

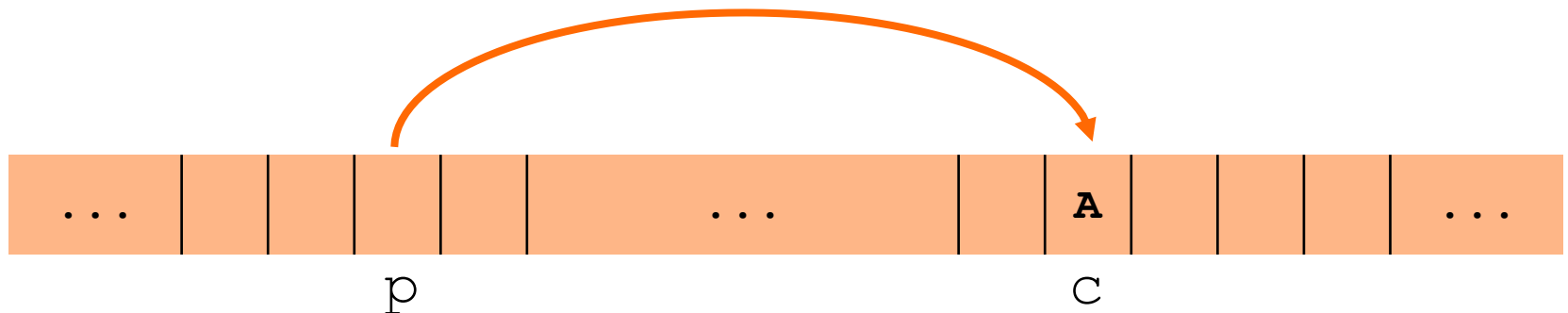


# PROGRAMAÇÃO ESTRUTURADA

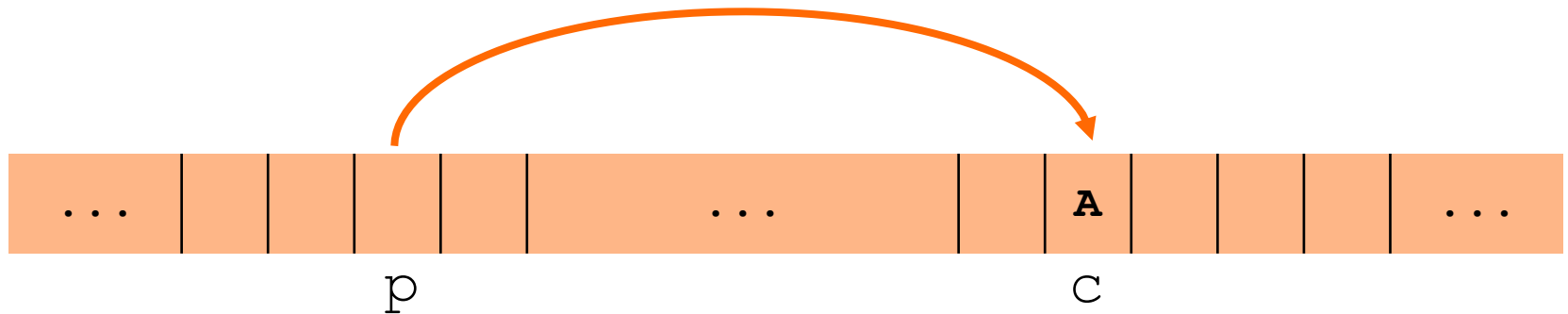
Ponteiros

# PONTEIROS / APONTADORES

- Um ponteiro é uma variável que é capaz de guardar um endereço de memória
- Os ponteiros são muito utilizados na linguagem C
- Exemplo:
  - Se  $c$  é um char e  $p$  um ponteiro que aponta para  $c$ , então  $p$  guarda o endereço da variável  $c$ .



