# 창의적 소프트웨어 설계 실습 문제 12 – hw12-1

**제출 기한**

12 월 4 일 화 23:59

I. hw12-1(mkdir hw12-1)라는 폴더를 만들고 GitLab 에 push
II. hw12-1 디렉토리에 각 문제에서 요구하는 파일들을 작성
III. **'make'명령을 수행하여 숙제가 모두 빌드**
IV. 최종 버전을 GitLab 에 commit
V. **시간과 파일명, 입력과 출력 방식 반드시 지키기**

## 과제 1

두 인자의 value 를 교환하는 myswap function 을 template 을 이용하여 작성한다. myswap function template 의 두 인자로 references (**T**& a, **T**&b)를 입력 받는다.

- function overloading 을 사용하면 안된다.
- myswap function 은 "=" operator 가 정의되어있는 자료형, class, struct 에 대해서 동작가능해야 한다.
- 주어진 main 함수의 내용은 변경하면 안된다.

파일명: myswap (myswap.cc, myswap.h)
입력 : 없음
출력 : swap 되기 전 변수 값과 swap 후의 변수 값

```
$ ./myswap
a= 3 b= 1
a= 1 b= 3
c= 3.1 d= 1.5
c= 1.5 d= 3.1
e= Hello f= World!
e= World! e= Hello
$
```

## 과제 2

임의의 자료형의 array 를 담을 수 있는 class template, MyContainer 를 주어진 선언을 기반으로 작성한다.

작성한 MyContainer 를 상속받아 std::vector 와 동일한 동작을 하는 MyVector class 를 주어진 선언을 기반으로 작성한다.

```
template <class T>
class MyContainer {
    public:
        MyContainer();
        MyContainer(int n);
        ~MyContainer();

        void clear();
        int size() const;

    protected:
        T* obj_arr;
        int n_elements;
};

template <class T>
class MyVector : public MyContainer<T> {
    public:
        MyVector();
        MyVector(int n);

        MyVector<T>& operator= (const MyVector<T>& rhs);

        bool empty() const;
        int max_size() const;

        void reserve(int new_cap);
        void push_back(T obj);
        void pop_back();
        T& front();
        T& back();
        T& at(const int idx);
        const T& at (const int idx) const;
        T& operator[] (const int idx);
        const T& operator[] (const int idx) const;

        MyVector<T> operator+(const MyVector<T>& rhs);
```

```
    private:
        int capacity;
};
```

- clear() 함수는 해당 배열에 있는 내용을 초기화한다. 단, 할당된 메모리를 해제하지는 않는다. 동시에 n_elements 를 0 으로 세팅한다.
- size() 함수는 현재 할당되어있는 배열의 크기가 아닌, 들어있는 원소의 개수를 반환한다.
- capacity 는 현재 할당되어있는 배열의 크기를 의미한다.
  일반적으로 capacity >= n_elements 이다.
- push_back()을 통해 MyVector 에 요소를 추가할 때, 현재 배열에 담긴 맨 마지막 요소 뒤에 새로운 요소를 추가한다. 배열의 공간이 부족할 경우 max(1, 2 * capacity) 크기의 배열을 새로 할당한 후 요소를 채워 넣는다.
- pop_back()을 통해 MyVector 의 맨 마지막 요소를 제거한다. 이 때 capacity 는 변하지 않으며, n_elements 만 변화한다.
- + 연산자는 두 배열을 더한 새로운 배열을 반환한다. 이 때, 새로운 배열의 capacity 는 두 배열의 capacity 의 합으로 한다.

main 함수에서는 사용자의 입력을 반복적으로 받으며, 다음과 같은 입력을 설명된 방식으로 처리한다. 상세한 동작 예시는 실행예시를 참고한다.

- 프로그램 시작시 int, double, string 중 하나를 입력받는다.
- new capacity
  - 주어진 타입을 요소로 가지고 주어진 capacity 를 가지는 MyVector 오브젝트를 생성한다.
  - 새로 만들어진 오브젝트는 0 부터 순차적으로 id 를 부여받는다.
- dump
  - 현재 생성되어있는 오브젝트들의 정보를 출력한다.
  - 출력할 정보: 오브젝트의 id, 오브젝트 내의 요소의 수, 오브젝트의 capacity
- add id item
  - id 에 해당하는 MyVector 오브젝트에 새로운 item 을 추가한다.
- pop id
  - id 에 해당하는 MyVector 오브젝트에 맨 마지막 요소를 삭제한다.
  - 삭제할 요소가 없을 경우 아무 일도 일어나지 않는다
- clear id
  - id 에 해당하는 MyVector 오브젝트를 clear() 한다
- join id1 id2
  - id1 에 해당하는 MyVector 오브젝트와 id2 에 해당하는 MyVector 오브젝트를 더한 결과를 출력한다.

- 두 오브젝트를 더한 결과는 출력이후 소멸한다.
- 요소의 type 이 다른 오브젝트를 더하는 경우는 존재하지 않는다.

- quit
  - 프로그램을 종료한다.

main 함수는 제공되며, 제공된 main 함수는 변형하여선 안된다.

파일명 : my_vector (my_vector.h my_vector_main.cc)

입력 / 출력

```
$ ./my_vector
int
new 20
new 5
dump
0, 0, 20
1, 0, 5
add 0 15
15
add 0 20
15, 20
add 0 1
15, 20, 1
add 0 5
15, 20, 1, 5
dump
0, 4, 20
1, 0, 5
add 1 4
4
add 1 2
4, 2
add 1 3
4, 2, 3
dump
0, 4, 20
1, 3, 5
new 2
dump
```

```
0, 4, 20
1, 3, 5
2, 0 ,2
add 2 15
15
add 2 60
15, 60
add 2 3
15, 60, 3
dump
0, 4, 20
1, 3, 5
2, 3, 4
pop 0
15, 20, 1
clear 0
dump
0, 0, 20
1, 3, 5
2, 3, 4
join 1 2
4, 2, 3, 15, 60, 3
quit
$
```