

# CP63B-DPGR3A

## COMPUTAÇÃO 2

APNP 06 - Passagem de Parâmetros  
(Valor e Referência)

Prof. Rafael Gomes Mantovani

# Licença

Este trabalho está licenciado com uma Licença CC BY-NC-ND 4.0:



maiores informações:

[https://creativecommons.org/licenses/by-nc-nd/4.0/deed.pt\\_BR](https://creativecommons.org/licenses/by-nc-nd/4.0/deed.pt_BR)

# Roteiro



- 1** Introdução
- 2** Passagem por valor
- 3** Passagem por referência
- 4** Vetores como parâmetros
- 5** Matrizes como parâmetros
- 6** Exercícios
- 7** Referências

# Roteiro

- 1** Introdução
- 2** Passagem por valor
- 3** Passagem por referência
- 4** Vetores como parâmetros
- 5** Matrizes como parâmetros
- 6** Exercícios
- 7** Referências

# Introdução

- Chama-se passagem de parâmetros a ação de informar os valores a serem processados por uma função;
- A Linguagem C/C++ define duas categorias de passagem de parâmetros
  - passagem por valor;
  - passagem por endereço (ou passagem por referência);
- Normalmente, a passagem de parâmetros a uma função é por valor;
- Mas, como os parâmetros de uma função são variáveis locais, alguns aspectos devem ser observados;

# Roteiro

- 1 Introdução
- 2 Passagem por valor
- 3 Passagem por referência
- 4 Vetores como parâmetros
- 5 Matrizes como parâmetros
- 6 Exercícios
- 7 Referências

# Passagem por Valor

- Exemplo de passagem por valor:

```
void alterar(int x, int y, int z)
{
    printf("valores recebidos... %d %d %d", x, y, z);
    x++; //incrementa valor de x
    y++; //incrementa valor de y
    z++; //incrementa valor de z
    printf("valores alterados... %d %d %d", x, y, z);
}

void main()
{
    int a = 1, b = 2, c = 3;
    alterar(a, b, c);
    printf("valores finais... %d %d %d", a, b, c);
}
```

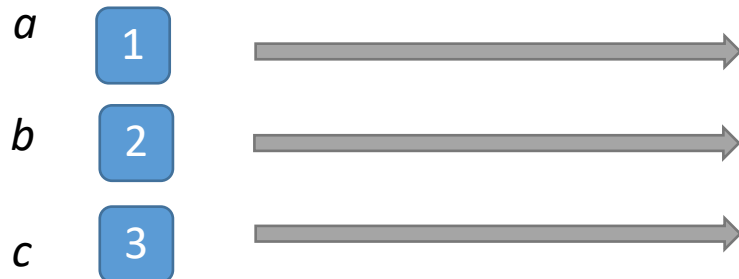
O que será exibido pelo programa?

Valores recebidos... 1 2 3  
Valores alterados... 2 3 4  
Valores finais... 1 2 3

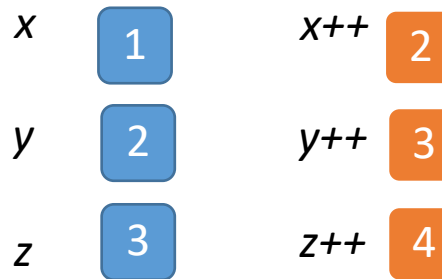
# Passagem por Valor

- Os valores das variáveis *a*, *b* e *c* não foram modificados após a execução da função *alterar*
- O tipo de passagem de parâmetros utilizado é por valor
  - são feitas apenas cópias dos valores das variáveis *a*, *b*, e *c* nas variáveis *x*, *y* e *z*.

