

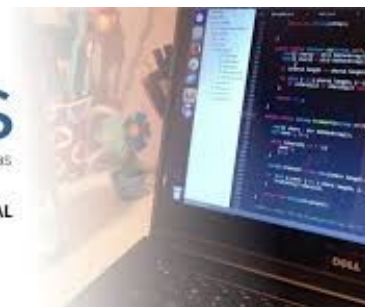
# *Tecnologia em Análise e Desenvolvimento de Sistemas - TADS*

## ***Estrutura de Dados I***

*Atividades Pedagógicas Não Presenciais – APNP 2020*

***Prof. Luciano Vargas Gonçalves***

*E-mail: [luciano.goncalves@riogrande.ifrs.edu.br](mailto:luciano.goncalves@riogrande.ifrs.edu.br)*



# Estrutura da Dados

- **Aula 2 – Ponteiros em Programação em C**

# Ponteiros em C

- Os conceitos de endereço de memória e ponteiro são fundamentais em qualquer linguagem de programação, embora fiquem ocultos em algumas linguagens.
- Em C, esses conceitos são explícitos. O conceito de ponteiro não é fácil; é preciso fazer algum esforço para dominá-lo.
  - Endereços:
    - A memória RAM de qualquer computador é uma sequência de bytes. Cada byte armazena um conjunto de 8 bits;
    - Os bytes são numerados sequencialmente. O número (endereço) de um byte é o seu **endereço (= address)**.
      - ***Similar ao número de uma casa em uma rua qualquer;***

# Ponteiros em C

- Tipos de dados e Espaço ocupado
  - Um char ocupa 1 byte.
  - Um int ocupa 4 bytes em alguns computadores e 8 em outros (o número exato é dado pela expressão sizeof (int)).
  - Um double ocupa usualmente 8 bytes (o número exato é dado pela expressão sizeof (double)).
  - O endereço de um objeto é o endereço do seu primeiro byte.

```
TestesC
int ocupa 4
float ocupa 4
char de 10 ocupa 10
double ocupa 8

Process returned 0 (0x0)   execution time : 0.001 s
Press ENTER to continue.
[]
```

# Ponteiros em C

- Um ponteiro **é uma variável que armazena um endereço de memória;**
  - Declaração de um Ponteiro;
    - Usa o operador “ \* ” ;

```
int *pi; //ponteiro inteiro
char *pc; //ponteiro de char
double *pd; //ponteiro de double
```

Memória RAM

Endereço	Valor
FFAE001	
FFAE010	
FFAE011	
FFAE100	

# Ponteiros em C

- Atribuição de um endereço de memória
  - Usamos o operador “&”

```
int vl = 10; //variável inteiro
```

```
pi = &vl; //copia do endereço vl
```

**vl = &pi**

**vl**

Ocupa  
4 Bytes

Memória RAM

Endereço	Valor
5CF5B50C	10
5CF5B50D	
5CF5B50E	
5CF5B50F	

**vl = \*pi**

# Ponteiros em C

- Atribuição de um endereço de memória
  - Usamos o operador “&”

```
int vl = 10; //variável inteiro  
pi = &vl; //copia do endereço vl
```

$vl = \&pi$

Memória RAM

Endereço	Valor
5CF5B50C	10
5CF5B50D	
5CF5B50E	
5CF5B50F	

Representam a mesma informação  $\rightarrow vl = *pi$

# Ponteiros em C

```
int *pi;    //ponteiro inteiro
char *pc;   //ponteiro de char
double *pd; //ponteiro de double
```

```
int vl = 10; //variável inteiro
pi = &vl;   //copia do endereço v
```

```
printf("Valor vl=%d o endereço de vl=%x\n", vl, &vl);
printf("Valor vl=%d o endereço de vl=%x\n", vl, pi);
```

vl = &pi

Memória RAM

Endereço	Valor
5CF5B50C	10
5CF5B50D	
5CF5B50E	
5CF5B50F	

```
TestesC
Valor vl=10 o endereço de vl=5cf5b50c
Valor vl=10 o endereço de vl=5cf5b50c

Process returned 0 (0x0)   execution time : 0,001 s
Press ENTER to continue.
```



# Ponteiros em C – Operador “\*”

- Operador “\*” consulta o valor do ponteiro

```
int *pi;    //ponteiro inteiro
```

```
int vl = 10; //variável inteiro
```

```
pi = &vl; //copia do endereço v
```

```
//ponteiro e ponteiro(endereço)
```

```
printf("Valor vl=%d o endereço de vl=%x\n", *pi, pi);
```

```
TestesC
Valor vl=10 o endereço de vl=5cf5b50c
Valor vl=10 o endereço de vl=5cf5b50c

Process returned 0 (0x0)   execution time : 0,001 s
Press ENTER to continue.
```

