

# Aula 2: Funcionamento da Memória e Ponteiros



DCC302-Estrutura de Dados I  
Prof. Me. Acauan C. Ribeiro

# Variáveis e Memória

Na linguagem C, cada **variável** está associada a:

- 1) Um nome;
- 2) Um tipo;
- 3) Um valor;
- 4) Um endereço.

# Variáveis e Memória



Fazer teste:  
CPU-Z e Benchmark

Na linguagem C, cada **variável** está associada a:

- 1) Um nome;
- 2) Um tipo;
- 3) Um valor;
- 4) Um endereço.

Memória RAM		
Endereço	Valor	Nome

# Variáveis e Memória

Fazer teste:  
CPU-Z e Benchmark



RAM? SDRAM? DDR? Ciclo? Frequência? Latência?

# Variáveis e Memória

Na linguagem C, cada **variável** está associada a:

- 1) Um nome;
- 2) Um tipo;
- 3) Um valor;
- 4) Um endereço.

```
int a = 10;  
int b, c;  
  
b = 20;  
c = a + b;
```

Memória RAM		
Endereço	Valor	Nome
8888	####	
...		
3028	####	
3024	####	
3020	10	a

# Variáveis e Memória

Na linguagem C, cada **variável** está associada a:

- 1) Um nome;
- 2) Um tipo;
- 3) Um valor;
- 4) Um endereço.

```
int a = 10;  
int b, c;  
  
b = 20;  
c = a + b;
```

Memória RAM		
Endereço	Valor	Nome
8888	####	
...		
3028	####	
3024	####	
3020	10	a



```
&a = 3020; a = 10
```



Little-endian

Big-endian

Ver:

[http://carlosdelfino.eti.br/programacao/cplusplus/Diferencas\\_entre\\_BigEndian\\_Little\\_Endian\\_e\\_Bit\\_Endianness/](http://carlosdelfino.eti.br/programacao/cplusplus/Diferencas_entre_BigEndian_Little_Endian_e_Bit_Endianness/)

# Variáveis e Memória

Na linguagem C, cada **variável** está associada a:

- 1) Um nome;
- 2) Um tipo;
- 3) Um valor;
- 4) Um endereço.

```
int a = 10;  
int b, c;  
  
b = 20;  
c = a + b;
```

Memória RAM		
Endereço	Valor	Nome
8888	####	
...		
3028	####	c
3024	####	b
3020	10	a

```
&a = 3020; a = 10  
&b = 3024; b = ####  
&c = 3028; c = ####
```

# Variáveis e Memória

Na linguagem C, cada **variável** está associada a:

- 1) Um nome;
- 2) Um tipo;
- 3) Um valor;
- 4) Um endereço.

```
int a = 10;  
int b, c;  
  
b = 20;  
c = a + b;
```

"&" endereço de memória

Memória RAM		
Endereço	Valor	Nome
8888	####	
...		
3028	####	c
3024	####	b
3020	10	a

`&a = 3020; a = 10`  
`&b = 3024; b = ####`  
`&c = 3028; c = ####`

lixo



# Variáveis e Memória

Na linguagem C, cada **variável** está associada a:

- 1) Um nome;
- 2) Um tipo;
- 3) Um valor;
- 4) Um endereço.

```
int a = 10;  
int b, c;  
  
b = 20;  
c = a + b;
```

Memória RAM		
Endereço	Valor	Nome
8888	####	
...		
3028	####	c
3024	20	b
3020	10	a



```
&a = 3020; a = 10  
&b = 3024; b = 20  
&c = 3028; c = ####
```

lixo



# Variáveis e Memória

Na linguagem C, cada **variável** está associada a:

- 1) Um nome;
- 2) Um tipo;
- 3) Um valor;
- 4) Um endereço.

```
int a = 10;  
int b, c;  
  
b = 20;  
c = a + b;
```

Memória RAM		
Endereço	Valor	Nome
8888	####	
...		
3028	30	c
3024	20	b
3020	10	a



```
&a = 3020; a = 10  
&b = 3024; b = 20  
&c = 3028; c = 30
```



