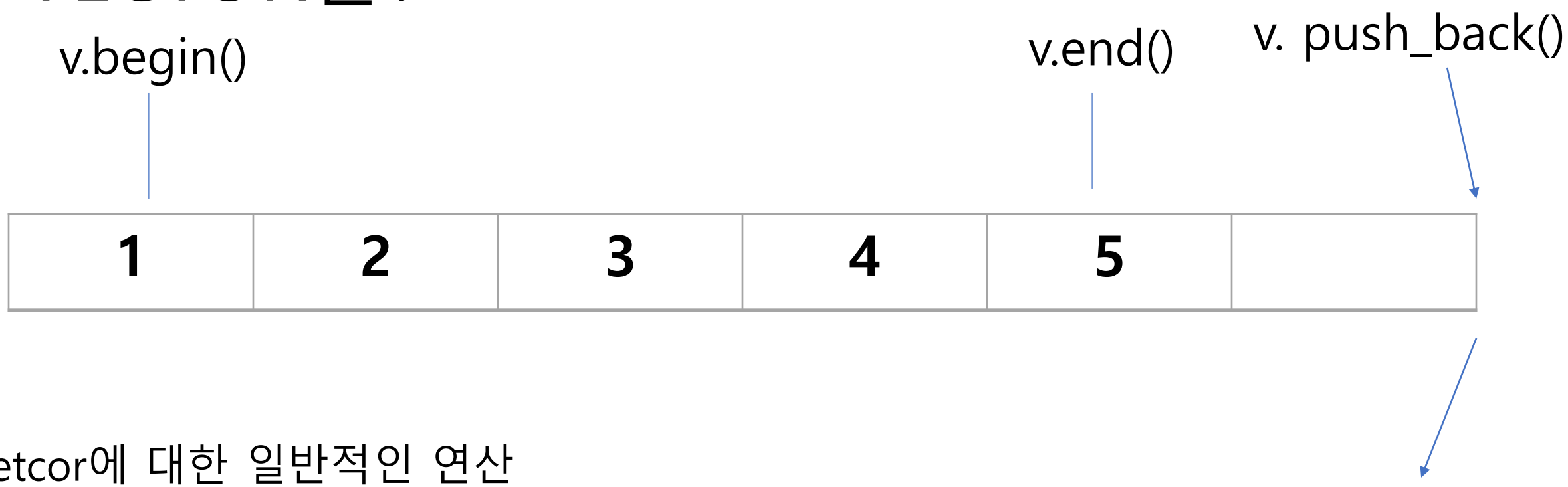
# VECTOR

# VECTOR란?

C++ STL에는 두개의 container 중에 vector는 Sequnece Container
자동으로 메모리가 할당되는 배열 이라고 설명된다.
Vector를 생성하면 **heap 메모리에 동적으로 할당** 된다.
또한 다른 container와 마찬가지로 template을 이용하기 때문에
**데이터 타입은 자유롭게 이용** 할 수 있다.

# VECTOR란?




Vetcor에 대한 일반적인 연산

- 임의 접근(Random Access)
- 벡터의 끝에서 원소를 삽입하거나 삭제
- 원소의 삽입과 삭제

# VECTOR의 장단점

<장점>

1. 배열과 달리 자동으로 메모리를 할당시켜 주어 처음부터 원소의 개수를 지정해둘 필요가 없고, 원소의 삽입/삭제 시 효율적인 메모리 관리 가능.
2. Vector의 중간의 원소를 삭제하거나, vector의 크기를 구하는 작업 등을 알아서 해주는 유용한 멤버 함수들이 많다.
3. 배열 기반이므로 랜덤 접근(Random Access)이 가능하다.

<단점>

1. 배열 기반의 container이므로 원소의 삽입, 삭제가 자주 수행되면 시간적인 측면에서 비효율 적이다.

# VECTOR의 사용 방법

- Vector를 사용하기 위한 헤더파일

  #include <vector>

- using namespace std;를 추가하면 vector 앞에 std 안써도 된다.

