

# Curso de C

*Apontadores*

# Apontadores

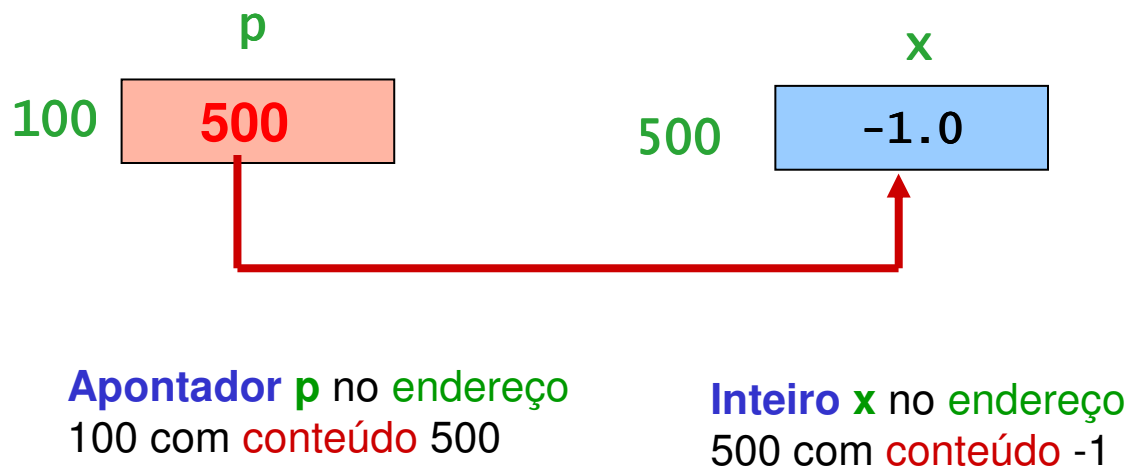
## Objetivos:

- Entender o conceito de apontadores em C
- Variáveis tipo apontadores

# Apontadores

## Apontador:

- Variável que armazena **um endereço**
- Endereço de outra variável, de uma função, etc.



