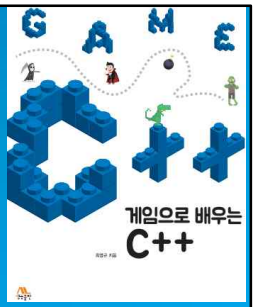
게임으로 배우는 C++

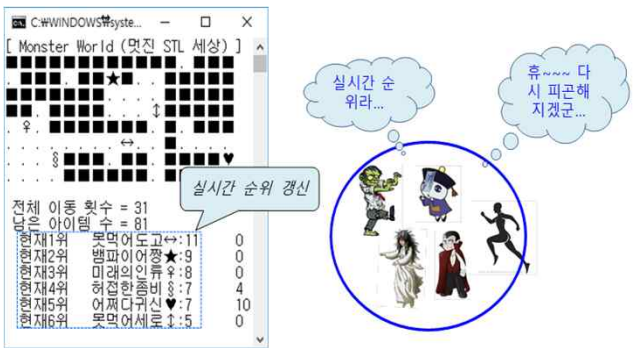


CHAPTER 14

표준 템플릿 라이브러리

몬스터 월드9 (실시간 순위 갱신)

- Monster World 9
  - 게임이 진행되는 동안 실시간으로 순위를 갱신함
  - STL의 벡터를 이용해 몬스터 월드를 꾸밈
  - STL의 sort 알고리즘 등을 이용해 순위 계산



2

14장 학습 목표

- 표준 템플릿 라이브러리를 이해한다.
- 컨테이너와 알고리즘 및 반복자의 용도를 이해한다.
- 다양한 컨테이너를 사용할 수 있는 능력을 기른다.
- 함수 객체의 개념을 이해한다.
- 다양한 알고리즘을 사용할 수 있는 능력을 기른다.

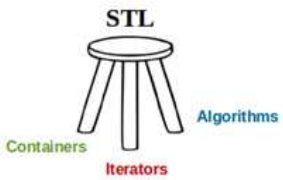
14.1 표준 템플릿 라이브러리

- C++ 표준 템플릿 라이브러리

C++ 표준 템플릿 라이브러리

- **Standard Template Library(STL)이란?**
  - 다양한 자료구조와 알고리즘을 지원
  - 검증된 라이브러리
  - 모든 C++ 컴파일러에서 지원
  - 어떤 자료형에도 적용할 수 있음

- STL 구성
  - 컨테이너(container)
  - 알고리즘(algorithm)
  - 반복자(iterator)