- Vector 선언 방식

  vector<[Data type]> [Name]

# VECTOR의 사용 방법

- Vector 생성자

vector<[type]>v -> [type]형의 빈 vector를 생성

vector<[type]>v(n) -> 0으로 초기화 된 n개의 원소를 가지는 [type]형의 빈 vector를 생성

vector<[type]>v(n, m) -> m으로 초기화 된 n개의 원소를 가지는 [type]형의 빈 vector를 생성

vector<[type]>v2(v1) -> v1을 복사하여 v2 vector를 생성

vector<vector<[type]>>v -> [type]형의 2차원 vector 생성

vector<[type]>v = {a1, a2, a3, ...} -> {a1, a2, a3, ...} 으로 초기화 된 [type]형의 vector 생성

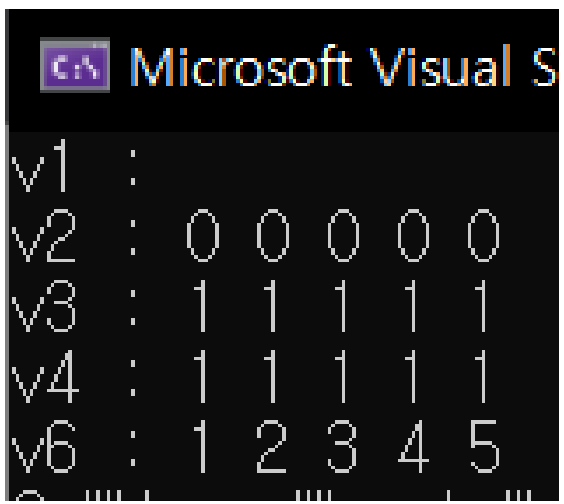
# VECTOR의 사용 방법

- 사용 예제

vector소스.cpp

Project3 (전역 범위)

```cpp
#include<iostream>
#include<vector>

using namespace std;

int main(void) {
    vector<int> v1;            // 빈 vector
    vector<int> v2(5);        // {0, 0, 0, 0, 0}
    vector<int> v3(5, 1);      //{1, 1, 1, 1, 1}
    vector<int> v4(v3);       //v4 = v3
    vector<vector<int>> v5;  //2차원 벡터
    vector<int> v6 = { 1, 2, 3, 4, 5 };

    cout << "v1 : ";
    for (int i = 0; i < v1.size(); i++) cout << v1[i] << ' ';
    cout << "₩nv2 : ";
    for (int i = 0; i < v2.size(); i++) cout << v2[i] << ' ';
    cout << "₩nv3 : ";
    for (int i = 0; i < v3.size(); i++) cout << v3[i] << ' ';
    cout << "₩nv4 : ";
    for (int i = 0; i < v4.size(); i++) cout << v4[i] << ' ';
    cout << "₩nv6 : ";
    for (int i = 0; i < v6.size(); i++) cout << v6[i] << ' ';
}
```



# VECTOR의 사용 방법

- vector 멤버 함수

v.assign(n, m) -> m으로 n개의 원소 할당

v.at(index) -> index번째 원소를 반환한다. 유효한 index인지 체크하기 때문에 안전하다.

v[index] -> index번째 원소를 반환한다. 배열과 같은 방식이며 유효성을 체크하지 않는다.

v.front() -> 첫 번째 원소를 반환한다.

v.back() -> 마지막 원소를 반환한다.

v.clear() -> 모든 원소를 제거한다. 메모리는 그대로 남아있게 된다.

v.begin() ->첫 번째 원소를 가리키는 반복자(iterator)를 반환한다.

v.end() -> 마지막 원소 다음을 가리키는 반복자(iterator)를 반환한다.

v.push_back(m) -> 마지막 원소 뒤에 원소 m을 삽입한다.

v.pop_back() -> 마지막 원소를 제거한다.

