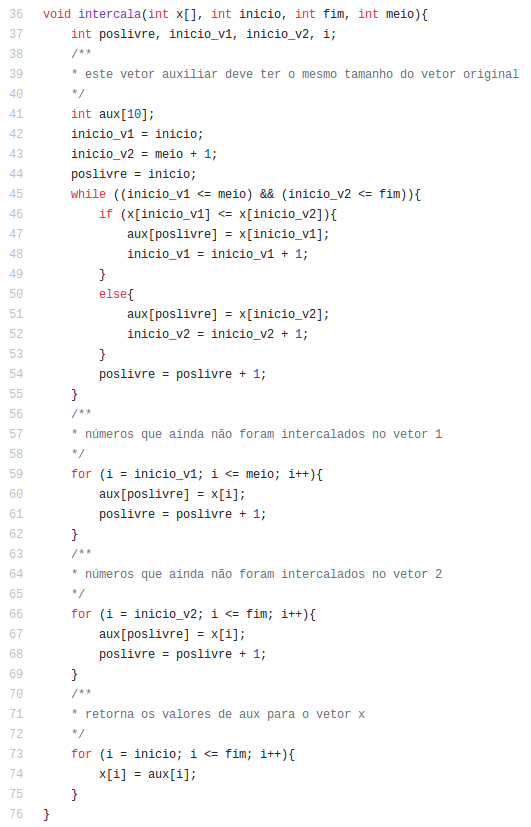
**Nome; Julio Cesario de Paiva Leão**

**RA: 1916033**

**Engenharia de Software**

**Estrutura de Dados 2018/01**

**MERGE SORT - FUNÇÃO INTERCALA**



É criado um vetor aux[10] onde irá receber os valores ordenados de v1 e v2 (dois vetores que serão intercalados)

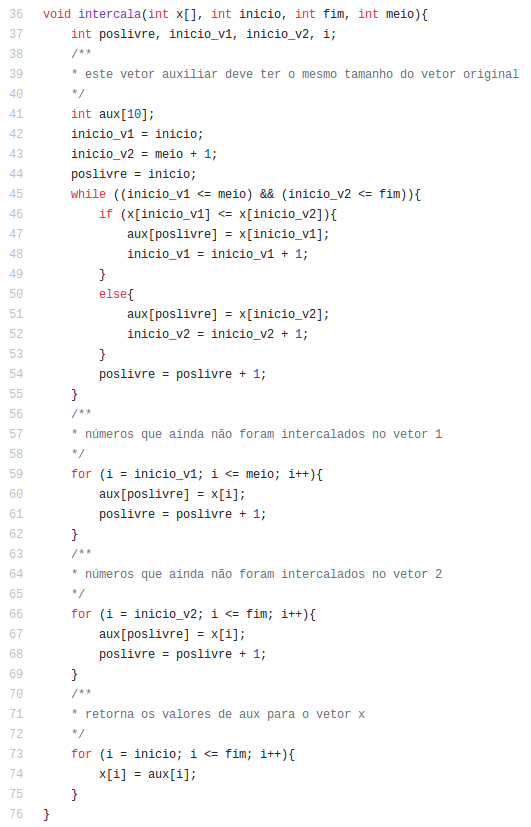


While para percorrer os vetores. E testar:

Se a posição do v1 for menor ou igual a posição de v2 (na posição central).

Vetor aux (na primeira posição) recebe o valor de v1, em seguida incrementa-se v1;

Senão o vetor aux 9 recebe o valor de v2 e em seguida incrementa-se v2.

Incrementa-se também a posição do vetor aux.



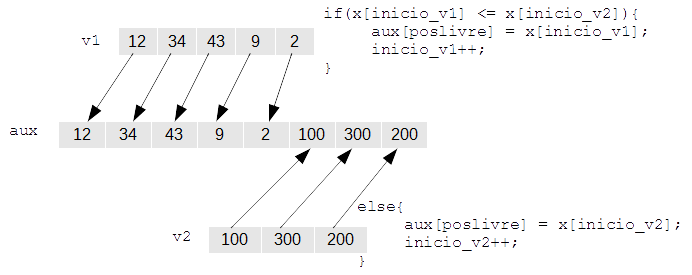
Saindo do primeiro laço e entrando no segundo.

