Estrutura de Dados 1

Prof. Igor Calebe Zadi
igor.zadi@ifsp.edu.br

INSTITUTO FEDERAL
São Paulo
Campus Catanduva

I. Fundamentos de Estruturas de Dados

INSTITUTO FEDERAL
São Paulo
Campus Catanduva

I. Fundamentos de Estruturas de Dados

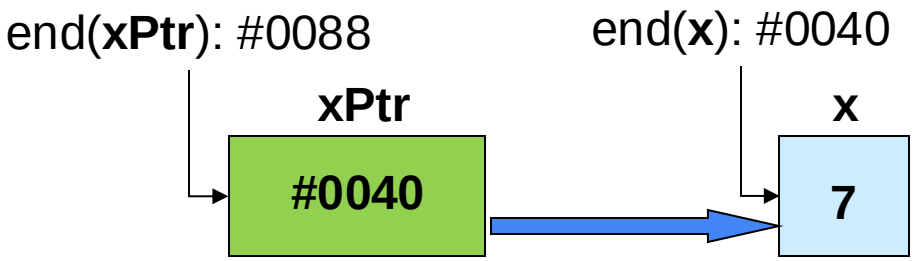
1. Definições
2. Classificação das Estruturas de Dados
3. Programação Orientada a Procedimentos
4. Tipos de dados primitivos
5. Cadeias de caracteres
6. Registros
7. Ponteiros

7. Ponteiros

INSTITUTO FEDERAL
São Paulo
Campus Catanduva

7. Ponteiros

- Ponteiro
  - Contém, como valor armazenado, um **endereço de memória**
  - Normalmente, não é considerado uma ED, mas possui organização e operações definidas
  - Dada uma certa variável **x**, um ponteiro **xPtr** contém o endereço de **x**



xPtr aponta para **x**

7. Ponteiros

- Ponteiro
  - É útil para criar e manipular ED **dinâmicas** (que podem mudar de tamanho durante a execução do programa)
  - Normalmente, ocupa **uma palavra** na memória (não depende do tipo da ED “apontada”)

7. Ponteiros

- Em linguagem **C**
  - Declaração de uma variável ponteiro
    - é preciso especificar o tipo da variável “apontada”

```
int * xPtr;
```

Tipo da variável “apontada”

Símbolo que identifica a declaração de um ponteiro

  - Inicialização com valor nulo

```
xPtr = 0;
```

7. Ponteiros

- Operadores
  - **&** : operador de **endereço**
    - fornece o endereço de seu operando
  - * : operador de **indireção** ou de **de-referência**
    - retorna o conteúdo do objeto para o qual o seu operando “aponta”

7. Ponteiros

- Operadores – Exemplos (linguagem **C**)

```
int y = 5;  // vamos supor que end(y) é #0200
int * yPtr;
...
yPtr = &y;  // equivale a fazer yPtr = #0200
...
printf ( *yPtr );   // equivale a fazer printf ( y );
...
*yPtr = 9;  // equivale a fazer y = 9
...
&*yPtr ...  // resulta no endereço armazenado pelo ponteiro
...
*&yPtr ...  // resulta no endereço armazenado pelo ponteiro
```

7. Ponteiros - Exemplos

- Ponteiros para *int*

```
int a = 10;
int *aPtr;
aPtr = &a;
printf("conteudo de a %d", a);
printf("\nconteudo de aPtr %p", aPtr);
printf("\nendereco de a %p", &a);
printf("\nendereco de aPtr %p", &aPtr);
printf("\nvalor indirecao de aPtr %d", *aPtr);
```

7. Ponteiros - Exemplos

- Ponteiros para *double*

```
double b = 11.99;
double *bPtr;
bPtr = &b;
printf("conteudo de b %g", b);
printf("\nconteudo de bPtr %p", bPtr);
printf("\nendereco de b %p", &b);
printf("\nendereco de bPtr %p", &bPtr);
printf("\nindirecao de bPtr %g", *bPtr);
```

