Ministério da Educação
Escola Técnica Aberta do Brasil
Universidade Tecnológica Federal do Paraná

Lógica de Programação

Everton Coimbra de Araújo

EdUFMT
Cuiabá, 2009

e-Tec Brasil UFMT UTFPR Ministério da Educação BRASIL UM PAÍS DE TODOS GOVERNO FEDERAL

Comissão Editorial Profª Drª Maria Lucia Cavalli Neder - UFMT
Profª Drª Ana Arlinda de Oliveira - UFMT
Profª Drª Lucia Helena Vendrusculo Possari - UFMT
Profª Drª Gleyva Maria Simões de Oliveira - UFMT
Prof. Dr. Henrique Oliveira da Silva - UTFPR
Prof. M. Sc. Oreste Preti - UAB/UFMT

Designer Educacional Oreste Preti e Gleyva Maria S. de Oliveira

Ficha Catalográfica

A663l Araújo, Everton Coimbra de
Lógica de programação / Everton Coimbra de Araújo.-
Cuiabá : EdUFMT, 2009.
132 p. : il. ; color.

Bibliografia: p. 132.
ISBN 978-85-61819-62-0

1.Informática. 2. Lógica de programação. I. Título

CDU - 004.422.612

Revisão Germano Aleixo Filho
Capa (lay out) Marcelo Velasco
Ilustração Marcelo Velasco
Diagramação Terencio Francisco de Oliveira

EdUFMT
Cuiabá, 2009

UNIDADE II

TIPOS DE DADOS, VARIÁVEIS, EXPRESSÕES, SINTAXE E SEMÂNTICA

Na unidade anterior, buscamos introduzir você no campo do raciocínio lógico e de algumas ferramentas para resolução de problemas.

Nesta segunda etapa, você terá contato com alguns conceitos que deve conhecer antes de iniciar a programação com uso de pseudocódigos: **tipos de dados**, **variáveis** e **expressões.** Depois disso, passaremos a estudar conceitos, exemplos e a prática de programar, usando **pseudocódigo** para resolução de algoritmos.

Assim, por meio do estudo desta unidade esperamos que você seja capaz de:

- Compreender o que são e quais são os tipos de dados básicos;
- Definir e compreender o uso de variáveis;
- Conhecer a definição e o correto uso de expressões e operadores;
- Compreender, identificar e aplicar sintaxe e semântica.

2.1. TIPOS DE DADOS

Salvetti & Barbosa (1998) ressaltam que um programa de computador é descrito em uma linguagem de programação. Geralmente, cada linguagem de programação tem seus próprios tipos de dados, isto é, conjunto de valores, operações e relações já implementadas (disponíveis para uso). Na implementação surge o conceito de domínio, isto é, da limitação do conjunto de valores dos elementos representados.

Os tipos de dados que serão tratados nesta sessão são classificados de acordo com o tipo de informação contida neles. Lembramos que a classificação apresentada aqui não se aplica a nenhuma linguagem de programação específica, pois a idéia é mostrar, de forma sintetizada, os padrões utilizados na maioria das linguagens.

2.1.1. DADOS NUMÉRICOS

Os dados numéricos são divididos, basicamente, em dois grandes conjuntos: **inteiros** e **reais**. Veremos nas definições que tudo que foi aprendido nas aulas básicas de matemática, é totalmente utilizado no conceito destas duas classes de tipos de dados numéricos.

Números inteiros são aqueles que não possuem componentes decimais ou fracionários, podendo ser positivos ou negativos.

Esse é fácil! Quer ver alguns exemplos?

Vamos lá!

36 é um número inteiro positivo
0 é um número inteiro
- 8 é um número inteiro negativo

Os dados de tipo **Real** são aqueles que podem possuir componentes decimais ou fracionários, podendo também ser positivos ou negativos.

Bem, agora vejamos...

Podemos citar como exemplos de dados de tipo real:

36.01	é um número real positivo com duas casas decimais
166.	é um número real positivo com zero (nenhuma) casa decimal
- 18.8	é um número real negativo com uma casa decimal
0.0	é um número real com uma casa decimal
0.	é um número real com zero (nenhuma) casa decimal.

É importante observar que há uma diferença entre **0**, que é um dado do tipo inteiro, e **0.** ou **0.0**, que são dados do tipo real. Portanto, a simples existência do ponto decimal serve para diferenciar um dado numérico do tipo **inteiro** de um tipo **real**.

2.1.2. DADOS LITERAIS

O que é um dado literal? Você saberia dizer?

