2021 신촌 연합 여름캠프 초급반 1회차

C언어 리뷰 & C++ 기본

서강대학교 박재형

강사 소개

- 서강대학교 수학 / 컴퓨터공학

- BOJ : pjh6792 / Codeforces : Rebro

- 블로그 : rebro.kr

- SUAPC 2021 Winter 은상(3등)

- 2021 Winter Camp 중급 Contest 은상(2등)

C언어 리뷰

- 자료형
- 입출력
- 조건문 / 반복문
- 포인터
- 배열
- 문자열

자료형

- 정수 자료형

자료형	메모리 크기	값의 범위
char	1 byte	-128 ~ 127
short unsigned short	2 byte	-32,768 ~ 32,767 0 ~ 65,535
int unsigned int	4 byte	-2,147,483,648 ~ 2,147,483,647 0 ~ 4,294,967,295
long long int unsigned long long int	8 byte	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807 0 ~ 18,446,744,073,709,551,615

자료형

- 실수 자료형

자료형	메모리 크기	값의 범위	유효 자릿수
float	4 byte	1.175494e-38 ~ 3.402823e+38	7자리
double	8 byte	2.225074e-308 ~ 1.797693e+308	16자리

- 부동 소수점 (링크)

- 오차 주의!

입출력

- printf / scanf

- #include<stdio.h>

- printf("오늘은 %d월 %d일", 7, 6);

- scanf("%d", &a);

서식 지정자	출력 데이터 형태
%c	문자 1개
%s	문자열
%d	int
%lld	long long
%f	float
%lf	double

조건문

```
if (조건식) {

        실행문

}
```

```
int score = 75;
if (score > 50) printf("pass");
```

```
if (조건식) {

        실행문

}
else {

        실행문

}
```

```
if (score > 50) printf("pass");
else printf("fail");
```

```
if (조건식) {

        실행문

}
else if (조건식2) {

        실행문

}
        ...
else {

        실행문

}
```

```
int score = 75;
if (score >= 90) {
    printf("A");
}
else if (score >= 80) {
    printf("B");
}
else if (score >= 70) {
    printf("C");
}
else if (score >= 60) {
    printf("D");
}
else {
    printf("F");
}
```

반복문

- for문

①　②　④

```
for(초기화식; 조건식; 증감식){

        실행 문장
            ③
}
```

- break / continue

```
for (int i = 1; i <= 10; i++) {
    printf("%d ", i);
}
for (int i = 1; ; i++) {
    printf("%d ", i);
}
for (int i = 1; i <= 10; ) {
    printf("%d", i);
}
int i = 1;
for (; i <= 10; i++) {
    printf("%d", i);
}
```

```
for (int i = 1; i <= 10; i++) {
    if (i % 2 == 0) continue;
    printf("%d ", i);
}
int sum = 0;
for (int i = 1; ; i++) {
    sum = sum + i;
    if (sum > 100) break;
}
```

반복문

- while문

①

```
while(조건식){

        실행 문장

        ②
}
```

```
int i = 1;
while (i <= 10) {
    printf("%d", i);
    i++;
}
while (true) {
    printf("a");
}
```