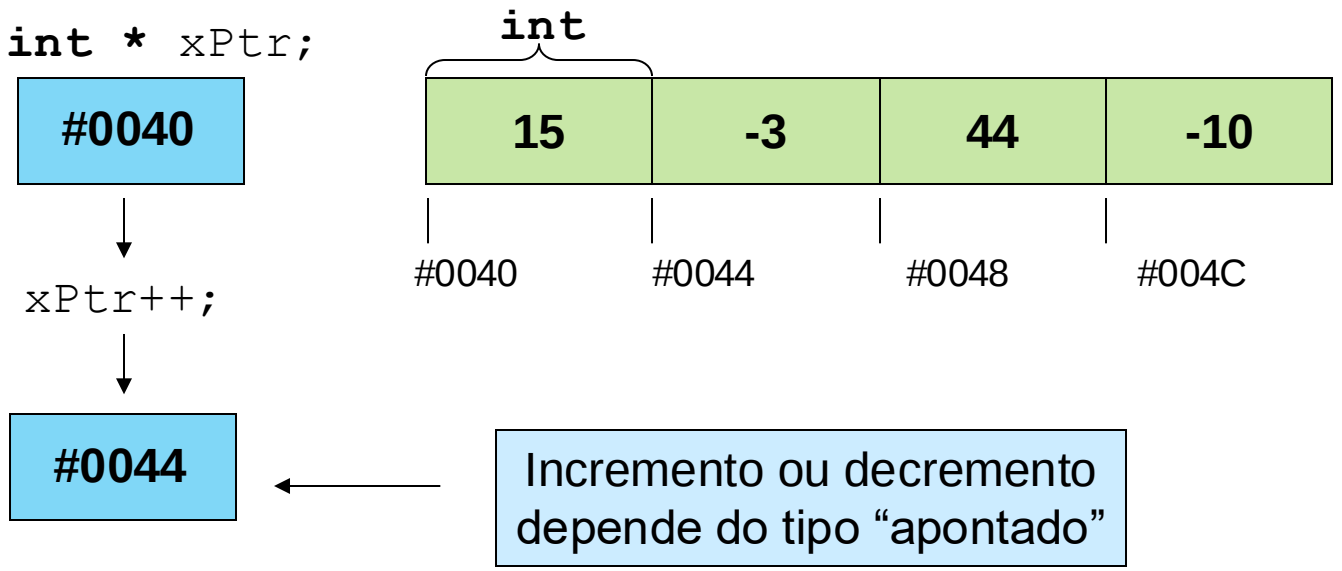
7. Ponteiros - Exemplos

- Ponteiros para *ponteiros*

```
double x = 10.00;
double * xPtr;
xPtr = &x;
double **ptrXPtr;
ptrXPtr = &xPtr;
printf("conteudo de x %f", x);
printf("\nconteudo de xPtr %p", xPtr);
printf("\nendereco de x %p", &x);
printf("\nendereco de xPtr %p", &xPtr);
printf("\nconteudo de ptrXPtr %p", ptrXPtr);
printf("\nendereco de ptrXPtr %p", &ptrXPtr);
printf("\nindirecao de ptrXPtr %p", *ptrXPtr);
printf("\nindirecao do conteudo de ptrXPtr %f", **ptrXPtr);
printf("\nindirecao de xPtr %f ",*xPtr);
```

