

# cpp\_11\_템플릿과 STL

---

## ☐ 개념 확인 학습

---

1. 다음 질문에 O, X로 답하세요.

- A. 템플릿을 선언하기 위해 사용하는 키워드는 `template` 이다. (   )
- B. 제네릭 타입을 선언하는 키워드는 `generic` 이다. (   )
- C. 템플릿 함수와 동일한 이름의 함수가 중복되어 있을 경우 템플릿 함수가 우선적으로 바인딩된다. (   )
- D. C++ 표준STL 라이브러리가 작성된 이름공간은 `std` 이다. (   )
- E. 템플릿 함수나 템플릿 클래스를 활용하는 프로그래밍 기법을 일반화(generic) 프로그래밍이라고 한다. (   )
- F. 템플릿 함수는 오버로딩을 할 수 없다. (   )
- G. 두 값을 그룹으로 묶는 클래스는 `pair` 이다. (   )
- H. `map` 컨테이너 요소들은 키값을 기준으로 내림차순으로 정렬된다. (   )
- I. `set` 컨테이너 요소들은 '키'로 '값'을 검색 한다. (   )
- J. 벡터 `v`의 각 원소에 대해 `print()`를 호출 하려면 `for_each(v.begin(), v.end(), print);`를 사용한다. (   )
- K. 람다식에서 리턴타입은 생략할 수 있다. (   )
- L. 람다식에서 함수 바깥의 변수 목록을 사용하려면 캡처리스트(`[ ]`)로 작성하면 된다. (   )

2. C++ 다음 STL의 각 기능을 사용하기 위해 필요한 헤더 파일을 제시하세요.

- A. `vector` 클래스
- B. `merge()` 함수
- C. `map` 클래스

3. 다음에서 템플릿 선언을 잘못된 것은?

- A. `template <class T>`
- B. `template (class T)`
- C. `template <typename T>`

D. template <typename T1, typename T2>

4. 다음 제네릭 함수 선언에서 잘못된 부분을 바르게 수정하세요.

```
template <typename T>
int max(T x, T, y) {
    if(x>y) return x;
    else return y;
}
```

5. main() 함수가 실행 될 수 있도록 다음 두 get() 함수를 일반화한 하나의 제네릭 get() 함수로 작성하세요.

```
int get(int a[], int size, int index) {
    if(index >=0 && index < size) return a[index];
    else return 0;
}
```

```
char get(char *a, int size, int index) {
    if(index >=0 && index < size) return *(a+index);
    else return 0;
}
```

```
int main(){
    const char carr [] = "daniel";
    int iarr [] = {1,2,3};

    cout << get(carr, sizeof(carr)/sizeof(char), 4) << endl;
    cout << get(iarr, sizeof(iarr)/sizeof(int), 4) << endl;
    return 0;
}
```

6. C++ STL 구성에 대하여 설명하시오

- 컨테이너  
: 템플릿 클래스(데이터를 담아두는 자료 구조를 표현한 클래스)  
: 리스트, 큐, 스택, 맵, 셋, 벡터
- iterator  
: 컨테이너의 원소들을 순회하면서 접근하기 위해 만들어진 컨테이너 원소에 대한 포인터
- 알고리즘  
: 컨테이너 원소에 대한 복사, 검색, 삭제, 정렬 등의 기능을 구현한 템플릿 함수  
: 컨테이너의 멤버 함수 아님

7. pair & tuple 템플릿 클래스에 대하여 설명하세요.

- pair 템플릿 클래스

: <utility> 헤더 파일에 정의  
: 두 값을 그룹으로 묶는 클래스  
: pair에 담긴 값은 first, second public 데이터 멤버로 접근

- tuple 템플릿 클래스  
: <tuple>헤더 파일에 정의  
: 여러 개의 하나로 묶어서 저장  
: 각각의 타입도 따로 지정

8. `vector<double> v;` 일 때, 제시된 문제를 해결하는 문장을 제시하세요.

