



**INSTITUTO FEDERAL**  
Sul-rio-grandense

Câmpus  
Pelotas

EDUCAÇÃO  
**PÚBLICA**  
**100%**  
GRATUITA

# Estrutura de Dados

Aula 2

## Funções

Passagem de Parâmetros por Referência  
**Ponteiros**



# O Problema

Como uma função pode retornar mais de 1 valor ?

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

```
void troca(int x,int y);
```

```
int main()
```

```
{
```

```
    int a, b;
```

```
    a = 3;
```

```
    b = 5;
```

```
    troca( a, b);
```

```
    printf("%d %d\n", a, b);
```

```
    return 0;
```

```
}
```

```
void troca(int x, int y)
```

```
{
```

```
    int aux;
```

```
    aux = x;
```

```
    x = y;
```

```
    y = aux;
```

```
}
```

Trocas somente  
em variáveis  
locais

## Passagem de parâmetros

?

> 3 5



EDUCAÇÃO  
PÚBLICA  
100%  
GRATUITA

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

```
void troca(int x,int y);
```

```
int main()
```

```
{
```

```
    int a, b;
```

```
    a = 3;
```

```
    b = 5;
```

```
    troca(a, b);
```

```
    printf("%d %d\n", a, b);
```

```
    return 0;
```

```
}
```

```
void troca(int a, int b)
```

```
{
```

```
    int aux;
```

```
    aux = a;
```

```
    a = b;
```

```
    b = aux;
```

```
}
```

Trocas somente  
em variáveis  
locais

## Passagem de parâmetros

?

> 3 5

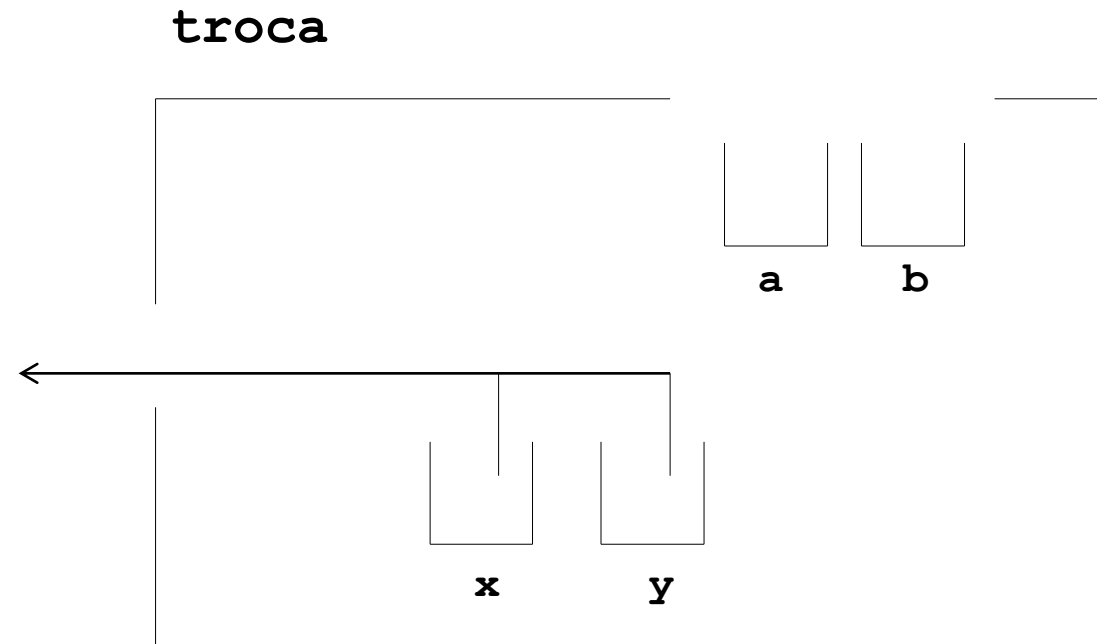


EDUCAÇÃO  
PÚBLICA  
100%  
GRATUITA

# Passagem de parâmetros

Como retornar 2 valores ???

Dessa forma **não** é possível...



O comando **return** só permite retornar **1** valor.



# PONTEIROS em C

O que são e para que servem ?

# Ponteiros

Um ponteiro é a representação de um **endereço de memória**.