5

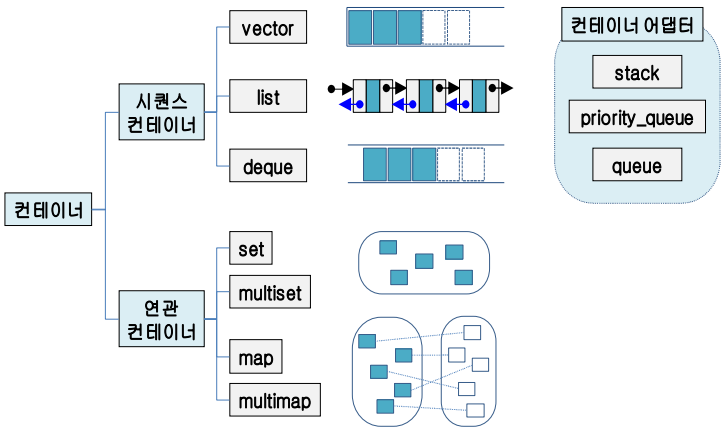
14.2 STL의 구성요소

- 컨테이너
- 알고리즘
- 반복자가 필요한 이유
- 반복자
- 반복자의 연산

6

컨테이너

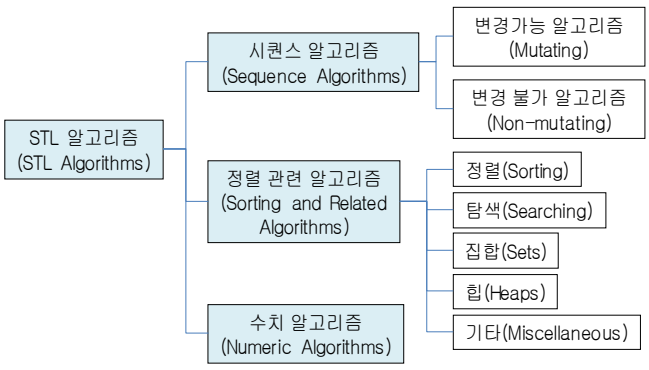
- 자료를 저장하는 창고



7

알고리즘

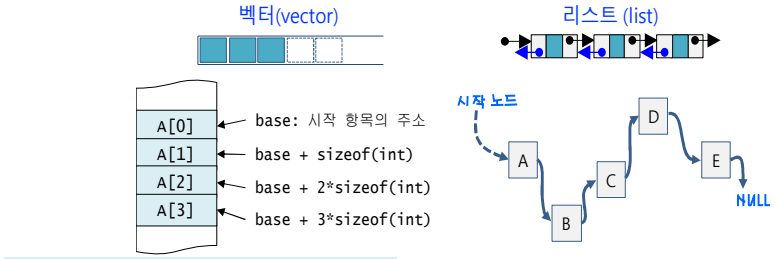
- 자료의 검색, 정렬, 초기화, 재배치, 분리, 병합 등



8

반복자가 필요한 이유

- 반복자가 필요한 이유
  - 컨테이너마다 항목의 접근 방법이 다름




```
for (int i = 0; i < v2.size(); i++)
        cout << v2[i] << " ";
```

```
for (Node* p = l.head() ; p != NULL ; p=p->link)
        cout << p->data << " ";
```

  - 동일한 알고리즘(함수)을 어떻게 사용할 수 있도록 할까???
  - 일반화된 포인터 ? ➔ 반복자 !!!

9

반복자

- 일반화된 포인터
  - 컨테이너의 멤버를 가리키는 객체
  - 일관된 방법으로 컨테이너의 요소에 접근
  - 특별한 위치의 반복자를 반환하는 함수를 지원
    - » begin()은 첫 번째 요소를 반환
    - » end()는 마지막 요소가 지났는지를 나타내는 값을 반환

- 종류
  - 전방 반복자(forward iterator): 앞으로만 이동
  - 양방향 반복자(bidirectional iterator): 양방향 이동
  - 임의 접근 반복자(random access iterator): 어떤 위치로 이동

10

반복자의 연산

- 반복자 종류별로 가능한 연산들(연산자 중복)

반복자	연산자
입력 반복자 InputIterator	==, != ++, *(값을 반환할 경우)
출력 반복자 OutputIterator	==, !=, ++, *(할당할 경우)
전방향 반복자 ForwardIterator	==, !=, ++, *
양방향 반복자 BidirectionalIterator	==, !=, ++, --, *
임의 접근 반복자 RandomAccIterator	==, !=, ++, --, [ ], * iterator + n: iterator 다음에 n번째 원소를 참조 iterator – iterator: 두 반복자 사이의 원소의 수

14.3 순차 컨테이너

- 벡터: vector
- 덱: deque
- 리스트: list

순차 컨테이너

- vector
- list
- deque

- 기본 연산들

연산	설명	vector	list	deque
container()	기본 생성자	O(1)	O(1)	O(1)
container(*size*)	*size* 크기의 컨테이너	O(1)	O(n)	O(1)
container(*size,value*)	*size* 크기, 초기값 *value*인 컨테이너	O(n)	O(n)	O(n)
container(*iterator, iterator*)	다른 컨테이너로부터 초기화	O(n)	O(n)	O(n)
size()	항목의 개수	O(1)	O(1)	O(1)
begin()	첫 번째 항목의 위치(반복자)	O(1)	O(1)	O(1)
end()	마지막 항목 다음 위치(반복자)	O(1)	O(1)	O(1)
rbegin()	끝 항목의 위치(역 반복자)	O(1)	O(1)	O(1)
rend()	첫 항목 바로 앞의 위치(역 반복자)	O(1)	O(1)	O(1)
front()	첫 번째 항목 반환	O(1)	O(1)	O(1)
back()	마지막 항목 반환	O(1)	O(1)	O(1)
pop_back()	마지막 항목 삭제	O(1)	O(1)	O(1)
push_back(*value*)	맨 뒤에 항목 삽입	O(1)+	O(1)	O(1)+
pop_front()	첫 번째 항목 삭제		O(1)	O(1)+
push_front(*value*)	맨 앞에 항목 삽입		O(1)	O(1)+
clear()	모든 항목 삭제	O(1)	O(1)	O(1)
empty()	공백상태 검사	O(1)	O(1)	O(1)
erase(*iterator*)	중간 위치 항목 삭제	O(n)	O(1)	O(n)
insert(*iterator,value*)	중간에 삽입	O(n)	O(1)	O(n)
operator=(*container*)	대입연산자 중복정의	O(n)	O(n)	O(n)
operator[](*int*)	인덱스를 이용한 항목 추출	O(1)		O(1)
at(*int*)	항목 반환	O(1)		O(1)
capacity()	할당된 크기	O(1)		
resize(*size,value*)	할당된 크기 재조정	O(n)		O(n)

