1. c++, 시간 복잡도, 정렬

참고 출처 : 나정휘 선배님 ppt
(https://github.com/justiceHui/SSU-SCCC-Study)

목차 (c++)

- cpp를 써야하는 이유
- namespace,
- 입출력{cout, cin, 파일 입출력 },
- 함수(연산자) 오버로딩,
- <vector>
- 범위 기반 for
- 레퍼런스
- <string>
- <algorithm>
- template

주의!

- 지금부터 할 c++ 강의는 문제 풀이에 집중한 c++ 강의입니다. 따라서 내용을 요약, 생략하는 것들이 많으므로 c++ 알고 있다고 말하려면 따로 공부하셔야 합니다!

- 추천 사이트

https://modoocode.com/135 (cpp의 전체적인 문법)

https://cppreference.com/ (cpp의 함수 찾기에 좋음)

C++를 써야하는 이유

- C 에서는 지원 안 하는 편리한 라이브러리가 정말 많음.

문제

정수를 저장하는 큐를 구현한 다음, 입력으로 주어지는 명령을 처리하는 프로그램을 작성하시오.

명령은 총 여섯 가지이다.

- push X: 정수 X를 큐에 넣는 연산이다.
- pop: 큐에서 가장 앞에 있는 정수를 빼고, 그 수를 출력한다. 만약 큐에 들어있는 정수가 없는 경우에는 -1을 출력한다.
- size: 큐에 들어있는 정수의 개수를 출력한다.
- empty: 큐가 비어있으면 1, 아니면 0을 출력한다.
- front: 큐의 가장 앞에 있는 정수를 출력한다. 만약 큐에 들어있는 정수가 없는 경우에는 -1을 출력한다.
- back: 큐의 가장 뒤에 있는 정수를 출력한다. 만약 큐에 들어있는 정수가 없는 경우에는 -1을 출력한다.

입력

첫째 줄에 주어지는 명령의 수 N (1 ≤ N ≤ 10,000)이 주어진다. 둘째 줄부터 N개의 줄에는 명령이 하나씩 주어진다. 주어지는 정수는 1보다 크거나 같고, 100,000보다 작거나 같다. 문제에 나와있지 않은 명령이 주어지는 경우는 없다.

출력

출력해야하는 명령이 주어질 때마다, 한 줄에 하나씩 출력한다.

namespace(이름 공간)

- C++의 표준 헤더 파일의 모든 함수, 클래스, 객체는 std라는 namespace라는 공간 안에 있다.

E.g.) <iostream>, <queue>

예외) <cstdio>

- 따라서 std안에 있는 것들을 쓰려면 std::{something}과 같이 std::를 앞에 붙여야 합니다.

C++에서 입출력

- <iostream>이 C에서의 <stdio.h>의 역할을 하는 헤더파일.

- 입출력 방법 :

입력 : std::cin >> {변수};
출력 : std::cout << {식};

- 예시+ :

https://www.acmicpc.net/problem/1000 (입출력 연습)
https://www.acmicpc.net/problem/10951 (cin 특징)
https://www.acmicpc.net/problem/1008 (소수점 출력)
https://www.acmicpc.net/problem/15552 (빠른 입출력)

함수(연산자) 오버로딩

- C++에서는 다음과 같은 코드가 가능함.

```cpp
#include <iostream>

using namespace std;

void print(int a) {
    cout << a << '₩n';
    return;
}

void print(double a) {
    cout << a << '₩n';
    return;
}

int main() {

    print(3);
    print(3.14);

    return 0;
}
```

함수(연산자) 오버로딩

- 함수(연산자) 오버로딩 덕분에
  cout << “hello” << 3 << 12.1234 << ‘₩n’; 이 가능하다.

- 연산자 정의하기

https://www.acmicpc.net/problem/11650 (비교 연산자 정의)

vector

- 동적 배열; c언어의 배열과 다르게 길이가 마구 변할 수 있다.

```cpp
#include <iostream>
#include <vector>
using namespace std;

int main(){
    // 원소의 타입을 지정해야 함
    // vector<int>는 int형 데이터를 저장하는 벡터
    // vector<double>은 double형 데이터를 저장하는 벡터

    vector<int> empty_vector; // 빈 벡터 선언
    vector<int> sized_vector(5); // 크기를 미리 지정할 수 있음, 0으로 초기화
    vector<int> init_vector(5, 1); // 초기값을 지정할 수 있음, 모두 1로 초기화
    vector<int> init_list_vector = {1, 2, 3, 4, 5}; // 이렇게 초기화할 수도 있음

    vector<int> v;
    for(int i=0; i<5; i++) v.push_back(i * 2); // 맨 뒤에 삽입
    for(int i=0; i<5; i++) cout << v[i] << "\n"; // 인덱스 접근 가능
    v.pop_back(); // 맨 뒤 원소 삭제
    cout << v.size() << " " << v.empty() << "\n"; // 원소의 개수, 비어있는지 확인
    cout << v.front() << " " << v.back() << "\n"; // 맨 앞/뒤 원소

    v.clear(); // 모든 원소 삭제
    cout << v.size() << " " << v.empty() << "\n";
}
```