**Ponteiro constante:** é um endereço.

**Ponteiro variável:** é uma variável que armazena endereços.

Conteúdo da  
variável **n**

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

```
int main()
```

```
{
```

```
    int n;
```

```
    n=3;
```

```
    ...
```

```
}
```

Endereço  
da variável **n**

1000

1001

1002

1003

1004

1005

1006

Memória

3

**n**

# Ponteiros

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

```
int main()
```

```
{
```

```
    int n;
```

Formato para inteiro  
sem sinal

```
    n=3;
```

```
    printf("%u\n", &n);
```

```
    return 0;
```

```
}
```

operador **&** : obtém o **endereço** de memória do seu operando.

Endereço  
da variável **n**

> 1000

Conteúdo da  
variável **n**

Memória

1000

1001

1002

1003

1004

1005

1006

3

**n**



# Ponteiros

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

```
int main()  
{
```

```
    int n, *p;
```

Declaração de um  
ponteiro para inteiro

```
    n = 3;
```

```
    p = &n;
```

```
    printf("%u\n", p);
```

```
    return 0;
```

```
}
```

variável que armazena  
um endereço.

> 1000

Memória

1000

1001

1002

1003

3

n

...

1100

1101

1102

1103

1000

p

# Ponteiros

O operador **&** não pode ser aplicado a constantes e expressões.

Ex:

~~&3~~

~~&(x+1)~~

## Declaração de um ponteiro

**int \*p;**

Indica que p é um ponteiro

Indica o tipo da variável cujo endereço  
será armazenado no ponteiro

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

```
int main()
{
    int *px;
    float x;
```

```
    x = 2.5;
    px = &x;
    . . . . .
```

**Incorreto!!**  
**Por que??**

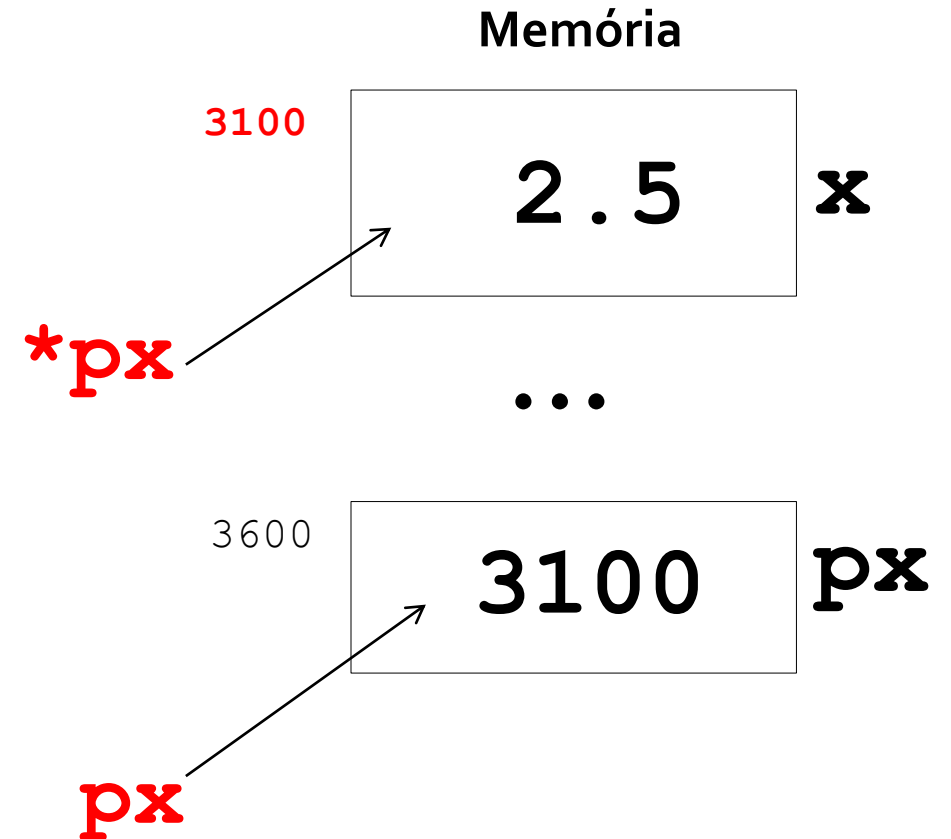
```
}
```

# Ponteiros

```
#include <stdio.h>

int main()
{
    float x, *px;

    x = 2.5;
    px = &x;
    printf("%f\n", *px);
    return 0;
}
```



O operador **\*** obtém o **conteúdo** da variável localizada no endereço armazenado no ponteiro.

