

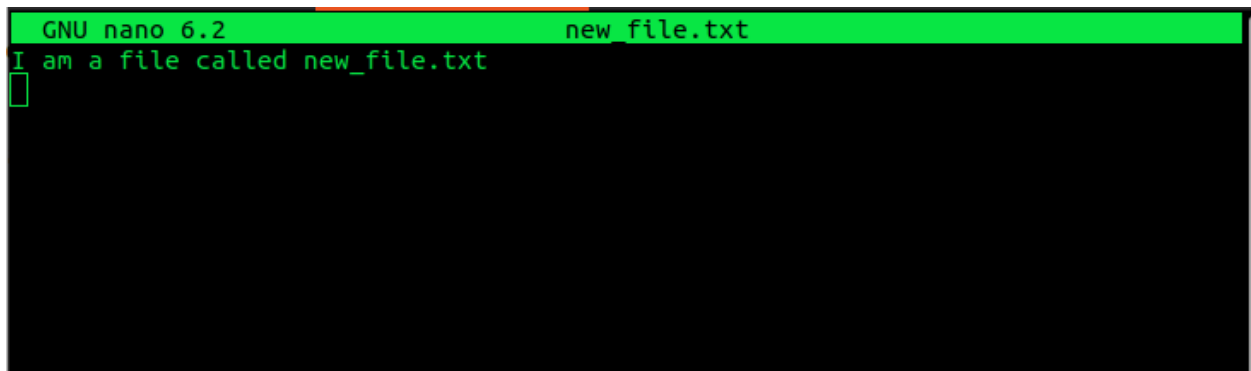
MC202AB - Estrutura de Dados

Lab 7 - Pilha e Fila

Data da Primeira Chance: 17 de outubro de 2022

Peso: 3

Neste lab, sua tarefa será implementar o seu próprio editor de texto. Para isso, você vai precisar utilizar os conceitos de pilha e fila explicados em aula.

A screenshot of the GNU nano 6.2 text editor. The title bar at the top is green and displays 'GNU nano 6.2' on the left and 'new_file.txt' on the right. The main editing area has a black background with green text. The first line of text reads 'I am a file called new_file.txt'. A green cursor is positioned at the beginning of the second line, which is currently empty.

Este editor a ser implementado contém uma única *string*, inicialmente vazia, que poderia ser um único vetor de *char*. **Porém**, essa *string* poderá ser extremamente grande e alocar tudo isso de forma sequencial pode ser problemático. Além disso, nosso editor apenas permite alterações no final do texto. Portanto, a forma de representar essa *string* será uma pilha **P** na forma de lista ligada, em que cada nó tem um vetor de *char* com 50 posições e um ponteiro para outro nó.

Essa pilha vai ser preenchida com *strings* de diferentes tamanhos e, por consequência, a cada vez que a quantidade máxima de caracteres do último nó for atingida, outro nó deve ser criado para ficar no topo da pilha **P**.

Um ponto importante é que esse editor também vai simular o funcionamento do *buffer* de entrada do terminal, em que as *strings* a serem adicionadas na pilha **P** serão primeiramente adicionadas em uma fila **F** (**implementada em um vetor circular de tamanho máximo igual a 500**). Somente quando o comando “EXEC [N]” for detectado em uma linha na entrada do programa é que os primeiros *N* caracteres da fila **F** serão desenfileirados e inseridos na pilha **P** (com *N* sendo no máximo o tamanho da fila **F**). Isso ilustra aquele momento em que o computador tem um pequeno travamento quando você está digitando e de repente tudo que você digitou aparece de uma única vez.

Outro ponto importante é que existe também o comando “DEL [M]” que faz com que os últimos **M** ($1 \leq M \leq 100$) caracteres na pilha sejam desempilhados, isto é, removidos da *string* única na pilha **P**.

Instruções para leitura da entrada:

- Use o *fgets()* para leitura das linhas de entrada do programa;
- Você precisará remover o “\n” de cada linha a ser lida na entrada do programa, e após isso inserir os caracteres na fila;
- Os caracteres de espaço “ ” devem ser mantidos, mesmo que no final da linha lida;
- A condição de parada do programa será o comando “EXIT” que estará sempre na última linha.

Dicas:

- Use a função *strcspn()* da biblioteca *string.h* para detectar o “\n” em uma linha lida;
- Outras funções da biblioteca *string.h* também podem ser úteis.

Entrada

A entrada do programa consiste em várias linhas envolvendo *strings* a serem adicionadas e comandos do tipo EXEC ou DEL. Cada string de entradas será dada em uma única linha, isto é, sem “\n” do meio dela.

