**TRABALHO DE ESTRUTURA DE DADOS EM C**

Alunos: Rafael Fernandes Fernandez De Oliveira E Vinicius Alarcon Vilela

Link GitHub: <https://github.com/AlarconVinicius/5-Tipos-de-Ordenacao-em-C.git>

SEGUE OS CÓDIGOS DAS 5 ORDENAÇÕES

**OBS.:** Os comentários estão no próprio código!

**BUBBLE SORT**

#include <stdio.h>

#include <stdlib.h>

//CÓDIGO BUBBLESORT

*int* main(){

*int* numeros[20]; //numero de posições

*int* contador1, aux, contador2, total\_count;

  printf("=-=-=-=BUBBLE\_SORT=-=-=-=\n");

  printf("\nQuantos numeros deseja inserir?\n");

  scanf("%d", &total\_count);

  printf("Digite %d numeros inteiros que devem ser ordenados.\n", total\_count);

  //pegar a quantidade de numeros e guardar nas posições

  for (contador1 = 0; contador1 < total\_count; contador1++){

  scanf("%d", &numeros[contador1]);

  }

  //Ordenação do Bubblesort

  for (contador2 = 1; contador2 < total\_count; contador2++){ //pega o primeiro item com o ultimo

    for (contador1 = 0; contador1 < total\_count - 1; contador1++){ //ver se o numero é menor do que o proximo numero

      if (numeros[contador1] > numeros[contador1 + 1]){ //comparar o numero com o proximo para ver se é maior

        //trocar numeros

        aux = numeros[contador1];

        numeros[contador1] = numeros[contador1 + 1];

        numeros[contador1 + 1] = aux;

      }

    }

  }

  //exibir os elementos ordenados

  printf("\nNumeros em ordem crescente:\n");

  for (contador1 = 0; contador1 < total\_count; contador1++) {

          printf("%d\n", numeros[contador1]);

  }

  printf("\n");

  return 0;

}

=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-====================

**SELECTION SORT**

#include <stdio.h>

#include <conio.h>

//CÓDIGO SELECTIONSORT

*void* main(){

*int* total\_count, contador1, contador2, menor, aux;

*int* numeros[20];

    printf("=-=-=-=SELECTION\_SORT=-=-=-=\n");

    printf("\nQuantos numeros deseja inserir?\n");

    scanf("%d", &total\_count);

    printf("Por favor digite %d de inteiros que devem ser ordenados.\n", total\_count);

    for (contador1 = 0; contador1 < total\_count; contador1++){

        scanf("%d", &numeros[contador1]);

    }

        //percorre toda a lista até encontrar o menor item entre eles

    for (contador1 = 0; contador1 < total\_count - 1; contador1++){

        menor = contador1;

        //se o item for menor do que o item da lista, ele troca de posição. isso acontece com CADA item da lista

        for(contador2 = contador1 + 1; contador2 < total\_count; contador2++){

            if(numeros[menor] > numeros[contador2])

                menor = contador2;

        }

        //pega o menor item e vai comparando com os outros

        if (menor != contador1){//garantir que não irá fazer uma troca da mesma posição

            aux = numeros[contador1];

            numeros[contador1] = numeros[menor];

            numeros[menor] = aux;

        }

    }

        //exibir os elementos ordenados

    printf("\nNumeros em ordem crescente:\n");

    for (contador1 = 0; contador1 < total\_count; contador1++){

        printf("%d\n", numeros[contador1]);

    }

    getch();

}

=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

**QUICK SORT**

#include <stdio.h>

#include <stdlib.h>

//CÓDIGO QUICKSORT

*void* quick\_sort(*int*[], *int*, *int*);

*int* main(){

*int* numeros[50], total\_count, contador;

    printf("=-=-=-=QUICK\_SORT=-=-=-=\n");

    printf("\nQuantos numeros deseja inserir? ");

    scanf("%d", &total\_count);

    printf("Por favor digite %d de inteiros que devem ser ordenados.\n", total\_count);

    //pegar a quantidade de numeros e guardar nas posições

    for (contador = 0; contador < total\_count; contador++) {

        scanf("%d", &numeros[contador]);

    }

    quick\_sort(numeros, 0, total\_count - 1);

    printf("Numeros em ordem crescente:\n");

    for (contador = 0; contador < total\_count; contador++){

        printf("%d\n", numeros[contador]);

    }

    printf("\n");

    return 0;

}

*void* quick\_sort(*int* *numeros*[], *int* *inicio*, *int* *fim*){

*int* pivo, esquerda, direita, aux;

    //comparações

    if(*inicio* < *fim*){

        pivo = *inicio*; //o numero que vai ser verificado

        esquerda = *inicio*;

        direita = *fim*;

        while (esquerda < direita){

            //percore o valor pela direita

            while (*numeros*[esquerda] <= *numeros*[pivo] && esquerda <= *fim*){

                esquerda++; //enquanto o vetor da esquerda for igual ou meno que o pivo, avança UM

            }

            //percorre o valor da direita

            while (*numeros*[direita] > *numeros*[pivo] && direita >= *inicio*){

                direita--;//enquanto o vetor da direita for maior que o pivo, recua UM

            }

            // se esquerda for menor que direita, fazer a troca dos numeros

            if (esquerda < direita){

                aux = *numeros*[esquerda];

