Documentação

Gustavo Pauzner Mezzovilla

2 de fevereiro de 2023

Is	Insere a string s na posição atual do texto
An	Carrega o conteúdo do arquivo de texto de nome n no editor
En	(Sobre)escreve o conteúdo do editor no arquivo de texto de nome n
F	Move o cursor um caractere à frente
T	Move o cursor um caractere para trás
0	Move o cursor para o início da linha atual
P	Move cursor para início da próxima palavra (dentro da mesma linha)
Q	Move cursor para início da palavra atual
$	Move o cursor para o fim da linha atual
:x	Move o cursor para o início da linha x
:F	Move o cursor para a última linha do arquivo
D	Apaga o caractere da posição atual
DW	Apaga a palavra em que o cursor se encontra
DL	Deleta a linha atual
V	Desempilha e insere o conteúdo do topo pilha na posição atual
C	Empilha o texto entre a posição marcada e a posição atual (sem modificá-lo)
X	Empilha o texto entre a posição marcada e a posição atual e o deleta
Bs	Busca pela próxima ocorrência do padrão s no texto
Ss/r	Substitui toda ocorrência de s por r no texto a partir da posição atual
N	Separa linha atual na posição do cursor
U	Unir linha atual e a próxima
!	Encerra o programa
J	Ir para próxima linha (manter a mesma coluna, se possível)
H	Ir para a linha anterior (manter a mesma coluna, se possível)

1 Estruturas

O presente editor de texto considera as seguintes estruturas de dados:

- `console`: estrutura responsável por capturar a entrada do usuário, que consistirá de funções e strings.

```c
typedef struct CONSOLE
{
    char *letras;
    int tamanho;
} console;
```

- `celula`: estrutura responsável por armazenar um caractere e ponteiro para as células adjacentes.

```c
typedef struct CELULA
{
        char val;
        struct CELULA *next;
        struct CELULA *prev;
} celula;
```

- `linha`: estrutura responsável por armazenar uma linha de texto, que consiste de uma lista duplamente encadeada de células, além de contemplar informações do início e do final da linha, bem como o tamanho da mesma.

```c
typedef struct LINHA
{
        int tamanho;
        struct CELULA *head;
        struct CELULA *tail;
        struct LINHA *up;
        struct LINHA *down;
} linha;
```

- `ponto`: estrutura da variável global cursor, que consiste de um ponteiro para a célula atual, além de os inteiros especificando sua linha e sua coluna.

```c
typedef struct PONTO
{
        int linha;
        int coluna;
        struct CELULA *cel;
} ponto;
```

As funções implementadas se dividem nas seguintes categorias: As que simplesmente realizam uma ação (como mover o cursor, apagar uma linha, etc.) sem nenhum parâmetro a mais, e as que dependem de um *prompt* de texto como entrada. Portanto, é possível compor as funções que não dependem de entradas de texto, e.g., mover o cursor 3 unidades pra trás, deletar um caracter e inserir outro, como ilustrado abaixo (**??**).

Listing 1: Exemplo de composição de funções

```
>0|vasa
      ^
------------------------
(0,4): T3DIc
------------------------
>0|casa
   ^
```

A utilização de listas duplamente encadeadas foi considerada para a implementação do editor de texto, pois a inserção e remoção de elementos em uma lista duplamente encadeada é feita em tempo constante, enquanto que em uma lista simplesmente encadeada, a inserção e remoção de elementos é feita em tempo linear.

Utilizar estruturas diferentes para as células e para listas também foi importante visando a navegabilidade do texto, em conjunto com a performance. O único problema dessa abordagem é no quesito complexidade de código. Dessa forma, a implementação da ferramenta de busca foi mais simplória, donde optou-se por checar desde o início do documento até o final linearmente. Para auxiliar essa abordagem, foi considerado o algorítimo de Knuth-Morris e Pratt, que consiste em um algorítimo de busca de padrões em strings, que possui complexidade de tempo linear no número de caracteres no pior caso.

Esse algorítmo foi desenvolvido para rodar em qualquer sistema operacional, contudo, para utilizar acentos gráficos e caracteres especiais, é necessário que o sistema operacional seja o Windows, de modo que algumas bibliotecas específicas foram consideradas. Contudo, a adaptação do mesmo não deve ser de difícil implementação para outros sistemas operacionais.

Alguns problemas surgiram durante a implementação desse projeto, como a tentativa de utilizar arquivos de cabeçalho para a implementação das funções, o que não foi possível devido a limitações do compilador utilizado. Como a maioria das funções tratavam de variáveis globais, priorizando simplicidade, optou-se por implementar todas as funções em um único arquivo, o que não é uma boa prática de programação, mas foi a melhor solução para o problema.