13

벡터: vector

```
template <typename T>
ostream& operator << (ostream& os, vector<T>& v) {
        os << "<";
        for (int i = 0; i < v.size(); i++)
                os << v[i] << " ";
        os << ">";
        return os;
}
```

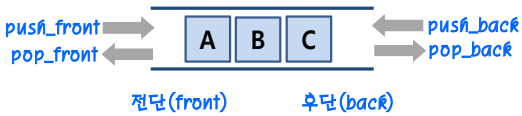
```
#include <vector>
void main()
{
        vector<int> vec(10);
        for (int i = 0; i < vec.size(); i++)
                vec[i] = rand() % 100;
        cout << "Before: " << vec << endl;
        for (vector<int>::iterator it = vec.begin() ; it != vec.end() ; ){
                if (*it % 2 == 0)
                        it = vec.erase(it);
                else ++it;
        }
        cout << "After : " << vec << endl;
}
```

짝수들 → 삭제

```
C:₩WINDOWS₩system32₩cmd.exe
Before: <41 67 84 0 69 24 78 58 62 64 >
After : <41 67 69 >
```

14

덱: deque




```
#include <deque>
void main()
{
        deque<int> dq;
        for (int i = 0; i < 10; i++) {
                int val = rand() % 100 + 1;
                if (val % 2 == 0) dq.push_back(val);
                else dq.push_front(val);
        }
        cout << "deque(인덱스): ";
        for (int i = 0; i < dq.size(); i++)
                cout << dq[i] << " ";
        cout << endl << "deque(반복자): ";
        for (auto curr = dq.begin(); curr != dq.end() ; ++curr)
                cout << *curr << " ";
        cout << endl;
}
```

```
C:₩WINDOWS₩system32₩cmd.exe
deque(인덱스): 65 63 59 79 25 1 35 42 68 70
deque(반복자): 65 63 59 79 25 1 35 42 68 70
```


리스트: list

```
template <typename T>
void printContainer(T& v, char* msg = "container") {
        cout << msg << "<";
        for (T::iterator it = v.begin(); it != v.end(); ++it)
                cout << *it << " ";
        cout << ">\n";
}
```

```
#include <list>
void main()
{
        list<int> sortList;
        for (int i = 0; i < 10; i++) {
                int val = rand() % 100;
                auto curr = sortList.begin();
                for (; curr != sortList.end(); ++curr)
                        if (val >= *curr) break;
                sortList.insert(curr, val);
        }
        printContainer(sortList, "Sorted List: ");
}
```

```
C:₩WINDOWS₩system32₩cmd.exe
Sorted List: <78 69 67 64 62 58 41 34 24 0 >
```

14.4 컨테이너 어댑터

- 스택: stack
- 큐: queue
- 우선순위 큐: priority_queue

17

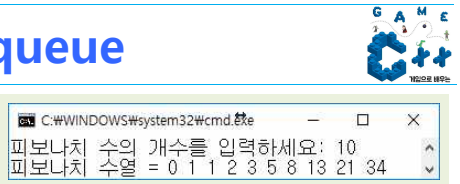
컨테이너 어댑터: stack

```
#include <stack>
…
void main()
{
        stack<char> st;
        string line;
        cout << "문장을 입력하세요 : ";
        getline(cin, line);
        for (int i = 0; i < line.length() ; i++)
                st.push(line[i]);
        cout << "입력 = " << line << endl;
        cout << "출력 = ";
        while (!st.empty()) {
                cout << st.top();
                st.pop();
        }
        cout << endl;
}
```

```
C:₩WINDOWS₩system32₩cmd.exe
문장을 입력하세요 : Game Over 12345
입력 = Game Over 12345
출력 = 54321 revO emaG
```

