Exemplos – Métodos de Passagem de Parâmetros

- Função em C

```
void troca1 (int a, int b) {
int temp = a;
a = b;
b = temp;
{
```

- Chamada: troca1(c, d);

Como C usa **passagem por valor,** as ações dessa chamada são:

- a = c copia o valor de c em a
- b = d copia o valor de d em b
- Executa as instruções de troca1 e retorna

os valores de c e d permanecem os mesmos, pois nada foi passado de volta para a unidade chamadora.

Exemplos – Métodos de Passagem de Parâmetros

- Usando parâmetros do tipo ponteiro, o efeito é o de passagem por referência:

```
void troca2 (int *a, int *b) {
int temp = *a;
*a = *b;
*b = temp;
{
```

- Chamada: troca2(&c, &d);

As ações da chamada de troca 2 são como de **passagem por referência**:

- a = &c copia o endereço de c em a
- b = &d copia o endereço de d em b
- Executa as instruções de troca2 e retorna

os valores de c e d são de fato trocados, pois foi passado para troca2 o endereço desses parâmetros.

Exemplos – Métodos de Passagem de Parâmetros

- Função troca2 em C++:

```
void troca2 (int &a, int &b) {
int temp = a;
a = b;
b = temp;
{
```

- Chamada: troca2(c, d);

As ações da chamada de troca 2 são como de **passagem por referência**:

- a = &c copia o endereço de c em a
- b = &d copia o endereço de d em b
- Executa as instruções de troca2 e retorna.

os valores de c e d são de fato trocados, pois foi passado para troca2 o endereço desses parâmetros.

Exemplos – Métodos de Passagem de Parâmetros

- Passagem por valor-resultado: semântica idêntica a de passagem por referência, exceto quanto os parâmetros reais envolvem arrays ou expressões.

- Função troca3 em sintaxe similar a ADA:

```
procedure troca3 (a : in out Integer, b : in out Integer) is
   temp : Integer;
   begin
   temp := a;
   a := b;
   b := temp;
   end swap3;
```

- Chamada: troca3(c, d);

- Chamada: troca3(c, d);
- As ações da chamada de troca 3 assumindo **passagem por valor-resultado** são:
  - addr_c = &c salva o endereço do parâmetro real c
  - addr_d = &d salva o endereço do parâmetro real d
  - a = *addr_cc copia o valor de c em a
  - b = *addr_d copia o valor de d em b
  - (executa as instruções de troca3)
  - *addr_c = a copia o valor de a na posição ocupada por c
  - *addr_d = b copia o valor de a na posição ocupada por c
- Os valores de c e d são de fato trocados.

- Chamada: troca3(i, list[i]);
- As ações da chamada de troca 3 assumindo **passagem por valor-resultado** **agora** são:
  - addr_i = &i %salva o endereço do parâmetro real i
  - addr_list_i = &list[i] %salva o endereço do parâmetro real list[i]
  - a = *addr_i % copia o valor de i em a
  - b = *addr_list_i %copia o valor de list[i] em b
  - (executa as instruções de troca3)
  - *addr_i = a %copia o valor de a na posição ocupada por i
  - *addr_list_i = b %copia o valor de b na posição ocupada por list[i]
- Os valores de c e d são de fato trocados corretamente. Os endereços para onde os valores devem ser copiados no retorno são calculados no momento da chamada da função.

Passagem por referência:

- a = &i copia o endereço de i em a
- b = &list[i] copia o endereço de list[i] em b
- temp := a; atribui a temp valor apontado por a (valor de i)
- a := b; atribui a posição apontada por a (i) o valor apontado por b (list[i]); aqui o valor de i está sendo alterado
- b := temp; atribui a posição apontada por b (list[i]) o valor armazenado em temp; aqui o valor de list[i] está sendo alterado, mas b refere-se a posição relativa ao valor original de i
- retorna

O endereço de list[i] é calculado no momento da chamada e não muda após isso, mesmo que o valor de i seja alterado. Quando um novo valor é atribuído a b, a posição referida é a que corresponde ao valor antigo de i.

Passagem por valor resultado e referência quando ocorre “sinonímia”

Exemplo 1:
Programa em sintaxe semelhante ao C:

```
int i = 3;  /* i é uma variável global   */
void fun (int a, int b) {
        i = b;
}
void main( )    {
   int list [10];
   list [i] = 5;
  fun(i, list[i]);
}
```

- Passagem por referência: i e a são sinônimos

  - a = &i copia o endereço de i em a
  - b = &list[i] copia o endereço de list[i] em b
  - i = b
  - retorna

O valor da variável global i foi alterado para 5 , nenhuma modificação foi feita nos parâmetros.

- Passagem por valor-resultado: i e a não são sinônimos

  - addr_i = &i %salva o endereço do parâmetro real i
  - addr_list_i = &list[i] %salva o endereço do parâmetro real list[i]
  - a = *addr_i % copia o valor de i em a
  - b = *addr_list_i %copia o valor de list[i] em b
  - i = b % atribui 5 a variável global i
  - *addr_i = a %copia o valor de a na posição ocupada por i
  - *addr_list_i = b %copia o valor de b na posição ocupada por list[i]

O valor da variável global i foi alterado para 5 na função, mas a cópia do valor do parâmetro antes de retornar faz com que o valor antigo de i seja restabelecido e retorne para 3.