



ALGORITMOS E ESTRUTURAS DE DADOS

Passagem de Parâmetros

PASSAGEM DE PARÂMETROS

- Na linguagem C, os parâmetros de uma função são sempre passados por **valor**, ou seja, uma cópia do valor do parâmetro é feita e passada para a função.
- Mesmo que esse valor mude dentro da função, nada acontece com o valor de fora da função.



PASSAGEM POR VALOR

```
int n = x;

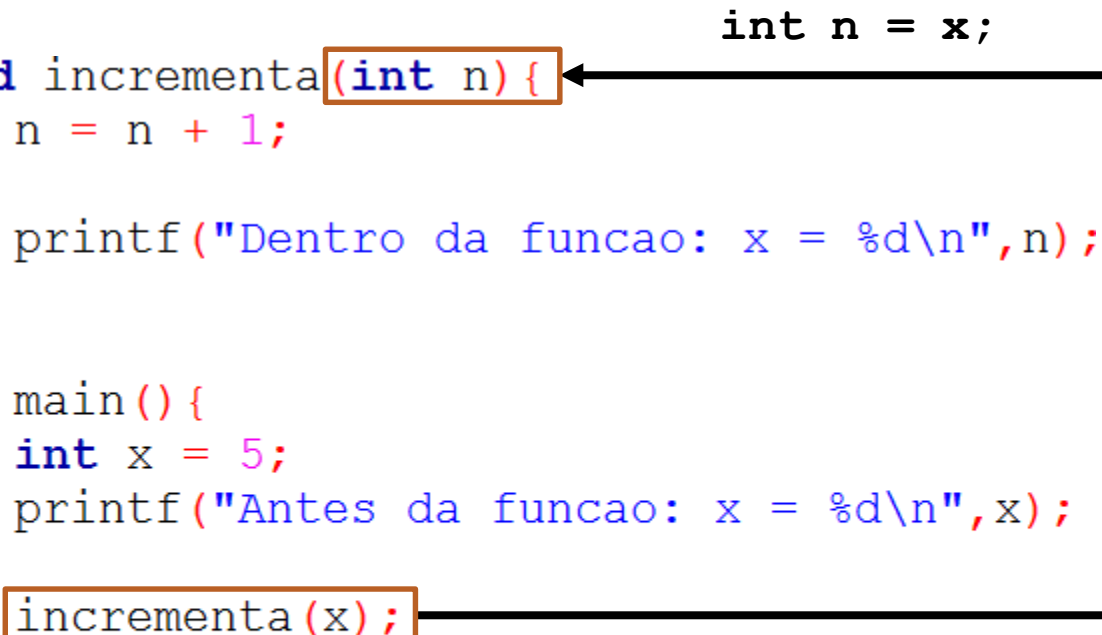
void incrementa(int n) {
    n = n + 1;

    printf("Dentro da funcao: x = %d\n", n);
}

int main() {
    int x = 5;
    printf("Antes da funcao: x = %d\n", x);

    incrementa(x);

    printf("Depois da funcao: x = %d\n", x);
    return 0;
}
```



Saída:

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



PASSAGEM POR REFERÊNCIA

- Quando se quer que o valor da variável mude dentro da função, usa-se passagem de parâmetros por ***referência***.
- Neste tipo de chamada, não se passa para a função o valor da variável, mas a sua ***referência*** (seu endereço na memória);



PASSAGEM POR REFERÊNCIA

- Utilizando o endereço da variável, qualquer alteração que a variável sofra dentro da função será refletida fora da função.
- Ex: função **scanf()**



PASSAGEM POR REFERÊNCIA

○ Ex: função **scanf()**

- Sempre que desejamos ler algo do teclado, passamos para a função **scanf()** o nome da variável onde o dado será armazenado.
- Essa variável tem seu valor modificado dentro da função **scanf()**, e seu valor pode ser acessado no programa principal

```
int main() {  
    int x = 5;  
    printf("Antes do scanf: x = %d\n", x);  
    printf("Digite um numero: ");  
    scanf("%d", &x);  
    printf("Depois do scanf: x = %d\n", x);  
  
    return 0;  
}
```



PASSAGEM POR REFERÊNCIA

- Para passar um parâmetro por referência, coloca-se um asterisco “*” na frente do nome do parâmetro na declaração da função:

```
//passagem de parâmetro por valor  
void incrementa(int n);
```

```
//passagem de parâmetro por referência  
void incrementa(int *n);
```

- Ao se chamar a função, é necessário agora utilizar o operador “&”, igual como é feito com a função **scanf()**:

```
//passagem de parâmetro por valor  
int x = 10;  
incrementa(x);
```

```
//passagem de parâmetro por referência  
int x = 10;  
incrementa(&x);
```



PASSAGEM POR REFERÊNCIA

- No corpo da função, é necessário usar um asterisco “*” sempre que se desejar acessar o conteúdo do parâmetro passado por referência.

```
//passagem de parâmetro por valor
void incrementa(int n) {
    n = n + 1;
}
//passagem de parâmetro por referência
void incrementa(int *n) {
    *n = *n + 1;
}
```



PASSAGEM POR REFERÊNCIA

```
int *n = &x;

void incrementa(int *n) {
    *n = *n + 1;

    printf("Dentro da funcao: x = %d\n", n);
}

int main() {
    int x = 5;
    printf("Antes da funcao: x = %d\n", x);

    incrementa(&x);

    printf("Depois da funcao: x = %d\n", x);
    return 0;
}
```

Saída:

Antes da funcao: x = 5

Dentro da funcao: x = 6

Depois da funcao: x = 6



EXERCÍCIO

- Crie uma função que troque o valor de dois números inteiros passados por referência.