- A. 생성된 벡터 `v` 마지막에 3.1 삽입
- B. 벡터 `v`에 저장된 원소 개수
- C. 벡터 `v`에 저장된 첫 번째 원소 삭제

9. `vector<char> v;` 일 때, 벡터 `v`에 저장된 모든 원소를 출력하고자 합니다. 제시된 방법을 사용하여 출력하는 문장을 작성하세요.

- A. 반복자 사용

```
vector<char>::iterator it;  
for (it = v.begin(); it != v.end(); it++)  
    cout << *it << endl;
```

- B. 범위기반 for 사용

```
for (auto &i : v)  
    cout << i << " ";
```

10. `map<string, int> scores;` 일 때, 제시된 문제를 해결하는 문장을 제시하세요.

- A. `scores` 객체에 임의의 원소를 추가하는 방법을 세 가지 이상 제시하세요.

```
scores.insert( {"Mary", 88} );  
scores["Robert"] = 77;  
scores.try_emplace("Dog", 90);  
scores.emplace("Bird", 99);  
scores.insert( make_pair("John", 52) );
```

- B. `scores` 객체 저장된 모든 원소를 출력하는 방법을 두 가지 이상 제시하세요.

```
//1  
for (iter = scores.begin(); iter != scores.end(); iter++){  
    cout << setw(10) << left << iter->first << " " ;  
    cout << setw(4) << iter->second << endl;  
}
```

```
//2
for (const auto &[key, value] : scores){ //C++ 17 구조적 바인딩 & 범위 기반 for 사용
    cout << setw(10) << key << " " << setw(4) << value << endl;
}

//3
for (const auto &value : scores){ //범위 기반 for
    cout << setw(10) << value.first << " " << setw(4) << value.second << endl;
}
```

11. 람다식 구성에 대하여 설명하세요.

캡처 리스트
매개변수 리스트
생략 가능
함수 바디  
`[ ] ( ) -> 리턴타입 { /* 함수 코드 작성 */ };`  
 (a) 람다식의 기본 구조

캡처 리스트 : 람다식에서 사용하고자 하는 함수 바깥의 변수 목록  
 매개변수 리스트 : 보통 함수의 매개변수 리스트와 동일  
 리턴 타입  
 함수 바디 : 람다식의 함수 코드

12. 다음 소스의 빈칸을 채워 람다식을 완성하세요. 빈칸은 `auto` 변수인 `method`에 매개변수로 배열의 크기(`size`)와 `int` 타입의 배열(`arr`)을 받아 오름차순으로 정렬하는 람다식입니다.

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

int main(){
    _____ { //람다식 완성
        sort(arr, arr+size);
    };

    int iarr[] = {4, 7, 2, 67, 4, 13, 6};
    method(sizeof(iarr)/sizeof(int), iarr);

    for(auto value : iarr){
        cout<<value<<" ";
    }
    cout<<endl;
    return 0;
}
```

```
PS C:\yanges\lecture\lecture_src\cpp> g++ cpptest.cpp
PS C:\yanges\lecture\lecture_src\cpp> ./a
2 4 4 6 7 13 67
```

---

☐ 응용 프로그래밍

---

13. main() 함수와 실행결과를 참고하여 배열의 순서를 역순으로 할 수 있는 템플릿 함수 **reverse()**와 **swap()** 함수를 작성합니다. 데이터를 교환하는 템플릿 함수, 배열의 요소를 출력하는 템플릿 함수도 함께 만들어서 활용합니다.

//reverse() : 전달 받은 배열의 양 끝 데이터를 순차적(첫 번째와 마지막, 두 번째와 마지막에서 두 번째...)으로 swap()의 매개변수로 전달

//swap() :매개변수로 전달된 두 데이터를 교환

```
template <typename T>
void print(string title, T arr[], int size) {
    cout << title << " : " ;
    for (int i = 0; i < size; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;
}
```

