

## Ponteiros

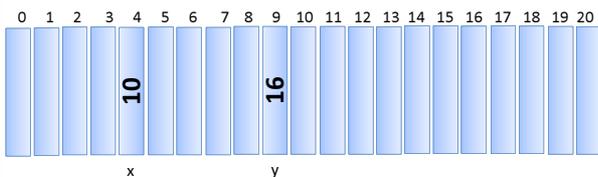
Introdução à Ciência da Computação I

Prof. Denis F. Wolf

## Memória



## Memória

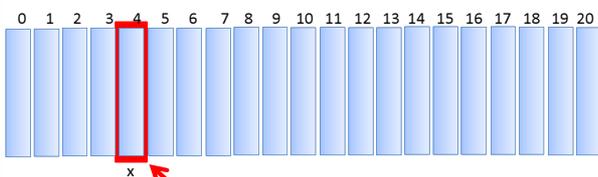


```
int main () {  
  int x;  
  x = 10;  
  return 0;  
}  
  
int y = x + 6;  
printf("%d", x);  
scanf("%d", &x);
```

## Endereço de Memória

- O operador “&” quando aplicado sobre um identificador (nome de variável, por exemplo) retorna o seu endereço

## Memória

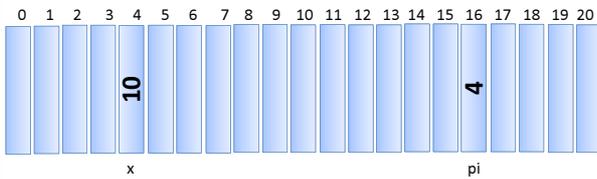


```
scanf("%d", &x);
```

## Ponteiros

- Um ponteiro é uma variável que contém (armazena) um endereço de memória
- Declaração:  
`tipo_dado *nome_ponteiro;`  
onde “\*” indica que a variável é um ponteiro
- Ex: `int x;`  
`int *px;` compilador sabe que px é ponteiro \*/  
/\* px é um ponteiro para inteiro \*/

## O que é um ponteiro?

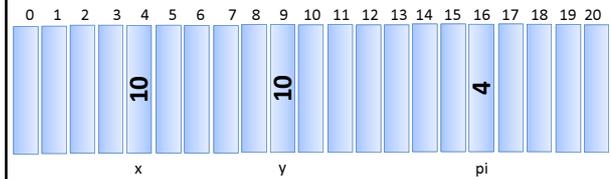


```
int x = 10, *pi;  
pi = &x;  
printf("&x: %p pi: %p", &x, pi);
```

Saída em tela:

**&x: 0x4 pi: 0x4**

## Como usar um ponteiro?



```
int x = 10, *pi, y;  
pi = &x;  
y = *pi;
```

O operador "\*" quando aplicado sobre um ponteiro retorna o dado apontado

## Utilizando Ponteiros

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

```
int main() {  
    int x = 10;  
    int *pi;  
    pi = &x; /* *pi é igual a 10 */  
    (*pi)++; /* *pi é igual a 11 */  
    printf("%d", x);  
    return 0;  
}
```

ao alterar **\*pi** estamos alterando o conteúdo de x

## Utilizando Ponteiros

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

```
int main() {  
    int x = 10;  
    int *pi, *pj;  
  
    pi = &x; /* *pi == 10 */  
    pj = pi; /* *pj == 10 */  
    (*pi)++; /* (*pi, *pj, x) == 11 */  
    (*pj)++; /* (*pi, *pj, x) == 12 */  
    printf("%d", x); /* Escreverá 12 */  
    return 0;  
}
```

## PONTEIROS & ARRAYS

## Referenciando Arrays

- Pode-se referenciar os elementos de um array através de ponteiros

- Ex: `float m[] = { 1.0, 3.0, 5.75, 2.345 };`  
`float *pf;`  
`pf = &m[2];`  
`printf("%f", *pf); /* Escreve 5.75 */`

## Referenciando Elementos

- Pode-se utilizar ponteiros e colchetes:
 

```
float m[] = { 1.0, 3.0, 5.75, 2.345 };
float *pf;
pf = &m[2];
printf("%f", pf[0]); /* ==> 5.75 */
```
- Note que o valor entre colchetes é o deslocamento a ser considerado a partir do endereço de referência
  - `pf[n]` => indica *n*-ésimo elemento a partir de `pf`

## PONTEIROS & PARÂMETROS DE FUNÇÕES

## Passagem de Informações

- Argumentos passados **por referência**
  - Quando chamada, a função passa a referenciar (apontar) a variável informada
  - Portanto o processo consiste em informar o endereço da variável para o que o parâmetro formal possa referenciá-lo.
  - Os argumentos deixam de existir após a execução do método, porém as variáveis informadas e que foram referenciadas permanecem (pois não são dadas por este bloco de comandos).

15

## Exemplo

```
#include <stdio.h> /* Definição das subrotinas */

/* Protótipos */
void funcPorValor(int a);
void funcPorRefer(int *b);

int main () {
    int x = 0, y = 0;

    funcPorValor(y);
    printf("%d %d\n", x, y);

    funcPorRefer(&y);
    printf("%d %d\n", x, y);

    return 0;
}
```

• Note que as variáveis *x* e *y* são locais a função *main*, enquanto os parâmetros *a* e *b* são locais a *funcPorValor* e *funcPorRefer*, respectivamente.

## Operações Válidas Sobre Ponteiros

| É válido   | Não é válido  |
|--|---|
| <ul style="list-style-type: none"> <li>• <i>somar</i> ou <i>subtrair</i> um inteiro a um ponteiro (<math>pi \pm int</math>)</li> <li>• <i>incrementar</i> ou <i>decrementar</i> ponteiros (<math>pi++</math>, <math>pi--</math>)</li> <li>• <i>subtrair</i> ponteiros (produz um inteiro) (<math>pf - pi</math>)</li> <li>• <i>comparar</i> ponteiros (<math>&gt;</math>, <math>&gt;=</math>, <math>&lt;</math>, <math>&lt;=</math>, <math>==</math>)</li> </ul> | <ul style="list-style-type: none"> <li>• somar ponteiros (<math>pi + pf</math>)</li> <li>• multiplicar ou dividir ponteiros (<math>pi * pf</math>, <math>pi / pf</math>)</li> <li>• operar ponteiros com <i>double</i> ou <i>float</i> (<math>pi \pm 2.0</math>)</li> </ul> |

## Exercícios

- 1) Faça um programa que leia 2 valores inteiros e chame uma função que receba estas 2 variáveis e troque o seu conteúdo, ou seja, esta rotina é chamada passando duas variáveis A e B por exemplo, e após a execução da rotina A conterá o valor de B e B terá o valor de A.
- 2) Escreva uma função que dado um número real passado como parâmetro, retorne a parte inteira e a parte fracionária deste número.

### Exercícios

- 3) Crie uma função que recebe os coeficientes de uma função do 2o. grau e retorna as raízes **sem usar vetor**.
- 4) Faça um programa leia um vetor de 5 inteiros e, **depois de ler**, imprima os valores, sem usar a notação de vetor [].

### Exercícios

- 5) Faça um programa que acha o maior e o menor inteiro dentro de um vetor de 10 inteiros.

**Obs: usar apenas as variáveis a seguir:**

```
int v[10], i, *maior, *menor;
```