Tutorial 70 - Containers in C++ STL

Introduction

- In the **previous tutorial**, we briefly introduced the three components of STL:
  ✅ **Containers** – Store data
  ✅ **Algorithms** – Process data
  ✅ **Iterators** – Point to elements in a container
- In this tutorial, we will focus on **Containers** in detail.

Types of Containers in STL

📌 **Containers are categorized into three types:**

1️⃣ **Sequence Containers** – Store data **linearly**.
2️⃣ **Associative Containers** – Store data in a **key-value** manner, optimized for fast access.
3️⃣ **Derived Containers** – Built on top of **sequence/associative containers**, used for specialized operations.

1️⃣ Sequence Containers

- **Definition:** Store elements in a **linear order** (like an array).
- **Examples:** `vector`, `list`, `deque`.
- **Illustration:**

```
10 → 20 → 30 → 40 → 50
(Elements are stored sequentially)

```

📌 **Common Sequence Containers and Their Properties:**
✔ **Vector (`vector`)**
✅ Fast **random access** to elements.
✅ **Insertion and deletion** at random positions **is slow** (except at the end).
✅ **Insertion at the end** is fast.

✔ **List (`list`)**
✅ **Random access is slow** (uses pointers to traverse elements).
✅ **Insertion and deletion at any position is fast** (pointers make manipulation easy).

✔ **Deque (`deque`)**
✅ Allows **insertion and deletion** from **both ends**.

2️⃣ Associative Containers

- **Definition:** Store data in a **tree-like structure** for **fast access**.
- **Examples:** `set`, `multiset`, `map`, `multimap`.

📌 **Characteristics:**
✔ **Optimized for quick access** – Searching, inserting, and deleting are **fast**.
✔ **Elements are stored in a sorted order**.
✔ **No direct/random access like arrays or vectors**.

3️⃣ Derived Containers

- **Definition:** Built using **sequence or associative containers**, designed for **specific functionalities**.
- **Examples:** `stack`, `queue`, `priority_queue`.

📌 **Common Derived Containers and Their Properties:**

✔ **Stack (`stack`)**

✅ Works on **LIFO (Last In, First Out)** principle.

✅ Example: **Undo operations in text editors**.

✔ **Queue (`queue`)**

✅ Works on **FIFO (First In, First Out)** principle.

✅ Example: **Processing tasks in order (e.g., printer queue, customer service queue).**

✔ **Priority Queue (`priority_queue`)**

✅ Elements are **sorted based on priority** instead of insertion order.

✅ Example: **Dijkstra's Algorithm for shortest path finding**.

When to Use Which Container?

📌 Sequence Containers

✅ Use **Vector** when:

- You need **fast random access**.
- Insertions/deletions mostly occur at the **end**.

✅ Use **List** when:

- Insertions/deletions happen **anywhere** (not just the end).
- **Random access is not a priority**.

📌 Associative Containers

✅ Use **Set/Map** when:

- You need **fast search and retrieval**.
- You don't need random access.

✅ Use **Multiset/Multimap** when:

- You need to **store duplicate keys**.

📌 Derived Containers

✅ Use **Stack** when:

- You need **LIFO operations** (e.g., function call stack, undo feature).

✅ Use **Queue** when:

- You need **FIFO operations** (e.g., task scheduling).

✅ Use **Priority Queue** when:

- You need elements to be **processed based on priority**.

Key Takeaways

✔ **Sequence Containers store elements in a linear order**.
✔ **Associative Containers store elements in a key-value pair for fast retrieval**.
✔ **Derived Containers are specialized containers for specific use cases**.
✔ **Choosing the right container depends on the type of operations you need**.

🚀 **Next Topic: Vectors in STL!** Stay tuned! 🔥

Short Notes

📌 What Are Containers?

- Containers are **objects that store data** in different ways.
- **Three types of containers in STL:**
  ✔ **Sequence Containers** – Linear storage (e.g., `vector`, `list`).
  ✔ **Associative Containers** – Fast search using key-value pairs (e.g., `map`, `set`).
  ✔ **Derived Containers** – Specialized operations (e.g., `stack`, `queue`).

📌 Key Properties of Containers

✔ **Vector:** Fast random access, slow insertion/deletion except at the end.
✔ **List:** Slow random access, fast insertion/deletion at any position.
✔ **Set/Map:** Fast search and retrieval, but no random access.
✔ **Stack:** LIFO (Last In, First Out) – Used for undo operations.
✔ **Queue:** FIFO (First In, First Out) – Used for processing tasks in order.

🚀 **Next Topic: Vectors in STL!** Keep learning! 🔥