Lista de Exercícios 06

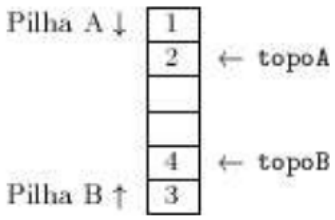
QXD0010 – Estruturas de Dados – Turma 03A – 2020.1

Prof. Atílio Gomes

8 de Setembro de 2020

1. Suponha que um dado problema requer o uso de duas pilhas A e B, onde cada pilha suporta no máximo 50 elementos e em nenhum momento as duas pilhas terão juntas mais do que 80 elementos. Assim, é possível implementar as duas pilhas em um único vetor usando apenas 80 posições ao invés de 100. Implemente a estrutura de dados e as operações de criar pilha vazia, empilhar, desempilhar, e consultar topo para estas duas pilhas.

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



2. Escreva um programa que implemente uma fila utilizando uma lista sequencial (vetor). Para evitar problemas de não ser capaz de inserir mais elementos na fila, mesmo quando ela não está cheia, implemente a lista sequencial de forma circular, como visto em sala de aula. O programa deve ser capaz de inserir e remover elementos na fila e informar o tamanho da fila em um dado momento.

3. Dada uma fila de inteiros, escreva um programa que exclua todos os números negativos sem alterar a posição dos outros elementos da fila.

4. Considere uma pilha p não vazia e uma fila f vazia. Utilizando apenas os testes de fila e pilha vazias, as operações `enqueue` (enfileirar), `dequeue` (desenfileirar), `pop` (desempilhar), `push` (empilhar), e uma variável `aux` do tipo da pilha, escreva uma função que inverta a ordem dos elementos da pilha.

5. Utilizando um dos TADs vistos em sala de aula (Lista, Pilha ou Fila) para auxiliá-lo na manipulação dos dados, implemente uma função que compute a fatoração prima de um número imprimindo os seus fatores em ordem crescente. Por exemplo, para o número 630 deverá ser impresso $2 \times 3 \times 3 \times 5 \times 7$. Justifique a escolha do TAD utilizado.

6. Considere o problema de verificar se uma sequência de parênteses e colchetes está balanceada. Por exemplo: “()”, “[([])]”, “((([([])])))” e “[]” são exemplos de sequências balanceadas enquanto “)(”, “(]”, “([)]” e “[[()])” não são. Escreva uma função que recebe uma sequência de parênteses e colchetes dada por uma string e que devolve 1 caso a sequência esteja balanceada ou 0 caso contrário.

   Obs.: Caso você queira usar um dos tipos abstratos de dados estudados em sala, você pode definir qual o tipo está usando e quais funções estão no seu “tipo.h” e se as funções já foram implementadas em alguma aula, elas podem ser usadas na construção da sua solução sem a necessidade de reimplementá-las.

7. Escreva uma função iterativa que simule o comportamento da seguinte função recursiva. Use uma pilha.

```
1  int ttt (int *x, int n){
2    if(n==-1) return 0;
3    if(x[n] > 0) return x[n] + ttt(x,n-1);
4    else return ttt(x,n-1);
5  }
```

8. Faça uma função que receba como entrada duas pilhas p_1 e p_2 e retorna 1 se as pilhas forem iguais e 0 caso contrario.

9. Considere uma pilha p vazia e uma fila f não vazia. Utilizando apenas os testes de fila e pilha vazias, as operações `enqueue`, `dequeue`, `pop`, `push`, e uma variável `aux` do tipo da fila, escreva uma função que inverta a ordem dos elementos da fila.

10. Inversão de palavras. Escreva uma função em C++ que inverta a ordem das letras de cada palavra de uma sentença, preservando a ordem das palavras. Suponha que as palavras da sentença são separadas por espaços. Por exemplo, a aplicação da sua operação à sentença AMU MEGASNEM ATERCES deve produzir a sentença UMA MENSAGEM SECRETA.

11. Mostre como implementar uma pilha usando duas filas. Analise o tempo de execução das operações sobre pilhas.

12. Considere o seguinte conjunto de números 1234 e as seguintes operações:

    - inserir o número 1 em uma pilha
    - inserir o 2 na pilha
    - retirar o 2 da pilha
    - inserir o 3 na pilha
    - inserir o 4 na pilha
    - retirar o 4 da pilha

- retirar o 3 da pilha
- retirar o 1 da pilha

A sequência de saídas do procedimento acima é 2431. Considere agora a sequência 123456.

(a) Podemos obter as sequênncias `325641` e `154623` utilizando um processo semelhante ao do exemplo anterior?

