INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO RIO GRANDE DO NORTE
Campus Nova Cruz – Código INEP: 24083003
Av. José Rodrigues de Aquino Filho, Nº 640, RN 120, Alto de Santa Luzia, CEP 59215-000, Nova Cruz (RN)
CNPJ: 10.877.412/0015-633 – Telefone: (84) 4005-4107

Texto Complementar:

Listas, Pilhas e Filas Sequenciais

(EDL – TADS4M)

1. Introdução

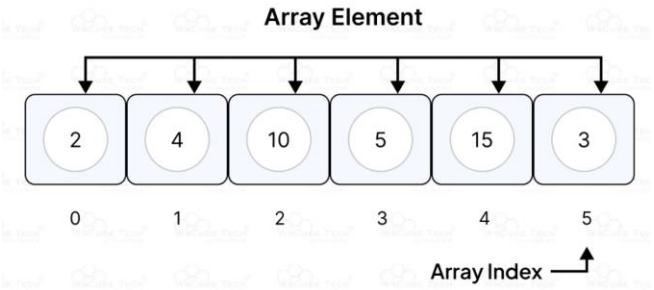
No estudo de Estruturas de Dados, as listas, pilhas e filas são abstrações fundamentais para organização e manipulação da informação. Quando implementadas de forma sequencial, essas estruturas utilizam **vetores (arrays)** como mecanismo de armazenamento, o que implica em restrições e características específicas quanto a acesso, inserção e remoção de seus elementos. Ao implementarmos a classe `Array` em nossas primeiras aulas da disciplina, fazendo uso de um vetor alocado dinamicamente já foi possível observar algumas dessas características.

1. Lista Linear Sequencial

Uma lista linear sequencial é uma coleção de elementos do mesmo tipo, armazenados em posições consecutivas de memória. Formalmente, uma lista L pode ser definida como:

$$L = \{a_1, a_2, \dots, a_n\} \ tal\ que\ \ n \geq 0$$

onde cada a_i pertence a um domínio D e n representa o tamanho da lista, ou seja, o número de elementos que ela armazena em um dado momento. Esta definição muito se assemelha à definição de um vetor da programação.



Quando uma lista é implementada sequencialmente, os seus n elementos são armazenados em um vetor V de tamanho m (capacidade máxima da lista).

$$V[0 \dots m-1], com\ n\ elementos\ armazenados\ (n \leq m)$$

Caso o vetor seja dinâmico, pode-se realizar operações de realocação de memória, fazendo com que o valor de m possa variar ao longo do tempo de uso da lista, conforme a necessidade da aplicação. Isso, logicamente, traz custos e cuidados adicionais à implementação, mas em contrapartida, elimina a limitação de quantidade de elementos que a lista sequencial em um vetor estático apresenta.

As operações fundamentais em uma lista sequencial são: inserção em uma posição k da lista, remoção de um elemento na posição k da lista, busca sequencial ou binária (dependendo da ordenação) e o acesso direto a uma posição k da lista. O acesso direto por índice tem custo assintótico $O(1)$, enquanto inserções e remoções em posições arbitrárias custam $O(n)$, devido ao possível deslocamento de elementos. Isso foi percebido na implementação da classe `Array`, feita em nossas primeiras aulas. Em relação aos algoritmos de busca na lista linear, a busca sequencial tem complexidade de pior caso de $O(n)$ e a busca binária de $O(logn)$. Deve-se destacar, que a busca binária só é aplicável se os elementos da estrutura estiverem ordenados.

As listas podem ser utilizadas no armazenamento de registros em um sistema de gestão (lista de alunos em um diário de classe), catálogo de produtos em um sistema de *e-commerce*, *playlists* de músicas ou vídeos, entre outros. Além disso, como veremos ao longo da disciplina de Estruturas de Dados Lineares, as listas são utilizadas como base para a implementação de outras estruturas de dados, como as pilhas e filas que ainda serão definidas neste documento.

2. Pilha Sequencial (*Stack*)

Uma pilha é uma estrutura linear que segue o princípio LIFO (do inglês *Last In*, *First Out* – último a entrar, primeiro a sair). Ou seja, o último elemento inserido nessa estrutura será o primeiro a ser removido. Isso assemelha-se a um conjunto de pratos empilhados, conforme a figura a seguir. O primeiro prato a ser removido da pilha será o que se encontra no topo da pilha, entretanto, ele foi o último a ser adicionado na pilha de pratos.

