Laboratório 7
Exercício 1: montar um programa que crie diversos vetores com valores inteiros aleatórios e ordene cada um desses vetores com os métodos de ordenação vistos em teoria.

```c
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>

//Define o tam que o vetor tem
#define tamanho 7

//Algoritmo de ordenacao QuickSort
void QuickSort(int* v, int tam) {
    int j = tam, k;
    if (tam <= 1)
        return;
    else {
        int x = v[0];
        int a = 1;
        int b = tam - 1;
        do {
            while ((a < tam) && (v[a] <= x))
                a++;
            while (v[b] > x)
                b--;
            if (a < b) { // faz troca
                int temp = v[a];
                v[a] = v[b];
                v[b] = temp;
                a++;
                b--;
            }
            for (k = 0; k < j; k++)
                printf("%d ", v[k]);
            printf("\n");
        } while (a <= b);
        // troca pivo
        v[0] = v[b];
        v[b] = x;
        // ordena sub-vetores restantes
        QuickSort(v, b);
        QuickSort(&v[a], tam - a);
        for (k = 0; k < j; k++)
            printf("%d ", v[k]);
        printf("\n");
    }
}
```

```c
//Algoritmo de ordenacao InsertionSort
void InsertionSort(int* v, int tam) {
    int i, j, k, chave;
    for (j = 1; j < tam; j++) {
        chave = v[j];
        i = j - 1;
        while ((i >= 0) && (v[i] > chave)) {
            v[i + 1] = v[i];
            i--;
        }
        v[i + 1] = chave;
        for (k = 0; k < j; k++)
            printf("%d ", v[k]);
        printf("\n");
    }
    for (k = 0; k < j; k++)
        printf("%d ", v[k]);
    printf("\n");
}

//Algoritmo de ordenacao SelectionSort
void SelectionSort(int* v, int tam) {
    int i, j, k, min;
    for (i = 0; i < (tam - 1); i++) {
        min = i;
        for (j = (i + 1); j < tam; j++) {
            if (v[j] < v[min]) {
                min = j;
            }
        }
        if (i != min) {
            int swap = v[i];
            v[i] = v[min];
            v[min] = swap;
            for (k = 0; k < tamanho; k++)
                printf("%d ", v[k]);
            printf("\n");
        }
    }
}
 //Algoritmo de ordenacao BubbleSort
void BubbleSort(int* v, int tam) {
    int i, j = tam, k;
    int trocou;
    do {
        tam--;
        trocou = 0;
        for (i = 0; i < tam; i++) {
            if (v[i] > v[i + 1]) {
                int aux = 0;
                aux = v[i];
                v[i] = v[i + 1];
                v[i + 1] = aux;
                trocou = 1;
                for (k = 0; k < j; k++)
                    printf("%d ", v[k]);
                printf("\n");
            }
        }
    } while (trocou);
}
```

```c
//Algoritm
o de ordenacao HeapSort
void PercorreArvore(int* v, int raiz, int folha) {
    int percorreu, maxfolhas, temp, k;
    percorreu = 0;
    while ((raiz * 2 <= folha) && (!percorreu)) {
        if (raiz * 2 == folha)
            maxfolhas = raiz * 2;
        else if (v[raiz * 2] > v[raiz * 2 + 1])
            maxfolhas = raiz * 2;
        else
            maxfolhas = raiz * 2 + 1;
        if (v[raiz] < v[maxfolhas]) {
            temp = v[raiz];
            v[raiz] = v[maxfolhas];
            v[maxfolhas] = temp;
            raiz = maxfolhas;
        }
        else
            percorreu = 1;
        for (k = 0; k < tamanho; k++)
            printf("%d ", v[k]);
        printf("\n");
    }
}

void HeapSort(int* v, int tam) {
    int i, k, temp;
    for (i = (tam / 2); i >= 0; i--)
        PercorreArvore(v, i, tam - 1);
    for (i = tam - 1; i >= 1; i--) {
        temp = v[0];
        v[0] = v[i];
        v[i] = temp;
        PercorreArvore(v, 0, i - 1);
    }
}
```

```c
//Algoritmo de ordenacao MergeSort
void MergeSort(int* v, int inicio, int fim) {
    int i, j, k, meio, * t, z;
    if (inicio == fim)
        return;
    //ordenacao recursiva das duas metades
    meio = (inicio + fim) / 2;
    MergeSort(v, inicio, meio);
    MergeSort(v, meio + 1, fim);
    //intercalacao no vetor temporario t
    i = inicio;
    j = meio + 1;
    k = 0;
    t = (int*)malloc(sizeof(int) * (fim - inicio + 1));
    while (i < meio + 1 || j < fim + 1) {
        if (i == meio + 1) { //i passou do final da primeira metade, pegar v[j]
            t[k] = v[j];
            j++; k++;
        }
        else if (j == fim + 1) { //j passou do final da segunda metade, pegar v[i]
            t[k] = v[i];
            i++; k++;