18

컨테이너 어댑터: queue



```
#include <queue>
…
void main()
{
        queue<int> que;
        int count;
        cout << "피보나치 수의 개수를 입력하세요: ";
        cin >> count;
        cout << "피보나치 수열 = ";
        que.push(0);
        que.push(1);
        for (int i = 0; i < count; i++) {
                int fibo = que.front();
                que.pop();
                cout << fibo << " ";
                que.push(fibo + que.front());
        }
        cout << endl;
}
```

```
C:₩WINDOWS₩system32₩cmd.exe
피보나치 수의 개수를 입력하세요: 10
피보나치 수열 = 0 1 1 2 3 5 8 13 21 34
```

19

컨테이너 어댑터: priority_queue

```
#include <queue>
…
void main()
{
        priority_queue<double> pque;
        double val;
        cout << "우선순위 큐 입력 순서 = ";
        for (int i = 0; i < 10; i++) {
                val = rand() % 1000 * 0.1;
                cout << val << " ";
                pque.push(val);
        }
        cout << endl;
        cout << "우선순위 큐 출력 순서 = ";
        while (!pque.empty()) {
                cout << pque.top() << " ";
                pque.pop();
        }
        cout << endl;
}
```

```
C:₩WINDOWS₩system32₩cmd.exe
우선순위 큐 입력 순서 = 4.1 46.7 33.4 50 16.9 72.4 47.8 35.8 96.2 46.4
우선순위 큐 출력 순서 = 96.2 72.4 50 47.8 46.7 46.4 35.8 33.4 16.9 4.1
```

20

14.5 연관 컨테이너

- 집합(set, multiset)
- 맵(map, multimap)

21

집합(set, multiset)

연산	설명
set(), set(*iterator, iterator*)	공집합과 주어진 범위로 집합을 생성하는 생성자
begin(), end()	집합의 시작과 끝을 가리키는 반복자
rbegin(), rend()	역순으로 집합의 시작과 끝을 가리키는 반복자
size(), empty()	항목의 개수 반환 및 공집합 검사
find(*value*)	value가 들어 있는 위치의 반복자 반환
insert(*value*)	주어진 값을 집합에 삽입
erase(*value*), erase(*iterator*)	주어진 값이나 반복자가 가리키는 원소 삭제
count(*value*)	값의 인스턴스 수를 세어 반환

22

집합: set, multiset

```
#include <set>
...
void main()
{
        set<int> simple;
        multiset<int> multiple;
        for (int i = 10; i < 15; i++)
                simple.insert(i);
        multiple.insert(simple.begin(), simple.end());
        for (int i = 13; i < 18; i++)
                multiple.insert(i);
        printContainer(simple, "Simple Set = ");
        printContainer(multiple, "Multiple Set = ");
        cout << "simple.count(14) = " << simple.count(14) << endl;
        cout << "multiple.count(14) = " << multiple.count(14)<<endl;
}
```

```
선택 C:₩WINDOWS₩system32₩cmd.exe
Simple   Set = <10 11 12 13 14 >
Multiple Set = <10 11 12 13 13 14 14 15 16 17 >
simple.count(14)   = 1
multiple.count(14) = 2
```

23

맵(map, multimap)

- 맵(map, multimap)

연산	설명
set(), set(*iterator, iterator*)	공집합과 주어진 범위로 집합을 생성하는 생성자
begin(), end()	집합의 시작과 끝을 가리키는 반복자
rbegin(), rend()	역순으로 집합의 시작과 끝을 가리키는 반복자
size(), empty()	항목의 개수 반환 및 공집합 검사
find(*value*)	value가 들어 있는 위치의 반복자 반환
insert(*value*)	주어진 값을 집합에 삽입
erase(*value*), erase(*iterator*)	주어진 값이나 반복자가 가리키는 원소 삭제
count(*value*)	값의 인스턴스 수를 세어 반환

