사수 이름	오창한	커리큘럼 명	C++ STL
부사수 이름	최예린	시작일 / 제출일	2021.11.04 / 2021.11.05

커리큘럼 목적

STL의 자료구조를 알고, vector, map, list 각각의 장단점과 차이점을 이해하고 구현할 수 있다.

알게 된 것

1. vector
자동으로 메모리가 할당되는 배열 (like 스택)
맨 뒤쪽에서 삽입 삭제 가능, 중간에 값 삽입 삭제 가능하나 빈번하게 발생하면 비효율적
선언 : vector <[data type]> [변수이름]

v[idx] : idx번째 원소 참조
v.clear() : 모든 원소 제거
v.push_back(n) : 마지막 원소 뒤에 원소 n을 삽입
v.pop_back() : 마지막 원소 제거
v.insert(i,n) : i번째 위치에 n을 삽입
v.size() : 원소의 개수를 반환
v.begin() : 첫번째 원소를 참조
v.end() : 마지막 원소 참조

2. map
Key와 Value가 쌍으로 저장, key 값은 중복 불가능
노드기반, 균형 이진트리 구조
선언 : map<[Data type 1],[Data type 2]> [변수이름];

m[key] = value : 원소를 추가 또는 수정
m.insert(k) : pair 객체인 k를 삽입
m.insert(iter, k) : 원소를 pair 형태로 추가
m.erase(start, end) : start 부터 end 까지 원소 삭제
m.find(key) : key 값에 해당하는 iterator를 반환

3. list
순서를 유지하는 구조 (like 이중연결리스트), 노드 기반 컨테이너
멤버 함수에서 정렬, 이어붙이기 가능
원소 탐색 시 임의접근 반복자는 불가능, 양방향 반복자(++, --)를 이용해 탐색
선언 : list<[Data Type]> [변수 이름]

l.front() : 맨 앞의 원소 반환
l.back() : 맨 뒤의 원소 반환
l.begin() : 맨 앞의 원소를 가리키는 iterator 반환
l.end() : 맨 뒤의 원소의 다음을 가리키는 iterator 반환
l.push_back(n) : 뒤쪽으로 k 삽입
l.pop_back() : 맨 마지막 원소 제거
l.pop_front() : 맨 첫번째 원소 제거
l.insert(iter, n) : iter가 가리키는 위치에 원소 k 삽입
l.erase(iter) : iterator가 가리키는 원소 삭제
l.size() : 원소의 개수 반환

문법예제

1. vector

```
vector<int> v;

v.push_back(10);
v.push_back(20);
for (vector<int>::iterator iter = v.begin(); iter != v.end(); iter++)
        cout << *iter << " ";
```

2. map

```
string s1, s2;
map<string, string> m;

for (int j = 0; j < 2; j++) {
    cin >> s1 >> s2;
    m.insert(pair<string, string>(s1, s2));
}

for (map<string, string> :: iterator iter = m.begin(); iter != m.end(); iter++)
    cout << iter->first << " : " << iter->second << "₩n";
```

3. list

```
list<int> l;

l.push_back(10);
l.push_back(20);
for (list<int> :: iterator iter = l.begin(); iter != l.end(); iter++) {
     cout << *iter << " ";
}
cout << "₩n";
```

질문

없습니다.