

바킹독 0x02

≡ 태그	
<input checked="" type="checkbox"/> 공개여부	<input checked="" type="checkbox"/>
<input type="text"/> 날짜	
<input type="text"/> 작성일자	

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    ios::sync_with_stdio(0)
    cin.tie(0)
}
```

함수인자와 STL

0x00 STL과 함수 인자

함수인자

```
01 void func(int a){
02     a = 5;
03 }
04 int main(void) {
05     int t = 0;
06     func(t);
07     cout << t;
08 }
```

0

```
01 void func(int arr[]){
02     arr[0] = 10;
03 }
04 int main(void) {
05     int arr[3] = {1,2,3};
06     func(arr);
07     cout << arr[0];
08 }
```

10

```
01 struct pt{
02     int x,y;
03 };
04 void func(pt a){
05     a.x = 10;
06 }
07 int main(void) {
08     pt tmp = {0,0};
09     func(tmp);
10     cout << tmp.x;
11 }
```

0

3

1. 함수 인자로 값을 복사해서 넘겨주는 경우(call by value) : 원본의 값이 바뀌지 X, t값을 func 함수의 a라는 새 변수에 복사해서 넘겨줌.

ex)int를 넘김, 구조체를 넘김, STL을 그냥 넘김


2. 함수 인자로 주소를 넘겨주는 경우(call by reference) : 원본의 값이 바뀜

ex) int 배열 arr의 주소(배열명)를, 포인터 변수를, reference variable을...

C++의 경우 참조자(reference)를 이용한 함수 작성 가능

[C++ 기본 공부정리] 12. 참조 변수(reference variable)

공부 내용을 정리하는 목적 이므로 참고용으로만 읽어 주시기 바랍니다. 틀린 부분에 대한 지적은 감사합니다. 일반 변수 : 값을 저장하기 위해 메모리에 공간을 할당받아 직접 저장하는 변수 포인터 변수 : 다

 <https://min-zero.tistory.com/entry/C-기본-공부정리-12-참조-변수reference-variable>

```
main() {  
    int num = 10;  
    int& refer = num;  
  
    cout << "일반 변수 호출: " << num << endl;  
    cout << "참조 변수 호출: " << refer << endl;  
  
    return 0;  
}
```

/visual Studio 디버그 콘솔

```
num: 10  
refer: 10
```

```
void swap(int& a, int& b) // *reference variable을 매개변수로*  
{  
    int tmp = a;  
    a = b;  
    b = tmp;  
}  
  
int main()  
{  
    int ra = 1; int rb = 2;  
    swap(ra,rb) //ra와 rb(원본)값이 바뀜  
    return 0;  
}
```

STL(Standard Template Library)?

STL은 C++에서 제공되는 라이브러리로, 필요한 자료구조들을 직접 구현할 필요 없이 가져다 쓸 수 있게 해줌.

ex)배열과 비슷한 기능을 하는 vector STL : 일종의 가변 배열로, 배열의 크기를 마음대로 늘렸다 줄였다 할 수 있음

STL을 뺏으로 함수 인자에 넣으면 복사해서 보낸다는 것에 주의! ->하나하나 복사한다는 점에서 시간복잡도 커질수도

0x00 STL과 함수 인자

STL을 함수 인자로 넘길 때

```
01 void func1(vector<int> v) {  
02     v[10] = 7;  
03 }  
04 int main(void) {  
05     vector<int> v(100);  
06     func1(v);  
07     cout << v[10];  
08 }
```

0

6

0x00 STL과 함수 인자

STL을 함수 인자로 넘길 때

```
01 bool cmp1(vector<int> v1, vector<int> v2, int idx) {  
02     return v1[idx] > v2[idx];  
03 }
```

7

그냥 STL을 썬으로 함수 인자에 넣으면 복사해서 보낸다는걸 꼭 유의하셔야 합니다. 이 사실을 머릿속에 넣어두고 cmp1 함수를 한 번 확인해봅시다.

이 함수의 시간복잡도는 충격적이게도 $O(N)$ 이 됩니다. 아니 함수 안에 연산을 딱 1번만 하는데 $O(N)$ 이라는게 무슨 말도 안되는

소리냐라고 생각이 들 수 있지만 v1, v2를 인자로 실어서 보낼 때 원본으로부터 복사본을 만드는 비용을 생각하지 못하고 계산

것입니다. v1, v2의 크기가 N이니까 N개의 원소들을 하나하나 복사하는 과정은 $O(N)$ 이 듭

니다. 그래서 이 함수는 의도하지 않게 시간복잡도가 $O(N)$ 이 됩니다.

그냥 idx번째 원소의 값만 비교하고 싶은데 매번 vector를 복사하는건 정말 말이 안되는 일입니다. **이럴 때 참조자를 이용하면 됩니다.** cmp2를 확인해보세요.

0x00 STL과 함수 인자

STL을 함수 인자로 넘길 때

```
01 bool cmp1(vector<int> v1, vector<int> v2, int idx){
02     return v1[idx] > v2[idx];
03 }
```

```
01 bool cmp2(vector<int>& v1, vector<int>& v2, int idx){
02     return v1[idx] > v2[idx];
03 }
```

7

따라서 참조자를 이용해 함수 인자로 넘긴다. ->복사X, 주소가 넘어감

표준 입출력

1. 문자열 처리할 때, C++ string 사용 가능 *cin/cout 사용할 경우임
2. cin은 공백 앞까지만 입력 받으므로 공백을 포함한 문자열을 입력받을 때 주의
(해결법 : type이 C++ string 한정일때, getline 이용)

```
string s;
getline(cin, s);
cout << s;
```

3. cin/cout을 사용할 때, 입출력량이 많을 때 시간초과를 막기 위한 두가지 명령

ios::sync_with_stdio(0)

C++ stream과 C stream의 동기화를 끄는 명령.

cout과 printf 섞어쓰기 불가(출력순서 꼬임) 여기서 0은 false와 같음.

cin.tie(0)

cout 버퍼를 비우지 않도록 하는 코드.

기본적으로는 cin명령을 수행하기 전에 cout 버퍼를 비우는데,

온라인 저지 사이트에서는 채점을 할 때 출력 글자만 확인하므로, 출력 글자 사이사이가 꼬여도(입력-출력간의 순서 상관X, 출력물끼리의 순서와 출력 글자내용만 확인) 채점에 영향X이기 때문.

여기서 0은 nullptr을 의미.

코드 작성 Tip

1. #include <bits/stdc++.h> 사용

제한된 시간 안에 정답을 받아야 하는 코딩테스트에서의 효율을 위해...^^

2. 출력 맨 마지막에 공백 혹은 줄바꿈이 추가로 있어도 정답처리되므로 별도의 예외처리는 필요하지 않음

3. $x \& 1$

비트연산자 AND

x가 홀수인지 짝수인지 확인할 때 사용 가능 ($x \% 2$ 와 유사)

x가 홀수일때 true/1를 반환, x가 짝수일때 false/0을 반환

cf. $x \gg 1$ 은 오른쪽으로 1만큼 시프트라는 비트연산자로, $x / 2$ 와 유사

4. 코딩테스트의 목표는 남이 알아볼 수 있는 클린코드를 작성하는게 아닙니다. 어떻게든 제한된 시간 안에 정답을 받아야합니다. 그렇기 때문에 코드를 거의 책에 예제로 실어도

될 정도로 깔끔하게 만들기 위해 노력하기 보다는, 좀 더럽더라도 내가 빠르게 짤 수 있는 방식으로 빠르게 구현하는게 훨씬 더 중요합니다.