

UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO

CENTRO MULTIDISCIPLINAR DE PAU DOS FERROS

DEPARTAMENTO DE ENGENHARIAS E TECNOLOGIA

DISCIPLINA: LABORATÓRIO DE ALGORITMOS E ESTRUTURAS DE DADOS I

DOCENTE: GEORGE FELIPE FERNANDES VIEIRA

MARIA VITÓRIA FERNANDES ROCHA

**LISTA DE EXERCÍCIO N° 01**

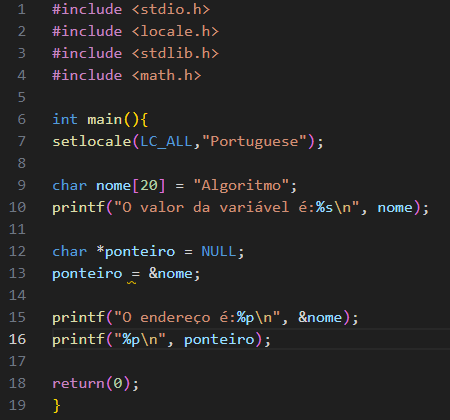
PAU DOS FERROS – RN

FEVEREIRO/2025

**Questão 01)** O que é e para que serve um ponteiro?

É uma variável que armazena o endereço de memória de outra variável. Ou seja, ela aponta para o local na memória onde o valor está alocado. Dessa forma, sua funcionalidade pode ser enxergada no acesso e manipulação direta da memória, o que facilita a passagem eficiente de parâmetros para funções – evitando cópias desnecessárias. Como também, na manipulação de arrays e strings, ou até mesmo na implementação de estruturas de dados dinâmicas, como pilhas ou árvores, exemplificadas em sala.

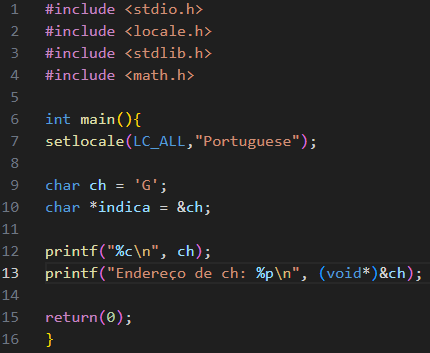
**Questão 02)** Declare uma variável e “*printe”* o valor dela e o seu endereço.



**Questão 03)** Qual é a maneira correta de referenciar **ch**, assumindo que o endereço de **ch** foi atribuído ao **ponteiro indica**?

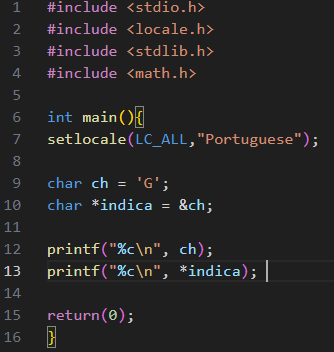
Levando em consideração que o momento de referenciamento acontece quando pegamos o endereço de uma variável e armazenamos em um ponteiro. Podemos visualizar na linha 10 do código abaixo, quando se declara o ponteiro indica (\*indica) do tipo char e em seguida lhe é referenciado - a partir do & - à variável ch.

OBS: usei o “(void\*)” para que o compilador não de erro na conversão do tipo no printf!



**Questão 04)** Na expressão **float \*ptr;** o que é do tipo **float**?

Float indica que o ponteiro recebe um endereço de memória de uma variável do tipo float. Por isso, é possível exibir o próprio valor da variável printando \*ptr. Segue exemplo abaixo.



**Questão 05)** Como seria o **output** se eu desse “*print”* nas variáveis a seguir:

**int** x=68, y;

**int** \*p;

p = &x;

y = \*p + 200;

Seguindo o mesmo fluxo de pensamento da questão passada, podemos perceber que o ponteiro p recebe o endereço de memória da variável x, a partir do uso do &. Então, ao ser exibido, veremos o endereço de memória onde está localizado o conteúdo de x. Já se exibirmos x, veremos o valor atribuído a ela, 68. O mesmo acontece com y, pois lhe é atribuído o conteúdo do referenciamento do ponteiro mais o número 200 (68 + 200), e não o conteúdo do ponteiro somado a outro valor, resultando em 268.

