

# [A2E1] - Modularização

Total de pontos 0/0 ?

Funções

0 de 0 pontos

## Objetivos

- Entender o conceito e as propriedades de uma função
- criar suas próprias funções
- resolver o exercício proposto

Nome Completo \*

Harllem Alves do Nascimento

Funções - Procedimento

0 de 0 pontos

Sintaxe:

```
<tipo do retorno> nome (<parâmetros>){  
<instruções>  
return //quando possui retorno  
}
```



Tendo como base o trecho de código acima, crie um programa que realize a troca de valores entre variáveis através de uma função, utilizando a chamada por referência. ou seja, a função não oferecerá retorno e será realizada a troca dos valores destas variáveis na própria função. Escopo da função: void swap(int \* x, int \* y); Obs.: Não utilizar os \* nas variáveis locais. \*

```
#include <stdio.h>
#include <locale.h>
```

```
void swapVal(int *x, int *y){
    int temp;
    temp = *x;
    *x = *y;
    *y = temp;
}
```

```
int main(void){
    system("cls");
    setlocale(LC_ALL, "Portuguese");

    //Define variaveis
    int x, y;

    //Solicita valores
    printf("Insira os valores para realizar a troca:\n\n");
    printf("Digite o Valor 01: ");
    scanf("%d", &x);
    printf("Digite o Valor 02: ");
    scanf("%d", &y);

    //Faz a troca e mostra o resultado
    swapVal(&x, &y);
    printf("\n\n**Verifique a troca dos valores:\n\nValor 01: %d\nValor 02: %d", x, y);

    printf("\n\n");
    return 0;
}
```

Este formulário foi criado fora de seu domínio. - [Termos de Serviço](#) - [Política de Privacidade](#)

Google Formulários



função com retorno (return) e procedimento (void - sem retorno)

## [A2V1] - Modularização



### Endereço de uma variável

Lembram que no C, relacionado a essa linha:

```
scanf("%d", &var); //leitura de uma valor para a variável var
```

quando utilizamos o & estamos referenciando a variável pelo seu endereço?

Pois bem, vamos entender agora que ao passarmos os argumentos em uma função podemos passar o seu valor ou o seu endereço. É o que chamamos de:

- chamada por valor: é a passagem normal do valor dos argumentos para a função. Esses valores não são modificados, pois é fornecida uma cópia dos valores para a função.

```
nomeFunc(var);
```

- chamada por referência: desta forma são passados os endereços de memória dos argumentos, logo, os valores podem ser modificados.

```
nomeFunc(&var);
```



Chamada por valor e chamada por referência.

### [A2V2] - Chamada por valor e chamada por referência



Exercício

0 de 0 pontos

No algoritmo swap é realizada a troca de valores entre variáveis, como nesse trecho:

```
int a, b, temp;  
a=4;  
b=5;  
//trocando os valores  
temp=a;  
a=b;  
b=temp
```

