

Sumário

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

1. Introdução:

O presente projeto visa solucionar um problema em que muitos professores têm ao longo de suas carreiras profissionais: A demora para lançar o resultado final de cada aluno de acordo com suas notas. Dessa forma, visando facilitar a vida dos professores, foi feito um programa utilizando a linguagem de programação C. Esse aplicativo tem como objetivo ler um arquivo CSV lançado pelo usuário (normalmente um professor), em que conterá as informações necessárias para nosso programa funcionar: Nome, telefone, curso, notas 1 e 2. Após a entrada, será processado e o programa irá gerar um novo arquivo chamado “SituacaoFinal.csv”.

GitHub:

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

2. Implementação:

Em primeira instância foi criada a variável composta do tipo struct para o aluno. Após isso, foi criado duas variáveis do tipo FILE para ler e escrever os arquivos, uma do tipo char com capacidade de até 1024 caracteres (valor máximo para ser lido por linha) e uma variável do tipo aluno. Após isso, foi recebido os valores do tipo FILE por meio do método fopen(), e caso eles forem nulos, exibirá uma mensagem de erro identificando este problema. Após isso, caso o programa não seja encerrado, ele irá executar um laço de repetição que só irá terminar quando a última linha for lida. Desse modo, ele irá percorrer cada linha e fazer a soma da média, optei por fazer dentro do laço principal mesmo pois é algo simples de entender, uma verificação que retorna uma situação, que é definida por variável, em que é “APROVADO” ou “REPROVADO”. Após isso, o método fprintf() serve para gravar no novo arquivo gerado o resultado com os parâmetros do nome, a média e o resultado final. Por fim, só fechei os arquivos com o método fclose() e encerrei a execução do meu programa.

3. Testes

Para meus testes, eu optei por gerar vários arquivos: Um que iria conter somente dados de alunos que foram REPROVADOS, outro que iria conter dados somente de alunos APROVADOS, e por fim, mesclado. Confesso que o que mais retornou resultado foi o mesclado, pois pude ver que o aplicativo estava funcionando corretamente, dessa forma, segue em anexo o resultado dos meus testes ao ser executado pelo Code blocks, IDE de minha preferência:

```
C:\Users\MBrio\Desktop\Trabalho-de-Estrutura-de-dados\principal.exe
Ana Silva, 4.25, REPROVADO
Pedro Santos, 6.85, REPROVADO
Maria Oliveira, 6.90, REPROVADO
João Pereira, 7.00, APROVADO
Sofia Alves, 5.00, REPROVADO
Ricardo Ferreira, 7.55, APROVADO
Juliana Lima, 9.35, APROVADO
André Santos, 8.65, APROVADO
Laura Vieira, 9.40, APROVADO
Felipe Rodrigues, 7.85, APROVADO
Mariana Costa, 8.60, APROVADO
Guilherme Almeida, 7.40, APROVADO
Carolina Oliveira, 9.55, APROVADO
Eduardo Santos, 8.40, APROVADO
Beatriz Lima, 7.10, APROVADO
Gustavo Vieira, 8.05, APROVADO
Isabela Almeida, 8.65, APROVADO
Rafael Santos, 7.85, APROVADO
Fernanda Oliveira, 9.40, APROVADO
Lucas Pereira, 6.60, REPROVADO
Mariana Alves, 8.80, APROVADO
Gustavo Ferreira, 7.45, APROVADO
Camila Lima, 9.70, APROVADO
André Santos, 8.20, APROVADO
Larissa Vieira, 9.40, APROVADO
Pedro Rodrigues, 7.25, APROVADO
Sofia Costa, 8.80, APROVADO
Lucas Almeida, 7.55, APROVADO
Gabriela Santos, 9.55, APROVADO
Mateus Oliveira, 8.45, APROVADO
```

4. Conclusão

Com base no apresentado, é possível tirar por conclusão que está não é, de longe, a melhor forma de fazer a resolução do problema. Porém, é uma forma bem eficiente de se resolver a questão proposta, utilizando a lógica. Mas, esse é um algoritmo que por ser simples, permite implementações constantes, tais como separar as funções específicas da parte principal do algoritmo.

Referências

<https://chat.openai.com/>

<https://stackoverflow.com/questions/8115944/writing-user-input-to-a-file-in-c-programming>

Anexos

principal.c

```
Start here X principal.c X
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  typedef struct{
6      char Nome[100];
7      char Telefone[20];
8      char Curso[50];
9      float Nota1;
10     float Nota2;
11 } Aluno;
12
13 int main(){
14     FILE *arquivoEntrada;
15     FILE *arquivoSaida;
16
17     char linha[1024];
18     Aluno aluno;
19
20     arquivoEntrada = fopen("DadosEntrada.csv", "r");
21     arquivoSaida = fopen("SituacaoFinal.csv", "w");
22
23     if (arquivoEntrada == NULL || arquivoSaida == NULL) {
24         printf("Ocorreu um erro na tentativa de abrir os arquivos.");
25         exit(1);
26     }
27
28     fgets(linha, sizeof(linha), arquivoEntrada);
29
30     while (fgets(linha, sizeof(linha), arquivoEntrada) != NULL) {
31         char *token = strtok(linha, ",");
32
33         if (token == NULL) {
34             continue;
35         }
36     }
```

```
Start here x principal.c x
33     if (token == NULL) {
34         continue;
35     }
36
37     strcpy(aluno.Nome, token);
38     token = strtok(NULL, ",");
39     strcpy(aluno.Telefone, token);
40     token = strtok(NULL, ",");
41     strcpy(alunoCurso, token);
42     token = strtok(NULL, ",");
43     sscanf(token, "%f", &aluno.Nota1);
44     token = strtok(NULL, ",");
45     sscanf(token, "%f", &aluno.Nota2);
46
47     float media = (aluno.Nota1 + aluno.Nota2) / 2.0;
48
49     const char *situacao;
50     if (media >= 7.0) {
51         situacao = "APROVADO";
52     } else {
53         situacao = "REPROVADO";
54     }
55
56     printf("%s, %.2f, %s\n", aluno.Nome, media, situacao);
57     fprintf(arquivoSaida, "%s, %.2f, %s\n", aluno.Nome, media, situacao);
58 }
59
60 fclose(arquivoEntrada);
61 fclose(arquivoSaida);
62
63 printf("Arquivo SituacaoFinal.csv criado com sucesso!\n");
64 return 0;
65 }
66
```