24

맵: map, multimap

```
#include <map>
…
void main()
{
        map<string, string> myDic;
        map<string, string>::iterator dp;
        myDic["hello"] = "안녕하세요?";
        myDic["world"] = "아름다운 세상";
        myDic["data"] = "자료";
        myDic["structure"] = "구조";
        myDic["list"] = "리스트";
        for (dp = myDic.begin(); dp != myDic.end(); ++dp)
                cout << setw(12) << dp->first << " == " + dp->second + "\n";
        dp = myDic.find("structure");
        if (dp == myDic.end())
                cout << "[검색 실패] 찾는 단어가 없습니다.\n";
        else    cout << "[검색 성공] " << dp->first + " " + dp->second << endl;
        if (myDic.find("C++") == myDic.end())
                cout << "[검색 실패] 찾는 단어가 없습니다.\n";
        myDic.erase("structure");
        for (dp = myDic.begin(); dp != myDic.end(); ++dp)
                cout << setw(12) << dp->first << " == " + dp->second + "\n";
}
```

```
C:₩WINDOWS₩syste...
        data == 자료
       hello == 안녕하세요?
        list == 리스트
   structure == 구조
       world == 아름다운 세상
[검색 성공] structure 구조
[검색 실패] 찾는 단어가 없습니다.
        data == 자료
       hello == 안녕하세요?
        list == 리스트
       world == 아름다운 세상
```

25

14.6 STL 알고리즘

- STL 알고리즘 분류
- 함수 객체
- 내장된 함수 객체
- 기본적인 알고리즘
- 변경 가능 알고리즘들
- 변경 불가능 알고리즘들
- 정렬 관련 알고리즘들

26

STL 알고리즘 분류

분류		주요 연산	알고리즘 함수
시퀀스 알고리즘	변경 가능	방문	for_each()
		값 설정	generate(), generate_n(), fill(), fill_n()
		변환	transform(), partition(), stable_partition(), random_shuffle(), reverse(), reverse_copy(), replace(), replace_if(), replace_copy(), replace_copy_if(), rotate(), rotate_copy()
		복사	copy(), copy_backward()
		삭제	remove(), remove_if(), remove_copy(), remove_copy_if(), unique(), unique_copy()
		순열 생성	next_permutation() prev_permutation()
		교환	swap(), swap_ranges()
	변경 불가	탐색	count(), count_if(), find(), find_if(), find_first_of(), find_end(), adjacent_find(), search(), search_n()
		비교	equal(), mismatch()
		최대/최소	max(), min(), max_element(), min_element()
정렬관련 알고리즘		정렬	sort(), stable_sort(), nth_element(), partial_sort(), partial_sort_copy()
		탐색 (Binary)	binary_search(), lower_bound(), upper_bound(), equal_range(), lexicographical_compare()
		병합	inplace_merge(), merge()
		집합	includes(), set_symmetric_difference(), set_difference(), set_union(), set_intersection()
		힙	make_heap(), sort_heap(), push_heap(), pop_heap()
수치 알고리즘			accumulate(), inner_product(), adjacent_difference(), partial_sum()

27

함수 객체

- 초기화 함수

```
vector<int> v(10);
generate(v.begin(), v.end(), rand);
```

```
int randRange1to6() { return rand()%6 + 1; }
...
generate(v.begin(), v.end(), randRange1to6);
```

- 함수 객체(Function Object)
  - 함수 호출 연산자 () 중복 함수를 정의한 클래스의 객체

```
class RandRange {
        int from, to;
public:
        RandRange(int f=1, int t=6): from(f),to(t) {}
        int operator()() { return rand()%(to-from+1) + from; }
}
```

```
generate(v.begin(), v.end(), RandRange(1,6));
generate(w.begin(), w.end(), RandRange(10,20));
```

28

내장된 함수 객체

- 미리 정의된 함수 객체
- 함수 객체의 장점은 생성자를 사용할 수 있다는 것

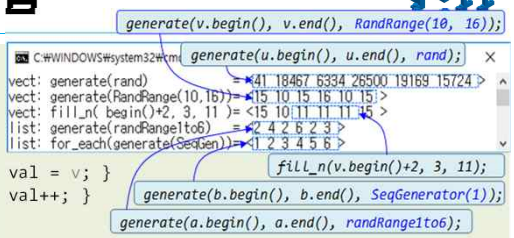
```
#include <functional>                    // 내장된 함수 객체 사용
```

