Unidade 4: Introdução à Lógica de Programação - Parte II

Português Estruturado

Prof. Daniel Caetano

Objetivo: Explicitar os elementos básicos envolvidos na programação e apresentar o Português Estruturado

Bibliografia: ASCENCIO, 2007; MEDINA, 2006; SILVA, 2010; SILVA, 2006.

INTRODUÇÃO

Nas aulas anteriores estudamos os conceitos básicos de programação e, ao final, um pequeno programa em Português Estruturado foi construído. Nesta aula, os conceitos apresentados no programa serão melhor apresentados, juntamente com os primeiros detalhes sobre o Português Estruturado.

Ao final desta aula, você saberá como declarar variáveis, como ler e escrever valores, como executar expressões aritméticas e armazenar seus resultados em variáveis, usando a representação do Português Estruturado.

1. VARIÁVEIS

Já tomamos contato com variáveis nas aulas anteriores, embora de uma maneira pouco explícita. O que são variáveis?

Em linhas gerais, podemos entender as **variáveis** como "**uma caixa onde posso armazenar um valor e da qual, posteriormente, posso recuperá-lo**", sendo que **sempre daremos um nome** à esta caixa. Essa interpretação, embora prática, não explica algumas coisas. Tomemos o exemplo abaixo em portugol:

```
ALGORITMO "Teste de variáveis"
VAR
INICIO
idade <- 20
escreva ("Minha idade é", idade)
FIMALGORITMO
```

Este programa simplesmente armazena o valor 20 na variável idade e, depois, imprime o valor dessa variável como parte da frase "Minha idade é xxx". Se você digitar esse programa no VisuAlg, entretanto, perceberá que ele acusará erro na seguinte linha:

idade <- 20

Por quê? Bem, porque não **declaramos** a variável. Toda vez que pretendemos usar uma variável, somos **obrigados** a declarar isso para o computador, no início do programa. Isso poderia ser feito dessa forma:

```
ALGORITMO "Teste de variáveis"
VAR
idade
INICIO
idade <- 20
escreva ("Minha idade é", idade)
FIMALGORITMO
```

Isso teria declarado a variável, mas **ainda não vai funcionar**. Para entender porque o problema ocorre, precisamos nos lembrar o que são as variáveis **de fato**.

Na aula de organização de computadores, foi apresentada a natureza da memória do computador, isto é, que ela composta por fios que podem estar conduzindo corrente ou não. Convencionou-se chamar a condução de corrente de valor 1 e a não condução de valor 0. Cada fio representa, então, **1 bit**. Cada posição de memória é representada por **8 bits**, conjunto ao qual se dá o nome de **byte.**

Ainda naquela aula, foi visto como um mesmo conjunto de 8 bits pode ser interpretado de diferentes formas: um número inteiro sem sinal (0 a 255), com sinal (-128 a 127) e, ainda, um número com vírgula (ponto flutuante). Ora, se o valor de uma mesma posição de memória pode ser interpretado de diferentes formas, é necessário **dizer ao computador como interpretá-lo**.

Considerando que o nome de uma variável nada mais é que **um apelido que se dá a uma posição de memória** (de maneira que não tenhamos que decorar um monte de números), quando declaramos o nome da variável **também** temos que declarar o **tipo da variável**. Os tipos comuns são: **inteiro**, **real** e **caractere** (literal). A função deles está explícita abaixo:

Inteiro: serve para armazenar números SEM vírgula.
Real: serve para armazenar números COM vírgula.
Caractere: serve para armazenar letras.

Como o computador é um equipamento que trabalha muito com operações lógicas, algumas linguagens apresentam uma variável do tipo **logico**, que apenas armazena valores **falso** e **verdadeiro**.

No caso da idade, normalmente ela é tratada como um valor **inteiro**, já que ninguém diz ter 20,32 anos. Assim, vamos corrigir o programa anterior da seguinte forma:

```
ALGORITMO "Teste de variáveis"
VAR
idade : inteiro
INICIO
idade <- 20
escreva ("Minha idade é", idade)
FIMALGORITMO
```

Em portugol, a declaração de uma variável é sempre feita na região entre a palavra VAR e a palavra INICIO. A declaração é feita no seguinte formato:

nome_da_variavel **:** tipo_da_variavel

Observe que o que separa o nome da varivel do tipo da variável é um sinal de dois pontos, e **ele é importante**.

Ao declararmos as variáveis, podemos indicar uma variável por linha:

```
ALGORITMO "Teste de variáveis"
VAR
idade : inteiro
dia : inteiro
nota : real
frequencia : real
INICIO
FIMALGORITMO
```

Ou podemos agrupar as variáveis do mesmo tipo na mesma linha, separando seus nomes por vírgula:

```
ALGORITMO "Teste de variáveis"
VAR
idade, dia : inteiro
nota, frequencia : real
INICIO
FIMALGORITMO
```

2. OPERADOR DE ATRIBUIÇÃO

Damos o nome de "atribuição" à operação de armazenar um valor em uma variável. Em portugol, a atribuição é sempre indicada no seguinte formato:

nome_da_variavel **<-** expressao

Em portugol, o sinal de atribuição é esse: <- (um sinal de **menor** seguido de um sinal de **menos**). Do lado esquerdo do sinal de atribuição há **sempre** o nome de uma variável, que indica **onde** eu quero guardar o valor.

O lado direito do sinal de atribuição, por sua vez, pode ter um simples valor (número ou texto, depenendo do tipo de variável que está do lado esquerdo), ou pode ter uma expressão matemática como, por exemplo, 2+5. A expressão pode conter variáveis também!

