2017 SCSC Workshop

Sogang univ.
System modeling & Optimization Lab
Sangdurk Han

Contents

- Introduction to programming contest
- Full Search and Time complexity
- STL

Competitive Programming

- 논리적 또는 수학적 문제를 컴퓨터를 이용하여 해결하는 프로그래밍
- 다양한 자료구조와 알고리즘을 이용한다.

어디에 쓰일까?

- KOI
- Application 제작
- Job interview!

어떻게 연습할까?

- Online Judge
- http://acmicpc.net
- http://algospot.com/
- http://www.codeforces.com/

어떻게 연습할까?

- 다양한 프로그래밍 대회 참가
  - Facebook hackercup
  - google codejam
  - acm-icpc
  - topcoder open

무엇을 배우면 될까?

- Math
- Data Structure
- Greedy
- Dynamic Programming
- Backtracking
- Algorithm Game
- Network Flow …

Online 대회

- Topcoder single round match
- COCI

자주하는 실수를 알아보자

```cpp
#include <cstdio>
int main() {
    int sum;
    int n = 10;
    for ( int i = 0 ; i < n ; i++ ) {
        sum += i;
    }
    printf("%d\n",sum);
    return 0;
}
```

```cpp
#include <cstdio>
int main() {
    int mul = 1;
    int n = 15;
    for ( int i = 1 ; i <= n ; i++ ) {
        mul *= i;
    }
    printf("%d\n",mul);
    return 0;
}
```

```cpp
#include <cstdio>
int main() {
    int a[10];
    int n = 15;
    for ( int i = 1 ; i <= n ; i++ )
        a[i] =i*i;
    return 0;
}
```

```
#include <cstdio>
int main() {
    int a[10];
    int n = 15;
    for ( int i = 1 ; i <= n ; i++ );
        a[i] =i*i;
    return 0;
}
```

```cpp
#include <cstdio>
int d[100][100];
int main() {
    int n = 100;
    for ( int i = 0 ; i < 100 ; i++ ) {
        for ( int i = 0 ; i < 100 ; i++ ) {
        }
    }
    return 0;
}
```

```c
#include <cstdio>
int d[100][100];
int main() {
    int n;
    scanf("%d",&n);
    switch(n%2) {
        case 0:
            printf("even\n");
        case 1:
            printf("odd\n");
        default:
            printf("what the fu..?\n");
    }
    return 0;
}
```

```cpp
#include <cstdio>
#define min(a,b) a<b?a:b
#define square(a) a*a
int main() {
    int n,m;
    scanf("%d%d",&n,&m);
    for ( int i = 0 ; i < n ; i++ )
        for ( int j = 0 ; j < n ; j++ ) {
            printf("%d %d %d\n",i+j,i-j,min(i+j,i-j));
            printf("%d %d\n",(i+j)*(i+j), square(i+j));
        }
    return 0;
}
```

```c
#include <cstdio>
#define min(a,b) (((a)<(b))?(a):(b))
int main() {
    int x = 10;
    int y = 20;
    int z = min(x++,y++);
    printf("%d %d %d\n",x,y,z);
    return 0;
}
```

```
#include <cstdio>
#include <cmath>
int main() {
    double angle;
    scanf("%lf",&angle);
    printf("%lf\n",cos(angle));
    return 0;
}
```

```cpp
#include <cstdio>
int main() {
    float a = 0.000001;
    float b = 0.000001;
    a = a * 1000000;
    for ( int i = 0 ; i < 1000000 ; i++ ) {
        b += 0.000001;
    }
    printf("%lf %lf\n",a,b);
    if ( a == b ) {
        printf("is same\n");
    } else {
        printf("is not same\n");
    }
    return 0;
}
```

```cpp
#include <iostream>
using namespace std;
int d[1000000];
int main() {
	int n = 1000000;
	for ( int i = 0 ; i < n ; i++ ) {
		cin >> d[i];
	}
	return 0;
}
```

```cpp
#include <cstdio>
#include <cstring>
using namespace std;
char a[100] = "My Name is I Don't Know";
int main() {
    for ( int i = 0 ; i < strlen(a) ; i++ ) {
        printf("%c\n",a[i]);
    }
    return 0;
}
```

```cpp
#include <cstdio>
#include <vector>
using namespace std;
vector<int> input() {
    int n;
    scanf("%d",&n);
    vector<int> a(n);
    for ( int i = 0 ; i < n ; i++ )
        scanf("%d",&a[i]);
    return a;
}
int main() {
    vector<int> a = input();
    for ( int i = 0 ; i < a.size() -1 ; i++ ) {
        if ( a[i] != a[i+1] ) {
            printf("What?\n");
        }
    }
    return 0;
}
```

```cpp
#include <cstdio>
int a = 1;
int b[a];
int main() {
    b[a--] = 5;
    printf("%d\n",b[a]);
    return 0;
}
```

```cpp
#include <cstdio>
const int a = 1;
int b[a];
int main() {
    b[a--] = 5;
    printf("%d\n",b[a]);
    return 0;
}
```

```cpp
#include <cstdio>
const int a = 1;
int b[a];
int main() {
    int c = a;
    b[c--] =5;
    printf("%d\n",b[c]);
    return 0;
}
```

```
#include <cstdio>
#define MAX_LOOP (1<<10)
int a[MAX_LOOP+1];
int main() {
    int i;
    for (aint i = 0 ; i < MAX_LOOP ; i++ ) ;
    a[i] = 12345;
    printf("%d\n",a[i]);
    return 0;
}
```

```cpp
#include <cstdio>
#define MAX_LOOP (1<<10)
int a[MAX_LOOP+1];
int main() {
    int i;
    for (aint i = 0 ; i < MAX_LOOP ; i++ );
    a[i] = 12345;
    printf("%d\n",a[i]);
    return 0;
}
```

```cpp
#include <cstdio>
#define MAX_LOOP (1<<30)
int a[MAX_LOOP+1];
int main() {
    int i;
    for (aint i = 0 ; i < MAX_LOOP ; i++ ) ;
    a[i] = 12345;
    printf("%d\n",a[i]);
    return 0;
}
```