Arithmetic operations		Comparisons	
plus	x + y	equal_to	x == y
minus	x – y	not_equal_to	x != y
multiplies	x * y	greater	x > y
divides	x / y	less	x < y
modulus	x % y	greater_equal	x >= y
negate	-x	less_equal	x <= y
Logical operations		**Bitwise operations**	
logical_and	x && y	bit_and	x & y
logical_or	x || y	bit_or	x | y
logical_not	!x	bit_xor	x ^ y

```
sort(v.begin(), v.end());                // 오름 차순 정렬
sort(v.begin(), v.end(), greater<int>());// 내림 차순 정렬
```


기본적인 알고리즘

```
#include <algorithm>
…
class SeqGenerator {
        int val;
public:
        SeqGenerator(int v = 1) { val = v; }
        int operator()() { return val++; }
};
void main() {
        vector<int> u(6), v(6);
        list<int> a(6), b(6);
        generate(u.begin(), u.end(), rand);
        generate(v.begin(), v.end(), RandRange(10, 16));
        printContainer(u, "vect: generate(rand) = ");
        printContainer(v, "vect: generate(RandRange(10,16))= ");
        fill_n(v.begin() + 2, 3, 11);
        printContainer(v, "vect: fill_n( begin()+2, 3, 11 )= ");
        generate(a.begin(), a.end(), randRange1to6);
        generate(b.begin(), b.end(), SeqGenerator(1));
        printContainer(a, "list: generate(randRange1to6) = ");
        cout << "list: for_each(generate(SeqGen))= <";
        for_each(b.begin(), b.end(), print);
        cout << ">" << endl;
}
```



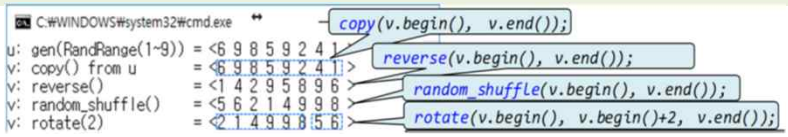
변경 가능 알고리즘들(1/3)

```
inline int square(int n) { return n * n; }
inline bool isOdd(int val) { return (val % 2) == 1; }
void main() {
        vector <int> u(8), v(8), w(8), y(3);
        list<int> a(8);
        generate(u.begin(), u.end(), RandRange(1, 9));
        printContainer(u, "u: gen(RandRange(1~9)) = ");
        copy(u.begin(), u.end(), v.begin());          // copy
        printContainer(v, "v: copy() from u = ");

        reverse(v.begin(), v.end());                  // reverse
        printContainer(v, "v: reverse() = ");

        random_shuffle(v.begin(), v.end());           // random_shuffle
        printContainer(v, "v: random_shuffle() = ");

        rotate(v.begin(), v.begin() + 2, v.end());    // rotate
        printContainer(v, "v: rotate(2) = ");
```



변경 가능 알고리즘들(2/3)

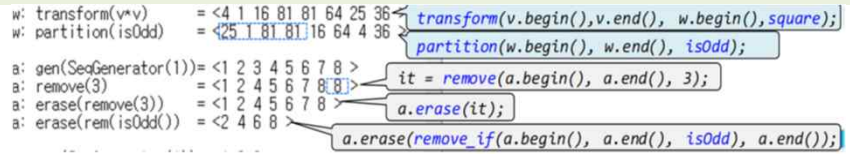
```
        transform(v.begin(),v.end(), w.begin(), square);        // transform
        printContainer(w, "w: transform(v*v) = ");

        partition(w.begin(), w.end(), isOdd);                   // partition
        printContainer(w, "w: partition(isOdd) = ");

        generate(a.begin(), a.end(), SeqGenerator(1));
        printContainer(a, "\na: gen(SeqGenerator(1))= ");

        auto it = remove(a.begin(), a.end(), 3);                // remove
        printContainer(a, "a: remove(3) = ");

        a.erase(it);                                            // erase
        printContainer(a, "a: erase(remove(3)) = ");
        a.erase(remove_if(a.begin(), a.end(), isOdd), a.end());
        printContainer(a, "a: erase(rem(isOdd()) = ");
```