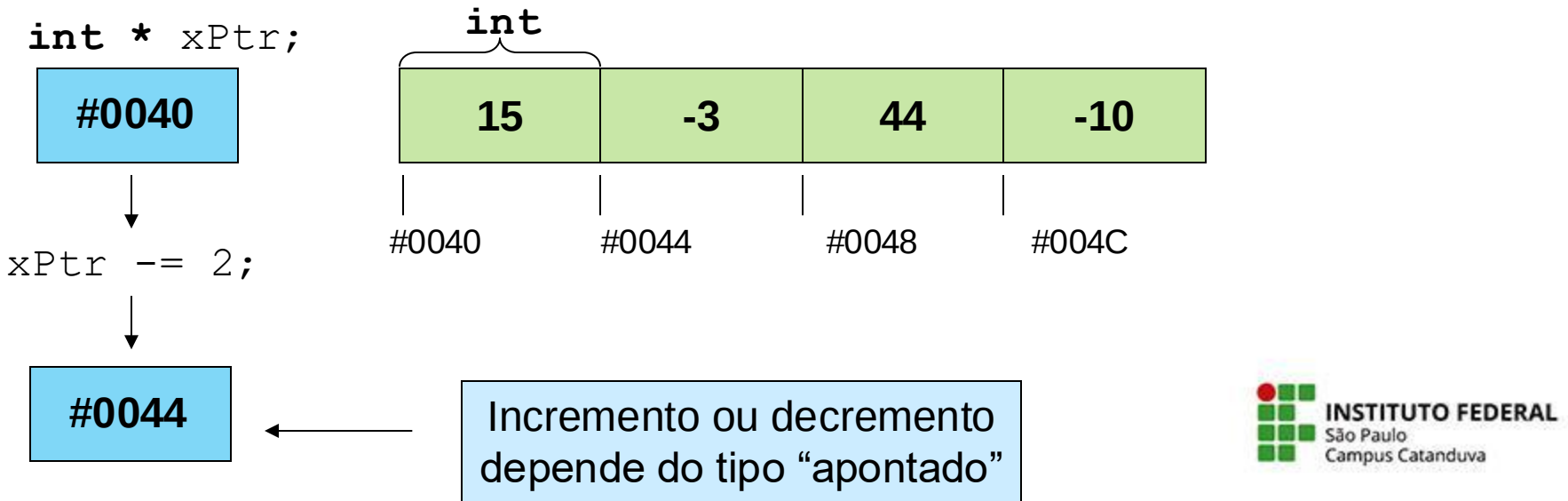
7. Ponteiros

- Aritmética de ponteiros
  - Um ponteiro pode ser incrementado (**++**) ou decrementado (**--**)



7. Ponteiros

- Aritmética de ponteiros
  - Um inteiro pode ser
    - somado a um ponteiro (**+** ou **+=**)
    - subtraído de um ponteiro (**-** ou **-=**)



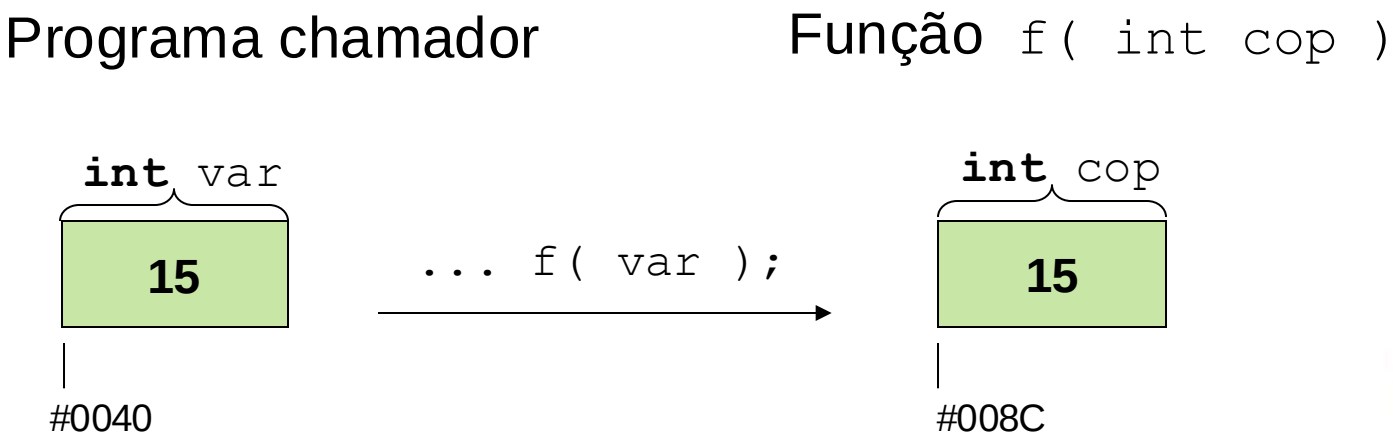
INSTITUTO FEDERAL
São Paulo
Campus Catanduva

7. Ponteiros

- Uma aplicação: passagem de parâmetros
  - Existem duas maneiras de passar parâmetros para uma função
    - por **valor** (a função cria uma cópia do parâmetro)
    - por **referência** (a função acessa a variável original)
  - Algumas linguagens só implementam a passagem por valor (exemplo: linguagem **C**)
  - Ponteiros são capazes de simular passagem de parâmetros por referência

