



# MCTA028-15: Programação Estruturada

## Aula 3: Funções - Tipos de Passagem de Parâmetros (Terceira Parte)

Wagner Tanaka Botelho

[wagner.tanaka@ufabc.edu.br](mailto:wagner.tanaka@ufabc.edu.br) / [wagtanaka@gmail.com](mailto:wagtanaka@gmail.com)

Universidade Federal do ABC (UFABC)

Centro de Matemática, Computação e Cognição (CMCC)

# Tipos de Passagem de Parâmetros

# Introdução

- Já vimos que os **parâmetros** de uma **função** são o mecanismo que o programador utiliza para **passar** a **informação** de um **trecho de código** para **dentro** da função;
- Existem **DOIS tipos** de passagem de parâmetro:
  - Por **valor**;
  - Por **referência**.

# Passagem Por Valor

```
1  #include <stdio.h>
2  void Incrementar_Um(int n){
3      n = n + 1;
4      printf("Dentro da funcao: x = %d\n", n);
5  }
6
7  int main(){
8      int x = 5;
9
10     printf("Antes da funcao: x = %d\n", x);
11
12     Incrementar_Um(x);
13
14     printf("Depois da funcao: x = %d\n", x);
15 }
```

 D:\UFABC\Disciplinas\2021-202

```
Antes da funcao: x = 5
Dentro da funcao: x = 6
Depois da funcao: x = 5
```

Na passagem de **PARÂMETROS** por **VALOR**, quaisquer modificações que a função fizer nos parâmetros existem **APENAS** dentro da própria função.

A variável **n** da função **Incrementar\_Um** é alterada **SOMENTE** dentro da função.

# Passagem Por Referência

# Passagem por Referência

- Existem casos em que é necessário que **TODA** modificação feita nos valores dos parâmetros **DENTRO** da função seja repassada para quem a chamou:
  - Um exemplo simples é a função `scanf()` já implementada na Linguagem C.
- Já estudamos que a função `scanf()` é utilizada para **ler** algo do **teclado**:
  - Para isso, o **nome da variável**, onde o dado será armazenado, é **passado**.

```
Ex_15.c x
1  #include <stdio.h>
2
3  void main() {
4      int x = 5;
5
6      printf("Antes do scanf: x = %d\n", x);
7
8      printf("Digite um numero:\n");
9      scanf("%d", &x);
10
11     printf("Depois do scanf: x = %d\n", x);
12 }
```

D:\UFABC\Disciplinas\2021-2025'

```
Antes do scanf: x = 5
Digite um numero:
8
Depois do scanf: x = 8
```

```
1  #include <stdio.h>
2
3  void main() {
4      int x = 5;
5
6      printf("Antes do scanf: x = %d\n", x);
7
8      printf("Digite um numero:\n");
9      scanf("%d", &x);
10
11     printf("Depois do scanf: x = %d\n", x);
12 }
```

O scanf() é uma OUTRA função já implementada na Linguagem C.

scanf.c

```
função scanf(...)
{
    ...
}
```

- + Na passagem de **parâmetros por referência**, o que é enviado para a função é o **endereço de memória** onde a variável está armazenada, e não uma simples **CÓPIA** de seu valor;
- + Assim, utilizando o **endereço da variável na memória**, qualquer **alteração** que a variável sofra **DENTRO** da **função** será também **refletida FORA** da função:
  - Por isso, variável **x** alterada **DENTRO** da função **scanf()** também é refletida **FORA** da função, ou seja, dentro da **main()**.

D:\UFABC\Disciplinas\2021-2025'

Antes do scanf: x = 5

Digite um numero:

8

Depois do scanf: x = 8

Dentro da main()

Número digitado pelo usuário e enviado para função scanf()

Como o x foi passado por **REFERÊNCIA** (&), o valor **ALTERADO** na função scanf() também é válido na main()



- + Para passar para a função um parâmetro por referência, a função precisa usar **PONTEIROS**;
- + Um **ponteiro** é um tipo especial de variável que armazena um **ENDEREÇO** de **MEMÓRIA**, da mesma maneira como uma variável armazena um valor.

### Passagem de Parâmetro Por Valor

```
*Ex_13.c x
1  #include <stdio.h>
2  void Incrementar_Um(int n){
3      n = n + 1;
4      printf("Dentro da funcao: x = %d\n", n);
5  }
6
7  void main(){
8      int x = 5;
9
10     printf("Antes da funcao: x = %d\n", x);
11
12     Incrementar_Um(x);
13
14     printf("Depois da funcao: x = %d\n", x);
15 }
```

 D:\UFABC\Disciplinas\2021-202

```
Antes da funcao: x = 5
Dentro da funcao: x = 6
Depois da funcao: x = 5
```

### Passagem de Parâmetro Por Referência

```
*Ex_16.c x
1  #include <stdio.h>
2  void Incrementar_Um(int *n){
3      *n = *n + 1;
4      printf("Dentro da funcao: x = %d\n", *n);
5  }
6
7  void main(){
8      int x = 5;
9
10     printf("Antes da funcao: x = %d\n", x);
11
12     Incrementar_Um(&x);
13
14     printf("Depois da funcao: x = %d\n", x);
15 }
```

**Passada por referência (\*)**

**Como o scanf(), na chamada da função, deve-se usar o &.**

 D:\UFABC\Disciplinas\2021-202

```
Antes da funcao: x = 5
Dentro da funcao: x = 6
Depois da funcao: x = 6
```

## Passagem de Parâmetro Por Referência

```
*Ex_16.c x
1  #include <stdio.h>
2  void IncrementarUm(int *n) {
3      *n = *n + 1;
4      printf("Dentro da funcao: x = %d\n", *n);
5  }
6
7  void main() {
8      int x = 5;
9
10     printf("Antes da funcao: x = %d\n", x);
11
12     IncrementarUm(&x);
13
14     printf("Depois da funcao: x = %d\n", x);
15 }
```

D:\UFABC\Disciplinas\2021-20

```
Antes da funcao: x = 5
Dentro da funcao: x = 6
Depois da funcao: x = 6
```

Para acessar o **CONTEÚDO** da posição de memória para a qual o **ponteiro aponta**, usa-se o operador *asterisco* (\*) na frente do nome do ponteiro (linha 3: `*n = *n + 1`).

## MEMÓRIA

10/11

Endereço	Variável	Conteúdo
.....	.....	
120	int x	<del>5</del> 6
.....	.....	
201	int *n	#120

\*n

Ponteiro \*n é inicializado com o endereço (&) de x.

O ponteiro n passa a conter o endereço de x, NÃO o seu valor.

Linha	x	*n
8	5	
10	{5}	
2		#120
3	6	6
4		{6}
14	{6}	

# Referências

- Slides do Prof. Luiz Rozante;
- SALES, André Barros de; AMVAME-NZE, Georges. Linguagem C: roteiro de experimentos para aulas práticas. 2016;
- BACKES, André. Linguagem C Completa e Descomplicada. Editora Campus. 2013;
- SCHILDT, Herbert. C Completo e Total. Makron Books. 1996;
- DAMAS, Luís. Linguagem C. LTC Editora. 1999;
- DEITEL, Paul e DEITEL, Harvey. C Como Programar. Pearson. 2011.