Escopo: função *main*



Escopo: função *alterar*

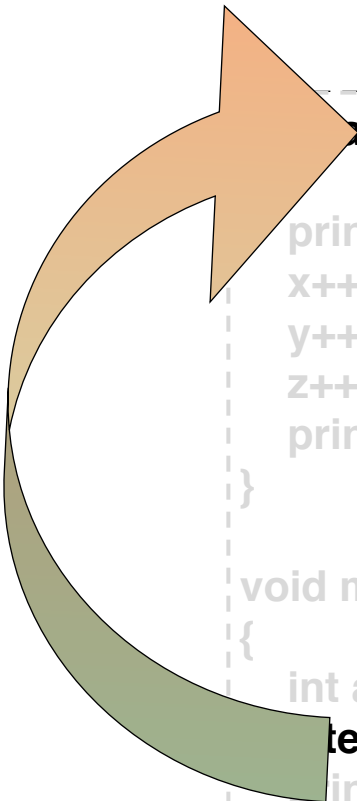


*Apenas os valores de *x*, *y* e *z* são alterados*



# Passagem por Valor

- Entendendo melhor a passagem de parâmetros por valor



```
alterar(int x, int y, int z)

printf("valores recebidos... %d %d %d", x, y, z);
x++; //incrementa valor de x
y++; //incrementa valor de y
z++; //incrementa valor de z
printf("valores alterados... %d %d %d", x, y, z);
}

void main()
{
    int a = 1, b = 2, c = 3;
    alterar(a, b, c);
    printf("valores finais... %d %d %d", a, b, c);
}
```

1	x
2	y
3	z
...	
1	a
2	b
3	c

# Roteiro

- 1 Introdução
- 2 Passagem por valor
- 3 Passagem por referência
- 4 Vetores como parâmetros
- 5 Matrizes como parâmetros
- 6 Exercícios
- 7 Referências

# Passagem por Referência

- E se for desejável que a função modifique os valores das variáveis **a**, **b** e **c** passadas a ela como parâmetros?
- Neste caso, em vez de passar para a função os valores destas variáveis, é preciso passar os seus **endereços** na memória;
- Dessa maneira, a função pode gravar diretamente sua saída na variável.

# Passagem por Referência

- Considere, por exemplo, que as variáveis **a**, **b** e **c** correspondem, respectivamente, aos endereços (hexadecimais) *F010*, *F020* e *F030*.

Endereço	Conteúdo	Variável	
	⋮		
F010	1	a	&a = F010 (endereço de a)
F020	2	b	&b = F020 (endereço de b)
F030	3	c	&c = F030 (endereço de c)
	⋮		

# Passagem por Referência

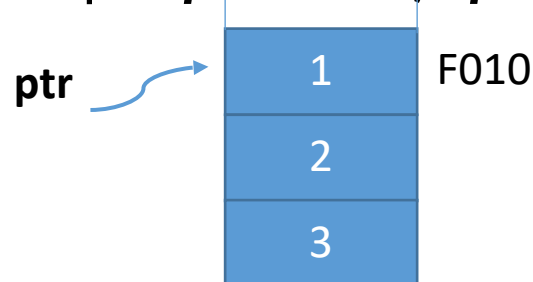
- Considerando uma variável declarada como:

```
int* ptr;
```

- ptr*** é um ponteiro para **int**, isto é, uma variável que armazena o endereço de uma variável do tipo **int**.
- Supondo que ***ptr*** armazene o valor F010, tem-se que:

```
ptr = &a;
```

- Define-se **\**ptr*** como sendo o valor contido na posição de memória apontada por ***ptr***. Assim, **\**ptr*** vale 1.



# Passagem por Referência

- Exemplo de passagem por referência:

```
void alterar(int* x, int* y, int* z)
{
    printf("valores recebidos... %d %d %d", *x, *y, *z);
    (*x)++; //incrementa valor de x
    (*y)++; //incrementa valor de y
    (*z)++; //incrementa valor de z
    printf("valores alterados... %d %d %d", *x, *y, *z);
}

void main()
{
    int a = 1, b = 2, c = 3;
    alterar(&a, &b, &c);
    printf("valores finais... %d %d %d", a, b, c);
}
```

O que será exibido pelo programa?

