**Passa Parâmetro por Valor**

***Mário Leite***

...

A “Programa Estruturada”, embora criticada por muitos programadores “modernos” (as intencionais) trouxe um novo recurso de implementação nas linguagens modernas: **“Passagem de Parâmetros”**, que pode ser definido como: **“*mecanismo pelo qual valores externos são entregues a uma sub-rotina, com os quais ela deve operar para executar a tarefa que lhe foi atribuída*”**. Esse(s) valor(es) externos são fornecidos por uma rotina *chamadora* que são passados para alguma sub-rotina (*chamada*) encarregada de executar determinada tarefa com esse(s) valor(es) recebido(s). Seja em qualquer paradigma de programação, existem dois tipos desse mecanismo: “Passagem Por Valor” e “Passagem Por Referência”. Nesta postagem valor nos ater ao primeiro tipo de passagem de parâmetros, que pode ser explicado pela seguinte situação prática: imagine que “*o chefe de um escritório (rotina principal) chama um office-boy (sub-rotina) e passa-lhe uma cédula de R$ 200,00 (parâmetro) para pagar uma duplicata numa agência bancária*”. Assim, com essa analogia, teremos o seguinte:

**OfficeBoyPagaDuplicata**(200)

Nome da sub-rotina (Argumento)

Mas... e se o valor da duplicata fosse, por exemplo, R$2.345,76 em vez de R$ 200,00 !?

Bem, neste caso bastaria dar em dinheiro esse valor, pois o que mudaria seria apenas o valor a ser pago; não precisaria trocar de funcionário (criar outra sub-rotina). Nesta nova situação a chamada seria assim:

**OfficeBoyPagaDuplicata**(2345.76)

Na passagem “Por Valor” o que é passado, tecnicamente, para a rotina chamada é uma cópia do valor da variável envolvida, e fazendo analogia com o caso do chefe do *office-boy* que o chama e passa-lhe uma cédula de R$200,00 para pagar uma duplicata. E caso o funcionário resolva alterar o valor recebido ele não conseguirá, pois o pagamento efetuado sempre constará R$ 200,00, pois é impossível alterar a cédula, e o que foi contabilizado no caixa da empresa. Isto quer dizer que na “Passagem por Valor”, caso esse valor seja alterado pela sub-rotina *chamada* tal alteração não será refletida na rotina *chamadora*.

O programa **“ProgPassaParamVal”**, codificado em Pascal, C# e Python, apresenta um exemplo de implementação para mostrar como funciona a passagem de parâmetros por valor, mostrando que o valor passado não é impactado na rotina principal (*chamadora*) quando a rotina *chamada* altera esse valor recebido. As **figuras 1**, **2** e **3** mostram as saídas do mesmo programa nessas três linguagens respectivas.

--------------------------------------------------------------------------------------------------------------------------------------

**Nota**: A linguagem Pascal foi criada por **Niklaus Wirth** em 1970 (falecido em janeiro/2024). E embora muitos programadores a desprezem, considerando-a como linguagem “ultrapassada”, ela ainda é a MELHOR para ensinar “Lógica de programação”, com uma excelente estrutura e fácil de entender.

--------------------------------------------------------------------------------------------------------------------------------------

**Program** **ProgPassaParamVal**;

**{ProgPassaParamVal.pas**

Mostra como funciona a passagem de parâmetros por Valor. Se a rotina chamada alterar o parâmetro recebido, esta alteração NÃO será refletida na rotina que a

chamou.

-------------------------------------------------------------------------------

**}**

**Uses** Crt;

**var** x: **integer**;

y: **real**;

*//--------------------------------------------------------------------------------------*

**Function** **Inverter**(n:**integer**): **real**;

*//Retorna um valor passado Por Valor*

**Begin**

**Inverter** := 1/n; *//retorno normal da função*

n := 5; *//a função altera o valor do parâmetro recebido* ***x***

**End**; *//fim da função "****Inverter****"*

*//=====================================================================================*

*//Programa principal*

**BEGIN**

**ClrScr**;

**Write**('Digite um número: ');

**Readln**(x);

**WriteLn**(**''**);

**WriteLn**('Valor de x na ida: ', x); *//valor de* ***x*** *antes de chamar a função*

y := **Inverter**(x); *//chama a função e lhe passa o valor de* ***x***

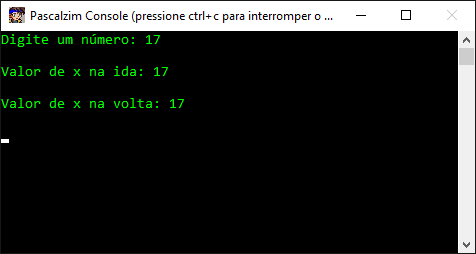
**WriteLn**('');

**WriteLn**('Valor de x na volta: ', x); *//valor de* ***x*** *NÃO é alterado depois da função*

**WriteLn**('');

**ReadKey**;

**END**. *//fim do programa "****ProgPassaParamRef****"*



**Figura 1 - Saída do programa em Pascal**

/\***ProgPassaParamVal.cs**

Mostra como funciona a passagem de parâmetros por Valor. Se a rotina chamada alterar o parâmetro recebido, esta alteração NÃO será refletida na rotina que a

chamou.

