IT - 신윤호

정리노트#7 (11/20)

① 템플릿 : 함수나 클래스 코드를 찍어내듯이 생산할 수 있도록 일반화 시키는 도구
* 선언 : template < class T>
template < class T1, class T2, class T3 > * 여러개 자료형 사용가능

② 구체화 : 사용에 따라 T의 형이 바뀌는 과정

③ 제네릭 함수와 일반 함수를 중복 선언할 경우 우선 순위는 일반함수의 우선순위가 더 높아 일반함수를 바인딩 한다.
* 제네릭 함수에도 디폴트 매개변수를 선언할 수 있다.

④ Generic 클래스 생성 가능

⑤ STL : 템플릿으로 작성된 많은 제네릭 클래스와 함수라이브러리

⑥ container : 템플릿 클래스, 데이터를 저장하고 검색하기 위해 많이쓰는 자료구조를 구현하는 클래스
↳ 종류: 리스트, 큐, 스택, 벡터, 디큐, 셋, 맵

⑦ iterator : 반복자, 컨테이너 원소에 대한 포인터, 원소들을 하나씩 순회 접근하기 위해서

⑧ auto : 컴파일러가 변수타입을 추론해서 붙여줌.

예제 10-8 (분석)

```
#include <iostream>
using namespace std;

template <class T1, class T2> // 두 개의 제네릭 타입 선언
class GClass {
   T1 data1;
   T2 data2;
public:
   GClass();
   void set(T1 a, T2 b);
   void get(T1 &a, T2 &b);
};

template <class T1, class T2>
GClass<T1, T2>::GClass() {
   data1 = 0; data2 = 0;
}

template <class T1, class T2>
void GClass<T1, T2>::set(T1 a, T2 b) {
   data1 = a; data2 = b;
}

template <class T1, class T2>
void GClass<T1, T2>::get(T1 & a, T2 & b) {
   a = data1; b = data2;
}
```

클래스 템플릿 선언

멤버변수

data1을 a에, data2를 b에 리턴하는 함수

생성자

```
int main() {
   int a;
   double b;
   GClass<int, double> x;
   x.set(2, 0.5);
   x.get(a, b);
   cout << "a=" << a << '₩t' << "b=" << b << endl;

   char c;
   float d;
   GClass<char, float> y;
   y.set('m', 12.5);
   y.get(c, d);
   cout << "c=" << c << '₩t' << "d=" << d << endl;
}
```