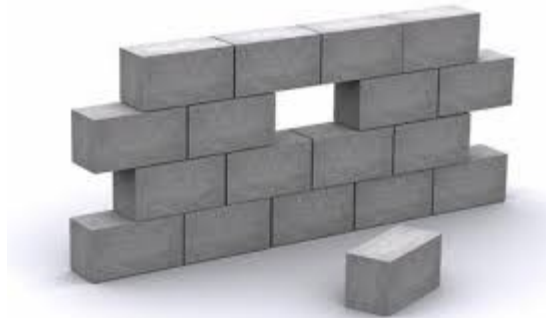




**Instituto Federal**  
Campus Goiânia

**Bacharelado em Sistemas de Informação**

# **Estrutura de Dados I**

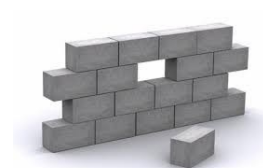


**Prof. Dory Gonzaga Rodrigues**



## Agenda

### Ponteiros












## Ponteiro

### Operações Válidas Sobre Ponteiros

- É válido:

- somar ou subtrair um inteiro a um ponteiro ( $p + \text{int}$  ou  $p - \text{int}$ ) 
- incrementar ou decrementar ponteiros ( $p++$  ou  $p--$ ) 
- subtrair ponteiros (produz um inteiro) ( $p - pi$ ) 
- comparar ponteiros ( $>$ ,  $>=$ ,  $<$ ,  $<=$ ,  $==$ ) 

- Não é válido:

- somar ponteiros ( $pi + pf$ ) 
- multiplicar ou dividir ponteiros ( $p * pi$  ou  $p / pi$ ) 
- operar ponteiros com double ou float ( $p + 2.0$  ou  $p - 2.0$ ) 





## Aritmética de Ponteiros

- É possível fazer operações aritméticas e relacionais entre ponteiros e inteiros
- Soma: ao somar-se um inteiro  $n$  a um ponteiro, endereçamos  $n$  elementos a mais ( $n$  positivo) ou a menos ( $n$  negativo)

$p[2]$	e	$*(p + 2)$	são equivalentes
$*(p + n)$			endereça $n$ elementos a frente
$*(p - n)$			endereça $n$ elementos atrás
$p++$			endereça o próximo elemento de uma array
$p--$			endereça o elemento anterior de uma array



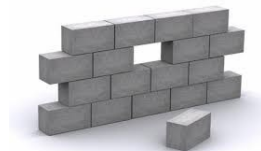


## Ponteiro

### Aritmética de Ponteiros

Exemplo:

```
void main () {  
    int arrayint[] = { 1,2,3,4,5,6,7 };  
    int tamanho = 7;  
    int i, *p;  
    p = arrayint;  
    for (i=0; i < tamanho; i++) {  
        printf(" %d ", *p);  
        p++;  
    }  
}
```





## Ponteiro

### Aritmética de Ponteiros

#### Exemplo

```
void main () {  
    int arrayint[] = { 1,2,3,4,5,6,7 };  
    int tamanho = 7;  
    int i, *p;  
    p = arrayint;  
    printf(" %d ", *p);  
    p += 2;  
    printf(" %d ", *p);  
    p += 2;  
    printf(" %d ", *p);  
}
```





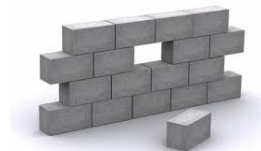
## Ponteiro

### Ponteiros Genéricos

- Um ponteiro genérico é um ponteiro que pode apontar para qualquer tipo de dado;
- Declaramos um ponteiro genérico utilizando-se o tipo **void**:

```
int x;  
float y;  
void *p;
```

```
x=10;  
y=4.0;  
p = &x;  
p = &y;
```





## Ponteiro

### Ponteiros Genéricos

- O tipo de dado apontado por um **void pointer** deve ser controlado pelo usuário;
- Usando um **type cast** (conversão de tipo) o programa pode tratar adequadamente o ponteiro;

```
int x;  
float y;  
void *p;
```

```
p = &x;  
printf("Inteiro: %d\n", *(int *)p);  
p = &y;  
printf("Real: %f\n", *(float *)p);
```







## Ponteiro

### Ponteiros e Strings

- strings são arrays de caracteres e podem ser acessados através de **char \***
- o incremento de **p** o posiciona sobre o próximo caractere (byte a byte)

```
void main () {
```

```
    char str[] = "abcdef";
```

```
    char *p;
```

```
    for (p = str; *p != '\0'; p++)
```

```
        putchar(*p);
```

```
}
```