O tipo de dados **literal**, conforme Saliba (1993), pode ser definido como constituído por uma sequência de caracteres com letras, dígitos e/ou símbolos especiais. Este tipo de dado é também muitas vezes chamado de ***alfanumérico, cadeia de caracteres*** ou, ainda, ***String***.

Usualmente, os dados literais são representados nos algoritmos pela coleção de caracteres, delimitada em seu início e término com o caractere aspas ("). É comum, em algumas linguagens, a diferenciação entre a representação de um único dado literal, que é chamado de caractere (por exemplo: **'A'**) e um conjunto de caracteres, chamado de **String** (por exemplo: **"Olá, Mundo"**). Note que, no exemplo de caractere, foram utilizadas *aspas simples (apóstrofo)*. Já no exemplo de string, *aspas duplas*. É interessante esta diferenciação, pois, para algumas linguagens, ela é necessária (por exemplo C e Java).

O dado do tipo literal possui um comprimento dado pelo número de caracteres nele contido. Veja os exemplos:

"QUEM ?"	- Literal de comprimento 5
" "	- Literal de comprimento 1
"cOmO !?#" -	- Literal de comprimento 8
"AbcDEFghi" -	- Literal de comprimento 9
"4+5-1="	- Literal de comprimento 6
"1"	- Literal de comprimento 1

Atenção aos detalhes!

Perceba que **"1.2"** representa um dado do tipo **LITERAL** de comprimento 3, constituído pelos caracteres "1", "." e "2", diferente de **1.2** que é um dado do tipo **REAL**.

2.1.3. DADOS LÓGICOS

São caracterizados, como tipos lógicos, os dados com valor verdadeiro e falso, ressaltando que este tipo de dado poderá representar apenas um dos dois valores. Ele é chamado por alguns de tipo *booleano*, devido à contribuição do filósofo e matemático inglês George Boole na área da lógica matemática.

Para facilitar a citação de um dado do tipo lógico e diferenciação entre nomes de variáveis, alguns autores/professores apresentam estes valores delimitados pelo caractere ponto (.).

Como exemplo deste tipo de dados, temos os valores: **.Falso.** (para o valor lógico: falso) e **.Verdadeiro.** (para o valor lógico: verdadeiro). Observe que isso não é regra, apenas convenção para alguns.

2.2. VARIÁVEIS

Para Lopes & Garcia (2002), uma *variável* é um local na *memória principal*, isto é, um endereço que armazena um conteúdo.

Para facilitar a programação, é permitido que demos nome a esse endereço. O conteúdo de uma variável pode ser de um dos vários tipos apresentados em passo anterior.

Para Forbellone & Eberspacher (2000), um dado é classificado como variável quando tem a possibilidade de ser alterado em algum instante no decorrer do tempo, ou seja, durante a execução do algoritmo, em que é utilizado, o valor do dado sofre alteração ou o dado é dependente da execução em certo momento ou circunstância.

Uma vez definidos o nome e o tipo de uma variável, não podemos alterá-los no decorrer de um algoritmo. Por outro lado, o conteúdo da variável é um objeto de constante modificação no decorrer do programa, de acordo com o fluxo de execução deste.

Quando formos dar nome às variáveis, faz-se necessário seguirmos algumas regras. É bom ressaltar que estas regras irão variar de acordo com a linguagem escolhida como ferramenta, mas a grande maioria adota as seguintes regras genéricas:

- O primeiro caractere é uma **letra**;
- Se houver mais de um caractere, só poderemos usar: **letra** ou **algarismo**;
- Nomes de variáveis escritas com letras maiúsculas serão diferentes de letras minúsculas;
- Nenhuma palavra reservada à ferramenta (linguagem de programação) poderá ser usada como nome de uma variável;
- Procure dar nomes representativos para a variável. Lembre-se de que ao ler seu nome, é importante saber o que ela contém.

As variáveis são definidas no início, pois isso permite a alocação (reserva) de uma área na memória (endereço) para a variável. Outro objetivo da declaração de variáveis é que, após a declaração, o algoritmo sabe os tipos de operação que cada variável pode realizar. Algumas operações só podem ser realizadas com variáveis do tipo inteiro. Outras só podem ser realizadas com variáveis dos tipos inteiro ou real, e outras só com variáveis de caractere, entre outras que serão vistas neste livro.

2.2.1. DECLARAÇÃO DE VARIÁVEIS

Para ilustrar uma declaração de variáveis, é necessário que sejam nominados os tipos de dados que encontramos nos problemas. Vejamos alguns tipos já vistos:

Inteiro – ***int*** ou ***integer***
Real – ***real***, ***float*** ou ***double***
Literal – ***char*** (*um caractere*) ou ***string*** (*cadeia de caracteres*).
Lógico – ***boolean*** ou ***lógico***.

