

**LUIZ HENRIQUE SERAFIM DA SILVA**  
**RAPHAEL ESPOSTI**

**TRABALHO FINAL ESTRUTURA DE DADOS – EDITOR DE TEXTO**

**Presidente Prudente**

**2022**



**Resumo.....1.0**

**Como Funciona.....2.0**


**Explicação Código.....3.0**

**Limitações.....4.0**

## 1.0 Resumo

O objetivo do trabalho é a implementação de um editor de texto em linguagem C, o trabalho foi realizado utilizando listas dinâmicas como estrutura de dados.

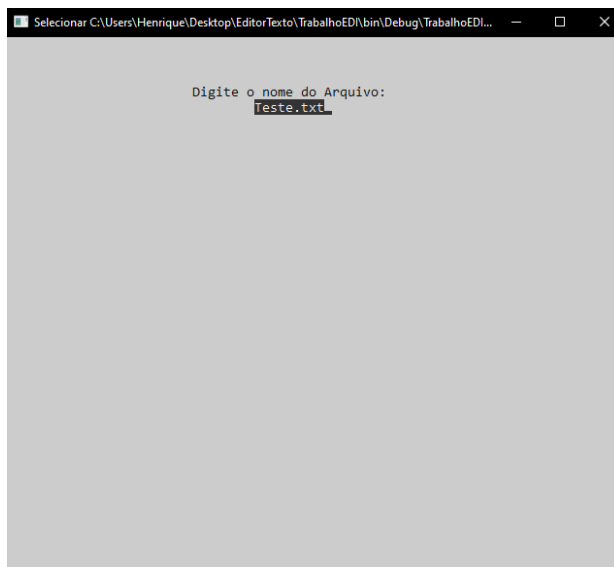
## 2.0 Como Funciona

Nome	Data de modificação	Tipo	Ta
 TrabalhoEDI.exe	13/07/2023 00:08	Aplicativo	

Abra o arquivo executável localizado na pasta “bin” da aplicação.



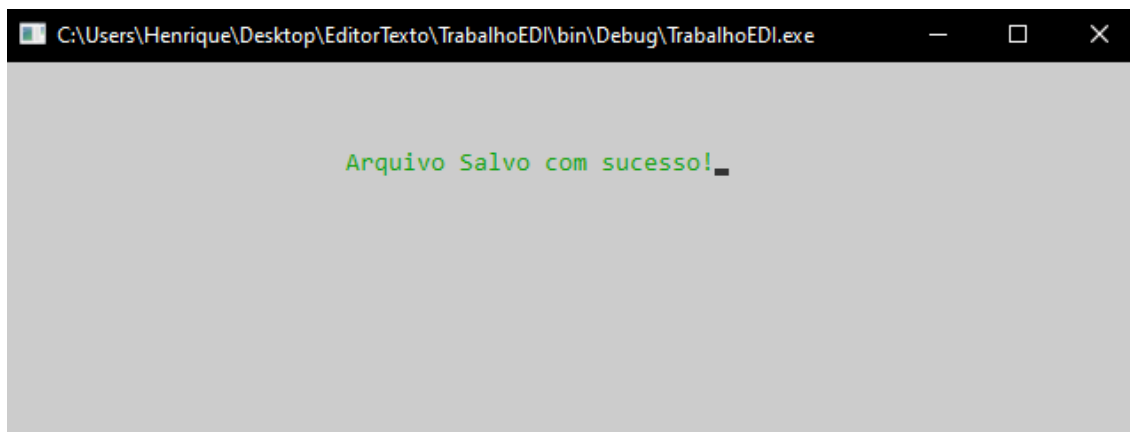
Após a execução o programa abrira na tela principal onde tem várias opções para serem escolhidas. No menu terá 4 opções de escolha a primeira para criar o arquivo, a segunda para editar um arquivo já criado, a terceira para deletar um arquivo e a quarta para sair da aplicação.



Ao selecionar a opção número 1 “criar”, o programa o redicionara para uma tela para inserir o nome do arquivo a ser criado.



Então o programa abrirá na tela do editor. Nessa tela você escreva o que deseja e após isso, para encerrar o programa digite um ponto final e pressione **[ENTER]**.



