Inatel
Instituto Nacional de Telecomunicações

Núcleo Básico das Engenharias

C202-E/H
Algoritmos e Estruturas de Dados I

02 – Variáveis, Operadores e Expressões

Prof. Edson J. C. Gimenez
soned@inatel.br

2019/Sem1

Material adaptado de:
Algoritmos e Estruturas de Dados I
- Profa. Rosanna Mara Rocha Silveira
- Prof. Evandro Luís Brandão Gomes

Inatel
Instituto Nacional de Telecomunicações

Bibliografias

- ASCENCIO, Ana Fernanda Gomes. **Fundamentos da programação de computadores:** Algoritmos, pascal, C/C++ e java. 3. ed. São Paulo, SP: Pearson Prentice Hall, 2012.
- MIZRAHI, Victorine Viviane. **Treinamento em linguagem C++ - Módulo 1.** 2. ed. São Paulo, SP: Pearson Addison-Wesley, 2005.
- STROUSTRUP, Bjarne. **A linguagem de programação C++.** Tradução de Maria Lúcia Blanck Lisboa, Carlos Arthur Lang Lisbôa. 3. ed. Porto Alegre, RS: Editora Bookman, 2000.
- AGUILAR, Luis Joyanes. **Fundamentos de programação:** algoritmos, estruturas de dados e objetos. 3. ed. São Paulo, SP: McGraw-Hill, 2008.
- MORAES, Celso Roberto. **Estruturas de dados e algorítmos:** uma abordagem didática. São Paulo, SP: Editora Futura, 2003.
- etc...

Sintaxe e Semântica:

- Sintaxe é a forma como os comandos dever ser escritos, a fim de que possam ser entendidos pelo compilador.
- Semântica é o significado do comando, ou seja, o conjunto de ações que serão executadas pelo computador durante a execução do comando.

A partir deste capítulo, todos os comandos novos serão apresentados por meio de sua sintaxe e sua semântica, ou seja, a forma como devem ser escritos e as ações que cada um executa.

Comentários em um algoritmo (ou código em C/C++):

- Os comentários são usados para aumentar a clareza de um programa.
- Não tem nenhum significado com a execução do programa, sendo "desprezados" pelo compilador.

Exemplos de utilização:

/* entre barras e asteriscos, tudo fica como comentário */

// após barras duplas até o final da linha tudo fica como comentário

Estrutura básica de um algoritmo

```
Algoritmo Nome_do_Algoritmo
declare
        v1;
        v2;
        :
        vm;

<definição dos sub-algoritmos>
...
...

início
        comando1;
        comando2;
        :
        :
        comando n;
fim.
```

Nome do algoritmo

Informa quais as variáveis utilizadas no algoritmo e seus respectivos tipos (tipo de dado que pode armazenar.

Define quais sub-algoritmos são utilizados neste algoritmo, se houver algum.

Descrição da sequência de comandos necessária para a execução da função proposta.

Início e fim delimitam esta sequência de comandos

Obs.: Algoritmo, declare, início e **fim** são palavras reservadas da linguagem em pseudo-código.

Estrutura básica de um programa em C/C++

```
#include <iostream>          // padrão de entrada e saída no C++
#include <cmath>             // inclui funções matemáticas
#include <cstring>           // inclui funções de manipulação de string
#include <locale>            // inclui função para se usar palavras em português
#include <iomanip>           // inclui funções de configuração de saída

using namespace std;         // para usar bibliotecas

int main( )
{
   setlocale(LC_ALL,"Portuguese");   // habilita a estrutura ortográfica em português

   // declarações

   // entrada de dados

   // processamento

   // saída de dados

   return 0;
}
```


Tipos de dados básicos:

Dado numérico inteiro: é aquele pertencente ao conjunto dos números inteiros (Z). Pode ser positivo ou negativo. Embora possa ser representado pela classe dos números reais, é classificado como dado do tipo inteiro por permitir economia de espaço de memória e maior velocidade de cálculos.
Exemplos: 224 0 -112 2

Dado numérico real (ponto flutuante): é aquele que pode possuir componentes decimais ou fracionários. Pode ser negativo ou positivo. Os dados reais seguem a notação da língua inglesa, ou seja, a parte decimal é separada da parte inteira por um "." (ponto) e não por uma "," (vírgula).
Exemplos: 224.01 -13.3 0.5 -144.8 2.