변경 가능 알고리즘들(3/3)

```
        generate(y.begin(), y.end(), SeqGenerator(1));          // : 1, 2, 3
        printContainer(y, "\ny: gen(SeqGenerator(1))= ");
        while (next_permutation(y.begin(), y.end())) { // next_permutation
                printContainer(y, " permutations of y.... = ");
        }
}
```

변경 불가능 알고리즘들(1/2)

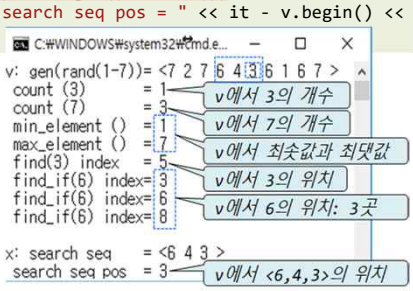
```
class ValueFinder{
        int val;
public:
        ValueFinder(int v = 1)      { val = v; }
        bool operator()(int v)      { return val == v; }
};
void main()
{
    vector <int> u(10), v(10), w(10), x(3);
    vector<int>::iterator it;
    generate(v.begin(), v.end(), RandRange(1, 7));
    printContainer(v, "v: gen(rand(1-7))= ");

    cout << " count (3) = " << count(v.begin(), v.end(), 3) << endl;
    cout << " count (7) = " << count(v.begin(), v.end(), 7) << endl;

    cout << " min_element () = " << *(min_element(v.begin(), v.end())) << endl;
    cout << " max_element () = " << *(max_element(v.begin(), v.end())) << endl;
```

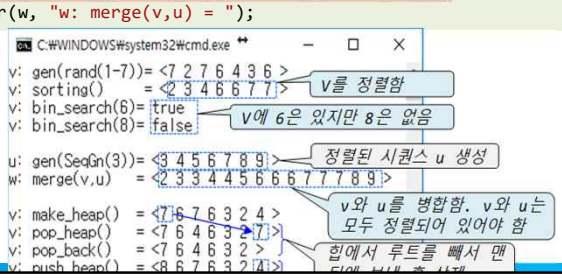
변경 불가능 알고리즘들(2/2)

```
    it = find(v.begin(), v.end(), 3);
    if (it != v.end()) cout << " find(3) index = " << it - v.begin() << endl;
    for (it = v.begin();; it++) {
         it = find_if(it, v.end(), ValueFinder(6));
         if (it == v.end()) break;
         cout << " find_if(6) index= " << it - v.begin() << endl;
    }
    copy_n(v.begin() + 3, 3, x.begin());
    printContainer(x, "\nx: search seq = ");
    it = search(v.begin(), v.end(), x.begin(), x.end());
    if (it != v.end()) cout << " search seq pos = " << it - v.begin() << endl;
}
```



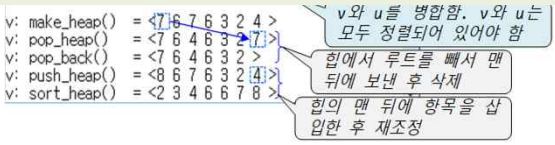
정렬 관련 알고리즘(1/2)

```
void main()
{
        vector <int> u(7), v(7), w(14);
        generate(v.begin(), v.end(), RandRange(1, 7));
        printContainer(v, "v: gen(rand(1-7))= ");
        sort(v.begin(), v.end());
        printContainer(v, "v: sorting() = ");
        bool b6 = binary_search(v.begin(), v.end(), 6);          // binary_search
        bool b8 = binary_search(v.begin(), v.end(), 8);
        cout << "v: bin_search(6)= " << (b6 ? "true" : "false") << endl;
        cout << "v: bin_search(8)= " << (b8 ? "true" : "false") << endl;
        generate(u.begin(), u.end(), SeqGenerator(3));
        printContainer(u, "\nu: gen(SeqGn(3))= ");
        merge(v.begin(), v.end(), u.begin(), u.end(), w.begin());// merge
        printContainer(w, "w: merge(v,u) = ");
```