Então o arquivo será salvo e adicionado na pasta do executável, e a aplicação o redirecionará para o menu inicial.



após escolher a opção número 2 "Editar", a aplicação te redirecionará para uma tela de inserir o nome do arquivo para ser aberto, após escrever o nome do arquivo, você será redirecionado para uma tela de exibição onde mostrará o conteúdo do arquivo, e você poderá escolher 3 opções, Sair que te redirecionará para o menu principal, Deletar que deletará o arquivo e a opção editar.

```
C:\Users\Henrique\Desktop\EditorTexto\TrabalhoEDI\bin\Debug\TrabalhoEDI.exe
Trabalho teste
1
2
2
323
2
1651561456 15616311
15 165 165 165 163 131
Digite o texto e pressione Ctrl+S para acrescentar.
```

Quando digitar a opção editar você será redirecionado para uma tela onde aparecerá o conteúdo do arquivo e uma opção para adicionar mais coisas ao arquivo texto, após escrever e digitar **[Ctrl+s]**, o arquivo salvará e você será redirecionado para o menu principal.

### 3.0 Explicação Código

```
#include <string.h>

#define ANSI_RESET_ALL          "\x1b[0m"           //Definições para interface em linguagem C

#define ANSI_COLOR_BLACK        "\x1b[30m"
#define ANSI_COLOR_RED          "\x1b[31m"
#define ANSI_COLOR_GREEN        "\x1b[32m"
#define ANSI_COLOR_YELLOW       "\x1b[33m"
#define ANSI_COLOR_BLUE         "\x1b[34m"
#define ANSI_COLOR_MAGENTA      "\x1b[35m"
#define ANSI_COLOR_CYAN         "\x1b[36m"
#define ANSI_COLOR_WHITE        "\x1b[37m"

#define ANSI_BACKGROUND_BLACK    "\x1b[40m"
#define ANSI_BACKGROUND_RED      "\x1b[41m"
#define ANSI_BACKGROUND_GREEN    "\x1b[42m"
#define ANSI_BACKGROUND_YELLOW   "\x1b[43m"
#define ANSI_BACKGROUND_BLUE     "\x1b[44m"
#define ANSI_BACKGROUND_MAGENTA  "\x1b[45m"
#define ANSI_BACKGROUND_CYAN     "\x1b[46m"
#define ANSI_BACKGROUND_WHITE    "\x1b[47m"

#define ANSI_STYLE_BOLD          "\x1b[1m"
#define ANSI_STYLE_ITALIC        "\x1b[3m"
#define ANSI_STYLE_UNDERLINE     "\x1b[4m"
```

Essas definições foram usadas na interface para pintar o prompt e adicionar estilização na interface como “Bold”, “Italic” e “Underline”.

```

typedef struct word {           //palavras
    char* text;
    struct word* proximo;
} Word;

typedef struct Line {           //lista encadeada de palavras "LINHAS"
    Word* primeiraPalavra;
    Word* ultimaPalavra;
    struct Line* proximo;
} Line;

typedef struct {                //lista encadeada de linhas "TEXTO"
    Line* primeiraLinha;
    Line* ultimaLinha;
} LineList;

int i, j, ch;
char fn[20], e, c;
FILE *fp1, *fp2, *fp;

LineList lineList;

```

As definições das estruturas, A lógica empregada foi inserir palavras em uma linha e inserir linhas em uma lista, que é o texto. Tinha pretensão de adicionar letras e fazer das palavras uma lista de letras, porém devido ao pouco tempo que tive para fazer as alterações só consegui fazer essa parte.

```

void adicionarLinhaLista(char* LineText) {           //Funcao adicionar linha na lista
    Line* novaLinha = (Line*)malloc(sizeof(Line));
    novaLinha->primeiraPalavra = NULL;
    novaLinha->ultimaPalavra = NULL;
    novaLinha->proximo = NULL;                       //Inicializa e aloca a memoria

    char* token = strtok(LineText, " ");
    while (token != NULL) {                           //Inseri as palavras na linha e depois as l
        Word* novaPalavra = (Word*)malloc(sizeof(Word));
        novaPalavra->text = token;
        novaPalavra->proximo = NULL;

        if (novaLinha->primeiraPalavra == NULL) {
            novaLinha->primeiraPalavra = novaPalavra;
            novaLinha->ultimaPalavra = novaPalavra;
        } else {
            novaLinha->ultimaPalavra->proximo = novaPalavra;
            novaLinha->ultimaPalavra = novaPalavra;
        }

        token = strtok(NULL, " ");
    }

    if (lineList.primeiraLinha == NULL) {
        lineList.primeiraLinha = novaLinha;
        lineList.ultimaLinha = novaLinha;
    } else {
        lineList.ultimaLinha->proximo = novaLinha;
        lineList.ultimaLinha = novaLinha;
    }
}

```

Essa função divide a linha de texto em palavras, cria uma nova estrutura **Word** para cada palavra e, em seguida, insere essas palavras em uma nova linha, que é adicionada à lista encadeada **lineList**. A função recebe um parâmetro **lineText**, que contém a linha de texto a ser adicionada. Uma nova estrutura

**Line** é alocada dinamicamente na memória usando malloc(). Os ponteiros **primeiraPalavra**, **ultimaPalavra** e **proximo** são inicializados como NULL.