Saída

A saída do programa deverá ser o estado atual da fila F e da pilha P a cada linha lida Da entrada do programa, seguindo o formato:

```
FILA ATUAL: [strings concatenadas]
PILHA ATUAL: [string na pilha]
#####
```

Exemplos

Exemplo 1:

Entrada

```
Minha
terra
EXEC 6
tem
palmeiras
EXEC 10
onde
canta
o
EXEC 14
sabiaaaa
EXEC 17
DEL 3
```

Saída

```
FILA ATUAL: Minha
PILHA ATUAL:
####
FILA ATUAL: Minha terra
PILHA ATUAL:
####
FILA ATUAL: terra
PILHA ATUAL: Minha
####
FILA ATUAL: terra tem
PILHA ATUAL: Minha
####
FILA ATUAL: terra tem palmeiras
PILHA ATUAL: Minha
####
FILA ATUAL: palmeiras
PILHA ATUAL: Minha terra tem
####
FILA ATUAL: palmeiras onde
PILHA ATUAL: Minha terra tem
####
FILA ATUAL: palmeiras onde canta
PILHA ATUAL: Minha terra tem
####
FILA ATUAL: palmeiras onde canta o
PILHA ATUAL: Minha terra tem
####
FILA ATUAL:  canta o
PILHA ATUAL: Minha terra tem palmeiras onde
####
FILA ATUAL:  canta o sabiaaaa
PILHA ATUAL: Minha terra tem palmeiras onde
####
FILA ATUAL:
PILHA ATUAL: Minha terra tem palmeiras onde canta o sabiaaaa
####
FILA ATUAL:
PILHA ATUAL: Minha terra tem palmeiras onde canta o sabia
####
```

Exemplo 2:

Entrada

```
Lembrete:::
EXEC 13
  escrever
relatorio
DEL 4
deeeePPP
  fisica
```

```
referente
a
EXEC 27
aula
DEL 5
12.
EXEC 20
EXEC 8
```

Saída

```
FILA ATUAL: Lembrete:::
PILHA ATUAL:
####
FILA ATUAL:
PILHA ATUAL: Lembrete:::
####
FILA ATUAL:  escrever
PILHA ATUAL: Lembrete:::
####
FILA ATUAL:  escrever relatorio
PILHA ATUAL: Lembrete:::
####
FILA ATUAL:  escrever relatorio
PILHA ATUAL: Lembrete:
####
FILA ATUAL:  escrever relatorio deeeeEPP
PILHA ATUAL: Lembrete:
####
FILA ATUAL:  escrever relatorio deeeeEPP fisica
PILHA ATUAL: Lembrete:
####
FILA ATUAL:  escrever relatorio deeeeEPP fisica referente
PILHA ATUAL: Lembrete:
####
FILA ATUAL:  escrever relatorio deeeeEPP fisica referente a
PILHA ATUAL: Lembrete:
####
FILA ATUAL:  fisica referente a
PILHA ATUAL: Lembrete: escrever relatorio deeeeEPP
####
FILA ATUAL:  fisica referente a aula
PILHA ATUAL: Lembrete: escrever relatorio deeeeEPP
####
FILA ATUAL:  fisica referente a aula
PILHA ATUAL: Lembrete: escrever relatorio de
####
FILA ATUAL:  fisica referente a aula 12.
PILHA ATUAL: Lembrete: escrever relatorio de
####
FILA ATUAL: aula 12.
PILHA ATUAL: Lembrete: escrever relatorio de fisica referente a
####
FILA ATUAL:
PILHA ATUAL: Lembrete: escrever relatorio de fisica referente a aula 12.
```

```
####
```

Regras e Avaliação

Seu código será avaliado não apenas pelos testes do CodePost, mas também pela qualidade. Dentre os critérios subjetivos de qualidade de código iremos analisar: o uso apropriado de funções, de comentários, e de structs; a escolha de bons nomes de funções, variáveis e de structs e seus campos; o correto uso de Tipos Abstratos de Dados e a correta separação em vários arquivos; a ausência de diversos trechos de código repetidos, e o tempo de execução e uso de memória dos algoritmos projetados. Note, porém, que essa não é uma lista exaustiva, pois outros critérios podem ser analisados dependendo do código apresentado visando mostrar ao aluno como o código poderia ser melhor.

Submissão

Você deverá submeter **cinco arquivos** na tarefa *Lab 07 - Pilha e Fila* no CodePost: ***fila.h***, ***fila.c***, ***pilha.h***, ***pilha.c*** e ***lab07.c***. No arquivo ***fila.h*** é esperado que você declare a *struct* com os devidos campos e funções necessárias para implementar os comportamentos da fila. Da mesma forma, no arquivo ***pilha.h*** é esperado que você declare a *struct* e as funções da pilha. No arquivo ***fila.c*** e ***pilha.c*** é esperado que você implemente as funções declaradas no ***fila.h*** e ***pilha.h***, respectivamente. Por fim, no arquivo ***lab07.c*** é esperado que o faça a leitura das entradas do programa e execute as chamadas de funções por meio dos dois TADs implementados.

Após o prazo estabelecido para a atividade, será aberta uma tarefa Lab 07 - Segunda Chance, com prazo de entrega até o fim do semestre.