

# Estrutura de Dados

Prof. Rafael Nunes

# Revisão - Ponteiros

## Parte 1

Por que devemos conhecer o  
funcionamento correto dos  
***Ponteiros*** na Linguagem C?

# Ponteiros - Vantagens

- Permite a passagem de parâmetros por **referência** (o endereço) ao invés de utilizar **cópia** de dados
  - Melhora o **desempenho** - a utilização sensata de ponteiros deixa o programa mais rápido;
- Oferece uma forma elegante de passar **matrizes e strings** como argumentos para funções;
- São a base para a criação de **estruturas de dados avançadas**, como listas encadeadas, pilhas, filas e árvores;

# Como funcionam os Ponteiros?

Escreva em um pedaço de papel o seu endereço residencial... no verso (atrás!) escreva o seu primeiro nome.

<<< 2 minutos >>>

Para compreender melhor o que são ponteiros, vamos relembrar a definição de variável.

O que é uma variável ?

# O que é uma variável ?

- É um **espaço de memória** reservado para guardar alguma coisa, referenciado por um nome.
- Tá bom, **mas que tipo** de coisa uma variável guarda? Depende...

# Variáveis armazenam...

- Os *ints* guardam dados *inteiros*.
- Os *floats* guardam dados *numéricos de ponto flutuante*.
- Os *chars* guardam dados do tipo *caracteres*.



E os ponteiros?

# Ponteiros

- Um ponteiro **é uma variável** que armazena o endereço de memória de outra variável.
- Armazenando o endereço de outra variável, ele pode referenciá-la **indiretamente**.
- Quando um ponteiro contém o endereço de outra variável, nós dizemos que ele **aponta** para essa variável...
- então...

Os ***ponteiros*** guardam  
dados do tipo:

***Endereços de memória***

# Ponteiros - Funcionamento

Ainda se lembram do  
pedaço de papel???

# Ponteiros - Funcionamento

- Ponteiros guardam **endereços de memória**.
- Quando você anota o **endereço de um colega** você está criando um ponteiro.
- O ponteiro é este seu pedaço de papel. Ele tem anotado um endereço.

Qual o sentido disso?

# Qual é o sentido disso?

- **Simples.** Quando você anota o endereço de um colega, depois você vai usar este endereço para *achá-lo*.
- O C funciona assim. Você anota o endereço de algo numa *variável ponteiro* para depois usar.
- Utilizando o mesmo raciocínio, uma agenda, onde são guardados endereços de vários amigos, poderia ser vista como sendo uma *matriz de ponteiros* no C.

Ponteiro possui tipo?



# Ponteiro possui tipo?

- **Sim.** Um ponteiro também tem tipo.
  - Quando você anota um *endereço residencial* de um amigo você o trata diferente de quando você anota o endereço de uma *firma comercial* qualquer.
  - Apesar de o endereço dos dois locais ter o *mesmo formato* (rua, número, bairro, cidade, etc.) eles indicam locais cujos **conteúdos são diferentes**.
  - Então os dois endereços são ponteiros de *tipos diferentes*.

# Ponteiro possui tipo?

- No C quando declaramos ponteiros nós informamos ao compilador para que *tipo de variável* vamos apontá-lo.
- Um *ponteiro int aponta para um dado inteiro*, isto é, guarda o endereço de um inteiro

# Declaração de Ponteiros

# Declaração de Ponteiros

- A forma geral da declaração de ponteiros é a seguinte:

*tipo\_do\_ponteiro* \*nome\_da\_variável;

# Declaração de Ponteiros

- É o asterisco (\*) que faz o compilador saber que aquela variável não vai guardar *um valor* mas sim um *endereço* para *aquele tipo* especificado.
- Ex:

**int**      **\*ptr;**

**char**    **\*temp, \*ptr2;**

# Declaração de Ponteiros

- Para declarar ponteiros, você deve deixar claro para o compilador para *qual tipo* de dado vai apontar.

Portanto, especificar um *tipo* é *obrigatório!*

- Ex:

```
int *ptr;
```

- Este comando declara uma variável ponteiro que aponta para (ou armazena o endereço de) uma variável do *tipo int*. Para qualquer tipo de dado funciona da mesma forma.

# Os operadores \* e &

Para que servem?

# Os operadores \* e &

- São operadores *unários*
  - Lembrem-se dos operadores binários
  - Soma, subtração e etc... Duas variáveis
- São utilizados na *manipulação de ponteiros*



O operador '&'

Alguém pode dar um exemplo?

# O operador '&'

- Muito utilizado na função *scanf()*
- Ex:

```
int num;  
printf("Digite um número: ");  
scanf("%d", &num);
```

- Retorna o *endereço de memória* de uma variável

O operador '\*'

???

# O operador '\*'

- O operador '\*' é chamado de *referência de ponteiros* ou *operador indireto*...
  - Não confundir com o *operador multiplicação* que é binário
- É o inverso do operador '&'
- Serve para obter ou manipular o *valor de uma variável* apontada por um ponteiro
  - Como assim Prof. ???

# O operador ‘\*’

- Quando colocamos o operador na frente de um ponteiro estamos dizendo o seguinte:
  - “Sr. Pontoeiro, eu não quero o endereço da variável que você está armazenando, quero apenas o **conteúdo** que está dentro dela”

- Ex:

```
int num, *ptr;  
num = 25;  
ptr = &num;  
printf(“%p”, ptr);  
Printf(“%d”, *ptr);
```

# O operador ‘\*’

- Voltando a nossa análise com o papel...
- Uma vez de posse do endereço no papel você poderia, por exemplo, fazer uma visita à casa de seu colega.
- No C você *faz uma visita* à casa aplicando o operador \* ao papel.
- Uma vez dentro da casa você pode copiar *seu conteúdo* ou modificá-lo.

# Atribuição entre Ponteiros

O que acontece se atribuirmos  
um ponteiro ao outro?

# Atribuição entre Ponteiros

- Se atribuirmos um ponteiro a outro...

```
int *p, *q;  
int num= 10;  
p= &num;  
q= p;  
printf("%d, %d", *p, *q);
```

- O ponteiro que estiver do lado esquerdo irá receber o endereço do ponteiro do lado direito...
- ou seja... os dois apontarão para o mesmo lugar



# Vamos praticar...

Digite o programa a seguir na  
ferramenta e veja o resultado...

<< 5 minutos >>

# ponteiros\_prg1.c

```
#include <stdio.h>
int j, k;
int *ptr;
int main(void)
{
    j = 1;
    k = 2;
    ptr = &k;

    printf("\n");
    printf("j tem valor %d e esta armazenado no end: %p\n", j, &j);
    printf("k tem valor %d e esta armazenado no end: %p\n", k, &k);
    printf("ptr tem valor %p e esta armazenado no end: %p\n", ptr, &ptr);
    printf("O valor do inteiro apontado por ptr eh %d\n", *ptr);
    return 0;
}
```

# Entendendo o resultado

j tem valor 1 e esta armazenado no end: 00404090

k tem valor 2 e esta armazenado no end: 004040B0

ptr tem valor 004040B0 e esta armazenado no end: 004040A0

O valor do inteiro apontado por ptr eh 2

Até a próxima...

# Referências

- Curso de C do CPDEE/UFMG - 1996-1999. <http://www.cpdee.ufmg.br/cursos/C>
- ...