Sumário

1. Introdução:

Cotidianamente é encontrado problemas de ordenação básicos em grandes empresas e em atividades de práticas diárias. Essa ordenação pode ser numérica, dada uma lista, na qual a ordenação é mais simples. Porém, há também a falta de ordenação por parte das palavras, em que deixa algumas vezes as estruturas dos dados complexas. Com isso, neste exercício, é proposto criar um algoritmo de ordenação Quicksort para a resolução de uma lista de palavras que não está ordenada. A partir da resolução disso, será possível de modo mais prático ordenar um conjunto de arquivos que contenham diversos valores declarados dentro de um array e gerar um arquivo de saída com as palavras em ordem.

GitHub:

https://github.com/marioprneto/Trabalho-de-Estrutura-de-dados > TP02

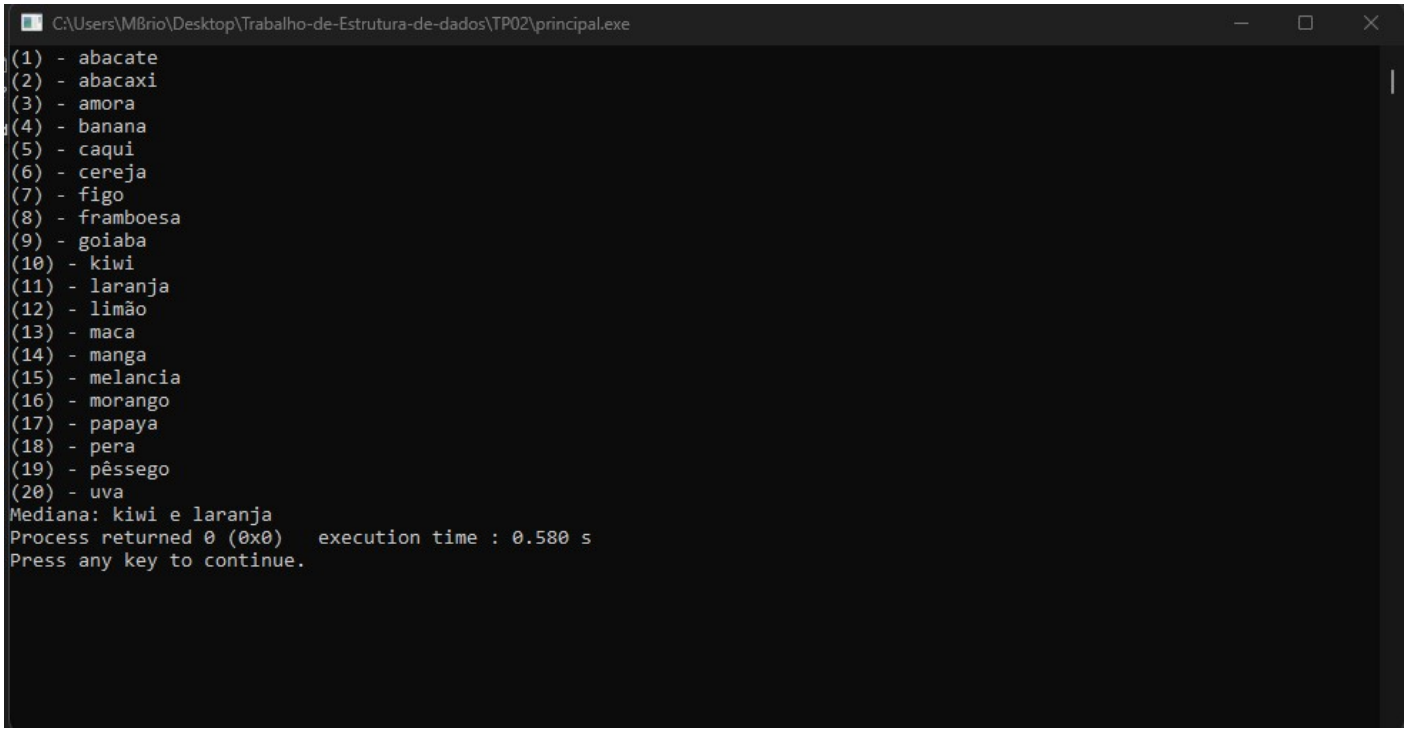
https://github.com/marioprneto/Trabalho-de-Estrutura-de-dados/tree/main/TP02