\*/

//-------------------------------------------------------------------------------

**using** System;

**namespace** **ProgPassaParamVal**

{

**class** **ProgPassaParamVal**

{

*//Função que recebe o parâmetro por valor*

**static** **double** **Inverter**(**int** n)

{

**double** resultado = 1.0 / n; *//retorno normal da função*

n = 5; *// a função altera o valor do parâmetro recebido...*

**return** resultado;

}

**static** **void** **Main**()

{

**int** x;

**double** y;

*//Solicitando o valor de* ***x***

**Console**.Write("Digite um número: ");

x = Convert.ToInt32(**Console**.ReadLine());

**Console**.WriteLine();

**Console**.WriteLine("Valor de x na ida: " + x); *//valor de* ***x*** *antes da chamada*

y = **Inverter** (x); *//chama a função e passa o valor de* ***x***

**Console**.WriteLine();

**Console**.WriteLine("Valor de x na volta: " + x); *//valor de* ***x*** *NÃO é alterado*

**Console***.WriteLine();*

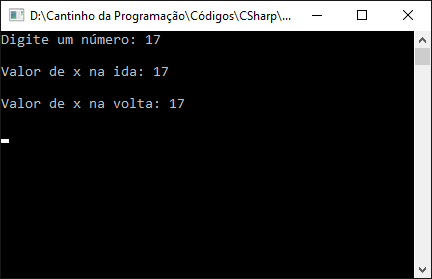
*//Pausa para ver o resultado*

**Console**.ReadKey();

}

}

} *//fim do programa "****ProgPassaParamRef****"*



**Figura 2 - Saída do programa em C#**

'''

**ProgPassaParamVal.py**

--------------------------------------------------------------------------------

Mostra como funciona a passagem de parâmetros por Valor. Se a rotina chamada alterar o parâmetro recebido, esta alteração NÃO será refletida na rotina que a chamou.

--------------------------------------------------------------------------------

'''

**def** **Inverter**(n):

"""Função que calcula o inverso de um número."""

resultado = 1.0 / n *#retorno normal da função*

n = 5 *#a função altera o valor de* ***'n'****, mas isso NÃO afeta o valor fora da função*

**return** resultado

*#----------------------------------------------------------------------------------------*

**def** **main**():

*#Solicitando o valor de* ***x***

x = **int**(**input**("Digite um número: "))

**print**()

**print**(f"Valor de x na ida: **{x}**") *#valor de x antes de chamar a função*

*#Chama a função e passa o valor de* ***x***

y = **Inverter** (x)

**print**()

**print**(f"Valor de x na volta: **{x}**") *#Valor de x NÃO é alterado depois da função*

**print**()

*#Pausa para exibir o resultado*

**print**("Pressione Enter para continuar...")

**input**()

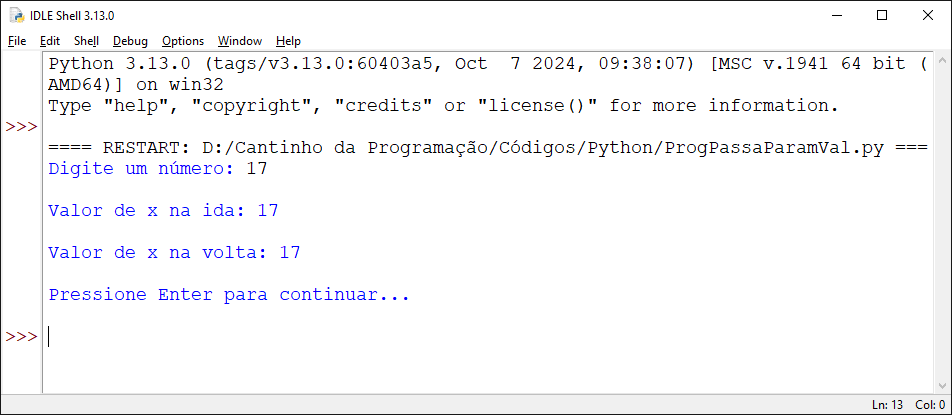
*#========================================================================================*

*#Rotina principal (chamadora)*

**if**(\_\_name\_\_ == "\_\_main\_\_"):

**main**()

*#Fim do programa "****ProgPassaParamVal****" ----------------------------------------------------*



**Figura 3 - Saída do programa em Python**