- std::string

## 핵심정리

1. std::string
   - <string> 헤더
   - <string.h> 또는 <cstring> ➔ C 문자열 함수를 위한 헤더.
   - strcpy(), strcmp() 등을 사용할 필요 없이, 일반 변수와 유사한 방식으로 코드 작성.

2. C문자열로의 변환
   - c_str() : const char* 로의 변환,반환된 문자열은 널 문자로 끝나는 문자열
   - data() : C++11 이전까지는 반환된 문자열이 널 문자로 끝나는 것을 보장하지 않음.
     C++11 이후에는 널 문자로 끝나는 것을 보장.

3. 정수/실수로 변환
   - 정수 변환 : stoi, stol, stoll, stoul, stoull
   - 실수 변환 : stof, stod, stold
   - 정수/실수 => string : to_string(), to_wstring()

4. 다양한 멤버 함수들
   - www.cppreference.com 참고
   - 본 과정의 후반부에서 다시 다루게 됨.

- literals

## user define literals

- literals 뒤에 접미사를 붙여서 특정 타입의 객체를 만드는 문법 – C++11에서 처음 소개됨.

| literals | 생성되는 객체 |
|---|---|
| “hello”s | string |
| 3s | seconds |
| 3i | complex<double> |

- `using namespace std::string_literals;`
- `using namespace std::literals;`

- std::complex

## 핵심 정리

1. 복소수를 나타내는 클래스.
2. <complex> 헤더
3. real(), imag()
4. 복소수 관련 다양한 수학 함수를 제공 – non member function
   - `Exponential  : exp, log, log10`
   - `Power        : pow, sqrt`
   - `Trigonometric : sin, cos, tan, asin, acos, atan`
   - `Hyperbolic   : sinh, cosh, tanh, asinh, acosh, atanh`
   - `cppreference.com 예제 참고`

5. `complex<double>, complex<float>, complex<long double>` 형태로 사용
   - double, float, long double 이외의 타입에 대해서는 unspecified

6. Literals

| literals | 생성되는 객체 |
|---|---|
| 3.2i | complex<double> |
| 3.2if | complex<float> |
| 3.2il | complex<long double> |

- std::bitset

## 핵심 정리

1. bit 를 관리하는 클래스
2. <bitset> 헤더
3. cout, cin 으로 입출력 가능. ( <<, >> )
4. 변환 함수
   - `to_string() / to_ulong() / to_ullong()`

5. 각 비트 접근 함수
   - `set, reset, [] 연산, flip()`

6. 조사 함수
   - `test, all, none, any, count`

7. 비트 연산 가능
   - `&, ¦, ^`

- std::pair

## 핵심 정리

1. 서로 다른 타입의 객체를 2개를 보관하는 타입
2. <utility> 헤더
3. first, second 멤버를 통해서 요소에 접근
4. 함수가 2개의 값을 리턴 하고 싶을 때 pair 를 리턴 하는 경우가 있다.
    - set::insert 멤버 함수

5. tuple
    - pair를 보다 일반화한 타입
    - 서로 다른 타입의 객체를 N개 보관하는 타입.
    - 과정 후반부에서…

6. make_pair
    - pair를 만드는 helper 함수

- std::pair

## 핵심 정리

1. 서로 다른 타입의 객체를 2개를 보관하는 타입

2. make_pair
   - pair를 만드는 helper 함수
   - 클래스 템플릿은 템플릿 인자를 생략할 수 없기 때문에 항상 복잡하다.
   - 함수 템플릿은 템플릿 인자를 생략할 수 있기 때문에 사용하기 쉽다.
   - C++17 부터는 클래스 템플릿도 인자를 생략 할 수 있다.

3. 클래스 템플릿을 만드는 함수 템플릿
   - make_pair
   - make_tuple
   - 삽입 반복자를 만드는 함수들.

- concept

## 핵심 정리

1. std::min
2. 전달된 인자 중에서 작은 값을 리턴 하는 함수
3. <algorithm>

4. concept
   - 특정 함수(클래스)를 사용하기 위해 타입이 가져야 하는 조건.
   - min 함수에 전달되는 인자는 LessThanComparable 을 만족해야 한다.
   - 코드를 사용해서 concept 를 정의 할 수 있다 – C++20 부터

5. STL은 템플릿 이지만 "모든 타입" 에 적용되는 것이 아니라 "조건을 만족" 하는 타입에 대해서 동작한다.