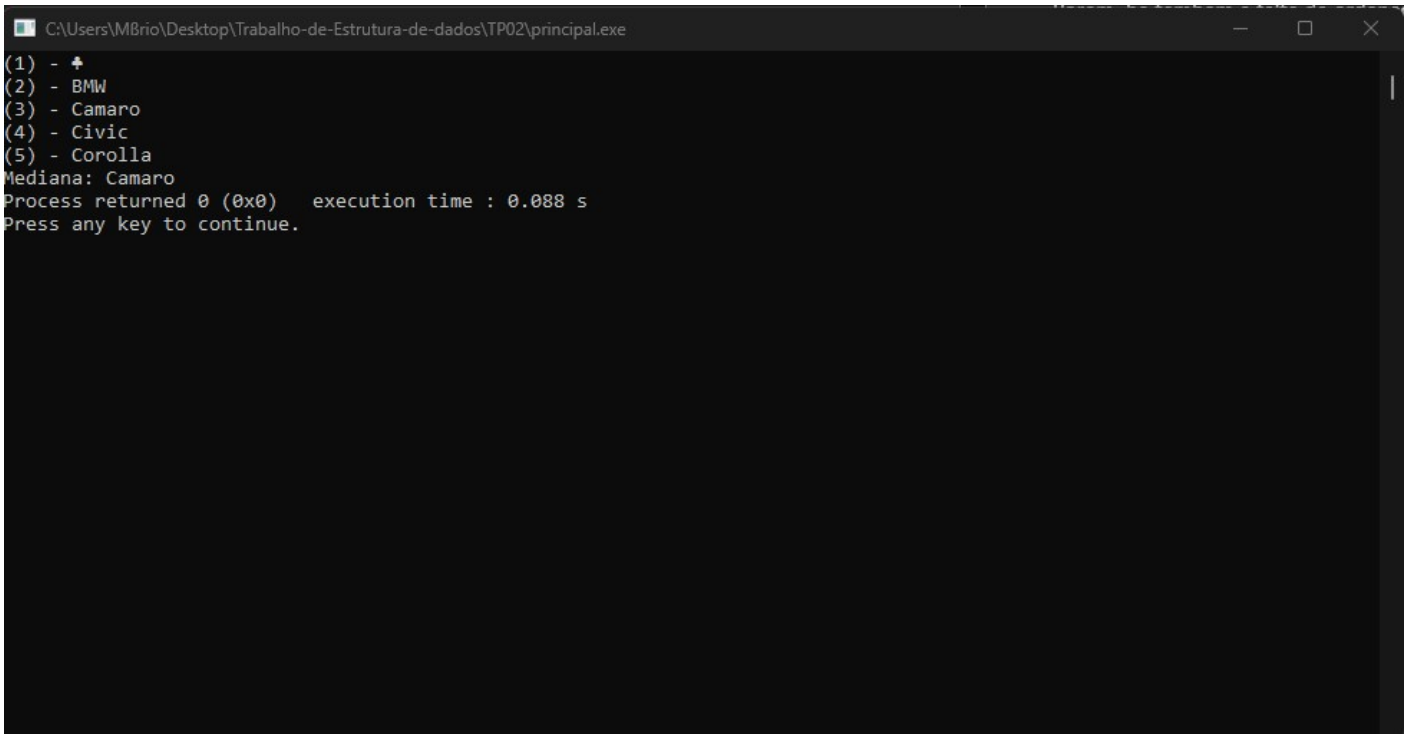
2. Implementação:

Em primeira instância, cabe salientar que diferentemente do exercício anterior, neste exercício iniciei meu código com a biblioteca string.h e locale.h, além da stdio.h de costume. A locale.h utilizei para fazer a acentuação das palavras que eram contidas no array e no printf também. Já a string.h utilizei para a função strcmp, que irei explicar mais adiante. Após declarar o idioma das palavras como português, declarei o array com as palavras pré-definidas. Criei duas variáveis do tipo inteiro para poder utilizá-las como contador. Após isso, criei uma variável chamada arquivoSaida do tipo FILE para gravar o arquivo com o resultado. Em seguida, fiz um diferencial: Criei uma variável do tipo size_t chamada tamanho que recebe o sizeof do array (arr) e divide pela posição 0, obtendo assim o length do array. Esse tamanho utilizei como parâmetro da estrutura de repetição for, para percorrer todo o array, posição por posição. Ao percorrer a primeira posição, para comparar ela com as demais, tive de criar uma outra estrutura de repetição for com os mesmos parâmetros para percorrer novamente a lista, e comparar assim, posição por posição, utilizando a estrutura strcmp. Essa estrutura é da biblioteca <string.h> e permite a comparação das strings. Ela foi criada em uma variável chamada resultado, e vai verificar a posição [i] do vetor, em que [i] representa o primeiro lanço, comparando com a posição [j] do vetor, em que [j] representa o segundo laço. Após verificar, irá retornar o resultado como maior que 0 para caso a posição [j] estiver na ordem incorreta e devesse estar antes da posição [i]. Após isso, incremento um ao contador das comparações, para cada comparação que for feita, ele irá somar +1. Em seguida, ele irá verificar se o resultado é maior que zero, caso seja, fazer a troca das posições adequadamente e incrementar um ao contador de trocas. Por fim, foi necessário abrir o arquivoSaida com fopen, declarando como parâmetros seu nome e a opção write, visto que vamos escrever nele. Após isso, com auxílio da função fprintf, escrevi no arquivoSaida o número de trocas, número de comparações e o texto fixo “Vetor ordenado”, ao qual quebrei uma linha após a escrita desse texto fixo. E então, fiz um novo laço de repetição para percorrer todo o vetor novamente, e imprimir cada elemento de maneira ordenada no console, e gravar no arquivo cada elemento, da mesma forma que no console. Para verificar a mediana, fiz uma verificação se as posições do vetor são pares ou ímpares, visto que a mediana quando for par, será os dois termos do meio dividido por dois, mas como são strings, serão apenas os dois termos do meio. Caso forem ímpares, exibirá apenas o termo central. Visto que não foi especificado como a mediana deveria ser exibida, se era por arquivo ou no console, decidi implementar em ambos. E encerrei meu algoritmo com o return 0;.