(b) Se `I` e `R` representam respectivamente inserção e remoção da pilha, o exemplo acima pode ser descrito como `IIRIIRRR`. Se possível, descreva as sequências do item anterior em termos de `I` e `R`.

(c) Qual seria uma regra simples para analisar se uma sequênncia de `I`'s e `R`'s é válida?

13. Uma lista encadeada pode ser vista de duas maneiras diferentes, dependendo do papel que seu primeiro nó desempenha. Na lista **com cabeça**, o primeiro nó serve apenas para marcar o início da lista e portanto o seu conteúdo é irrelevante. O primeiro nó é a **cabeça** da lista. Já na lista **sem cabeça**, o conteúdo do primeiro nó é tão relevante quanto o dos demais. A lista está vazia se não tem nó algum. Dadas essas definições, implemente os seguintes itens:

    (a) Implemente o TAD Fila em C++ usando lista encadeada com nó cabeça.

    (b) Implemente o TAD Fila em C++ usando lista encadeada *circular* com nó cabeça. O primeiro elemento da fila ficará no segundo nó e o último elemento ficará no nó anterior à cabeça. Para manipular a fila basta conhecer o endereço do nó cabeça.

    (c) Implemente o TAD Fila em C++ usando lista duplamente encadeada sem nó cabeça. Matenha um ponteiro para o primeiro nó e um ponteiro para o último.

    (d) Implemente o TAD Pilha em C++ usando lista encadeada sem nó cabeça. O topo da pilha será o endereço da primeira célula da lista.

14. Escreva uma função que imprime os elementos de uma pilha na mesma ordem em que eles foram empilhados. O conteúdo da pilha deve ser o mesmo antes e depois da execução da função. Pode ser utilizada uma pilha auxiliar.

15. Escreva uma função para determinar se uma cadeia de caracteres é da forma:

    xCy

    em que x e y são cadeias de caracteres compostas por letras 'A' e/ou 'B', e y é o inverso de x. Isto é, se x = "ABABBA", y deve equivaler a "ABBABA". Em cada ponto, você só poderá ler o próximo caractere da cadeia (é mandatório o uso de pilha).

16. Para um dado número inteiro $n > 1$, o menor inteiro $d > 1$ que divide n é chamado de fator primo. é possível determinar a fatoração prima de n achando-se o fator primo d e substituindo n pelo quociente n/d, repetindo essa operação até que n seja igual a 1. Utilizando um dos TADs vistos em sala (Lista, Pilha ou Fila) para auxiliá-lo na manipulação dos dados, implemente uma função que compute a fatoração prima de um número imprimindo os seus fatores em ordem decrescente. Por exemplo, para n=3960, deverá ser impresso 11 * 5 * 3 * 2 * 2 * 2. Justifique a esquelha do TAD utilizado.

17. [Permutações produzidas pelo desempilhar] Suponha que os números inteiros 1,2,3,4 são colocados, nesta ordem, numa pilha inicialmente vazia. Depois de cada operação de empilhar, você pode retirar zero ou mais elementos da pilha. Cada número retirado da pilha é impresso numa folha de papel. Por exemplo, a sequência de operações `E, E, D, E, D, D, E, D`, onde `E` significa "empilhar o próximo número da sequência" e `D` significa "desempilhar", produz a impressão da sequência 2,3,1,4. Quais das 24 permutações de 1,2,3,4 podem ser obtidas dessa maneira?

18. Faça uma função `pop` alternativa que recebe como entrada um parâmetro n e desempilha n elementos da pilha. A função deve retornar um vetor com os elementos removidos.

19. Usando apenas as funções `push` e `pop`, implemente uma função que receba uma pilha p como entrada e retorna a cópia dessa pilha.

20. Faça um programa que recebe como entrada um número inteiro e retorna seu respectivo valor em binário usando Pilha.

21. Como você implementaria uma fila de pilhas? Uma pilha de filas? Uma fila de filas? Escreva rotinas para implementar as operações corretas para cada uma destas estruturas de dados.