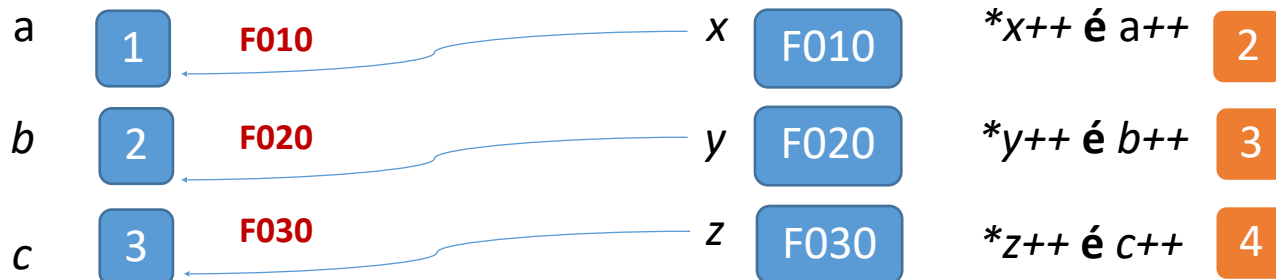
Valores recebidos... 1 2 3  
Valores alterados... 2 3 4  
Valores finais... 2 3 4

# Passagem por Referência

- Agora os valores das variáveis **a**, **b** e **c** foram modificados na função *alterar*;
- Foram passados os endereços das variáveis a, b, e c para os ponteiros x, y e z.

Escopo: função *main*

Escopo: função *alterar*



Os valores de **a**,  
**b** e **c** são  
*realmente*  
*modificados*

# Passagem por Referência

```
void troca(int a, int b){  
    int aux;  
    aux = a;  
    a = b;  
    b = aux;  
}  
  
void main(){  
    int x = 2, y = 5;  
    troca(x, y);  
  
    printf("%d %d", x, y);  
}
```

```
void troca(int* a, int* b){  
    int aux;  
    aux = *a;  
    *a = *b;  
    *b = aux;  
}  
  
void main(){  
    int x = 2, y = 5;  
    troca(&x, &y);  
  
    printf("%d %d", x, y);  
}
```