ARRAYS COMO PARÂMETROS

- Para utilizar arrays como parâmetros de funções alguns cuidados simples são necessários.
- Arrays são sempre passados por referência para uma função;
 - A passagem de arrays ***por referência*** evita a cópia desnecessária de grandes quantidades de dados para outras áreas de memória durante a chamada da função, o que afetaria o desempenho do programa.



ARRAYS COMO PARÂMETROS

- É necessário declarar um segundo parâmetro (em geral uma variável inteira) para passar para a função o tamanho do array separadamente.
 - Quando passamos um array por parâmetro, independente do seu tipo, o que é de fato passado é o endereço do primeiro elemento do array.



ARRAYS COMO PARÂMETROS

- Na passagem de um array como parâmetro de uma função podemos declarar a função de diferentes maneiras, todas equivalentes:

```
void imprime(int *m, int n);  
void imprime(int m[], int n);  
void imprime(int m[5], int n);
```



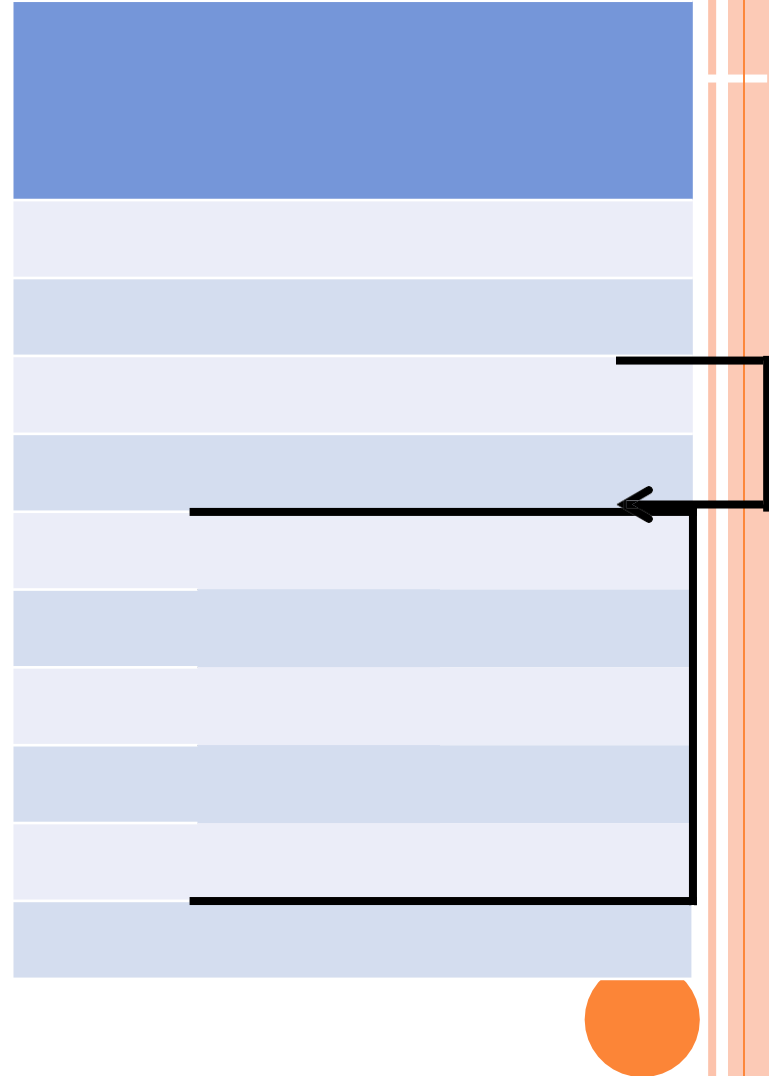
ARRAYS COMO PARÂMETROS

- Exemplo:
 - Função que imprime um array

```
void imprime(int *m, int n){
    int i;
    for (i=0; i< n;i++)
        printf ("%d \n", m[i]);
}

int main (){
    int vet[5] = {1,2,3,4,5};
    imprime(vet,5);

    return 0;
}
```



ARRAYS COMO PARÂMETROS

- Vimos que para arrays, não é necessário especificar o número de elementos para a função.

```
void imprime (int*m, int n);  
void imprime (int m[], int n);
```

- No entanto, para arrays com mais de uma dimensão, é necessário especificar o tamanho de todas as dimensões, exceto a primeira

```
void imprime (int m[][5], int n);
```



ARRAYS COMO PARÂMETROS

- Na passagem de um array para uma função, o compilador precisa saber o tamanho de cada elemento, não o número de elementos.
- Uma matriz pode ser interpretada como um array de arrays.
 - **int m[4][5]**: array de 4 elementos onde cada elemento é um array de 5 posições inteiras.



ARRAYS COMO PARÂMETROS

- Logo, o compilador precisa saber o tamanho de cada elemento do array.

```
int m[4][5]
```

```
void imprime (int m[][5], int n);
```

- Na notação acima, informamos ao compilador que estamos passando um array, onde cada elemento dele é outro array de 5 posições inteiras.



ARRAYS COMO PARÂMETROS

- Isso é necessário para que o programa saiba que o array possui mais de uma dimensão e mantenha a notação de um conjunto de colchetes por dimensão.
- As notações abaixo funcionam para arrays com mais de uma dimensão. Mas o array é tratado como se tivesse apenas uma dimensão dentro da função

```
void imprime (int*m, int n);  
void imprime (int m[], int n);
```