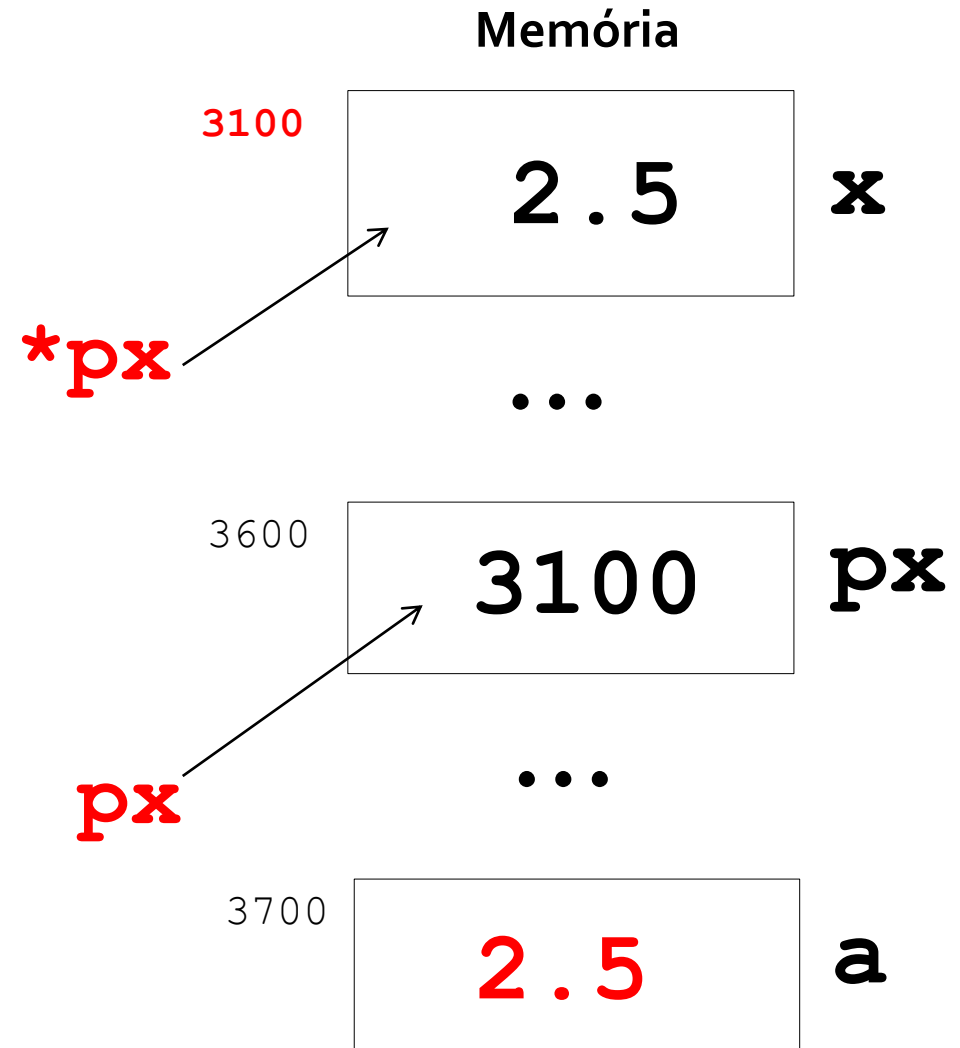
# Ponteiros

```
#include <stdio.h>

int main()
{
    float x, *px, a;

    x = 2.5;
    px = &x;
    a = *px;
    printf("%f\n", a);
    return 0;
}
```

Copia **x** para **a** de forma **indireta**.



# Ponteiros

É possível **alterar** o valor de uma variável de forma indireta.  
Basta que seu **endereço** esteja armazenado em um **ponteiro**.

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

```
int main()
```

```
{
```

```
    int n, *p;
```

```
    n = 3;
```

```
    p = &n;
```

