Ponteiros

* Toda variável ocupa uma localização na memória;
* Seu endereço é sempre o primeiro da sequência reservada (onde a variável está alocada);
* *Ponteiros*: modo de acesso *indireto* à variável (sem referenciá-la);
* *Ponteiro variável*: variável que armazena um endereço de memória de outra variável - *variável que aponta para outra variável;*
* Ponteiros também têm endereço de memória (4 bytes);
* Podemos criar ponteiros de qualquer tipo (int, float, double, char) e *structs;*
* É preciso inicializar o ponteiro.

DECLARANDO PONTEIROS ------

*tipo \*nome\_variável;*

int \*p; ou int\* p; (ponteiro selvagem – sem inicialização)

CUIDADO!!

Ponteiros selvagens podem ocasionar em erros

INICIALIZANDO PONTEIROS -------

int \*ptrN = NULL; *NULL* é uma constante da biblioteca *stdio.h* que indica que o ponteiro aponta para nada

IMPRIMINDO PONTEIROS -------

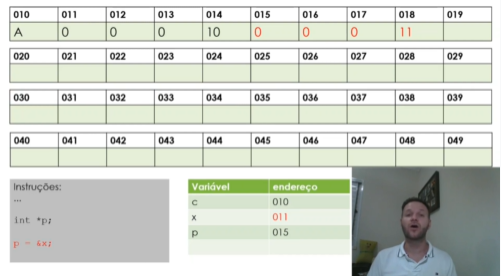
printf(“O endereço de x eh: %p”, **&x**) não apresenta o valor da variável x, mas sim seu endereço

PONTEIROS APONTANDO PARA OUTRA VARIÁVEL ------

int x = 10;

int \*p;

p = &x; p aponta para x



OPERADOR DE INDIREÇÃO *&* (DESREFERENCIAÇÃO) ------

* Retorna o valor da memória apontada

int x, \*p;

x = 10;

p = &x;

printf(“O valor apontado por p eh: %d, \*p”);

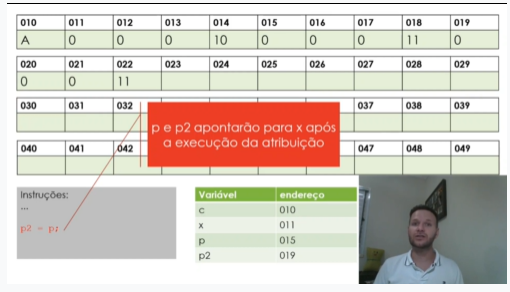
UM PONTEIRO PODE APONTAR PARA OUTRO ------

int x, \*p, \*p2;

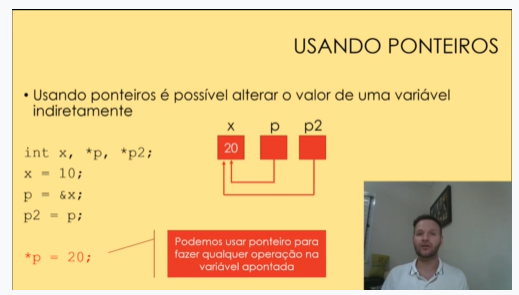
x = 10;

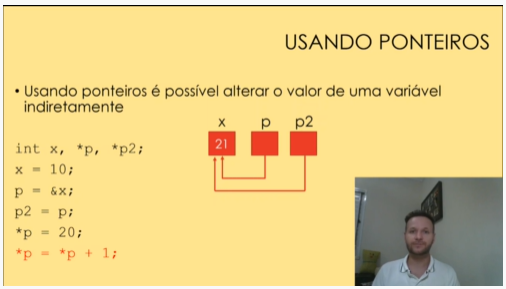
p = &x;

p2 = p;

