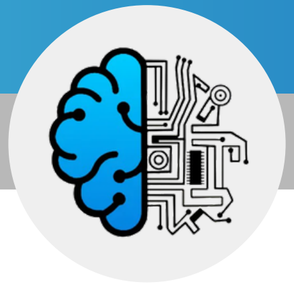


MINICURSO DE ALGORITMOS 2
PET-SIMC

PONTEIROS

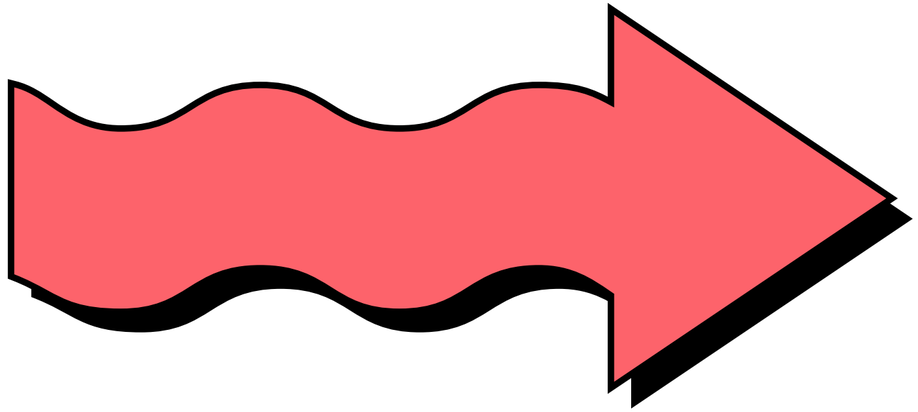
PET de Sistemas de Informação do Campus Monte Carmelo

PET de Sistemas de Informação do Campus Monte Carmelo

CRONOGRAMA:

1. Definição
2. Sintaxe
    a. Criando
    b. Acessando
    c. Modificando
3. Onde e por que usar?
4. Exercícios

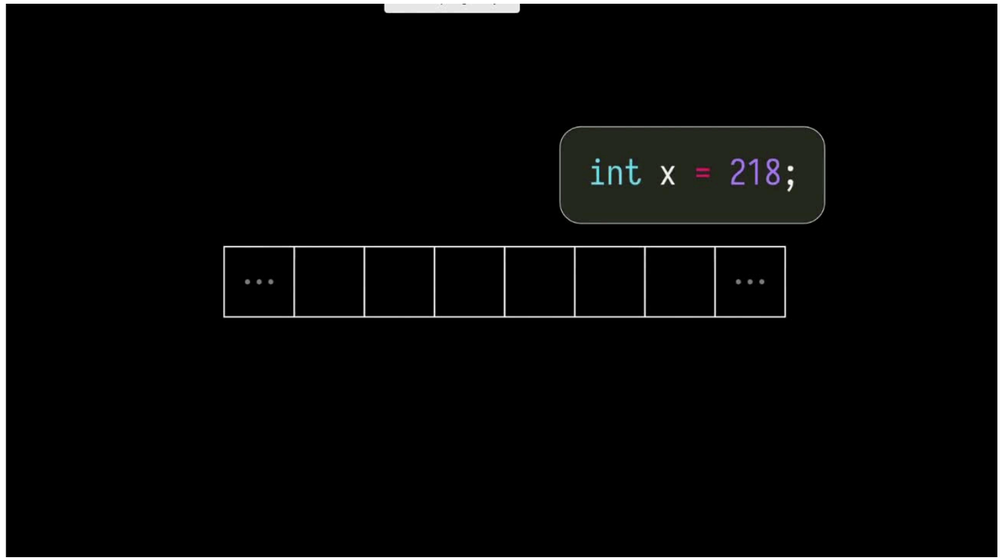
O que é um ponteiro?



Os ponteiros são um **tipo de variável especial do C/C++.**

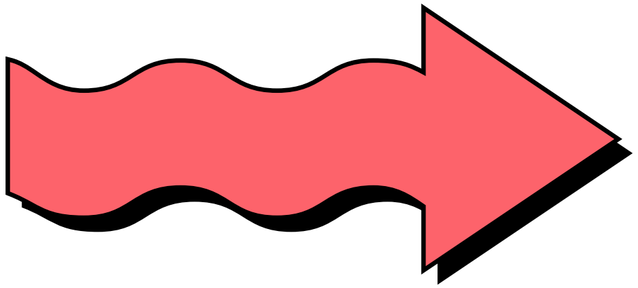
Como o nome sugere, eles **apontam** para alguma coisa, nesse caso para **uma posição da memória, um endereço.**

O que é um ponteiro?