**Questão 06)** Assumindo que queremos ler o **valor** de **x**, e o **endereço** de **x** foi atribuído a **px**, a instrução seguinte é correta? Por que?

**scanf** ( “%d”, \*px );

Está incorreto, pois a função scanf ao ser acionada vai esperar o endereço de memória da variável que o usuário pretende informar. Mas, ao pedir para ler \*px o código não devolve isso e sim o próprio conteúdo da variável que está vazia. O correto seria - scanf("%d", px); -, já que assim a função realmente acessaria o espaço de memória correto para armazenar o valor da variável x.

Ou seja, px = endereço de memória de x.

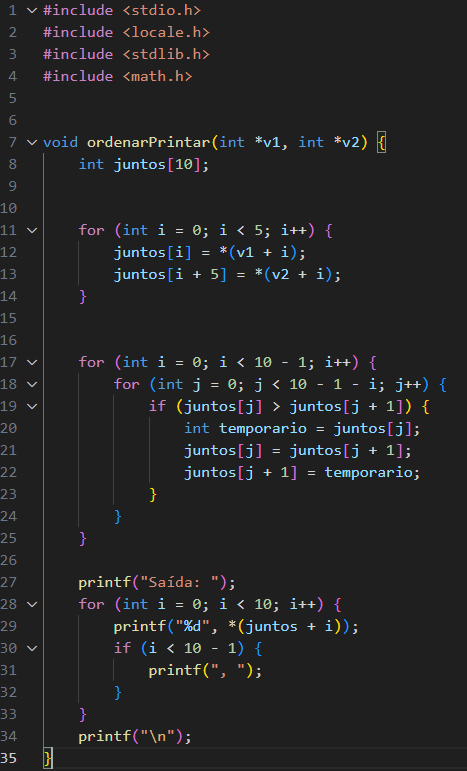
\*px = x.

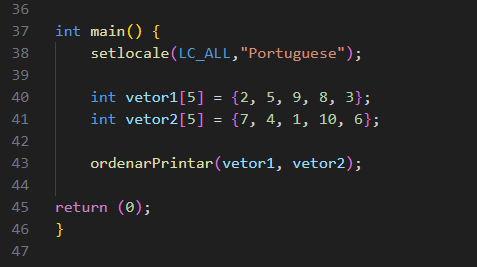
**Questão 07)** Desenvolva uma **função** que receba como parâmetro os **ponteiros** de **dois vetores de 5 posições**. O procedimento deverá imprimir na tela os **valores contidos nos dois vetores de forma crescente** (utilize ponteiros). Exemplo:

**Vetor1** = 2, 5, 9, 8, 3

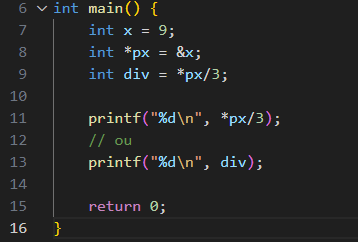
**Vetor 2** = 7, 4, 1, 10, 6

**Saída:** 1, 2, 3, 4, 5, 6, 7, 8, 9, 10





**Questão 08)** Assumindo que o **endereço** da variável x foi atribuído a um **ponteiro px**, escreva uma **expressão que não usa x** e devida **x** por 3.



**Questão 09)** Seja a seguinte sequência de instruções em um programa C:

**int** \*pti;

**int** i = 10;

pti = &i;

Qual afirmativa é falsa? Justifique a resposta

I - pti armazena o endereço de i;

II - \*pti é igual a 10;

III - Ao se executar \*pti = 20; i passará a ter o valor 20;

IV - Ao se alterar o valor de i, \*pti será modificado;

V - pti é igual a 10;

A justificativa dessa questão segue o mesmo raciocínio da questão 06, onde pti representa o endereço de memória ocupado pela variável. Considerando que, 10 seria o valor de i e não o endereço, a alternativa é falsa.