A função strtok() é usada para dividir a linha de texto em palavras individuais. O delimitador usado é o espaço em branco. Então dentro de um loop, as palavras individuais são extraídas usando strtok() até que não haja mais palavras na linha. Para cada palavra extraída, uma **Word** é alocada dinamicamente na memória usando malloc(). O ponteiro **text** é definido como a palavra extraída e o ponteiro **proximo** é inicializado como NULL.

Se for a primeira palavra da linha (**novaLinha->primeiraPalavra** é `NULL`), a primeira e a última palavra da linha são definidas como a nova palavra alocada. Caso contrário, a última palavra da linha atual tem seu ponteiro **proximo** apontando para a nova palavra, e a nova palavra se torna a última palavra da linha. Após processar todas as palavras da linha, verifica-se se **lineList** está vazia. Se for o caso, a nova linha se torna a primeira e a última linha em **lineList**. Caso contrário, a última linha de **lineList** tem seu ponteiro **proximo** apontando para a nova linha, e a nova linha se torna a última linha de **lineList**.

```
fp1 = fopen(arquivo, "w");
while (1)
{
    c = getchar();

    if (c == '.') {
        lineBuffer[lineBufferIndex] = '\0';
        adicionarLinhaLista(lineBuffer);
        lineBufferIndex = 0;

        system("cls");
        fclose(fp1);
        printf(ANSI_COLOR_GREEN"\n\n\n\t\t\tArquivo Salvo com sucesso!"ANSI_RESET_ALL);
        sleep(1);
        break;
    } else if (c == ' ') {
        lineBuffer[lineBufferIndex] = '\0';
        adicionarLinhaLista(lineBuffer);
        lineBufferIndex = 0;
        fputc(c, fp1);
    } else if (c == '\n') {
        isNewLine = 1;
    } else {
        lineBuffer[lineBufferIndex] = c;
        lineBufferIndex++;
        fputc(c, fp1);
    }

    if (isNewLine) {
        lineBuffer[lineBufferIndex] = '\0';
        adicionarLinhaLista(lineBuffer);
        lineBufferIndex = 0;
        fputc('\n', fp1);
        isNewLine = 0;
    }
}
```

O código da função **Criar()** é responsável por criar um novo arquivo de texto e permitir ao usuário digitar o texto desejado. Essa função cria um arquivo de texto e armazena o texto digitado pelo usuário no arquivo, ao mesmo tempo



em que mantém uma representação dinâmica do texto em uma struct de lista encadeada.

O usuário é solicitado a inserir o nome do arquivo através da função `scanf()` e o valor é armazenado na variável `arquivo`. A função **`adicionarLinhaLista()`** é chamada para adicionar dinamicamente cada linha do texto digitado pelo usuário em uma lista encadeada. Um loop infinito é iniciado para ler cada caractere digitado pelo usuário. O programa verifica se o caractere digitado é um ponto (`.`). Se for, o loop é interrompido, indicando que o usuário deseja salvar o arquivo.