# PONTEIROS / APONTADORES



- $p$  é uma variável ponteiro que guarda o endereço da variável  $c$
- $p = \&c$
- O operador unário  $\&$  fornece o endereço de uma variável
- O operador unário  $*$ , quando aplicado a um ponteiro, acessa o objeto que o ponteiro aponta

# PONTEIROS

## ○ Exemplo:

```
int x=1, y=2;
```

```
int *ap;
```

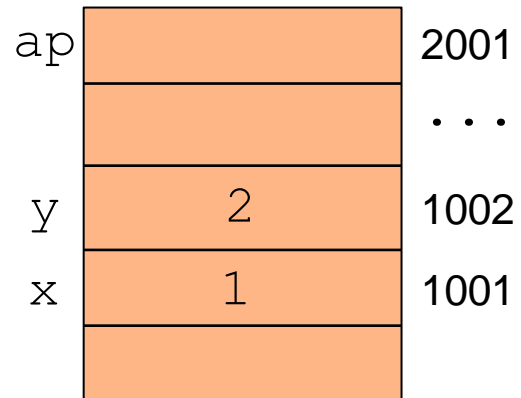


# PONTEIROS

## ○ Exemplo:

```
int x=1, y=2;
```

```
int *ap;
```



# PONTEIROS

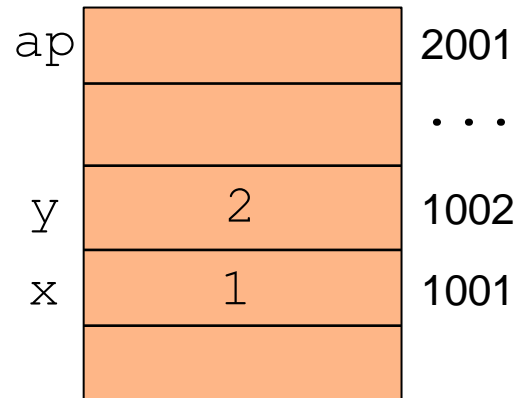
## ○ Exemplo:

```
int x=1, y=2;
```

```
int *ap;
```

```
...
```

```
ap = &x;
```



# PONTEIROS

## ○ Exemplo:

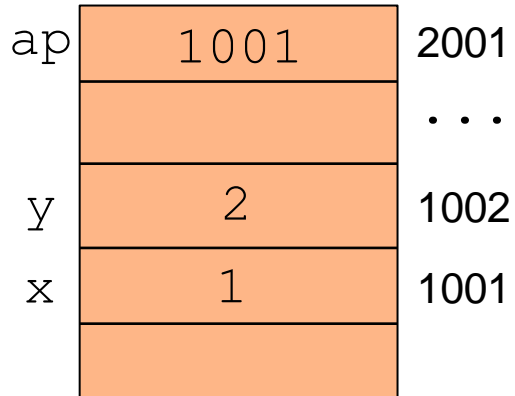
```
int x=1, y=2;
```

```
int *ap;
```

```
...
```

```
ap = &x;
```

ap recebe o  
endereço de x



# PONTEIROS

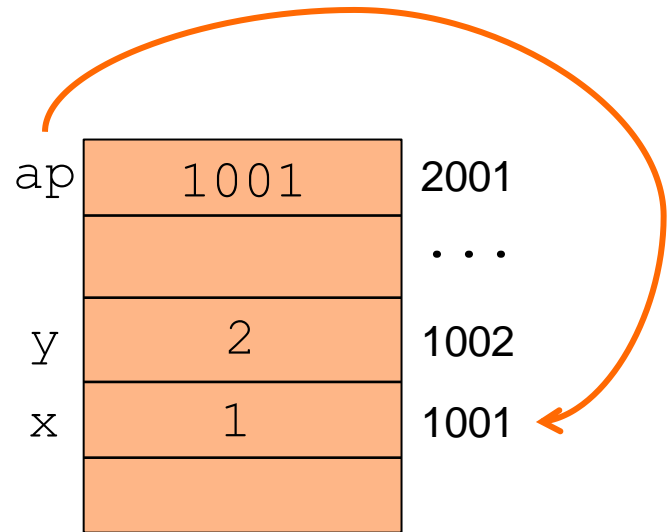
## ○ Exemplo:

```
int x=1, y=2;
```

```
int *ap;
```

```
...
```

```
ap = &x;
```





