ESTRUTURAS DE DADOS I
Ciência da Computação

ATENÇÃO:

Conforme observação no Plano de Ensino disponível e publicitado a todos os alunos desta disciplina, comunico que não autorizo a gravação, filmagem, captação de imagens por meio de fotografias e congêneres, reprodução e ou divulgação dos materiais didáticos e de ensino de minha autoria produzidos, organizados e utilizados em sala de aula, sem a expressa autorização da minha parte. Dada ciência, sob a égide da Constituição Federal de 1988, e demais legislação vigente, em especial a Lei nº 9610/98, Art. 46, IV e de direito à imagem, desde já ficam todos sujeitos às devidas sanções administrativas e ou legais cabíveis.

Igor Yepes
SIAPE 2289347 - Professor EBTT
IFFar - Campus Frederico Westphalen

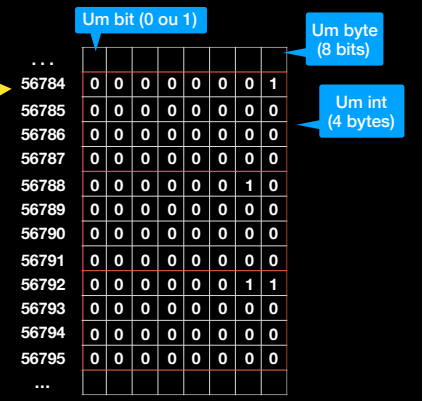
Ponteiros
(Continuação)

Igor Yepes

Ponteiros vs vetores

```
int vet[3] = {1, 2, 3,};
```

0	1	2
1	2	3



Saber o tamanho em bytes de uma variável x:
sizeof(x)

Ponteiros vs vetores

```
int vet[10] = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100};
```

0	1	2	3	4	5	6	7	8	9
10	20	30	40	50	60	70	80	90	100

vet[0] ——> Conteúdo do primeiro elemento do vetor
&vet[0] ——> Endereço na memória de uma posição do vetor
vet ——> Endereço na memória onde inicia o armazenamento do vetor

Ponteiros vs vetores

```c
#include<stdio.h>

void mudar(int vet[]) {
    vet[0]+=20;
}

int main (void) {
    int vet[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
    printf("\nEnd. inic. Vetor:  %p", vet);
    printf("\nTam. Vetor:        %lu", sizeof(vet));
    printf("\nEnd. Vetor:        %p", &vet);
    printf("\nEnd. Elem. 0:      %p", &vet[0]);
    printf("\nCont. Elem. 0:     %i", vet[0]);
    printf("\nCont. Elem. 0:     %i", *vet);
    printf("\nCont. Elem. 1:     %i", *(vet+1));
    printf("\nCont. Elem. 9:     %i", *(vet+9));

    mudar(vet);
    printf("\nCont. Elem. 0:     %i", vet[0]);
}
```

```
End. inic. Vetor:  0x7ffee9daa690
Tam. Vetor:        40
End. Vetor:        0x7ffee9daa690
End. Elem. 0:      0x7ffee9daa690
Cont. Elem. 0:     1
Cont. Elem. 0:     1
Cont. Elem. 1:     2
Cont. Elem. 9:     10
Cont. Elem. 0:     21
```

Ponteiros vs structs

```c
#include<stdio.h>

typedef struct funcionario {
    int cod;
    float salario;
}tFuncionario;

int main (void) {
    tFuncionario func1;

    printf("Digite o código: ");
    scanf("%d", &func1.cod);
    printf("Digite o salário: ");
    scanf("%f", &func1.salario);
    printf("\nCódigo: %d Salário: %.2f\n", func1.cod, func1.salario);
}
```

```
Digite o código: 1234
Digite o salário: 1000

Código: 1234 Salário: 1000.00
```

Ponteiros vs structs

```c
#include<stdio.h>

typedef struct funcionario {
    int cod;
    float salario;
}tFuncionario;

void alteraSalario(tFuncionario *func1) {
    (*func1).salario *= 1.2;  //func1->salario *= 1.2;
}

int main (void) {
    tFuncionario func1;
    //tFuncionario *pFunc1 = &func1;

    printf("Digite o código: ");
    scanf("%d", &func1.cod);
    printf("Digite o salário: ");
    scanf("%f", &func1.salario);
    printf("\nCódigo: %d Salário: %.2f\n", func1.cod, func1.salario);

    alteraSalario(&func1);

    printf("\nCódigo: %d Novo salário: %.2f\n", func1.cod, func1.salario);
}
```

```
Digite o código: 1234
Digite o salário: 1000.00

Código: 1234 Salário: 1000.00

Código: 1234 Novo salário: 1200.00
```

Se for necessário criar o ponteiro fora da função, ocorre da mesma forma que com uma variável comum.

Note que **func1->salario** é outra forma de acessar o mesmo conteúdo de **(*func1).salario**.

Tal como uma variável normal, basta enviar o endereço da struct para a função.

Valor do novo salário alterado dentro da função.