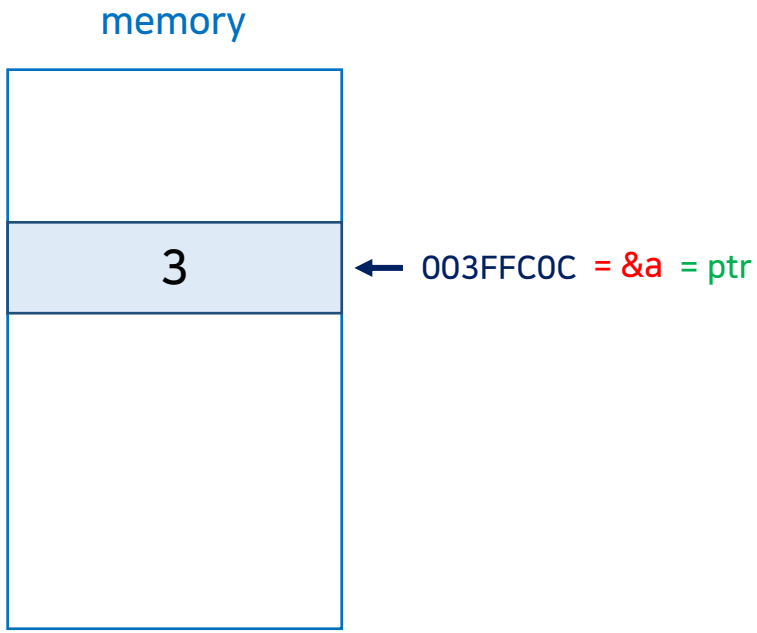
포인터

- int a = 3;

- 참조 연산자(&) / 역참조 연산자(*)

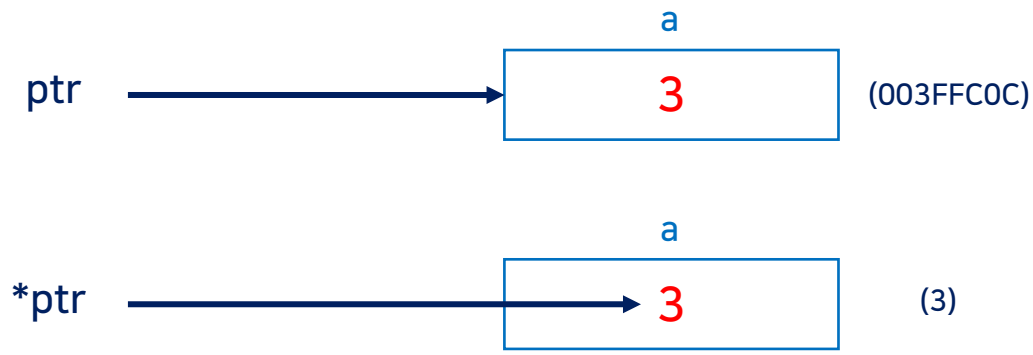
- printf("%p", &a);

- int* ptr = &a;



포인터

```
int a = 3;

int* ptr = &a;

*ptr = 20;

printf("%d", a);
```



배열

- 자료형 배열이름[크기]

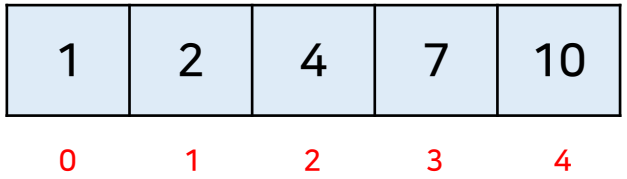
  ex) int a[5];

  int a[5] = {1, 2, 4, 7, 10};

  int a[] = {1, 2, 4, 7, 10};

- 연속된 메모리 공간

- 0-base



- Index 접근 가능

배열

- 배열의 크기는 반드시 상수
- 배열의 이름 = 첫번째 원소의 포인터
- 전역변수 vs 지역변수
- Undefined Behavior

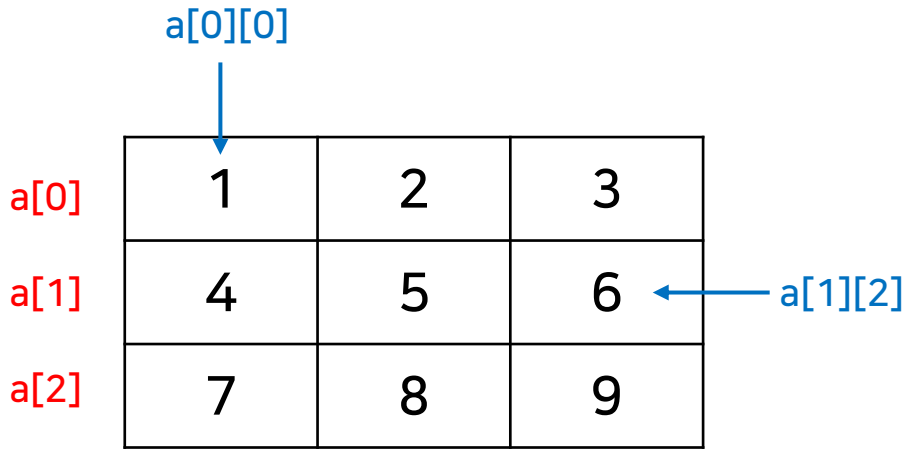
```
int main(void) {
    int a[3] = { 1, 2, 3 };
    for (int i = 0; i < 4; i++) {
        cout << a[i] << '¥n';
    }
}
```