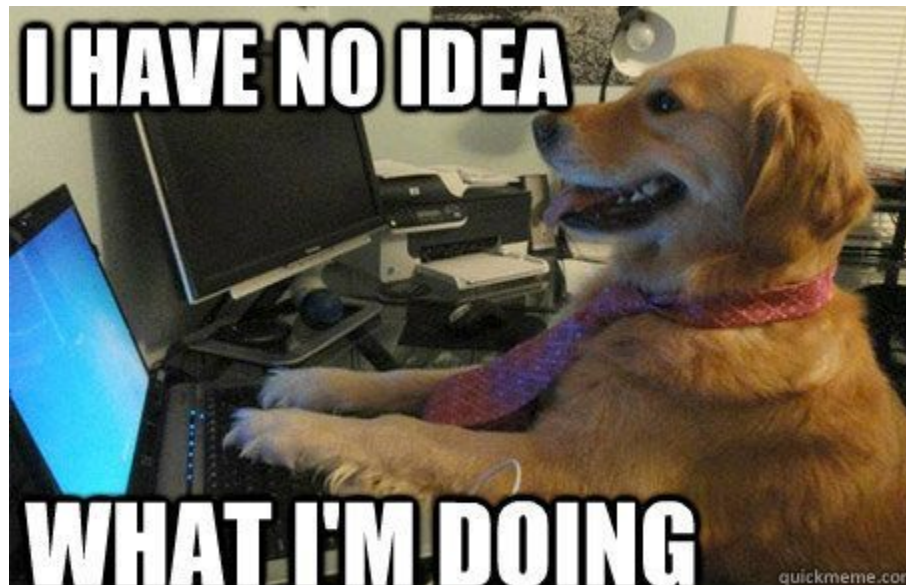


# Passagem por Referência

- Isso pode parecer apenas uma maneira de complicar as coisas, mas na realidade tem diversas utilidades
  - Transmitir uma grande quantidade de dados a outra parte do programa.
    - Podemos passar apenas um ponteiro para esses dados em vez de fazer uma cópia dos dados e transmitir a cópia;
    - Economiza tempo, processamento e memória;
  - Uma função em C só pode devolver um valor com a instrução **return**
    - Se uma função recebe como parâmetros ponteiros para outras variáveis, você poderá gravar valores nessas variáveis;
    - Com isso uma função pode gerar vários valores de “saída”.

# Erros comuns

- Esquecer o ‘&’ na passagem de parâmetros por referência;
- Esquecer o ‘\*’ no uso de um parâmetro passado por referência;
- Confundir o escopo de variáveis com mesmo nome.



# Roteiro

- 1 Introdução
- 2 Passagem por valor
- 3 Passagem por referência
- 4 Vetores como parâmetros
- 5 Matrizes como parâmetros
- 6 Exercícios
- 7 Referências

# Vetores como Parâmetros de Função

- Vetores e matrizes são sempre passados por **REFERÊNCIA** para funções;
- Os vetores e matrizes podem ser bem grandes, e portanto não seria muito prático copiar todos os valores para a função;
- Quando você passa um vetor para uma função, ela na verdade recebe apenas o endereço dos dados;
- Com isso, qualquer modificação num vetor ou matriz passado como parâmetro para uma função é refletida na matriz/vetor original.

# Vetores como Parâmetros de Função

- Sintaxe da função com vetor como parâmetros
  - basta imitar a declaração de vetores, exceto por um detalhe: não precisamos fornecer o tamanho do vetor — os colchetes podem ser deixados vazios

```
void imprime_primeiro(int v[])  
{  
    printf("%d\n", v[0]);  
}
```

- Para mandar um vetor como parâmetro de uma função, você simplesmente deve escrever o nome dele, sem colchetes ou qualquer outro símbolo

```
int vetor[] = {1, 2, 3, 4};  
imprime_primeiro(vetor);
```

*O nome do vetor nada mais é do que um ponteiro para a sua primeira posição*

# Roteiro

- 1 Introdução
- 2 Passagem por valor
- 3 Passagem por referência
- 4 Vetores como parâmetros
- 5 Matrizes como parâmetros
- 6 Exercícios
- 7 Referências

# Matrizes como Parâmetros de Função

```
void alteraMatriz(int mat[3][3])
```

```
{  
    mat[0][0] = 0;  
    mat[1][1] = 0;  
    mat[2][2] = 0;  
}
```

```
int main()
```

```
{  
    int i, j;  
    int A[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
```



```
    alteraMatriz(A);
```

```
    for(i=0;i<3;i++)  
        for(j=0;j<3;j++)  
            printf("%d", A[i][j]);
```

```
    return 0;
```

```
}
```

Deve ser fornecida pelo menos a quantidade de **colunas** da matriz


$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 2 & 3 \\ 4 & 0 & 6 \\ 7 & 8 & 0 \end{bmatrix}$$

# Matrizes como Parâmetros de Função

- O que devemos fazer se desejarmos que os elementos de um vetor, passado como parâmetro para uma função, não sejam alterados?

R: Criamos um vetor B dentro da função e copiamos o conteúdo do vetor passado como parâmetro para B. Toda a manipulação realizada dentro da função deverá ser em relação ao novo vetor B.



