MAC0122
Princípios de Desenvolvimento de Algoritmos
Projeto Final - Seu primeiro editor de texto

2º. semestre de 2022

Considerações iniciais

Para o seu projeto final da disciplina, você deve utilizar todo o conhecimento aprendido para implementar, de maneira eficiente, correta e elegante, um editor de texto baseado em comandos. Além de colocar em prática os conhecimentos aprendidos sobre estruturas de dados e algoritmos de busca em texto e ordenação, nesse projeto você vai praticar as habilidades de documentação, escrita e análise de um software complexo. Para isso, você deve buscar **planejar o desenvolvimento do software** antes de começar sua implementação em código, incluindo formas de avaliar a correção e a eficiência computacional. Espera-se que suas escolhas sejam embasadas por critérios teóricos ou empíricos, por exemplo, comparando tempos de execução de diferentes implementações de soluções para uma mesma tarefa, em um conjunto bem justificado de instâncias. **O processo de desenvolvimento e as escolhas de projeto devem ser relatadas em uma monografia que será avaliada como parte da nota do projeto, assim como o código e o programa em si.**

Editores de texto baseados em comandos

Os editores de textos modernos como o bloco de notas do Windows possuem uma interface sofisticada de edição, com funcionalidades de implementação não trivial como elementos gráficos e comandos via mouse, além de serem integrados aos recursos do sistema operacional (menu e janelas para abertura de arquivos, funções de copiar e colar conteúdo entre aplicativos etc). Tais editores em geral seguem uma arquitetura chamada WSIWYG, do inglês

What-You-See-Is-What-You-Get (o que você vê é o que você recebe). Isso significa que o conteúdo de um arquivo de texto sendo editado é exibido da maneira mais fidedigna possível (incluindo anotações de formatação, formato de página etc) e qualquer edição é imediatamente refletida na visualização. Os primeiros editores de texto não eram nada parecidos a isso, em grande parte devido a escassez de recursos computacionais – não era comum haver interfaces gráficas e os computadores em geral possuíam memória escassa e eram acessados remotamente com latência significativa entre um comando e a resposta do resultado. Por isso, era comum que editores de texto operassem por comandos cujo resultado era exibido apenas parcialmente. Um exemplo prominentes da época é o programa `ed`, precursor de editores de textos mais modernos e poderosos como o `vim`.[1] Em tal editor, precisamos explicitamente indicar, via um comando, quais linhas queremos ver do texto sendo editado, e modificações no texto não são imediatamente mostradas.

Seu editor baseado em comandos

Sua tarefa nesse projeto é desenhar e implementar um editor de texto baseado em comandos de texto. Para isso, você precisará especificar como o texto sendo editado será armazenado e manipulado pelo programa (estruturas de dados, tipos de dados abstratos etc), além de implementar a interatividade do programa. Seu programa deve a cada iteração receber do usuário uma *string* contendo comandos de edição (inserção, deleção ou substituição de caracteres) ou navegação (movimentação do cursor, busca por padrão, etc) e então executá-los, exibindo apenas a linha atual do cursor. Por exemplo, para se inserir o texto `simples ` na posição atual do cursor, você deve digitar o comando de inserir, indicado pela letra `I`, seguido pelo texto a ser inserido e a tecla ENTER, que determina o fim dessa interação. O texto então é inserido na posição do cursor, que é deslocado para o final do texto inserido e o resultado linha atual é exibido logo abaixo. Tal exemplo de interação é ilustrado abaixo.

```
Um exemplo de texto.
          ^
2,11> Isimples <ENTER>
Um exemplo simples de texto.
                 ^
```

[1] Você pode saber mais sobre o ed em `https://en.wikipedia.org/wiki/Ed_(text_editor)`.

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
M	Marca (lembra) a posição atual do cursor
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)
Z	Exibe a pilha de memória, começando pelo topo.

Tabela 1: Resumo dos comandos do editor de texto

A primeira linha exibe o conteúdo da linha atual do texto sendo editado e a posição do cursor (`^`). A linha seguinte começa pela indicação da linha e coluna (`2,11`)), do símbolo `>`, que indica que um comando é esperado, e do comando digitado pelo usuário. O resultado da inserção é mostrado logo abaixo, com o cursor tendo sido deslocado para o fim da inserção.

Além do comando de inserção, você deve implementar comandos para movimentação do cursor, interação com arquivos de texto, cópia e deleção de texto, entre outros. A Tabela 1 descreve o conjunto mínimo de comandos e funcionalidades que seu editor de texto deve implementar.

O texto sendo editado deve ser organizado em linhas e colunas. O cursor guarda a posição de uma linha e da coluna dentro da linha. Note que as linhas possuem em geral larguras (número de colunas) distintas. Ao iniciar o editor ou abrir um arquivo de texto, a posição 0 é marcada, o cursor é definido

como o início do arquivo (linha 0, coluna 0) e a pilha é iniciada vazia. A operação `:F` move o cursor para a última linha do arquivo. Depois de cada operação o programa deve exibir o conteúdo da linha do cursor seguida pela indicação da coluna do cursor pelo caractere `^` e da marcação (`M`), se na linha atual e diferente do cursor, como nos exemplos abaixo. Por padrão, `M` é inicializado na posição (0,0). Em seguida deve exibir a posição do cursor `linha,coluna>` e aguardar por um nova sequência de comandos. Vamos por simplicidade assumir que cada sequência de comandos termina com a tecla ENTER, que não faz parte da sequência. Note que alguns comandos, tal como B e I, requerem uma string, para simplificar, vamos assumir que não é possível colocar outros comandos após eles na mesma entrada.