배열

- 2차원 배열

ex) int a[3][3]

int a[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}



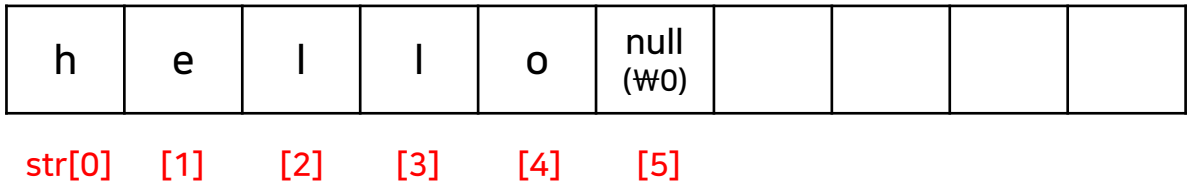
- n차원 배열

문자열

- 문자(char) type의 배열

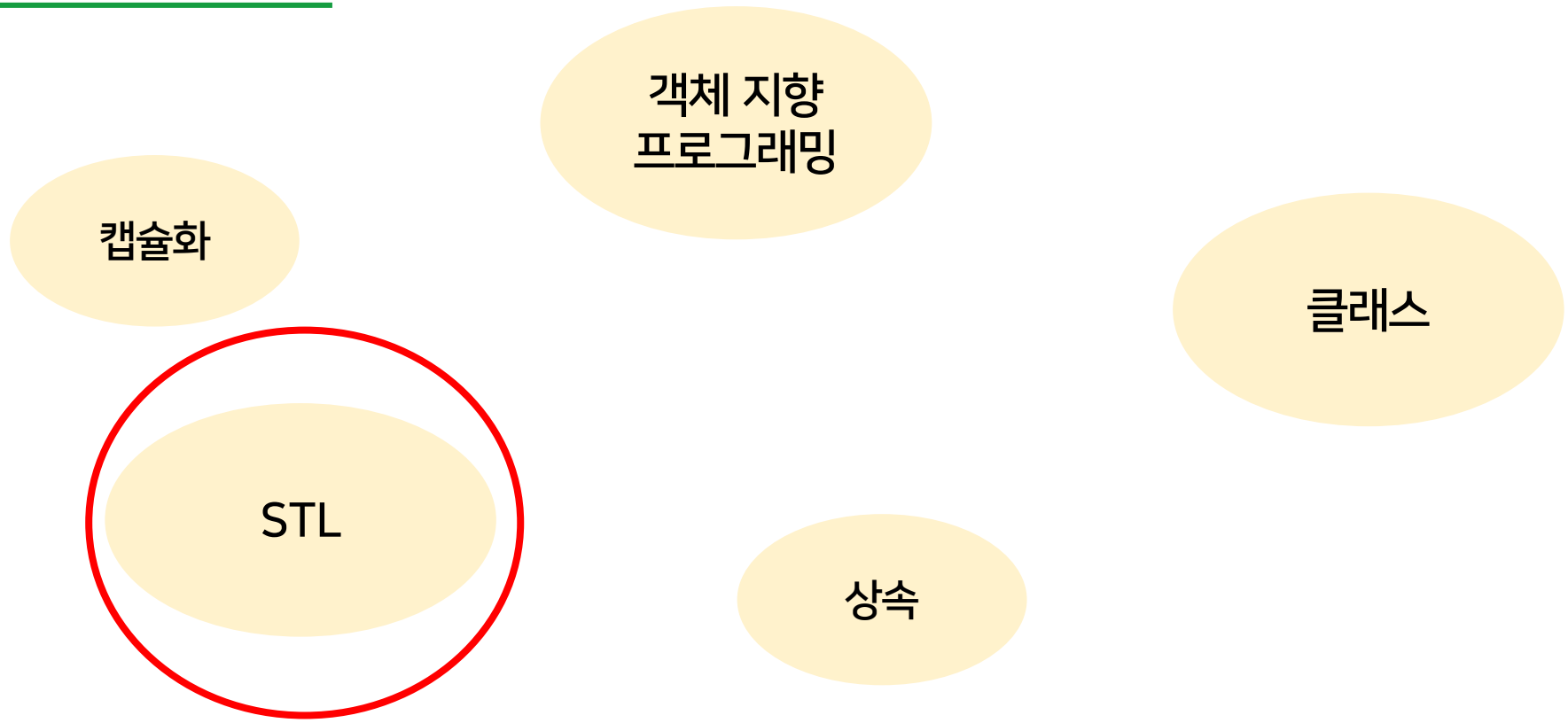
ex) char str[] = “hello”