# PONTEIROS

## Exemplo:

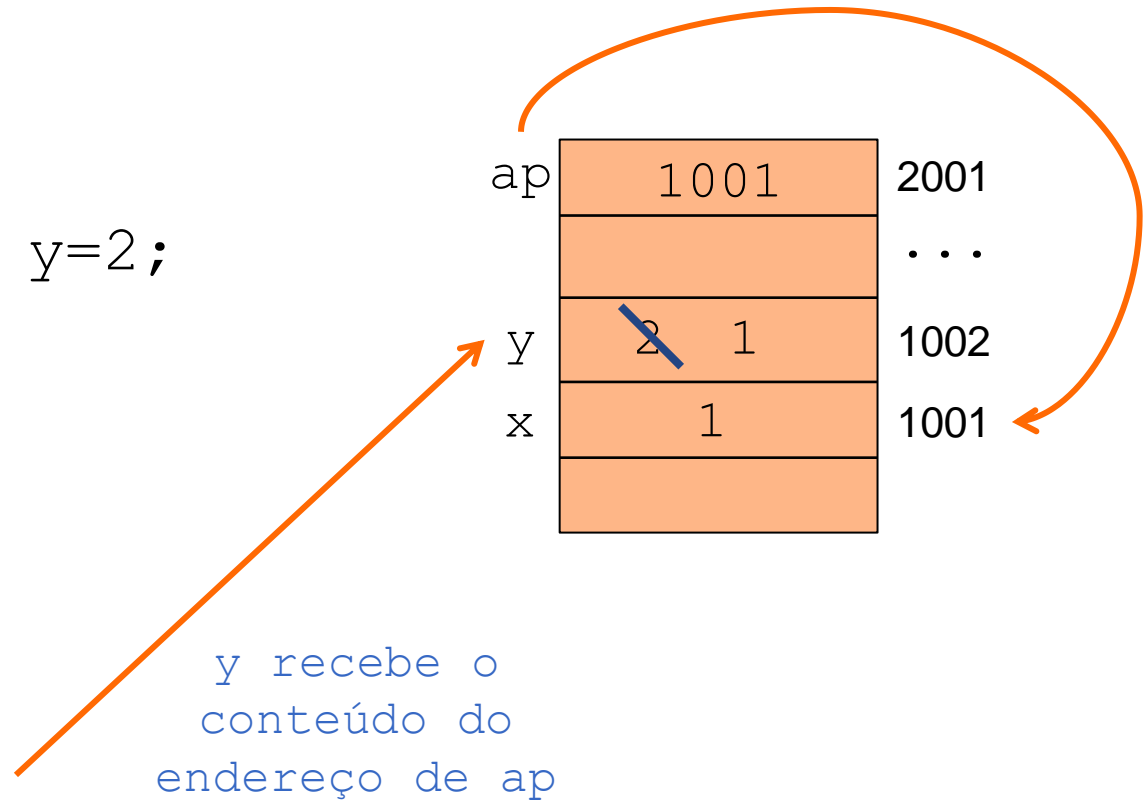
```
int x=1, y=2;
```

```
int *ap;
```

```
...
```

```
ap = &x;
```

```
y = *ap;
```



# PONTEIROS

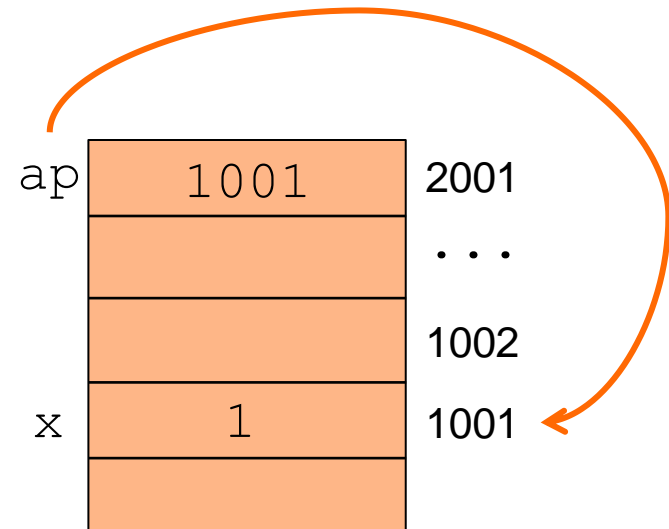
- Um apontador restringe-se a apontar para algum tipo particular de objeto

```
int *ap;
```

- Neste caso, `ap` aponta para um objeto do tipo inteiro.

```
int x=1;
```

```
int *ap;
```



# PONTEIROS

- Se `ap` aponta para um inteiro, então `*ap` pode ocorrer em qualquer contexto onde `x` poderia.

```
int x=1;
```

```
int *ap;
```

- Então:

`*ap = *ap + 10;` → equivale a `x = x + 10`

`(*ap)++;` → equivale a `x++`

`*ap++;` → incrementa o conteúdo de `ap`

# PONTEIROS E PASSAGEM DE PARÂMETROS

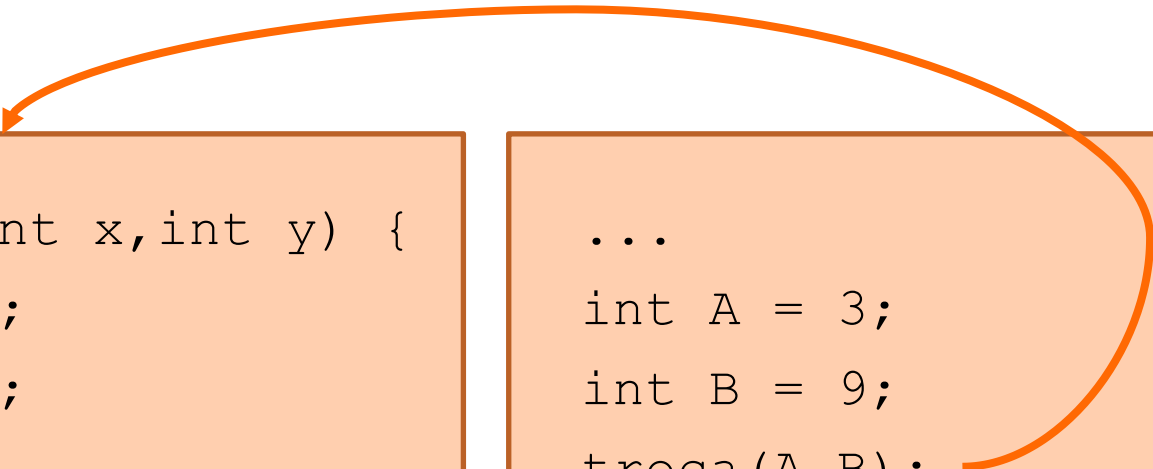
- Até agora usamos apenas a passagem de parâmetros por valor
  - A função chamadora passa uma **cópia** dos valores das variáveis para a função chamada
  - Logo, os valores que são manipulados/alterados dentro da função chamada são apenas cópias
  - Os valores das variáveis **não são alterados** na função chamadora

# PONTEIROS E PASSAGEM DE PARÂMETROS

## ○ Exemplo:

```
void troca(int x,int y) {  
    int temp;  
    temp = x;  
    x = y;  
    y = temp;  
}
```

```
...  
int A = 3;  
int B = 9;  
troca(A,B);  
printf("%d %d",A,B);  
...
```

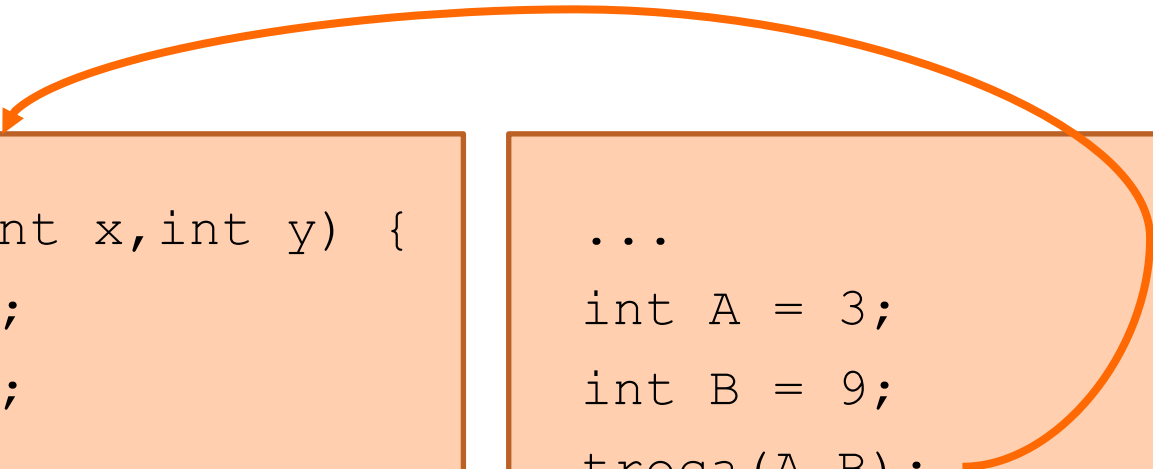


# PONTEIROS E PASSAGEM DE PARÂMETROS

## ○ Exemplo:

```
void troca(int x,int y) {  
    int temp;  
    temp = x;  
    x = y;  
    y = temp;  
}
```

```
...  
int A = 3;  
int B = 9;  
troca(A,B);  
printf("%d %d",A,B);  
...
```