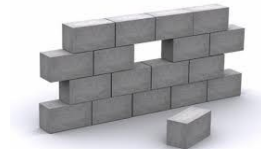
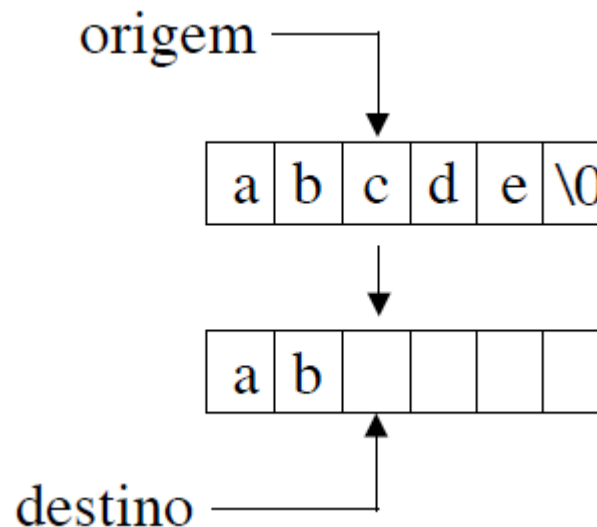




## Ponteiros e Strings

### Exercício

- Faça um programa que leia uma string e depois copie o conteúdo para uma outra variável string. Utilize dois ponteiros para realizar o procedimento.





## Ponteiro

### Ponteiros e Strings

#### Exercício

- Faça um programa que leia uma string e depois copie o conteúdo para uma outra variável string. Utilize dois ponteiros para realizar o procedimento.

```
while (*origem) {  
    *destino=*origem;  
    origem++;  
    destino++;  
}  
*destino='\0';
```





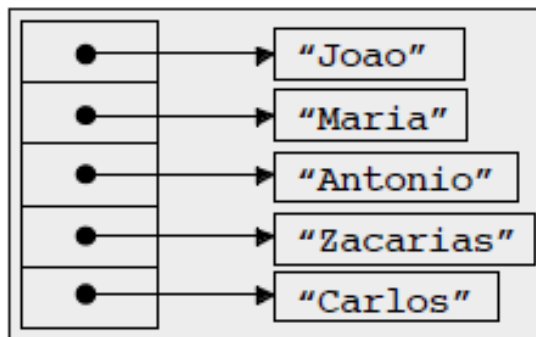
## Ponteiro

### Array de Strings

- Neste caso, cada elemento do array é um ponteiro para um caractere;
- Comparando array de string com matriz de char:

```
char *array_str[] = {"João", "Maria", "Antônio", "Zacarias", "Carlos"};  
char ma[5][10]   = {"João", "Maria", "Antônio", "Zacarias", "Carlos"};
```

Ponteiros (as)



Matriz (ma)

J	o	a	o	\0					
M	a	r	i	a	\0				
A	n	t	o	n	i	o	\0		
Z	a	c	a	r	i	a	s	\0	
C	a	r	l	o	s	\0			



## Ponteiro

### Ponteiro para Ponteiro

- É possível definir ponteiros para ponteiros sem restrição de nível;

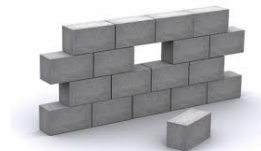
Ex:

```
char *p;  
char **pp;
```

```
p = "teste";  
pp = &p;
```

```
putchar(**pp);
```

- Qual o resultado acima ?

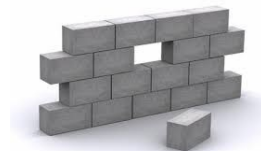




## Ponteiro

### Resumo

- 1) Um ponteiro é um endereço de memória. Qualquer coisa armazenada na memória do computador tem um endereço e este endereço é um ponteiro constante;
- 2) Ponteiro variável é um lugar na memória que armazena o endereço de uma variável.
- 3) Podemos encontrar o endereço de variáveis usando o operador de endereço &
- 4) Se um ponteiro variável p contem o endereço de uma variável x, dizemos que p aponta para x, ou que x é a variável apontada por p;

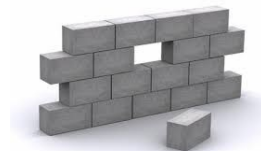




## Ponteiro

### Resumo

- 5) Os ponteiros variáveis são declarados usando asterisco (\*) que significa ponteiro para...
- 6) Todo ponteiro deve ter o tipo da variável apontada. O tipo deve sempre ser especificado na declaração do ponteiro para que o compilador possa executar operações aritméticas corretamente com o ponteiro.
- 7) O operador indireto (\*) pode ser usado com ponteiros para obter o conteúdo da variável apontada por ele ou para executar qualquer operação na variável apontada.