char str[10] = “hello”



- 맨 뒤에 null(‘₩0’) 문자

C++ 기본



C++ 입출력

- cin / cout
- #include<iostream>
- std::cin >> [변수]
- std::cout << [출력할 값]

C++ 입출력 예시

```cpp
#include<iostream>
int main(void) {
    int a;
    std::cin >> a;
}
```

```cpp
#include<iostream>
int main(void) {
    int a;
    float b;
    char c;
    std::cin >> a >> b >> c;
}
```

```cpp
#include<iostream>
int main(void) {
    std::cout << "Hello world!";
}
```

```cpp
#include<iostream>
int main(void) {
    int a = 6;
    std::cout << "a value is " << a << '!';
}
```

C++ 입출력 속도

- 출력 개행 : std::endl OR '₩n'
- 출력 속도 비교 (https://www.acmicpc.net/blog/view/57)
- '₩n' (0.9229s) vs endl (11.5322s) (이유 링크)

```cpp
#include<iostream>
int main(void) {
    int a = 6;
    std::cout << "a value is " << a << '!' << std::endl;
    std::cout << "a value is " << a << '!' << '¥n';
}
```

- std::ios::sync_with_stdio(false)
- std::cin.tie(NULL), std::cout.tie(NULL)

```cpp
int main(void) {
    std::ios::sync_with_stdio(false);
    std::cin.tie(NULL); std::cout.tie(NULL);
}
```

using namespace std;

<namespace>

- 큰 프로젝트에서 이름이 충돌하는 것을 방지

- 함수나 구조체, 변수 이름 등의 소속

```
#include<iostream>
using namespace std;
int main(void) {
    int a;
    cin >> a;
    cout << "a value is " << a << '!' << endl;
    cout << "a value is " << a << '!' << '¥n';
}
```

