4 bits, contando sempre da direita para esquerda, e trocar pelos valores da coluna "hexadecimal" da Tabela 1.2.

Por exemplo:

1. Veja como converter $(1010111100110111)_2$ na base hexadecimal:

   $\underset{A}{\underline{1010}}\ \underset{F}{\underline{1111}}\ \underset{3}{\underline{0011}}\ \underset{7}{\underline{0111}}$

   É mais fácil trabalhar com um número hexadecimal como o AF37 do que com o binário 1010111100110111.

   $(1010111100110111)_2 = (1010|1111|0011|0111)_2 = (AF37)_{16}$

2. Veja como fazer o retorno do resultado obtido:

   $(AF37)_{16} = (1010|1111|0011|0111)_2 = (1010111100110111)_2$