

Estruturas Lineares

Parte A

1. **Complete a sentença:** Em uma estrutura linear, os elementos formam uma _____.
2. Marque as estruturas abaixo que são estruturas lineares.
 - ☐ *array*
 - ☐ árvore binária de busca
 - ☐ fila
 - ☐ grafo
 - ☐ *heap*
 - ☐ lista
 - ☐ pilha
 - ☐ vetor
3. Marque as bibliotecas de C++ abaixo que implementam estruturas lineares.
 - ☐ `bitset`
 - ☐ `deque`
 - ☐ `map`
 - ☐ `queue`
 - ☐ `set`
 - ☐ `stack`
 - ☐ `vector`
4. **Preencha os espaços com uma das opções sugeridas.** Um tipo de dado abstrato é definido pela sua _____ (implementação/interface). Cada tipo de dado abstrato tem _____ (uma única implementação/várias implementações possíveis) e _____ (uma única interface/várias interfaces possíveis).
5. Defina:
 - (a) uma lista encadeada
 - (b) uma lista duplamente encadeada
 - (b) uma lista auto-organizável
 - (c) uma *skip list*
6. Em uma estrutura FIFO,
 - ☐ O primeiro elemento a entrar é o primeiro a sair
 - ☐ O primeiro elemento a entrar é o último a sair
 - ☐ O último elemento a entrar é o primeiro a sair
 - ☐ O último elemento a entrar é o último a sair
7. Em uma estrutura LIFO,
 - ☐ O primeiro elemento a entrar é o primeiro a sair
 - ☐ O primeiro elemento a entrar é o último a sair
 - ☐ O último elemento a entrar é o primeiro a sair

- ☐ O último elemento a entrar é o último a sair

8. Complete o quadro abaixo assinalando, em cada coluna, as operações “eficientes” da estrutura.

Característica	<i>array</i>	vetor	lista	pilha	fila
Acesso aleatório aos elementos					
Inserção					
Inserção no início					
Inserção no fim	x				
Remoção					
Remoção no início					
Remoção no fim		x			
Busca					

Parte B

9. Dê uma definição precisa de estrutura linear.
10. Usando a definição acima, quais das estruturas listadas a seguir seriam consideradas estruturas lineares?
- ☐ *arrays*
 - ☐ filas
 - ☐ listas circulares
 - ☐ listas com prioridades
 - ☐ pilhas
 - ☐ *skip lists*
11. Implemente, em C++, uma lista circular.
12. Escreva um algoritmo, em pseudocódigo, que implemente a inserção de um elemento em uma lista duplamente encadeada.
13. Quais são as vantagens e as desvantagens de se substituir uma lista duplamente encadeada por uma lista simplesmente encadeada?
14. O algoritmo abaixo implementa uma pilha em Python. Reimplemente o mesmo código em C++.

```

1 class Stack:
2
3     def __init__(self):
4         self.elems = []
5
6     def push(self, x):
7         self.elems.append(x)
8
9     def pop(self):
10        return self.elems.pop()
11
12    def __str__(self):
13        xs = [str(x) for x in self.elems[::-1]]
14        return '(' + ' '.join(xs) + ')'
15
16    def empty(self):
17        return len(self.elems) == 0
18
19
20 if __name__ == '__main__':
21
22     s = Stack()
23     print 'Nova pilha = {}, vazia? {}'.format(s, s.empty())
24
25     s.push(1)
26     s.push(2)
27     s.push(3)
28     print 'Apos insercoes =', s
29
30     print 'topo = {}'.format(s.pop())
31     print 'nova pilha = {}, vazia? {}'.format(s, s.empty())

```

15. O pseudocódigo abaixo implementa a remoção de um elemento em uma lista encadeada. Implemente o algoritmo em sua linguagem de preferência.

Algoritmo 1 Remoção de elemento de lista encadeada

Entrada: Uma lista encadeada L e a posição do elemento n a ser removido

Saída: A lista remanescente, após a remoção do n -ésimo elemento

```
1. if  $L.size \geq n$  then
2.   return
3. end if
4.
5.  $p \leftarrow L.head$ 
6.  $q \leftarrow null$ 
7.  $i \leftarrow 1$ 
8.
9. while  $i < n$  do
10.   $q \leftarrow p$ 
11.   $p \leftarrow p.next$ 
12.   $i \leftarrow i + 1$ 
13. end while
14.
15. if  $q \neq null$  then
16.   $q.next \leftarrow p.next$ 
17. end if
18.
19.  $L.head \leftarrow p.next$ 
20. delete  $p$ 
21.
22. return
```