## Ponteiro

### Resumo

8) Os ponteiros acessam os elementos de vetor/matrizes por meio de colchetes, ou por meio da notação ponteiro, com asterisco. Exemplo: `M[1]` ou `*(m+1)`

9) O endereço de uma matriz é um ponteiro constante.

10) Uma cadeia de caracteres constante pode ser definida como uma matriz ou como um ponteiro.

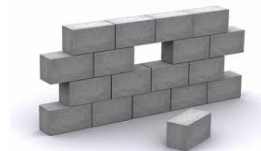
Exemplo:

`Char texto[] = "exemplo" //é um ponteiro constante`

`Char *texto = "exemplo" //é um ponteiro variável`

11) Um ponteiro pode apontar para outro ponteiro.

Este tipo de variável é declarada usando duas vezes o asterisco (\*\*)







## Ponteiro

### Exercícios

- 1) Um ponteiro é:
  - a) o endereço de uma variável
  - b) uma variável que armazena endereços
  - c) o valor de uma variável
  - d) um indicador da próxima variável a ser acessada
  
- 2) Escreva uma instrução que imprima o endereço da “variável x”
  
- 3) A instrução `int *p`
  - a) cria um ponteiro constante do tipo `int`
  - b) cria um ponteiro com valor zero
  - c) cria um ponteiro que aponta para uma variável `int`
  - d) cria um ponteiro do tipo `int`





## Ponteiro

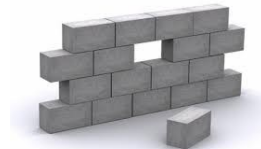
### Exercícios

4) o que significa o operador \* em cada um dos casos abaixo:

- a) `int *p;`
- b) `*p = x*5;`
- c) `printf("%d",*p);`
- d) `printf("%c", *(p+1) );`

5) Qual das seguintes instruções declara(m) um ponteiro para uma variável float?

- a) `float *p;`
- b) `*float p;`
- c) `float* p;`
- d) `*p float;`





## Ponteiro

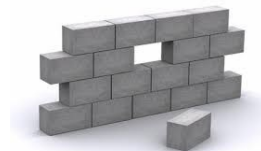
### Exercícios

6) Na expressão `int *p`, o que é do tipo `int`?

- a) a variável `p`;
- b) o endereço de `p`;
- c) a variável apontada por `p`
- d) o endereço da variável apontada por `p`

7) Se o endereço da variável “`x`” foi atribuído a um ponteiro variável “`p`”, qual(quais) das seguintes expressões são verdadeiras ?

- a) `x == &p`;
- b) `x == *p`;
- c) `p == *x`;
- d) `p == &x`;





## Ponteiro

### Exercícios

8) Assuma as declarações abaixo e indique qual é o valor das seguintes expressões:

```
int i=3, j=5;
```

```
int *p=&i, *q=&j;
```

a) `p == &i;` Valor: \_\_\_\_\_

b) `*p - *q;` Valor: \_\_\_\_\_

c) `**&q;` Valor: \_\_\_\_\_

d) `2 - *p / *q + 5` Valor: \_\_\_\_\_

9) Qual é a saída deste programa ?

```
main() {  
    int i=5, *p;  
    p = &i;  
    printf("%p",p);  printf("%d",*p+2);  printf("%d",**&p); }
```





## Exercícios

10) Anote C(certo) e E(errado) para as seguintes expressões de atribuição ?

```
int i, j;  
int *p, *q;
```

`p = &i` R: \_\_\_\_\_

`p = &*&i` R: \_\_\_\_\_

`i = *&*&j` R: \_\_\_\_\_

`i = (*p)++` R: \_\_\_\_\_

`*q = &j` R: \_\_\_\_\_

`i = (*&)j` R: \_\_\_\_\_

`q = &p` R: \_\_\_\_\_

`if( p == i ) i++` R: \_\_\_\_\_

11) Qual a diferença entre `m[3]` e `*(m+3)`?

12) Admitindo a declaração `int m[10]`, porque a instrução `m++` é incorreta?





## Ponteiro

### Exercícios

13) Qual a diferença entre as duas instruções abaixo:

```
char s[] = "texto";  
char *s = "texto";
```

14) Temos a seguinte variável declarada abaixo, o que será impresso?

```
char *s = "eu estou estudando ponteiro no cti";
```

<pre>printf("%s",s);</pre>	Resposta: _____
<pre>printf("%p",&amp;s[0]);</pre>	Resposta: _____
<pre>printf("%s",(s+11));</pre>	Resposta: _____
<pre>printf("%c",s[4]);</pre>	Resposta: _____





## Exercícios

15) Qual o resultado a ser impresso no comando printf ?

```
int x=3;  
int *p, **p;  
p = &x;  
pp = &p;  
printf("\n valor a ser impresso=%d", **p);
```

Resposta: \_\_\_\_\_

