**STL 기초 – for\_each , equal**

for\_each : 지정구간을 반복하면서 지정한 작업을 수행한다.

* for\_each는 분류상 읽기 알고리즘에 속하는데 다른 읽기 알고리즘과는 달리 요소를 변경할 수도 있다. 순회중에 요소값을 바꾼다거나 요소가 가리키는 대상체를 삭제할 수도 있다. 그래서 포인터의 컨테이너에서 요소의 메모리를 정리할 때 흔히 사용된다. 그러나 요소의 내용물만 건드릴 수 있을 뿐 컨테이너 자체를 변경하지는 못하므로 여전히 읽기 알고리즘이다.

\*for\_each 예제

#include <iostream>

#include <string>

#include <vector>

#include <algorithm>

using namespace std;

void func(string str)

{

cout<< str << endl;

}

int main()

{

vector<string> vs;

vs.push\_back("로보트 태권 브이");

vs.push\_back("들장미 소녀 캔디");

vs.push\_back("바보 온달과 평강 공주");

vs.push\_back("독수리 오형제");

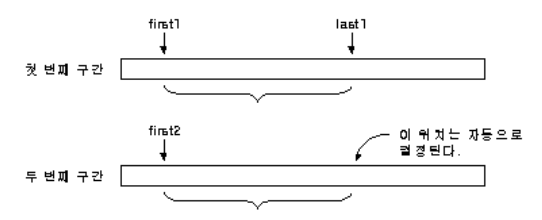
for\_each(vs.begin(),vs.end(),func);

}

equal : 두개의 반복자 구간을 비교하여 두 구간이 완전히 일치하는지 아닌지 검사한다.

**bool equal(InIt1 first1, InIt1 last1, InIt2 first2 [, BinPred F]);**

* first1 ~ last1 사이의 구간과 first2 이후의 구간에 있는 요소들을 일대일로 비교해 보고 모든 요소가 일치하면 true를 리턴하고 하나라도 틀리면 false를 리턴한다. 두 번째 구간은 시작 위치를 지정하는 반복자만 전달되고 끝 반복자는 전달되지 않는데 두 번째 구간도 첫 번째 구간과 길이가 같다고 가정한다



두 개의 똑같은 구간을 전달받는 함수들은 보통 두 번째 구간의 길이를 전달받는 인수가 따로 없고 첫 번째 구간의 길이를 사용한다. 두 반복자 구간은 반드시 같은 컨테이너에 소속될 필요는 없으며 컨테이너의 타입이 달라도 상관없다

equal 예제

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

int main()

{

int ari[]={8,9,0,6,2,9,9};

vector<int> vi(&ari[0],&ari[7]);

if (equal(&ari[0],&ari[7],vi.begin())) {

puts("두 구간은 동일하다");

} else {

puts("두 구간은 틀리다.");

}

}

🡺정수 배열로부터 정수 벡터를 만들고 이 두 컨테이너가 동일한지 비교했으므로 비교 결과는 같다고 나옴, 벡터 초기화 후에 ari[5]=99; 대입문으로 배열의 요소 하나를 변경해 놓고 비교하면 틀리다는 결과가 출력된다. equal은 전체가 같아야만 true를 리턴하며 단 하나라도 틀리면 결과는 false이다. 함수 객체를 지정하면 두 요소가 같다는 조건을 마음대로 변경할 수 있다.

equal 예제2

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

bool compare(double a,double b)

{

return ((int)a == (int)b);

}

int main()

{

double af1[]={ 45.34, 77.84, 96.22, 91.04, 85.24 };

double af2[]={ 45.99, 77.25, 96.86, 91.23, 86.13 };

if (equal(&af1[0],&af1[4],&af2[0],compare)) {

puts("지정 구간의 정수부가 모두 같다.");

} else {

puts("지정 구간의 정수부 중 일부가 일치하지 않는다.");

}

}

\*mismatch : equal의 반대 함수인데 두 반복자 구간 중 최초로 틀린 부분이 어디인가를 찾는다

🡺equal은 같다 다르다만 조사하는데에 비해 mismatch 는 틀리다면 어디쯤이 틀린지도 조사한다.

**pair<InIt1, InIt2> mismatch(InIt1 first1, InIt1 last1, InIt2 first2 [,BinPred F]);**

🡺 리턴값은 두 구간이 최초로 달라진 지점의 반복자 쌍을 pair 객체로 묶어서 리턴한다. 이 구조체의 first, last를 점검해 보면 어디가 최초로 다른 지점인지를 알 수 있다. 모든 구간이 일치한다면 last1과 first2+(last1-first1)의 쌍이 리턴되는데 first2+(last1-first1)이라는 수식이 복잡해 보이지만 말로 간단히 설명하면 두 번째 구간의 끝다음이다.

mismatch 예제

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

int main()

{

int ari[]={8,9,0,6,2,9,9};

vector<int> vi(&ari[0],&ari[7]);

vi[3]=7;

pair<int \*,vector<int>::iterator> p;

p=mismatch(&ari[0],&ari[7],vi.begin());

if (p.first != &ari[7]) {

printf("%d번째 자리(%d,%d)부터 다르다.\n",

p.first-ari,\*(p.first),\*(p.second));

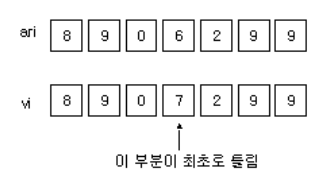
} else {

puts("두 컨테이너가 일치한다.");

}

}

* vi[3]=7; 로 벡터를 초기화 해 주었기 때문에, 두 컨테이너는 일치하지 않는다.



시험 오답 체크 (mismatch 이용)

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

int main()

{

int answer[]={1,1,4,3,2,4,3,2,3,4,1,2,4,4,3,2,1,3,2,4};

int paper[]= {1,1,4,3,3,4,3,1,3,4,1,2,4,4,3,4,1,3,2,2};

pair<int \*,int \*> p;

int i;

for (i=0;;) {

p=mismatch(&answer[i],&answer[20],&paper[i]);

if (p.first == &answer[20]) break;

printf("%d번 틀림, 정답=%d, 학생 답=%d\n",

p.first-answer+1,\*(p.first),\*(p.second));

i=p.first-answer+1;

}

}