기본 템플릿

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

int main(void) {
    ios::sync_with_stdio(false); cin.tie(NULL); cout.tie(NULL);
    /*
        main code
    */
}
```

STL이란?

- Standard Template Library (표준 템플릿 라이브러리)
- 프로그램에 필요한 자료구조와 알고리즘을 템플릿으로 제공
- 컨테이너, 반복자, 알고리즘
- 장/단점

1. 컨테이너 (Container)

- 임의 타입의 객체(원소)들을 보관하는 저장소

- 순차 컨테이너 (Sequence Container) : vector, list, string, deque 등

- 연관 컨테이너 (Associative Container) : set, map, multiset, multimap 등

- 컨테이너 어댑터 (Container Adaptor) : stack, queue, priority_queue 등

- 컨테이너 별로 유용한 함수들 존재

2. 알고리즘 (Algorithm)

- 컨테이너의 원소들을 조작할 수 있는 함수들의 모임
- 검색, 정렬, 수정, 개수 세기 등
- 반복자 또는 포인터로 작업할 원소를 가리킴
- [first, last)
- https://en.cppreference.com/w/cpp/algorithm

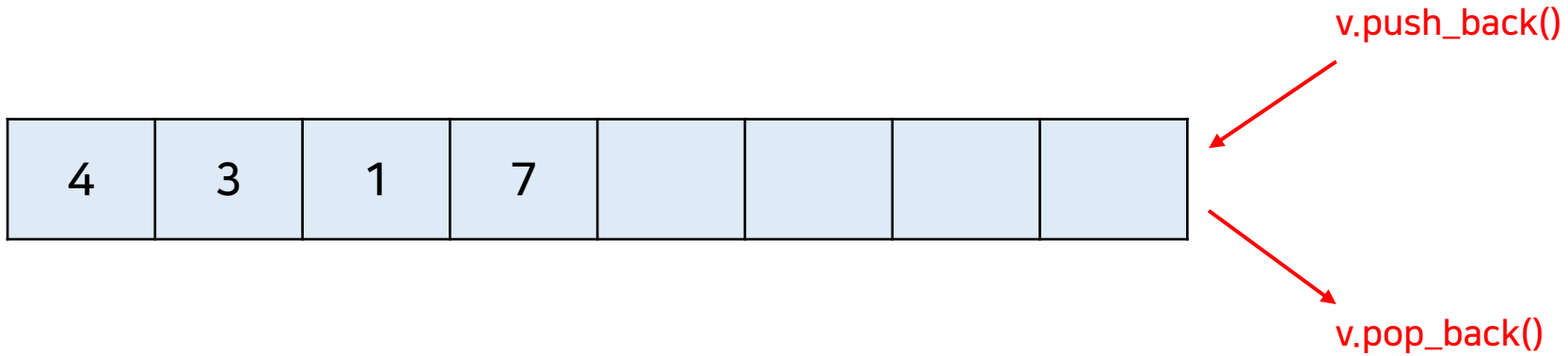
벡터 (vector)

- Sequence Container의 한 종류

- #include<vector>

- vector<[data type]> [name]

- 자동으로 메모리가 할당되는 배열 (동적 배열)



벡터 (vector)

<배열 (Array) vs 벡터 (vector)>

- Random Access
- 메모리 / 시간
- 멤버 함수
- Undefined Behavior

벡터 (vector) 생성하기

- vector<[type]> v

- vector<[type]> v = {a1, a2, a3, …}

- vector<[type]> v(n)

- vector<[type]> v(n, m)

- vector<[type]> v2(v1)

- vector<vector<[type]>> v

- vector<vector<[type]>> v(n, vector<[type]>(m))

벡터 (vector) 생성하기

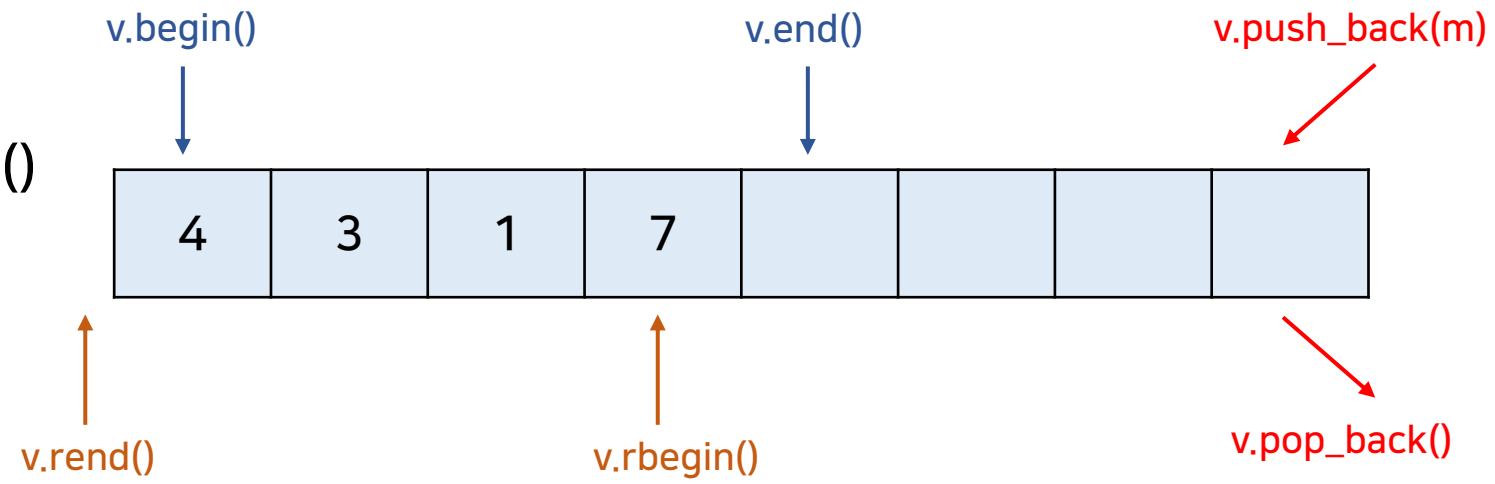
```cpp
#include<vector>
using namespace std;