Nós podemos imaginar **a memória como sendo uma lista de bytes**, toda vez que declaramos um int o computador escolhe uma **posição da memória** e reserva 4 bytes pra colocar o inteiro.

O que é um ponteiro?



O ponteiro é **uma váriavel que recebe uma posição na memória,** por isso dizemos que ele “aponta” para uma variável.

Utilizando o ponteiro podemos alterar o valor que está escrito na posição da memória que ele aponta, ou seja, **podemos alterar uma variável usando um ponteiro.**

Sintaxe:

Para **acessar** em qual **posição da memória** uma determinada variável está alocada, usamos **&.**

```c
#include <stdio.h>
int main() {
    int x;

    // Imprimindo a posição na memória que a variável x está localizada:
    printf("%p", &x); // 0x7ffd22810b84
    // O valor muda sempre que o programa é reiniciado, pois a
    // variavél é colocada em uma posição diferente na memória.
    return 0;
}
```

Sintaxe:

Para **declarar** um ponteiro usamos ***** ao lado do tipo da variável:

```c
int main() {
    // Para declarar um ponteiro usamos o * do lado do tipo da variável
    int* p;

    int x = 42;
    printf("%d",x); // 42

    // Agora temos um ponteiro que aponta para x
    p=&x;
```

Sintaxe:

Para **acessar/alterar** o conteúdo dentro do ponteiro:

```
// Também usando o * é possível acessar o valor que
// está na posição que o ponteiro aponta.
printf("%d",(*p)); // 42

// É possível alterar o valor da variável que o ponteiro aponta.
*p = 27;

// Após isso o valor na posição que ele aponta vira 27,
// logo a variável x também recebe o valor 27
printf("%d", x); // 27
```

Operadores unários

Operador "Endereço-de" (&)

- Retorna o endereço de memória de uma variável ou elemento de vetor.
- Com variáveis: &x → retorna onde x está armazenada na memória.
- Com vetores: &v[3] → retorna o endereço do 4º elemento do vetor v.

Operador "Conteúdo-de" (*)

- Acessa o valor armazenado no endereço apontado por um ponteiro.
- Com ponteiros: *p → retorna o valor guardado no endereço que p aponta.
- Com vetores: *(v+3) → acessa o 4º elemento de v (equivalente a v[3]).

Passagem por valor

Se não usarmos ponteiros, a função recebe uma **cópia do valor**, e não modifica a variável original.

```c
void incrementar(int x) {
    x++;
}

int main() {
    int valor = 5;
    incrementar(valor);
    printf("Valor após incrementar: %d\n", valor);
    return 0;
}
```

O que será printado em valor?

Ponteiros e funções

Quando passamos um ponteiro como parâmetro de uma função, dizemos que estamos passando essa variável por **referência.**

```c
void incrementar(int *x) {
    (*x)++;
}

int main() {
    int valor = 5;
    incrementar(&valor);
    printf("Valor incrementado: %d\n", valor);
    return 0;
}
```

O que será printado em valor?

Principais diferenças

Passagem por Referência (`int *x`)	Passagem por Valor (`int x`)
Recebe o **endereço** da variável (`&valor`)	Recebe uma **cópia** do valor
Modifica o **valor original**	Modifica apenas a **cópia local**
Útil para funções que alteram variáveis	Útil quando a função não precisa alterar a variável original

Pontos importantes

- Um ponteiro só pode apontar para variáveis ou elementos de vetores do seu tipo
- Ou seja, um ponteiro para int só pode apontar para objetos de memória do tipo int (só podemos atribuir a ele endereços de int)

Exemplo:

```
int *p = NULL;
int *q = &p;
printf (“%p”, *q);
```

Vamos praticar?!

Exercício 01

Faça uma função para trocar o valor de duas variáveis utilizando ponteiros.

Exercício 02

Crie uma função `void dobrar(int *num)` que recebe um ponteiro para um inteiro e dobra seu valor.

Exercício 03

Crie uma função `void resetar(int *num)` que define o valor de uma variável para zero.

CONTATO

Como entrar em contato em caso de dúvidas?

E-mail

laviniabd@ufu.br
nicolly.luz@ufu.br

Sala do PET-1BMC206

UFU - Monte Carmelo - Campus Araras

Site do PET-SIMC

petsimc.facom.ufu.br

Grupo do Wpp

@petsimc