Faculdade de Computação (FACOM) - UFMS
Algoritmos e Programação 2
Prof. Dr. Anderson Bessa da Costa

Lista 13 - Pilhas

1. Considere uma pilha de inteiros. Execute a sequência de operações a seguir na pilha, mostrando ao final de cada operação `desempilhe` o valor que será retornado. Responda a questão sem nenhuma implementação.

```
empilhe(33)
empilhe(7)
empilhe(11)
desempilhe()
desempilhe()
empilhe(2)
desempilhe()
```

   Do exercício 2 ao 6, você implementará uma pilha de inteiros em um vetor. Siga o passo a passo.

2. Declare um vetor de inteiros `p` com o tamanho definido por uma constante `N` (inicialmente, defina `N` como uma constante de valor 30). Além disso, declare a variável inteira `t`, que será o índice que utilizaremos para manipular o topo da pilha.

3. Em C, variáveis logo após serem declaradas possuem valores desconhecidos. Torna-se, portanto, imprescindível inicializar a pilha após a sua declaração, a colocando em um estado válido. Na função `main`, inicialize a pilha de modo que ela comece vazia, ou seja, atribua o valor 0 à variável `t`, a fim de representar uma pilha vazia. Lembre-se de que $0 \leq t \leq N$ (de acordo com a nossa implementação em sala).

4. Implemente a função `void empilhe(int p[], int &t, int y)` que insere o elemento `y` na pilha. De acordo com a aula teórica, a nossa pilha deverá armazenar o elemento `y` na posição `t` do vetor `p` e, em seguida, incrementar o índice `t` em 1. Caso a pilha esteja cheia, exiba uma mensagem de erro indicando que ocorreu transbordamento (*overflow*), mais precisamente Transbordamento da Pilha (*Stack Overflow*). Obs.: O argumento `t` está sendo passado por referência, por isso o operador `&`.

5. Implemente a função `int desempilhe(int p[], int &t)` que remove e retorna o elemento pilha mais recentemente inserido, seguindo a política LIFO (*Last In, First Out*). A função deve copiar o elemento da posição `t - 1` do vetor `p` e, em seguida, decrementar o índice `t`. A operação de desempilhar deve sempre retornar o elemento que foi removido. Caso a pilha esteja vazia, exiba uma mensagem de erro indicando que ocorreu esvaziamento (*underflow*), mais precisamente Esvaziamento da Pilha (*Stack Underflow*).

6. Na `main`, teste as operações de empilhar e desempilhar para verificar se a sua pilha implementada segue corretamente a política LIFO. Utilize a sequência de operações do exercício 1 e verifique se o resultado obtido é o esperado.

7. Visualização é uma parte sempre fundamental para compreensão de algoritmos. Copie o código desenvolvido e o cole em: `https://pythontutor.com/cpp.html#mode=edit`. Visualize o passo a passo do código construído.

8. Implemente uma pilha em uma lista encadeada com célula-cabeça.

9. Implemente uma pilha em uma lista encadeada sem célula-cabeça. A pilha será especificada pelo endereço da primeira célula da lista.

10. Escreva um algoritmo que inverta a ordem de uma fila (você pode utilizar outras estruturas filas ou pilhas). Essa inversão não pode ser feita com você manipulando diretamente o vetor, mas sim somente fazendo uso das operações suportadas por filas e pilhas, no caso `enfileire`, `desenfileire`, `empilhe` e `desempilhe`.