22. Implemente uma fila onde cada item da fila consista em um número variável de inteiros.

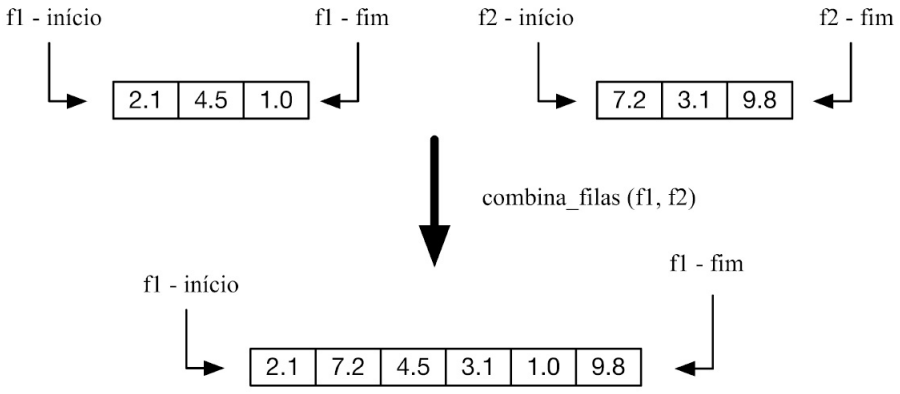
23. No estoque de uma grande empresa todas as caixas possuem pesos: 13, 11 e 7 toneladas. Há três pilhas p_1, p_2 e p_3. Na pilha p_1 encontram-se todas as caixas que chegam no depósito. Com um detalhe: caixas maiores não podem ser empilhadas sobre caixas menores. Implemente uma função chamada `chegada`(Caixa *$nova$, Pilha *p) que efetue o controle das caixas, de forma que caso uma caixa de maior peso do que uma que já está em p_1 deva ser empilhada, então, todas as caixas que estão em p_1 são movidas para as pilhas auxiliares p_2 (contendo somente caixa de 11 toneladas) e p_3 (contendo somente caixas de 7 toneladas) até que se possa empilhar a nova caixa. Depois, todas as caixas são movidas de volta para a pilha p_1. Crie e utilize as funções com os seguintes protótipos:
    `bool Pilha::vazia( );`
    `bool Pilha::cheia( );`
    `void Pilha::empilhar`(Caixa *$nova$);

```
void Pilha::desempilhar( );
Caixa* Pilha::topo( );
```
Obs.: uma caixa deve conter seu peso e descrição.

24. Considere a existência de um tipo abstrato Fila de números reais, cuja interface está definida no arquivo "`fila.h`" da seguinte forma:

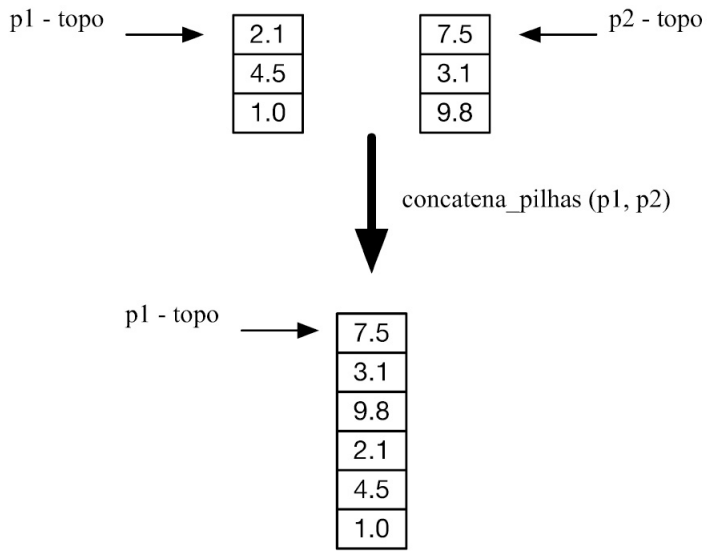
```
typedef struct fila Fila;
Fila* fila_cria(void );
void fila_insere (Fila* f, float v);
float fila_retira (Fila* f);
int fila_vazia (Fila* f);
void fila_libera (Fila* f);
```

Esta fila é definida e implementada sem o uso de classes, apenas por meio de structs e funções. Sem conhecer a representação interna desse tipo abstrato e usando apenas as funções declaradas no arquivo de interface, implemente uma função que receba três filas, `f_res`, `f1`, `f2`, e transfira alternadamente os elementos de `f1` e `f2` para `f_res`, conforme ilustrado na figura a seguir:



Note que, ao final dessa função, as filas `f1` e `f2` vão estar vazias e a fila `f_res` vai conter todos os valores que estavam originalmente em `f1` e `f2` (inicialmente `f_res` pode ou não estar vazia). Se uma fila for maior que a outra, os valores excedentes devem ser transferidos para a nova fila no final.

25. Implemente uma função que recebe duas pilhas `p1` e `p2` e passa todos os elementos da pilha `p2` para a pilha `p1`. A figura a seguir ilustra essa concatenação de pilhas. Note que, ao final dessa função, a pilha `p2` vai estar vazia e a pilha `p1` conterá todos os elementos das duas pilhas.



Implemente uma versão usando recursividade e uma versando usando uma terceira pilha.