Sumário

[1. Introdução: 2](#_Toc145018367)

[2. Implementação: 2](#_Toc145018368)

[3. Testes 2](#_Toc145018369)

[4. Conclusão 4](#_Toc145018370)

[Referências 4](#_Toc145018371)

[Anexos 4](#_Toc145018372)

[tp01.c 4](#_Toc145018373)

[DadosEntrada.csv 4](#_Toc145018374)

## Introdução:

O objetivo deste trabalho é ler os dados de um vetor de palavras, processá-los e organiza-los em ordem crescente. Será utilizado o algoritmo de ordenação QuickSort.

#### GitHub:

<https://github.com/arthursleite/Estrutura-Dados-UCB/tree/main/tp01>

## Implementação:

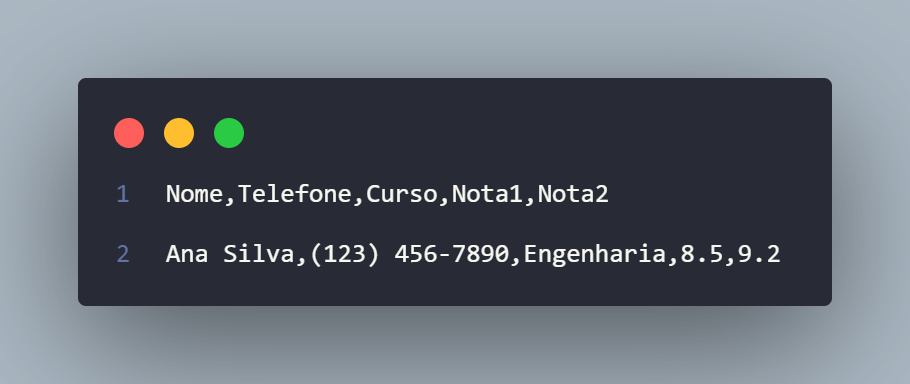
Foi utilizada a linguagem C com assistência da IDE Visual Studio Code e compilador GCC. Os arquivos .csv foram manipulados através de métodos fornecidos pela biblioteca stdio.h e a manipulação de algumas strings foi feita por métodos fornecidos pela biblioteca string.h.  
A implementação foi feita utilizando somente um arquivo .c contendo toda a lógica, funções e variáveis do programa.

## Testes

Na imagem abaixo, nas linhas 1 e 2, é possível ver quais são as bibliotecas que serão utilizadas. Nas linhas 4 e 5, a função main é chamada. Da 6 a 12, é possível ver a declaração das variáveis do tipo FILE, char e double. Nas linhas 14 e 15, os arquivos são abertos para leitura e escrita, respectivamente. Da linha 16 a 20, é checado se algum dos arquivos é nulo, caso sim, uma mensagem de erro será empresa e a função main() retornará 1.



A primeira linha do arquivo CSV (primeira imagem abaixo) contém o a descrição de cada dado e os dados começam a partir da segunda linha.



Para pular essa linha de descrições mencionada acima, foi preciso usar um fscanf() que leia os dados da primeira linha do arquivo DadosEntrada, não os armazene em nenhuma variável e pule para a próxima linha assim que encontrar um caractere \n.  
Na linha 2 é utilizado uma função while() que verificará se os 5 dados foram verificados e armazenado corretamente, caso resulte em falso, o loop será interrompido na linha 12.  
Dentro do while(), na linha 4 é declarada uma variável ]que armazena a média entre as duas notas do aluno. Nas linhas 6 a 10 o programa verifica a média do aluno, e caso seja menor que 7, ele será reprovado, caso não, será aprovado. Esse valor em formato de texto será armazenado na variável declarada na linha 5.  
Na linha 11 é feita a impressão do nome, média das notas e situação final (aprovado ou reprovado) do aluno no arquivo SituacaoFinal.



Na imagem abaixo, nas linhas 1 e 2, os arquivos são fechados. Na linha 4, o programa retornará caso o resto do programa rode com sucesso. E na linha 4 é fechada a função main().



## Conclusão

O programa funcionou perfeitamente de acordo com o que foi pedido no enunciado. O que poderia ser melhorado em próximas versões seria uma forma de formatar a leitura dos dados igualmente efetiva à atual porém com legibilidade e entendimento melhor para qualquer um que leia o código.

## Referências

<https://en.cppreference.com/w/c/io>

<https://stackoverflow.com/questions/41125193/using-fscanf-to-read-from-a-csv-file-in-c>

<https://stackoverflow.com/questions/30432856/best-way-to-get-number-of-lines-in-a-file-c>

<https://stackoverflow.com/questions/2799612/how-to-skip-a-line-when-fscanning-a-text-file>

## Anexos

### [tp01.c](https://github.com/arthursleite/Estrutura-Dados-UCB/blob/main/tp01/tp01.c)

### [DadosEntrada.csv](https://github.com/arthursleite/Estrutura-Dados-UCB/blob/main/tp01/DadosEntrada.csv)

[**SituacaoFinal.csv**](https://github.com/arthursleite/Estrutura-Dados-UCB/blob/main/tp01/SituacaoFinal.csv)