STL 기초

STL Iterator

한컴에듀케이션

Iterator

포인터와 비슷하게 컨테이너에 저장된 원소들을 참조할 때 사용되는 **객체**

RandomAccess Iterator

- vector
- deque
- string
- array

[]
+
-
+=
-=

<
>
<=
>=

Bidirectional Iterator

- list
- set
- map

--
reverse_iterator

Forward Iterator

- forwad_list
- unordered_set
- unordered_map

*it=

Input Iterator

- iostream

=*it
->
++

==
!=

Iterator Operation

operation

- distance(v.begin(), it) : 두 iterator 사이 거리 반환
- advance(it, 3) : It를 다음 3번째 iterator로 이동
- new_it = next(it, 3) : new_it에 it 다음 3번째 iterator 저장
- new_it = prev(it, 3) : new_it에 it 이전 3번째 iterator 저장

random access iterator : O(1)

advance(it, 3) -> it += 3

next(it, 3) -> it + 3

otherwise : O(n)

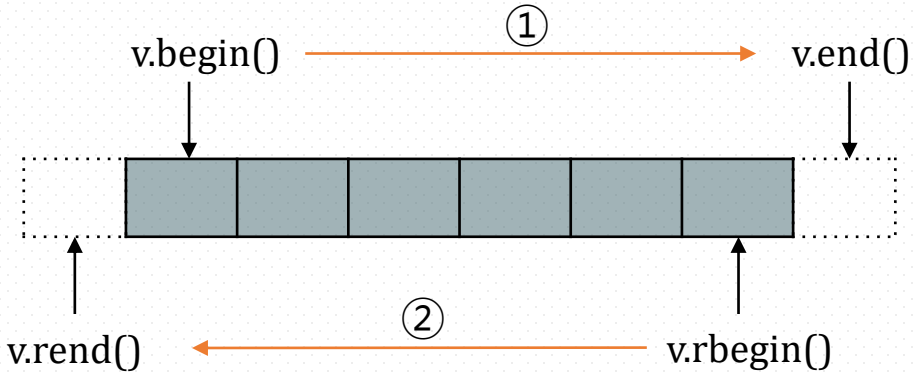
reverse_iterator

- 반대방향으로 진행하기 위한 iterator adaptor
- Bidirectional Iterator 부터 제공
- container의 begin(), end() 는 iterator type 반환
- container의 rbegin(), rend()는 reverse_iterator type 반환
- ++reverse_it 로 역방향 진행

ex) set<T>::iterator , list<T>::iterator

ex) set<T>::reverse_iterator

list<T>::reverse_iterator



Traverse

① 정방향 순회

- iterator

```
for (auto it = v.begin(); it != v.end(); ++it){
        /* process */
}
```

② 역방향 순회

- iterator

```
for (auto it = v.end(); it != v.begin();) {
     --it;
     /* process */
}
```

- reverse_iterator

```
for (auto it = v.rbegin(); it != v.rend(); ++it) {
     /* process */
}
```

① 정방향 순회 삭제

- iterator

```
for (auto it = v.begin(); it != v.end();){
     if (condition to erase) it = v.erase(it);
     else ++it;
}
```

② 역방향 순회 삭제

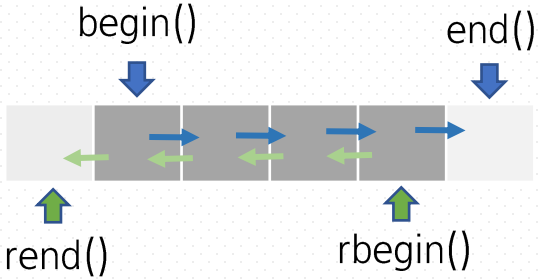
- iterator

```
for (auto it = v.end(); it != v.begin();) {
     --it;
     if (condition to erase) it = v.erase(it);
}
```

