

Insert Iterators Solutions

- How do insert iterators differ from the other iterators we have studied so far?
 - Insert iterators are used with new elements. The other iterators can only be used with existing elements
- Describe how to obtain an insert iterator for a container instance
 - We call an “inserter” function
 - The argument is the container object we wish to insert into
 - The return value is the insert iterator for that object

- Explain what calling `back_inserter()` does
 - It returns an insert iterator that can be used to insert elements at the back of the container
 - Every time we assign to the iterator, a new element is added to the container, using the container's `push_back` member function
- Write a simple program that uses `back_inserter()`

- Explain what calling `front_inserter()` does
 - It returns an insert iterator that can be used to insert elements at the front of the container
 - Every time we assign to the iterator, a new element is added to the container, using the container's `push_front` member function
- What are the limitations on the use of `front_inserter()` ?
 - Requires the container has a `push_front` member function
- Write a simple program that uses `front_inserter()`
 - Use `std::list` to overcome the limitations

- Explain what calling `inserter()` does
 - It returns an insert iterator that can be used to insert elements at a given point in the container
 - It takes a second argument, which is an iterator to the insertion point
 - Every time we assign to the iterator, a new element is added to the container, just before the insertion point
- Write a simple program that uses `inserter()`