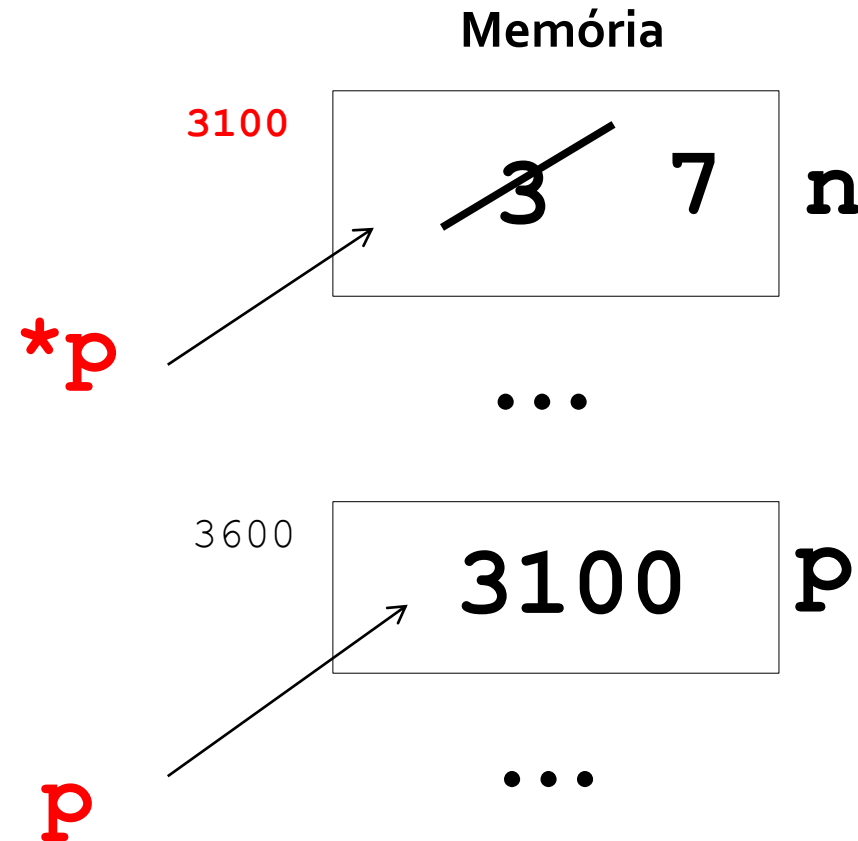
```
    *p = 7;
```

```
    printf("%d\n", n);
```

```
    return 0;
```

```
}
```

Altera **n**





# Passagem de Parâmetros por Referência

Aplicando Ponteiros em Funções

# Passagem de parâmetros por referência

## PROBLEMA:

Escreva uma função chamada **somaDez** que receba como entrada um inteiro e acrescente 10 na variável passada como argumento.

```
#include <stdio.h>

void somaDez(int x);

int main()
{
    int n;

    n = 2;
    somaDez(n);
    printf("%d\n", n);
    return 0;
}
```

Continuação...

```
void somaDez(int x)
{
    x = x + 10;
}
```

Não funciona!

Por quê???

# Passagem de parâmetros por referência

Como permitir que a função `somaDez` **altere** uma variável `n` que foi passada como parâmetro? Passando para a função o **endereço** de `n`.

```
#include <stdio.h>

void somaDez(int *x);

int main()
{
    int n;

    n = 2;
    somaDez(&n);
    printf("%d\n", n);
    return 0;
}
```

Continuação...

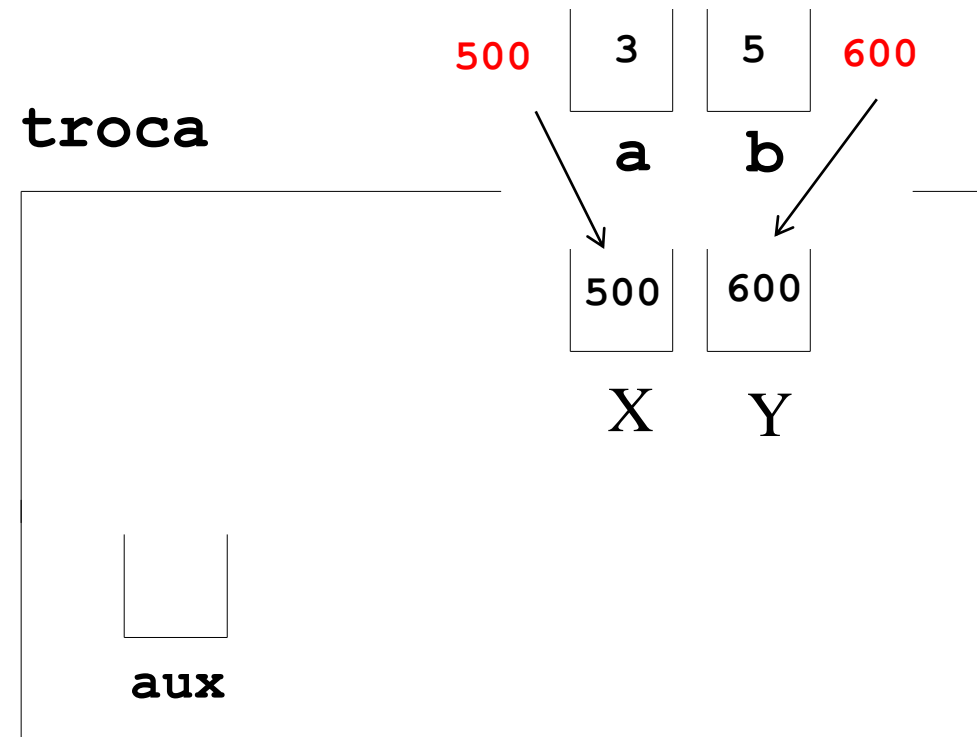
```
void somaDez(int *x)
{
    *x = *x + 10;
}
```



# Passagem de parâmetros por referência

Como implementar  
a função troca?