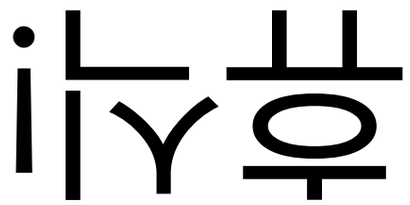
```cpp
#include <cstdio>
int main() {
    for ( int i = 0 ; i < 100000000 ; i++ );
    return 0;
}
```

```cpp
#include <cstdio>
#include <string>
using namespace std;
int main() {
    string s = "sangkeun is good!\n";
    for ( int i = 0 ; i < s.length() ; i++ )
        printf("%c",s[i]);
    puts("");
    return 0;
}
```

```
#include <cstdio>
bool isPrime(long long a) {
    for ( int i = 2 ; i < a ; i++ )
        if ( !(a%i) ) return false;
    return true;
}
int main() {
    long long a;
    scanf("%lld",&a);
    printf("%s\n",isPrime(a)?"is prime!":"is
not prime");
    return 0;
}
```

```
#include <cstdio>
typedef long long ll;
bool isPrime(long long a) {
    for ( int i = 2 ; i*i <= a ; i++ )
        if ( !(a%i) ) return false;
    return true;
}
int main() {
    long long a;
    scanf("%lld",&a);
    printf("%s\n",isPrime(a)?"is prime!":"is
not prime");
    return 0;
}
```

```cpp
#include <cstdio>
typedef long long ll;
bool isPrime(long long a) {
    for ( ll i = 2 ; i*i <= a ; i++ )
        if ( !(a%i) ) return false;
    return true;
}
int main() {
    long long a;
    scanf("%lld",&a);
    printf("%s\n",isPrime(a)?"is prime!":"is
not prime");
    return 0;
}
```



이번엔 안쉬어요

영어를 잘하면 좋다.

