**STL 기초 – 변경알고리즘2**

요소생성 : 요소 생성 함수는 새로운 요소를 만들어 지정한 위치에 대입하는 함수들이다.

* 요소를 만드는건 맞지만 컨테이너에 삽입하는 것이 아니라, 기존 요소를 파괴하고 대입한다는 점에 주의

void fill(FwdIt first, FwdIt last, const T& val);

void fill\_n(OutIt first, Size n, const T& val);

\*fill : fill은 반복자 구간을 지정하는데 비해 fill\_n은 시작위치와 개수를 지정한다는 점이 다르다. 반복자 구간은 이미 메모리가 확보되어 있어야 하며 , 만약 생성된 값을 꼭 넣으려면 삽입 반복자를 사용해야하며 이때에는

void fill\_n(Outlt first, Size n , const T& val); 만 사용 할 수 있다.

fill 예제

#include <iostream>

#include <vector>

#include <algorithm>

#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; }

int main()

{

int ari[]={1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16};

vector<int> vi(&ari[0],&ari[16]);

fill(vi.begin()+2,vi.end()-5,99);

dump("vi",vi);

}

* fill\_n(vi.begin()+2,5,99); 으로 수정하면 2번째부터 5개의 요소가 99로 바뀐다. 컨테이너의 모든 요소에 대해 일관적으로 대입하고 싶을 때 이 함수를 사용한다.

fill(vi.begin()+2,vi.end()-5,99); 🡺 정수 값을 가진 벡터의 일정 구간을 99로 채움 , 구간 내 모든 값 99로 변경

void random\_shuffle(RanIt first, RanIt last[, UniOp& op]);

* 반복자 구간의 요소들을 무작위로 마구 섞으며 , 반드시 반복자 구간은 유효해야한다.
* 난수로 요소를 생성해서 대입하는 것이 아니라 이미 존재하는 값들의 순서를 난수로 변경함으로써 섞는다.

random\_shuffle 예제

#include <Turboc.h>

#include <iostream>

#include <algorithm>

#include <cstring>

using namespace std;

int main()

{

char str[]="abcdefghijklmnopqrstuvwxyz";

randomize();

puts(str);

random\_shuffle(&str[0],&str[strlen(str)]);puts(str);

random\_shuffle(&str[0],&str[strlen(str)]);puts(str);

random\_shuffle(&str[0],&str[strlen(str)]);puts(str);

}

void generate(FwdIt first, FwdIt last, Gen g);

void generate\_n(OutIt first, Dist n, Gen g);

* fill 함수와 마찬가지로 반복자 구간을 인수로 전달받을 수 있고, 시작 위치와 개수를 인수로 받을 수도 있다.
* g 는 인수를 받지않고 컨테이너 요소 타입을 리턴하는 함수 객체이다.

generate 예제

#include <iostream>

#include <vector>

#include <algorithm>

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 fibo()

{

static int i1=1,i2=1;

int t;

t=i1+i2;

i1=i2;

i2=t;

return t;

}

void main()

{

vector<int> vi(10);

generate(vi.begin(),vi.end(),fibo);

dump("vi",vi);

}