# Apontadores

Declaração de variável tipo apontador:

```
tipo * nome;
```

Qualquer tipo da linguagem C

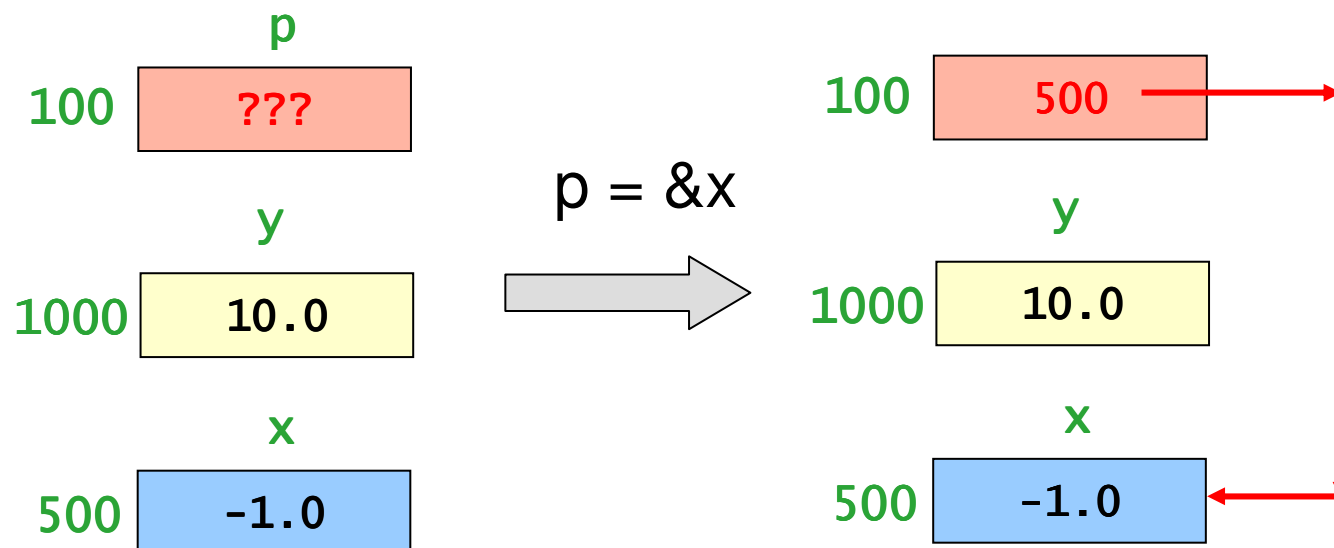
Nome do apontador

**nome** é uma variável que conterá o endereço de variáveis cujo tipo é **tipo**.

# Apontadores

**Operador &:** retorna o endereço do operando

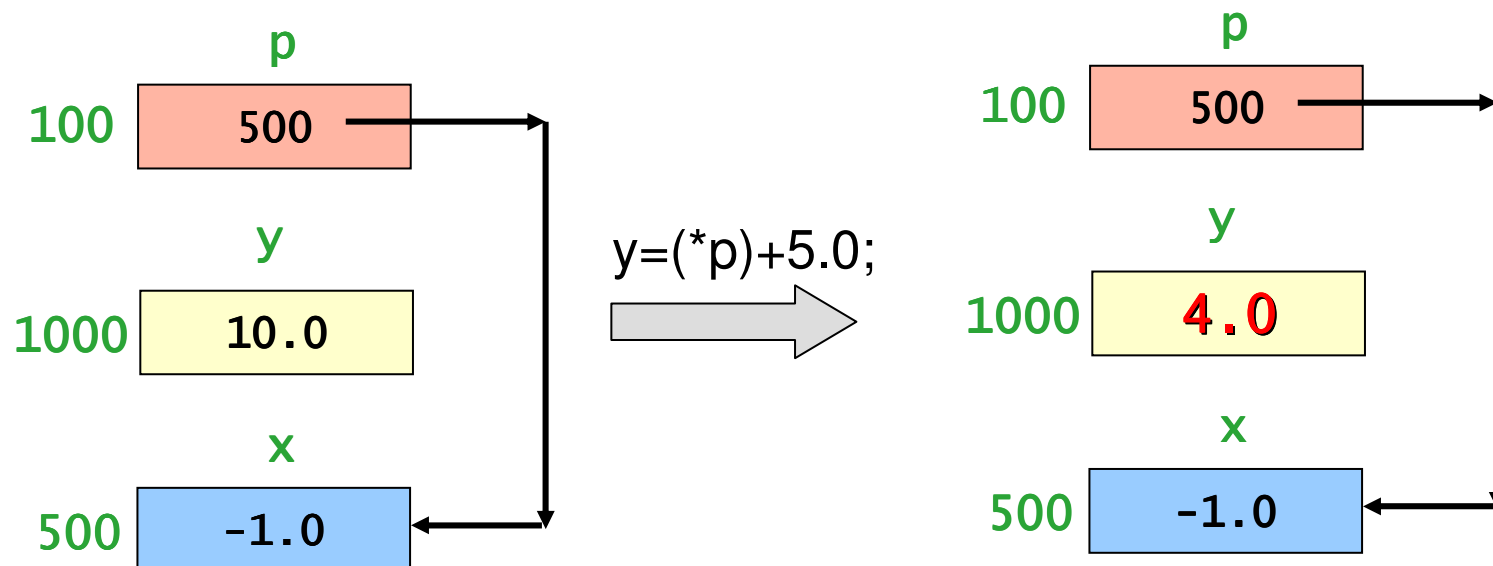
```
float x=-1.0, y=10.0;  
float *p;  
p = &x;
```