```
int main() {
    int arr1[] = { 3, 7, 2, 12, 14, 1 };
    double arr2[] = { 22.7, 14.2, 3.8, 12.23, 11.2 };
    char arr3[] = { 'C', 'a', 'B', 'E', 'N', 'Q' };
    string arr4[] = { "John", "Lu", "Mary", "Su" };
```

```
    print("Original array", arr1, sizeof(arr1)/sizeof(int));
    reverse(arr1, sizeof(arr1)/sizeof(int));
    print("Reversed array", arr1, sizeof(arr1)/sizeof(int));
    cout << endl;
```

```
    print("Original array", arr2, sizeof(arr2)/sizeof(double));
    reverse(arr2, sizeof(arr2)/sizeof(double));
    print("Reversed array", arr2, sizeof(arr2)/sizeof(double));
    cout << endl;
```

```
    print("Original array", arr3, sizeof(arr3)/sizeof(char));
    reverse(arr3, sizeof(arr3)/sizeof(char));
    print("Reversed array", arr3, sizeof(arr3)/sizeof(char));
    cout << endl;
```

```
    print("Original array", arr4, sizeof(arr4)/sizeof(string));
    reverse(arr4, sizeof(arr4)/sizeof(string));
    print("Reversed array", arr4, sizeof(arr4)/sizeof(string));
    cout << endl;
```

```
    return 0;
}
```

```
Original array : 3 7 2 12 14 1
Reversed array : 1 14 12 2 7 3
```

```
Original array : 22.7 14.2 3.8 12.23 11.2
Reversed array : 11.2 12.23 3.8 14.2 22.7
```

```
Original array : C a B E N Q
Reversed array : Q N E B a C
```

```
Original array : John Lu Mary Su
Reversed array : Su Mary Lu John
```

14. main() 함수와 실행결과를 참고하여 벡터를 사용하여 **Sudent** 정보를 저장하고 출력하는 프로그램을 완성 하세요.

```

class Student{
    string name;
    int id;
public:
    Student() = default;
    Student(string name, int id) : name(name), id(id) {};
    ~Student() = default;
    void show() { cout << "name: " << name << ", id: " << id << endl; }
};

class Manager{
    vector<Student> vec;
public:
    void run();
    void input2save();
    void print(); //show()함수를 이용하여 vec에 저장된 모든 Student 객체 정보 출력
};

void Manager::run() {
    cout << ">> 벡터에 학생 데이터를 저장합니다." << endl;
    input2save();
    cout << ">> 벡터에 저장된 모든 학생 데이터를 출력합니다." << endl;
    print();
}

void Manager::input2save() {
    //Student 객체와 관련된 정보를 입력 받아 vec에 저장
}

void Manager::print() {
    //show()함수를 이용하여 vec에 저장된 모든 Student 객체 정보 출력
}

int main(){
    Manager man;
    man.run();
}

```

```

>> 벡터에 학생 데이터를 저장합니다.
학생 데이터를 입력하세요.(입력 종료는 quit)
name : daniel
id : 11
name : benny
id : 22
name : quit
2 명의 학생이 저장되었습니다.

>> 벡터에 저장된 모든 학생 데이터를 출력합니다.
name: daniel, id: 11
name: benny, id: 22

```

15. 국가의 수도 맞추기 게임을 **vector**를 사용하여 작성합니다.

- **Nation** 클래스는 국가와 수도를 문자열 멤버변수로 저장합니다.
  - **vector<Nation> v;**로 생성한 벡터는 국가와 수도가 저장된 **Nation** 타입의 객체를 저장합니다.
  - 프로그램 내에서 벡터에 **Nation** 객체를 여러 개 미리 저장하여 사용합니다.
- ```

Nation n[] = {
    Nation("미국", "워싱턴"), Nation("영국", "런던"), Nation("프랑스", "파리"),
    Nation("중국", "베이징"), Nation("일본", "도쿄"), Nation("러시아", "모스크바"),
    Nation("브라질", "브라질리아"), Nation("독일", "베를린"), Nation("멕시코", "멕시코시티")
};

```

- `vector<Nation>v`는 국가와 수도를 입력 받아 새로운 객체를 추가 할 수도 있습니다.
- `vector<Nation>v`에 저장된 데이터를 이용하여 랜덤하게 데이터를 꺼내 퀴즈를 볼 수도 있습니다.
- 실행 화면은 아래와 같습니다.