# VECTOR의 사용방법

- 사용 예제

vector소스.cpp

Project3 (전역 범위)

```
1  #include<iostream>
2  #include<vector>
3
4  using namespace std;
5  void print_(vector<int> v) {
6      for (int i = 0; i < v.size(); i++)cout << v[i] << ' ';
7      cout << '₩n';
8  }
9
10 int main(void) {
11     vector<int> v = { 1, 2, 3, 4, 5 };
12     print_(v);  //{1, 2, 3, 4, 5}
13     v.assign(2, 5);
14     print_(v);  //{5, 5}
15     v.push_back(3);
16     v.push_back(4);
17     print_(v);  //{5, 5, 3, 4}
18     cout << v.at(0) << ' ' << v.at(2) << '₩n';  // 5 3
19     cout << v[1] << ' ' << v[3] << '₩n';  // 5 4
20     cout << v.front() << ' ' << v.back() << " ₩n"; // 5 4
21     v.pop_back();
22     print_(v);  // {5, 5, 3}
23     v.clear();
24     print_(v);  // {  }
25 }
```

```
Microsoft V
1 2 3 4 5
5 5
5 5 3 4
5 3
5 4
5 4
5 5 3
```

# VECTOR의 사용 방법

- vector 멤버 함수

v.rbegin() ->거꾸로 시작해서 첫 번째 원소를 가리키는 반복자(iterator)를 반환한다.

v.rend() -> 거꾸로 시작해서 마지막 원소를 가리키는 반복자(iterator)를 반환한다.

v.reserve(n) -> n개의 원소를 저장할 공간을 예약한다.

v.resize(n) -> 크기를 n개로 변경한다. 커진 경우에는 빈 곳을 0으로 초기화한다.

v.resize(n, m) -> 크기를 n개로 변경한다. 커진 경우에는 빈 곳을 m으로 초기화한다.

v.size() -> 원소의 개수를 반환한다.

v.capacity() -> 할당된 공간의 크기를 반환한다. (size()와 다름)

v2.swap(v1) -> v1과 v2를 swap한다.

v.insert(iter, m) -> iter가 가리키는 위치에 m의 값을 삽입한다. 그리고 해당 위치를 가리키는 반복자(iterator)를 반환

# VECTOR의 사용 방법

- vector 멤버 함수

v.insert(iter, k, m) -> iter가 가리키는 위치부터 k개의 m 값을 삽입한다. 다음 원소들은 뒤로 밀린다.

v.erase(iter) -> iter 반복자가 가리키는 원소를 제거한다. capacity는 그대로 유지된다

v.erase(start, end) -> start 반복자부터 end 반복자까지 원소를 제거한다.

v.empty() -> vector가 비어있으면 true를 반환한다.