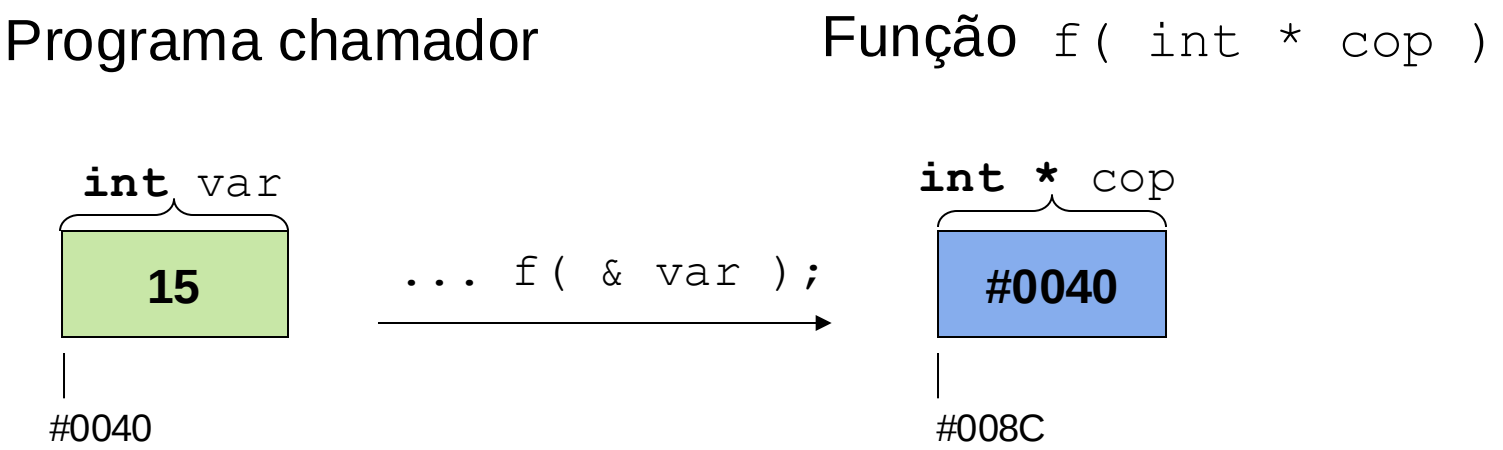
7. Ponteiros

- Passagem por valor (cópia)
  - a função trabalha com uma **cópia** da variável original
  - a variável original não é alterada



7. Ponteiros

- Passagem com ponteiros
  - simulação de passagem por referência
  - o **ponteiro** para a variável original é copiado
  - vantagem: passagem de parâmetros *grandes*



INSTITUTO FEDERAL
São Paulo
Campus Catanduva

Exemplo

```
int main(){
 int x, y, *p;
 y = 0;
 p = &y;
 x = *p;
 x = 4;
 (*p)++;
 --x;
 (*p) += x;
 printf("x %d",x);
 printf("y %d",y);
}
```

x	y	p
0	0	&y
4	1	
3	4	

INSTITUTO FEDERAL
São Paulo
Campus Catanduva

Exercício

```
int main(){
   int a,b,*p1, *p2;
   a = 4;
   b = 3;
   p1 = &a;
   p2 = p1;
   *p2 = *p1 + 3;
   b = b * (*p1);
   (*p2)++;
   p1 = &b;
   printf("%d %d\n", *p1, *p2);
   printf("%d %d\n", a, b);
}
```

a	b	p1	p2
4	3		

Observações sobre o material eletrônico

- O material ficará disponível na pasta compartilhada que é acessada sob convite
- O material foi elaborado a partir de diversas fontes (livros, internet, colegas, alunos etc.)
- Alguns trechos podem ter sido inteiramente transcritos a partir dessas fontes
- Outros trechos são de autoria própria
- Esta observação deve estar presente em qualquer utilização do material fora do ambiente de aulas do IFSP - Catanduva

INSTITUTO FEDERAL
São Paulo
Campus Catanduva