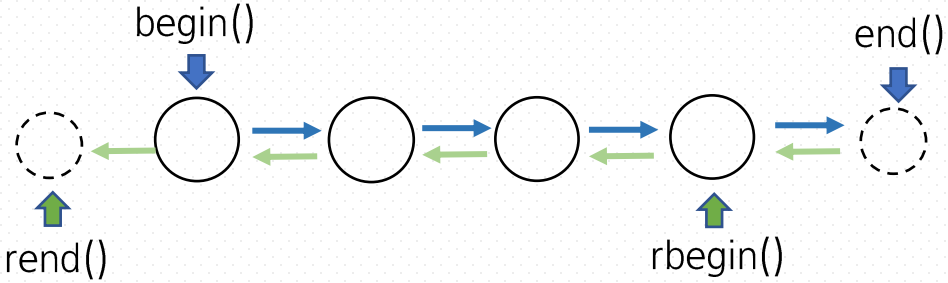
Container 별 Iterator

iterator
reverse_iterator

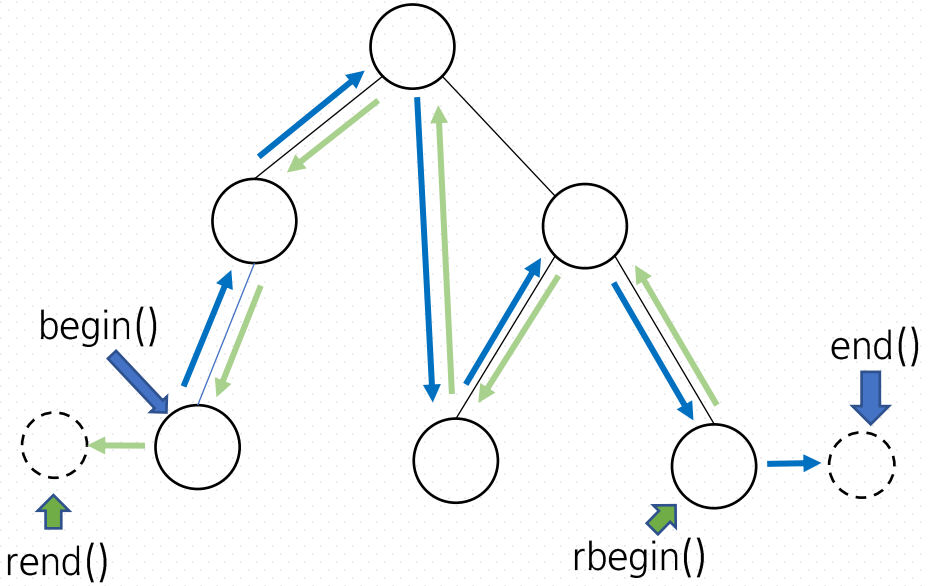
vector



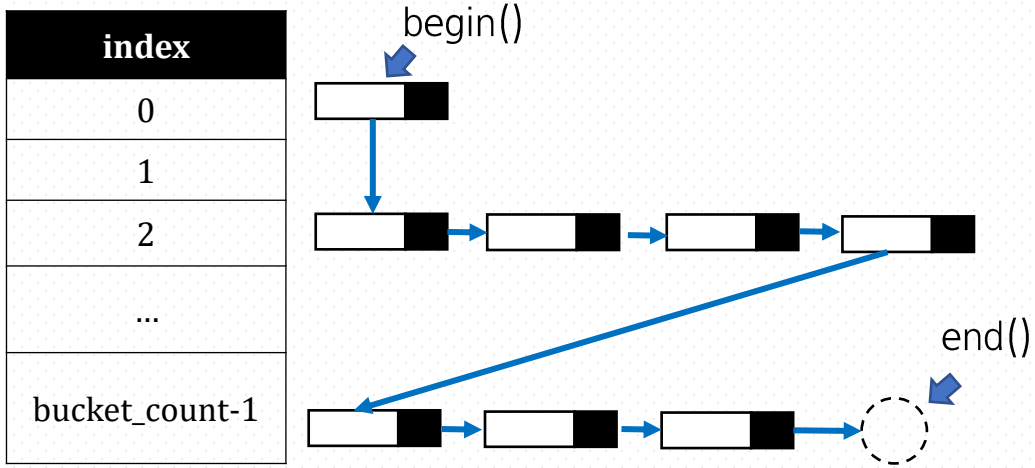
list



set/map

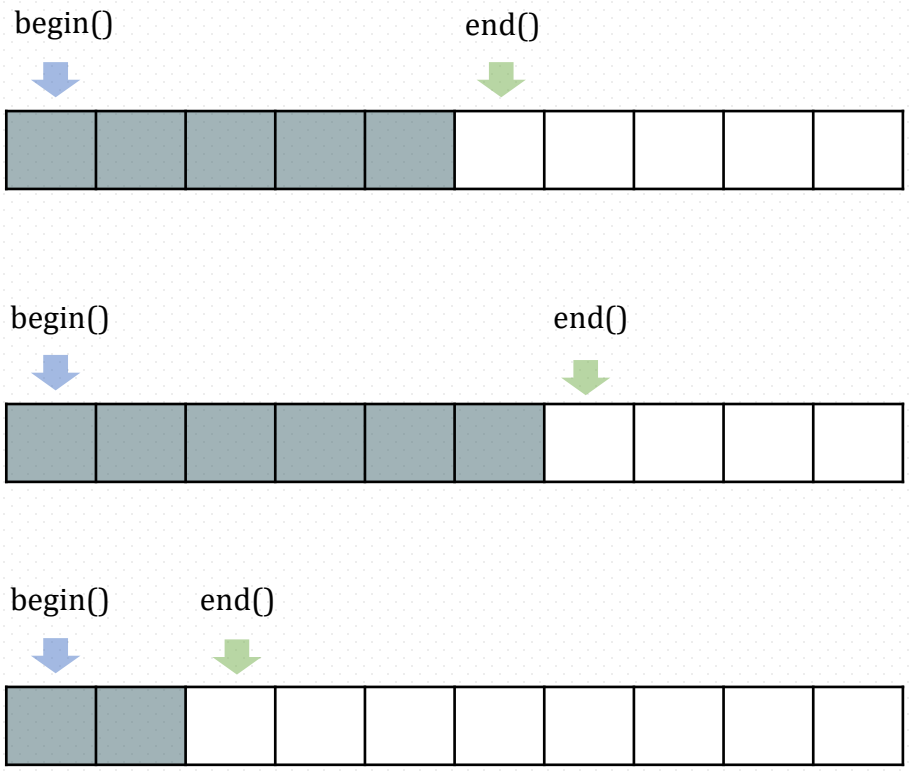


unordered_set/map

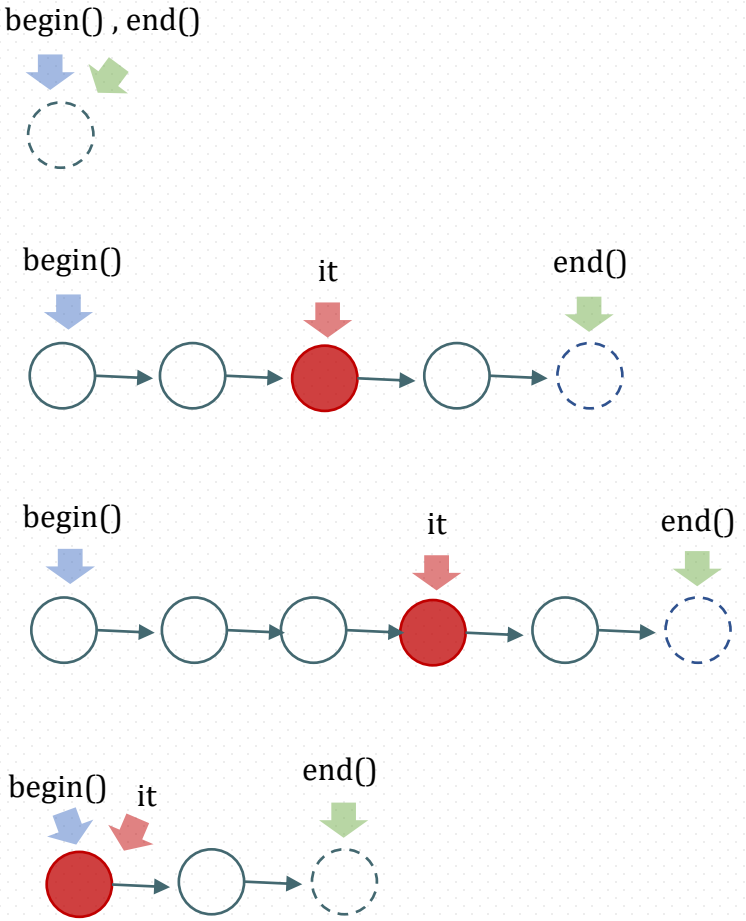


Iterator 성질

Continuous (vector, string,…)



Node (list, set, map, …)



Iterator 초기화

- pointer 등은 null 값(0) 으로 초기화 하면 되지만 iterator는 null값이 없다.
- 대신, end() 위치를 통해 표현 가능하다.
- container 멤버함수, algorithm 함수에서도 원하는 결과를 못 찾으면 end()가 반환 된다.
- 값이 지워지면 *it = s.end()* 로 설정

  it == s.end() 이면, 값이 없음을 판단

Iterator invalidation

Category	Container	After **insertion**, are...		After **erasure**, are...		Conditionally
		iterators valid?	**references** valid?	**iterators** valid?	**references** valid?	
Sequence containers	`array`	N/A		N/A		
	`vector`	No		N/A		Insertion changed capacity
		Yes		Yes		Before modified element(s)
		No		No		At or after modified element(s)
	`deque`	No	Yes	Yes, except erased element(s)		Modified first or last element
			No	No		Modified middle only
	`list`	Yes		Yes, except erased element(s)		
	`forward_list`	Yes		Yes, except erased element(s)		
Associative containers	`set` `multiset` `map` `multimap`	Yes		Yes, except erased element(s)		
Unordered associative containers	`unordered_set` `unordered_multiset` `unordered_map` `unordered_multimap`	No	Yes	N/A		Insertion caused rehash
		Yes		Yes, except erased element(s)		No rehash

list, set, map

- iterator가 가리키는 element가 지워지지 않는 한, 항상 유효하다.
- iterator를 기록하여 효율적인 활용이 가능하다.


감사합니다