Ponteiros

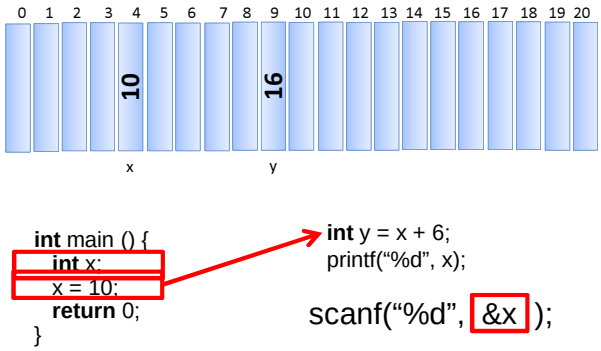
Introdução à Ciência da Computação I

Prof. Denis F. Wolf
Estagiário PAE: Patrick Y. Shinzato

Memória

Memória

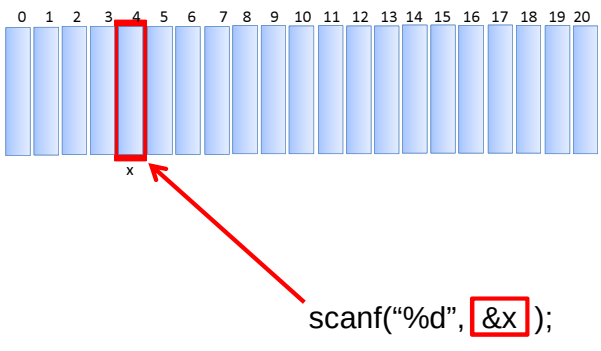
Memória



Endereço de Memória

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

Memória



Ponteiros

- Um ponteiro é uma variável que contém (armazena) um endereço de memória
- Declaração:

  tipo_dado *nome_ponteiro;

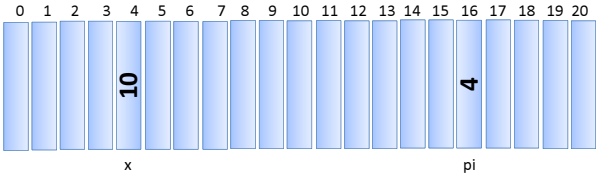
  onde "*" indica que a variável é um ponteiro

- Ex: **int** x;

  int *px; /* compilador *sabe* que px é ponteiro */

  /* px é um *ponteiro* para inteiro */

O que é um ponteiro?

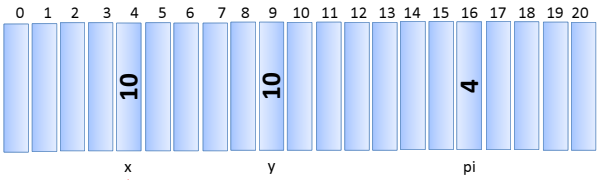



```
int x = 10, *pi;
pi = &x;
printf("&x: %p pi: %p", &x, pi);
```

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

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

Exercício

- Faça um programa que acha o maior e o menor inteiro dentro de um vetor de 10 inteiros e some-os. Obs: usar apenas as váriaveis a seguir:

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

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).

16

Exemplo

```
#include <stdio.h>

/* 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;
}
```

```
/* Definição das subrotinas */

void funcPorValor(int a){
  a = 1;
}

void funcPorRefer(int *b){
  *b = 2;  /* ... o conteúdo apontado
              por b recebe 2 */
}
```

- 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
• *somar* ou *subtrair* um inteiro a um ponteiro (pi ± int)	• somar ponteiros (pi + pf)
• *incrementar* ou *decrementar* ponteiros (pi++, pi--)	• multiplicar ou dividir ponteiros (pi*pf, pi/pf)
• *subtrair* ponteiros (produz um inteiro) (pf - pi)	• operar ponteiros com *double* ou *float* (pi ± 2.0)
• *comparar* ponteiros (>, >=, <, <=, ==)	

Exercícios

1) Crie uma função que recebe os coeficientes de uma função do 2o. grau e retorna as raízes.

2) Faça uma função que recebe 3 valores inteiros e retorna-os ordenados.

Exercícios

3) Crie uma função que receba um vetor e substitua todos os valores menores que zero por zero.

4) Faça uma função que recebe 2 vetores de 10 elementos inteiros e que calcule e retorne o vetor soma dos dois primeiros.

Exercícios

5) Criar uma função que receba como argumento um número real e um número inteiro e retorne a raiz quadrada do número real através do

$R_1 = E/2$

$R_{n+1} = (R_n + (E/R_n))/2$

para E = entrada, R = raiz quadrada
e n = número de iterações