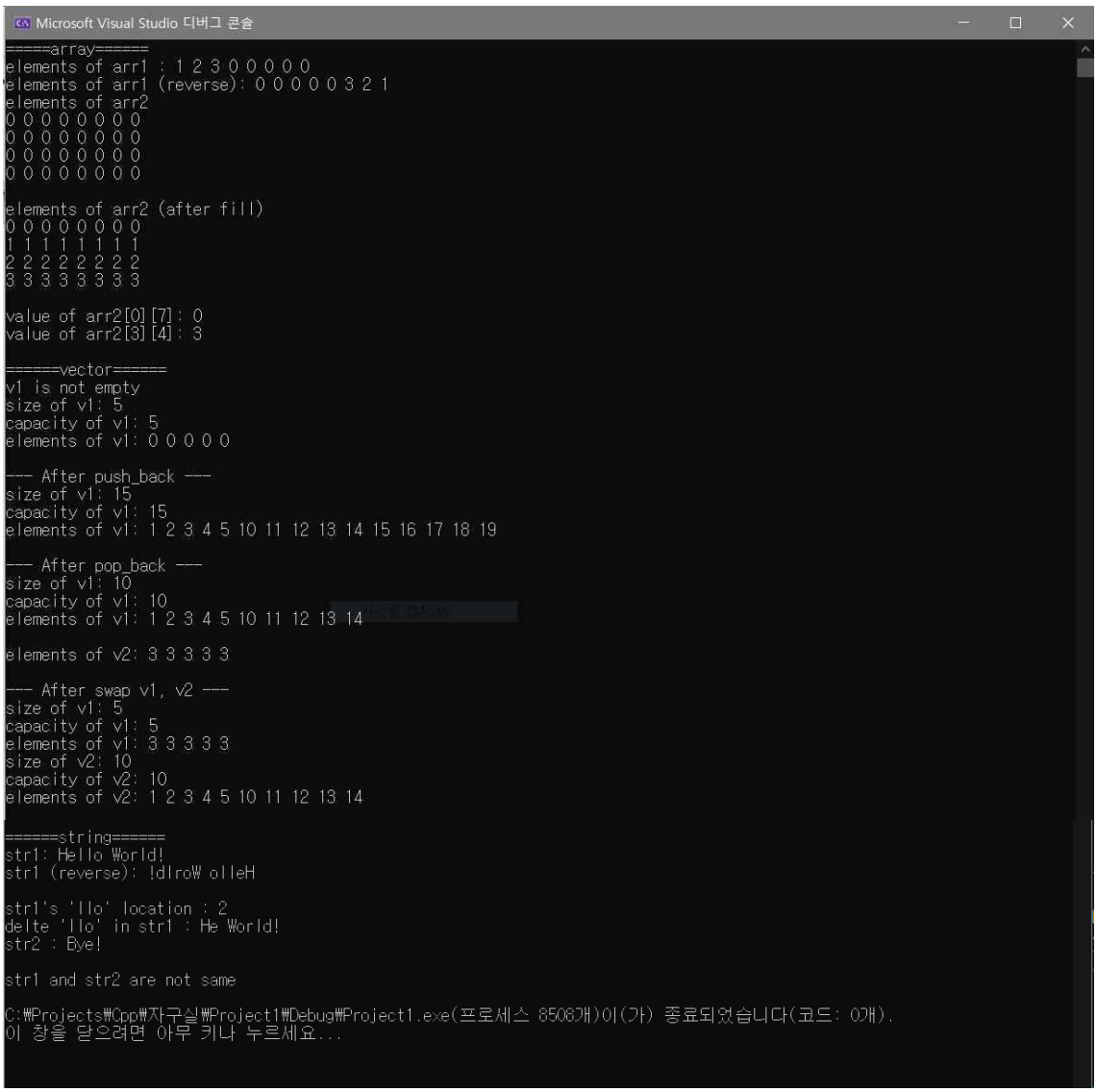
HW05

201920310 유지성

실행 결과



```
12      array<int, 8> arr1 = { 1, 2, 3 };
13
14      cout << "elements of arr1 : ";
15      for (int i= 0 ; i < arr1.size(); ++i) {
16          cout << arr1[i] << " ";
17      }
18      cout << endl;
```

