Container: Vector tutorial

- It is equivalent to a list data structure in python
- There are more member functions in vector class, but I just wrote down some of popularly used functions

characteristics	python	c++
A dynamic ordered list	list	vector
A dynamic set of (key-value) pairs	dict	unordered_map
A doubly linked list	deque	list
A dhynamic set of unordered unique values	set	unordered_set

```
// constructing vectors
#include <iostream>
#include <vector>

int main ()
{
  // constructors used in the same order as described above:
  std::vector<int> first;                                // empty vector of ints
  std::vector<int> second (4,100);                       // four ints with value
100
  std::vector<int> third (second.begin(),second.end());  // iterating through
second
  std::vector<int> fourth (third);                       // a copy of third

  // the iterator constructor can also be used to construct from arrays:
  int myints[] = {16,2,77,29};
  std::vector<int> fifth (myints, myints + sizeof(myints) / sizeof(int) ); //
myints: starting point of the range, which is a pointer to the beginning of the
myints array
  // myints + sizeof(myints)/sizeof(int) is the ending point of the range

  std::cout << "The contents of fifth are:";
  for (std::vector<int>::iterator it = fifth.begin(); it != fifth.end(); ++it)
    std::cout << ' ' << *it;
  std::cout << '\n';

  return 0;
} // output the contetns of fifth are: 16 2 77 29
```

Member function

Iterators

- **begin iterator -> 1 -> .... -> 10 -> end iterator**
    - begin: return iterator pointing to the first element in the vector
    - end: return iterator to end
    - iterator it에 저장되어 있는 value를 알고 싶은 경우에는 *it 사용
- rbegin: return reverse iterator to reverse beginning
- rend: return reverse iterator to reverse end

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

int main(){

    std::vector<int> my_vec;
    for (int i =1 ; i<=5; i++>) my_vec.push_back(i);

    std::cout << "my vector contains: ";
    // random access iterator type : vector의 어디든 pointing할 수 있음. 여기서는
iterator member type에 begine, end와 같은 다양한 곳을
    // point하는 iterator여서 그런 이름이 붙여진 것 같다.
    for (std::vector<int>::iterator it = my_vec.begin(); it != myvector.end();
++it){
        std::cout << " " << *it;
    }

    std::cout << '\n';

    return 0
} // output: my vector contains: 1 2 3 4 5
```

Capacity

- size
- max_size: the maximum number of elements that the vector can hold
- resize
    - void resize (size_type n)
    - void resize(size_type n, const value_type& val)
- empty

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

int main(){
    vector<int> my_vector;

    for (int i = 0; i<= 5; i++) my_vector.push_back(i);
```

```
    std::cout << "my_vector has a size of "
              << my_vector.size() << '\n';

    std::cout << "my_vector max_size "
              << my_vector.max_size() << '\n';

    while (!my_vector.empty()){
        sum += my_vector.back();
        my_vector.pop_back();
    }

    std::cout << "total: " << sum << '\n';

    std::vector<int> myvector;

    // set some initial content:
    for (int i=1;i<10;i++) myvector.push_back(i);

    myvector.resize(5);
    myvector.resize(8,100);
    myvector.resize(12);

    std::cout << "myvector contains:";
    for (int i=0;i<myvector.size();i++)
        std::cout << ' ' << myvector[i];
    std::cout << '\n';
    // output myvector contains: 1 2 3 4 5 100 100 100 0 0 0 0
}
```

Element access

- front:
  - returns a refrence to the first element in the vector
  - unlike member vector::begin, which returns an iterator to this same element, this function returns a direct reference
- back

```
//vector::front
#include <iostream>
#include <vector>

int main(){
    std::vector<int> myvector;

    myvector.push_back(78);
    myvector.push_back(16);

    //now front equals 78, and back 16
```

```
    myvector.front() -= myvector.back();

    std::cout << "myvector.front() is now" << myvector.front() << '\n';

    return 0;
} // output myvector.front() is now 62
```

Modifiers

- push_back
- pop_back
- insert
- swap : exchange the content of the container by the content of x, which is anotehr vector object of the same type. Sizes may differ
- erase : erase elements
    - erase(const_iterator position)
    - erase(const_iterator first, const_iterator last)
- clear : remove "all" elements

```
//swap function
#include <iostream>
#include <vector>

int main(){

    std::vector<int> foo (3, 100); // three ints with a value of 100
    std::vector<int> bar (5, 200); // five ints with a value of 200

    foo.swap(bar); // swap function

    std::cout << "foo contains: " ;
    for (unsigned i =0; i<foo.size() ; i++){
        std::cout << ' ' << foo[i]; // python list처럼 access 가능
    }
    std::cout << '\n';

    std::cout << "bar contains: " ;
    for (unsigned i = 0; i < bar.size(); i++){
        std::cout << " " << bar[i];
    }
    std::cout << '\n';

    return 0;
}
```

```cpp
//inserting into a vector
#include <vector>
#include <iostream>

int main(){
    std::vector<int> myvector (3, 100);
    std::vector<int>::iterator it;

    it = myvector.begin();
    it = myvector.insert(it, 200);

    myvector.insert(it, 2, 300);

    // 'it' no longer valid, get a new one;
    it = myvector.begin();
    return 0;
}
```

```cpp
//erase
// erasing from vector
#include <iostream>
#include <vector>

int main ()
{
  std::vector<int> myvector;

  // set some values (from 1 to 10)
  for (int i=1; i<=10; i++) myvector.push_back(i);

  // erase the 6th element
  myvector.erase (myvector.begin()+5);

  // erase the first 3 elements:
  myvector.erase (myvector.begin(),myvector.begin()+3);

  std::cout << "myvector contains:";
  for (unsigned i=0; i<myvector.size(); ++i)
    std::cout << ' ' << myvector[i];
  std::cout << '\n';

  return 0;
}//output myvector contains: 4 5 7 8 9 10
```