Memória RAM

Endereço	Valor
5CF5B50C	10
5CF5B50D	
5CF5B50E	
5CF5B50F	

vl = &pi

vl = \*pi

# Ponteiros em C

- O segundo argumento da função de biblioteca scanf é o endereço da posição na memória onde devem ser depositados os objetos lidos do dispositivo padrão de entrada:

```
int i;  
scanf ("%d", &i);
```

# Ponteiros

- Um ponteiro (= apontador = pointer) é um tipo especial de variável que armazena endereços.
- Um ponteiro pode ter o valor especial - *NULL (zero)*
  - Se um ponteiro **p** armazena o endereço de uma variável **i**, podemos dizer “p aponta para i ou p é o endereço de i”.
  - Se um ponteiro **p** tem valor diferente de NULL ,então *\*p é o valor do objeto apontado por p.*
    - *Exemplo:*
      - Por exemplo, se “i” é uma variável e “p” é igual a “&i” então dizer “\*p” é o mesmo que dizer “i”.
      - Sem usar o nome “i”

# Ponteiros em C

- Há vários tipos de ponteiros:
  - ponteiros para caracteres, para inteiros, para registros etc.
  - O computador precisa saber de que tipo de ponteiro você está falando.
  - Para declarar um ponteiro “\*p” para um inteiro, diga
    - `int *p;`
  - Para declarar um ponteiro “\*p” para um registro cel, diga
    - `struct cel *p;`
  - Um ponteiro “\*r” para um ponteiro que apontará um inteiro é declarado assim:
    - `int **r;`

# Ponteiros em C

- Exemplo 1:
  - Suponha que a, b e c são variáveis inteiras. Eis um jeito de fazer  $c = a+b$ :

```
int *p, *q;    /* p é um ponteiro para um inteiro */  
int a=10, b=5, c;  
p = &a;        /* o valor de p é o endereço de a */  
q = &b;        /* q aponta para b */  
c = *p + *q;
```

# Ponteiros em C

- Exemplo 2:

```
int *pa,*pb, **r; /* r é um ponteiro para um ponteiro para um inteiro */
int a=10, b=5, c;
pa = &a;          /* p aponta para a */
pb = &b;          /* p aponta para b */
r = &pa;          /* r aponta para p e *r aponta para a */
c = **r + b;

printf("Resultado de %d + %d = %d\n",*pa,*pb,c);
```

Ponteiro (r)	→	Ponteiro (p)	→	variável (a)
**r	→	*p	→	a

# Ponteiros em C

- Suponha que precisamos de uma função que troque os valores de duas variáveis inteiras, digamos i e j.
- Sem Ponteiros

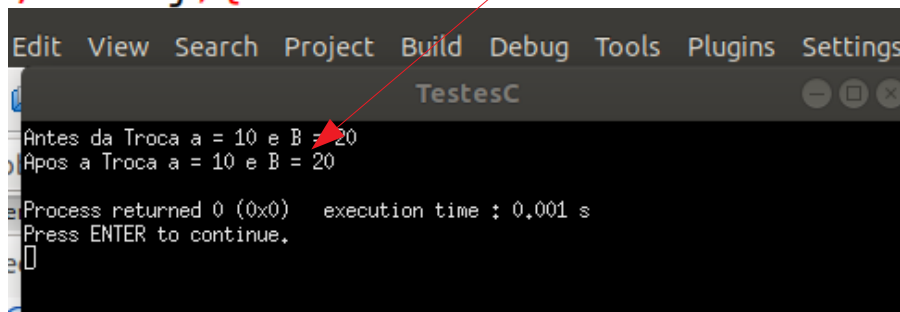
```
void troca (int i, int j)
{
    int temp;
    temp = i;
    i = j;
    j = temp;
}
```

# Ponteiros em C

- Suponha que precisamos de uma função que troque os valores de duas variáveis inteiras, digamos i e j.

```
int a = 10, b = 20;
printf("Antes da Troca a = %d e B = %d\n", a, b);
trocaSimples(a, b);
printf("Apos a Troca a = %d e B = %d\n", a, b);
return 0;
}
```

```
void trocaSimples (int i, int j){
    int temp;
    temp = i;
    i = j;
    j = temp;
}
```

A screenshot of a C program execution in a terminal window titled 'TestesC'. The program prints 'Antes da Troca a = 10 e B = 20' and 'Apos a Troca a = 10 e B = 20', indicating that the values of 'a' and 'b' were not swapped. The terminal also shows 'Process returned 0 (0x0) execution time : 0.001 s' and 'Press ENTER to continue.' A red arrow points from the text on the right to the output of the program.

```
Edit View Search Project Build Debug Tools Plugins Settings
TestesC
Antes da Troca a = 10 e B = 20
Apos a Troca a = 10 e B = 20
Process returned 0 (0x0) execution time : 0.001 s
Press ENTER to continue.
```