# Let's code

**&var:** endereço da variável **var**

**%p:** flag para imprimir **endereços**

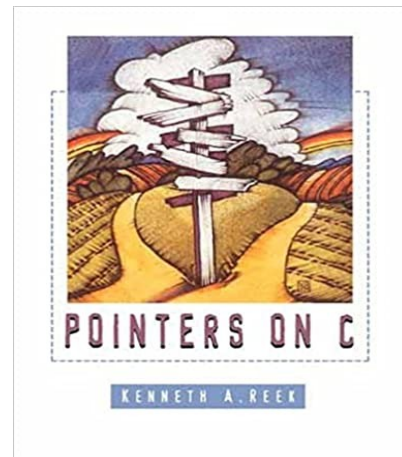
```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main() {
5      int a = 10;
6      int b, c;
7
8      printf("&a = %p, a = %d\n", &a, a);
9      printf("&b = %p, b = %d\n", &b, b);
10     printf("&c = %p, c = %d\n\n", &c, c);
11
12     b = 20;
13     c = a + b;
14
15     printf("&a = %p, a = %d\n", &a, a);
16     printf("&b = %p, b = %d\n", &b, b);
17     printf("&c = %p, c = %d\n\n", &c, c);
18
19     return 0;
20 }
```

# Ponteiros

- O que é um ponteiro?

# Ponteiros

- O que é um ponteiro?
  - É um tipo de dado, que ao invés de armazenar um inteiro, um caractere, um float, ele vai armazenar um **ENDEREÇO DE MEMÓRIA**.
  - Ou seja, um ponteiro é uma variável (especial) que armazena um **endereço de memória** de outra **variável** de um determinado **tipo**.



Um apontador é uma variável que contém o endereço de outra variável. Apontadores são muito usados em C, em parte porque eles são, às vezes, a única forma de se expressar uma computação, e em parte porque eles normalmente levam a um código mais compacto e eficiente que o obtido de outras formas.

Apontadores têm sido comparados ao comando goto como uma forma maravilhosa de se criar programas impossíveis de entender. Isto é certamente verdade quando eles são usados sem cuidado, e é fácil criar apontadores que apontem para algum lugar inesperado. Com disciplina, entretanto, apontadores podem ser usados para se obter clareza e simplicidade. Este é o aspecto que tentaremos ilustrar.

Livro: The C Programming Language - Brian Kernighan e Dennis Ritchie

# Ponteiros

Tipo: Jogador  
Seleção  
Brasileira



Seleção Brasileira



Nome: Neymar = 10

Endereço: Av. Ney – Paris

nome\_variável = Neymar

&Neymar = Av. Ney – Paris

FedEx<sup>®</sup>  
Express



Tipo:  
Entregador  
(ponteiro)



Nome: Jaime

Endereço: Rua Jaime – Bogotá

nome\_variável = Jaime

&Neymar = Rua Jaime – Bogotá

# Ponteiros

Tipo: Jogador  
Seleção  
Brasileira



Seleção Brasileira



Nome: Neymar = 10

Endereço: Av. Ney – Paris

nome\_variável = Neymar

&Neymar = Av. Ney – Paris



Tipo:  
Entregador  
(ponteiro)



FedEx<sup>®</sup>  
Express



Nome: Jaime

Endereço: Rua Jaime – Bogotá

nome\_variável = Jaime

&Neymar = Rua Jaime – Bogotá

# Ponteiros

Tipo: Jogador  
Seleção  
Brasileira



Seleção Brasileira



Nome: Neymar = 10

Endereço: Av. Ney – Paris

nome\_variável = Neymar

&Neymar = Av. Ney – Paris

Tipo:  
Entregador  
(Jogador  
Seleção  
Brasileira)



FedEx®  
Express



Nome: Jaime = &Neymar

Endereço: Rua Jaime – Bogotá

nome\_variável = Jaime

&Neymar = Rua Jaime – Bogotá



# Ponteiros

Seleção Brasileira



Tipo: Jogador  
Seleção  
Brasileira



Nome: Neymar = 10

Endereço: Av. Ney – Paris

nome\_variável = Neymar

&Neymar = Av. Ney – Paris



Tipo:  
Entregador  
(Jogador  
Seleção  
Brasileira)