Observe que, como o computador só armazena **números** na memória, se o lado direito da atribuição for composto por uma expressão, o computador **primeiro resolve a expressão, transformando-a em um número** e só depois disso armazena o resultado na variável. Desta forma, a expressão destacada abaixo é perfeitamente válida:

```
ALGORITMO "Teste de variáveis"
VAR
idade : inteiro
INICIO
idade <- 20
idade <- idade + 1
escreva ("Minha idade é", idade)
FIMALGORITMO
```

Depois da linha destacada, a variável idade valera 21. Pense como o computador faz o processamento: na linha anterior, o valor 20 era atribuído à variável idade; na linha destacada, PRIMEIRO ele realiza o cálculo do lado direito: idade + 1 => 20 + 1 => 21. Agora ele lê essa linha destacada como "idade <- 21" e armazena o valor 21 na variável idade.

3. OPERADORES MATEMÁTICOS

Quase todos os operadores básicos estão disponíveis no portugol. Os símbolos usados para cada um deles estão indicados a seguir, bem como sua prioridade:

Operação	Sinal	Prioridade
Adição:	+	1
Subtração:	-	1
Multiplicação:	*	2
Divisão:	/	2
Divisão Inteira:	\	2
Resto da Divisão:	%	2
Exponenciação:	^	3

Qualquer um deles pode ser usado diretamente em uma atribuição. Por exemplo: para guardar o resto da divisão de 37 por 7 na variável X, basta escrever usar o código:

X <- 37 % 7

O número de prioridade é útil quando se realiza várias operações na expressão. Um número de prioridade maior significa que a operação será executada primeiro, independente da ordem em que aparece na expressão. Por exemplo:

X <- A + B * 2

Será executado na seguinte ordem:

a) temp1 <- B*2
b) temp2 <- A + temp1
c) X <- temp2

Quando as operações tiverem a mesma prioridade, elas serão executadas da esquerda para a direita. Por exemplo:

X <- A + B - C

Será executado na seguinte ordem:

a) temp1 <- A*B
b) temp2 <- temp1 - C
c) X <- temp2

Como na matemática, é possível usar parênteses para forçar uma ordem específica de cálculo. Por exemplo:

X <- (A + B) * 2

Isso forçará a seguinte ordem de execução:

a) temp1 <- A+B
b) temp2 <- temp1 * 2
c) X <- temp2

4. SAÍDA DE DADOS

Um dos recursos mais importantes do computador é propiciar a saída de dados. De nada adiantaria que eles nos fizesse inúmeras contas se não pudéssemos ver o resultado das mesmas. Para solicitar que o computador **escreva algo na tela**, no portugol, usamos a instrução **escreva**, que tem a seguinte sintaxe:

escreva (dado_a_ser_escrito)

Dentro do parênteses, **dado a ser escrito** pode ser um número:

```
ALGORITMO "Teste de escrita"
VAR
INICIO
escreva(10)
FIMALGORITMO
```

Um texto (lembrando que texto deve SEMPRE vir entre aspas:

```
ALGORITMO "Teste de escrita"
VAR
INICIO
escreva("um texto qualquer")
FIMALGORITMO
```

Ou, ainda, pode ser um nome de variável, situação na qual o portugol irá imprimir **o valor da variável**:

```
ALGORITMO "Teste de escrita"
VAR
idade : inteiro
INICIO
idade <- 20
escreva( idade )
FIMALGORITMO
```

O comando escreva aceita, ainda, que indiquemos uma expressão matemática, quando então ele **imprime o resultado da expressão**:

```
ALGORITMO "Teste de escrita"
VAR
INICIO
escreva( (20+10)*3 )
FIMALGORITMO
```

Podemos compor uma linha com várias instruções **escreva**:

```
ALGORITMO "Teste de escrita"
VAR
idade : inteiro
INICIO
idade <- 20
escreva( "Minha idade é: " )
escreva( idade )
FIMALGORITMO
```

Ou podemos pedir que um único escreva imprima várias coisas, separando-as por vírgula:

```
ALGORITMO "Teste de escrita"
VAR
idade : inteiro
INICIO
idade <- 20
escreva( "Minha idade é: ", idade )
FIMALGORITMO
```

NOTA: Para "pular uma linha" após o texto escrito, use a instrução **escreval** (observe que há uma letra "ele" a mais no final, indicando para pular **L**inha depois do texto).

5. ENTRADA DE DADOS

Quase tão importante quanto a saída de dados, é a entrada de dados: é a entrada de dados que permite que, a cada execução, o computador execute um cálculo diferente e útil ao usuário! Para solicitar que o computador **leia um dado do teclado**, no portugol, usamos a instrução **leia**, que tem a seguinte sintaxe:

leia (nome_da_variavel)

Observe que a sintaxe é bem menos flexível que a do escreva; cada valor deve ser lido com uma linha "leia" e, adicionalmente, o valor digitado pelo usuário **deve** ser do mesmo tipo da variável. Observe o programa abaixo:

```
ALGORITMO "Teste de Entrada de Dados"
VAR
idade : inteiro
INICIO
escreva( "Digite sua idade: ")
leia (idade)
escreva( "Minha idade é: ", idade )
FIMALGORITMO
```

Como idade é um inteiro, se o usuário digitar algo como **20,5** ou mesmo escrever a palavra **vinte** , o programa provavelmente vai parar indicando um erro.

7. BIBLIOGRAFIA

ASCENCIO, A.F.G; CAMPOS, E.A.V. **Fundamentos da Programação de Computadores**. 2ed. Rio de Janeiro, 2007.

MEDINA, M; FERTIG, C. **Algoritmos e Programação: Teoria e Prática**. 2ed. São Paulo: Ed. Novatec, 2006.

SILVA, I.C.S; FALKEMBACH, G.M; SILVEIRA, S.R. **Algoritmos e Programação em Linguagem C**. 1ed. Porto Alegre: Ed. UniRitter, 2010.