- 많은 정보가 영어로 되어있다.

- 여러분은 영어를 잘하니까 상관없어요.

영어를 잘하면 좋다.

- 많은 정보가 영어로 되어있다.
- 여러분은 영어를 잘하니까 상관없어요.
- 내가 공부해야되요 ...

컴퓨터를 많이 할 필요는 없다.

- 다양한 문제가 주어졌을 때, 어떻게 해결하는지 컴퓨터적으로 생각할 수만 있으면 된다.
- Application 을 만들 때에는 대부분 예전에 작성한 소스나 아름다운 소스를 복사 붙여넣기해서 만든다.
- 손목 터널 증후군에 걸릴 수 있다.
- 많이하면 눈이 나빠진다.

난 못하니까 처음부터 공부해야지!

```
#include <cstdio>
#include <algorithm>
using namespace std;
#define swap4(x,y) int t;t=x;x=y;y=t;
#define swap5(x,y) x^=y;y^=x;x^=y;
void swap2(int *x,int *y) {
    int z = *x; *x = *y; *y = z;
}
void swap3(int &x,int &y) {
    int z = x; x = y; y = z;
}
int main() {
    int a = 10, b = 20;
    swap(a,b); swap2(&a,&b); swap3(a,b);
    swap4(a,b); swap5(a,b);
    return 0;
}
```

책은 신기술을 못따라간다.

- 대부분의 정보는 구글링을 통해서 얻도록 한다.
- 문법을 외울 필요는 없다. 개발을 하면서 익숙해지는 것이다.

앱을 만들고 싶다.

- 앱은 생각보다 만들기 쉽다.
- 객체 지향은 앱을 만들면서 배우는 것이 더 쉽다.
- 알고리즘과 자료구조의 기초가 잘 쌓였으면 앱은 만들기 쉽다.
- 인터넷 검색을 많이 해야 한다.

정말 시작!

Question

- N장의 종이가 들어간 종이상자에서 4번을 뽑아서 숫자의합이 m이면 승리한다. 승리할 수 있을까? YES or NO

- 종이에 적힌 수 : $K_1, K_2, \dots, K_n$

$$1 \le N \le 1000$$
$$1 \le m \le 10^8$$
$$1 \le K_i \le 10^8$$

Simple solution

```
bool go(int *K,int N,int m) {
    for ( int i = 0 ; i < N ; i++ )
        for ( int j = 0 ; j < N ; j++ )
            for ( int k = 0 ; k < N ; k++ )
                for ( int l = 0 ; l < N ; l++ )
                    if ( K[i] + K[j] + K[k] + K[l] == m )
                        return true;
    return false;
}
```

Computational complexity
계산량

- 무엇에 비례할까 ? = Order

$$N \times N \times N \times N = O(N^4)$$

실행 시간

- 계산량에 비례
- $1초 = 100,000,000 = 10^8$
- $O(N^4) > O(1000^4) > O(10^{12})$: 10000초
  = 166분
  = 2시간

수업시간은 .. ?

Thinking

소주병뚜껑 게임

소주병뚜껑 게임

- 신기하게도 소주 병뚜껑에는 숫자가 적혀있다. (1~50)
- 게임 시작자가 혼자 병뚜껑 숫자를 본다.
- 시작자를 제외한 사람이 차례로 숫자를 부르고, 시작자는 Up, Down을 말해준다. 제한 턴(6번) 안에 숫자를 맞추면 시작자가 ~~마시고~~ 턴을 넘기면 맞춘 사람이 ~~마신다~~.
- 필승 전략이 존재한다(!)

