## Arrays vs Listas encadeadas

Arrays e listas servem para que se armazene múltiplos itens na memória de seu computador.

- **Array:**
  Usar um Array significa que seus itens devem ser armazenados continuamente, ou seja, em endereços sequenciais da memória do computador, uma do lado da outra. Suponha que, utilizando um array você deseje adicionar mais um item, porém o próximo endereço está ocupado. É como se você fosse ao cinema com seus amigos, encontrassem um lugar para se sentar e posteriormente outro amigo se juntasse a vocês, mas não haveria lugar para ele. Todos precisam se mover e encontrar um local onde todos coubessem. Nesse caso, você precisaria solicitar ao computador um espaço de memória que todos coubessem e então se moveriam para lá. Dessa maneira, adicionar itens a um array se torna um processo muito lento… Se outro amigo aparecesse, vocês ficariam sem lugar suficiente novamente e mais uma vez precisam se mover. Uma forma de resolver isso seria 'reservando' espaços: suponha que vocês inicialmente são um grupo de tres pessoas, mas por via das dúvidas garantem 10 lugares, então podemos receber mais 7 amigos sem a necessidade de nos movermos. Mas existem algumas desvantagens:

    - E se nenhum outro amigo vier? Então a memória será desperdiçada! Você não está usando a memória e mais ninguém também poderá utilizá-la…
      .
    - E se mais de 7 amigos vierem? Então voltaremos ao problema inicial…

  Listas  Encadeadas resolvem esse problema de adição de itens.

- **Listas Encadeadas:**
  Com listas encadeadas seus itens podem estar em qualquer lugar da memória, pois cada item armazena o endereço do próximo item, então um monte de endereços aleatórios da memória estão ligados.
  Com listas encadeadas você nunca precisa mover seus itens. Pensando no problema descrito no tópico sobre Arrays, a estratégia da lista encadeada seria como se você e seus amigos fossem a um cinema lotado que não possuísse lugar para todos se sentarem juntos. Assim vocês optam por se dividirem e assistir o filme mesmo assim. Pensando computacionalmente, se você estivesse buscando espaço para armazenar um array de 10.000 elementos e sua memória possui exatamente 10.000 slots livres, com lista encadeada você ainda consegue arrumar um lugar para seu array, sem desperdício de memória ou necessidade de alocação.

  Se listas encadeadas são muito melhores para inserções, pra que servem os arrays?

  Imagine que você queira obter o último elemento de uma lista encadeada. Você não pode fazer isso porque você simplesmente não sabe o endereço dele. Dessa forma, você teria que ir ao primeiro elemento, pegar o endereço do segundo, ir até o

segundo, pegar o endereço do terceiro e assim por adiante até alcançar o elemento desejado…
Listas encadeadas são ótimas se você quiser ler todos os itens de forma sequencial, mas péssimas se você quiser um item específico.
Com array o problema está resolvido, você sabe o endereço de cada item, imagine que você queira encontrar o quinto elemento de um array. Você sabe que o primeiro item está na casa X, então, obviamente o quinto está na quarta casa após a casa X.

Arrays são ótimos para leituras não sequenciais, pois você consegue encontrar qualquer elemento instantaneamente.

- **Prós do Array:**
  - Leitura rápida de itens, principalmente de itens específicos.

- **Contras do Array:**
  - Inserção demorada.

- **Prós da Lista Encadeada:**
  - Inserção rápida;
  - Baixo desperdício de memória;

- **Contras da Lista Encadeada:**
  - Necessidade de leitura sequencial para encontrar um item específico.

## Exercício:

Suponha que você esteja criando um aplicativo para acompanhar suas finanças e todos os dias você anote o que gastou e onde gastou. No final do mês, você lerá a lista e revisará seus gastos. Você prefere utilizar um array ou uma lista encadeada para desenvolver o aplicativo?

**Resposta:** Lista encadeada, pois você faz muitas inserções (diariamente) e poucas leituras (1x por mês).

## Arrays vs Listas encadeadas: Inserindo um elemento no meio

Numa lista encadeada, se você desejar inserir um elemento no meio basta alterar o endereço no qual o elemento anterior aponta, agora apontará para o endereço do seu novo elemento e o novo elemento aponta para o elemento que seu anterior apontava anteriormente.

Num array você precisaria mover todos os itens que estao abaixo do endereço de inserção e se caso não houvesse espaço para todos os elementos seria necessário ainda mover o array.

## Arrays vs Listas Encadeadas: Deleção

Na lista encadeada ao deletar qualquer elemento basta apenas mudar o endereço no qual o elemento anterior apontará.

No array é necessário que tudo seja movido quando um elemento é deletado. **Obs**: ao contrário do que acontece nas inserções, a deleção sempre funcionará, pois você está diminuindo o uso de memória então nunca terá o perigo da falta da mesma, como no caso da inserção.

OBS: inserções e deleções só terão tempo de execução O(1) se você souber exatamente a localização do elemento a ser manipulado. Por isso, em listas encadeadas é comum acompanhar o primeiro e último elemento, para que o tempo de excluí-los seja O(1). Para o restante devemos localizá-lo sequencialmente e então excluí-lo. Esse é o trade off da comparação entre arrays e lista encadeadas e o que desempata isso seria o tamanho da base a ser manipulada e também a localização do elemento a ser manipulado.

## Arrays vs Lista encadeada: Quem é mais utilizado?

Obviamente isso depende de caso para caso, mas arrays são mais comuns por permitirem o acesso aleatório. Existem dois tipos de acesso: aleatório, quando voce sabe exatamente a localização do elemento a ser manipulado e sequencial, tendo que consultar um a um. Listas encadeadas só conseguem lidar com acesso sequencial.