

바둑킹의 알고리즘 2

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

참조자(Reference)

```
01 void swap1(int a, int b){
02     int tmp = a;
03     a = b;
04     b = tmp;
05 }
```



```
01 void swap2(int* a, int* b){
02     int tmp = *a;
03     *a = *b;
04     *b = tmp;
05 }
```



```
01 void swap3(int& a, int& b){
02     int tmp = a;
03     a = b;
04     b = tmp;
05 }
```



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

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

$O(N)$

N개의 원소들을 하나하나 복사하는 과정은 $O(N)$ 이 들겠죠. 그래서

6:02 / 18:39 • 0x00 STL과 함수 인자 >

$O(N)$

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

$O(1)$

```
01 int main(void) {
02     char a[10];
03     printf("input : ");
04     scanf("%s", a);
05     printf("a is %s\n", a);
06 }
07 /**result**
08 input : hi hello
09 a is hi
10 *****/
```

```
01 int main(void) {
02     string s;
03     cout << "input : ";
04     cin >> s;
05     cout << "s is " << s;
06 }
07 /**result**
08 input : hi hello
09 s is hi
10 *****/
```

```
01 // 1. scanf의 옵션
02 char a1[10];
03 scanf("%[^\\n]", a1);
04
05 // 2. gets 함수(보안상의 이유로 C++14 이상에서는 제거됨)
06 char a2[10];
07 gets(a2);
08 puts(a2);
09
10 // 3. getline 함수
11 string s;
12 getline(cin, s);
13 cout << s;
```

공백이 포함된 문자열을 받을 때 단순히 scanf or cin을 쓰면 안된다.

`ios::sync_with_stdio(0), cin.tie(0)`

cin, cout 쓸때 위에 꺼 써야한다. 왼쪽에 있는거는 c++이랑 c랑 같이 동시에 동기화 하는 것을 막아주는 것이고 오른쪽에 있는것은 온라인 백준저지에서 버퍼를 비워주는 것을 안해 줘도 결과가 잘 나와서 버퍼를 안 비워주는 것을 나타낸다.

endl은 절대 쓰지 마시오!! endl은 \n을 출력하고 버퍼를 비워주는 명령어이다

코딩테스트 코드 작성 팁

1.코딩테스트와 개발은 다르다.

클린 코딩보다 좀 더럽더라도 빠르게 구현할 수 있는 코드를 짜야 한다.

2.출력 맨 마지막 공백 혹은 줄바꿈이 있어도 상관없다.

3.디버거는 굳이 사용하지 않아도 된다.