소주병뚜껑 게임

- 매번 절반을 부르면 어떻게 될까?
- 병뚜껑 숫자는 41, 앞으로 봐야할 숫자의 범위는 [A,B]
- [1~50] 25! (up) > [26~50] 38! (up) > [39~50] 44! (down) > [39~43] 41 (정답!)
- 매번 절반의 경우가 사라짐!

$$O(\log_2(N)) = O(\log_2(50)) = O(7)$$

Better solution

- 소주병뚜껑 게임을 접목

  (소주병뚜껑 게임 = Binary search)

- **Binary search는 정렬 필요**

$$sort : O(nlog(n))$$

$$loop : O(n^3 logn)$$

$$total : O(n^3 \log(n)) = O(10^9 * c)$$

Better solution

```
bool go(int *K /*sorted*/,int N,int m) {
    for ( int i = 0 ; i < N ; i++ )
        for ( int j = 0 ; j < N ; j++ )
            for ( int k = 0 ; k < N ; k++ ) {
                int le = 0,ri = N-1;
                while ( le <= ri ) {
                    int mid = (le+ri)/2;
                    if ( K[mid] == m-K[i]-K[j]-K[k] ){
                        return true;
                    } else if ( K[mid] > m-K[i]-K[j]-K[k] ){
                        ri = mid - 1;
                    } else {
                        le = mid + 1;
                    }
                }
            }
    return false;
}
```

More better solution

- 만들 수 있는 모든 Pair를 두 쌍만든다.
- Pair 하나를 정렬하고, 다른 Pair를 돌면서 binary search

$$sort : O(n^2 logn)$$

$$loop : O(n^2 logn)$$

$$total : O(n^2 \log(n)) = O(10^6 * c)$$

Question

- 길이가 L cm 인 장대 위에 n마리의 개미가 초당 1cm로 걷고 있다. 개미가 장대 끝에 도착하면 떨어진다.

- 개미가 서로 만나면 반대 방향으로 움직인다.

- 모든 개미의 현재 위치를 알고 있다. (x_i)
- 방향은 모른다.
- 개미가 떨어지는 최소 시간, 최대시간은 ?

$$(1 \leq L, n \leq 10^6), (1 \leq x_i \leq L)$$

https://www.acmicpc.net/problem/4307

Simple solution

- 모든 개미의 방향 : 왼쪽, 오른쪽
- 모든 개미마다 경우의 수가 2개이므로 나올 수 있는 모든 경우의 수는?

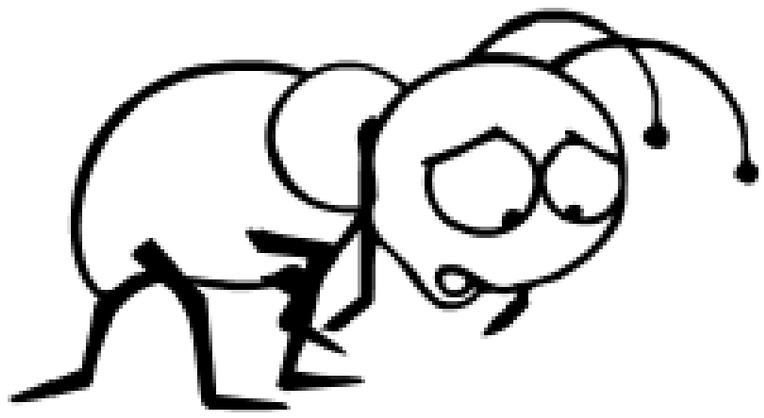
$$2 * 2 * 2 * \dots * 2 = 2^n = O(2^n)$$

$$O(2^{10^6}) \cong O(10^{30})$$

Thinking

Better solution

- O(n)



Quick STL Tutorial

- STL (Standard Template Library)
- C++이지만 C같은 C아닌 C++
- 자주 쓰이는 자료구조 및 알고리즘이 구현되어 있다.

STL 종류

- iostream, bitset, deque, list, map, queue, set, stack, vector, algorithm, functional, iterator, limits, regex, string, tuple, unordered_map, unordered_set, etc …

- 굉장히 많죠? 하지만!

필요한 STL 종류

- iostream, map, queue, set, stack vector, algorithm, string
,

- 이것만 알아도 충분합니다. 중요도 순으로 정렬하면?