Neste caso, para nomear os tipos de dados de modo válido, precisamos retomar a premissa de que interpretá-los de modo correto é fundamental.

Que tal observarmos alguns exemplos para tornar ainda mais clara essa ideia? Vamos lá...

Sendo dada uma lista de compras com o código, quantidade e preço de oito produtos, faça um algoritmo que escreva o valor total da compra.
string CODIGO ***int*** QUANTIDADE ***float*** PRECO, VALOR TOTAL

Ao serem fornecidos um valor a ser pago e uma taxa para multa, pois o pagamento está sendo feito após o vencimento, calcule o valor da multa e o valor total a ser pago.
float VLRCONTA, TAXAMULTA, VLRMULTA, VLRTOTAL

É solicitada a um motorista, recém-chegado de uma viagem, a quantidade de quilômetros por ele percorrida. O motorista informa o solicitado, e você deverá informar a ele a quantos metros se refere a quantidade de quilômetros.
int QUILOMETROS, METROS

Observe que o trabalho de identificação de variáveis e de seus tipos é, na maioria das vezes, muito fácil, pois basta identificá-los no enunciado. O que ocorre também com grande frequência é a identificação de algumas variáveis

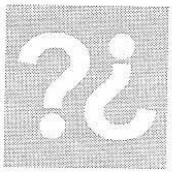
necessárias apenas durante a resolução do algoritmo, mas isso não é problema. Basta relacionarmos estas variáveis à lista já existente.

2.3. EXPRESSÕES E OPERADORES

O conceito de **expressão** em termos computacionais, segundo nos informa Lopes & Garcia (2002), está intimamente ligado ao conceito de expressão (ou fórmula) matemática, no qual um conjunto de variáveis e constantes numéricas se relaciona por meio de operadores, compondo uma fórmula que, uma vez avaliada, resulta em um valor.

Para Saliba (1993), o conceito de **expressão** aplicado à computação assume uma conotação mais ampla: uma expressão é uma combinação de variáveis, constantes e operadores, que, uma vez avaliada, resulta em um valor.

Aqui constam algumas definições. É importante que você leia com atenção esses conceitos e busque outras fontes de informação sobre isso. Também se faz necessário que você procure o significado das palavras que não conhece.



No conceito de Saliba (1993), descrito acima, por exemplo, você saberia conceituar "operadores"?

"Operadores", ainda segundo esse mesmo autor, são elementos funcionais que atuam sobre operandos e produzem determinado resultado.

Ah! Essa esta é bem simples...
Vejamos o exemplo abaixo:

A expressão **5 – 2** relaciona dois operandos (os números 5 e 2) por meio do operador (-) que representa a operação de subtração.

Os operadores podem ser classificados em binários, unários e ternários. Esta classificação é atribuída de acordo com o número de operandos sobre os quais o operador atua. Neste momento, serão abordados somente os operadores binários e unários, mas é importante destacar que os ternários estão presentes em várias linguagens de programação e que sua forma de uso depende exclusivamente da linguagem que está sendo utilizada. Observe, a seguir, exemplos de operadores binários e unários.

- binários: operações matemáticas simples nas quais o operador envolve dois algarismos.
  1+3
  4-1
- unários: o operador somente se encarrega de informar se o número é positivo ou negativo.
  +2

Outra classificação feita aos operadores é em consideração ao tipo de dado de seus operandos e do valor resultante de sua avaliação. Esta classificação é de operadores aritméticos, lógicos, relacionais e literais. Vejamos de que se trata.

2.3.1. EXPRESSÕES ARITMÉTICAS

São aquelas cujo resultado da avaliação é do tipo numérico, seja ele inteiro seja real. Somente o uso de operadores aritméticos e de variáveis numéricas é permitido em expressões deste tipo.

Operador	Tipo	Operação	Prioridade	Exemplo
-	Unário	Inversão de sinal	1	-(-1) = 1
+	Unário	Manutenção de sinal	1	+1 = 1
**	Binário	Exponenciação	2	9**2 = 81
*	Binário	Multiplicação	3	6 * 7 = 42
/	Binário	Divisão	3	8 / 2 = 4
+	Binário	Adição	4	1 + 2 = 3
-	Binário	Subtração	4	4 – 5 = -1

A prioridade entre operadores define a ordem em que eles devem ser avaliados dentro de uma mesma expressão.

