

Linguagem de Programação

Linguagem de Programação C
Ponteiro

Sumário

- 1 Definição
- 2 Declaração
- 3 Atribuição
- 4 Operadores

- 5 Referência e Valor
- 6 Vetor
- 7 Aritmética de Ponteiros
- 8 Ponteiro para Ponteiro
- 9 Ponteiro para Função

Definição

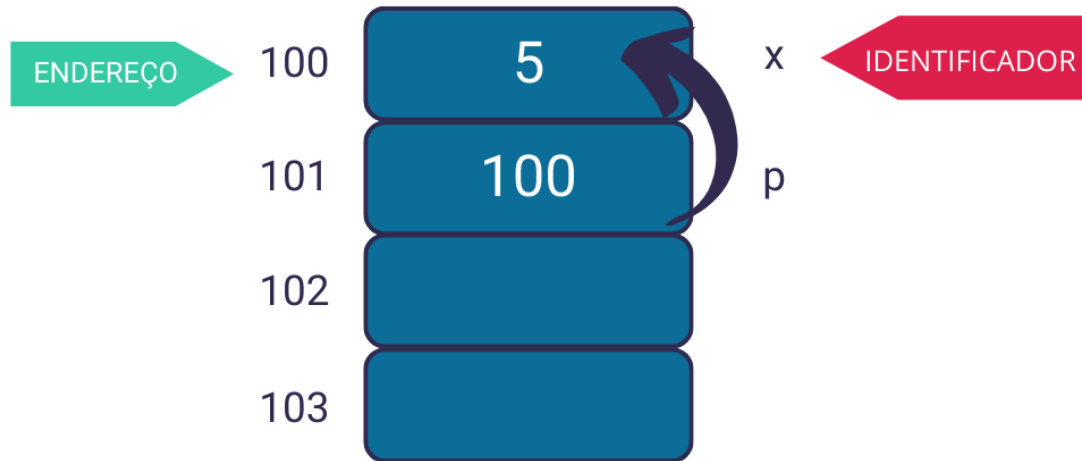
O que é um Ponteiro?

Os ponteiros são variáveis que contêm endereços de memória como valores.

(Deitel; Deitel, 2011)

*Um ponteiro é uma variável que contém um endereço de memória. Esse endereço é normalmente a posição de uma outra variável na memória. Se uma variável contém o endereço de uma outra, então **a primeira variável é dita para "apontar" para segunda**.*

(Schildt, 1996)



- › Uma variável faz uma referência direta a um valor específico.
- › Um ponteiro, por outro lado, contém um endereço de uma variável que contém um valor.
- › Sob esse ponto de vista, um nome de variável faz uma referência direta a um valor, e um ponteiro faz referência indireta a um valor.

Declaração

```
1 int main(){  
2     int *p;  
3 }
```

Código 1: Declarar ponteiro.

```
1 #include <stdio.h>
2 int main(){
3     int *p = NULL;
4 }
```

Código 2: Inicializar ponteiro com NULL^a.

^aOs ponteiros devem ser inicializados ao serem declarados ou em uma instrução de atribuição. Um ponteiro pode ser inicializado com 0, NULL ou um endereço. Um ponteiro com o valor NULL não aponta para lugar algum. NULL é uma constante simbólica definida no arquivo de cabeçalho <stdio.h>

Atribuição

```
1 #include <stdio.h>
2 int main(){
3     int x = 5;
4     int *p = NULL;
5     p = &x;
6 }
```

Código 3: Atribuição.

Operadores

Operador	Descrição
*	É um operador unário que devolve o endereço de memória do seu operando.
&	É um operador unário que devolve o valor da variável localizada no endereço a

Tabela 1: Operadores de ponteiros.

```
1 #include <stdio.h>
2 int main(){
3     int x = 5;
4     int *p = NULL;
5     p = &x;
6     printf("Valor de X: %d\n", x);
7     printf("Endereco de X atraves de operador: %p\n", &x);
8     printf("Endereco de X atraves de P: %p\n", p);
9     printf("Valor de P: %p\n", p);
10    printf("Valor de X atraves de P: %d\n", *p);
11    printf("Endereco de P: %p\n", &p);
12 }
```

Código 4: Exemplo de manipulação com operadores.

Referência e Valor

- **Argumento por valor:** A função recebe uma cópia do valor da variável, sem modificar a original.
- **Argumento por referência:** A função recebe um ponteiro ou referência à variável original, permitindo modificá-la diretamente.

```
1 #include <stdio.h>
2 int somar(int a,int b){
3     return a = b;
4 }
5 int main(){
6     int resultado = somar(10,8);
7     printf("%d", resultado);
8     return 0;
9 }
```

Código 5: Exemplo de argumento por valor.

```
1 #include <stdio.h>
2 int somar(int a,int b, int *resultado){
3     resultado = a + b;
4 }
5 int main(){
6     int resultado = somar(10,8,&resultado);
7     printf("%d", resultado);
8     return 0;
9 }
```

Código 6: Exemplo de argumento por referência.

```
1 #include <stdio.h>
2 int main(){
3     int x;
4     printf("Digite X:\n");
5     //A variável x é passada por referência para scanf().
6     scanf("%d", &x);
7     printf("%d", x);
8     return 0;
9 }
```

Código 7: Exemplo de argumento por referência `scanf()`.