공부할 STL 종류

- iostream, algorithm, string, map, stack, queue, set, vector

- algorithm : 저어어엉말 빠른 정렬, 이분탐색

- string : 캐릭터 배열이 아닌 정말 문자열

- map, set : 레드 블랙 트리로 만들어진 굉장한 자료구조

- vector, stack, queue, iostream : 여러분을 편하게 만들어 줄 친구들

STL의 기초

- STL 각각은 헤더파일(.h 안붙음)
- using namespace std; 를 꼭 헤더 선언 밑에 넣어줘야함

```
#include <cstdio>
#include <algorithm>
#include <string>
#include <map>
#include <stack>
#include <queue>
#include <set>
#include <vector>
#include <iostream>
using namespace std;
```

iostream

- C언어의 stdio.h 와 유사하다.
- cin 이 scanf의 기능을 담당하는데, 기능이 더 많아지고 편리해진 대신 굉장히 느리다.
- cout이 printf의 기능을 담당하는데, 기능이 더 많아지고 편리해진 대신 굉장히 느리다.
- **입출력이 많은 문제는 절대 cin, cout을 쓰지 않는다**
- **특히 *endl* 이 매우 느림**

iostream

```
#include <iostream>
using namespace std;
int main() {
    int a,b;
    cin >> a >> b;
    cout << a << b << endl;
    return 0;
}
```

algorithm

- sort : intro sort로 quick sort보다 안정적으로 빠르다.
- binary_search : 이분 탐색(직접 작성하는 것보다 느리다.)
- lower_bound : 이분 탐색 중 중복된 key값의 처음 주소 반환
- upper_bound : 이분 탐색 중 중복된 key값의 마지막 다음 주소 반환

https://www.acmicpc.net/problem/2750

algorithm

```
#include <cstdio>
#include <algorithm>
using namespace std;
int main() {
    const int n = 10;
    int a[n]= {1,5,2,3,1,5,5,6,7,9};
    sort(a,a+n);
    for ( int i = 0 ; i < n ; i++ )
        printf("%d\n",a[i]);
    if ( binary_search(a,a+n,5) ) {
        printf("5 is exists\n");
    } else {
        printf("5 is not exists\n");
    }
    printf("%d %d\n",int(lower_bound(a,a+n,5)-a),
            int(upper_bound(a,a+n,5)-a));
    return 0;
}
```

string

- 그동안의 캐릭터 배열을 더이상 쓰지 않아도 된다(!!)
- strcmp, strcat, strcpy 등을 쓰지 않아도 된다.
- 입출력은 iostream의 cin, cout을 쓴다.
- **다음 예제 장에서는 printf를 사용할 때의 예를 보여준다.**

string

```
#include <cstdio>
#include <string>
#include <iostream>
using namespace std;
int main() {
    string s1 = "hello";
    string s2 = " world ";
    s2 += "!!!!!!";
    if ( s1 != s2 ) {
        printf("%s\n",(s1+s2).c_str());
    }
    string s3;
    cin >> s3;
    cout << s3 << endl;
    return 0;
}
```

map

- Key, Value 쌍으로 구성된 배열 같지만 배열 아닌 친구
- 특정 Key 에 해당하는 저장공간에 Value를 저장할 수 있다.
- insert, read, update 모두 O(log n)이다.
- 다음 장의 예는 이름(Key), 나이(Value) 를 쌍으로 저장하는 예제이다.

https://www.acmicpc.net/problem/1620

map

```cpp
#include <cstdio>
#include <map>
#include <string>
using namespace std;
int main() {
	map<string,int> mp;
	mp["sangkeun"] = 27;
	mp["taewan"] = 23;
	printf("%d %d\n",mp["sangkeun"],mp["taewan"]);
	for ( map<string,int>::iterator it=mp.begin();
		it!=mp.end();it++ )
	printf("%s %d\n",it->first.c_str(),it->second);
	return 0;
}
```

first = “sangkeun”, “taewan”
second = 27, 23

first값이 작은 순으로 출력

stack

stack

- 가장 위에 올려져 있는 것을 뽑을 수 있는 구조
- 대표적으로 괄호 문제와 히스토그램 문제에 쓰인다.
- 재귀함수를 iterative로 구현하기 위해 쓰인다.