# Apontadores

**Operador  $*$  :** no lado direito: retorna o conteúdo do endereço apontado (um **valor**)

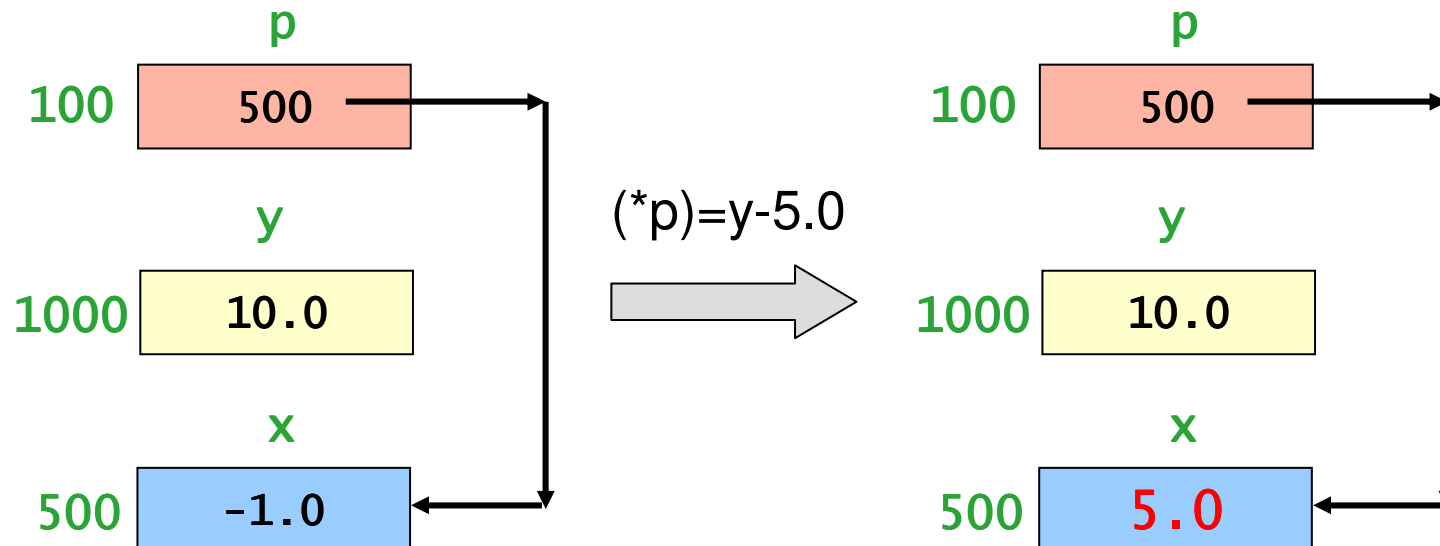
```
float x=-1.0, y=10.0; float *p = &x;  
y = (* p) + 5.0;
```



# Apontadores

**Operador  $*$ :** lado esquerdo, retorna o endereço de memória apontado.

```
float x=-1.0, y=10.0; float *p=&x;
(*p) = y - 5.0f;
```



# Apontadores

## Declarações complexas usando `[ ]`, `( )` e `*`

### Precedências:

<b>1</b>	<code>[ ]</code> , <code>( )</code>
<b>2</b>	<code>*</code>

### Agrupamento:

<code>[ ]</code>	→
<code>( )</code>	→
<code>*</code>	←



# Apontadores

Tipo da variável **x** é:

**int x**

tipo inteiro.

**int \*x**

apontador para um inteiro.

**int x[ ]**

vetor que contém inteiros.



# Apontadores

Tipo da variável **x** é:

```
int x( );
```

função que retorna um inteiro.

```
int *x[ ];
```

vetor que contém apontadores para inteiros.

Mesmo que **int \*(x[ ]);**

```
int *x( );
```

função que retorna um apontador para inteiro.