Vetor

```
1 #include <stdio.h>
2 int main(){
3     int vetor[5];
4     for(int i = 0; i < 5; i++){
5         printf("Digite um inteiro:\n");
6         scanf("%d", vetor[i]);
7     }
8     return 0;
9 }
```

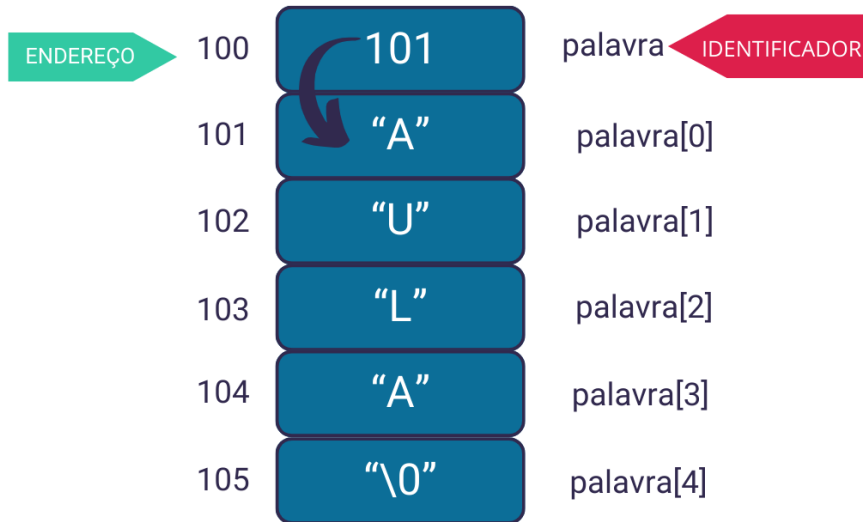
Código 8: Exemplo de vetor para Função.

Por que ao passar um vetor por referência, não é necessário usar o operador &?

Atenção!

- Quando você passa um vetor para uma função, ele já é passado por referência, ou seja, a função recebe o endereço de memória do primeiro elemento do vetor.
- Isso acontece porque o nome de um vetor, em C, é um ponteiro para seu primeiro elemento.
- Portanto, não é necessário usar o operador `&`, pois o vetor em si já se comporta como um ponteiro.

Aritmética de Ponteiros



- A alocação de memória de vetores é sequencial, ou seja, todos os elementos estão contíguos na memória.
- Ao acessar um vetor com um índice, como `vetor[i]`, isso é tratado como `*(vetor + i)`.
- Incrementando o índice `i`, você está movendo o ponteiro para o próximo endereço de memória do vetor.

Ponteiro para Ponteiro

```
1 #include <stdio.h>
2 int main(){
3     int matriz[4][4];
4     for(int i = 0; i < 4; i++){
5         for(int j=0; j < 4; j++){
6             printf("Digite um inteiro:\n");
7             scanf("%d", matriz[i][j]);
8         }
9     }
10    return 0;
11 }
```

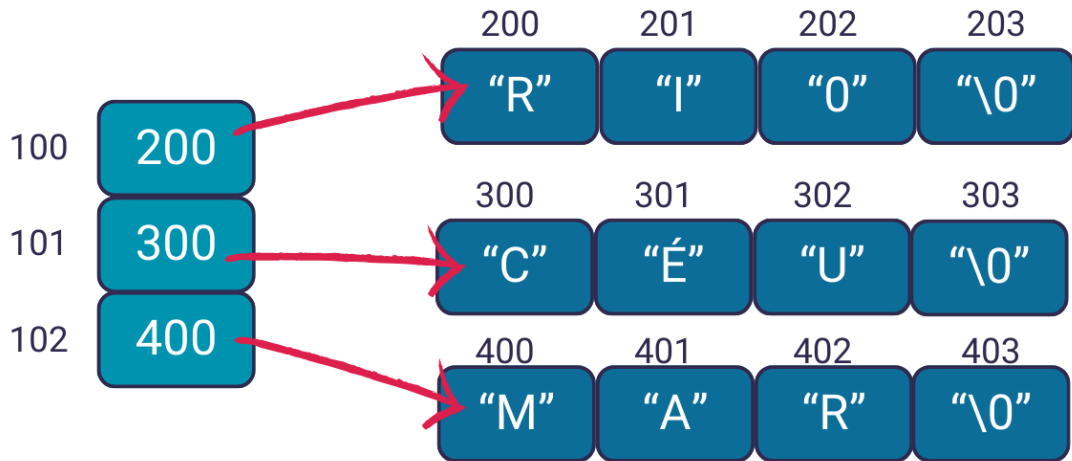
Código 9: Exemplo de matriz.