FedEx<sup>®</sup>  
Express



Nome: Jaime =

Endereço: Rua Jaime – Bogotá

nome\_variável = Jaime

&Jaime = Rua Jaime – Bogotá

Tipo: Jogador  
Seleção  
Portuguesa



Nome: Cristiano = 7

Endereço: Rua Cris – Inglaterra

nome\_variável = Neymar

&Cristiano = Rua Cris – Inglaterra

# Ponteiros

Seleção Brasileira



Tipo: Jogador  
Seleção  
Brasileira



Nome: Neymar = 10

Endereço: Av. Ney – Paris

nome\_variável = Neymar

&Neymar = Av. Ney – Paris

Tipo: Jogador  
Seleção  
Portuguesa



Nome: Cristiano = 7

Endereço: Rua Cris – Inglaterra

nome\_variável = Neymar

&Cristiano = Rua Cris – Inglaterra

FedEx<sup>®</sup>  
Express



Nome: Jaime =

Endereço: Rua Jaime – Bogotá

nome\_variável = Jaime

&Jaime = Rua Jaime – Bogotá

Tipo:  
Entregador  
(Jogador  
Seleção  
Brasileira)

# Ponteiros

Seleção Brasileira



Tipo: Jogador  
Seleção  
Brasileira



Nome: Neymar = 10

Endereço: Av. Ney – Paris

nome\_variável = Neymar

&Neymar = Av. Ney – Paris

Tipo:  
Entregador  
(Jogador  
Seleção  
Brasileira)



Nome: Jaime =

Endereço: Rua Jaime – Bogotá

nome\_variável = Jaime

&Jaime = Rua Jaime – Bogotá

Chegou  
correspondência  
para o Jaime!!

FedEx<sup>®</sup>  
Express



Tipo:  
Entregador  
(ponteiro)



Nome: Zé

Endereço: Rua Zéferino – China

nome\_variável = Zé

&Zé = Rua Zéferino – China

# Ponteiros

Seleção Brasileira



Tipo: Jogador  
Seleção  
Brasileira



Nome: Neymar = 10

Endereço: Av. Ney – Paris

nome\_variável = Neymar

&Neymar = Av. Ney – Paris

Tipo:  
Entregador  
(Jogador  
Seleção  
Brasileira)



Nome: Jaime =

Endereço: Rua Jaime – Bogotá

nome\_variável = Jaime

&Jaime = Rua Jaime – Bogotá

Chegou  
correspondência  
para o Jaime!!

FedEx<sup>®</sup>  
Express



Tipo:  
Entregador  
(Entregador  
Ponteiro)



Nome: Zé = &Jaime

Endereço: Rua Zéferino – China

nome\_variável = Zé

&Zé = Rua Zéferino – China

## OPERADORES

& : retorna endereço de memória

\* : acessa o conteúdo que está apontando  
("executa" o ponteiro – recupera valor da variável  
que esta apontando) - \*indireção

\* <https://docs.microsoft.com/pt-br/cpp/c-language/indirection-and-address-of-operators?view=msvc-170>

# Ponteiro – Exemplo mais real

- Um ponteiro nada mais é do que uma variável que guarda o endereço de uma outra variável;

```
int a = 10;  
int *p1 = NULL;  
int *p2;  
  
p1 = &a;  
p2 = p1;  
*p2 = 4;
```

Completar

&a=\_\_\_\_ a=\_\_\_\_  
&p1=\_\_\_\_ p1=\_\_\_\_ \*p1=\_\_\_\_  
&p2=\_\_\_\_ p2=\_\_\_\_ \*p2=\_\_\_\_

Memória RAM

Endereço	Valor	Nome
8888		
...		
5012		
5004		
5000		

# Ponteiro – Exemplo mais real

Todo ponteiro (i.e, endereços de memória) ocupam 8bytes  
(arquiteturas modernas)

- Um ponteiro nada mais é do que uma variável que guarda o endereço de uma outra variável;



```
int a = 10;  
int *p1 = NULL;  
int *p2;  
  
p1 = &a;  
p2 = p1;  
*p2 = 4;
```

&a=5000      a=10  
&p1=5004      p1=NULL  
&p2=5012      p2=###


Um ponteiro pode ter o valor especial NULL  
que não é o endereço de lugar algum.  
A constante NULL está definida em `stdlib.h`  
e seu valor é 0 na maioria dos computadores.