# Roteiro

- 1 Introdução
- 2 Passagem por valor
- 3 Passagem por referência
- 4 Vetores como parâmetros
- 5 Matrizes como parâmetros
- 6 Exercícios
- 7 Referências

# Exercícios

Universidade Tecnológica Federal do Paraná – Câmpus Apucarana  
Computação 2 (CP63B) - DPGR3A – Passagem de Parâmetros  
Prof. Dr. Rafael Gomes Mantovani

Mais exercícios no Moodle :)

## Instruções:

- Antes de codificar, esboce em um papel a sequência de passos necessários para criar o seu programa. Isso ajuda a programar a solução;
- Crie um arquivo .c para cada um dos exercícios. Por exemplo, na resolução do exercício 01, crie um arquivo chamado 'ex01.c'.
- em todos os exercícios faça uma função `main` para testar sua função.

## Exercícios sobre Passagem de Parâmetros

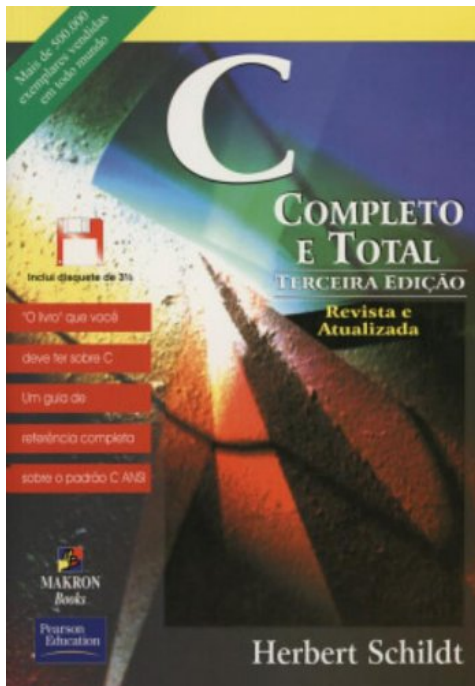
**Exercício 1.** Escreva um programa que receba um número inteiro representando a quantidade total de segundos e, usando passagem de parâmetros por referência, converta a quantidade informada de segundos em horas, minutos e segundos. Imprima o resultado da conversão no formato HH:MM:SS. Utilize o seguinte protótipo da função:

```
void converteHora(int total_segundos, int *hora, int *min, int *seg);
```

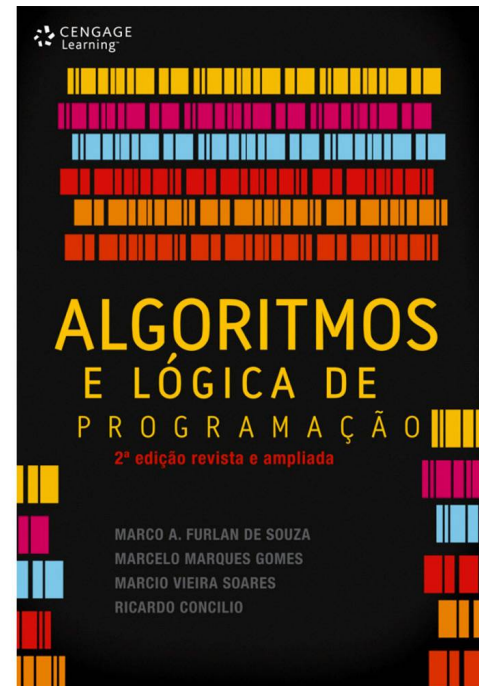
# Roteiro

- 1 Introdução
- 2 Passagem por valor
- 3 Passagem por referência
- 4 Vetores como parâmetros
- 5 Matrizes como parâmetros
- 6 Exercícios
- 7 Referências

# Referências



[Schildt, 1997]



[de Souza et al, 2011]

# Referências

- [Schildt, 1997] SCHILDT, H. **C Completo e Total**. 3. ed. São Paulo: Pearson, 1997.
- [de Souza et al, 2011] DE SOUZA, M. A. F. et al. **Algoritmos e lógica de programação**. 2. ed. São Paulo: Cengage Learning, 2011.

# Perguntas?

Prof. Rafael G. **Mantovani**

[rafaelmantovani@utfpr.edu.br](mailto:rafaelmantovani@utfpr.edu.br)