Se o caractere digitado for um espaço em branco, a função **adicionarLinhaLista()** é chamada para adicionar a palavra atualmente armazenada no **lineBuffer** à lista de palavras da linha atual. Porém se o caractere digitado for uma nova linha ('\n'), a variável **isNewLine** é definida como 1 para indicar que uma nova linha começará em breve. Se o caractere não for um ponto, espaço em branco ou nova linha, ele é considerado parte da palavra atual e é armazenado no **lineBuffer**. Se a variável **isNewLine** for verdadeira, significa que uma nova linha será iniciada, então a palavra atual é adicionada à lista de palavras da linha atual e a variável **isNewLine** é redefinida para 0. O caractere digitado é escrito no arquivo usando a função **fputc()**.

```
void Mostrar();  
{  
    int escolha;  
  
    system("cls");  
    printf("\n\n\n\t\t\tDigite o nome do Arquivo: \n\t\t\t\t");  
    scanf("%s", fn);  
    system("cls");  
    printf(ANSI_BACKGROUND_BLUE ANSI_COLOR_BLUE"-----");  
    printf(ANSI_BACKGROUND_BLUE ANSI_COLOR_WHITE"\n %s:"ANSI_RESET_ALL ANSI_BACKGROUND_BLUE ANSI_COLOR_BLUE" ");  
    printf(ANSI_BACKGROUND_BLUE ANSI_BACKGROUND_BLACK ANSI_COLOR_WHITE"ANSI_RESET_ALL ANSI_BACKGROUND_BLUE ANSI_COLOR_WHITE"  
    printf(ANSI_BACKGROUND_BLUE ANSI_COLOR_BLUE"ANSI_RESET_ALL ANSI_BACKGROUND_BLACK ANSI_COLOR_WHITE"ANSI_RESET_ALL  
    printf(ANSI_BACKGROUND_BLUE ANSI_COLOR_BLUE"ANSI_RESET_ALL ANSI_BACKGROUND_BLACK ANSI_COLOR_WHITE"ANSI_RESET_ALL\nANSI_R  
  
    fp1 = fopen(fn, "r");  
    if (fp1 == NULL)  
    {  
        system("cls");  
        printf(ANSI_RESET_ALL ANSI_BACKGROUND_BLACK ANSI_COLOR_RED"\n\n\n\t\t\tArquivo nao encontrado!"ANSI_RESET_ALL ANSI_BACKGROUND_WHI  
        sleep(1);  
        Mostrar();  
        goto end1;  
    }  
    else{  
        while (!feof(fp1))  
        {  
            c = getc(fp1);  
            printf("%c", c);  
        }  
    }  
end1:  
    fclose(fp1);  
    printf(ANSI_BACKGROUND_BLUE ANSI_COLOR_WHITE"\n\n\n\n\t\t\tDIGITE A OPCAO DESEJADA:\n\n\t\t\t1-EDITAR\t2-DELETAR\t3-SAIR\n\n"ANSI_R  
    scanf("%d",&escolha);  
  
    switch(escolha){  
        case 1: Acrescentar();break;  
        case 2: Delete();break;  
        case 3: printf("\n\n\n\t\t\t pressione qualquer tecla para continuar\n"ANSI_RESET_ALL);getc();break;  
    }  
}
```

Essa função exibe o conteúdo do arquivo na tela, permite ao usuário realizar operações adicionais no arquivo e controla o fluxo do programa com base na escolha do usuário. O usuário é solicitado a inserir o nome do arquivo através da função `scanf()` e o valor é armazenado na variável **fn**. O programa tenta abrir o arquivo em modo de leitura usando `fopen()`. Se o arquivo não for

Já se o arquivo for aberto com sucesso, o conteúdo do arquivo é lido caractere por caractere usando `getc(fp1)` dentro de um loop que continua até o final do arquivo. Cada caractere lido é impresso na tela. É exibido um menu com opções para editar, deletar ou sair.

Essa função permite que o usuário exclua um arquivo de texto existente.

O programa tenta abrir o arquivo em modo de leitura usando `fopen()` para verificar se o arquivo existe. Se o arquivo não for encontrado, uma mensagem de erro é exibida, a função `Delete()` é chamada novamente recursivamente, e o controle é transferido para **end2** usando **goto**. O arquivo é fechado com `fclose(fp1)`.

Se a função `remove(arquivo)` retornar um valor negativo, isso indica que ocorreu um erro ao excluir o arquivo, e uma mensagem de erro é exibida.



```

void Editar()
{
    char arquivo[500];
    int lineIndex, lineCount;
    Line* currentLine;

    system("cls");
    printf("\n\tDigite o nome do arquivo:");
    scanf("%s", &arquivo);
    fp1 = fopen(arquivo, "r");
    if (fp1 == NULL)
    {
        system("cls");
        printf("\n\tArquivo não encontrado!");
        goto end;
    }

    lineCount = 0;
    currentLine = lineList.primeiraLinha;
    while (currentLine != NULL)
    {
        lineCount++;
        currentLine = currentLine->proximo;
    }

    fclose(fp1);
}

```

O código possui uma implementação de uma função Editar, que supostamente substituiria a função acrescentar porém não consegui acabar ela, ela possui alguns erros que não sei quais são e não está funcionando como devia por isso não deixei ela na interface.

#### **4.0 Limitações**

Bem, o programa em si funciona muito bem, consegui caprichar na interface e consegui implementar o armazenamento dos caracteres, com muita ajuda da internet pois são muitas estruturas a serem gerenciadas mas consegui, porém a aplicação em si não funciona como eu gostaria pois não consegui acabar a função Editar() que substituiria a função acrescentar que foi usada como um "Tapa buraco".