배열선언

#include <vector>

Vector<int> strvector; // 이렇게 하면 int형 배열이 선언됨

for(int I = 0; I < 5; I++){

strvector.push\_back(I);

}

pop\_back — POP하는것

-> strvector.pop\_back();

Find는?

#include <algorithm>

auto it = find(v.begin(), v.end(), num);

if (it == v.end()) {

cout << num << "은 찾을 수 없습니다.\n";

}

else {

cout << num << "는 존재하며 인덱스는 " << it - v.begin() << " 입니다.\n";

}

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

String

str.substring(5,2)

Index 5부터 2번쨰까지 출력 하거나 가져오기

str.substring(5)

Index 5부터 끝까지 가져오는 거

ToString

-> str.to\_string() // String으로 만들기

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

Map은 이렇게 만듬

#include <map>

map<char, int> m1;

m1.insert(make\_pair('a', 1)); m1.insert(make\_pair('e', 5));

m1.insert(make\_pair('c', 3)); m1.insert(make\_pair('d', 4));

m1.insert(make\_pair('b', 2));

m1['e'] = 6;

m1['f'] = 7;

map<char, int>::iterator it;

for (it = m1.begin(); it != m1.end(); it++) {

printf("<%c %d> ", (\*it).first, (\*it).second);

}

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

< 정렬 >

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

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

< Split 방법 >

#include <sstream>

stringstream ss(rep);

string from, to;

ss >> from >> to;

#**include** <iostream>

#**include** <vector>

#**include** <sstream>

**using** **namespace** std;

vector<string> split(string str, **char** delimiter);

**int** **main**(){

string test = "min sung kang";

vector<string> result = split(test, ' ');

**for** (**int** i=0;i<result.size();i++){

cout << result[i] << " ";

}

}

vector<string> split(string input, **char** delimiter) {

vector<string> answer;

stringstream **ss**(input);

string temp;

**while** (getline(ss, temp, delimiter)) {

answer.push\_back(temp);

}

**return** answer;

}

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

**[ 우선순위 큐를 이용한 다익스트라 소스코드]**

void Dijkstra\_Using\_Heap()

{

    priority\_queue<pair<int, int>> PQ;

    PQ.push(make\_pair(0, Start));

    Dist[Start] = 0;

    while (PQ.empty() == 0)

    {

        int Cost = -PQ.top().first;

        int Cur = PQ.top().second;

        PQ.pop();

        for (int i = 0; i < Vertex[Cur].size(); i++)

        {

            int Next = Vertex[Cur][i].first;

            int nCost = Vertex[Cur][i].second;

            if (Dist[Next] > Cost + nCost)

            {

                Dist[Next] = Cost + nCost;

                PQ.push(make\_pair(-Dist[Next], Next));

            }

        }

    }

    for (int i = 1; i <= V; i++)

    {

        if (Dist[i] == INF) cout << "INF" << endl;

        else cout << Dist[i] << endl;

    }

}

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

자리수 반올림

cout<<fixed;

cout.precision(number); // number은 2번째자리수 등등.. 으로 숫자로 넣어준다.

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

**#include <queue> 우선순위큐**

    priority\_queue<int> pq;  // - >  priority\_queue<int, vector<int>, less<int>> pq;

    // 우선순위 큐에 원소를 삽입 (push) 한다

    pq.push(4);

    pq.push(7);

    while (!pq.empty()) {

        cout << pq.top() << '\n';

        pq.pop();

    }

// 종료 시간 큐 (가장 작은 값이 우선순위가 되는 큐)

priority\_queue< int, vector<int>, greater<int> > pq\_less;

// 두개이상의 배열

vector< pair<int, int> > class\_time; class\_time.push\_back(make\_pair(start\_time, end\_time));

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

# 순열 조합

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

int main(){

vector<int> v = {1, 2, 3};

// 3P3 출력하는 코드

do

{

for(int i = 0; i < v.size(); i++)

{

cout << v[i] << " ";

}

cout << '\n';

}while(next\_permutation(v.begin(),v.end()));

}

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

Char

ㅁ isalpha([word.at](http://word.at)(i))) - 알파벳인지 아닌지

ㅁ word[i] = ::toupper(word[i]); - 대문자로 만들기

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

Set

set<int> s1;

s1.insert(1); s1.insert(2); s1.insert(6);

s1.insert(5); s1.insert(4); s1.insert(3);

set<int>::iterator it;

for (it = s1.begin(); it != s1.end(); it++) {

printf("%d ", \*it);

}