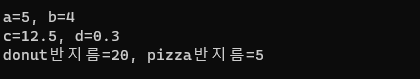
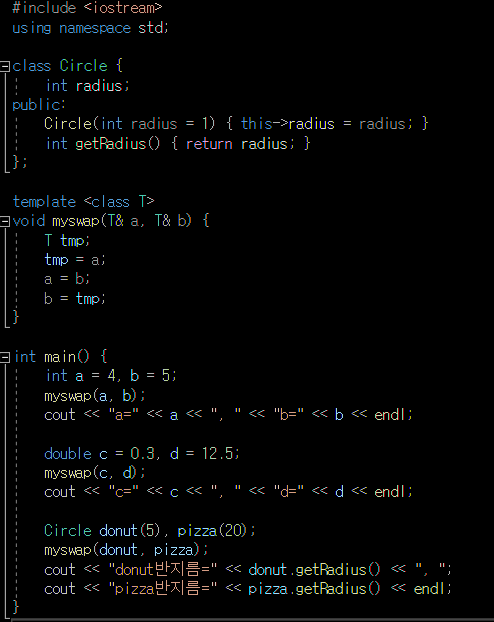
템플릿과 표준 템플릿

202004103 정원석, 202004904 이종하

템플릿은 매개 변수만 다르고 나머지 코드는 동일한 중복 코드로 사용하는 것이 재사용이 힘들고 나중에 오류도 발생시킬 수도 있어 이것을 해결하기 위해 만들었다.  
템플릿은 함수나 클래스를 일반화하는 C++ 도구로  
template <class T> or template <typename T>으로 선언한다.   
template <class T1, class T2, class T3>로 3개를 동시에 선언도 가능하다.

<예시>  
template <class T>  
void myswap (T & a, T & b) {

T tmp;  
 tmp = a;  
 a = b;  
 b = tmp;  
}  
타입을 지정하지 않더라도 받아오는 타입을 대입하여 구체화하고 컴파일 후 실행된다.

예제 10-1)  


3가지 타입을 템플릿을 이용하여 하나의 함수로 실행이 가능하다.

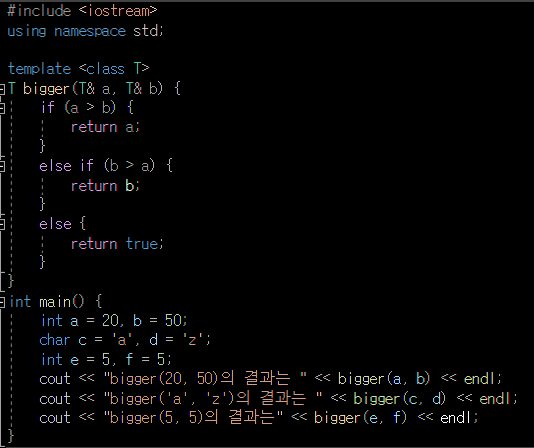
<주의점>



템플릿을 사용할 때 타입을 동일하게 설정하였으면 사용할 때에도 두 변수를 동일한 타입으로 선언하지 않으면 오류가 발생한다.

<장단점>  
장점은 함수 코드의 재사용성이 높다. → 높은 생산성과 유용성을 가짐  
단점은 포팅에 취약하다. → 컴파일러에 따라 지원하지 않을 수 있음  
 컴파일 오류 메시지가 빈약하고 디버깅에 어려움이 있다.

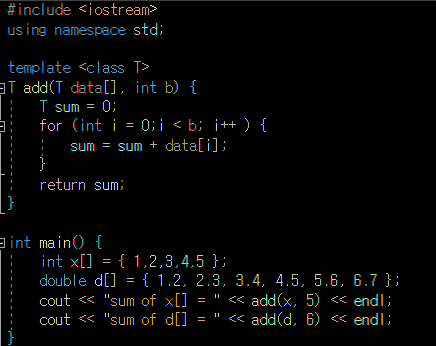
예제 10-2)





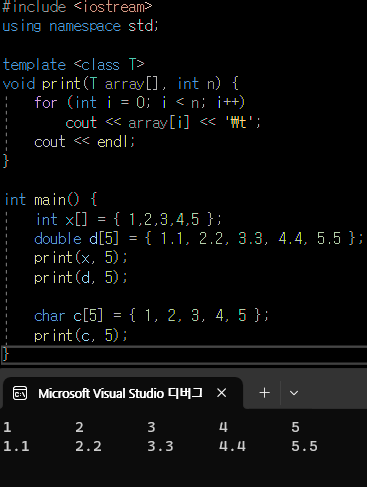
값이 같은 경우도 추가해 작성해보았다. 그런데 char경우 값이 나오지 않는다.