int main(void) {
    vector<int> v;                          // {} (empty vector)
    vector<int> v(5);                       // {0, 0, 0, 0, 0}
    vector<int> v = { 1, 2, 3, 4, 5 };      // {1, 2, 3, 4, 5}
    vector<int> v(5, 1);                    // {1, 1, 1, 1, 1}
    vector<int> v2(v);
    vector<vector<int>> v;
    vector<vector<int>> v(3, vector<int>(2, 5)); // {{5, 5}, {5, 5}, {5, 5}}
}
```

벡터 (vector) 다루기

- assign(n, m)

- size() / empty()

- begin() / end() / rbegin() / rend()



- front() / back()

- push_back(m) / pop_back()

벡터 (vector) 다루기

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

int main(void) {
    vector<int> v = { 1, 2, 3 };
    for (int i = 0; i < v.size(); i++) {
        cout << v[i]<<'¥n';
    }
    vector<int>::iterator it;
    for (it = v.begin(); it != v.end(); it++) {
        cout << *it << '¥n';
    }
}
```

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

int main(void) {
    vector<int> v = { 1, 2, 3 };
    for (int i = (int)v.size() - 1; i >= 0; i--) {
        cout << v[i] << '¥n';
    }
    vector<int>::reverse_iterator it;
    for (it = v.rbegin(); it != v.rend(); it++) {
        cout << *it << '¥n';
    }
}
```

벡터 (vector) 다루기

- resize(n) / resize(n, m)

- insert(iter, m) / insert(iter, k, m)

- erase(iter) / erase(start, end)

Pair

ex) 순서쌍, 좌표평면 위의 점

- 두 객체를 하나의 객체로 묶어 사용할 수 있게 해주는 클래스

- #include<utility> (vector or algorithm 헤더에 포함)

- pair<[타입1], [타입2]> p;

- first, second로 원소 접근 가능

- 연산자 사용 가능

Pair 사용 예시

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

int main(void) {
    pair<int, int> p1, p2, p3;
    p1.first = 2;
    p1.second = 1;
    p2 = make_pair(3, 1);
    p3 = { 0, 1 }; // since c++11

    if (p1 < p2) cout << "<";
    else if (p1 == p2) cout << "same";
    else cout << ">";
}
```

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

int main(void) {
    pair<int, pair<int, int>> p;
    p = { 1, {2, 3} };
    cout << p.second.first;
}
```

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

int main(void) {
    vector<pair<int, int>> v;
    v.push_back({ 2, 3 });
}
```

<Algorithm>

- min / max

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

int main(void) {
    cout << min(2, 4); // 2
    cout << max(3, 10); // 10

    //since C++14
    cout << min({ 4, 1, -5, 2 }); // -5
    cout << max({ 3, 0, 1 }); // 3
}
```

- min_element / max_element

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

int main(void) {
    vector<int> v = { 3, 1, 2, 0, 7, 4 };
    int arr[5] = {5, 4, 3, 2, 1};
    cout << *max_element(v.begin(), v.end());
    cout << *min_element(v.begin()+1, v.end()-3);
    cout << *min_element(arr, arr + 5);
}
```

