

C++ Programming

16th Study: Standard Template Library #2

- pair, tuple
- associative container (map, set)



C++ Korea 윤석준 (icysword77@gmail.com)

The background is a solid blue color. It features several white-outlined squares of various sizes scattered across the surface. Some squares are simple, while others are nested or arranged in a way that suggests a larger geometric pattern. The squares are located in the top-left, top-right, center, and bottom-right areas of the slide.

std::pair

2개 값의 묶음

std::pair

- 2개 값의 묶음
- 함수의 리턴 값이 2개가 필요할 경우 사용

STL 내부적으로 많이 사용

3개 이상인 경우는 std::tuple를 사용

```
#include <iostream>
std::pair<타입1, 타입2> 변수명
변수명.first
변수명.second
```

std::pair 예제

```
#include <iostream>
```

```
std::pair<int, double> P(10, 15.5);
```

```
P = std::make_pair<int, double>(10, 15.5);
```

```
std::cout << P.first << '\t' << P.second << std::endl;
```



std::tuple

값들의 집합

std::tuple

- 여러 개의 값의 묶음

```
#include <tuple>
```

```
std::tuple<타입1, 타입2, ... > 변수명
```

```
std::get<값의 index>(변수명)
```

std::tuple 예제

```
#include <tuple>
```

```
std::tuple<int, double, bool> T(10, 15.5, true);
```

```
T = std::make_tuple<int, double, bool>(10, 15.5, true);
```

```
std::cout << std::get<0>(T) << '\t'  
          << std::get<1>(T) << '\t'  
          << std::get<2>(T) << std::endl;
```

Associative container

연관 저장공간

Associative container

- Key 와 Value 의 짝(pair)로 이루어진 자료구조
사실은 그렇지 않은 것도 있는데, 잘 안 쓴다.
- 저장된 자료의 순서와 상관없이 특정 기준에 맞게 저장
주로 Key 값의 정렬 순서대로 저장

Associative container의 구분

std::저장구조 _ 중복여부, Value 여부

- map : Key 와 Value 의 짝(pair)로 이루어짐
set : Key 만으로 구성
- unordered : 정렬되지 않은 상태로 저장, 빠른 검색에 좋음
default : tree 구조로 저장, 정렬되어 있음
- multi : 중복된 Key 값으로 데이터 저장이 가능
default : 중복된 Key값은 입력이 불가능

Associative container의 구분 예시

- Key 와 Value 의 짝(pair)로 이루어졌으면서, 빠른 검색이 필요하고, 중복된 Key로 여러 개의 Value가 저장되어야 할 경우에는 ?

➡ `std::unordered_multimap`

- Key만으로 이루어져 있으며, 정렬되어 있어야 하며, 중복된 값은 저장되면 안되는 경우에는 ?

➡ `std::set`

The background is a solid blue color. It is decorated with several dotted white squares and lines of varying sizes and orientations, scattered across the slide. Some are simple squares, while others are more complex, resembling a maze or a series of connected lines.

map

std::map

std::map

- STL 연관 컨테이너에서 가장 많이 사용됨
- Key + Value 의 pair로 이루어짐
- 정렬되어 있어야 하며
- 많은 자료를 저장하고 검색이 빨라야 하며
- 자주 삽입, 삭제가 일어나지 않는 경우에 효율적

std::map 예제

```
#include <map>
```

```
std::map<int, int> M;
```

```
M.insert(std::make_pair<int, int>(5, 100));
```

```
M.insert(std::pair<int, int>(7, 14));
```

```
M.insert({ 3, 50 }); // using initializer_list
```

```
int d = M[7]; // 14
```

```
for (auto it = M.begin(); it != M.end(); it++)  
    std::cout << "(" << it->first  
                << ", " << it->second << ")" << std::endl;
```

std::map의 다른 종류

- `std::multimap` : 같은 Key 로 여러 개의 Value 입력 가능
- `std::unordered_map` : 정렬되지 않음, 대신 검색 속도가 더 빠름
- `std::unordered_multimap` : 같은 Key로 여러 개의 Value 입력이 가능
정렬되지 않은 채로 저장

The background is a solid blue color. Scattered across it are several squares of various sizes, each formed by a dotted white border. These squares are positioned in the corners and along the edges, creating a minimalist geometric pattern.

set

std::set

std::set

- Key 만으로 이루어진 이진트리 자료구조
- 정렬되어 있어야 하며
- 특정 값이 있는지 없는지 알아야 할 때
- 많은 자료를 저장하고, 검색 속도가 빨라야 할 때

std::set 예제

```
#include <set>
```

```
std::set<int> S{ 9, 7, 5, 1, 3 }; // 1, 3, 5, 7, 9
```

```
S.insert(4); // 1, 3, 4, 5, 7, 9
```

```
for (auto it = S.begin(); it != S.end(); it++)  
    std::cout << (*it) << '\t';  
std::cout << std::endl;
```

```
auto f = S.find(5);  
if (f != S.end())  
    std::cout << "found !" << std::endl;
```

std::set의 다른 종류

- `std::multiset` : 같은 Key가 중복으로 입력 가능
- `std::unordered_set` : 정렬되지 않음, 대신 검색 속도가 더 빠름
- `std::unordered_multiset` : 같은 Key가 중복으로 입력 가능
정렬되지 않은 채로 저장