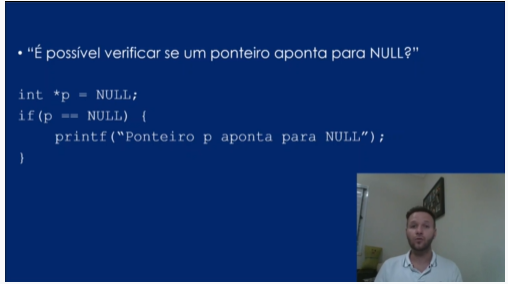


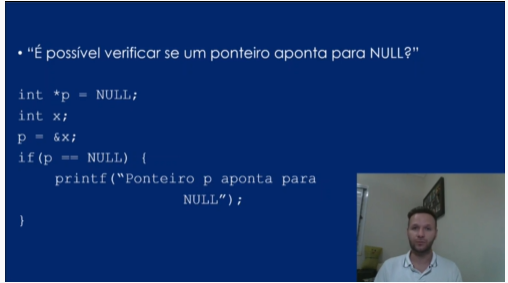
ALTERANDO O VALOR DE UMA VARIÁVEL ATRAVÉS DE UM PONTEIRO -----

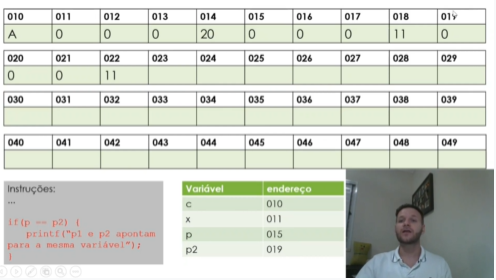


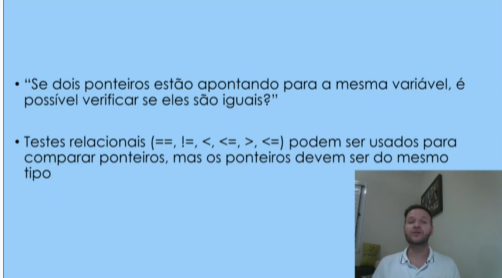


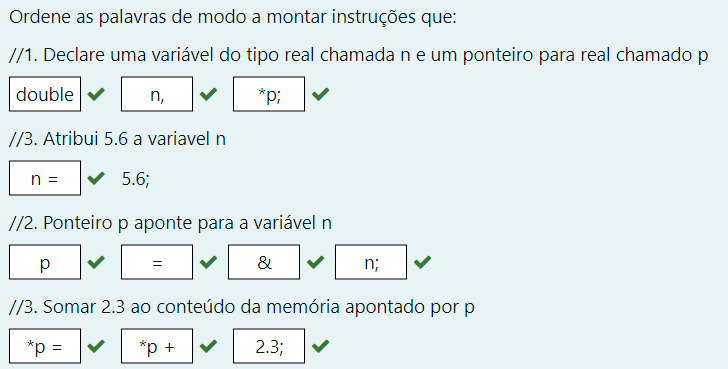
DESCOBRINDO VALOR DO PONTEIRO -----

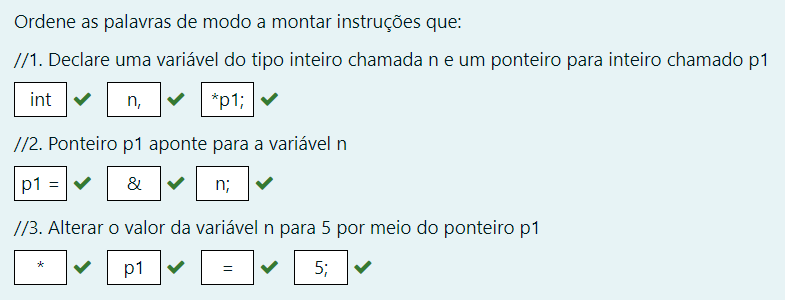


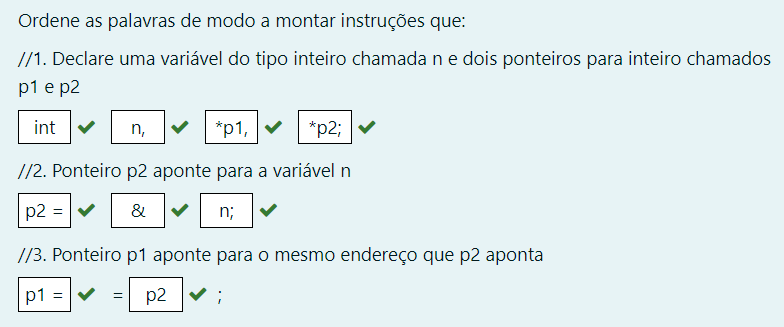


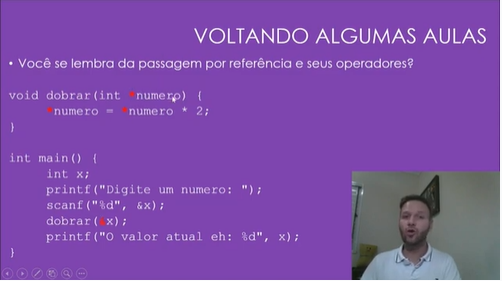


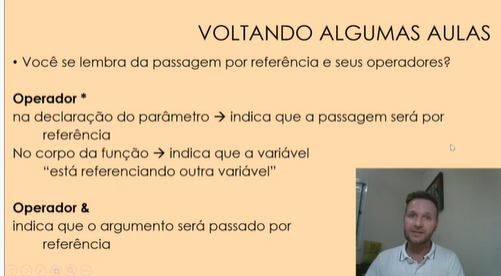


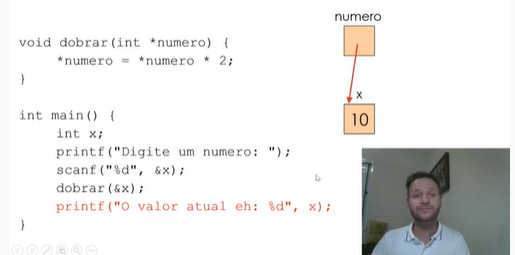












Alocação dinâmica de memória

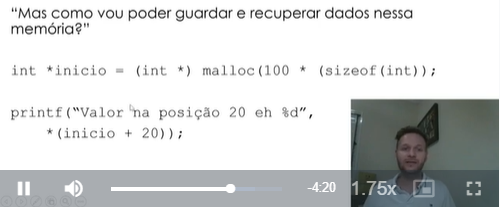
* Área chamada de *heap:* memória disponível que não foi utilizada (não alocada);
