

Sumário

1. Introdução:	2
2. Implementação:	2
3. Testes	2
4. Conclusão	3
Referências	3
Anexos	3
Quicksort.c	3

1. Introdução:

Este trabalho é uma adaptação do código Quicksort, em que deve-se ordenar o vetor de palavras inserido a seguir: `char *arr[20] = {"maca", "banana", "pera", "uva", "laranja", "abacaxi", "limão", "manga", "abacate", "kiwi", "cereja", "morango", "pêssego", "goiaba", "melancia", "framboesa", "amora", "caqui", "figo", "papaya"};`. Além de ordená-lo, deve-se indicar a mediana, contar o número de trocas e comparações que o código fez e criar um arquivo .txt contendo o vetor ordenado e o número de comparações e trocas que ele fez.

GitHub:

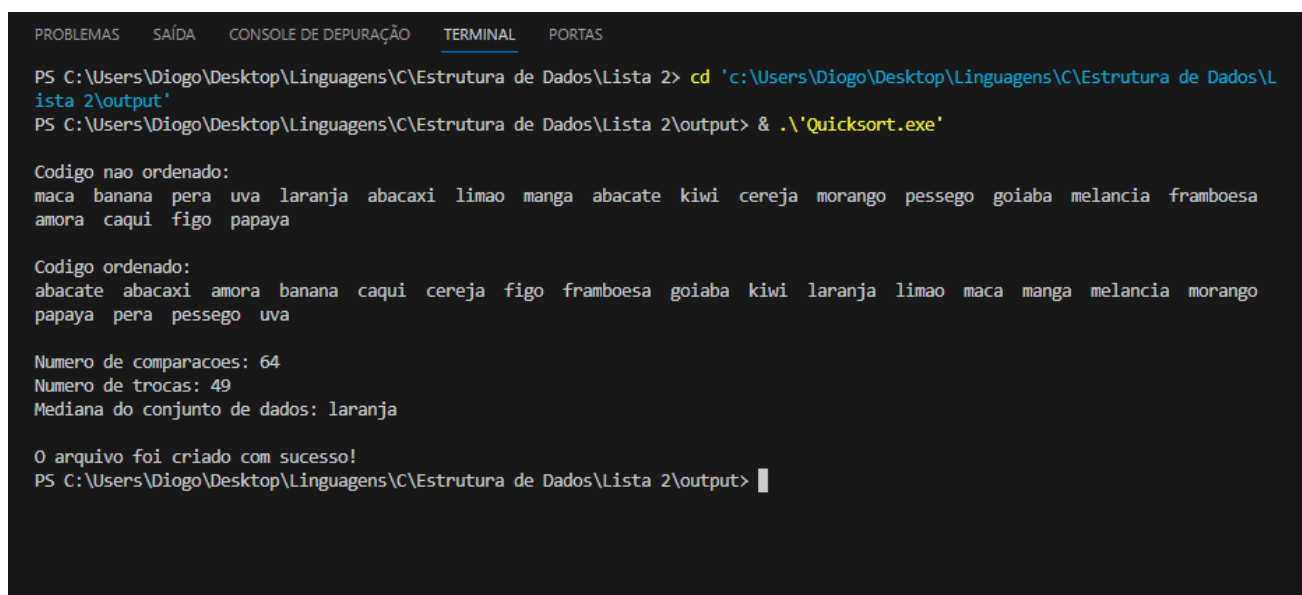
https://github.com/Diogo-Fr/Estrutura_de_Dados.git

2. Implementação:

O código começa com as bibliotecas `stdio`, `stdlib` e `string`, logo depois foi declarado 2 variáveis globais: `comp` e `swaps` (número de comparações e de trocas), logo depois criamos a função `swap` que serve para trocar dois ponteiros para strings, depois criamos a função `partição` que serve para fazer comparações dentro do código utilizando um pivô, caso o número ou palavra selecionada for menor ou igual ao pivô ele faz uma troca, a seguir chegamos a função `Quicksort` que ordena o vetor e o divide em partições menores e chama a função `partição` para encontrar o pivô, por fim a função principal, a `Main`, onde tem as variáveis: `char arr[20]`, `int n` e `int arr_mediana`. Foi criado um `for` para imprimir o vetor não ordenado, depois a função `Quicksort` foi chamada para ordenar o vetor e o vetor foi imprimido, depois foi criada a variável `char mediana` para poder calcular a mediana do vetor e no final foi imprimido o número de trocas, comparações e a mediana. Para finalizar o código foi criado um arquivo com o nome `VetorOrdenado.txt`, onde foi inserido o número de comparações, trocas e o próprio vetor ordenado.