<Algorithm>

- swap

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

int main(void) {
    int a = 3;
    int b = 5;
    swap(a, b);
    vector<int> v1 = { 1, 2 };
    vector<int> v2 = { 0, 1 };
    swap(v1, v2);
}
```

- reverse

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

int main(void) {
    vector<int> v = { 3, 1, 2, 0, 7, 4 };
    reverse(v.begin(), v.end()); // {4, 7, 0, 2, 1, 3}
    reverse(v.begin(), v.begin() + 3); // {0, 7, 4, 2, 1, 3}
}
```

How to study C++

2206번 | 제출 | 맞은 사람 | 숏코딩 | 재채점 결과 | 채점 현황 | 내 제출 | 난이도 기여 | 강의 | 질문 검색

벽 부수고 이동하기 성공

시간 제한	메모리 제한	제출	정답	맞은 사람
2 초	192 MB	58314	14460	8913

문제

N×M의 행렬로 표현되는 맵이 있다. 맵에서 0은 이동할 수 있는 곳을 나타내고, 1은 이동할 수 없는 벽이 있는 곳을 나타낸다. 당신은
는데, 이때 최단 경로로 이동하려 한다. 최단경로는 맵에서 가장 적은 개수의 칸을 지나는 경로를 말하는데, 이때 시작하는 칸과 끝나

만약에 이동하는 도중에 한 개의 벽을 부수고 이동하는 것이 좀 더 경로가 짧아진다면, 벽을 한 개 까지 부수고 이동하여도 된다.

메모리	시간	언어	코드 길이	제출한 시간
3008 KB	16 ms	C++17	2507 B	7달 전
4000 KB	24 ms	C++11	1609 B	1년 전
4092 KB	28 ms	C99	3154 B	5년 전
6032 KB	28 ms	C++14	4198 B	11달 전
6052 KB	28 ms	C99	1175 B	1년 전
6052 KB	28 ms	C99	1194 B	10달 전
9076 KB	28 ms	C++98	6679 B	1년 전
11928 KB	28 ms	C++17	2528 B	6달 전
2204 KB	32 ms	C++14	843 B	11달 전
4948 KB	32 ms	C++14	772 B	3달 전
5052 KB	32 ms	C++17	1120 B	6달 전
5056 KB	32 ms	C++14	1903 B	11달 전
5108 KB	32 ms	C++14	1601 B	10달 전
5232 KB	32 ms	C++20 (Clang)	938 B	1년 전
6080 KB	32 ms	C++17	3140 B	2달 전
7984 KB	32 ms	C++14	3248 B	2년 전

How to study C++

ex) a, b, c, d 네 변수의 값 중 최댓값을 구하시오

C

```c
int main(void) {
    int a, b, c, d;
    if (a >= b && a >= c && a >= d) cout << a;
    else if (b >= a && b >= c && b >= d) cout << b;
    else if (c >= a && c >= b && c >= d) cout << c;
    else cout << d;
}
```

```c
int main(void) {
    int a, b, c, d;
    int ans = a;
    if (ans < b) ans = b;
    if (ans < c) ans = c;
    if (ans < d) ans = d;
    cout << ans;
}
```

C++

```cpp
int main(void) {
    int a, b, c, d;
    cout << max({ a, b, c, d });
}
```

공부하면 좋은 것들

- 반복자 (Iterator)

- Container : Set / Map / Multiset / Multimap / String

- Algorithm : next_permutation / unique / copy / fill / find

출석 / 연습문제

[출석문제]

- 15552 (빠른 A+B)
- 10804 (카드 역배치)
- 1158 (요세푸스 문제)
- 1547 (공)

[연습문제]

- 1008 (A/B)
- 2993 (세 부분)
- 9093 (단어 뒤집기)
- 2346 (풍선 터뜨리기)