O vetor aux irá receber os valores do vetor x para as primeiras posições até a central.

Em seguida entra em outro laço, onde o vetor aux irá receber os valores do vetor v2 para a posição central+1 até o fim.

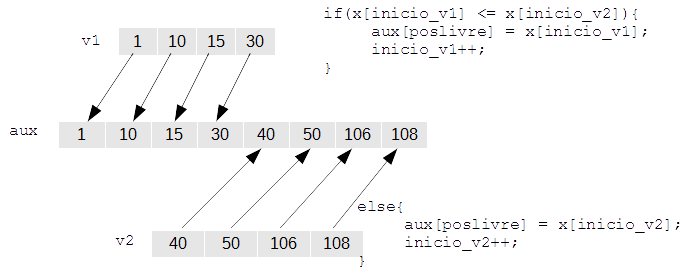
Finalizando com o último laço, onde o vetor original, x, irá receber todos os valores já intercalados em aux, por v1 e v2

* + ***caso 1:***
    - v1 = {12, 34, 43, 9, 2} e v2={100, 300, 200}





* + ***caso 2:***
    - v1 = {1, 10, 15, 30} e v2 = {40, 50, 106, 108}

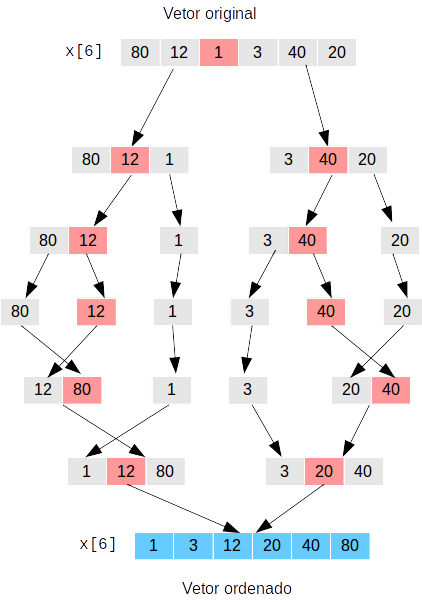


Conforme circulado nas posições do vetor **aux** no **caso 1**, vemos que os valor **9** e **2** estão em fora de ordem, sendo assim que apenas esse método de particionamento sozinho, não é suficiente para que seja ordenado dois vetores, se ambos não estiverem já previamente ordenados.

Já no **caso 2**, como ambos os vetores já estavam ordenados a função retorna o vetor auxiliar que por sua vez é atribuído ao vetor original encerrando assim a função **intercala**.

* analise os algorítmos disponibilizados no **GitHub**;
* para o ***MergeSort*** e o ***QuickSort***, faça seu teste de mesa;
  + crie uma ***tabela*** com todos os valores e teste a ordenação até que o vetor esteja ordenado;
  + crie um ***texto*** explicando o funcionamento de cada método de ordenação;
    - demonstre qual dos dois métodos foi mais eficiente e justifique;
* utilize o seguinte vetor:
  + *int v = {80, 12, 1, 3, 40, 20}*

**MergeSort**

No mergeSort consiste em achar o meio do vetor original e a partir desse meio, dividi-lo em duas partes, enquanto o inicio do vetor for menor que seu final.

void mergeSort(int x[], int inicio, int fim){  
 int meio;  
 if (inicio < fim){

meio = ((inicio + fim) / 2);

Sobrando assim duas metades de 3 posições cada, deve-se encontrar o meio novamente e ir dividindo até que todas as posições fiquem separadas uma das outras.

mergeSort(x, inicio, meio);

mergeSort(x, (meio + 1), fim);

intercala(x, inicio, fim, meio);

Em seguida deve começar a juntar as posições mas com os valores já ordenados.

Ao final, em que todas as partes já estarão reunidas, teremos o vetor ordenado.

Sua recursividade garante que nas primeiras passadas o vetor seja dividido até que todas as suas posições estejam separadas, e enquanto na volta todas as posições comece a se reunir de forma ordenada.

QuickSort