Mesmo que **int \*(x( ));**



# Apontadores

• `int x( )( )`                      `int (x( ))( )`

função que retorna outra função que retorna um inteiro      **erro!**

• `int *x[ ]( )`                      `int *(x[ ])( )`

vetor que contém funções que  
retornam apontadores para inteiros      **erro!**

• `int *x( )[ ]`                      `int *(x( ))[ ]`

função que retorna vetores de  
apontadores para inteiros      **erro!**

# Apontadores

• `int *x[ ][ ]`                      `int *( (x([]))[] )`

vetor que contém vetores de apontadores  
para inteiros

• `int *x( ) ( )`                      `int *( (x()) ( ) )`

função que retorna uma segunda função que  
retorna um apontador para um inteiro

erro!

• `int (*x)[ ]`

apontador para um vetor de inteiros



# Apontadores

- `int * (* (*x) ( )) [ ]`
  - `(*x) ( )` apontador para uma função que retorna...
  - `(* ( . . . ) )` um apontador para ...
  - `int * ( . . . ) [ ]` um vetor de apontadores para inteiros.

*um apontador para uma função que retorna  
um apontador para  
um vetor de apontadores para inteiros.*

# Apontadores

## Apontadores e vetores:

```
int vet[4];
```

```
vet[0] == *vet
```

vet

101

inicia em 0

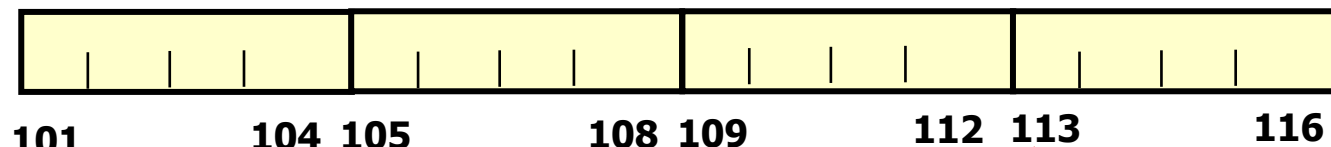
contém um int

vet[0]

vet[1]

vet[2]

vet[3]



4 bytes

endereços de  
memória

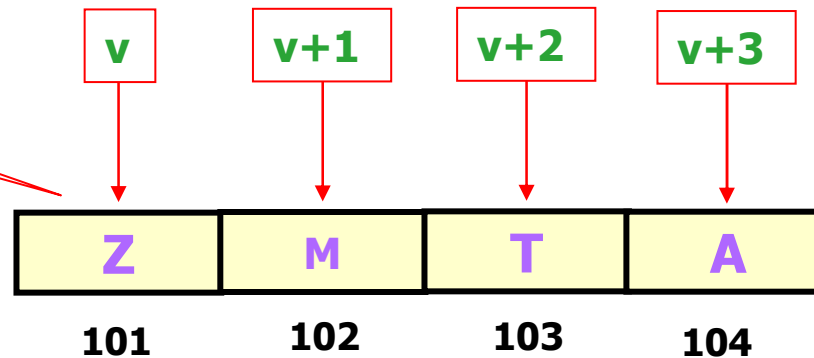
```
vet[2] == *(vet+2)
```

# Apontadores

## Apontadores e vetores:

```
char v[ ] = { 'Z', 'M', 'T', 'A' } ;
```

1 byte



**v**

101

500

```
v[2] == *(v+2) == 'T'
```

# Apontadores

## Exemplo: apontadores para funções

```
int soma(int x, int y) { return (x+y); }
```

```
int (* f)();
```

```
f=soma;
```

```
soma(2,3);  
f(2,3);  
(* soma)(2,3);  
(*** f)(2,3);
```