Passar os **endereços**  
de **a** e **b** para a função



Os ponteiros X e Y permitem a  
alteração de **a** e **b**.

# Passagem de parâmetros por referência

```
#include <stdio.h>

void troca( int *x, int *y );

int main()
{
    int a,b;

    a=3;
    b=5;
    troca( &a, &b );
    printf("%d %d\n", a, b);
    return 0;
}

void troca( int *x, int *y )
{
    int aux;

    aux = *x;
    *x = *y;
    *y = aux;
}
```

Como implementar  
a função troca?

# Passagem de parâmetros

## Passagem de parâmetros **por valor**:

**Uma cópia da variável** utilizada como argumento é passada para a função.  
**Não permite** que a função altere o valor do parâmetro real.

## Passagem de parâmetros **por referência**:

**O endereço da variável** utilizada como argumento é passada para a função.  
**Permite** que a função altere o valor do parâmetro real.

# Passagem de parâmetros

Podemos **classificar** os parâmetros em 3 categorias:

**De entrada:** Parâmetros que têm seus valores estabelecidos fora da função e não são modificados dentro dela. Normalmente utilizamos passagem de parâmetros **por valor**.

**De saída:** Parâmetros que têm seus valores estabelecidos dentro da função. Utilizamos passagem de parâmetros **por referência**.

**De entrada/saída:** Parâmetros que têm seus valores estabelecidos fora da função mas que são alterados dentro dela. Utilizamos passagem de parâmetros **por referência**.

# Três versões para a função soma:

## Versão 1

Dois parâmetros passados **por valor**.  
Resultado devolvido com **return**.

**Nome:** Soma

**Entrada:** 2 inteiros

**Saída:** Nenhuma

**Retorno:** valor da soma

```
#include <stdio.h>

int soma(int a,int b);

int main()
{
    int x,y,s;

    x = 10;
    y = 20;
    s = soma( x, y);
    printf("%d\n", s);
    return 0;
}

int soma(int a, int b)
{
    return a + b;
}
```

Diagram illustrating the function call and return:

Two arrows labeled 'E' point from the arguments 'x' and 'y' in the function call `soma(x, y)` to the parameters 'a' and 'b' in the function definition `int soma(int a, int b)`.

# Três versões para a função soma:

## Versão 2

Dois parâmetros passados **por valor**.  
Resultado obtido com parâmetro  
passado por referência

**Nome:** Soma

**Entrada:** 2 inteiros

**Saída:** valor da soma

**Retorno:** nenhum

```
#include <stdio.h>

void soma(int a,int b,int *r);

int main()
{
    int x,y,s;

    x=10;
    y=20;
    soma(x,y,&s);
    printf("%d\n",s);
    return 0;
}

void soma(int a, int b, int *r)
{
    *r = a + b;
}
```

Diagram illustrating the function call and return:

- The function `soma` is called with arguments `x`, `y`, and `&s` in `main`.
- The first two arguments, `x` and `y`, are passed by value (indicated by 'E' and a downward arrow).
- The third argument, `&s`, is passed by reference (indicated by 'S' and an upward arrow).

# Três versões para a função soma:

## Versão 3

Um parâmetros passados **por valor**.  
Resultado obtido com parâmetro  
passado por referência (modifica o  
segundo operando).

**Nome:** Soma

**Entrada:** 1 inteiro

**Entrada/Saída:** valor da soma

**Retorno:** nenhum

```
#include <stdio.h>

void soma(int a,int *b);

int main()
{
    int x,y;

    x=10;
    y=20;
    soma(x,&y);
    printf("%d\n",y);
    return 0;
}

void soma(int a, int *b)
{
    *b = a + *b;
}
```

Diagram illustrating the function call and return:

- E** (Entrada) points to the argument `x` in the function call.
- E/S** (Entrada/Saída) points to the pointer `*b` in the function signature, which is passed by reference.



**INSTITUTO FEDERAL**  
Sul-rio-grandense

Câmpus  
Pelotas

EDUCAÇÃO  
**PÚBLICA**  
**100%**  
GRATUITA

# Estrutura de Dados

Aula 2

## Funções

Passagem de Parâmetros por Referência  
**Ponteiros**