algoritmos e programação

com exemplos em Pascal e C

nina edelweiss
maria aparecida castro livi

bookman

as autoras

Nina Edelweiss é engenheira eletricista e doutora em Ciência da Computação pela Universidade Federal do Rio Grande do Sul. Durante muitos anos, lecionou em cursos de Engenharia e de Ciência da Computação na UFRGS, na UFSC e na PUCRS. Foi, ainda, orientadora do Programa de Pós-Graduação em Ciência da Computação da UFRGS. É coautora de três livros, tendo publicado diversos artigos em periódicos e em anais de congressos nacionais e internacionais. Participou de diversos projetos de pesquisa financiados por agências de fomento como CNPq e FAPERGS, desenvolvendo pesquisas nas áreas de bancos de dados e desenvolvimento de software.

Maria Aparecida Castro Livi é licenciada e bacharel em Letras, e mestre em Ciência da Computação pela Universidade Federal do Rio Grande do Sul. Desenvolveu sua carreira profissional na UFRGS, onde foi programadora e analista de sistema, antes de ingressar na carreira docente. Ministrou por vários anos a disciplina de Algoritmos e Programação para alunos dos cursos de Engenharia da Computação e Ciência da Computação. Sua área de interesse prioritário é o ensino de Linguagens de Programação, tanto de forma presencial quanto a distância.

E22a Edelweiss, Nina.
Algoritmos e programação com exemplos em Pascal e C [recurso eletrônico] / Nina Edelweiss, Maria Aparecida Castro Livi. – Dados eletrônicos. – Porto Alegre : Bookman, 2014.

Editado também como livro impresso em 2014.
ISBN 978-85-8260-190-7

1. Informática. 2. Algoritmos – Programação. I. Livi, Maria Aparecida Castro. II. Título.

CDU 004.421

Catalogação na publicação: Ana Paula M. Magnus – CRB 10/2052

2.3 → expressões

No exemplo simulado da Seção 1.1.1 foram realizadas algumas operações que envolvem cálculos de expressões aritméticas. Ao escrever o programa correspondente àquele exemplo, essas expressões deverão ser escritas de forma que sejam entendidas corretamente pelo compilador. Cada linguagem de programação define regras bem claras para escrever expressões aritméticas, lógicas e de *strings*.

2.3.1 expressões aritméticas

Expressões aritméticas são expressões cujos resultados são valores numéricos, inteiros ou fracionários. A sintaxe de uma expressão aritmética é a seguinte:

```
<operando>  <operador aritmético>  <operando>
```

Na pseudolinguagem utilizada neste livro, os operadores que podem ser usados em expressões aritméticas são os mesmos utilizados nas expressões aritméticas comuns. Mas, da mesma forma que nas linguagens de programação, o símbolo utilizado para a multiplicação é o asterisco, e o símbolo de divisão é a barra inclinada. A Tabela 2.1 mostra os operadores que podem ser utilizados em expressões aritméticas, na forma adotada pela pseudolinguagem.

tabela 2.1 Operadores aritméticos na pseudolinguagem

Operador	Significado	Observação
+	Soma	-
-	Subtração	-
*	Multiplicação	-
/	Divisão	-
**	Potência	-
Div	Divisão inteira	Operandos inteiros
Mod	Resto da divisão inteira	Operandos inteiros

Os operadores aritméticos têm diferentes precedências na execução das operações: primeiro são calculadas as potências, depois as multiplicações e as divisões e, no final, as somas e as subtrações. Expressões com operadores de mesma precedência justapostos são avaliadas da esquerda para a direita. Essa ordem de precedência pode ser alterada através do uso de parênteses.

Os seguintes tipos de operandos podem ser utilizados:

1. valores numéricos literais;
2. variáveis numéricas;
3. chamadas a funções[1] que devolvem um valor numérico;
4. expressões aritméticas, as quais podem incluir partes entre parênteses.

Se uma expressão aritmética incluir funções, essas terão precedência maior na execução.

Exemplos de expressões aritméticas:

```
a + 1
a * 2 + 7,32
( x / 2 )  / C - ( valor + 1 / 2 )
2 + cos(x)                                  onde cos(x) é uma função
```

As expressões aritméticas devem ser escritas horizontalmente, em uma mesma linha, com eventuais valores fracionários expressos linearmente. Muitas vezes é necessário o emprego de parênteses para garantir a execução na ordem correta. A necessidade de linearização possibilita a uma expressão aritmética ter sua aparência inicial bastante modificada, como no caso da expressão a seguir:

$$a + \frac{(b-4)(\frac{a}{2}+z42)}{c+d}$$

A representação dessa expressão em pseudolinguagem fica:

```
a + ( ( b - 4 ) * ( a / 2  + 4 * z42) / ( c + d ) )
```

[1] Uma FUNÇÃO é um subprograma. Pode receber parâmetros (valores) para realizar sua tarefa e normalmente devolve um valor em seu nome, sendo o tipo do valor devolvido o próprio tipo da função. Mais detalhes sobre definição de funções são vistos no Capítulo 9.

Algumas funções básicas predefinidas já vêm embutidas nas linguagens de programação. Entre elas, funções matemáticas, como o cálculo do cosseno de um ângulo utilizado no exemplo anterior. Algumas dessas funções necessitam de alguma informação para calcular o que é pedido como, por exemplo, o valor do ângulo do qual se quer o cosseno. As informações requeridas são chamadas de parâmetros da função e são listadas logo após o nome da função, entre parênteses. Um parâmetro pode ser fornecido através de uma expressão cujo valor, depois de avaliado, será utilizado pela função.