Dado literal char: representa um único caractere, sendo apresentado entre ' ' (apostrofes). Ocupa um byte de memória
Exemplos: 'u' '5' 'C' '%'

Dado literal string: consiste de uma sequência de caracteres, sendo apresentado entre " " (aspas).
Exemplos: "qual ?" "1234.56" "Nome:" "alunos@inatel.br" "1 + 2 = 3"

Tipos de dados básicos:

Dado lógico ou *booleano*: usado para representar dois únicos valores lógicos possíveis: verdadeiro (*true*) e falso *(false*).
Obs: Em C/C++, o valor inteiro 0 (zero) equivale ao valor lógico *false;* qualquer valor diferente de 0 é considerado *true*.

Tipos comuns em C/C++:

Tipo	Bytes	Valores armazenados
char	1	-128 a 127 (caracteres).
int	2	-32.768 a 32.767 (ambiente de 16 bits)
int	4	-2.147.483.648 a 2.147.483.647 (amb. de 32 bits)
short	2	-32.765 a 32.767
long	4	-2.147.483.648 a 2.147.483.647
float	4	$-3.4x10^{-38}$ a $3.4x10^{38}$
double	8	$1.7x10^{-308}$ a $1.7x10^{308}$
long double	10	$3.4x10^{-4932}$ a $3.4x10^{4932}$
void	0	nenhum valor
unsigned char	1	0 a 255 (caracteres)
unsigned long	4	0 a 4.294.967.295
unsigned short	2	0 a 65.535

Variáveis

Entidade destinada a guardar uma informação, podendo esta ser de diferentes tipos, ocupando um ou mais bytes de memória, dependendo de seu tipo

Cada variável ocupa uma posição de memória e seu conteúdo pode variar durante a execução de um programa, porém, em um determinado instante, possui apenas um valor.

Toda variável é identificada por um nome ou identificador.

Este nome deve ser formado por um ou mais caracteres, sendo que o primeiro deve ser uma letra (maiúscula ou minúscula) ou o caractere sublinhado (_) e os demais, letras (maiúsculas ou minúsculas), dígitos ou o caractere sublinhado (_).

Não é permitido o uso de símbolos especiais, exceto o sublinhado (_), nem o uso de palavras reservadas (palavras que pertencem à linguagem**).**

Identificadores permitidos	Identificadores não permitidos
a	5b
nota	e(13)
a32b	a:b
_ndepend	123
x25	for
matricula	nome comprido
nome_comprido	matrícula
caixa_preta	if

Algumas linguagens de programação, na maioria, são *case sensitive*, isto é, diferenciam maiúsculas de minúsculas (caso da linguguagem C/C++).

Sintaxe de declaração de variáveis em algoritmos

Todas as variáveis utilizadas em um algoritmo devem ser declaradas antes de serem utilizadas.

Utiliza-se a palavra-chave **declare**, o(s) nome(s) da(s) variável(eis) e o tipo de dado da(s) variável(eis).

```
declare
        nome1  tipo1;
        nome2, nome3 tipo2;
        ...
        nomeM tipoM;
```

Exemplo:

```
declare
        media_aluno, matricula, idade numérico;
        sexo, nomedoaluno literal;
        teste lógico;
```

Em que:

• **nome1**, **nome2**, ..., **nomeM** são os nomes dados às variáveis.

• **tipo1**, **tipo2**, **tipoM** são os tipos de dados das variáveis (numérico, literal, e lógico).

Obs.: variáveis do mesmo tipo, podem ser declaradas na mesma linha, separadas por vírgula; variáveis de tipos diferentes devem ser declaradas em linhas diferentes, finalizadas com ponto e vírgula (**;**);

Sintaxe de declaração de variáveis em C/C++

```
tipo1 nome1;
tipo2 nome2, nome3;
        ...
tipoM nomeM = valor;
```

Exemplo:

```
float     media_aluno = 0, nota1, nota2;
int       matricula, idade;
char      sexo, nome[51];
string  aluno   // cuidado!!!
```

Em que:

• **nome1**, **nome2**, ..., **nomeM** são os nomes dados às variáveis; devem ter no máximo 247 caracteres.

• **tipo1**, **tipo2**, **tipoM** são os tipos de dados das variáveis.

Obs.: variáveis do mesmo tipo, podem ser declaradas na mesma linha, separadas por vírgula; variáveis de tipos diferentes devem ser declaradas em linhas diferentes, finalizadas com ponto e vírgula (**;**).