Memória RAM		
Endereço	Valor	Nome
8888		
...		
5012	###	p2
5004	NULL	p1
5000	10	a

# Ponteiro – Exemplo mais real

Todo ponteiro (i.e, endereços de memória) ocupam 8bytes  
(arquiteturas modernas)

- Um ponteiro nada mais é do que uma variável que guarda o endereço de uma outra variável;




```
int a = 10;
int *p1 = NULL;
int *p2;

p1 = &a;
p2 = p1;
*p2 = 4;
```

&a=5000      a=10  
&p1=5004      p1=5000  
&p2=5012      p2=###


Memória RAM		
Endereço	Valor	Nome
8888		
...		
5012	###	p2
5004	5000	p1
5000	10	a



# Ponteiro – Exemplo mais real

Todo ponteiro (i.e, endereços de memória) ocupam 8bytes  
(arquiteturas modernas)

- Um ponteiro nada mais é do que uma variável que guarda o endereço de uma outra variável;




```
int a = 10;
int *p1 = NULL;
int *p2;

p1 = &a;
p2 = p1;
*p2 = 4;
```

&a=5000      a=10  
&p1=5004      p1=5000  
&p2=5012      p2=5000

Memória RAM		
Endereço	Valor	Nome
8888		
...		
5012	5000	p2
5004	5000	p1
5000	10	a





# Ponteiro – Exemplo mais real

Todo ponteiro (i.e, endereços de memória) ocupam 8bytes  
(arquiteturas modernas)

- Um ponteiro nada mais é do que uma variável que guarda o **endereço** de uma outra variável;

```
int a = 10;  
int *p1 = NULL;  
int *p2;  
  
p1 = &a;  
p2 = p1;  
*p2 = 4;
```

&a=5000    a=4  
&p1=5004    p1=5000    \*p1=4  
&p2=5012    p2=5000    \*p2=4

Altera o valor da variável  
pela referência de memória  
armazenada em p2.

\*p2 = 4 → \*(5000) = 4  
“a = 10” → a = 4

Memória RAM		
Endereço	Valor	Nome
8888		
...		
5012	5000	p2
5004	5000	p1
5000	4	a

# Ponteiros – Prática em sala



Fazer...

# Ponteiros – Prática em sala



```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main() {
5
6      int a = 10;
7      int *p1 = NULL;
8      int *p2; //uma boa prática é sempre declarar uma variável e atribuir um valor (instanciar)
9
10     printf("&a = %p, a = %d\n", &a, a);
11     printf("&p1 = %p, p1 = %p\n", &p1, p1);
12     printf("&p2 = %p, p2 = %p\n\n", &p2, p2);
13
14     p1 = &a;
15     p2 = p1;
16     *p2 = 4;
17
18     printf("&a = %p, a = %d\n", &a, a);
19     printf("&p1 = %p, p1 = %p, *p1 = %d\n", &p1, p1, *p1);
20     printf("&p2 = %p, p2 = %p, *p2 = %d\n", &p2, p2, *p2);
21
22     return 0;
23 }
```

`p1 = &p2`

Vai funcionar? Ou Vai dar erro  
ou warn?

# Prática em sala



1) Implementar a função `troca(int v1, int v2)`. A função troca deve receber dois valores inteiros e realizar a troca de seus conteúdos. A função **`troca()`** deve estar fora do escopo da função **`main()`**.

Ex.:

```
int a = 10;  
int b = 999;  
troca(a, b);  
printf("a = %d    b = %d\n", a, b);
```

Saída:

```
a = 999    b = 10
```

# Ponteiros – Exemplos de sizeof()

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main() {
5      int a;
6
7      printf("sizeof(a) = %ld bytes\n", sizeof(a));
8      printf("sizeof(int) = %ld bytes\n", sizeof(int));
9      printf("sizeof(short) = %ld bytes\n", sizeof(short));
10     printf("sizeof(long) = %ld bytes\n", sizeof(long));
11     printf("sizeof(float) = %ld bytes\n", sizeof(float));
12     printf("sizeof(double) = %ld bytes\n\n", sizeof(double));
13
14     printf("sizeof(void *) = %ld bytes\n", sizeof(void *));
15     printf("sizeof(int *) = %ld bytes\n", sizeof(int *));
16     printf("sizeof(int **) = %ld bytes\n", sizeof(int **));
17     printf("sizeof(int ***) = %ld bytes\n", sizeof(int ***));
18     printf("sizeof(float *) = %ld bytes\n", sizeof(float *));
19
20
21     return 0;
22 }
```

# Próxima aula...

- Ponteiros de Ponteiros
- Exercícios

# Referências

- Aulas Estrutura de Dados I – Prof. Samuel Martins – IFSP
- Cap 5 - The C Programming Language – B. Kernighan e D. Ritchie