**TRABALHO.C**

**PARTE PRINCIPAL COM O MAIN:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "funcoes.h"

// Define o tamanho máximo de uma palavra e o tamanho do alfabeto

#define TAMANHO\_MAXIMO\_PALAVRA 256

#define TAMANHO\_ALFABETO 26

int main()

{

    char entrada[TAMANHO\_MAXIMO\_PALAVRA];           // Array para armazenar a palavra de entrada do usuário

    char entradaSemEspacos[TAMANHO\_MAXIMO\_PALAVRA]; // Array para armazenar a palavra de entrada sem espa�os

    char palavra[TAMANHO\_MAXIMO\_PALAVRA];           // Array para armazenar cada palavra lida do arquivo

    FILE \*arquivo;                                  // Ponteiro para o arquivo de palavras

    // Loop principal para retornar ao inicio do programa apos cada palavra lida até a saida

    while (1)

    {

        // Informe a possibilidade de saida

        printf("\nEscreva '1' ou de enter sem qualquer entrada para sair\n");

        // Ler a palavra de entrada do usuario

        printf("Digite uma palavra: ");

        if (scanf("%255[^\n]%\*c", entrada) != 1)

        {

            break;

        }

        // Trata a entrada do usuario para remover espaços

        receberEntradaSemEspacos(entrada, entradaSemEspacos);

        // Verifica se o usuário deseja sair

        if (entradaSemEspacos[0] == '1' && entradaSemEspacos[1] == '\0')

        {

            break;

        }

        // Converte a palavra de entrada para minusculas

        converterParaMinusculas(entradaSemEspacos);

        // Abre o arquivo de palavras no modo leitura

        arquivo = fopen("palavras.txt", "r");

        if (!arquivo)

        {

            perror("Não foi possível abrir o arquivo palavras.txt"); // Exibe mensagem de erro se não conseguir abrir o arquivo

            return 1;                                                // Retorna 1 para indicar falha ao abrir o arquivo

        }

        // Imprime mensagem inicial

        printf("Palavras que podem ser formadas com as letras de '%s':\n", entrada);

        // Lê cada palavra do arquivo ate o final

        while (fscanf(arquivo, "%s", palavra) != EOF)

        {

            // Verifica se a palavra do arquivo pode ser formada com as letras da palavra de entrada

            if (podeFormarPalavra(entradaSemEspacos, palavra))

            {

                printf("%s\n", palavra); // Imprime a palavra se ela pode ser formada com as letras da palavra de entrada

            }

        }

        // Fecha o arquivo

        fclose(arquivo);

    }

    return 0; // Retorna 0 para indicar sucesso

}

**FUNCOES.H**

**ARQUIVO COM AS FUNÇÕES**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

// Define o tamanho maximo de uma palavra e o tamanho do alfabeto

#define TAMANHO\_MAXIMO\_PALAVRA 256

#define TAMANHO\_ALFABETO 26

// Funcao para tratar caso o usuario faca a adicao de palavras em Maiuscula

void converterParaMinusculas(char \*palavra)

{

    // Percorre a palavra e converte cada letra para minuscula

    while (\*palavra)

    {

        if (\*palavra >= 'A' && \*palavra <= 'Z')

        {

            \*palavra = \*palavra + 32;

        }

        palavra++;

    }

}

// Funcao para contar a frequência das letras em uma palavra

void contarLetras(const char \*palavra, int \*contagem)

{

    // Inicializa o array contagem com zeros

    int i = 0;

    for (i = 0; i < TAMANHO\_ALFABETO; i++)

    {

        contagem[i] = 0;

    }

    // Percorre a palavra e incrementa o contador da letra correspondente

    // 'a' - 'a' resulta em 0, 'b' - 'a' resulta em 1, ..., 'z' - 'a' resulta em 25

    while (\*palavra)

    {

        contagem[\*palavra - 'a']++;

        palavra++;

    }

}

// Função para verificar se uma palavra pode ser formada com as letras de outra palavra

int podeFormarPalavra(const char \*entrada, const char \*palavra)

{

    int contagemEntrada[TAMANHO\_ALFABETO] = {0}; // Array para contar as letras da palavra de entrada

    int contagemPalavra[TAMANHO\_ALFABETO] = {0}; // Array para contar as letras da palavra do arquivo

    // Conta as letras da palavra de entrada

    contarLetras(entrada, contagemEntrada);

    // Conta as letras da palavra do arquivo

    contarLetras(palavra, contagemPalavra);

    // Verifica se cada letra da palavra de entrada está presente na palavra do arquivo

    // em quantidade igual ou maior

    int i = 0;

    for (i = 0; i < TAMANHO\_ALFABETO; i++)

    {

        if (contagemEntrada[i] > 0 && contagemPalavra[i] < contagemEntrada[i])

        {

            return 0; // A palavra do arquivo contém menos ocorrências de uma letra do que a palavra de entrada

        }

        if (contagemEntrada[i] == 0 && contagemPalavra[i] > 0)

        {

            return 0; // A palavra do arquivo contém uma letra que não está na palavra de entrada

        }

    }

    return 1; // Retorna verdadeiro se a palavra do arquivo pode ser formada com as letras da palavra de entrada

}

// Função para tratar a entrada do usuário e remover espaços

void receberEntradaSemEspacos(char \*entrada, char \*resultado)

{

    // insere as variaveis

    int j = 0;

    int i;

    // Percorre a entrada e copia os caracteres diferentes de espaço e nova linha para o resultado

    for (i = 0; entrada[i] != '\0'; i++)

    {

        if (entrada[i] != ' ' && entrada[i] != '\n')

        {

            resultado[j] = entrada[i];

            j++;

        }

    }

    resultado[j] = '\0'; // Termina a string corretamente

}