Vamos supor que estejamos em uma linha 0 contendo apenas a palavra "aro" e com o cursor na posição `(0,1)`. Isso significa que o cursor está entre as letras `a` e `r`. Assim, o cursor pode ser exibido até uma posição depois da última letra da frase, indicando estar na última posição. Em um texto vazio, o cursor estará na posição `(0,0)`.

Para os comandos `J` e `H`, se o número de colunas da linha destino for menor que a coluna da atual, então o cursor deve ir para o final da linha destino. Como dica geral, atente-se para casos limites tais como, mas não limitados a: ir para linhas inexistentes, mover o cursor para colunas negativas, apagar um texto vazio. Ao executar o comando `B` e encontrar a string s, o cursor deve ficar no fim da palavra s. Ao apagar um caractere, o cursor deve voltar uma posição — caso seja possível.

Outro ponto importante é sobre a pilha de memória. O seu programa deve ser capaz de empilhar textos em uma memória e gerencia-la. Ou seja, seria possível você armazenar (copiar para memória) um fragmento f_1 de texto, depois armazenar um fragmento f_2, ir para outro local do texto e colocar ali os conteúdos de f_2 e f_1, necessariamente nessa ordem. Deve ser possível empilhar e desempilhar mais de uma linha. Por exemplo, vamos supor que M esteja na posição (0,0), o cursor em (2,0), e o usuário dá o comando C: será empilhado todo o conteúdo da linha 0 e da linha 1, quando este conteúdo for desempilhado, as duas linhas serão desempilhadas de uma vez — igual em um editor de texto de hoje. Vamos assumir que `M` está sempre numa posição anterior ao `cursor`.

Também semelhante aos editores de hoje, ao executar o comando A, tudo que está no editor deve ser descartado e substituído pelo conteúdo do arquivo.

Exemplos de execução:

```
0,0> Aexemplo.txt<ENTER>
Um exemplo de texto.
^
```

```
0,0> FFF<ENTER>
Um exemplo de texto.
M  ^
0,3> MP<ENTER>
Um exemplo de texto.
   M       ^
0,10> Isimples <ENTER>
Um exemplo simples de texto.
   M               ^
0,19> DD<ENTER>
Um exemplo simples texto.
   M               ^
0,19> TTQT<ENTER>
Um exemplo simples texto.
   M      ^
0,10> X<ENTER>
Um simples texto.
   ^
0,2> Eexemplo2.txt<ENTER>
```

Considere, como outro exemplo, um arquivo de texto de nome `exemplo3.txt`, no qual cada linha i possui o texto *linha* i.

```
0,0> Aexemplo3.txt<ENTER>
linha 0
^
0,0> $C <ENTER>
linha 0
M      ^
0,7> Slinha/verso<ENTER>
linha 0
       ^
0,7> Bverso 10<ENTER>
verso 10
        ^
10,8> OV<ENTER>
linha 0verso 10
       ^
10,7> :0<ENTER>
linha 0
^
```

```
0,0> U<ENTER>
linha 0verso 1
^
0,0> :9<ENTER>
linha 0verso 10
^
9,0> PFN<ENTER>
linha 0
       ^
9,7> Eexemplo2.txt<ENTER>
linha 0
       ^
9,7> !<ENTER>
```

Relatório

Você deve produzir um relatório descrevendo e justificando todas as decisões de projeto tomadas (estruturas de dados e algoritmos conhecidos usados etc). Você deve também relatar a interface de programação desenvolvida (protótipos de funções e descrições de pré-condições esperadas e argumentos de chamada da função) – suponha que outra pessoa irá usar seu código e precisa entendê-lo. **Não replique o código-fonte do seu programa no relatório**, você deverá entregar todo o código-fonte conjuntamente ao PDF do relatório (no entanto, você pode usar trechos de códigos nas explicações, documentação e justificativas dadas). Sua documentação da interface deve incluir uma análise teórica e/ou empírica da complexidade de tempo e memória de cada função, assim como a descrição de possíveis casos patológicos (nos quais a função possui complexidade acima do esperado ou não funciona adequadamente) e alguma discussão sobre o uso esperado da função em face da aplicação (um editor de texto interativo).

Entrega

Você deve submeter no local indicado da página da disciplina na plataforma e-disciplinas um arquivo comprimido contendo todo o código-fonte de sua implementação e um arquivo em formato pdf contendo o relatório até **23:59 de 15/12/2022**.

Avaliação

A avaliação se dará em parte pela completude e clareza do relatório e em parte pela elegância da implementação. Seu relatório deve descrever detalhadamente e justificar as estruturas de dados utilizadas para representar os principais elementos do editor (texto, pilha etc), analisar e relatar a complexidade de cada comando, idealmente exibindo resultados de testes empíricos que corroborem o resultado teórico ou comparem diferentes implementações e abordagens. Ele também pode conter impressões pessoais sobre a dificuldade da tarefa e questões sobre o código escrito. O código-fonte deve ser bem documentado, de forma que seja utilizável mesmo na ausência da documentação do relatório. É recomendável que junto além do programa principal, haja programas que sirvam para testar a correção e/ou eficiência das funções implementadas.