https://www.acmicpc.net/problem/10828

stack

```
#include <cstdio>
#include <stack>
using namespace std;
int main() {
	stack<int> st;
	for ( int i = 0 ; i < 10 ; i++ )
		st.push(i);
	while ( !st.empty() ) {
		int tp = st.top();
		st.pop();
		printf("%d\n",tp);
	}
	return 0;
}
```

queue

queue

- 먼저 줄 선 사람이 먼저 들어가는 구조
- 탐색 알고리즘, 스케쥴링 등 다양한 곳에 쓰인다.
- 다음에 할 BFS에서 쓰이는데, 이 BFS는 최단 경로, 싸이클 체크, 최대 유량 알고리즘 등 여러 곳에 쓰인다.
- **priority_queue 라는 heap 구조가 안에 숨어있다!**

https://www.acmicpc.net/problem/10845

queue

```
#include <cstdio>
#include <queue>
using namespace std;
int main() {
	queue<int> q;
	for ( int i = 0 ; i < 10 ; i++ )
		q.push(i);
	while ( !q.empty() ) {
		int frt = q.front();
		q.pop();
		printf("%d\n",frt);
	}
	return 0;
}
```

priority_queue

```cpp
#include <cstdio>
#include <queue>
using namespace std;
int main() {
    priority_queue<int> mxpq;
    priority_queue<int,vector<int>,greater<int> > mnpq;
    for ( int i = 0 ; i < 10 ; i++ ) {
        mnpq.push(i);
        mxpq.push(i);
    }
    while ( !mnpq.empty() ) {
        int tp=mnpq.top();mnpq.pop();
        printf("%d\n",tp);
    }
    while ( !mxpq.empty() ) {
        int tp=mxpq.top();mxpq.pop();
        printf("%d\n",tp);
    }
    return 0;
}
```

set

- 집합 (중복된 원소 없음)
- 집합 안에 값의 존재 유무를 $O(\log n)$ 만에 가능

  (이유는 이분 탐색과 유사)

set

```
#include <cstdio>
#include <set>
using namespace std;
int main() {
    set<int> s;
    for ( int i = 0 ; i < 10 ; i++ )
        s.insert(i);
    s.insert(1);s.insert(2)
    for ( set<int>::iterator it=s.begin();
            it!=s.end();it++ )
        printf("%d\n",*it);
    if ( s.count(5) != 0 ) {
        printf("exists\n");
    } else {
        printf("is not exists\n");
    }
    return 0;
}
```

vector

- 사실 가장 많이 쓰이는 친구
- 가변 길이 배열이라는 이름을 갖고, 사실상배열
- 배열의 크기를 몰라도 원소를 계속 추가할 수 있다는 장점
- ~~보통 graph를 인접 리스트로 구성할 때 많이 쓰인다.~~

vector

```
#include <cstdio>
#include <vector>
using namespace std;
int main() {
    vector<int> v;
    for ( int i = 0 ; i < 10 ; i++ )
        v.push_back(i);
    sort(v.begin(),v.end());
    printf("%d %d %d\n”,v[0],v[5],v.size());
    return 0;
}
```

vector

```
#include <cstdio>
#include <vector>
using namespace std;
int main() {
    vector<int> v(10);
    for ( int i = 0 ; i < 10 ; i++ )
        v[i] = i;
    sort(v.begin(),v.end());
    printf("%d %d %d\n”,v[0],v[5],v.size());
    return 0;
}
```

