Sumário

1. Introdução:

O desenvolvimento da atividade busca aplicar os conhecimentos adquiridos em sala em cima de uma situação problema criada pelo professor.

GitHub:

2. Implementação:

(Colocar os detalhes de implementação. A estrutura de dados utilizada, tipos de variáveis, funções e procedimentos criados, funcionamento do programa principal além das decisões de implementação e de informações técnicas como o compilador / ambiente utilizado, como rodar o seu trabalho, etc)

O programa foi desenvolvido utilizando funções para facilitar o processo de diminuir a complexidade, permitindo um código mais limpo e com diversas anotações.

A primeira é função é uma função de troca chamada *SWAP* que utiliza ponteiros para trocar o conteúdo de duas variáveis.

A segunda função (em ordem de exeução) é função chamada particionar, função mais importante do método de ordenação quicksort onde os elementos são separados em grupos menores e ordenados dentro desse grupo e depois no geral, esse método seleciona um pivô (que é um número, nesse caso o central, utilizado para dividir os grupos entre maiores que o pivô e menores que o mesmo). Dentro dessa função é onde ficam os contadores de comparações e trocas e função de troca.

A terceira função é o quicksort em si que funciona basicamente chamando a função de particionar e a si mesmo até que tudo tenha sido ordenado, ou seja, a condição de elemento anterior > elemento posterior, os quais no caso são comparações lexicográficas(ordem alfabética) utilizando strcmp.

3. Testes

A lógica do projeto foi consideravelmente fácil de se desenvolver com base no arquivo *Quicksort.c* fornecido pelo professor, necessitando apenas da adaptação de int para string. O uso de ponteiros sempre se mostra um pouco problemático mas é notável sua necessidade quando trabalhando com projetos mais complexos. Os testes inciais funcionaram mais para permitir a execução do código sem erros, o que demorou bastante devido a diversos detalhes de pontuação ou a falta dela.

O erro mais demorado para resolver foi :

```
Ordenador.c:15:20: warning: implicit declaration of function 'particionar' [-Wimplicit-function-declaration]
   15 |         int pivo = particionar(arr, inicio, fim, comparacoes, trocas);
      |                    ^~~~~~~~~~~~
Ordenador.c: In function 'main':
Ordenador.c:66:15: warning: passing argument 1 of 'quicksort' from incompatible pointer type [-Wincompatible-pointer-types]
   66 |     quicksort(arr, 0, 19, &comparacoes, &trocas);
      |               ^~~
```

Onde eu não havia declara o char arr, que era a lista das frutas com dois ** e foi necessário algumas horas para entender que deveriam ser usados dois ** pois era um ponteiro apontando para um ponteiro.

Outro “erro” que não atrapalhava o processo mas continuou não funcionando até o final foi o fato de quando executado a ordenação a palavra “morango” aparecia antes da execução, o que se dava pelo morango ter sido escolhido como pivo, e o programa não entendia o que fazer pois quando comparado morango com morango não havia uma condicional do que fazer já que as únicas opções era > ou <. Mesmo sabendo qual era o problema quando eu alterava o código adicionando uma condicional para ignorar quando a strcmp não retornasse nem 1 nem 0 o programa parava de funcionar, então não foi resolvido na versão final.

4. Conclusão

Foi interessante ver a aplicação de um algoritmo tão utilizado em C mesmo com as dificuldades em aplicação da linguagem. Acredito que a lógica esteja tranquila uma vez compreendida como o método deve funcionar, porém os detalhes da linguagem sempre se mostram ser uma complicação talvez por falta de atenção nas aulas.

Referências

StackOverflow.com

ChatGpt

Anexos

Ordenador.c

```c
#include <stdio.h>

#include <stdlib.h>

#include <string.h>


// Função para trocar duas palavras em um vetor(?)

void trocar(char **a, char **b) {

    char *temp = *a;

    *a = *b;

    *b = temp;

}
```

```
// Função Quicksort para ordenar o vetor de palavras
void quicksort(char *arr, int inicio, int fim, int *comparacoes, int *trocas) {
    if (inicio < fim) {
        int pivo = particionar(arr, inicio, fim, comparacoes, trocas);

        quicksort(arr, inicio, pivo, comparacoes, trocas);
        quicksort(arr, pivo + 1, fim, comparacoes, trocas);
    }
}

// Função para encontrar o pivô e particionar o vetor
//particionar para comparar em conjuntos de numeros menores e ordena-lo de forma geral
int particionar(char **arr, int inicio, int fim, int *comparacoes, int *trocas) {
    // Escolher o elemento central como pivô
    int meio = (inicio + fim) / 2;
    char *pivo = arr[meio];

    int i = inicio;
    int j = fim;

    while (1) {
        while (strcmp(arr[i], pivo) < 0) {
            (*comparacoes)++;
            i++;
        }
        while (strcmp(arr[j], pivo) > 0) {
```

```
        (*comparacoes)++;
        j--;
    }
    if (i >= j) {
        break;
    }

    (*trocas)++;
    trocar(&arr[i], &arr[j]);
    i++;
    j--;
  }

  return j;
}

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

  int comparacoes = 0; // Contador de comparações
  int trocas = 0; // Contador de trocas
```

```
    // Ordenar o vetor de palavras usando o Quicksort
    quicksort(arr, 0, 19, &comparacoes, &trocas);

    // Exibir o vetor ordenado
    printf("Vetor ordenado:\n");
    for (int i = 0; i < 20; i++) {
        printf("%s\n", arr[i]);
    }

    // Gerar um arquivo de saída
    FILE *arquivo_saida = fopen("saida.txt", "w");
    if (arquivo_saida == NULL) {
        printf("Erro ao abrir o arquivo de saída.\n");
        return 1;
    }

    for (int i = 0; i < 20; i++) {
        fprintf(arquivo_saida, "%s\n", arr[i]);
    }
    fclose(arquivo_saida);

    // Calcular a mediana
    int indice_mediana = 20 / 2;
    char *mediana = arr[indice_mediana];
```

```
    printf("Mediana: %s\n", mediana);

    printf("Número de comparações: %d\n", comparacoes);

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


    return 0;
}
```