**STL 기초 – 변경알고리즘(요소 재배치)**

요소 재배치 : 요소 재배치 함수들은 요소들을 특정한 다른 값으로 바꾸거나 요소끼리 교환함으로써 위치를 변경한다.

void replace (FwdIt first, FwdIt last, const Type& Old, const Type& New);

void reverse(BiIt first, BiIt last);

void rotate(FwdIt first, FwdIt middle, FwdIt last);

* replace 함수는 first ~ last 사이의 Old값을 모조리 뒤져 New로 대체한다. (나머지 값은 그대로 남는다)
* replace\_if 는 특정값이 아닌 특정 조건을 만족하는 값을 다른 값으로 변경 할 수 있다.
* replace\_copy 는 복사함수이다.
* replace\_copy\_if는 조건자를 취하는 복사함수이다.

🡪 reverse는 구간의 모든 요소의 순서를 반대로 뒤집는다. 제일 앞에 요소는 제일 뒤쪽으로 가고, 제일 뒤쪽에 있는 요소가 제일 앞쪽으로 온다.

#include <iostream>

#include <vector>

#include <algorithm>

#include <iterator>

#include <cstring>

using namespace std;

template<typename C> void dump(const char \*desc, C c) { cout.width(12);cout << left << desc << "==> ";

copy(c.begin(),c.end(),ostream\_iterator<typename C::value\_type>(cout," ")); cout << endl; }

int main()

{

const char \*str="Notebook Computer";

vector<char> vc(&str[0],&str[strlen(str)]);

dump("원본",vc);

replace(vc.begin(),vc.end(),'o','a');

dump("replace",vc);

rotate(vc.begin(),vc.begin()+2,vc.end());

dump("rotate",vc);

reverse(vc.begin(),vc.end());

dump("reverse",vc);

}

BiIt partition(BiIt first, BiIt last, UniPred F);

BiIt stable\_partition(BiIt first, BiIt last, UniPred F);

* 단항조건자 F는 구간내의 요소들을 인수로 전달받아 이 요소가 조건에 맞는지 아닌지 판별한다.
* partition 함수는 F의 평가 결과에 따라 조건에 맞는 요소는 구간의 앞쪽으로 이동시키고 그렇지 않은 요소는 뒤쪽으로 이동시키며 뒤쪽 그룹의 시작 위치를 리턴한다.
* stable\_partition 함수는 재배치후에도 요소들의 원래 순서가 유지되는 안정된 버전이다. 같은 그룹에 속하는 값들끼리라도 원래 앞족에 있었다면 재배치 후 에도 여전히 앞쪽에 배치되며, 안정성이 있는 대신 속도는 partition 보다 느리며 더 많은 메모리를 소모한다.

#include <iostream>

#include <vector>

#include <algorithm>

#include <functional>

#include <iterator>

using namespace std;

template<typename C> void dump(const char \*desc, C c) { cout.width(12);cout << left << desc << "==> ";

copy(c.begin(),c.end(),ostream\_iterator<typename C::value\_type>(cout," ")); cout << endl; }

void main()

{

int ari[]={3,1,4,1,5,9,2,6,5,3,5,8,9,7,9,3,2,3,8};

vector<int> vi(&ari[0],&ari[sizeof(ari)/sizeof(ari[0])]);

dump("원본",vi);

partition(vi.begin(),vi.end(),bind2nd(greater<int>(),5));

dump("partition",vi);

vector<int> ar2(&ari[0],&ari[sizeof(ari)/sizeof(ari[0])]);

stable\_partition(ar2.begin(),ar2.end(),bind2nd(greater<int>(),5));

dump("stable",ar2);

}

* 5보다 큰 값들이 왼쪽으로 옮겨졌고 오른쪽에는 그렇지 않은 값들이 배치된다. 5는 5보다 크지 않으므로 이동 대상이 아니다. 안정성이 있는 재배치 함수는 요소의 원래 순서를 그대로 유지하는데 원본에 있는 5보다 큰 값 9, 6, 8, 9가 원본 순서 그대로 재배치되어 있다. 반면 안정성이 없는 재배치 함수는 조건을 기준으로 좌우로 옮기기만 할 뿐 순서 유지는 하지 않는다