* Função *malloc*: para alocar memória - indica a quantidade de bytes a serem alocados, retorna o endereço do primeiro byte alocado e se não houver memória o suficiente, retorna *NULL;*
* *Malloc*  e *free* são da classe *stdilib.h.*

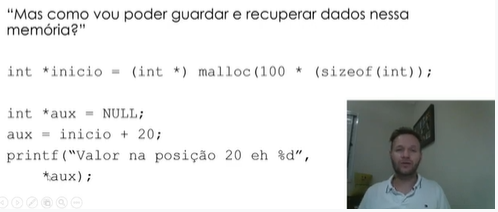
int \*pNumero;

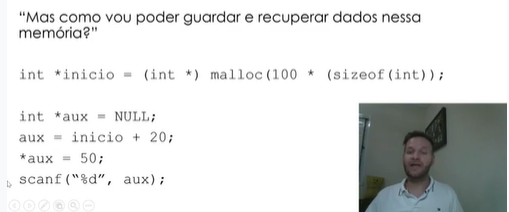
pNumero = (int \*) malloc(4); ou malloc(sizeof(int)) retorna o número de bytes necessários

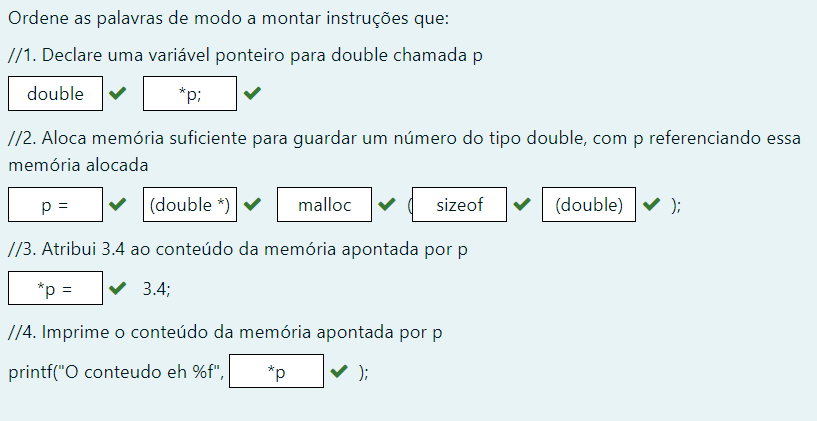
free (pNumero); liberar memória alocada

int \*inicio = (int \*) malloc(100 \* (sizeof(int)); como se tivéssemos alocando memória de um vetor de 100 posições

