



Universidade de Brasília
Faculdade UnB Gama
Disciplina: Estrutura de Dados e Algoritmos - EDA

Alocação Dinâmica de Memória

Aula sobre Ponteiros em C/C++

Prof. Nilton Correia da Silva

27 de setembro de 2023

Introdução a Ponteiros

- A variável na memória
- O que é um Ponteiro?
- Declaração de Ponteiros
- Operações

Componentes de um Ponteiro

- Parte de Endereçamento
- Parte de Valor

Exemplos de Códigos

- Exemplo 1: Endereço e Valor de uma Variável
- Exemplo 2: Troca de Valores

Considerações Finais

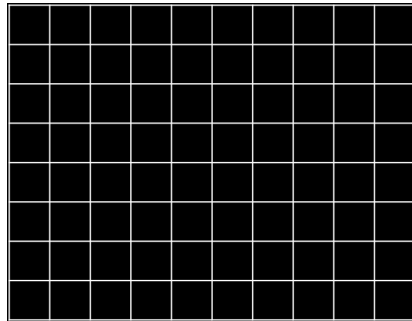
- Boas Práticas e Cuidados
- Conclusões

Introdução a Ponteiros

A variável na memória



```
#include <stdio.h>
int main() {
    return 0;
}
```



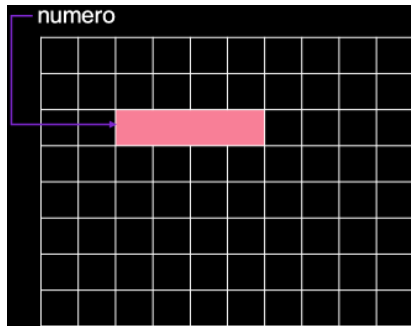
Área de memória do programa.

Introdução a Ponteiros

A variável na memória



```
#include <stdio.h>
int main() {
    int numero;
    return 0;
}
```



Área de memória do programa.

Introdução a Ponteiros

A variável na memória



```
#include <stdio.h>
```

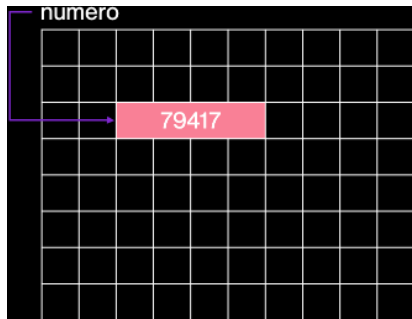
```
int main() {
```

```
    int numero;
```

```
    numero = 79417;
```

```
    return 0;
```

```
}
```



Área de memória do programa.

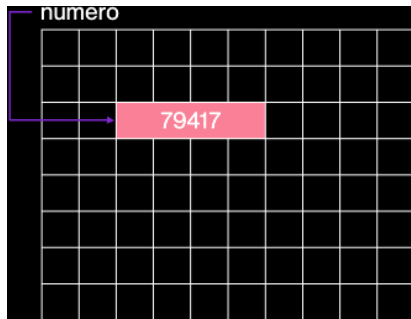
Introdução a Ponteiros

A variável na memória



```
#include <stdio.h>

int main() {
    int numero;
    numero = 79417;
    //Valor de 'numero'
    printf("Valor: %d\n", numero);
    //Endereço de 'numero'
    printf("Endereço: %p\n", &numero);
    return 0;
}
```



Área de memória do programa.

Introdução a Ponteiros

O que é um Ponteiro?



Um **ponteiro** é uma variável que armazena o endereço de memória de outra variável. Isso nos permite acessar e manipular diretamente a memória do sistema.

Em C/C++, os ponteiros são declarados usando a seguinte sintaxe:

```
1      tipo *nome_do_ponteiro;
```

onde **tipo** é o tipo de dado ao qual o ponteiro aponta.

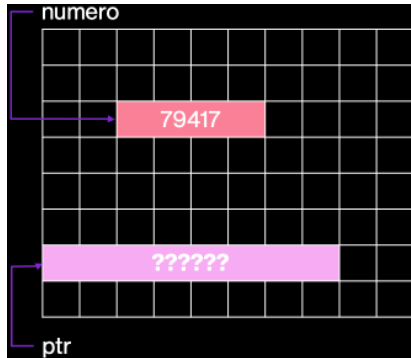
Introdução a Ponteiros

Operações: Declaração



```
#include <stdio.h>

int main() {
    int numero, *ptr;
    numero = 79417;
    return 0;
}
```



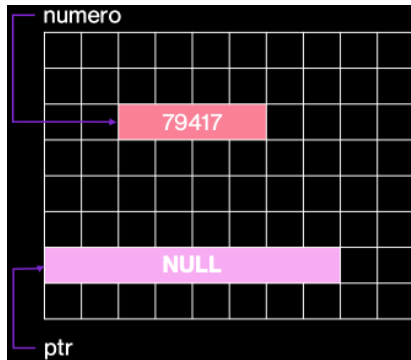
Área de memória do programa.

