ANSI

C

Introdução à Programação

Eduardo Silva Lira

XLVIII Programa de Verão do IME-USP

São Paulo - SP, Jan 2019

INSTITUTO DE MATEMATICA E ESTATÍSTICA
UNIVERSIDADE DE SÃO PAULO

USP

Parâmetros da função

Questão: Quais serão os valores exibidos?

```c
#include<stdio.h>

int dobro(int a){
  a = a * 2;
  return a;
}

int main(){
  int a = 5;
  printf("%d\n", a);
  printf("%d\n", dobro(a));
  printf("%d\n", a);
  return 0;
}
```

Parâmetros da função

- Parâmetros por **valor**
  - Vai o valor
    - Cópia, outro espaço de memória
  - A variável original não é alterada

- Parâmetros por **referência**
  - Vai o endereço de memória
  - A variável original é alterada

- Como fazemos isto?!

Parâmetros da função

- Parâmetros por **valor**
  - Já conhecemos
- Parâmetros por **referência**
  - Veremos!
- Como fazemos isto?!

Parâmetros por referência

Questão: E agora, quais serão os valores exibidos?

```c
#include<stdio.h>

int dobro(int *a){
  *a = *a * 2;
  return *a;
}

int main(){
  int a = 5;
  printf("%d\n", a);
  printf("%d\n", dobro(&a));
  printf("%d\n", a);
  return 0;
}
```

Parâmetros por referência

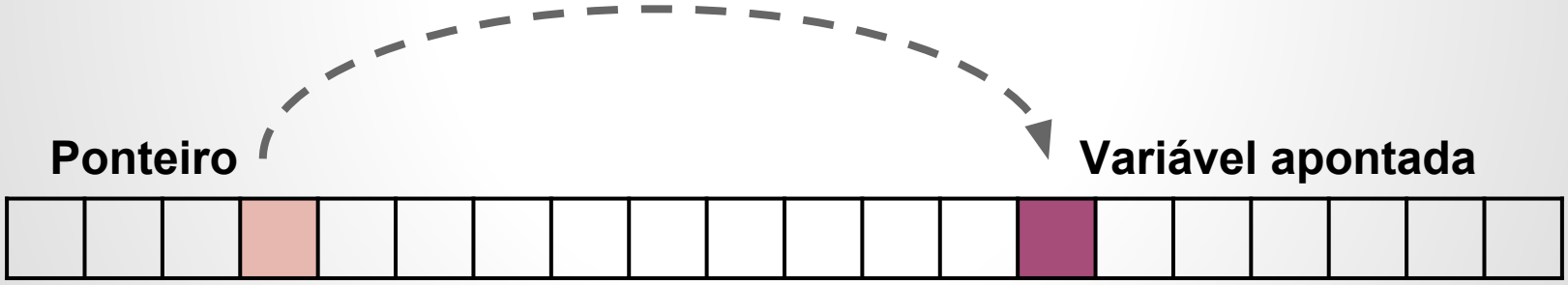
- Vocês acabaram de serem introduzidos ao conteúdo de ponteiros!
  - Variáveis que armazenam endereços de memória
    - Ao invés de armazenar um dado, elas apontam onde o dado deve ser buscado

Ponteiros

- Uma variável que contém um endereço de memória
- Código compacto
- Cuidado para não se complicar!
  - Programas impossíveis de entender!

Ponteiros

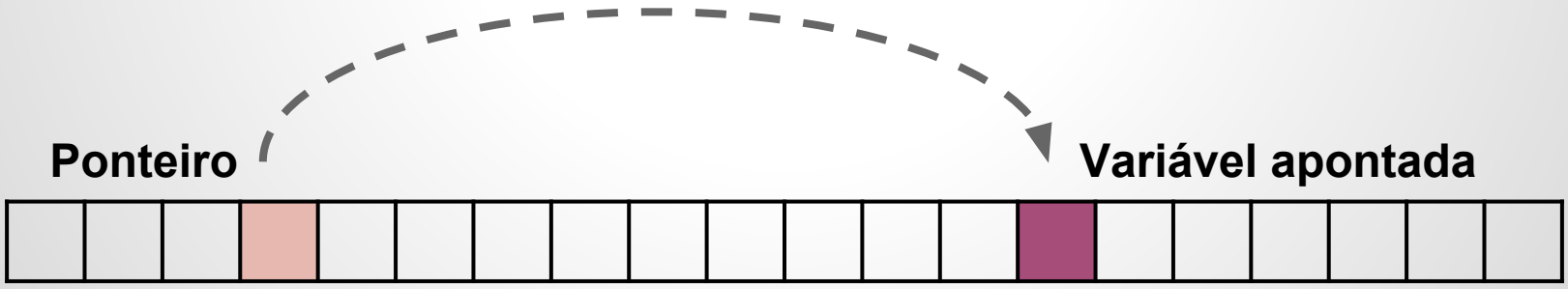
- Uma variável que contém um endereço de memória