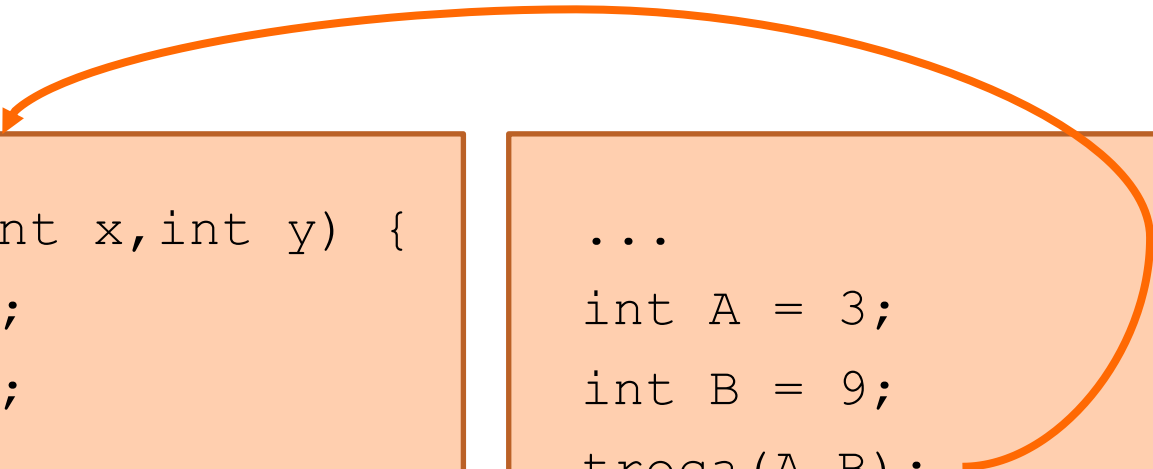
A	B	x	y	temp
3	9			

# PONTEIROS E PASSAGEM DE PARÂMETROS

## ○ Exemplo:

```
void troca(int x,int y) {  
    int temp;  
    temp = x;  
    x = y;  
    y = temp;  
}
```

```
...  
int A = 3;  
int B = 9;  
troca(A,B);  
printf("%d %d",A,B);  
...
```



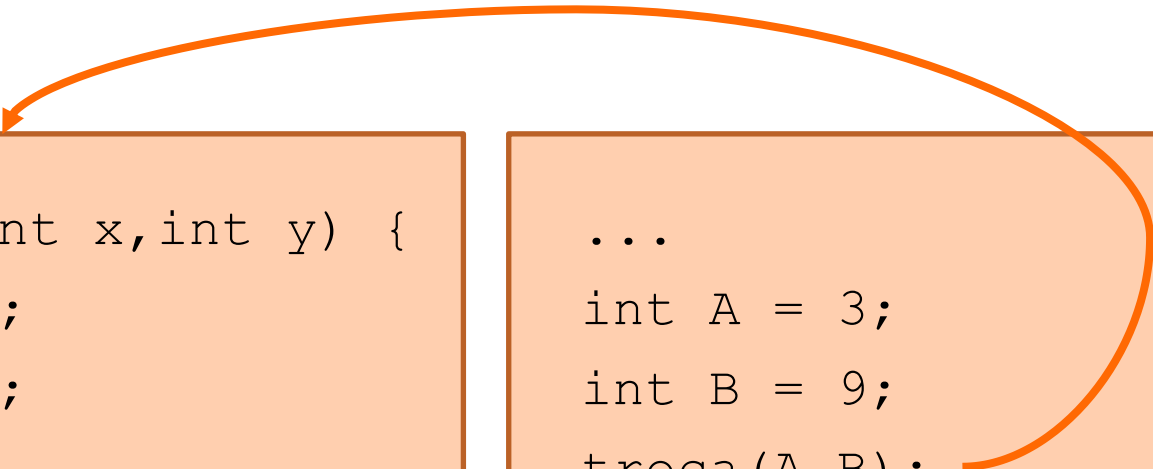
A	B	x	y	temp
3	9	3	9	

# PONTEIROS E PASSAGEM DE PARÂMETROS

## ○ Exemplo:

```
void troca(int x,int y) {  
    int temp;  
    temp = x;  
    x = y;  
    y = temp;  
}
```

```
...  
int A = 3;  
int B = 9;  
troca(A,B);  
printf("%d %d",A,B);  
...
```



