- 삽입 반복자(Insert Iterator)

핵심 정리

1. 컨테이너 요소를 추가하는 방법.
   - 멤버 함수 (push_xxx / emplace_xxx / insert) 사용
   - 삽입 반복자 (insert iterator)

2. 삽입 반복자
   - <iterator> 헤더
   - 컨테이너에 요소를 삽입(추가)할 때 사용하는 반복자
   - 3가지 종류 (전방 삽입, 후방 삽입, 임의 삽입) 제공.

3. 기본 모양
   - back_insert_iterator< 컨테이너 클래스 이름 > p(컨테이너 객체);

4. copy 와 같은 STL 알고리즘을 사용해서 컨테이너에 요소를 추가할 수 있다.

핵심 정리

1. back_inserter() 함수
   - 후방 삽입 반복자(back_inserter_iterator)를 생성해 주는 도움 함수
   - 대부분 back_inserter_iterator 를 직접 사용하기 보다는 back_inserter()를 사용.
   - “클래스 템플릿은 반드시 템플릿 인자를 지정해야 하지만 함수 템플릿은 템플릿 인자를 생략할 수 있다.” - C++ 17 이전 까지.

- 삽입 반복자의 종류

핵심 정리

1. 삽입 반복자의 종류

클래스	생성 함수	내부 구현 원리
back_insert_iterator	back_inserter()	push_back()
front_insert_iterator	front_inserter()	push_front()
insert_iterator	inserter()	insert()

2. 주의 사항

- vector는 앞에 삽입할 수 없다. (push_front 멤버 함수가 없다.)
- 임의 삽입의 경우 생성자 인자가 2개 이다. (컨테이너와 삽입할 위치를 나타내는 반복자)

3. 전방 삽입과 임의 삽입

- 전방 삽입을 사용해서 컨테이너에 삽입하면 요소의 순서가 반대로 삽입 되지만
- 임의 삽입을 사용해서 컨테이너의 앞에 삽입하면 요소의 순서대로 삽입된다.

- 삽입 반복자 구현

핵심 정리

1. 모든 반복자는 * 로 요소에 접근 가능하고 ++ 연산자로 이동 가능해야 한다

2. 삽입 반복자의 연산

연산	연산의 결과
++	no-op
*	no-op
=	container->push_back, push_front, insert

구현 원리에 따라서 "p = 10" 으로도 삽입 가능한 경우도 있지만 표준 에서 정의한 "*p = 10" 으로 사용해야 한다.

3. Member Type

```
using iterator_category = output_iterator_tag;

using value_type = void;
using pointer = void;
using reference = void;
using difference_type = void;

using container_type = C;
```

- 삽입 반복자 구현

핵심 정리

1. 삽입 반복자의 종류

클래스	생성 함수	내부 구현 원리
back_insert_iterator	back_inserter()	push_back()
front_insert_iterator	front_inserter()	push_front()
insert_iterator	inserter()	insert()

2. 주의 사항

- vector는 앞에 삽입할 수 없다. (push_front 멤버 함수가 없다.)
- 임의 삽입의 경우 생성자 인자가 2개 이다. (컨테이너와 삽입할 위치를 나타내는 반복자)

3. 전방 삽입과 임의 삽입

- 전방 삽입을 사용해서 컨테이너에 삽입하면 요소의 순서가 반대로 삽입 되지만
- 임의 삽입을 사용해서 컨테이너의 앞에 삽입하면 요소의 순서대로 삽입된다.