*/\*ERRO\*/*

Não produz o efeito desejado, pois recebe apenas os valores das variáveis e não as variáveis propriamente ditas. Passagem de parâmetro por valor



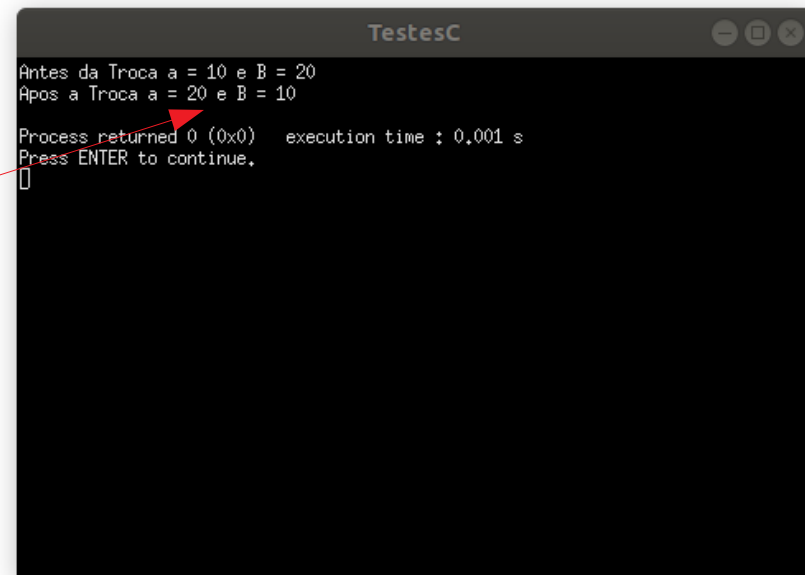
# Ponteiros em C

- Suponha que precisamos de uma função que troque os valores de duas variáveis inteiras, digamos i e j.

```
int a = 10, b = 20;  
printf("Antes da Troca a = %d e B = %d\n", a, b);  
trocaPonteiros(&a, &b);  
printf("Apos a Troca a = %d e B = %d\n", a, b);  
return 0;
```

```
void trocaPonteiros (int *i, int *j){  
    int temp;  
    temp = *i; //copia da informação em *p  
    *i = *j;    //copia da informação em *q  
    *j = temp;  //copia da informação em temp  
}
```

**Sucesso \*/**  
**Passagem de**  
**parâmetro por**  
**REFERÊNCIA**



```
TestesC  
Antes da Troca a = 10 e B = 20  
Apos a Troca a = 20 e B = 10  
Process returned 0 (0x0) execution time : 0.001 s  
Press ENTER to continue.
```

Sucesso – Troca Ok

# Ponteiros em C

- Exercício 1
  - Por que o código abaixo está errado?

```
void troca (int *i, int *j) {  
    int *temp;  
    *temp = *i;  
    *i = *j;  
    *j = *temp;  
}
```

**Temp agora é ponteiro;  
Qual o problema na troca?**

# Ponteiros em C

- Exercício 1

- Por que o código abaixo está errado?

```
void troca (int *i, int *j) {  
    int *temp;  
    *temp = *i;  
    *i = *j;  
    *j = *temp;  
}
```

**Temp agora é ponteiro;  
Temp não recebeu o endereço  
para apontar;  
Temp está NULL;**

# Ponteiros em C

- Exercício 1
  - Por que o código abaixo está errado?

```
void troca (int *i, int *j) {  
    int c;  
    int *temp;  
    temp = &c;  
    *temp = *i;  
    *i = *j;  
    *j = *temp;  
}
```

**Temp usa C como auxiliar para armazenar**

# Ponteiros em C

- Exercício 2
  - Um ponteiro pode ser usado para dizer a uma função onde ela deve depositar o resultado de seus cálculos. Escreva uma função **Converte** que converta minutos em horas-e-minutos.
    - A função recebe um inteiro ***mnts*** e os endereços de duas variáveis inteiras, digamos ***h*** e ***m***, e atribui valores a essas variáveis de modo que ***m*** seja menor que **60** e **que  $60 * h + m$  seja igual a *mnts***. Escreva também uma função main que use a função **Converte**.

# Ponteiros em C

- Exercício 3
  - Escreva uma função ***maiorMenor*** que receba um vetor inteiro ***v[0..n-1]*** e os endereços de duas variáveis inteiras, digamos ***min*** e ***max***, e deposite nessas variáveis o valor de um elemento ***mínimo*** e o valor de um elemento ***máximo*** do vetor.
    - Escreva também uma função main que use a função mm.