CyclicList C++ Template Class Documentation

CyclicList<T> - Cyclic Doubly Linked List Template Class

**Overview:**

---------

CyclicList<T> is a container implemented in the style of the C++ Standard Template Library (STL) that stores elements in a cyclic doubly linked list structure. It fully supports iterator protocols, standard concepts, error handling, and standard operators.

**Key Features:**

-------------

- Cyclic doubly linked list

- Supports bidirectional iteration

- Implements const and non-const iterators

- Supports copy and move semantics

- Provides comparison operators

- Robust error handling (null iterator dereference, invalid decrement)

- STL-compatible interface

**Internal Structure:**

-------------------

**Each element is stored in a node (Node) that contains:**

- T value: The data value

- Node\* prev: Pointer to the previous node

- Node\* next: Pointer to the next node

**If the list is empty, `head` is nullptr. Otherwise, the list is cyclic, meaning:**

- head->prev points to the tail

- tail->next points to the head

**Public API:**

-----------

**1. Constructors / Destructor:**

- CyclicList()

- CyclicList(const CyclicList& other)

- CyclicList(CyclicList&& other) noexcept

- ~CyclicList()

**2. Assignment:**

- CyclicList& operator=(const CyclicList& other)

- CyclicList& operator=(CyclicList&& other) noexcept

**3. Insertion:**

- void push\_back(const T& val)

- void push\_back(T&& val)

**4. Capacity:**

- size\_t size() const noexcept

- bool empty() const noexcept

**5. Modifiers:**

- void clear() noexcept

**6. Iterators:**

- iterator begin()

- iterator end()

- const\_iterator begin() const

- const\_iterator end() const

- const\_iterator cbegin() const

- const\_iterator cend() const

**7. Comparison:**

- bool operator==(const CyclicList& other) const

- bool operator!=(const CyclicList& other) const

**Iterator Details:**

-----------------

- Implemented as a nested template class: Iterator<IsConst>

- Supports bidirectional iteration

- Stores pointer to node and current list

- Protects against invalid dereference or increment/decrement

- Steps counter tracks iteration progress to prevent infinite loops

**Error Handling:**

---------------

- Dereferencing or incrementing/decrementing null iterator throws std::runtime\_error

- Decrementing an iterator at the beginning throws an exception

**Compliance:**

-----------

- Full iterator system

- STL-style syntax

- Clean, readable code

- Follows modern C++20 best practices

**Use Case:**

---------

This container is suitable for applications where cyclic navigation is required, such as simulations, circular buffers, round-robin schedulers, and more.