실습 054 - class & map

제출 마감일: 2023-04-09 23:59
업로드 가능한 파일 수: 4
제출 방식: 개인

목적

이 실습은 STL의 map 컨테이너와 class 정의를 연습합니다.

설명

컴퓨터에서 주어진 데이터에서 특정 값을 가지는 데이터를 찾는 문제는 잘 알려져 있다. 데이터에서 빠르게 원하는 값을 검색하기 위해서 주로 사용하는 전략은, 데이터를 저장할 때 빠른 검색을 고려한 특별한 구조를 구축하는 것이다. 대표적인 방법으로 특정 순서로 정렬된 tree 자료구조를 이용하는 방법이 있다.

C++ STL은 이런 문제를 위한 자료구조와 검색에 적합한 map 컨테이너를 제공하고 있다.

map 은 pair <const key, value> 구조로 데이터를 저장한다. 또한, 일반적으로 데이터가 입력되면, 키를 tree 자료구조로 구축하여, key 값으로 value 데이터를 빠르게 가져올 수 있다. 키값을 비교하여 순서대로 정렬되어 있기 때문에 입력된 키값을 수정할 수 없도록 const로 정의되며, 키값은 유일해야 하며, 중복을 허용하지 않아야 하는 제약도 있다.

예를 들어, 다음과 같은 Person 데이터를 map 에 저장해 보자.

Person s1 {"Kim", 22}; Person s2 {"Lee", 21}; Person s3 {"Park", 20};

map<string, Person> persons;

pair<string, Person> data1 = {"Lee", s2}; persons.insert(data1);

pair<string, Person> data2 = {"Park", s3}; persons.insert(data2);

pair<string, Person> data3 = {"Kim", s1}; persons.insert(data3);

//원소를 입력할 때 이미 map에 들어 있는 경우 입력이 실패합니다.

//따라서 다음과 같이 안전하게 입력하는 것이 추천됩니다.

// if (! persons.count("Lee")) persons.insert(data1);

```
for (const auto& d : persons) {
        cout << "Key: " << d.first << ", ";
        cout << "Data: "; d.second.print();
}
```

출력 결과를 보면 키값이 오름차순으로 정렬되어 출력되는 것을 확인할 수 있다.

map 의 원소에 접근하기 위해서 대표적으로 at() 멤버 함수를 이용합니다.

persons.at("Lee"); // out_of_range 예외가 발생할 수 있음

또다른 방법으로는 배열의 인덱스 연산자를 이용해서 원소에 접근할 수 있다.

persons["Lee"]; //"Lee" 키값이 없으면 새로운 원소를 생성해서 입력함

따라서, map 에 있는 원소를 키값으로 접근할 때는 해당 키값이 저장됐는지 여부를 확인하여 진행하는 것이 안전합니다.

map 의 가장 유용한 기능인 키를 이용한 빠른 찾기는 find() 멤버함수를 이용해서 수행할 수 있습니다.

```
auto find_it = peoples.find("Lee");
 if (find_it != peoples.end() )
      cout << "Found!" << '\n';
```

map 에서 원소를 삭제하는 방법은 erase (키값) 멤버함수이며 다음과 같이 사용할 수 있다.

```
auto remove_it = peoples.erase(id);
if (remove_it != peoples.end())
    cout << "Deleted!" << '\n';
```

문제

N 명의 연락처 정보 (이름, 전화번호)가 주어지면, 이 정보를 저장공간에 입력/삭제/검색/수정하는 전화번호부 관리 프로그램을 작성하시오. 단, 연락처 정보는 Person 클래스, 저장공간은 map 컨테이너를 사용하시오.

<stop_words> *stop_words 는 소스 코드에 포함되면 안됩니다.
- stop_words=("printf" "scanf" "cstring" "strcmp" "strlen" "strcpy" "vector")
- for 문을 사용하지 않고 구현해 보는 것을 추천 드립니다.

<참고>
//Person.h

```
class Person {
public:
   Person(std::string name, std::string number);
```

```
    void modifyNumber(std::string number);
    void print() const;

private:
    // your code here
}
```

//Person.cpp (개별 멤버 함수를 구현하시오)

```
Person::Person(std::string name, std::string number) {
    // your code here
 }


void Person::modifyNumber(std::string number) {
    // your code here
 }


void Person::print() const {
    // your code here
 }
```

입력

첫째 줄에 N이 주어진다. (0 <= N <= 100)

둘째 줄부터 N명의 이름과 연락처가 순차적으로 주어진다.
N+2 줄부터 프로그램 명령이 주어진다.
ADD 이름 연락처 : 전화번호부에 연락처를 추가한다.
DEL 이름 : 전화번호부에서 해당 이름를 삭제한다.
MOD 이름 연락처 : 전화번호부에서 해당 이름의 연락처 정보를 수정한다.
FIN 이름 : 전화번호부에서 해당 이름의 연락처 정보를 출력한다.
QUI : 프로그램을 종료한다.

출력

입력받은 명령을 모두 수행 후, 프로그램 종료 시 전화번호부 전체 정보를 한사람씩 한줄에 출력한다.

제출파일

Person.h, Person.cpp, PersonTest.cpp, 54.csv

입출력

입력	출력
1 Kim 1111-1111 ADD Lee 2222-2222 QUI	Kim 1111-1111 Lee 2222-2222
2 Kim 1111-1111 Lee 2222-2222 DEL Kim QUI	Lee 2222-2222
2 Kim 1111-1111 Lee 2222-2222 MOD Kim 3333-3333 QUI	Kim 3333-3333 Lee 2222-2222
2 Kim 1111-1111 Lee 2222-2222 FIN Lee QUI	Lee 2222-2222 Kim 1111-1111 Lee 2222-2222

VPL