Mãos à obra! - Respostas

Variáveis, Operadores e Funções

Exercício 1

A proposta do primeiro exercício é substituir as três fórmulas seguintes por **linhas de programação**, ou seja, deixá-las linearizadas. Para isso, vamos precisar entender o funcionamento dos operadores em uma linguagem de programação.

$$Y = A + 3 \times B^2$$

$$Z = \frac{A + B}{5 \times C}$$

$$S = 5^2 \times \left((2 \times 5) + \left(\frac{15}{3} \right) \right)^3$$

Veja a primeira fórmula:

$$Y = A + 3 \times B^2$$

A variável **Y** receberá o valor de **A** mais 3 vezes B ao quadrado. Seguindo os princípios de prioridade dos operadores, não precisamos nos preocupar em colocar parênteses nas operações, já que a multiplicação terá prioridade sobre a adição. Como já vimos na aula, o mesmo se aplica à programação. Sabendo disso, transformar esta fórmula em uma linha de programação é bem simples. Veja como ficou:

```
Y = A + 3 * B ^ 2
```

Não se esqueça que, na programação, o operador de multiplicação é representado por um asterisco e o de potenciação, por um acento circunflexo.

Na segunda fórmula do exercício, a variável Z receberá o resultado de uma divisão.

$$Z = \frac{A + B}{5 \; x \; C}$$

O intuito, neste caso, é que primeiramente sejam realizadas as operações de adição e multiplicação para, então, realizar a divisão com o resultado das duas.

Para que o nosso programa interprete a fórmula corretamente, será necessário inserir parênteses nas operações que iremos realizar primeiro. Levando isso em consideração, conseguimos chegar neste resultado:

```
Z = ( A + B ) / ( 5 * C )
```

Vamos para a terceira e última fórmula:

$$S = \; 5^2 \; x \left((2 \; x \; 5) + \left(\frac{15}{3} \right) \right)^3$$

Apesar de parecer complexo, tudo que precisamos fazer é escrevê-la de uma maneira que o computador consiga interpretar corretamente.

A primeira parte consiste em uma potenciação, que terá seu resultado multiplicado por tudo o que está dentro dos parênteses. Não vamos precisar mexer nos parênteses, já que eles vão evitar que o computador realize uma operação errada. O único trabalho que teremos, neste caso, será a troca dos operadores pelos símbolos utilizados em programação. Veja só como ficou:

```
5 ^ 2 * ( (2 * 5) + (15/3) ) ^ 3
```

Exercício 2

Desta vez, a proposta é criar um algoritmo que faça a soma de vários números digitados pelo usuário, mostrando no final a soma total e a quantidade de números que foram digitados. Veja:

Crie um algoritmo que realize as seguintes tarefas enquanto números são digitados em uma variável de nome NRO:

- Conte os números digitados;

- Acumule os valores dos números digitados;

- Ao final, exiba quantos números foram digitados e a soma dos valores digitados.

A resposta é esta:

```
INICIO
    Declara CN, NRO, SNRO numéricas
    CN = 0
    SNRO = 0
    Enquanto NRO <> 0
        LER NRO
        CN = CN + 1
        SNRO = SNRO + NRO
    Fim do Enquanto
    Exibir CN, SNRO
FIM
```

Para começar, vamos analisar o problema e verificar quais variáveis serão necessárias. Como precisamos contar a quantidade de números que o usuário digitou, vamos utilizar uma variável que funcionará como um **contador**, ou seja, sempre que for digitado um novo número, o valor dessa variável será incrementado.

Contador: CN

Precisaremos, também, atribuir os números que o usuário digitou a uma variável. Ao invés de utilizar diversas variáveis para guardar todos os números, utilizaremos apenas uma, que armazenará o número temporariamente.

Número digitado: NRO

Para finalizar, será necessária uma última variável, que será responsável por armazenar a soma dos números que o usuário digitou, em um processo contínuo.

Soma dos números digitados: SNRO

Agora que já definimos as variáveis do nosso algoritmo, podemos começar a desenvolvê-lo. Vamos declarar as variáveis numéricas que acabamos de definir, para que seja possível utilizá-las durante o algoritmo.

INÍCIO
    Declara CN, NRO, SNRO numéricas

Precisamos, agora, atribuir o valor **zero** às variáveis de contador e de soma dos números. Note que isso não será feito com a variável que representa o número digitado, já que o próprio usuário irá atribuir um valor a ela.

    CN = 0
    SNRO = 0

Para guardar o número que o usuário digitou temporariamente, vamos utilizar um laço que irá validar uma condição e, a partir disso, pedir para o usuário digitar um número, incrementar o nosso contador e somar os números que o usuário já colocou:

```
Enquanto (???)
    LER NRO
    CN = CN + 1
    SNRO = SNRO + NRO
Fim do Enquanto
```

Apenas com esse laço, já conseguimos fazer toda a operação principal do nosso algoritmo. Perceba que a condição do laço foi deixada em branco. Já sabemos qual é o trecho de código que o laço irá executar. Mas qual condição será validada antes disso ser feito?

Uma maneira de analisar seria a seguinte: Se queremos que o usuário digite uma certa quantidade de vezes, podemos limitar a execução do laço através do nosso contador. Por exemplo, o laço será executado **ENQUANTO** o contador for menor que 5, ou seja, **ENQUANTO CN < 5**.

Uma maneira melhor para resolver isso é definir um valor que, quando digitado pelo usuário, irá parar a execução do laço. Podemos definir que **ENQUANTO** o número digitado for diferente de **0** (**zero**), por exemplo, o laço continuará em execução; ou seja, quando o usuário digitar **zero**, o laço deixará de ser executado. Atente para o operador que utilizamos para indicar **diferença** (**<>**):

```
Enquanto NRO <> 0
    LER NRO
    CN = CN + 1
    SNRO = SNRO + NRO
Fim do Enquanto
```

Agora que já temos o laço para fazer as principais operações, basta exibir os resultados obtidos, que são o número de vezes que o usuário digitou algum número até sair do laço e a soma total dos números digitados.

```
Exibir CN, SNRO
FIM
```