예제 10–3)



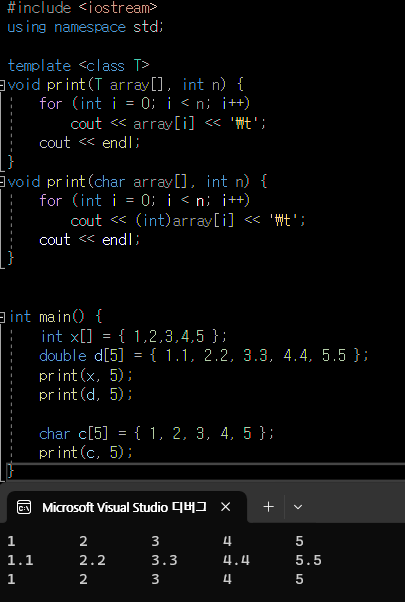


<템플릿 함수의 문제점>



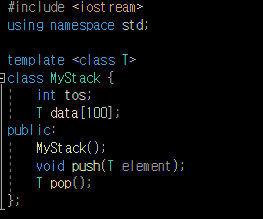
12345를 char로 선언해 정수 12345에 해당하는 그래픽 문자가 출력된다.

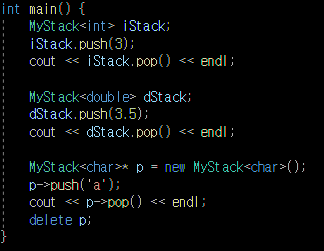
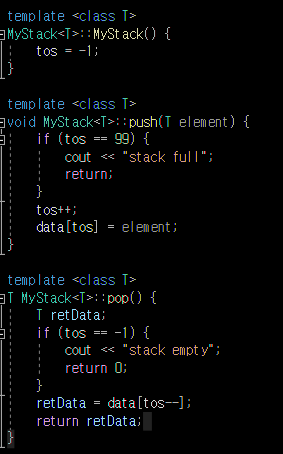
문제점 해결 예제 10-5)

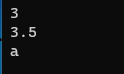


오류가 발생하는 부분을 중복함수로 선언해 문제가 생기는 부분을 해결한다.

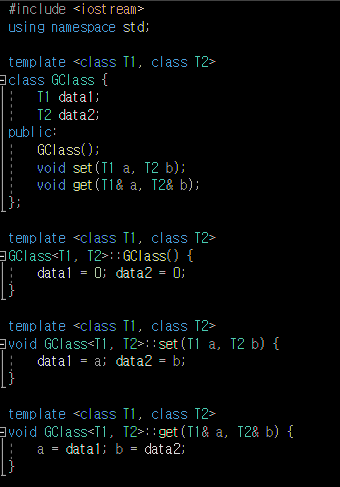
10-6 예제)

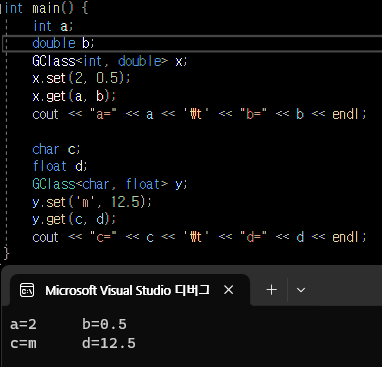






메인에서 제너릭 클래스를 선언할 때 타입을 지정해야만 하고 제너릭 클래스도 동적 생성을 할 수 있다.

예제 10-8)  




입력할 값의 타입이 다른 경우 2가지 템플릿을 사용하여 구현하고 선언할 때도 두 개로 나눠 선언하여야 한다.

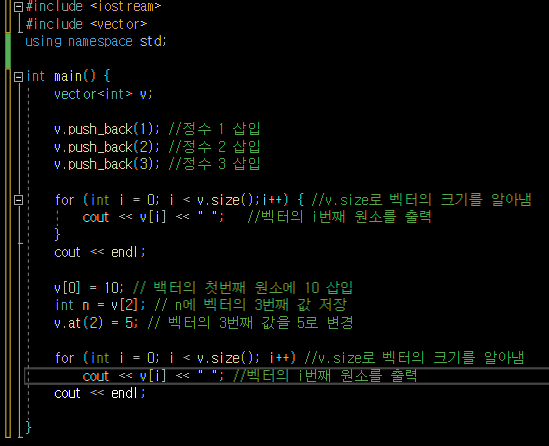
<STL>  
STL은 표준 템플릿 라이브러리이다.  
많은 제너릭 클래스와 제너릭 함수가 포함되어 이들을 이용하여 쉽게 응용 프로그램을 작성할 수 있다.  
구성)  
컨테이너는 템플릿 클래스로 데이터를 담아두는 자료 구조를 표현한 클래스이다.  
(리스트, 큐, 스택, 맵, 셋, 벡터)  
iterator는 컨테이너 원소에 대한 포인터로 컨테이너 원소들을 순회하면서 접근하기 위해 만들어진 원소에 대한 포인터이다.  
알고리즘은 템플릿 함수로 컨테이너 원소에 대한 복사, 검색, 삭제, 정렬 등의 기능을 구현한 템플릿 함수이지만 컨테이너의 멤버 함수는 아니다.  




