Marisangila Alves, MSc
marisangila.alves@catolicasc.org.br
marisangila.com.br

Católica de Santa Catarina

2025/1

Estrutura de Dados

Estrutura de Dados
Ponteiros

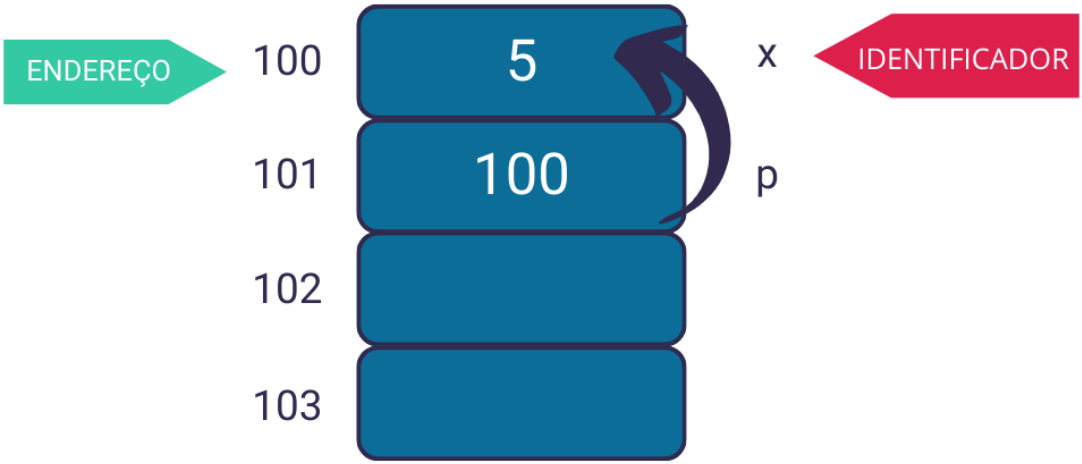
Sumário

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)


ENDEREÇO
100
101
102
103
5
100
x
p
IDENTIFICADOR

- Uma variavel 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

```c
int main(){
    int *p;
}
```

Código 1: Declarar ponteiro.

```c
#include <stdio.h>
int main(){
    int *p = NULL;
}
```

Código 2: Inicializar ponteiro com `NULL`[a].

[a]Os 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 constantesimbólica definida no arquivo de cabeçalho `<stdio.h>`

Atribuição

```c
#include <stdio.h>
int main(){
    int x = 5;
    int *p = NULL;
    p = &x;
}
```

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 ar

Tabela 1: Operadores de ponteiros.

```c
#include <stdio.h>
int main(){
    int x = 5;
    int *p = NULL;
    p = &x;
    printf("Valor de X: %d\n", x);
    printf("Endereco de X atraves de operador: %p\n", &x);
    printf("Endereco de X atraves de P: %p\n", p);
    printf("Valor de P: %p\n", p);
    printf("Valor de X atraves de P: %d\n", *p);
    printf("Endereco de P: %p\n", &p);
}
```

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.

```c
#include <stdio.h>
int somar(int a,int b){
    return a = b;
}
int main(){
    int resultado = somar(10,8);
    printf("%d", resultado);
    return 0;
}
```

Código 5: Exemplo de argumento por valor.

```c
#include <stdio.h>
int somar(int a,int b, int *resultado){
    resultado = a = b;
}
int main(){
    int resultado = somar(10,8,&resultado);
    printf("%d", resultado);
    return 0;
}
```

Código 6: Exemplo de argumento por referência.

```c
#include <stdio.h>
int main(){
    int x;
    printf("Digite X:\n");
    //A variável x é passada por referência para scanf().
    scanf("%d", &x);
    printf("%d", x);
    return 0;
}
```

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

Vetor

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

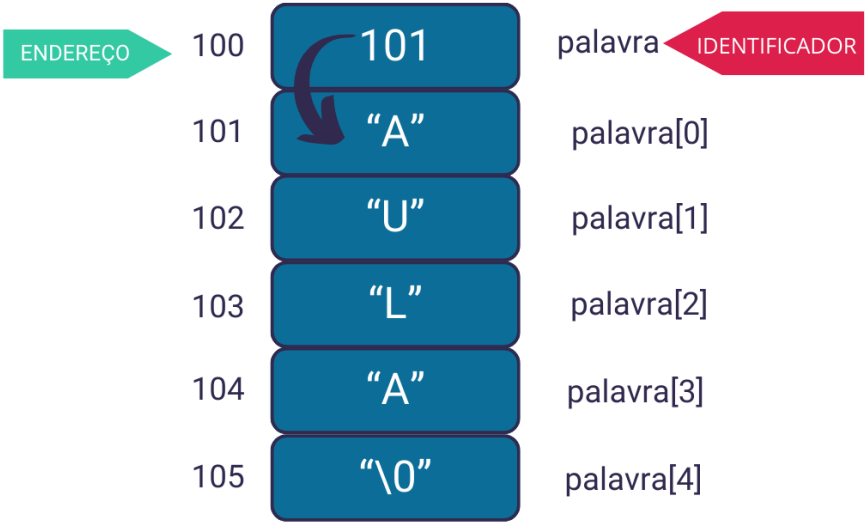
Código 8: Exemplo de vetor para Função.