Obs.: a linguagem C/C++ permite na declaração de uma variável já atribuir um valor inicial à mesma. Exemplo: float soma = 0;

Obs.: Em C, strings são implementadas através de vetores de caracteres (tipo char), cujo último elemento deve ser um caractere nulo (\0). Já em C++, pode-se o tipo string para declarar uma string de tamanho variável; porém, deve ser incluída a biblioteca string (#include <string>).

Ap1.3) Verifique quais os identificadores são válidos. Se não, explique por quê:

a) abc#3

b) _matric

c) numero_de_pontos_obtidos

d) acd1

e) aluno@inatel.br

f) 1nota

g) x y za

h) sal/hora

Ap1.4) Supondo que as variáveis nt, na, nmat, sx sejam utilizadas para armazenar a nota do aluno, o nome do aluno, o número de matrícula e o sexo, respectivamente, declare-as corretamente, associando os tipos adequados aos dados que serão armazenados.

Ap1.5) Encontre os erros das seguintes declarações de variáveis:

a) int endereço-aluno, nfilhos, 1peso;
b) bool Lâmpada:;
c) string idade;

Expressões Aritméticas

Denomina-se expressão aritmética aquela cujos operadores são aritméticos e cujos operandos são variáveis ou constantes do tipo numérico.

Principais operadores aritméticos em C/C++:

Operador	*Finalidade*
=	Atribuição
+	Adição
-	Subtração
*	Multiplicação
/	Divisão
%	Resto da divisão de 2 inteiros
*=	Aritmético de atribuição (multiplicação)
/=	Aritmético de atribuição (divisão)
%=	Aritmético de atribuição (módulo)
+=	Aritmético de atribuição (soma)
-=	Aritmético de atribuição (subtração)
++	Incremento de um
--	Decremento de um

Expressões Aritméticas

Exemplos de expressões aritméticas onde aparecem as operações de adição, subtração, multiplicação, divisão real, resto de divisão, potenciação e radiciação:

a) x + y
b) sqrt(p)
c) x--
d) a / b
e) a % b
i) pow(a, (1/2))
b) x – y
j) x++
c) 2 * nota
g) pow(soma,2)
h) pow(soma, z)

Obs.: CUIDADO!!! Divisão de inteiros, resulta em um valor inteiro (truncado).

Exemplo: 2 / 3 = 0 2. / 3 = 0.666667 → Qual a diferença?

PRIORIDADE DAS OPERAÇÕES	
PRIORIDADE	**OPERAÇÃO**
1a.	POTENCIAÇÃO, RADICIAÇÃO
2a.	MULTIPLICAÇÃO, DIVISÃO, RESTO DA DIVISÃO
3a.	ADIÇÃO, SUBTRAÇÃO

Para a quebra de prioridades faz-se uso de parênteses.

Exemplo: a) (43 * (55 / (30 + 2)))

b) (2+a)/(b-3) – 2*x + **pow**(x,3)

Expressões aritméticas - funções

Além das operações básicas, podem-se usar nas expressões aritméticas algumas funções comuns na Matemática, funções estas incluídas na biblioteca matemática da linguagem.

Em C/C++, estas se encontram na biblioteca matemática (math.h ou cmath).

Exemplo de algumas funções matemáticas do C/C++:

sin(a)	seno de A
cos(a)	cosseno de A
log10(a)	logaritmo na base 10 de A
log(a)	logaritmo natural (base e) de A
exp(a)	a-ésima potência de *e* (e^a)
abs(a)	valor absoluto de a
ceil(a)	arredonda um número real para cima
floor(a)	arredonda um número real para baixo
random(a)	número aleatório entre 0 e a inteiro

AP 1.6) Sendo a, b, x e y variáveis do tipo numérico, qual o resultado de cada uma das seguintes expressões, sendo a = 10, b = 3, x = 2.5 e y = 1.2

a) A / B A % B

b) X / 2 X % 2

c) **floor**(**pow**(b,2) + x) **ceil**(a/3 + 1) **floor**(y – x)

d) **abs**(a – **pow**(b,3)) floor(x-3.2) **abs**(a - b) **ceil**(a – x)

e) (b + y) / (x + 1)