A	B	x	y	temp
3	9	3	9	3

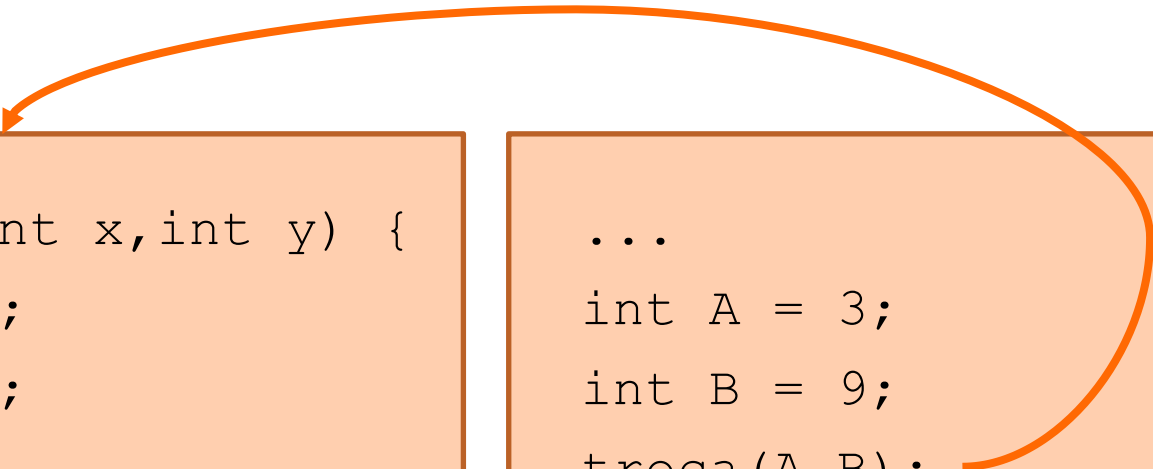


# PONTEIROS E PASSAGEM DE PARÂMETROS

## ○ Exemplo:

```
void troca(int x,int y) {  
    int temp;  
    temp = x;  
    x = y;  
    y = temp;  
}
```

```
...  
int A = 3;  
int B = 9;  
troca(A,B);  
printf("%d %d",A,B);  
...
```



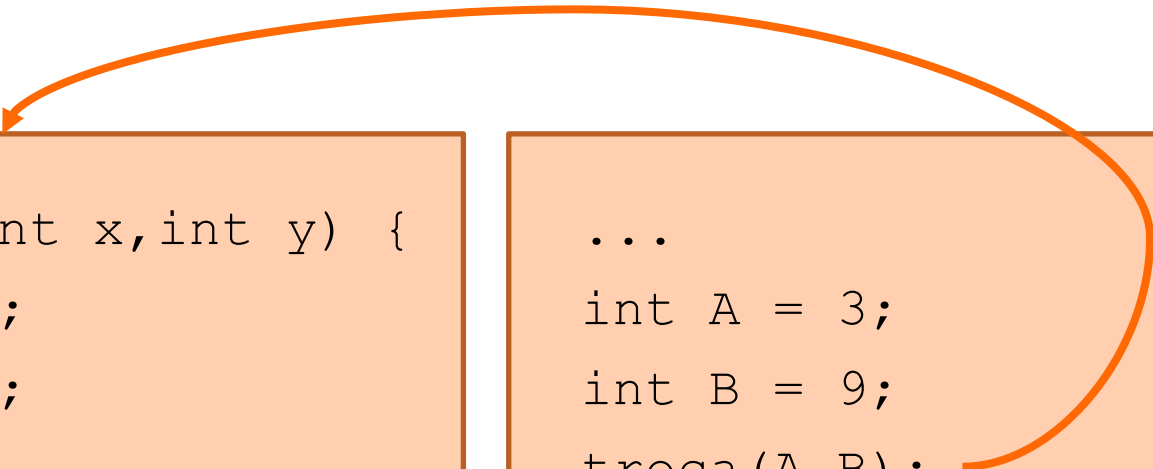
A	B	x	y	temp
3	9	3	9	3
		9		

# PONTEIROS E PASSAGEM DE PARÂMETROS

## ○ Exemplo:

```
void troca(int x,int y) {  
    int temp;  
    temp = x;  
    x = y;  
    y = temp;  
}
```

```
...  
int A = 3;  
int B = 9;  
troca(A,B);  
printf("%d %d",A,B);  
...
```