3. Testes

Nessa imagem tem o vetor não ordenado, logo embaixo tem o vetor ordenado junto com o número de comparações, trocas, a mediana e uma mensagem onde diz que o arquivo .txt foi criado com sucesso.



```
PROBLEMAS  SAÍDA  CONSOLE DE DEPURAÇÃO  TERMINAL  PORTAS

PS C:\Users\Diogo\Desktop\Linguagens\C\Estrutura de Dados\Lista 2> cd 'c:\Users\Diogo\Desktop\Linguagens\C\Estrutura de Dados\Lista 2\output'
PS C:\Users\Diogo\Desktop\Linguagens\C\Estrutura de Dados\Lista 2\output> & .\'Quicksort.exe'

Codigo nao ordenado:
maca banana pera uva laranja abacaxi limao manga abacate kiwi cereja morango pessego goiaba melancia framboesa
amora caqui figo papaya

Codigo ordenado:
abacate abacaxi amora banana caqui cereja figo framboesa goiaba kiwi laranja limao maca manga melancia morango
papaya pera pessego uva

Numero de comparacoes: 64
Numero de trocas: 49
Mediana do conjunto de dados: laranja

O arquivo foi criado com sucesso!
PS C:\Users\Diogo\Desktop\Linguagens\C\Estrutura de Dados\Lista 2\output> |
```

4. Conclusão

O trabalho apresenta um código fornecido pelo professor Marcelo que foi modificado para ordenar vetores, porém com uma leve diferença, agora teria que ordenar strings (o código original era com inteiros). A implementação não tão foi difícil, porque não tive que mexer no código, somente adicionar coisas, como por exemplo o strcmp, então isso deixou as coisas mais fáceis. Foi um aprendizado interessante para conhecer melhor o funcionamento do Quicksort e entender a sua eficiência para ordenar vetores.

Referências

<https://www.youtube.com/watch?v=wx5juM9bbFo>

Anexos

Quicksort.c

```
#include<stdio.h>

#include<stdlib.h>

#include<string.h>


// Essas duas são variáveis globais.

int comp = 0;

int swaps = 0;


void swap(char **A, char **B){

    char *Temp = *A;

    *A = *B;

    *B = Temp;

    // Contador de trocas.

    swaps++;

}
```

```
int Particao(char *Vetor[], int inf, int sup){  
  
    char *Pivo = Vetor[(sup)];  
  
    int i = inf - 1;  
  
  
    for(int j = inf; j <= sup - 1; j++){  
  
        // Contador de comparações.  
  
        comp++;  
  
        if(strcmp(Vetor[j], Pivo) <= 0){  
  
            i++;  
  
            swap(&Vetor[i], &Vetor[j]);  
  
        }  
  
    }  
  
    swap(&Vetor[i + 1], &Vetor[sup]);  
  
    return(i + 1);  
  
}
```

```
void Quicksort(char *Vetor[], int inf, int sup) {  
  
    if(inf < sup) {  
  
        int P = Particao(Vetor, inf, sup);  
  
        Quicksort(Vetor, inf, P - 1);  
  
        Quicksort(Vetor, P + 1, sup);  
  
    }  
  
}
```

```
int main() {
```

// Eu retirei os acentos.

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

```
int n = sizeof(arr) / sizeof(arr[0]);
```

```
int arr_mediana = n/2;
```

```
printf("\nCodigo nao ordenado:\n");
```

```
for(int i = 0; i < n; i++) printf("%s ", arr[i]);
```

```
printf("\n");
```

```
Quicksort(arr, 0, n - 1);
```

```
printf("\nCodigo ordenado:\n");
```

```
for(int i = 0; i < n; i++) printf("%s ", arr[i]);
```

```
char *mediana;
```

```
if (n % 2 == 1) {
```

```
    mediana = arr[arr_mediana];
```

```
} else {
```

```
    char *mediana1 = arr[arr_mediana];
```

```
    char *mediana2 = arr[arr_mediana - 1];
```

```
// Malloc é uma conversão de tipo, ele vai garantir que o ponteiro retornado seja do tipo char.
```

```
mediana = (char *)malloc(strlen(mediana1) + strlen(mediana2) + 1);
```

```
strcpy(mediana, mediana1);
```

```
// strcat anexa uma string a outra.
```

```
strcat(mediana, "/");
```

```
strcat(mediana, mediana2);
```

```
}
```

```
printf("\n\n");

printf("Numero de comparacoes: %d\n", comp);

printf("Numero de trocas: %d\n", swaps);

printf("Mediana do conjunto de dados: %s\n\n", mediana);
```

```
FILE *arquivo;
```

```
arquivo = fopen("VetorOrdenado.txt", "w");
```

```
if(arquivo == NULL){

    printf("Arquivo não criado!");

    exit(1);

}
```

```
fprintf(arquivo, "Número de comparações: %d\n", comp);
```

```
fprintf(arquivo, "Número de trocas: %d\n", swaps);
```

```
fprintf(arquivo, "Vetor ordenado:\n");
```

```
for(int i = 0; i < n; i++){

    fprintf(arquivo, "(%i) %s\n", i+1, arr[i]);

}
```

```
printf("O arquivo foi criado com sucesso!");
```

```
fclose(arquivo);
```

```
    return 0;  
}
```