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:

Estrutura de Dados Array em C++
(EDL – TADS4M)

1. Introdução

Os *arrays*, também conhecidos como vetores, apesar de ser uma estrutura de dados linear básica, são amplamente utilizados e fundamentais no desenvolvimento de software. Eles atuam como um contêiner capaz de armazenar um conjunto de elementos do **mesmo tipo de dado** em posições de memória contíguas. Essa característica de armazenamento sequencial é crucial para a eficiência de muitas operações efetuadas em vetores. Deve-se destacar que as listas lineares sequenciais e suas derivações são comumente implementadas em *arrays*.

2. Armazenamento e Acesso aos Dados

A forma como os vetores organizam os dados permite um acesso direto e rápido a qualquer elemento. Como os elementos são armazenados sequencialmente na memória, é possível calcular a posição exata de um elemento a partir de seu **índice** (sua posição na sequência, geralmente começando em 0). Essa capacidade de acesso direto é conhecida como acesso **aleatório** ou **por índice**, e é uma das principais vantagens dos vetores.

Por exemplo, se um vetor de inteiros $\boldsymbol{v}$ armazena 10 elementos e o seu primeiro elemento encontra-se armazenado no endereço de memória $\boldsymbol{M}$, o elemento $\boldsymbol{v}[\boldsymbol{i}]$ (o elemento no índice i do vetor) estará localizado no endereço $\boldsymbol{M + i * tamanho\_do\_inteiro}$, em que $\boldsymbol{tamanho\_do\_inteiro}$ corresponde ao espaço em memória para armazenar um número inteiro. Essa previsibilidade no cálculo de endereços torna a recuperação de um elemento em um vetor uma operação de tempo constante, que em termos de complexidade de algoritmos, utilizando a notação Big O, é representada como $O(1)$.

3. Vetores Estáticos *vs* Vetores Dinâmicos

Ao trabalhar com vetores, é importante distinguir entre duas abordagens principais de alocação de memória:

- **Vetores Estáticos:** neste caso, o tamanho do vetor é definido no momento da compilação e não pode ser mais alterado durante a execução do programa. Como *vantagens*, esse tipo de vetor é simples de utilizar, possui acesso rápido e gerenciamento automático de memória. Como *desvantagens*, os vetores estáticos possuem tamanho fixo, podendo levar a desperdício de memória caso a necessidade real de armazenamento seja menor do que o tamanho declarado, ou a uma situação de *overflow* se o tamanho for excedido.

- **Vetores Dinâmicos:** o tamanho dos vetores dinâmicos pode ser alterado em tempo de execução. A memória é alocada, permitindo que o vetor cresça ou encolha conforme a necessidade. Na linguagem de programação C++, isso é frequentemente implementado usando ponteiros e alocação dinâmica de memória, utilizando `new` para alocação de memória e `delete` para desalocar o espaço de memória utilizado. Esse tipo de vetores possui como *vantagens* a flexibilidade de tamanho, o que permite a adaptação a diferentes quantidades de dados, e o uso mais eficiente da memória quando a necessidade de armazenamento é variável. Por outro lado, como *desvantagens*, os vetores dinâmicos apresentam uma maior complexidade no gerenciamento de memória, uma vez que as operações de redimensionamento podem ser custosas em termos de tempo (requerem cópia de dados para um novo local de memória). Além disso, estes vetores estão suscetíveis a erros de memória (vazamento de memória – *memory leak* e acessos indevidos) quando não gerenciados corretamente.

4. Vantagens dos Vetores como Estruturas de Dados

- **Acesso rápido por índice - $O(1)$:** a capacidade de acessar qualquer elemento diretamente pelo seu índice é uma das maiores forças dos vetores.
- **Simplicidade:** a lógica de acesso e manipulação é relativamente simples de entender e implementar.
- **Uso eficiente de memória (para vetores estáticos):** quando o tamanho é conhecido antecipadamente, vetores estáticos podem ser muito eficientes em termos de uso de memória.

5. Desvantagens dos Vetores como Estruturas de Dados

- **Inserção e remoção no meio do vetor - $O(n)$:** inserir ou remover um elemento no meio de um vetor (ou no início) é uma operação que, dependendo do tipo de estrutura, geralmente requer o deslocamento de muitos outros elementos. Isso torna essas operações ineficientes, especialmente para vetores grandes.
- **Tamanho fixo (para vetores estáticos):** o tamanho fixo de vetores estáticos pode ser um grande obstáculo em cenários onde a quantidade de dados é imprevisível. Vetores dinâmicos atenuam isso, mas o ato de redimensionar o vetor apresenta um custo.
- **Sobrecarga de memória (para vetores dinâmicos):** para garantir espaço para crescer, vetores dinâmicos podem alocar mais memória do que o estritamente necessário em determinado momento, levando a um certo desperdício.

6. Reflexão

Considerando as limitações de inserção e remoção no meio de um vetor, que muitas vezes exigem o deslocamento de vários elementos, podemos nos perguntar:

> ➢ *Seria possível definir outras estruturas de dados lineares que permitissem operações de inserção e remoção mais eficientes, mesmo em posições intermediárias, sem a necessidade de mover tantos elementos quanto em um vetor? Como essas estruturas poderiam gerenciar o acesso aos seus elementos e a alocação de memória para garantir essa eficiência?*