Nome: Luis Henrique Ferracciu Pagotto Mendes RA:2272016

Resumo Vídeo 19

Durante a vídeoaula de Ponteiros, o professor Muriel reforçou uma forma de passagens de parâmetros, em específico, Ponteiros. Que, é inicializado como uma variável comum, mas ao invés de armazenar um valor, armazena o endereço da memória em que foi salvo.

A partir de sua inicialização, é possível operar o Ponteiro de algumas formas para podermos trabalhar com ele: Referenciação (&), que retorna o endereço de memória em que a variável está sendo salva e Dereferenciação (\*), que retorna o conteúdo presente no endereço de memória, ou seja, o valor salvo naquele específico endereço.

Diante disto, podemos especificar mais algumas coisas. Um ponteiro só pode exibir ou receber o endereço de uma variável que seja do mesmo tipo, ou seja, se um ponteiro for inicializado como Inteiro, ele só pode receber endereços de variáveis do tipo inteiro. Além disso, é possível realizar contar aritméticas a partir dos ponteiros, por exemplo: se eu tenho a posição 495 da memória e somar 5 à variável do ponteiro, ele passará a me apresentar a posição 500 da memória. Além disso, é possível comparar ponteiros, ou, no caso, os endereços das variáveis.

Com a ideia de que um ponteiro só pode receber e exibir endereços do mesmo tipo, podemos declarar ponteiros como sendo genéricos, neste caso, para declarar um ponteiro deste tipo, basta declarar: void\* <nome\_do\_ponteiro>. Mas, para utilizá-lo corretamente, é necessário converter o ponteiro na hora da exibição, colocando o tipo da variável desejada antes do nome da variável que quiser exibir, por exemplo: (int) numero. A aritmética com ponteiros genéricos funciona da mesma forma que um ponteiro previamente declarado, com exceção do contador (ponteiro++), que ao invés de incrementar 4 no valor, que é o tanto que uma variável int ocupa, ele acrescentará apenas 1.

Também é possível trabalhar ponteiros com vetores, levando em conta que vetores já são algo como ponteiros, então é podemos enviar valores de um vetor diretamente para um ponteiro sem ter que usar o &, que deve ser usado normalmente, pois é um processo de vetor para vetor.