3. Testes

Para realizar o teste deste aplicativo, além do exemplo fornecido fui atrás de outros vetores de textos que necessitavam de ordenação com tamanhos diferentes, visto que fiz um método abrangente para diversas situações.





4. Conclusão

Após este árduo trabalho, pode entender-se a utilidade de um algoritmo de ordenação de palavras. O principal problema foi a implantação da lógica, pois de resto foi mais tranquilo. Uma possível melhoria, assim como no trabalho anterior foi mencionado, é a separação dos métodos e deixar a função main para apenas chamar as funções. Mas para questões de facilidade, decidi por implementar tudo dentro da main neste trabalho.

Referências

https://www.programiz.com/c-programming/library-function/string.h/strcmp

Anexos

```c
#include <stdio.h>
#include <string.h>
#include <locale.h>

int main(){
    setlocale(LC_ALL,"Portuguese");
    char *arr[20] = {"maca", "banana", "pera", "uva", "laranja", "abacaxi", "limão", "manga", "abacate", "kiwi", "cereja", "morango", "pêssego", "goiaba", "melancia", "framboesa", "amora", "caqui", "figo", "papaya"};
    int contador = 0, contadorComp = 0;

    FILE *arquivoSaida;

    size_t tamanho = sizeof(arr) / sizeof(arr[0]);

    for(int i = 0; i<=tamanho; i++){
        for(int j = 0; j<=tamanho; j++){
            int resultado = strcmp(arr[i], arr[j]);
            contadorComp++;
            if (resultado<0){
                char *temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
                contador++;
            }
        }
    }

    arquivoSaida = fopen("ResultadoOrdenacao.txt","w");

    fprintf(arquivoSaida, "Número de trocas: %d\nNúmero de comparações: %d\nVetor ordenado:\n",contador,contadorComp);

    for (int k=0; k<tamanho;k++){
        printf("(%d) - %s\n",k+1,arr[k]);
        fprintf(arquivoSaida, "(%d) - %s\n",k+1,arr[k]);
    }
```

```c
    for (int k=0; k<tamanho;k++){
        printf("(%d) - %s\n",k+1,arr[k]);
        fprintf(arquivoSaida, "(%d) - %s\n",k+1,arr[k]);
    }

    if(tamanho%2==0){
        printf("Mediana: %s e %s", arr[(tamanho/2) - 1], arr[tamanho/2]);
        fprintf(arquivoSaida, "Mediana: %s e %s", arr[(tamanho/2) - 1], arr[tamanho/2]);
    } else{
        printf("Mediana: %s", arr[(tamanho/2)]);
        fprintf(arquivoSaida, "Mediana: %s", arr[(tamanho/2)]);
    }

    return 0;
}
```

principal.c

#include <stdio.h>

#include <string.h>

#include <locale.h>

```
int main(){

   setlocale(LC_ALL,"Portuguese");

   char *arr[20] = {"maca", "banana", "pera", "uva", "laranja", "abacaxi", "limão", "manga", "abacate", "kiwi",
"cereja", "morango", "pêssego", "goiaba", "melancia", "framboesa", "amora", "caqui", "figo", "papaya"};

   int contador = 0, contadorComp = 0;



   FILE *arquivoSaida;



   size_t tamanho = sizeof(arr) / sizeof(arr[0]);



   for(int i = 0; i<=tamanho; i++){

      for(int j = 0; j<=tamanho; j++){

         int resultado = strcmp(arr[i], arr[j]);

         contadorComp++;

         if (resultado<0){

            char *temp = arr[i];

            arr[i] = arr[j];

            arr[j] = temp;

            contador++;

         }

      }

   }

   arquivoSaida = fopen("ResultadoOrdenacao.txt","w");

   fprintf(arquivoSaida, "Número de trocas: %d\nNúmero de comparações: %d\nVetor
ordenado:\n",contador,contadorComp);

   for (int k=0; k<tamanho;k++){
```

```
        printf("(%d) - %s\n",k+1,arr[k]);
        fprintf(arquivoSaida, "(%d) - %s\n",k+1,arr[k]);
    }
    if(tamanho%2==0){
        printf("Mediana: %s e %s", arr[(tamanho/2) - 1], arr[tamanho/2]);
        fprintf(arquivoSaida, "Mediana: %s e %s", arr[(tamanho/2) - 1], arr[tamanho/2]);
    } else{
        printf("Mediana: %s", arr[(tamanho/2)]);
        fprintf(arquivoSaida, "Mediana: %s", arr[(tamanho/2)]);
    }
    return 0;
}
```