vector

```
#include <cstdio>
#include <vector>
using namespace std;
int main() {
    vector<int> v(10, 0); //모두 0으로 초기화
    for ( int i = 0 ; i < 10 ; i++ )
        v[i] = i;
    sort(v.begin(),v.end());
    printf("%d %d %d\n”,v[0],v[5],v.size());
    return 0;
}
```

vector

```
#include <cstdio>
#include <vector>
using namespace std;
int main() {
    vector<int> v(10, 0); //모두 0으로 초기화
    for ( int i = 0 ; i < 10 ; i++ )
        v[i] = i;
    sort(v.begin(),v.end());
    printf("%d %d %d\n”,v[0],v[5],v.size());
    return 0;
}
```

vector

```
#include <stdio.h>
#include <vector>
using namespace std;

void print(vector <int> a){
    for(int i=0; i<a.size(); i++)
        printf("%d ", a[i]);
    puts("");
}

int main(){
    int n;
    scanf("%d", &n);
    vector <int> a(n);
    for(int i=0; i<n; i++) a[i] = i;

}
```

함수로 넘겨 줄 때
vector 전체 값을 복사

vector

```cpp
#include <stdio.h>
#include <vector>
using namespace std;

void print(vector <int> &a){
    for(int i=0; i<a.size(); i++)
        printf("%d ", a[i]);
    puts("");
}

int main(){
    int n;
    scanf("%d", &n);
    vector <int> a(n);
    for(int i=0; i<n; i++) a[i] = i;

}
```

참조 변수를 이용

vector

```cpp
#include <vector>
#include <algorithm>

using namespace std;

int main(){
    int n;
    scanf("%d", &n);
    vector <int> a(n);
    for(int i=0; i<n; i++) scanf("%d", &a[i]);

    sort( a.begin(), a.end() );

    // 중복 된 값 제거
    a.erase( unique( a.begin(), a.end() ), a.end() );
    int key;
    scanf("%d", &key);
    if( binary_search(a.begin(), a.end(), key) ) puts("YES");
    else puts("NO");
}
```

vector 정렬 및 unique

next_permutation

```cpp
#include <cstdio>
#include <vector>
#include <algorithm>
using namespace std;
int main() {
    vector<int> v;
    for ( int i = 0 ; i < 5 ; i++ )
        v.push_back(i);
    sort(v.begin(),v.end());
    do {
        for ( int i = 0 ; i < (int)v.size() ; i++ )
            printf("%d ",v[i]);
        puts("");
    }while (next_permutation(v.begin(),v.end()) );
    return 0;
}
```

구조체 정렬

```
#include <cstdio>
#include <algorithm>
#include <string>
using namespace std;
struct Student {
    string name;
    int rank;
    int id;
};
bool cmp(const Student& s1,const Student& s2) {
    if ( s1.rank == s2.rank )
        return s1.id < s2.id;
    return s1.rank < s2.rank;
}
int main() {
    const int n = 10;
    Student student[n];
    /* build student */
    sort(student,student+n,cmp);
    return 0;
}
```

Pair

```cpp
#include <algorithm>

using namespace std;

typedef pair <int, int> ii;
typedef pair <long long, long long> ll;
typedef pair <string, int> si;
typedef pair <string, string> ss;
typedef pair <string, ii> sii;
```

Pair

```cpp
#include <algorithm>

using namespace std;

typedef pair <int, int> ii;

int main(){
    int n;
    scanf("%d", &n);
    for(int i=0; i<n; i++){
        int a, b;
        scanf("%d %d", &a, &b);
        p[i] = ii( a, b );
    }
    sort( a, a+n );
    /* 첫 번째 값 기준으로 정렬
       만약 첫 번째 값이 같을 경우 두 번째 값 기준으로 정렬 */
    for(int i=0; i<n; i++)
        printf("%d %d\n", p[i].first, p[i].second);
}
```