*numeros*[esquerda] = *numeros*[direita];

*numeros*[direita] = aux;

            }

        }

        //os numeros menores vão para esquerda e os maiores para direita, assim fazendo as comparações

        aux = *numeros*[direita];

*numeros*[direita] = *numeros*[pivo];

*numeros*[pivo] = aux;

        quick\_sort(*numeros*, *inicio*, direita - 1);

        quick\_sort(*numeros*, direita + 1, *fim*);

    }

}

=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

**MERGE SORT**

#include <stdio.h>

#include <stdlib.h>

//CÓDIGO MERGESORT

*void* merge\_sort(*int* *numeros*[], *int* *couter1*, *int* *counter2*);

*void* executar\_merge(*int* *numeros*[], *int* *counter11*, *int* *counter12*, *int* *counter22*, *int* *counter21*);

*int* main(){

*int* numeros[100], total\_count, contador1;

    printf("=-=-=-=,MERGE\_SORT=-=-=-=\n");

    printf("Quantos numeros deseja inserir? \n");

    scanf("%d", &total\_count);

    printf("Por favor digite %d de inteiros que devem ser ordenados.\n", total\_count);

    //pegar a quantidade de numeros e guardar nas posições

    for(contador1 = 0; contador1 < total\_count; contador1++)

        scanf("%d", &numeros[contador1]);

    //exibir os elementos ordenados

    merge\_sort(numeros, 0, total\_count - 1);

    printf("\nNumeros em ordem crescente:\n");

    for(contador1 = 0; contador1 < total\_count; contador1++)

        printf("%d\n", numeros[contador1]);

    return 0;

}

*void* merge\_sort(*int* *numeros*[], *int* *contador1*, *int* *contador2*){ //montar a operação

*int* mid;

    /\*divide os numeros em pequenas partes para fazer a comparação. assim que é comparada,

    realiza uma ação, a troca.\*/

    if(*contador1* < *contador2*){

        mid = (*contador1* + *contador2*) / 2; //divide os dados para fazer organizações pequenas para se tornar mais rápida

        merge\_sort(*numeros*, *contador1*, mid);

        merge\_sort(*numeros*, mid + 1, *contador2*);

        executar\_merge(*numeros*, *contador1*, mid, mid + 1, *contador2*);

    }

}

*void* executar\_merge(*int* *numeros*[], *int* *contador11*, *int* *contador12*, *int* *contador22*, *int* *contador21*){ //executa a operação

*int* temp\_num[50];

*int* c1, c2, c3;

    c1 = *contador11*; //comparação dos numeros

    c2 = *contador22*;

    c3 = 0;

    //enquanto o numero for maior, ele vai comparando para achar o menor

    while(c1 <= *contador12* && c2 <= *contador21*){ //comparação

        if(*numeros*[c1] < *numeros*[c2]) //se o primeiro numero for menor que o segundo, realiza a troca

            temp\_num[c3++] = *numeros*[c1++];

        else

            temp\_num[c3++] = *numeros*[c2++]; //senão, mantem a posição

    }

    //comparar com os demais numeros separados e mudar de posição

    //só terminará o while quando todos estiverem em suas posições

    while(c1 <= *contador12*)

        temp\_num[c3++] = *numeros*[c1++];

    while(c2 <= *contador21*)

        temp\_num[c3++] = *numeros*[c2++];

    for(c1 = *contador11*, c2 = 0; c1 <= *contador21*; c1++, c2++)

*numeros*[c1] = temp\_num[c2];

}

=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=--=-=-=-=-=-=-=-

**INSERTION SORT**

#include <stdio.h>

#include <stdlib.h>

//CÓDIGO INSERTIONSORT

*int* main(){

*int* contador1,contador2, total\_count, temp\_num, numeros[100];

    printf("=-=-=-=,INSERTION\_SORT=-=-=-=\n");

    printf("Quantos numeros deseja inserir? \n");

    scanf("%d", &total\_count);

    printf("Por favor digite %d de inteiros que devem ser ordenados.\n", total\_count);

    //pegar a quantidade de numeros e guardar nas posições

    for(contador1 = 0; contador1 < total\_count; contador1++){

        scanf("%d", &numeros[contador1]);

    }

    //comparar o item atual com o intem a sua esquerda

    //se o total do contador for maior ou igual ao numero do item atual, ele troca de posição

    for(contador1 = 1; contador1 <= total\_count - 1; contador1++){

        temp\_num = numeros[contador1];

        contador2 = contador1 - 1;

        //enquanto o item da esquerda for maior, ele vai mudar a posição

        while((temp\_num < numeros[contador2]) && (contador2 >= 0)){ //comparar os numeros

            numeros[contador2 + 1] = numeros[contador2]; //enquanto o item da esquerda for maior, ele vai mudar a posição

            contador2 = contador2 - 1;

        }

        numeros[contador2 + 1] = temp\_num;

    }

    //exibir os elementos ordenados

    printf("\nNumeros em ordem crescente:\n");

    for(contador1 = 0; contador1 < total\_count; contador1++){

        printf("%d\n", numeros[contador1]);

    }

    return 0;

}