MC202 — Estruturas de Dados

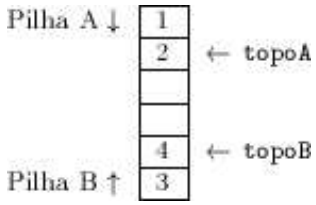
Professor

Ricardo da Silva Torres

<rtorres@ic.unicamp.br>

Segunda Lista de Exercícios

1. Duas pilhas A e B podem compartilhar o mesmo vetor, como esquematizado na figura abaixo.



   Faça as declarações de constantes e tipos necessárias e escreva as seguintes rotinas:

   (a) cria_pilhas, que inicia os valores de topoA e topoB;

   (b) vazia_A e vazia_B;

   (c) empilha_A e empilha_B;

   (d) desempilha_A e desempilha_B.

   Obs.: Só deve ser emitida uma mensagem de pilha cheia se todas as posições do vetor estiverem ocupadas.

2. Escreva uma função que verifique se uma string de entrada é da forma

   $str_1 \mathrm{C} str_2$,

   tal que str_1 é uma string composta apenas por caracteres A e B e str_2 é a string reversa de str_1. Por exemplo, a cadeia ABABBACABBABA é do formato especificado. A string de entrada deve ser percorrida uma única vez da esquerda para a direita!

   ```
   int st1Cstr2(char* str);
   ```

3. Escreva uma função que verifique se uma string de entrada formada apenas por '(' e ')' está balanceada. É necessária a utilização de uma pilha para resolver este problema?

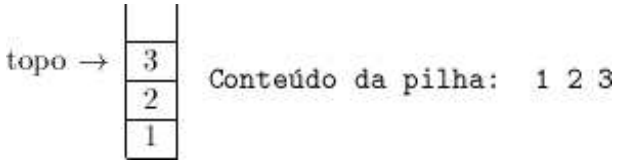
   ```
   int balanc_parenteses(char* str);
   ```

   Para os exercícios 4 e 5, você só pode utilizar uma pilha ou uma fila através das funções definidas na interface destes tipos abstratos (por exemplo, as funções cria_pilha, pilha_vazia, empilha, desempilha, destroi_pilha no caso de uma pilha; ou as funções cria_fila, fila_vazia, enfileira, desenfileira, destroi_fila no caso de uma fila). Não faça nenhuma suposição a respeito das implementações ou acesso direto a um campo de estrutura.

4. Escreva uma função que inverte os elementos de uma fila usando uma pilha.

```
void inverte_fila(Fila *fila);
```

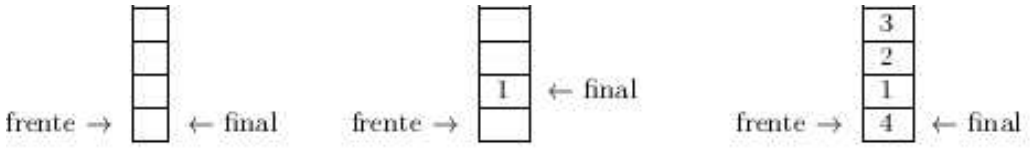
5. Escreva uma função que imprime os elementos de uma pilha na mesma ordem em que eles foram empilhados, como no exemplo abaixo:



O conteúdo da pilha deve ser o mesmo antes e depois da execução da função. Pode ser utilizada uma pilha auxiliar.

```
void imprime(Pilha* pilha);
```

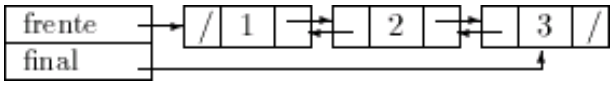
6. Por que na implementação de filas em vetor este é, em geral, circular? Considere que estamos utilizando apontadores (índices) para a frente e o final da fila: frente aponta para a posição imediatamente anterior ao primeiro elemento da fila e final aponta para o último elemento inserido, se existir. Comente a dificuldade para se diferenciar fila cheia de fila vazia.



(a) Fila vazia (b) Fila com um elemento (c) Fila com quatro elementos

Implemente as funções cria_fila, fila_vazia, insere_fila e remove_fila utilizando um *flag* (variável booleana) para indicar se a fila está cheia ou vazia.

7. Uma fila simétrica permite a realização das operações de inserção e remoção em ambas as extremidades. Considere uma implementação em que o tipo Fila_Simetrica é uma estrutura que contém apontadores para o primeiro e o último elemento da fila, sendo esta implementada através de uma lista duplamente ligada. Implemente as funções cria_fila, enfileira, desenfileira e destroi_fila.




```
typedef struct no* ap_no;

struct no {
  int v;
  ap_no ant, prox;
};

struct fila_simetrica {
  ap_no frente;
  ap_no final;
};

typedef struct fila_simetrica Fila_Simetrica;
```