# DSA Study Notes Day 9:

## Chapter 8 - Vectors (Introduction to STL)

---

### Introduction to Vectors

- **Vectors** are similar to arrays but with more flexibility.
- Unlike arrays, vectors are dynamic in size, meaning they can grow or shrink as elements are added or removed.
- Vectors are part of the **Standard Template Library (STL)** in C++.
- **Why Use Vectors?**:
  - Vectors are useful when you need a collection of elements but don't know the exact size at the time of creation.
  - While arrays have a fixed size, vectors are flexible and can dynamically resize based on the number of elements.

### Vector Syntax & Creation

- To declare a vector, you need to include the `<vector>` header and specify the data type:

  ```
  vector<int> vec;  // Declares a vector of integers
  ```

### Vector Functions

Vectors come with several built-in functions that make operations like adding, removing, or accessing elements much easier compared to arrays.

1. **size()**: Returns the number of elements currently in the vector.

   ```
   cout << "Size of the vector: " << vec.size();
   ```

2. **push_back()**: Adds an element to the end of the vector.

   ```
   vec.push_back(10);  // Adds 10 to the vector
   ```

3. **pop_back()**: Removes the last element from the vector.

   ```
   vec.pop_back();  // Removes the last element
   ```

4. **front()**: Returns the first element of the vector.

   ```
   cout << "Front element: " << vec.front();
   ```

5. **back()**: Returns the last element of the vector.

```
    cout << "Back element: " << vec.back();
```

### Static vs. Dynamic Allocation

- **Static Allocation**: Involves arrays where the size is fixed at compile time.
- **Dynamic Allocation**: Involves vectors, where the size can change during runtime as elements are added or removed.

## Common Vector Problems

1. **Significant Numbers Problem**:
   - This is a common problem where vectors are used to store significant values (like primes, unique numbers, etc.) and perform operations like sorting, searching, or filtering.

## Home Task

1. **Write code for linear search in a vector**.
   - Implement a function that takes a vector and a target value, then searches for that value using a linear search algorithm.
2. **Write code to reverse a vector**.
   - Write a function that reverses the elements of a vector.

## Conclusion

Vectors provide a flexible and powerful alternative to arrays in C++, especially when dealing with data whose size can change dynamically. The built-in functions of vectors make operations such as adding, removing, and accessing elements straightforward and efficient. By mastering vectors, you'll gain a deeper understanding of how dynamic data structures work in C++.

Day 8 Notes

***Prepared by Munawar Johar***