ALUNO: GABRIEL ALESSI POSONSKI **RA:2259583**

ATIVIDADES DE FIXAÇÃO: PONTEIROS

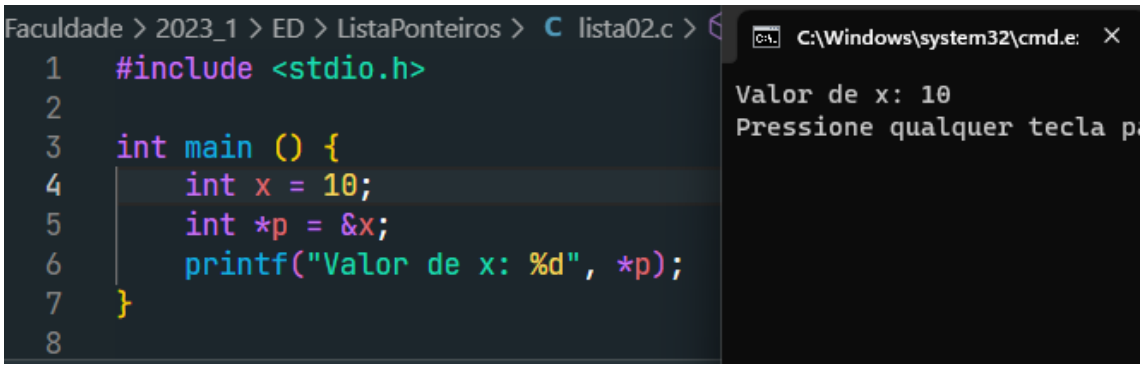
1. Espera se armazenar um endereço de memória. Pode ser utilizado para diversas funcionalidades, um exemplo é a alteração de uma variável de maneira indireta:
int a = 2; //saída: a = 2;
int *p = &a;

*p=3; //saída: a = 3;

Isso ocorre porque é possível alterar o conteúdo da variável cujo ponteiro está apontando, é prático em momentos que é preciso alterar o valor de uma variável dentro de funções, assim se for utilizado ponteiros, a função não precisa retornar algum valor, já que o valor da variável em questão será alterado diretamente na memória.

2.

3.



4. Podemos verificar através de uma estrutura condicional (ex. if) se um ponteiro é nulo ao verificar o endereço para o qual ele aponta, por exemplo:
if (p == NULL){

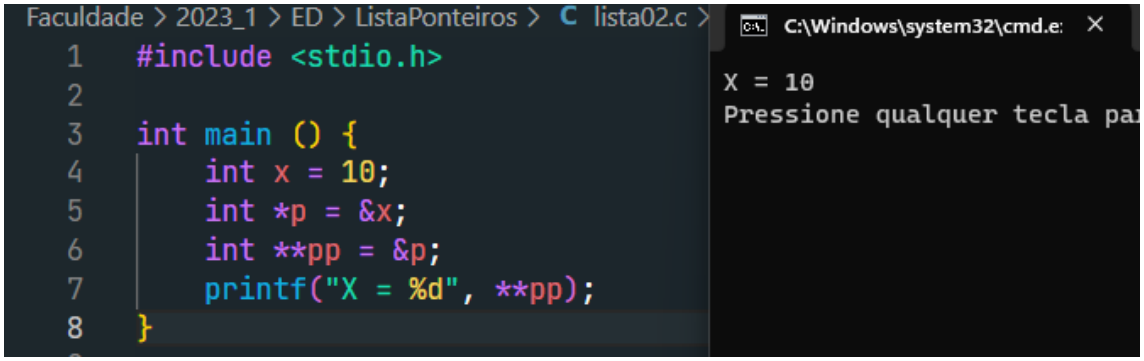
...}

Ou ainda sem conectivo lógico:
if(p){

...}

Nesse caso, se o ponteiro possuir valor, ou seja, apontar para alguém, ele será considerado verdadeiro.

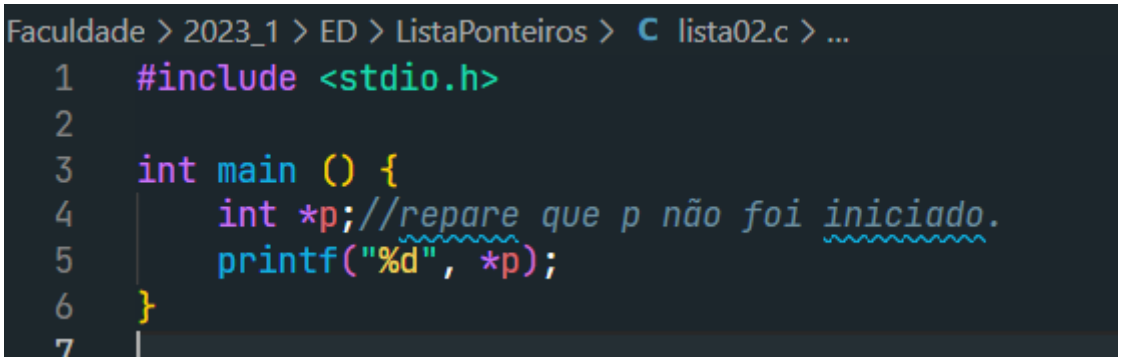
5.



6. Dado um ponteiro *p que aponta para NULL, se tentarmos escrever algo através dele, provavelmente teremos um erro de segmentação ou quaisquer outros comportamentos indevidos, afinal estamos tentando inserir um valor em uma região de memória inválida, um local sem endereço.

7.

***Leitura ilegal: tentar ler um ponteiro que aponta para uma região que não pertence ao programa/código em si.



*Escrita ilegal: tentar escrever através de um ponteiro uma região na memória que não pertence ao programa/código em si.

*Vazamento de memória: pode ocorrer após um processo de alocação dinâmica. Caso o programa não a libere após o uso, pode ocasionar em problemas de esgotamento e consequentemente um mal funcionamento do sistema.

*Falha de segmentação: ocorre ao tentar acessar uma área da memória não autorizada, como áreas utilizadas por outras aplicações ou o próprio sistema, áreas não alocadas ou inválidas.