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


ENDEREÇO
100
101
palavra
IDENTIFICADOR
101
“A”
palavra[0]
102
“U”
palavra[1]
103
“L”
palavra[2]
104
“A”
palavra[3]
105
“\0”
palavra[4]

- 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

```c
#include <stdio.h>
int main(){
    int matriz[4][4];
    for(int i = 0; i < 4; i++){
        for(int j=0; j < 4; j++){
            printf("Digite um inteiro:\n");
            scanf("%d", matriz[i][j]);
        }
    }
    return 0;
}
```

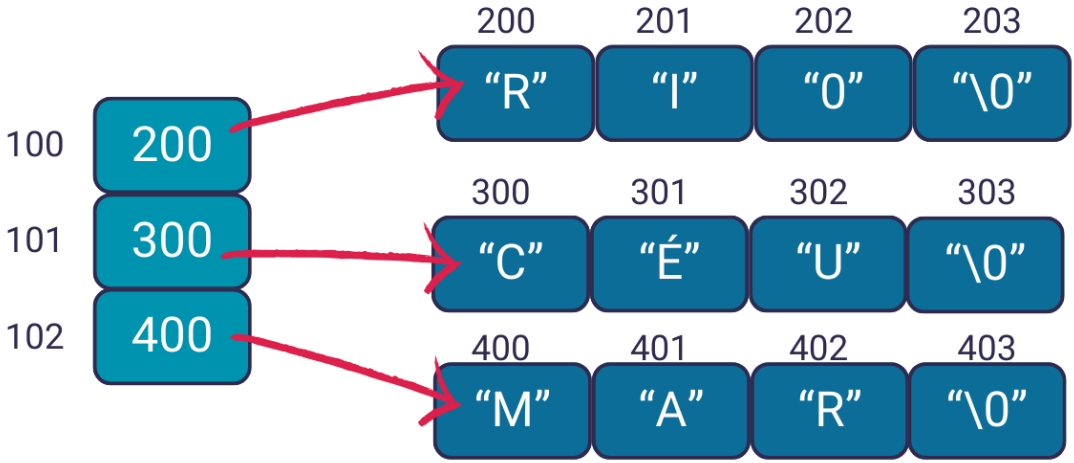
Código 9: Exemplo de matriz.

Nota:

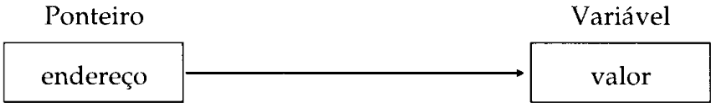
Note que também não é necessário usar o operador & para acessar uma posição na matriz. No entanto, usamos duas variáveis contadoras. Como uma matriz se estrutura na memória?

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

Código 10: Exemplo de matriz para função.


200
201
202
203
“R”
“I”
“0”
“\0”
100
200
101
300
102
400
300
301
302
303
“C”
“É”
“U”
“\0”
400
401
402
403
“M”
“A”
“R”
“\0”

- 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.



Indireção Simples

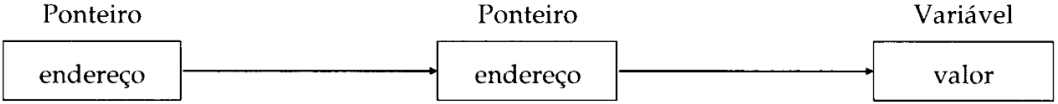


Figura 1: Ponteiro Indireção Múltipla (Schildt, 1996)

```c
#include <stdio.h>
int main() {
    int x = 10;
    int *p = NULL;
    int **pp = NULL;

    p = &x;   // p recebe o endereço de x
    pp = &p;  // pp recebe o endereço de p

    printf("%d\n", **pp); // Acessando o valor de x através do ponteiro para ponteiro
    return 0;
}
```

Código 11: Exemplo ponteiro para ponteiro.

Ponteiro para Função

```c
#include <stdio.h>
int dobro(int x) {
    return 2 * x;
}

int main() {
    int (*funcPtr)(int);
    funcPtr = &dobro;

    printf("Resultado: %d\n", funcPtr(5)); // Chama a função usando o ponteiro
    return 0;
}
```

Código 12: Exemplo ponteiro para função.

```c
#include <stdio.h>
int soma(int a, int b){
    return a + b;
    }
int multiplica(int a, int b){
    return a * b;
}
int calcular(int x, int y, int (*operacao)(int, int)) {
    return operacao(x, y); // Chama a função apontada pelo ponteiro
}
int main() {
    int resultado1 = calcular(5, 3, soma);
    int resultado2 = calcular(5, 3, multiplica);
    printf("Soma: %d\n", resultado1);
    printf("Multiplicação: %d\n", resultado2);
    return 0;
}
```

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.

Marisangila Alves, MSc
marisangila.alves@catolicasc.org.br
marisangila.com.br

Católica de Santa Catarina

2025/1

Estrutura de Dados

Estrutura de Dados
Ponteiros