



2023-2 고급프로그래밍

Tutorial

과제 1: 쭈쭈미게임(1)



한남대학교 컴퓨터공학과

TUTORIAL



tutorial.c

- Sleep()
 - 1초마다 1 증가하는 변수

```
#include <stdio.h>
#include <Windows.h>

int main(void) {
    int y = 1;
    while (1) {
        printf("%d\n", y);
        y++;
        Sleep(1000); // ms
    }
}
```

```
1
2
3
4
5
6
7
...
```

tutorial.c

- tick: 이 프로그램이 동작하는 최소 시간 단위(1 tick = 10 ms)

```
#include <stdio.h>
#include <Windows.h>

int tick = 0;
int main(void) {
    int y = 1;
    while (1) {
        // 3초마다 인사
        if (tick % 3000 == 0) {
            printf("안녕하세요\n");
        }
        // 1초마다 x++
        if (tick % 1000 == 0) {
            printf("sec: % d\n", y);
            y++;
        }
        Sleep(10);
        tick += 10;
    }
}
```

안녕하세요
1
2
3
안녕하세요
4
5
6
안녕하세요
7
...

tutorial.c

- gotoxy(int row, int col)
 - 커서를 특정 행, 열로 옮겨주는 함수
 - **커서(Cursor)**: 콘솔 화면에서 다음에 출력할 위치

```
#include <conio.h>
#include <Windows.h>
```

```
void gotoxy(int row, int col);
```

```
void gotoxy(int row, int col) {
    COORD pos = { col, row }; // 행, 열 반대로 전달
    SetConsoleCursorPosition(
        GetStdHandle(STD_OUTPUT_HANDLE),
        pos
    );
}
```

*

1칸씩 움직인다.

```
int main(void) {
    int y = 1;
    while (1) {
        // 0.1초마다 '*' 이동
        if (tick % 100 == 0) {
            gotoxy(1, y); printf(" "); // 지우고
            y++;                       // 이동
            gotoxy(1, y); printf("*"); // 새 위치에 출력
        }
        Sleep(10);
        tick += 10;
    }
}
```

tutorial.c

- **<conio.h>:_kbhit()**
 - 키 입력이 있는지 확인하는 함수
- **<conio.h>:_getch()**
 - 키 입력을 받는 함수(엔터 키가 입력될 때까지 기다리지 않음)
 - blocking function이므로 _kbhit()가 참일 때만 사용

```
int main(void) {
    int y = 1;
    while (1) {
        // 'q'를 누르면 종료
        if (_kbhit()) { // 키 입력은 loop 돌 때마다 확인
            int key = _getch();
            if (key == 'q') { break; }
        }

        // 0.1초마다 '*' 이동
        if (tick % 100 == 0) { ... }
        Sleep(10);
        tick += 10;
    }
}
```

tutorial.c

- 방향키 입력 받기

- 다른 키('a', 'b', '1', '!', ...)와 달리 2byte로 입력 된다.
 - MSB(224) + LSB(72/80/75/77)
 - 이 예제에서는 사용하지 않음. 과제에는 적용되어 있음

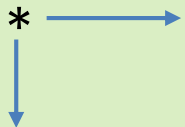
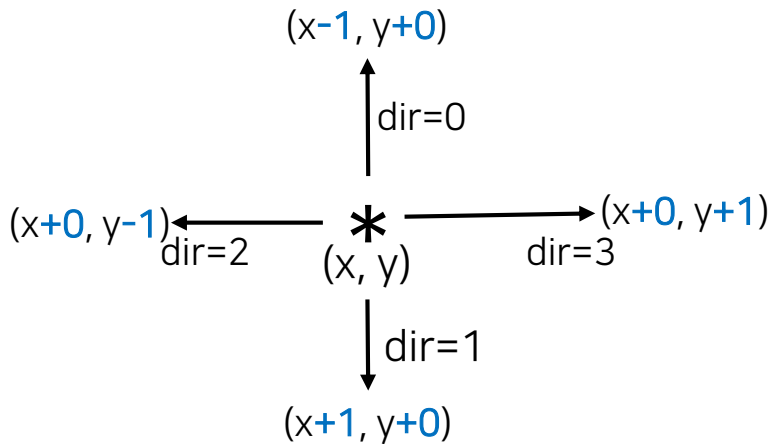
방향키를 눌렀으면
_getch()를 한번 더 호출
다른 키(예:'q')면 그냥 통과

```
if (_kbhit()) {  
    int key = _getch();  
    if (key == 224) {  
        key = _getch();  
        switch (key) {  
            case 75: printf("<-\n"); break;  
            case 77: printf("->\n"); break;  
        }  
    }  
}
```