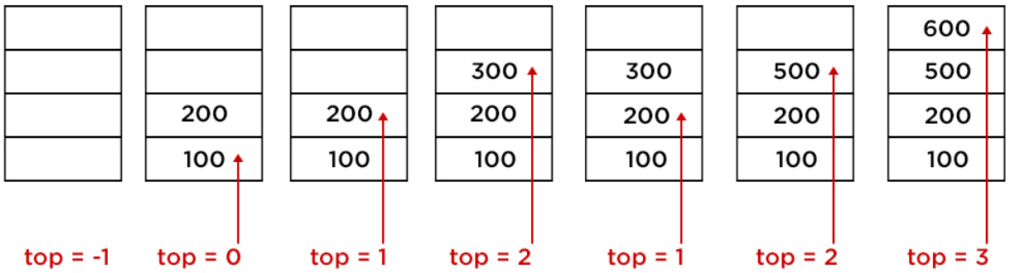
Pode-se afirmar que uma pilha sequencial é uma coleção de elementos de mesmo tipo, armazenados em posições consecutivas de memória. Formalmente, uma lista P pode ser definida como:

$$P = \{a_1, a_2, \dots, a_n\} \ tal\ que\ \ n \geq 0$$

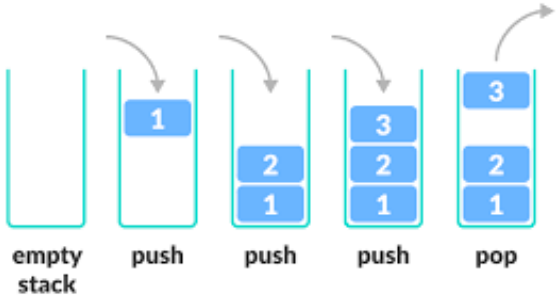
onde a_n é o topo da pilha. Na pilha sequencial, os seus n elementos são armazenados em um vetor V de tamanho m (capacidade máxima da pilha) e um índice t que indica a posição do topo da pilha.

$$V[0 \dots m-1], com\ n\ elementos\ armazenados\ (n \leq m)$$

Assim como nas listas sequenciais, também é possível implementar a estrutura de dados pilha sequencial utilizando um vetor dinâmico. Isto permite que a capacidade máxima da pilha se torne variável ao longo do seu tempo de uso.



As operações fundamentais em uma pilha são: **i**. `push` (empilhar), utilizada para inserir um elemento no topo da pilha; **ii**. `pop` (desempilhar), que retorna e remove o elemento no topo da pilha e **iii**. `top` ou `peek`, que consulta/retorna o elemento do topo sem removê-lo da pilha. Todas essas operações possuem complexidade constante, $O(1)$, uma vez que elas são realizadas por meio de acesso direto a uma posição específica do vetor: a posição de topo, dada pelo índice t.



A estrutura de dados pilha pode ser encontrada na navegação de páginas web, quando voltamos e avançamos no histórico de páginas visitadas, nas operações de *undo*/*redo* (desfazer a última ação) que é disponibilizada em inúmeras aplicações, gerenciamento de chamadas de funções por meio da pilha de execução (*call stack*), avaliações de expressões aritméticas em notação pós-fixa, entre outros.

3. Fila Sequencial (*Queue*)

Uma fila é uma estrutura linear que segue o princípio FIFO (do inglês *First In*, *First Out* – primeiro a entrar, primeiro a sair). Ou seja, o primeiro elemento inserido nessa estrutura será o primeiro a ser removido. Se fizermos uma relação com uma fila de espera, sem considerar prioridades (veremos lista de prioridades em um outro momento), o elemento que estiver na fila a mais tempo (o que chegou primeiro) será o primeiro a ser removido ou atendido.

Uma fila sequencial é uma coleção de elementos do mesmo tipo, armazenados em posições consecutivas de memória. Formalmente, uma lista F pode ser definida como:

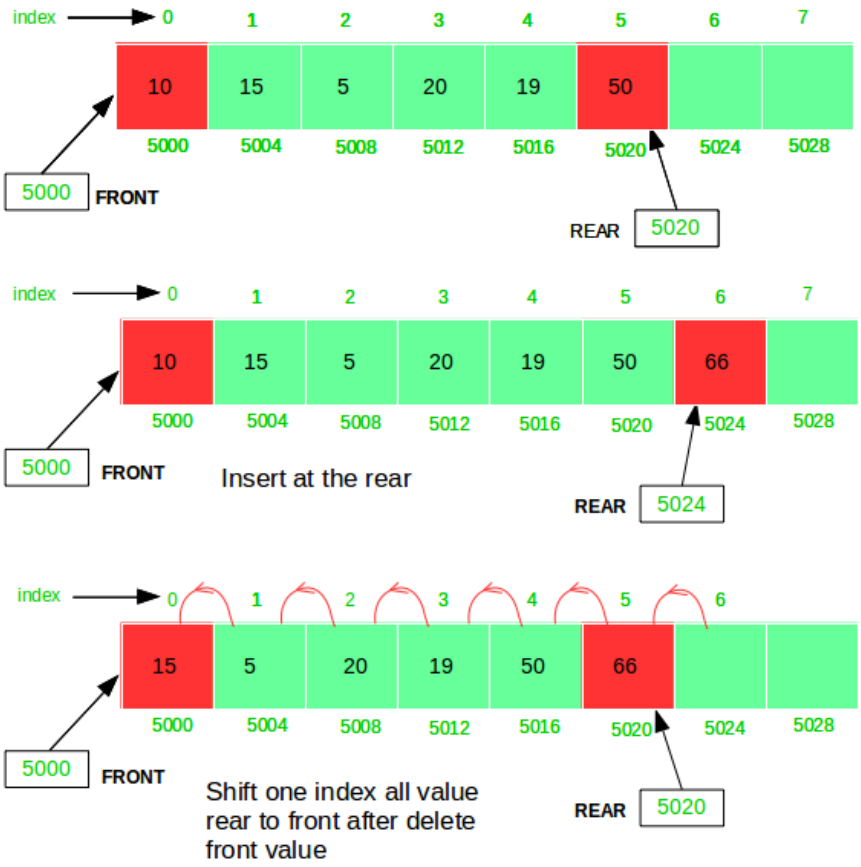
$$F = \{a_1, a_2, \dots, a_n\} \ tal\ que\ \ n \geq 0$$