Pair를 이용하여
1, 2, 3, ..., n 순위
정렬 기준을 빠르게 구현 가능

Pair

```cpp
#include <algorithm>

using namespace std;

typedef pair <int, int> ii;

int main(){
    int n;
    scanf("%d", &n);
    for(int i=0; i<n; i++){
        int a, b;
        scanf("%d %d", &a, &b);
        p[i] = ii( a, b );
    }
    sort( a, a+n );
    /* 첫 번째 값 기준으로 정렬
       만약 첫 번째 값이 같을 경우 두 번째 값 기준으로 정렬 */
    for(int i=0; i<n; i++)
        printf("%d %d\n", p[i].first, p[i].second);
}
```

Pair를 이용하여
1, 2, 3, ..., n 순위
정렬 기준을 빠르게 구현 가능

번외

C++11,14 문법

범위 기반 for문

```
#include <stdio.h>

int main() {
    int a[] = { 1, 2, 3, 4, 5 };

    for (auto key : a)
        printf("%d ", key);
}
```

배열 a 전체를 출력

But 값을 복사한 값이기 때문에
배열 값 변경 불가능

```
#include <stdio.h>

int main() {
    int a[] = { 1, 2, 3, 4, 5 };

    for (auto &key : a)
        key = 7;
}
```

참조 변수 사용

C++11,14 문법

```cpp
#include <map>

using namespace std;

int main() {
    map <int, string> mp;
    map <int, string>::iterator it;
    mp[3] = "hello";
    mp[4] = "world";

    for (it = mp.begin(); it != mp.end(); it++)
        printf("%d %s\n", it->first, it->second.c_str());
}
```

C++11,14 문법

```
#include <map>

using namespace std;

int main() {
    map <int, string> mp;
    mp[3] = "hello";
    mp[4] = "world";

    for (auto &it : mp)
        printf("%d %s\n", it.first, it.second.c_str());
}
```

Lower, upper bound 응용

```cpp
#include <algorithm>
#include <stdio.h>

using namespace std;

int main() {
    int a[8] = { 1, 2, 2, 2, 2, 3, 4, 4 };
    // sorted

    int count = (int)(upper_bound(a, a + 8, 2) - lower_bound(a, a + 8, 2));
    printf("2의 개수 : %d\n", count);
}
```

Algorithm 추가

```cpp
#include <algorithm>
#include <stdio.h>

using namespace std;

int main() {
    int a = 4, b = 3, c[] = { 1, 2, 3, 4  };

    min(a, b);
    max(a, b);
    abs(a);
    min_element(c + 1, c + 3);
    // c[1]~c[2]까지의 min값
}
```

string.h

```c
#include <string.h>
#include <string>
#include <stdio.h>

const int L = 101;
char s[L];

int main(){
    fgets( s, L, stdin );
    // 띄어쓰기 포함, 줄 단위 문자열 입력 받기
    int l = strlen( s );
    if( s[l-1] == '\n' ) s[l-1] = 0;
    // 줄 바꿈 문자 삭제

    // 단어 단위로 출력
    char *p = strtok( p, " \n\t\r" );
    while( p != NULL ){
        printf("%s\n", p);
        p = strtok( NULL, " \n\t\r" );
    }
}
```

string.h

```
int TestCase;
scanf("%d", &TestCase);

//fgets를 사용 할 경우 앞에 '\n'이 사용 되는지 확인
getchar();
//getchar 함수를 이용하여 '\n' 입력 받음

for(int R=1; R<=TestCase; R++) {
    fgets( s, L, stdin );
}
```

Q&A

끝!

원저자 : 김상근

2차수정 : 한상덕

한상덕(nada, kaseo)
hansgd7 at naver.com

2015 대학생 프로그래밍 경시대회 금상

Google Codejam Round3 (2016)

ACM-ICPC Daejeon regional 본선 2012(11th place), 2013(10th place), 2014(15th place), 2015(6th place)

Sogang Programming Contest 2012 ~ 2016 (출제 및 검토)

전국 대학생 프로그래밍 대회 동아리 연합 대회 2013(7th), 2014(6th), 2015(8th)

LG Codechallenge 2015(6th)

제 1회 SCPC 본선

Hacker cup Round2 (2015)