**Passa Parâmetro por Referência**

***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). Embora algumas linguagens ofereçam somente um tipo de passagem de parâmetros (***Java*** apenas “Por Valor” e ***C*** apenas “Por Referência”), a maioria das linguagens de programação oferece dois tipos desse mecanismo: “Passagem Por Valor” e “Passagem Por Referência”. Nesta postagem valor nos ater ao segundo 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 um cheque no valor 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 um cheque nesse 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)

Assim, o chefe manda o *office-boy* pagar a duplicata no valor de R$ 200,00; porém com um cheque em branco que deverá ser preenchido no momento do pagamento; lá no banco. Suponha ainda, hipoteticamente, que o funcionário preencha o cheque com um valor diferente do pedido pelo chefe: por exemplo, R$ 250,00; o que aconteceria quando o chefe consultasse a contabilidade da empresa? É claro que seria detectado um “rombo” de R$ 50,00, pois o cheque seria compensado em R$ 250,00 e não R$ 200,00 como foi mandado (*passado*) pelo chefe ao subordinado: o subordinado (*sub-rotina*) teria alterado o valor (*parâmetro*)! Em outras palavras, o que foi passado ao funcionário não foi um valor e sim um cheque que representa o ENDEREÇO (*referência*) da conta bancária da empresa!

O programa **“ProgPassaParamRef”**, codificado em Pascal, C# e Python, apresenta um exemplo de implementação para mostrar como funciona a passagem de parâmetros por referência, mostrando que o valor passado 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 alguns 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** **ProgPassaParamRef**;

**{ProgPassaParamRef.pas**

Mostra como funciona a passagem de parâmetros “Por Referência”. Se a rotina chamada alterar o parâmetro recebido, essa alteração será refletida na rotina chamadora.

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

**}**

**Uses** Crt;

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

y: **real**;

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

**Function** **Inverter**(**var** n:**integer**): **real**; *//sem o “****var”*** *seria “Por Valor”*

*//Retorna um valor passado “Por Referência”*

**Begin**

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

n := 250; *//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***

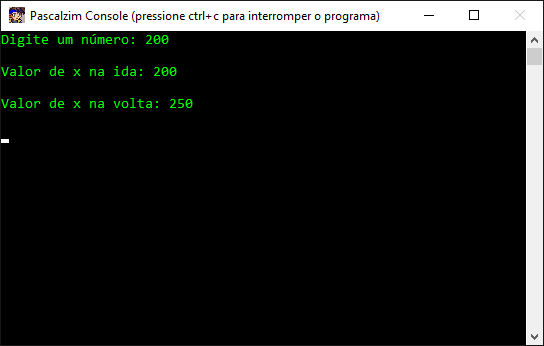
**WriteLn**('');

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

**WriteLn**('');

**ReadKey**;

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



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

/\***ProgPassaParamRef.cs**

Mostra como funciona a passagem de parâmetros “Por Referência”. Se a rotina chamada alterar o parâmetro recebido, essa alteração será refletida na rotina chamadora.

\*/

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

**using** System;

**namespace** **ProgPassaParamRef**

{

**class** **ProgPassaParamRef**

{

*//Função que recebe o parâmetro por referência*

**static** **double** **Inverter**(**ref** **int** n) *//sem o* ***ref*** *seria “Por Valor”*

{

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

n = 250; *// 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**(**ref** x); *//chama a função e passa o valor de x "Por Referência"*

**Console**.WriteLine();

**Console**.WriteLine("Valor de x na volta: " + x); *//altera valor para* ***250***

**Console**.WriteLine();

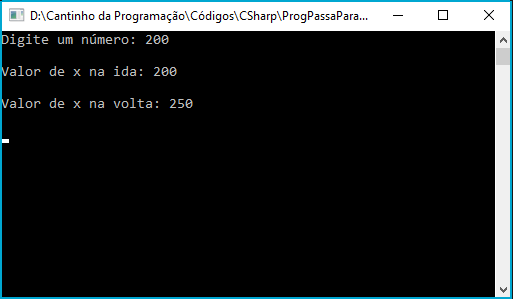
*//Pausa para ver o resultado*

**Console**.ReadKey();

}

}

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



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

'''

**ProgPassaParamRef.py**

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

Mostra como funciona a passagem de parâmetros "Por Referência" (usando lista).

Se a rotina chamada alterar o parâmetro recebido, esta alteração será refletida na rotina chamadora.

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

'''

**def** **Inverter**(LstPar):

*"""Função que calcula o inverso de um número e altera o valor de 'n'."""*

resultado = 1.0 / LstPar[0] *#usa* ***LstPar[0]*** *para acessar o valor*

LstPar[0] = 250 *#altera o valor dentro da lista, o que afetará a variável fora da função*

**return** resultado

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

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

*#Solicita o valor de x*

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

*#Coloca* ***x*** *dentro de uma lista para simular a passagem "Por Referência"*

LstRef = [x]

**print**()

**print**(f"Valor de **x** na ida: **{LstRef[0]}**") *#valor de* ***x*** *antes de chamar a função*

*#Chama a função e passa a lista que contém* ***x***

y = **Inverter**(LstRef)

**print**()

**print**(f"Valor de x na volta: **{LstRef[0]}**") *#valor de* ***x*** *será alterado depois da função*

**print**()

*#Pausa para exibir o resultado*

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

**input**()

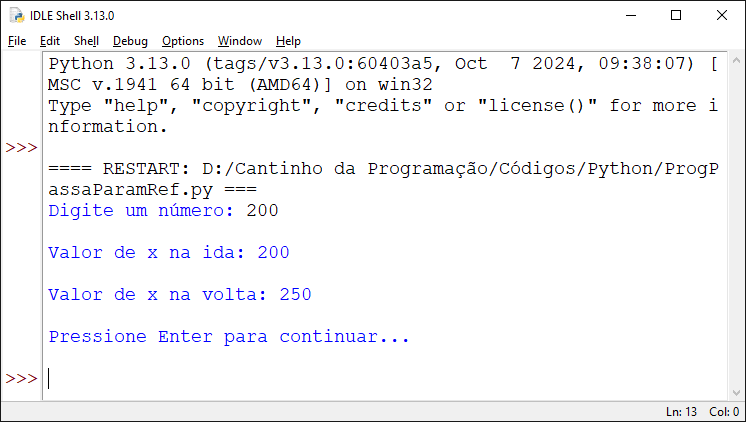
*#==========================================================================================*

*#Programa principal (roina chamadora)*

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

**main**()

**#Fim do programa "ProgPassaParamRef" ---------------------------------------------**



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