알고리즘들은 대부분 구현되어 있어 만들고자 하는 것을 만들 수 있다.

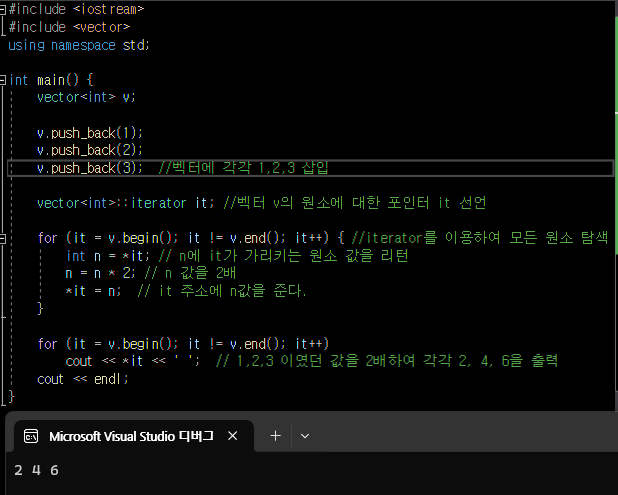
<헤더파일>  
컨테이너 클래스를 사용하기 위한 헤더파일  
벡터를 사용하기 위한 #include<vector>, 리스트를 사용하기위한 #include <list>  
알고리즘 함수를 사용하기 위한 헤더파일  
#include <algorithm>  
  
<vector 컨테이너>  
가변 길이 배열을 구현한 제너릭 클래스로 벡터의 길이에 대한 고민할 필요가 없음  
원소의 저장, 삭제, 검색 등 다양한 멤버 함수를 지원한다.  
벡터에 저장된 원소는 인덱스로 접근 가능하고 인덱스는 0부터 시작한다.  
(주요 멤버와 연산자)  
예제 10-9)





<iterator>  
반복자라고도 부르고 컨테이너의 원소를 가리키는 포인터이다.  
vector<int>::iterator it;  
it = v.begin(); 와 같이 구체적인 컨테이너를 지정하여 반복자 변수를 선언한다.

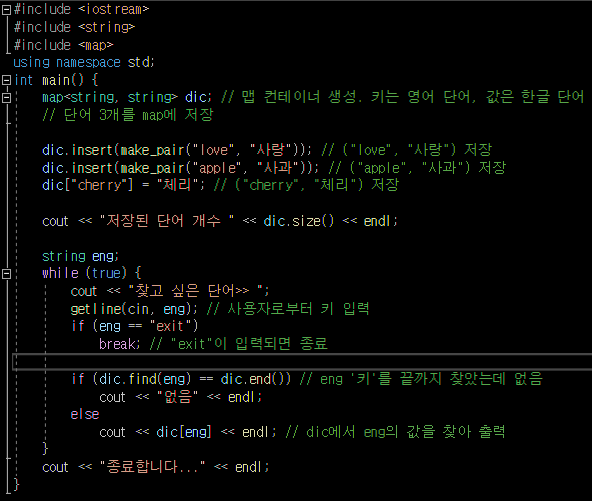
예제 10-11)

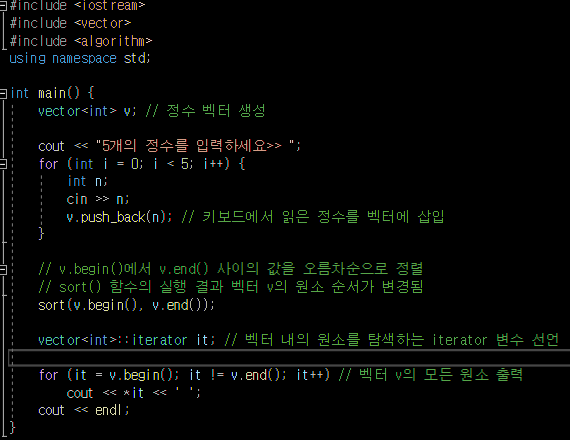


<map 컨테이너>  
(‘키’,’값’)의 쌍을 원소로 저장하는 제너릭 컨테이너이다.  
#include <map>을 필요로 한다.  




예제 10-12)



<STL 알고리즘>  
템플릿 함수, 전역 함수, iterartor와 함께 작동한다.  
예제 10-13)  


<auto를 이용한 변수 선언>  
컴파일러에게 변수선언문에서 추론하여 타입을 자동 선언하도록 지시한다.  
장점으로 복잡한 변수 선언을 간소하게, 긴 타입 선언 시 오타를 줄인다.  
(사용 사례)  
auto pi = 3.14; // 3.14가 실수이므로 pi는 double 타입으로 선언됨   
auto n = 3; // 3이 정수이므로 n을 int 타입으로   
auto \*p = &n; // 변수 p는 int\* 타입으로 추론

int n = 10;   
int & ref = n; // ref는 int에 대한 참조 변수   
auto ref2 = ref; // ref2는 int& 변수로 자동 선언