Unidade 4: Introdução à Lógica de Programação - Parte II

Português Estruturado - "Versão Python"

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"
idade <- 20
escreva ("Minha idade é", idade)
```

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".

Uma variável pode guardar dados dos seguintes tipos:

Inteiro: serve para armazenar números SEM vírgula.
Real: serve para armazenar números COM vírgula.
Texto: serve para armazenar letras.
Lógico: server para armazenar os valores “true” ou “false”.

No caso da idade, armazenamos um valor inteiro nela - o 20. **A partir do momento em que armazenamos um valor em uma variável, podemos utilizá-la em nosso programa para realizar outros cálculos**.

Sendo assim, dizemos que ao armazenar um valor em uma variável, estamos “**declarando**” a variável, porque a partir desse ponto o programa passa a reconhecê-la.

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**, construindo uma espécie de seta para a esquerda). 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), 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"

idade <- 20
idade <- idade + 1
escreva ("Minha idade é", idade)
```

Depois da linha destacada, a variável idade valerá 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 “para baixo”:	//	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)

Após a plavra "escreva", **dado a ser escrito** pode ser um número:

```
ALGORITMO "Teste de escrita"

escreva (10)
```

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

```
ALGORITMO "Teste de escrita"

escreva ("um texto qualquer")
```

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

```
ALGORITMO "Teste de escrita"

idade <- 20
escreva idade
```

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

```
ALGORITMO "Teste de escrita"

escreva (20+10)*3
```

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

```
ALGORITMO "Teste de escrita"

idade <- 20
escreva "Minha idade é: "
escreva idade
```

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

```
ALGORITMO "Teste de escrita"

idade <- 20
escreva "Minha idade é: ", idade
```

NOTA: Para "pular uma linha" após o texto escrito, use sempre vários comandos “escreva”. Separando com vírgulas, nenhuma linha será pulada entre os valores.

IMPORTANTE: para imprimir o valor de uma variável, devemos primeiramente declará-la, isto é, armazenar um valor inicial. O programa abaixo, por exemplo, indicaria erro na linha do “escreva”:

```
ALGORITMO "Teste de escrita"

escreva "Minha idade é: ", idade
```

A versão abaixo, por sua vez, funciona normalmente. Observe que a única diferença é que no segundo caso foi armazenado um valor em “idade” antes de se tentar apresentar seu valor para o usuário:

```
ALGORITMO "Teste de escrita"

idade <- 20
escreva "Minha idade é: ", idade
```

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 (“Pergunta para o usuário”)

Esse comando permite que o usuário digite um valor, mas para guardá-lo é preciso armazenar o resultado em uma variável, isto é, usando uma atribuição:

variavel = leia (“Pergunta para o usuário”)

Observe o programa abaixo:

```
ALGORITMO "Teste de Entrada de Dados"

idade = leia(“Digite sua idade:”)
escreva ("Minha idade é: ", idade)
```

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.