정렬 관련 알고리즘(2/2)

```
        make_heap(v.begin(), v.end());                  // make_heap
        printContainer(v, "\nv: make_heap() = ");
        pop_heap(v.begin(), v.end());                   // pop_heap
        v.pop_back();                                   // v.pop_back()
        printContainer(v, "v: pop_heap() = ");
        v.push_back(8);                                 // v.push_back()
        push_heap(v.begin(), v.end());                  // push_heap()
        printContainer(v, "v: push_heap() = ");
        sort_heap(v.begin(), v.end());                  // sort_heap()
        printContainer(v, "v: sort_heap() = ");
}
```



37

14.7 응용: MonsterWorld 9

- Monster World 8: 실시간 순위 갱신
- 실시간 정렬
- 고찰

38

Monster World 9: 실시간 순위 갱신

- STL의 벡터 vector를 사용
  - 몬스터 월드 수정

```
class MonsterWorld {
        vector < vector<int> > world;
        ...
public:
        MonsterWorld(int w,int h): world(h),canvas(w,h),xMax(w),yMax(h){
                for (int y = 0; y < yMax; y++)
                        world[y] = vector<int>(w,1);          // 생성 + 초기화
        }
```

  - 몬스터 클래스

```
void eat(vector<vector<int>>& map) {...}
virtual void move(vector<vector<int>>& map, int maxx, int maxy) {...}
```

  - 캔버스 클래스

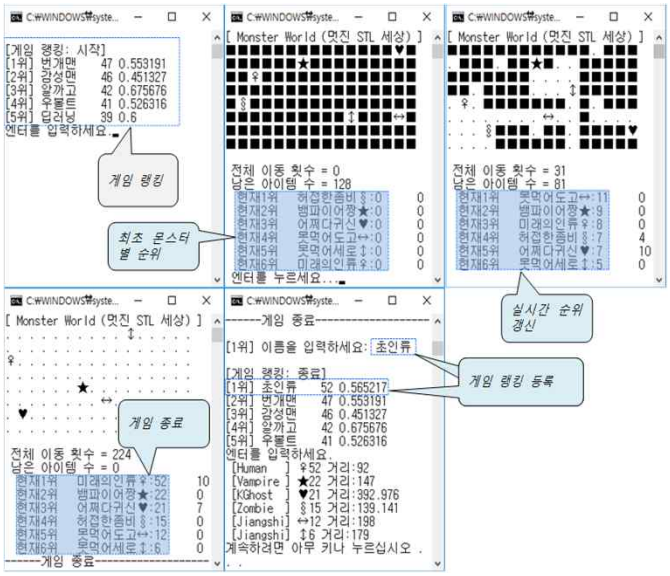
```
class Canvas {
        vector<string> line;        // 화면 출력을 위한 벡터 객체
        ...
```


실시간 정렬

- 게임이 진행되는 동안 먹은 아이템을 순으로 정렬
  - STL의 sort 알고리즘 + count 알고리즘

```
#include <algorithm>                    // STL의 알고리즘 사용
inline bool compare(Monster* first, Monster* second){
        return first->nItem > second->nItem;
}
class MonsterWorld {
        ...
        int countItems() {
                int nItems = 0;
                for (int y = 0; y < yMax; y++)
                        nItems += count(world[y].begin(),world[y].end(),1);
                return nItems;
        }
        void print() {
                ...
                sort(pMon.begin(), pMon.end(), compare);
                ...
        }
};
```

실행 결과

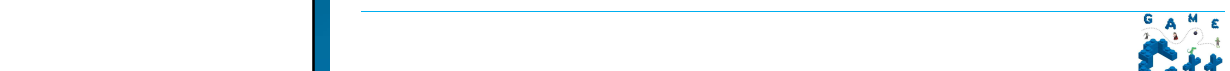


고찰

- 템플릿을 만드는 과정은 약간 복잡하지만, 사용하는 것은 일반 클래스와 거의 비슷하다. 복잡한 클래스나 함수가 템플릿으로 제공되면 적극적으로 사용해 보아야 할 것이다.
- 몬스터 맵과 같은 2차원 형태의 자료도 vector를 이용해 손쉽게 만들 수 있었다. 특히 동적 메모리 할당이나 해제를 더 이상 신경 쓰지 않아도 되어 매우 편리하다.

14장 요약문제, 연습문제, 실습문제

43



감사합니다!

44