onde a_1 é o primeiro elemento da fila (frente da fila) e a_n é o último elemento da fila (final da fila). Na pilha sequencial, os seus n elementos são armazenados em um vetor V de tamanho m (capacidade máxima da fila).

$$V[0 \ldots m-1], com\ n\ elementos\ armazenados\ (n \leq m)$$

Assim como nas listas e pilhas sequenciais, também é possível implementar a estrutura de dados fila sequencial utilizando um vetor dinâmico. Isto permite que a capacidade de armazenar elementos na fila se torne variável ao longo do seu tempo de uso.

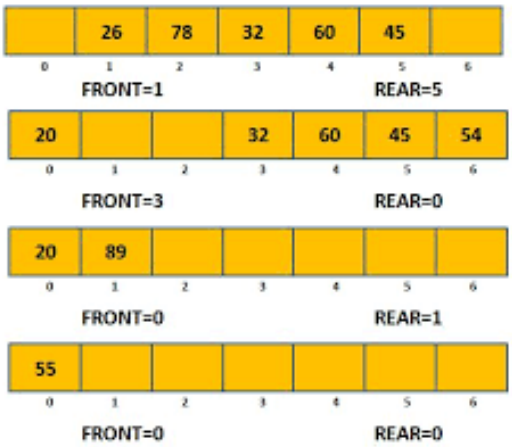
As operações fundamentais em uma fila são: **i**. `enqueue` (enfileirar), operação utilizada para inserir um elemento no final da fila; **ii**. `dequeue` (desenfileirar), que retorna e remove o elemento do início da fila e **iii**. `front`, que consulta/retorna o elemento do início da fila. A operação de inserção de um elemento na fila ocorre em $O(1)$, uma vez que ela é realizada por meio de acesso direto ao índice do final da fila. Entretanto, após remover o primeiro elemento da fila, é necessário deslocar todos os demais elementos em direção ao início da fila, de forma semelhante ao que acontece em uma fila de caixa de supermercado após um cliente da fila ter seu atendimento concluído. Essa característica faz com que a operação `dequeue` tenha comportamento assintótico $O(n)$.



A estrutura de dados fila pode ser encontrada, por exemplo, no gerenciamento de impressão em sistemas operacionais, controle de requisições a servidores web, gestão de transmissão de dados em redes de computadores, organização de tarefas do processador e em sistemas de atendimento ao cliente (banco, *call center*, clínicas médicas, entre outros), garantindo que as solicitações sejam processadas na ordem em que elas chegam.

3.1 Fila Circular Sequencial (*Circular Queue*)

Uma forma de contornar o problema de deslocamento observado ao remover um elemento da fila é implementar a fila de forma circular. Além disso, permite reutilizar o espaço de memória disponível no vetor de forma mais eficiente e contínua. Nessa estrutura, torna-se crucial o uso dos índices f (*front* ou frente da fila) e r (*rear* ou traseira/final da fila). A fila é chamada circular porque, ao alcançar o final do vetor, os índices retornam ao início.



O comportamento circular é obtido a partir de um cálculo modular da nova posição desses índices após a realização das operações na fila circular.

- **Inicialização**. No início, define-se $f = 0$ e $r = 0$, com a fila vazia.
- **Inserção** (`enqueue`). A inserção somente ocorre se a fila não estiver cheia (considerando implementação em vetor estático). Nesse caso, o novo elemento é inserido na posição r. Em seguida, atualiza-se o índice r com a seguinte expressão:

$$r = (r + 1)\ mod\ m$$

- **Remoção** (`dequeue`). A remoção somente ocorre se a fila não estiver vazia. Nesse caso, o elemento da posição f é removido e este índice é atualizado:

$$f = (f + 1)\ mod\ m$$

- **Condição de fila vazia**. A fila é considerada vazia quando $f = r$.
- **Condição de fila cheia**. Para distinguir entre fila cheia e fila vazia, considera-se que a fila está cheia quando:

$$(r + 1)\ mod\ m = f$$

Assim, uma posição do vetor permanece inutilizada, garantindo que as condições de cheio e vazio sejam distintas.