v.max_size() -> v가 담을 수 있는 최대 원소의 개수(메모리 크기) 반환

v.shrink_to_fit() -> capacity의 크기를 vector의 실제 크기에 맞춤

# VECTOR의 사용방법

- 사용 예제

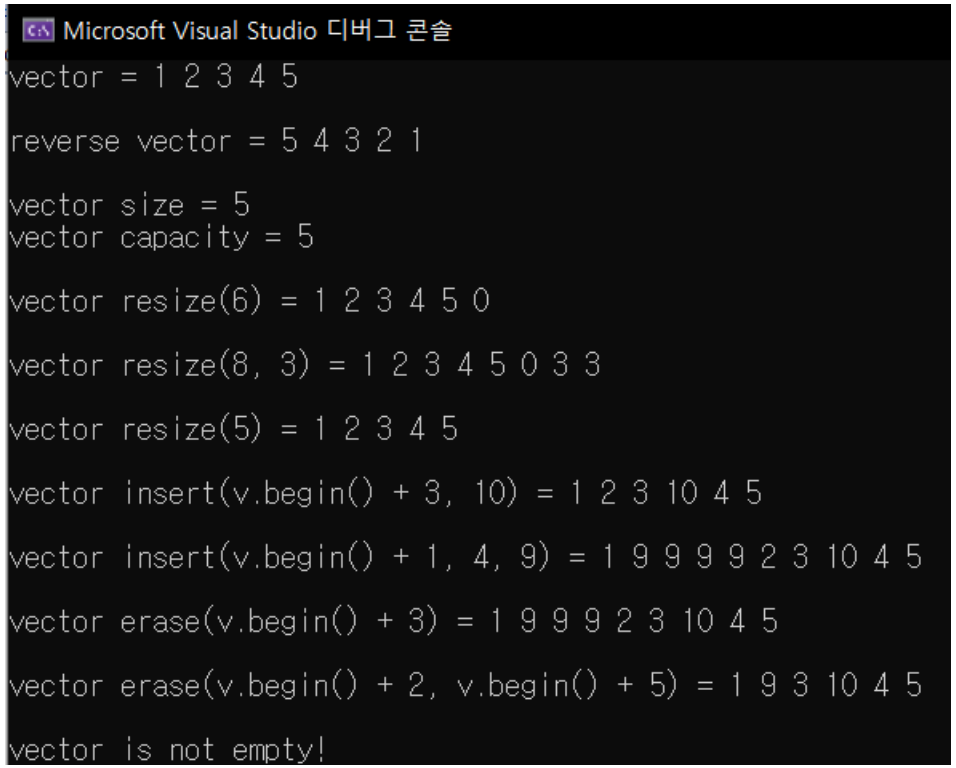
vector소스.cpp*

Project3 (전역 범위)

```
#include<iostream>
#include<vector>

using namespace std;

int main(void) {
    vector<int> v = { 1, 2, 3, 4, 5 };
    vector<int> ::iterator iter;

    cout << "vector = ";
    for (iter = v.begin(); iter != v.end(); iter++)    cout << *iter << ' ';  //iterator 이용
    cout << "₩n₩n";

    cout << "reverse vector = ";
    vector<int> ::reverse_iterator riter;
    for (riter = v.rbegin(); riter != v.rend(); riter++) cout << *riter << ' '; //reverse iterator 이용
    cout << "₩n₩n";

    cout << "vector size = " << v.size() << '₩n';
    cout << "vector capacity = " << v.capacity() << "₩n₩n";

    v.resize(6);
    cout << "vector resize(6) = ";
    for (int i = 0; i < v.size(); i++)     cout << v[i] << ' ';
    cout << "₩n₩n";

    v.resize(8, 3);
    cout << "vector resize(8, 3) = ";
    for (int i = 0; i < v.size(); i++) cout << v[i] << ' ';
    cout << "₩n₩n";

    v.resize(5);
    cout << "vector resize(5) = ";
    for (int i = 0; i < v.size(); i++) cout << v[i] << ' ';
    cout << "₩n₩n";

    v.insert(v.begin() + 3, 10);
    cout << "vector insert(v.begin() + 3, 10) = ";
    for (int i = 0; i < v.size(); i++) cout << v[i] << ' ';
    cout << "₩n₩n";

    v.insert(v.begin() + 1, 4, 9);
    cout << "vector insert(v.begin() + 1, 4, 9) = ";
    for (int i = 0; i < v.size(); i++) cout << v[i] << ' ';
    cout << "₩n₩n";

    v.erase(v.begin() + 3);
    cout << "vector erase(v.begin() + 3) = ";
    for (int i = 0; i < v.size(); i++) cout << v[i] << ' ';
    cout << "₩n₩n";

    v.erase(v.begin() + 2, v.begin() + 5);
    cout << "vector erase(v.begin() + 2, v.begin() + 5) = ";
    for (int i = 0; i < v.size(); i++) cout << v[i] << ' ';
    cout << "₩n₩n";

    if (v.empty()) cout << "vector is empty!";
    else cout << "vector is not empty!";
}
```

# VECTOR의 사용방법

- 사용 예제



# 2차원 VECTOR

```
1       vector<vector<int>> v;
2       vector<vector<int>> v(N, vector<int>(K));
3       vector<vector<int>> v(N, vector<int>(K, val));
```

1번째 줄 빈 vector를 생성하는 것이다.
2번째 줄은 N행 K열 vector를 생성하는 것이다.
3번째 줄은 vector를 val 값으로 모두 초기화하는 것이다.