```cpp
#include<iostream>
#include<locale>
using namespace std;

// Nos parâmetros formais da função MERGESORT temos
// o vetor "VEC", o início e o fim dele.
// O mergeSort só será realizado se o vetor for sempre mais de 1.
void mergeSort (int VEC[], int inicio, int fim);

// A função que vai fazer a combinação dos vetores.
void merge(int VEC[], int inicio_vetor1, int fim_vetor1, int inicio_vetor2, int fim_vetor2);

// Principal
int main() {
    setlocale(LC_ALL, "portuguese");
    int i, n, vetor[10], final=0;
    cout << "\n Quantos elementos você quer inserir? : ";
    cin >> n;
    for (i=0; i<n; i++) {
        cout << "\n Posição " << i << ": ";
        cin >> vetor[i];
    }

    cout << " \n\n Foram inseridos " << i << " elementos: ";
    for (i=0; i<n; i++) {
        cout << vetor[i] << "  ";
    }


    mergeSort(vetor, 0, n-1);

    cout << " \n\n ORDENADOS DO MENOR PARA O MAIOR: ";
    for (i=0; i<n; i++) {
        cout << vetor[i] << "   ";
    }

}


void mergeSort (int VEC[], int inicio, int fim) {

     // O primeiro passo é dividir o vetor em duas partes.
     // (meio a meio) então crio a variável "meio" para me
     // auxiliar para dividir o vetor.
     int meio;

     // Se inicio for menor que fim, pegando "por exemplo um vetor com 8 posições",
     // seria 0 < 7.
     if (inicio < fim) {

        // Como estamos utilizando a questão da recursividade
        // Vai voltando a função e passando pela análise if(inicio < fim)
        // dividindo o vetor até chegar a 1.

        meio = (inicio + fim)/2;

        // Faz a ordenação da primeira parte ou primeira metade.
        // Por exemplo: vetor com 8 posições que vai de 0 a 7.
        // inicio = 0 e meio = 3
```

```
        mergeSort(VEC, inicio, meio);

        // Faz a ordenação da segunda parte ou segunda metade.
        // Por exemplo: vetor com 8 posições que vai de 0 a 7.
        // meio+1 = 3+1 = 4  e  fim=7.
        mergeSort(VEC, meio+1, fim);

        // Realizará a ordenação das duas partes.
        merge(VEC, inicio, meio, meio+1, fim);
    }

}


// Nesta função "void merge" é aonde ocorre a ordenação em si.
// É passado 5 parâmetros.
void merge(int VEC[], int inicio_vetor1, int fim_vetor1, int inicio_vetor2, int fim_vetor2){

    // temp é um vetor auxiliar que vai alocar todos
    // os elementos que forem ordenados.

    // (Nesse código está sendo utilizado um vetor para alocação,
    // também poderia ser utilizado um malloc).
    int temp[50];



    int iv1, iv2, k; // Variáveis auxiliares.

    iv1 = inicio_vetor1; // Início do primeiro vetor (VETOR 1).
    iv2 = inicio_vetor2; // Início do segundo vetor (VETOR 2).
    k = 0; //variável de controle que vai percorrer o vetor.

    // Enquanto o início do VETOR 1 for "menor e igual" ao fim do VETOR 1
    //                              E
    // o início do VETOR 2 for "menor e igual" ao fim do VETOR 2, faça...
    while (iv1 <= fim_vetor1 && iv2 <= fim_vetor2) {

        // Se o elemento do início do VETOR 1 for "menor"
        // que o elemento da início do VETOR 2 faça...
        if(VEC[iv1] < VEC[iv2]) {

            // Será guardado, o respectivo número que está no início do VETOR 1,
            // no vetor temporário temp[k++], e o k++, já deixa o vetor temp[k++]
            // para a próxima posição.

            // E após guardar o valor no temp[k++], o "VEC[iv1++]"
            // irá avançar para a próxima posição do VETOR 1.

            temp[k++] = VEC[iv1++];

         } else {  // Senão...

            // Será guardado, o respectivo número que está no início do VETOR 2,
            // no vetor temporário temp[k++], e o k++, já deixa o vetor temp[k++]
            // para a próxima posição.

            // E após guardar o valor no temp[k++], o "VEC[iv2++]"
            // irá avançar para a próxima posição do VETOR 2.
```

```
            temp[k++] = VEC[iv2++];

        }
    }

        // Copiar os elementos restantes do primeiro vetor.
        // Equanto o início do VETOR 1 for "menor e igual" ao
        // fim do VETOR 1 faça...
        while (iv1 <= fim_vetor1) {
              temp[k++] = VEC[iv1++];
        }


        // Copiar os elementos restantes do segundo vetor.
        // Equanto o início do VETOR 2 for "menor e igual" ao
        // fim do VETOR 2 faça...
        while (iv2 <= fim_vetor2) {
              temp[k++] = VEC[iv2++];
        }

        // Por fim...
        // Transferir os elementos que estão guardados no meu vetor
        // temporário temp[] para o nosso "vetor principal VEC[]".

        // Começa com o início do VETOR 1 e determina iv2 = 0.
        // E termina com o fim do VETOR 2.
        // Temos os incrementos iv1++ e iv2++ irá percorrer o for.
        for(iv1 = inicio_vetor1, iv2=0;  iv1 <= fim_vetor2;  iv1++, iv2++) {

            //Tranfere todos os elementos do vetor temp[] para o vetor VEC[]
            VEC[iv1] = temp[iv2];
        }

}
```