Nota:

Note que também não é necessário usar o operador `&` para acessar uma posição na matriz. No entanto, usamos duas variáveis contadoras. Como uma matriz se estrutura na memória?

```
1 #include <stdio.h>
2 void mostrar_matriz(int **matriz, int linha, int coluna){
3     for(int i=0; i < linha; i++){
4         for(int j=0; j < coluna; j++){
5             printf("%d", matriz[i][j]);
6             printf("\n");
7         }
8     }
9 int main(){
10     int mairtiz = {
11         1,2,3,
12         4,5,6,
13         7,8,9
14     };
15     mostrar_matriz(matriz,3,3);
16 }
```

Código 10: Exemplo de matriz para função.



- Usamos ** em matrizes porque, em C, uma matriz bidimensional pode ser representada como um ponteiro para ponteiro.
- Uma matriz é essencialmente um **vetor de vetores**, onde cada linha é um vetor armazenado em um endereço de memória.

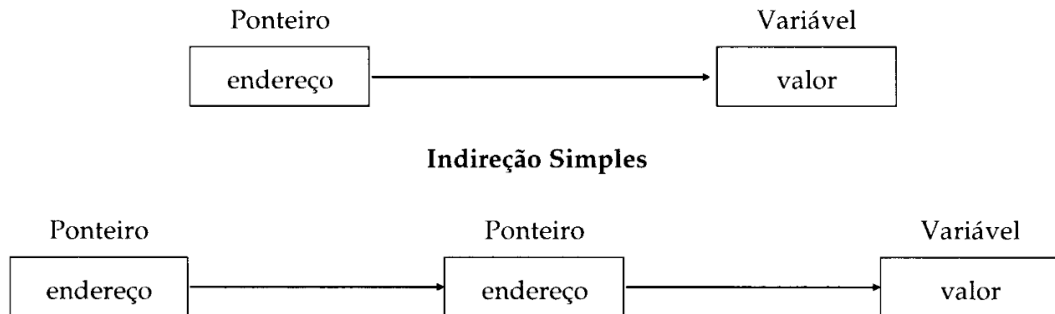


Figura 1: Ponteiro Indireção Múltipla (Schildt, 1996)

```
1 #include <stdio.h>
2 int main() {
3     int x = 10;
4     int *p = NULL;
5     int **pp = NULL;
6
7     p = &x;    // p recebe o endereço de x
8     pp = &p;   // pp recebe o endereço de p
9
10    printf("%d\n", **pp); // Acessando o valor de x através do ponteiro para ponteiro
11    return 0;
12 }
```

Código 11: Exemplo ponteiro para ponteiro.

Ponteiro para Função

```
1 #include <stdio.h>
2 int dobro(int x) {
3     return 2 * x;
4 }
5
6 int main() {
7     int (*funcPtr)(int);
8     funcPtr = &dobro;
9
10    printf("Resultado: %d\n", funcPtr(5)); // Chama a função usando o ponteiro
11    return 0;
12 }
```

Código 12: Exemplo ponteiro para função.

```
1 #include <stdio.h>
2 int soma(int a, int b){
3     return a + b;
4 }
5 int multiplica(int a, int b){
6     return a * b;
7 }
8 int calcular(int x, int y, int (*operacao)(int, int)) {
9     return operacao(x, y); // Chama a função apontada pelo ponteiro
10 }
11 int main() {
12     int resultado1 = calcular(5, 3, soma);
13     int resultado2 = calcular(5, 3, multiplica);
14     printf("Soma: %d\n", resultado1);
15     printf("Multiplicação: %d\n", resultado2);
16     return 0;
17 }
```

Código 13: Exemplo de uso de ponteiro para função.

- Um **ponteiro para função** é uma variável que armazena o endereço de uma função, permitindo chamá-la indiretamente.
- Sua **utilidade** está na flexibilidade de passar funções como parâmetros, permitindo a personalização de comportamentos e a reutilização de código.

(Deitel; Deitel, 2011) - Capítulo 7



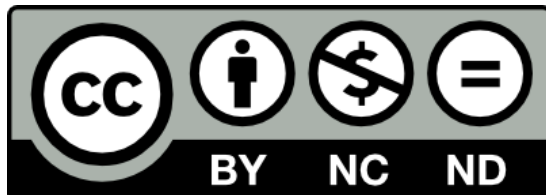


DEITEL, Paul; DEITEL, Harvey. **C: Como Programar**. 6. ed. São Paulo: Pearson Universidades, 2011.



SCHILDT, Herbert. **C Completo e Total: O Guia Definitivo para Programação em C**. 3. ed. São Paulo: Makron Books, 1996. ISBN 978-8534606928.

Estes slides estão protegidos por uma licença Creative Commons



Este modelo foi adaptado de Maxime Chupin.

Linguagem de Programação

Linguagem de Programação C
Ponteiro