Quando há dois ou mais operadores de mesma prioridade em uma expressão, a execução se dá da esquerda para a direita.

É óbvio que, se utilizarmos o conceito de agrupamento de operações através de parênteses, isso poderia ser facilmente visto.

O caractere * é adotado na maioria das linguagens de programação para representar a operação de multiplicação, ao invés do caractere **x**, por força da possibilidade de ocorrência do mesmo nome de variável. Pela mesma razão, o símbolo ** é adotado para representar a operação de exponenciação. Algumas linguagens de programação adotam o símbolo ^, e outras adotam **função** para resolução deste problema.

Por exemplo: em **Java**, utiliza-se **Math.pow(3, 2)** para elevar o número 3 ao quadrado.

2.3.2. EXPRESSÕES LÓGICAS

São aquelas cujo resultado da avaliação é um valor lógico (**.Verdadeiro.** ou **.Falso.**). Veja, a seguir, operadores lógicos para expressões lógicas.

Para tratar expressões lógicas, vamos recorrer à lógica matemática, na qual Filho (2000) define **proposição** como todo o conjunto de palavras ou símbolos que exprimem um pensamento de sentido completo.

As proposições transmitem pensamentos, isto é, **afirmam fatos** ou **exprimem juízos** que formamos a respeito de determinados entes.

Podemos citar como exemplo de proposição:

a) A Lua é um satélite da terra.
b) Recife é a capital de Pernambuco.
c) 2^3 é 8.

A lógica matemática (assim como a que veremos para algoritmos) adota como regras fundamentais do pensamento os dois seguintes princípios:

a) **Princípio da não contradição**
   Uma proposição não pode ser verdadeira e falsa ao mesmo tempo.
b) **Princípio do terceiro excluído**
   Toda a proposição ou é verdadeira ou é falsa, isto é, verifica-se sempre um destes casos, e nunca um terceiro.

As proposições vistas no exemplo anterior são ditas **simples** ou ***atômicas***, pois não contêm nenhuma outra proposição como parte integrante de si mesma. Há somente a ocorrência de uma informação sendo repassada.

Chama-se **proposição composta** ou ***proposição molecular*** aquela formada pela combinação de duas ou mais proposições, como nos exemplos:

a) Carlos é careca **e** Pedro é estudante.
b) Carlos é careca **ou** Pedro é estudante.
c) **Se** Carlos é careca, **então** é infeliz.

Quando pensamos, efetuamos, muitas vezes, certas operações sobre proposições, chamadas **operações lógicas**.

a) **Negação**
   Chamamos **negação de uma proposição p** a proposição representada por ***não p***, cujo valor lógico é a **verdade** quando **p** é falsa, e a **falsidade** quando **p** é **verdadeira**. Assim ***não p*** tem o valor lógico oposto daquele de ***p***.
b) **Conjunção**
   Chamamos **conjunção de duas proposições p e q** a proposição representada por ***p E q***, cujo valor lógico é **verdade** quando as proposições **p** e **q** são ambas verdadeiras, e **falsa** nos demais casos.
c) **Disjunção**
   Chama-se **disjunção de duas proposições p e q** a proposição representada por **p OU q** cujo valor lógico é a **verdade** quando ao menos uma das proposições **p** e **q** é verdadeira, e a **falsidade** quando as proposições **p** e **q** são ambas falsas.

Os operadores lógicos também são chamados de operadores booleanos.

Operador	Tipo	Operação	Prioridade
.OU.	Binário	Disjunção	3
.E.	Binário	Conjunção	2
.NÃO.	Unário	Negação	1

Suponha duas perguntas feitas a quatro pessoas que se candidataram a uma entrevista de emprego de programador. As respostas às perguntas serão **Sim** ou **Não**. Suponha também que só será chamado para a entrevista o candidato que responder **Sim** às duas perguntas.

Tabela verdade do operador .E.			
Candidato	Você conhece a linguagem C ?	Você conhece a Linguagem Pascal ?	Candidato aprovado para a entrevista ?
César	**Não**	**Não**	**Não**
Itamar	**Não**	**Sim**	**Não**
Dudu	**Sim**	**Não**	**Não**
Neusa	**Sim**	**Sim**	**Sim**

Neste caso, apenas a candidata **Neusa** seria chamado para a entrevista, pois o operador **.E.** só considera a expressão como verdadeira se todas as expressões testadas forem verdadeiras.

Vejamos, a seguir, **operadores relacionais** para expressões lógicas.