Introdução a Ponteiros

Operações: Inicialização



```
#include <stdio.h>
int main() {
    int numero, *ptr;
    numero = 79417;
    ptr = NULL;
    return 0;
}
```



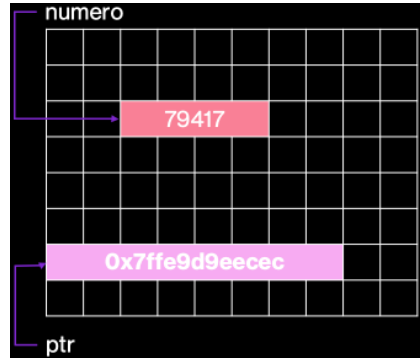
Área de memória do programa.

Introdução a Ponteiros

Operações: Atribuição



```
#include <stdio.h>
int main() {
    int numero, *ptr;
    numero = 79417;
    ptr = &numero;
    return 0;
}
```



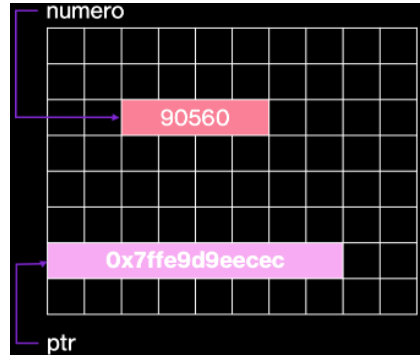
Área de memória do programa.

Introdução a Ponteiros

Operações: Atribuição



```
#include <stdio.h>
int main() {
    int numero, *ptr;
    numero = 79417;
    ptr = &numero;
    *ptr = 90560
    return 0;
}
```



Área de memória do programa.

Componentes de um Ponteiro

Parte de Endereçamento



Um ponteiro armazena o endereço de memória de uma variável. Isso é obtido usando o operador de endereço (&) e permite que trabalhem com o endereço real da variável.

Para acessar o valor apontado por um ponteiro, usamos o operador de desreferência (*). Isso nos permite obter o valor armazenado na memória em um endereço específico.

Exemplos de Códigos

Exemplo 1: Endereço e Valor de uma Variável



```
1  #include <stdio.h>
2
3  int main() {
4      int numero = 79417;
5      int *ptr; // Declaração de um ponteiro para int
6
7      //Atribuição do endereço de 'numero' ao ponteiro ptr:
8      ptr = &numero;
9
10     //Impressão do endereço de 'numero'
11     // e seu valor usando o ponteiro
12     printf("Endereço de 'numero': %p\n", ptr);
13     printf("Valor de 'numero': %d\n", *ptr);
14
15     return 0;
16 }
```

Exemplos de Códigos

Exemplo 2: Troca de Valores



```
1  #include <stdio.h>
2  // Função para trocar o valor de duas variáveis float:
3  void trocarValores(float *ptr1, float *ptr2) {
4      float temp = *ptr1; //Armazena o valor de ptr1 em uma variável
5      *ptr1 = *ptr2;      //Atribui o valor de ptr2 a ptr1
6      *ptr2 = temp;      //Atribui o valor temporário a ptr2
7  }
8  int main() {
9      float a = 3.14;
10     float b = 2.71;
11     printf("Valores originais: a = %.2f, b = %.2f\n", a, b);
12     //Chama a função para trocar os valores de 'a' e 'b':
13     trocarValores(&a, &b);
14     printf("Valores trocados: a = %.2f, b = %.2f\n", a, b);
15     return 0;
16 }
```


- ▶ Inicializar ponteiros antes de usá-los.
- ▶ Não tentar acessar áreas de memória não alocadas.

Lições Aprendidas

- ▶ O que são ponteiros.
- ▶ Como declarar um ponteiro.
- ▶ Fazer uso adequado da parte de valor do ponteiro
- ▶ Fazer uso adequado da parte de endereçamento do ponteiro
- ▶ Ter boas práticas de programação envolvendo ponteiros