tutorial.c

- 4방향 이동

- $x+dx[dir], y+dy[dir]$



'w', 's', 'a', 'd'를 입력하면
상/하/좌/우로 움직인다.

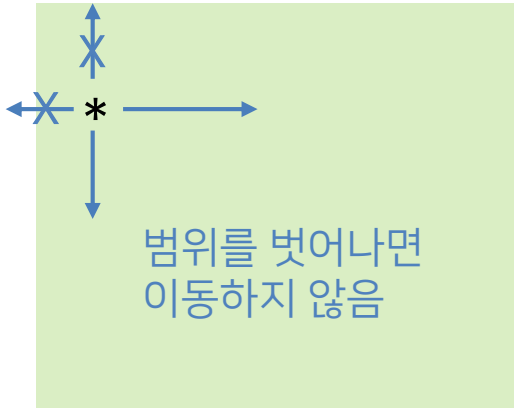
```
int main(void) {
    // 4방향의 변위( $\Delta x, \Delta y$ )
    int dx[4] = {-1, 1, 0, 0};
    int dy[4] = {0, 0, -1, 1};
    int x = 1, y = 1;
    int dir = 3; // 방향(direction). 처음에는 오른쪽
    while (1) {
        // 'w', 's', 'a', 'd': 이동, 'q': 종료
        if (_kbhit()) {
            int key = _getch();
            switch (key) {
                case 'w': dir = 0; break; // up (x-1, y+0)
                case 's': dir = 1; break; // down (x+1, y+0)
                case 'a': dir = 2; break; // left (x+0, y-1)
                case 'd': dir = 3; break; // right (x+0, y+1)
                case 'q': return 0;
            }
        }

        // 0.1초마다 '*' 이동
        if (tick % 100 == 0) {
            gotoxy(x, y); printf(" "); // 지우고
            x += dx[dir]; y += dy[dir]; // 이동
            gotoxy(x, y); printf("*"); // 새 위치에 출력
        }
        Sleep(10);
        tick += 10;
    }
}
```


tutorial.c

- **Boundary check**

- 다음에 이동할
- 위치 (nx, ny)를
- 미리 확인해 본다.



```
int main(void) {
    ...
    while (1) {
        // 'w', 's', 'a', 'd': 이동, 'q': 종료
        if (_kbhit()) { ... }

        // 0.1초마다 '*' 이동
        if (tick % 100 == 0) {
            // nx, ny: 다음 위치
            int nx = x + dx[dir];
            int ny = y + dy[dir];
            // (nx, ny)로 이동할 수 없으면 pass
            if (nx > 0 && nx < 9 &&
                ny > 0 && ny < 19) {
                gotoxy(x, y); printf(" ");
                x = nx; y = ny;
                gotoxy(x, y); printf("*");
            }
        }
        Sleep(10);
        tick += 10;
    }
}
```

