Ponteiros

uma introdução ao caos

Introdução à memória

endereço	valor
0×1000	0×4
0×1004	
0×1008	
0×100C	

endereço	valor
0×1000	0×4
0×1004	0×1000
0×1008	
0×100C	

int x = 4

int * ponteiro = & x

int x = 4

int * ponteiro = & x

int y = * ponteiro

int x = 4

“O inteiro x recebe o valor 4”

int * ponteiro = & x

“O ‘ponteiro’ direciona/recebe o endereço de x”

int y = * ponteiro

“y recebe o que ‘ponteiro’ armazena”

int x = 4

int * ponteiro = & x

int y = * ponteiro

endereço	valor
0×1000	0×4
0×1004	0×1000
0×1008	0×1004
0×100C	

int x = 4

int * ponteiro = & x

int y = * ponteiro

endereço	valor
0×1000	0×4
0×1004	0×1000
0×1008	0×1004
0×100C	

Por que usar ponteiro?

mesmo sendo essa loucura toda...

Por que usar ponteiro?

mesmo sendo essa loucura toda...

Toda vez que você aloca algo, você usa espaço na memória
mas se você referencia, você reaproveita algo que já foi feito!

Como alocar espaço na memória?

```
int* ptr = new int;                 // Aloca memória
*ptr = 42;                          // Usa a memória alocada

cout << *ptr << endl;               // Exibe o valor

delete ptr;                         // Libera a memória
ptr = nullptr;                      // Evita ponteiro pendente
```

Como alocar espaço na memória?

```
int* ptr = (int*)malloc(sizeof(int));          // Aloca memória para um inteiro

if (ptr == nullptr) {
        cout << "Falha na alocação de memória" << endl;
        return 1;
}

*ptr = 42;                                      // Atribui valor ao espaço alocado

cout << *ptr << endl;                           // Exibe o valor

free(ptr);                                      // Libera a memória alocada
ptr = nullptr;                                  // Boa prática: definir o ponteiro para nullptr após liberar
```

Vamos para o código

ou pelo menos tentar

Obrigado pela atenção

Universidade Federal de São Carlos - Maratona de Programação