

STL Containers

STL이란?

▶ Standard Template Library

- 프로그램에 필요한 자료구조와 알고리즘을 **template**으로 제공하는 라이브러리
- 반복자(iterator), 컨테이너(container), 알고리즘(algorithm)

▶ 컨테이너

- 데이터를 저장하고 관리하기 위한 클래스

▶ 반복자(iterator)

- STL 컨테이너에 저장된 원소들을 가리키는 포인터
- 컨테이너에 사용 되는 반복문

▶ 알고리즘

- STL에서 제공하는 함수

STL Containers

▶ 자주 쓰는 컨테이너

- **array** : 배열
 - = 적은 양의 자료에 유리
 - = **크기 변경 불가**
- **vector** : 가변 배열
 - = 적은 양의 자료에 유리
 - = **크기 변경이 가능**
- **list** : 양방향 연결 리스트
 - = 적은 양의 자료에 유리
 - = 크기 변경 가능
 - = **중간 삽입 삭제가 가능**
- **map** : 이진 탐색 트리(균형 이진 트리 / 레드-블랙 트리)
 - = **key와 value**를 가지며 따로 저장
 - = **많은 양의 자료에 유리**
 - = **적은 양에는 오버헤드로 손해**

STL Containers

▶ 컨테이너 사용법

- **array** : #include <array>

```
std::array<데이터타입, 크기> _array;
```

```
case1 : _array[index] = value;
```

```
case2 : _array.at(index) = value;
```

- **vector** : #include <vector>

```
std::vector<데이터타입> _vector;
```

```
_vector.push_back(value);
```

```
_vector.clear();
```

- **list** : #include <list>

```
std::list<데이터타입> _list;
```

```
_list.push_back(value);
```

```
_list.remove(value);
```

```
_list.clear();
```

STL Containers

▶ 컨테이너 사용법

```
- map : #include <map>
std::map<키 데이터타입, 데이터타입> _map;

case1 : _map.insert(std::pair<키 데이터타입, 데이터타입>(key, value));
case2 : _map[key] = value; // 키가 없다면 새로 만들어 값을 추가한다.

std::map<키 데이터타입, 데이터타입>::iterator iter = _map.find(key);
iter->first : key
iter->second : value

_map.erase(key);
_map.clear();

if (_map.end() == _map.find(key))
{
    // 해당 키 값을 가진 데이터가 없다.
}
```

STL Containers

▶ 범위 기반 for 문

```
for(const 데이터타입& element : _array)
{
    element
}
```

```
for(const 데이터타입& element : _list)
{
    element
}
```

```
for(const 데이터타입& element : _map)
{
    element.first
    element.second
}
```

STL Containers

▶ 반복자(iterator)

```
std::array<데이터타입>::iterator iter;  
std::vector<데이터타입>::iterator iter;  
std::list<데이터타입>::iterator iter;  
for (iter = 컨테이너.begin(); 컨테이너.end() != iter; iter++)  
{  
    iter->데이터;  
}
```

```
std::map<키 데이터타입, 데이터타입>::iterator iter;  
for (iter = _map.begin(); _map.end() != iter; iter++)  
{  
    iter->first;  
    iter->second;  
}
```