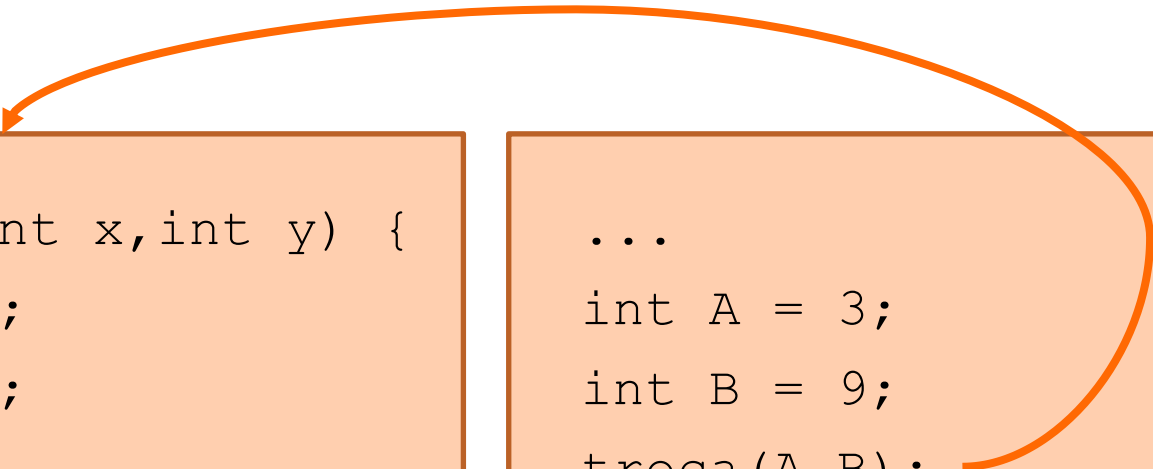
A	B	x	y	temp
3	9	3	9	3
		9	3	

# PONTEIROS E PASSAGEM DE PARÂMETROS

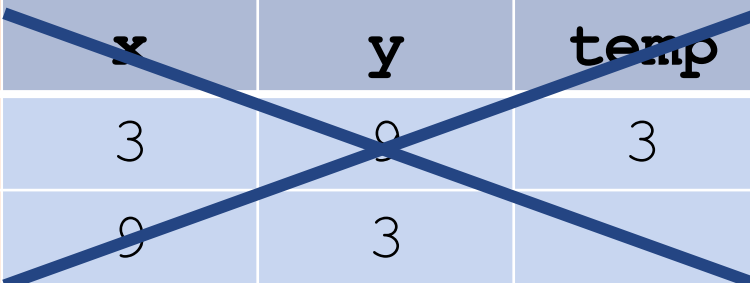
## ○ Exemplo:

```
void troca(int x,int y) {  
    int temp;  
    temp = x;  
    x = y;  
    y = temp;  
}
```

```
...  
int A = 3;  
int B = 9;  
troca(A,B);  
printf("%d %d",A,B);  
...
```



A	B	x	y	temp
3	9	3	9	3
		9	3	



# PONTEIROS E PASSAGEM DE PARÂMETROS


- O resultado da função troca não é o desejado!
- Não podemos usar a passagem de parâmetros por **VALOR**
- Temos que usar a passagem por **REFERÊNCIA**
  - Ao invés de enviarmos cópias dos valores para a função chamada, temos que enviar o endereço das variáveis
  - Desta forma, as alterações feitas dentro da função chamada se refletirão na função chamadora.

# PONTEIROS E PASSAGEM DE PARÂMETROS

## ○ Exemplo:

```
void troca(int *x,int *y){  
    int temp;  
    temp = *x;  
    *x = *y;  
    *y = temp;  
}
```

```
...  
int A = 3;  
int B = 9;  
troca(&A,&B);  
printf("%d %d",A,B);  
...
```



A <b>*x</b>	B <b>*y</b>	temp
3	9	

# EXERCÍCIOS

1. Faça um programa que leia uma cadeia de caracteres em formato de data: DD/MM/AAAA
  - Crie uma função que receba esta data como parâmetro e que retorne o dia, o mês e o ano em três variáveis inteiras distintas. Escreva os valores retornados no programa principal.
  - Crie uma função que receba o ano como parâmetro e retorne se a data caiu em um ano bissexto.