

List Solutions

Double-linked List

- Which list types does the C++ standard library provide?
 - `std::forward_list` for single-linked list
 - `std::list` for double-linked list
- What is the main difference between these two types?
 - `std::forward_list` only has a link to the following element. It does not directly support iterating backwards
 - `std::list` has a link to the element before it. It supports iterating in either direction

Appending an Element

- Explain what happens when a new element is added at the end of an `std::list`
 - Allocate memory for the node
 - Set the last element's "next" pointer to the address of the new node
 - Set the new node's "previous" pointer to the address of the last node

Inserting an Element

- Explain what happens when a new element is added in the middle of an `std::list`
 - Allocate memory for the new node
 - Find the nodes immediately before and after where the node will be added
 - Set the "before" node's next pointer to the address of the new node
 - Set the "after" node's previous pointer to the address of the new node

Removing an Element

- Explain what happens when an existing element is removed from an `std::list`
 - An existing element is removed from the list
 - Find the nodes immediately before and after the node to be removed
 - Set the "before" node's next pointer to the address of the "after" node
 - Set the "after" node's previous pointer to the address of the "before" node
 - Release the memory allocated for the removed node

List Example

- Write a simple program which creates list instance, initializes it, and adds and removes some elements. After each operation, display the list elements

List Pros and Cons

- Give some advantages and disadvantages of using lists compared to vector
 - Adding and removing elements is faster than for vector
 - Lists are not indexed (no subscript notation)
 - Accessing elements and searching is slower than for vector
 - Lists use more memory per element than vector
 - Useful when we expect to add or remove elements frequently