Exercício 1.6)

```
#include <stdio.h>
#include <conio.h>
#include <math.h>

#include <iostream>
using namespace std;

int main()
{
int A = 10, B = 3;
float X = 2.5, Y = 1.2;

cout<<"A/B="<<A/B <<endl;
cout<<"**A/B="<< float(A)/B <<endl;
cout<<"A%B="<< A%B <<endl;
cout<<"X/2="<<X/2 <<endl;
cout<<"X%2="<<"ERRO % somente para inteiros" <<endl;
cout<<"floor(pow(B,2)+X)="<< floor(pow(B,2)+X) <<endl;
cout<<"ceil(A/3+1)="<< ceil(A/3+1) <<endl;
cout<<"floor(Y-X)="<< floor(Y-X) <<endl;
cout<<"abs(A-pow(B,3))="<< fabs(A-pow(B,3)) <<endl;
cout<<"floor(X-3.2)="<< floor(X-3.2) <<endl;
cout<<"abs(A-B)="<< fabs(A-B) <<endl;
cout<<"ceil(A-X)="<< ceil(A-X) <<endl;
cout<<"(B+Y)/(X+1)="<< (B+Y)/(X+1) <<endl;

return 0;
}
```

```
C:\Users\edsonjcg\Documents\2018_Sem1\
A/B=3
**A/B=3.33333
A%B=1
X/2=1.25
X%2=ERRO % somente para inteiros
floor(pow(B,2)+X)=11
ceil(A/3+1)=4
floor(Y-X)=-2
abs(A-pow(B,3))=17
floor(X-3.2)=-1
abs(A-B)=7
ceil(A-X)=8
(B+Y)/(X+1)=1.2

O Processo retornou 0   tempo de
Pressione uma tecla para continua
```

Expressões Lógicas

Denomina-se expressão lógica a expressão cujos operadores são lógicos e cujos operandos são relações e variáveis de diversos tipos.

Operadores Relacionais - indicam a comparação a ser realizada entre os termos da relação:

Operador	*Operação*
= =	igual a
! =	diferente de
>	maior que
<	menor que
> =	maior ou igual a
< =	menor ou igual a

O resultado obtido de uma relação é sempre um valor lógico: falso ou verdadeiro.

Exemplos:
a != b
nome == "JOAO"
x == 1
(pow(b,2) - 4*a*c) < 0

Ap1.7) Dadas as variáveis numéricas X, Y, Z e as variáveis literais NOME e COR, observar os resultados obtidos para as relações a partir dos valores atribuídos a estas variáveis.

	VARIÁVEIS					**RELAÇÕES**		
	X	**Y**	**Z**	**cor**	**nome**	**(pow(X,2)+Y) > Z**	**cor == "AZUL"**	**nome != "JOSÉ"**
a)	1	2	5	'AZUL'	'PAULO'			
b)	1	1	2	'BRANCO'	'PEDRO'			

Expressões Lógicas

Operadores Lógicos:

e (&&) - para a CONJUNÇÃO
ou (||) - para a DISJUNÇÃO
não (!) - para a NEGAÇÃO

Tabelas Verdade

É o conjunto de todas as possibilidades combinatórias entre os diversos valores lógicos, os quais se encontram em apenas duas situações (F ou V), e um conjunto de operadores lógicos.

A	B	A && B
F	F	F
F	V	F
V	F	F
V	V	V

A	!A
F	V
V	F

A	B	A || B
F	F	F
F	V	V
V	F	V
V	V	V

Prioridade dos Operadores

Como podemos ter mais de um operador lógico na mesma expressão, a ordem em que são efetuadas estas operações afeta o resultado final.

Assim, como acontece entre as operações aritméticas, também existe uma relação de prioridade entre os operadores lógicos conforme tabela abaixo.

PRIORIDADE	*OPERADOR*
1ª.	ARITMÉTICO
2ª.	RELACIONAL
3ª.	NÃO
4ª.	E
5ª.	OU

Ap1.8) Dadas as variáveis numéricas **x**, **y** e **z**, contendo os valores 2, 5 e 9, respectivamente; a variável string **nome**, contendo "MARIA"; e a variável bool **sim**, contendo o valor lógico *false,* observar os resultados obtidos das expressões lógicas a seguir.

a) (x + y > z) **&&** (nome == "MARIA")

d) (nome == "JORGE") **&&** sim || (pow(x,2) < z + 10)

b) sim || (y >= x)

c) **!**(sim) **&&** (((z / y) + 1) == x)

Exercícios do capítulo 1