tutorial.c

- 맵 만들기

```
...
void draw(void);
...
int tick = 0;
char map[10][20]; // 전역변수

void draw(void) {
    system("cls"); // 깨끗이 지우기
    for (int i = 0; i < 10; i++) {
        for (int j = 0; j < 20; j++) {
            printf("%c", map[i][j]);
        }
        printf("\n");
    }
}
```

```
#####
#               #
#               #
#               #
#      *      #
#               #
#               #
#               #
#               #
#####
```

```
...
int main(void) {
    // 맵 초기화
    for (int i = 0; i < 10; i++) {
        for (int j = 0; j < 20; j++) {
            if (i == 0 || i == 9 || j == 0 || j == 19)
                map[i][j] = '#';
            else
                map[i][j] = ' ';
        }
    }
    map[1][1] = '*';
    ...
    while (1) {
        // 'w', 's', 'a', 'd': 이동, 'q': 종료
        if (_kbhit()) { ... }

        // 0.1초마다 '*' 이동
        if (tick % 100 == 0) {
            ...
            if (nx > 0 && nx < 9 &&
                ny > 0 && ny < 19) {
                gotoxy(x, y); printf(" ");
                map[x][y] = ' ';
                x = nx; y = ny;
                gotoxy(x, y); printf("*");
                map[x][y] = '*';
            }
        }
        draw(); ← 매번 새로 그림
        Sleep(10);
        tick += 10;
    }
}
```

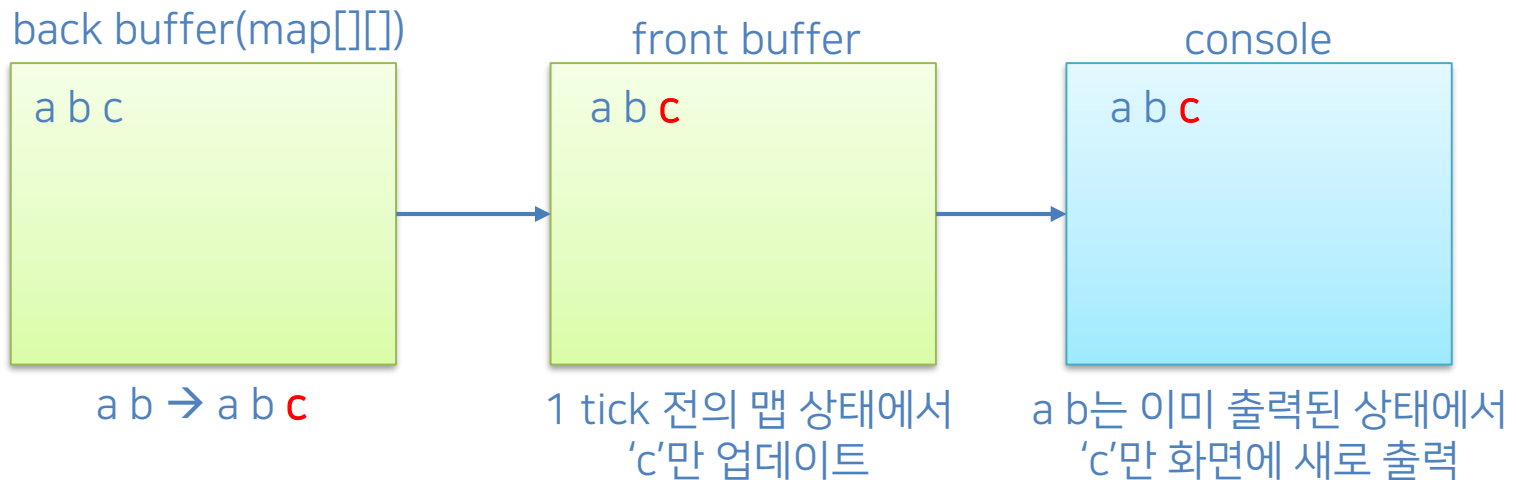
tutorial.c

- 화면 깜박임 이유

- 매번 다 지우고 다시 쓰니까.
- 입출력은 일반 연산보다 훨씬 느린 작업

- 더블 버퍼링

- 화면 전체를 다시 출력하는 게 아니라 back buffer에 먼저 기록하고,
- front buffer와 비교해서 다른 부분만 업데이트 & 출력한다.
 - 참고: <https://codevang.tistory.com/39>
 - 게임 UI
 - 영상 압축에도 비슷한 원리가 활용됨(Key frame)



