- 반복자(iterator) 개념과 종류

## 핵심 정리

1. 반복자(iterator) in GoF's Design Pattern
   - Provide a way to access the elements of an aggregate object sequentially without exposing its underlying representation.
   - 복합 개체의 내부 구조에 상관 없이 순차적으로 요소에 접근하기 위한 방법을 제공하는 것.

2. 반복자(iterator) in STL
   - "반복자 처럼 동작하는 모든 것은 반복자 이다."
   - ++ 연산자로 이동 가능하고, * 연산자로 요소에 접근 가능한 것.

3. 반복자의 다양한 형태
   - Raw Pointer
   - 컨테이너의 요소를 열거 위한 객체 ( begin() )
   - 스트림 반복자 ( stream iterator )
   - 삽입 반복자 ( insert iterator )
   - directory_iterator, ···

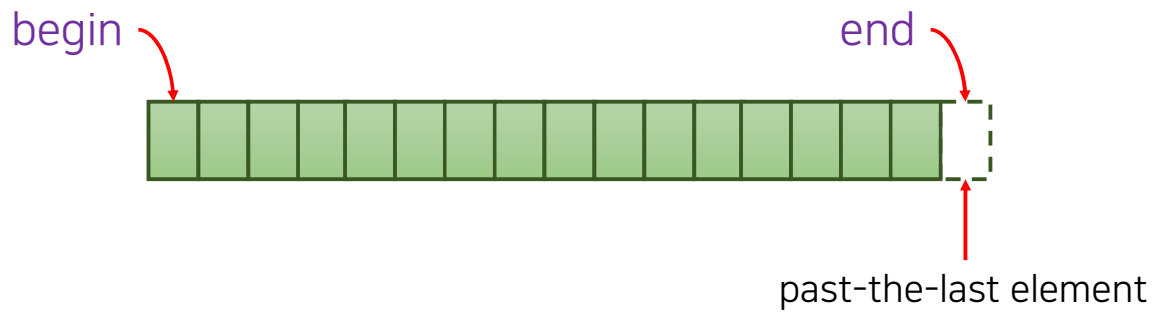
- 컨테이너에서 반복자 꺼내기

## 핵심 정리

1. 반복자 타입
   - 컨테이너이름<요소 타입>::iterator
   - auto 사용

2. 반복자를 얻는 방법
   - 멤버 함수 사용 : begin() / end()
   - 일반 함수 사용 : STL 컨테이너 뿐 아니라 배열도 사용 가능 – C++11
   - size(), empty(),

3. past-the-end iterator
   - end() 로 얻는 반복자는 마지막 다음 요소를 가리키므로 dereference(*) 하면 안된다.



- 반복자의 무효화

## 핵심 정리

1. 반복자 무효화 ( invalidate )
    - vector 등의 컨테이너의 내부 버퍼가 재할당 될 때
    - 반복자가 가리키던 요소가 제거(erase) 될 때
    - 컨테이너의 종류에 따라 무효화 되는 조건이 다르다.

- 반복자의 구간(range)

## 핵심 정리

1. 반복자의 구간(range)
   - [first, last ) : 시작과 마지막 다음 요소(past the end)을 가리키는 반복자의 쌍
   - 유효한(valid) 구간 : first 부터 시작해서 ++ 연산으로 last 에 도달 할 수 있는 구간.
   - 빈(empty) 구간 : first == last 인 경우, 빈 구간도 유효한 구간이다.

2. 대부분의 STL 의 알고리즘은 인자로 전달되는 구간이 유효한 구간이라는 가정하에 동작한다.

- copy 알고리즘

## 핵심 정리

1. copy 알고리즘
   - 하나의 구간의 내용을 다른 구간으로 복사하는 알고리즘
   - <algorithm>
   - 반복 문의 일반화된 표현
   - 1번째 구간은 완전한 구간(first, last)가 전달되지만 2번째 구간은 시작만 전달된다. 1번째 구간을 통해서 2번째 구간의 끝을 예측할수 있다.