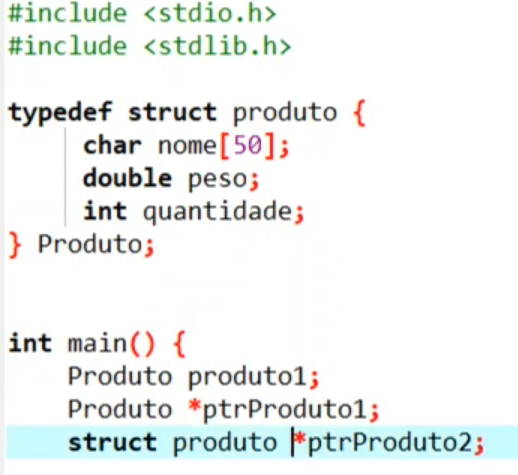


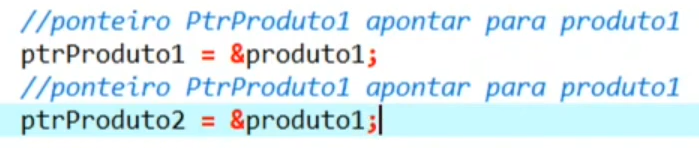


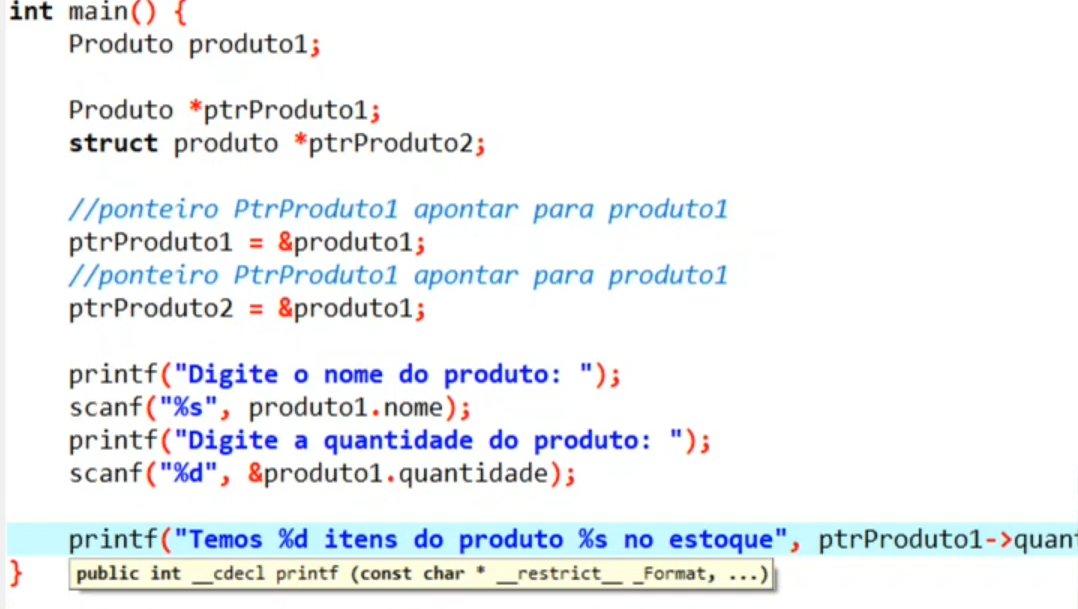


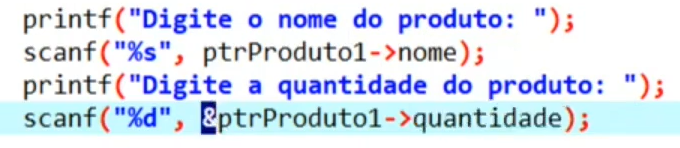


Ponteiros e estruturas









-> para estruturas e ponteiros ao invés de .

