Sequence Containers

- **Description**:
  - Sequence containers store elements in a linear sequence. The order of elements is maintained as per the sequence in which they were inserted.
- **Examples**:
  - **`vector`**: Dynamic array that provides random access to elements and automatically resizes.
  - **`deque`**: Double-ended queue that allows insertion and deletion at both ends.
  - **`list`**: Doubly-linked list, efficient for insertion and deletion at any position.
  - **`forward_list`**: Singly-linked list, more memory efficient than **`list`**.
  - **`array`**: Fixed-size array, the size is determined at compile time.
  - **`string`**: Specialized container for characters, providing string manipulation functions.
- **Characteristics**:
  - Provide methods for inserting, deleting, and accessing elements.
  - Suitable for scenarios where the order of elements matters.

Associative Containers

- **Description**:
  - Associative containers store elements in a way that allows fast retrieval based on keys. The elements are ordered by keys.
- **Examples**:
  - **`set`**: Collection of unique elements, ordered by keys.
  - **`multiset`**: Collection of elements, allows duplicates, ordered by keys.
  - **`map`**: Collection of key-value pairs with unique keys, ordered by keys.
  - **`multimap`**: Collection of key-value pairs, allows duplicate keys, ordered by keys.
- **Characteristics**:
  - Provide methods for fast search, insertion, and deletion.
  - Suitable for scenarios where fast lookup by key is required.

Unordered Containers

- **Description**:
  - Unordered containers are similar to associative containers but do not maintain any specific order. They provide average constant time complexity for search, insert, and delete operations.
- **Examples**:
  - **`unordered_set`**: Collection of unique elements, unordered.
  - **`unordered_multiset`**: Collection of elements, allows duplicates, unordered.
  - **`unordered_map`**: Collection of key-value pairs with unique keys, unordered.
  - **`unordered_multimap`**: Collection of key-value pairs, allows duplicate keys, unordered.
- **Characteristics**:
  - Use hash tables internally for fast operations.
  - Suitable for scenarios where order is not important but fast access is.

Container Adapters

- **Description**:
  - Container adapters are not containers themselves but provide a different interface for sequential containers. They are built on top of other container types and provide restricted access to the underlying elements.
- **Examples**:
  - **`stack`**: LIFO (Last In First Out) structure built on top of other containers.
  - **`queue`**: FIFO (First In First Out) structure built on top of other containers.
  - **`priority_queue`**: Elements ordered by priority, highest priority element at the top.
- **Characteristics**:
  - Adapt the functionality of sequence containers to provide different behaviors (e.g., LIFO for **`stack`**, FIFO for **`queue`**).
  - Suitable for scenarios where specific access patterns are required (e.g., stack for LIFO operations).