범위 기반 for

- C++에서 for의 다른 사용법; vector같은 자료구조 전체 순회할 때 편하게 할 수 있다.
- 사용법 : for ({자료형} {변수명} : {자료구조}) {}
- {변수명}에 자료구조에 담겨 있던 값이 하나씩 할당 됨.

래퍼런스

- C든 C++이든 함수 메개변수로 변수를 전달하면 Call by Value 로 값이 "복사"되어서 전달 됨.

```
#include <iostream>

using namespace std;

void print(int a) {
    cout << &a << " : " << a << '\n';
    return;
}

int main() {
    int a = 4;
    cout << &a << " : " << a << '\n';
    print(a);

    return 0;
}
```

결과 :

```
00000023E670F7A4 : 4
00000023E670F780 : 4
```

래퍼런스

- 만약 vector와 같이 크기가 매우 큰 자료를 매개변수로 넘긴다면?
  실행 시간이 매우 매우 매우 많이 들게 됨.

- 따라서 call by reference가 필요함.

래퍼런스

- 사용법 : {자료형}& {변수명} = {변수}; //초기화 무조건 해야함.

```cpp
#include <iostream>

using namespace std;

void print(int& a) {
    cout << &a << " : " << a << '₩n';
    return;
}

int main() {
    int a = 4;
    cout << &a << " : " << a << '₩n';
    print(a);

    return 0;
}
```

결과 :

```
000000CE241BF8D4 : 4
000000CE241BF8D4 : 4
```

<string>

- <string> : 문자열을 쉽게 다룰 수 있게 해주는 C++ 표준 헤더 파일

- strcmp, strcat, strlen 등등 보다 훨씬 직관적임.

- 예

https://www.acmicpc.net/problem/9086

```cpp
#include <iostream>
#include <string>

int main(){
    std::string a = "12", b = "34";
    std::cout << a + b << "\n";    // 1234
    a += b;
    b = "56";
    std::cout << a + b << "\n";    // 123456
    std::cout << (a == b) << "\n"; // 0
    b = "1234";
    std::cout << (a == b) << "\n"; // 1
    std::cout << a.size() << "\n"; // 4

    std::cout << a[0] << " ";
    std::cout << a.front() << " ";
    std::cout << a[a.size()-1] << " ";
    std::cout << a.back() << "\n";
    for(int i=0; i<a.length(); i++) std::cout << a[i];
    std::cout << "\n";

    a.clear();
    std::cout << a.size() << " " << a.empty() << "\n";
}
```

<algorithm>

- 여러가지 유용한 알고리즘들이 정의 되어 있음.

  예를 들면, 이진 탐색, 최대/최소 값 찾기, 정렬, 순열 찾기, swap, 뒤집기 등등등...

- https://modoocode.com/256 에서 한 번 둘러보면 좋음.

- https://www.acmicpc.net/problem/10974 (순열 찾기)

시간 복잡도

- 정휘 선배님 ppt에서!

정렬

- 정렬은 순서대로 값을 나열하는 것.
  정확히 따지면 오름차순, 내림차순 정렬이 있음.

- 오름차순은 수열에 어떤 값 a, b가 있고, a<b가 만족하면 a는 무조건 b 앞에 위치해야 한다.
- 내림차순은 오름차순 정반대.

정렬

- 정렬 하는 방법은 퀵 소트, 머지 소트, 버블 소트 등등 여러가지가 있지만 문제 풀이에서 쓰이는 정렬은?

- std::sort( ) (https://www.acmicpc.net/problem/16466) //대부분 이것만 써도 ok
- Count sort (https://www.acmicpc.net/problem/10989)
- 직접 구현 하게 되는 경우 (https://www.acmicpc.net/problem/1517)

정렬한 배열이 있으면?

- 토요일 강의에서 이진 탐색과 함께 이야기 하겠습니다.

- 숙제는
- https://www.acmicpc.net/problem/1427
- ...