Representação da memória (um bloco representa uma variável e não necessariamente um byte)

Ponteiros

- **&**
  - Retorna o endereço de um objeto
- ***
  - Na declaração, informa que é um ponteiro.
  - No código, retorna a variável apontada (valor)



Representação da memória (um bloco representa uma variável e não necessariamente um byte)

Ponteiros

- Exemplo 1:

```c
#include<stdio.h>
int main(){
   int *pt;
   int x;

   pt = &x;
   scanf("%d", pt);

   printf("%d\n", x);
   printf("%d\n", *pt);
   return 0;
}
```

Ponteiros

- Exemplo 2:

```
#include<stdio.h>
int main(){
   int *pt;
   int a = 5, b = 12, c = 19;

   pt = &a;
   b = *pt;
   *pt = c;

   return 0;
}
```

Ponteiros

- Exemplo 2: comentários

```c
#include<stdio.h>
int main(){
   int *pt;
   int a = 5, b = 12, c = 19;

   pt = &a; /*pt aponta para a*/
   b = *pt; /*b vale o valor de a (5)*/
   *pt = c; /*a vale o valor de c (19)*/

   return 0;
}
```

Ponteiros

- Exemplo 3:

```c
#include<stdio.h>
int main(){
  double *pt1, *pt2;
  double x, y = 3.6, zv[2] = {5.4, 3.97};

  pt1 = &y;
  pt2 = &zv[1];
  pt2 = pt1;
  y = 2.0;
  x = *pt1 + *pt2 + zv[0];

  printf("%f\n", x);
  return 0;
}
```

Ponteiros

- Exemplo 3: comentários

```c
#include<stdio.h>
int main(){
  double *pt1, *pt2;
  double x, y = 3.6, zv[2] = {5.4, 3.97};

  pt1 = &y; /*pt1 aponta para y*/
  pt2 = &zv[1]; /*pt2 aponta para zv[1]*/
  pt2 = pt1; /*os dois ponteiros apontam para y*/
  y = 2.0; /*y vale 2.0*/
  x = *pt1 + *pt2 + zv[0];

  printf("%f\n", x);
  return 0;
}
```

Ponteiros

- Exemplo 4:

```c
#include<stdio.h>
int main(){
   int *p1, *p2;
   int a = 5, b = 5;

   p1 = &a;
   p2 = &b;

   printf("%c\n", p1 == p2 ? 'S' : 'N');
   return 0;
}
```

Ponteiros

- Exemplo 5:

```c
#include<stdio.h>
int main(){
   int *p1, *p2;
   int a = 5, b = 5;

   p1 = &a;
   p2 = &b;

   printf("%c\n", *p1 == *p2 ? 'S' : 'N');
   return 0;
}
```

Ponteiros

- Exemplo 6:

```c
#include<stdio.h>
int main(){
  double *p1, *p2;
  double vet[3] = {8.2, 3.5, 6.8};

  p1 = vet;
  p2 = &vet[0];

  printf("%c\n", p1 == p2 ? 'S' : 'N');
  (*p1)++;
  p2++;

  printf("Valor var apontada por p1: %f\n", *p1);
  printf("Valor var apontada por p2: %f\n", *p2);
  return 0;
}
```

Ponteiros

Ponteiro: qual o tamanho em memória?

```c
#include<stdio.h>
int main(){
  char *p1;
  int  *p2;

  printf("Tam pont char: %ld\n", sizeof(p1));
  printf("Tam var char apontada: %ld", sizeof(*p1));

  printf("\nTam pont int: %ld\n", sizeof(p2));
  printf("Tam var int apontada: %ld\n", sizeof(*p2));
  return 0;
}
```

Ponteiros

Ponteiro: qual o tamanho em memória?

```c
#include<stdio.h>
int main(){
  char *p1;
  int  *p2;

  printf("Tam pont char: %ld\n", sizeof(p1));
  printf("Tam var char apontada: %ld", sizeof(*p1));

  printf("\nTam pont int: %ld\n", sizeof(p2));
  printf("Tam var int apontada: %ld\n", sizeof(*p2));
  return 0;
}
```

Altere o formatador de %ld para %d se estiver em sistema 32 bits

Ponteiros - Exercícios

- Faça um programa para
  - Ler um double **x**
  - Apontar **x** pelo ponteiro **px**
  - Dividir **x** por 2.54 sem utilizar **x**

Ponteiros - Exercícios

- Crie uma função que altere o número para sua metade
  - Não pode utilizar return!

Ponteiros - Exercícios

- Crie uma função de **swap**: uma função que recebe dois parâmetros e troca seus valores.

Ponteiros - Exercícios

- Crie uma função que converta uma sequência de caracteres para maiúscula utilizando ponteiros.

Ponteiros - Exercícios

- Crie uma função para ler um inteiro não negativo com ponteiro.

Dúvidas?