크기가 8이고 1, 2, 3, 0, 0, 0, 0, 0 으로 초기화 되어있는 배열 생성 후 각 요소를 size함수와 []로 출력

```
20      cout << "elements of arr1 (reverse): ";
21      array<int, 8>::reverse_iterator riter;
22      for (riter = arr1.rbegin(); riter != arr1.rend(); ++riter) {
23          cout << *riter << " ";
24      }
25      cout << endl;
```

reverse_iterator로 배열을 거꾸로 출력

```
28      array<array<int, 8>, 4> arr2 = { 0 };
29      array<array<int, 8>, 4>::iterator row;
30      array<int, 8>::iterator col;
31      cout << "elements of arr2" << endl;
32      for (row = arr2.begin(); row != arr2.end(); ++row) {
33          for (col = (*row).begin(); col != (*row).end(); ++col) {
34              cout << *col << " ";
35          }
36          cout << endl;
37      }
38      cout << endl;
```

0으로 초기화 한 2차열 배열을 생성 후 iterator로 출력

```
40      int i = 0;
41      for (row = arr2.begin(); row < arr2.end(); ++row) {
42          (*row).fill(i++);
43      }
```

각 row마다 fill 함수로 같은 수 채워넣기

```
45      cout << "elements of arr2 (after fill)" << endl;
46      for (int i=0; i < arr2.size(); i++) {
47          for (int j = 0; j < arr2[i].size(); j++) {
48              cout << arr2[i][j] << " ";
49          }
50          cout << endl;
51      }
52      cout << endl;
```

size와 []을 이용해 2차원 배열 출력

```
54      cout << "value of arr2[0][7]: ";
55      cout << (arr2.front()).back() << endl;
56      cout << "value of arr2[3][4]: ";
57      cout << (arr2.at(3)).at(4) << endl;
58      cout << endl;
```

front 와 back, at을 사용해 배열의 요소 출력

```
62      vector<int> v1(5);
63
64      if (v1.empty())
65          cout << "v1 is empty" << endl;
66      else
67          cout << "v1 is not empty" << endl;
```

크기가 5이고 0으로 초기화 된 벡터 선언 후 비어있나 확인

```
69      cout << "size of v1: " << v1.size() << endl;
70      cout << "capacity of v1: " << v1.capacity() << endl;
71      cout << "elements of v1: ";
72      for (int i = 0; i < v1.size(); ++i) { cout << v1[i] << " "; }
73      cout << endl << endl;
```

벡터의 각 요소들을 size 함수와 []로 출력

```
75      for (int i = 0; i < v1.size(); ++i) { v1[i] = i + 1; }
76      for (int i = 10; i < 20; ++i) { v1.push_back(i); }
```

기존 벡터 요소들을 교체 후 push_back으로 새로이 추가

```
78      cout << "--- After push_back ---" << endl;
79      cout << "size of v1: " << v1.size() << endl;
80      cout << "capacity of v1: " << v1.capacity() << endl;
81      cout << "elements of v1: ";
82      vector<int>::iterator it;
83      for (it = v1.begin(); it != v1.end(); ++it) { cout << *it << " ";
84      cout << endl << endl;
```

iterator로 벡터 요소들 출력

```
86      for (int i = 0; i < 5; i++)
87          v1.pop_back();
88      v1.shrink_to_fit();
```

pop_back으로 제거 후 제거 된 size 에 맞게 shrink_to_fit으로 capacity 축소

```
97      vector <int> v2(5, 3);
98      cout << endl << "elements of v2: ";
99      for (int i = 0; i < v2.size(); ++i) { cout << v2[i] << " "; }
100     cout << endl << endl;
101
102     v1.swap(v2);
```

크기가 5, 3으로 초기화 된 벡터 v2를 선언 후 v1과 swap

```
117     string str1 = "Hello World!";
118     cout << "str1: " << str1 << endl;
119     string::reverse_iterator rit;
120     cout << "str1 (reverse): ";
121     for (rit = str1.rbegin(); rit != str1.rend(); ++rit) { cout << *r
122     cout << endl << endl;
```

“Hello World!” 로 초기화 된 string선언 후 출력, reverse_iterator로 str1 역순 출력

```
124     int n = str1.find("llo", 0);
125     cout << "str1's 'llo' location : " << n << endl;
126     str1.erase(2, 3);
```

find로 특정 문자열 위치 찾기

```
126     str1.erase(2, 3);
127     cout << "delte 'llo' in str1 : " << str1 << endl;
```
erase로 문자열 제거

```
129     string str2 = "Bye!";
130     cout << "str2 : " << str2 << endl << endl;
131     if (str1.compare(str2))
132         cout << "str1 and str2 are not same" << endl;
133     else
134         cout << "str and str2 are same" << endl;
```
새로운 문자열 str2 생성 후 str1이랑 compare 함수로 같은지 비교

느낀점

기존에 array와 벡터 사용할 때와 다른 생성 방법이 있다는 것을 알게 되었다.
iterator가 무엇인지 알게 되었다.