Na Tabela 2.2 são listadas algumas funções que podem ser utilizadas na pseudolinguagem, definidas de forma idêntica ou similar àquela em que ocorrem na maioria das linguagens de programação.

tabela 2.2 Funções predefinidas na pseudolinguagem

Nome da função	Parâmetro	Significado
abs	valor	Valor absoluto do valor
sen	ângulo	Seno do ângulo
cos	ângulo	Cosseno do ângulo
tan	ângulo	Tangente do ângulo
arctan	valor	Arco cuja tangente tem o valor
sqrt	valor	Raiz quadrada do valor
sqr	valor	Quadrado do valor
pot	base, expoente	Base elevada ao expoente
ln	valor	Logaritmo neperiano
log	valor	Logaritmo na base 10

2.3.2 expressões lógicas

Expressões lógicas são aquelas que têm como resultado valores lógicos, ou seja, um dos dois valores `verdadeiro` ou `falso`.

Uma expressão lógica pode ter uma das seguintes formas:

```
<relação lógica>
<operando>  <operador lógico binário>  <operando>
<operador lógico unário>  <expressão lógica>
```

Uma relação lógica compara dois valores, numéricos ou alfanuméricos, resultando em um valor lógico `verdadeiro` ou `falso`. A sintaxe de uma relação lógica é a seguinte:

```
<expressão>  <operador relacional>  <expressão>
```

Os operadores relacionais utilizados na pseudolinguagem são listados na Tabela 2.3. Outros operadores serão vistos quando se analisar operações sobre *strings*. É importante lembrar que os dois operandos de uma relação devem ser do mesmo tipo para que possam ser comparados.

tabela 2.3 Operadores relacionais na pseudolinguagem

Operador relacional	Significado
=	Igual
≠	Diferente
>	Maior
<	Menor
≥	Maior ou igual
≤	Menor ou igual

Exemplos de relações:

`idade > 21`	onde `idade` é uma variável numérica
`nome = 'Ana Terra'`	onde `nome` é uma variável *string*
`a < (b + 2 * x)`	onde `a, b e x` são variáveis numéricas

Os operandos de expressões lógicas devem resultar em valores lógicos, que são então comparados através de um operador lógico. Podem ser:

- os valores lógicos literais `verdadeiro` e `falso`;
- variáveis declaradas como lógicas;
- relações lógicas;
- chamadas a funções que tenham resultado lógico;
- outras expressões lógicas.

O uso de parênteses é permitido, tanto para dar prioridade a algumas comparações, como simplesmente para tornar o entendimento das expressões mais claro.

Os operadores lógicos comparam valores lógicos, resultando em `verdadeiro` ou `falso`. Na Tabela 2.4 estão os operadores lógicos usualmente empregados – `e`, `ou`, `oux` (ou exclusivo) e `não` (negação) – identificando como é obtido o resultado da comparação. A Tabela 2.5 apresenta os resultados produzidos por cada operador lógico de acordo com os resultados das expressões lógicas A e B, representando "V" o valor lógico verdadeiro e "F" o falso.

tabela 2.4 Operadores lógicos na pseudolinguagem

Operador lógico	Tipo	Resultado
e	Binário	Verdadeiro somente se ambos os operandos são verdadeiros
ou	Binário	Verdadeiro se um dos operandos for verdadeiro
oux	Binário	Verdadeiro se somente um dos operandos for verdadeiro
não	Unário	Verdadeiro se o operando for falso, falso se o operando for verdadeiro

tabela 2.5 Tabela-verdade dos operadores lógicos

A	B	A e B	A ou B	A oux B	não A
V	V	V	V	F	F
V	F	F	V	V	F
F	V	F	V	V	V
F	F	F	F	F	V

A ordem de precedência na avaliação das operações incluídas em uma expressão lógica pode variar conforme a linguagem de programação utilizada. Na pseudolinguagem é adotada a seguinte ordem de precedência na avaliação das operações: primeiro são avaliadas as expressões aritméticas, depois as relações e, por último, as expressões lógicas. Nas expressões lógicas, primeiro são realizadas as negações e, depois, são aplicados os operadores lógicos, entre os quais o "e" tem maior prioridade, seguido pelos operadores "ou" e "oux". As ordens de precedência utilizadas nas linguagens Pascal e C serão mostradas nas seções específicas para essas linguagens, mais adiante neste capítulo.

Independentemente do conhecimento da ordem de precedência adotada na linguagem, o uso de parênteses é recomendado não só porque garante a correta avaliação das expressões, mas também porque facilita o entendimento do que está sendo executado.

Supondo:

a `i` uma variável inteira
b `r` uma variável real
c `c` uma variável do tipo caractere
d `achou` uma variável lógica

as expressões lógicas a seguir são válidas na pseudolinguagem:

```
( i ≠ 10 ) ou achou
( i mod 2 = 7 ) e ( r / 4 < 2 )
( r ≥ 0 ) e ( r + 1 < 10 ) ou achou
c = 'w' oux não achou
```

Encerra aqui o trecho do livro disponibilizado para esta Unidade de Aprendizagem. Na Biblioteca Virtual da Instituição, você encontra a obra na íntegra.