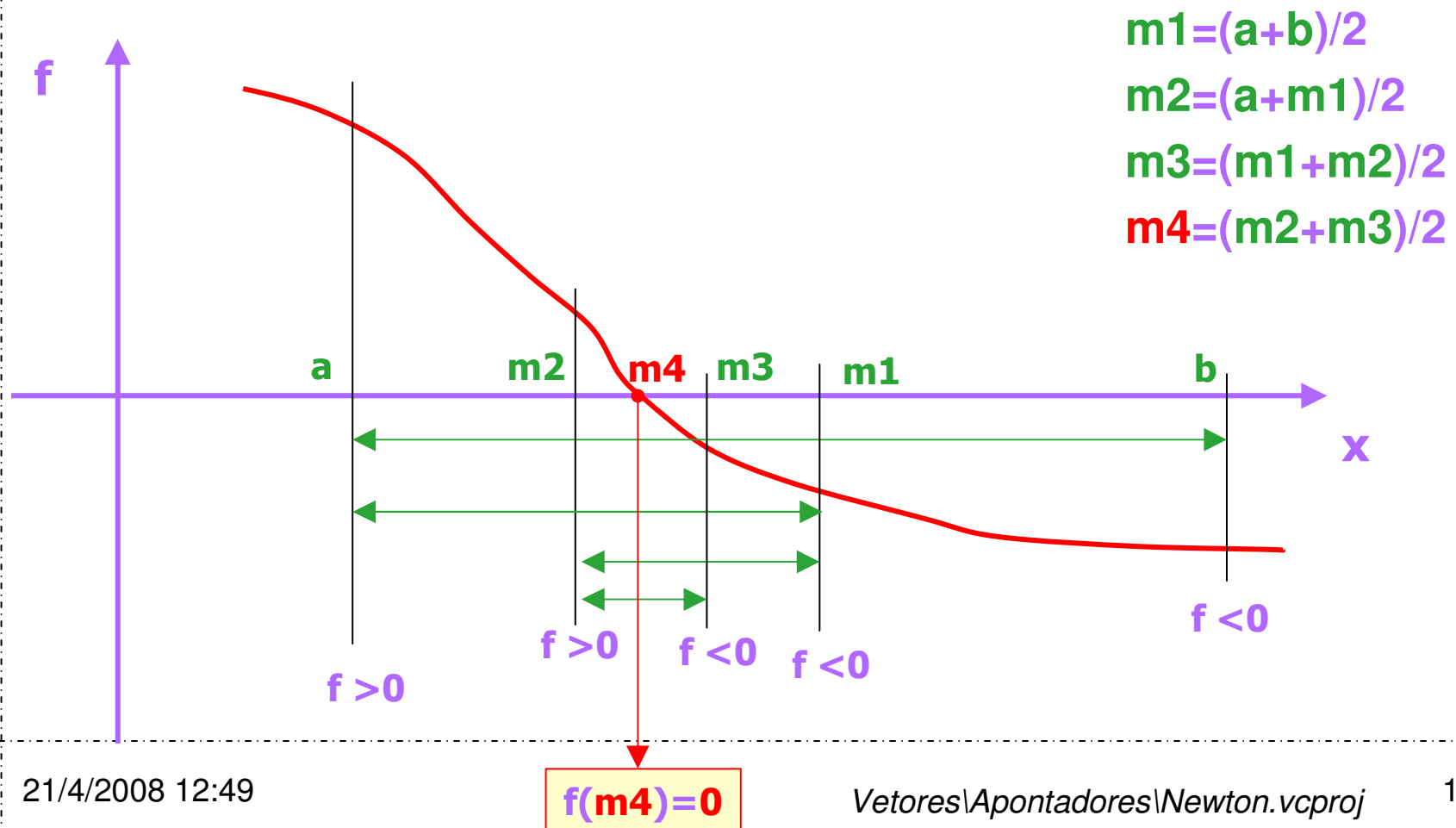
soma	código	103
f	???	500

soma	código	103
f	103	500

5

# Apontadores

## Apontadores para funções: Método de Newton



# Apontadores

*Fim*