Operador	Comparação
=	Igual
<>	Diferente
<	Menor
=<	Menor ou igual
>	Maior
=>	Maior ou igual

Estes operadores são somente usados quando se deseja efetuar comparações.

Vale destacar que as comparações só podem ser feitas entre objetos de mesma natureza, isto é, variáveis do mesmo tipo de dado. O resultado de uma comparação é sempre um valor lógico.

Toda expressão respeita uma ordem de execução de seus operadores, sempre da esquerda para a direita, assim como na matemática. Para isso, segue uma tabela com as prioridades.

TABELA COM PRIORIDADES DOS OPERADORES	
Prioridade	Pseudocódigo
Parênteses e funções	Parênteses e funções
Potência e resto	Exp( ) e resto( )
Multiplicação e divisão	* e /
Adição e subtração	+ e -
Operadores relacionais	E, OU, NÃO
Operadores lógicos	= ,>< ,=> ,=< ,< ,>

Exemplos do uso de operadores relacionais			
Exemplo	**Valores**	**Questionamento ?**	**Resultado**
A <> B	A = 5 e B = 6	A é diferente de B ?	*Verdadeiro*
A <> B	A = 6 e B = 6	A é diferente de B ?	*Falso*
X == 1	X = 2	X é igual a 1 ?	*Falso*
X == 1	X = 1	X é igual a 1 ?	*Verdadeiro*
7 > 6		7 é maior que 6 ?	*Verdadeiro*
8 < 9		8 é menor que 9 ?	*Verdadeiro*
1 <= Y	Y = 1	1 é menor ou igual a Y ?	*Verdadeiro*
1 <= Y	Y = 2	1 é menor ou igual a Y ?	*Verdadeiro*
1 <= Y	Y = 0	1 é menor ou igual a Y ?	*Falso*
4 >= W	W = 4	4 é maior ou igual a W ?	*Verdadeiro*
4 >= W	W = 3	4 é maior ou igual a W ?	*Verdadeiro*
4 >= W	W = 5	4 é maior ou igual a W ?	*Falso*

2.4. SINTAXE E SEMÂNTICA

O conceito de linguagem está associado a um *objeto de comunicação*: indivíduos que partilham uma mesma linguagem são capazes de se comunicar. As línguas naturais são utilizadas como meio *formal* de se estabelecer uma linguagem de comunicação. Para tanto, são necessários um vocabulário, ou *léxico (dicionário/glossário)*, e um conjunto de regras gramaticais ou *sintaxe*: para construir um objeto de comunicação nessa língua, a sintaxe permite associar e manipular os itens do léxico. A esse objeto, construído segundo as *normas* da língua, dá-se o nome de *asserção válida*, ou *gramatical*. Asserções válidas com diversos graus de complexidade podem ser construídas: em grau crescente de complexidade, podemos ter orações, sentenças, conjunto de sentenças (ou parágrafos) e conjuntos de parágrafos (ou textos). A essas asserções, é possível também associar um *significado* e, assim, estaremos trabalhando no campo da *semântica*.

Bem, de acordo com o que foi exposto acima, podemos, de modo conciso, dizer:

SINTAXE : São regras gramaticais de formação de sentenças/asserções válidas ou gramaticalmente corretas;

SEMÂNTICA : É a associação das asserções ao significado, permitindo sua *interpretação*.

Tendo estas definições sido trazidas do idioma humano, pode-se afirmar que não há diferença ao encontrado na área tecnológica, pois podemos dizer que a **sintaxe** de uma linguagem expressa as regras que devem ser obedecidas para atingir determinado resultado, fazendo uso dela. E a **semântica** representa o conteúdo das palavras da linguagem, permitindo assim uma interpretação correta do escrito com determinada linguagem.

RELEMBRANDO

Esta unidade tem grande importância no aprendizado de programação. Foram aqui apresentados conceitos e exemplos de tipos de dados possíveis de uso em nossos pseudocódigos, conceitos e regras para uso de variáveis, além de uma introdução sobre expressões e operações. Na parte de expressões lógicas, trouxemos uma pequena demonstração sobre suas operações, as quais serão fortemente tratadas mais adiante. Já nos conceitos de sintaxe e semântica, que foram tratados também de forma breve, foi comentada a grande importância delas, principalmente da sintaxe. Todos estes conceitos serão utilizados nas próximas unidades. Daí a relevância de uma releitura, caso ainda restem dúvidas.

Encerramos esta unidade por aqui, esperando, é claro, que esses conceitos e exemplos tenham ajudado você a compreender ainda mais as etapas da programação.

Mas ainda temos um longo caminho. Vamos em frente!