```
PS C:\yanges\lecture\lecture_src\cpp> ./a
***** 국가의 수도 맞추기 게임을 시작합니다. *****

1(정보 입력), 2(퀴즈), 3(종료) : 2
영국의 수도는? 서울
NO !!
독일의 수도는? 베를린
Correct !!
브라질의 수도는? quit

1(정보 입력), 2(퀴즈), 3(종료) : 1
현재 9개의 나라가 입력되어 있습니다.

국가와 수도를 입력하세요(quit quit이면 입력 종료)
10 : 중국 베이징
already exists !!
10 : 대한민국 서울
11 : 북한 평양
12 : quit quit

1(정보 입력), 2(퀴즈), 3(종료) : 2
멕시코의 수도는? 평양
NO !!
영국의 수도는? 평양
NO !!
북한의 수도는? 평양
Correct !!
미국의 수도는? quit

1(정보 입력), 2(퀴즈), 3(종료) : 3
PS C:\yanges\lecture\lecture_src\cpp>
```

```
#include <iostream>
#include <string>
#include <vector>
#include <ctime>
#include <cstdlib>
using namespace std;

class Nation {
    string nation; //나라이름
    string capital; //수도
public:
    Nation(string nation, string capital) {
        this->nation = nation;
        this->capital = capital;
    }
    string getCapital() { return capital; }
    string getNation() { return nation; }
    void show() {
        cout << '(' << nation << ',' << capital << ')';
    }
}
```

```

};

class NationGame {
    vector<Nation> v;
    void input();
    void list();
    void quiz();
    bool exist(string nation);
public:
    NationGame();
    void run();
};

NationGame::NationGame() {
    Nation n[] = {
        Nation("미국", "워싱턴"), Nation("영국", "런던"), Nation("프랑스", "파리"),
        Nation("중국", "베이징"), Nation("일본", "도쿄"), Nation("러시아", "모스크바"),
        Nation("브라질", "브라질리아"), Nation("독일", "베를린"), Nation("멕시코", "멕시코시티")
    };

    for (int i = 0; i < 9; i++)
        v.emplace_back(n[i]); //v.push_back(n[i]);

    srand((unsigned)time(0)); //시작할 때마다, 다른 랜덤수를 발생시키기 위한 seed 설정
}

void NationGame::run() {
    cout << "***** 국가의 수도 맞추기 게임을 시작합니다. *****" << endl;
    while (true) {
        int cmd;
        cout << "Wn1(정보 입력), 2(퀴즈), 3(종료) : ";
        cin >> cmd;
        switch (cmd) {
            case 1: input(); break;
            case 2: quiz(); break;
            case 3: return;
        }
    }
}

void NationGame::quiz() {
    while (true) {
        int index = rand() % v.size(); // 0에서 RAND_MAX(32767) 사이의 랜덤한 정수가 n에 발생

        cout << v[index].getNation() << "의 수도는? ";
        //
        //이곳을 완성합니다.
        //
    }
}

void NationGame::input() {

```



```

string nation, capital;

cout << "현재 " << v.size() << "개의 나라가 입력되어 있습니다." << endl;
cout << "Wn국가와 수도를 입력하세요(quit quit이면 입력 종료)" << endl;

while (true) {
    //
    //이곳을 완성합니다.
    //
}
}

bool NationGame::exist(string nation) {

}

int main() {
    NationGame game;
    game.run();
}

```