        }
        else if (v[i] < v[j]) { //v[i]<v[j], pegar v[i]
            t[k] = v[i];
            i++; k++;
        }
        else { //v[j]<=v[i], pegar v[j]
            t[k] = v[j];
            j++; k++;
        }
        for (z = 0; z < tamanho; z++)
            printf("%d ", v[z]);
        printf("\n");
    }
    //copia vetor intercalado para o vetor original
    for (i = inicio; i <= fim; i++)
        v[i] = t[i - inicio];
    for (z = 0; z < tamanho; z++)
        printf("%d ", v[z]);
    printf("\n");
    free(t);
}
```

```
void main() {
    int i;
    int vbs[tamanho];
    int vqs[tamanho];
    int vis[tamanho];
    int vss[tamanho];
    int vhs[tamanho];
    int vms[tamanho];
    int vus[tamanho];
    clrscr();
    //Cria numeros aleatorios para os vetores
    for (i = 0; i < tamanho; i++) {
        vbs[i] = rand();
        vqs[i] = rand();
        vis[i] = rand();
        vss[i] = rand();
        vhs[i] = rand();
        vms[i] = rand();
        vus[i] = rand();
    }
    /*Ordenacao com BubbleSort
    printf("=============================================\n");
    printf("Ordenacao com BubbleSort: \n");
    printf("=============================================\n");
    printf("Vetor original: \n");
    printf("---------------------------------------------\n");
    for(i = 0; i < tamanho; i++){
       printf("%d ", vbs[i]);
    }
    printf("\n---------------------------------------------\n");
    printf("Passos da ordenacao: \n");
    printf("---------------------------------------------\n");
    BubbleSort(vbs, tamanho);
    */

    /*Ordenacao com QuickSort
    printf("\n\n\n=============================================\n");
    printf("Ordenacao com QuickSort: \n");
    printf("=============================================\n");
    printf("Vetor original: \n");
    printf("---------------------------------------------\n");
    for(i = 0; i < tamanho; i++){
       printf("%d ", vqs[i]);
    }
    printf("\n---------------------------------------------\n");
    printf("Passos da ordenacao: \n");
    printf("---------------------------------------------\n");
    QuickSort(vqs, tamanho);
    */

    /*Ordenacao com InsertionSort
    printf("\n\n\n=============================================\n");
    printf("Ordenacao com InsertionSort: \n");
    printf("=============================================\n");
    printf("Vetor original: \n");
    printf("---------------------------------------------\n");
    for(i = 0; i < tamanho; i++){
       printf("%d ", vis[i]);
    }
    printf("\n---------------------------------------------\n");
    printf("Passos da ordenacao: \n");
```

```
    printf("-------------------------------------------\n");
    InsertionSort(vis, tamanho);
    */

    /*Ordenacao com SelectionSort
    printf("\n\n\n==========================================\n");
    printf("Ordenacao com SelectionSort: \n");
    printf("==========================================\n");
    printf("Vetor original: \n");
    printf("-------------------------------------------\n");
    for(i = 0; i < tamanho; i++){
       printf("%d ", vss[i]);
    }
    printf("\n-------------------------------------------\n");
    printf("Passos da ordenacao: \n");
    printf("-------------------------------------------\n");
    SelectionSort(vss, tamanho);
    */

    /*Ordenacao com HeapSort
    printf("\n\n\n==========================================\n");
    printf("Ordenacao com HeapSort: \n");
    printf("==========================================\n");
    printf("Vetor original: \n");
    printf("-------------------------------------------\n");
    for(i = 0; i < tamanho; i++){
       printf("%d ", vhs[i]);
    }
    printf("\n-------------------------------------------\n");
    printf("Passos da ordenacao: \n");
    printf("-------------------------------------------\n");
    HeapSort(vhs, tamanho);
    */

    /*Ordenacao com MergeSort
    printf("\n\n\n==========================================\n");
    printf("Ordenacao com MergeSort: \n");
    printf("==========================================\n");
    printf("Vetor original: \n");
    printf("-------------------------------------------\n");
    for(i = 0; i < tamanho; i++){
       printf("%d ", vms[i]);
    }
    printf("\n-------------------------------------------\n");
    printf("Passos da ordenacao: \n");
    printf("-------------------------------------------\n");
    MergeSort(vms, 0, tamanho-1);
    */

    getchar();
}
```