tutorial.c

- 더블 버퍼링: 깜박임 없애기

부드럽게 움직임

```
#####  
#                                     #  
#                                     #  
#                                     #  
#      ↑                               #  
#     ← * →                           #  
#      ↓                               #  
#                                     #  
#                                     #  
#                                     #  
#####
```

...

```
int tick = 0;  
char map[10][20], front[10][20];
```

...

```
void draw(void) {  
    system("cls");  
    for (int i = 0; i < 10; i++) {  
        for (int j = 0; j < 20; j++) {  
            printf("%c", map[i][j]);  
            if (front[i][j] != map[i][j]) {  
                front[i][j] = map[i][j];  
                gotoxy(i, j);  
                printf("%c", front[i][j]);  
            }  
        }  
        printf("\n");  
    }  
}
```

tutorial.c

- Full Source Code

```
#include <stdio.h>
#include <Windows.h>
#include <conio.h>

void gotoxy(int row, int col);
void draw(void);

int tick = 0; // 시계
char map[10][20], front[10][20];

void gotoxy(int row, int col) {
    COORD pos = { col, row }; // 행, 열 반대로 전달
    SetConsoleCursorPosition(
        GetStdHandle(STD_OUTPUT_HANDLE),
        pos
    );
}

void draw(void) {
    for (int i = 0; i < 10; i++) {
        for (int j = 0; j < 20; j++) {
            if (front[i][j] != map[i][j]) {
                front[i][j] = map[i][j];
                gotoxy(i, j);
                printf("%c", front[i][j]);
            }
        }
    }
}
```

```
int main(void) {
    for (int i = 0; i < 10; i++) {
        for (int j = 0; j < 20; j++) {
            if (i == 0 || i == 9 || j == 0 || j == 19)
                map[i][j] = '#';
            else
                map[i][j] = ' ';
        }
    }
    map[1][1] = '*';

    // 4방향의 변위(Δx, Δy)
    int dx[4] = {-1, 1, 0, 0};
    int dy[4] = {0, 0, -1, 1};
    int x = 1, y = 1;
    int dir = 3; // 방향(direction). 처음에는 오른쪽

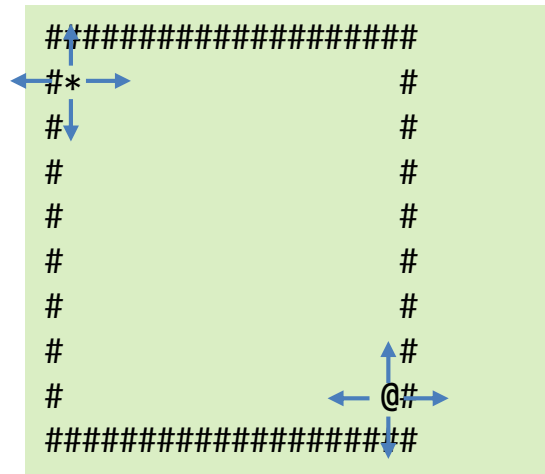
    while (1) {
        // 'w', 's', 'a', 'd': 이동, 'q': 종료
        if (_kbhit()) {
            int key = _getch();
            switch (key) {
                case 'w': dir = 0; break; // up (x-1, y+0)
                case 's': dir = 1; break; // down (x+1, y+0)
                case 'a': dir = 2; break; // left (x+0, y-1)
                case 'd': dir = 3; break; // right (x+0, y+1)
                case 'q': return 0;
            }
        }

        // 0.1초마다 '*' 이동
        if (tick % 100 == 0) {
            // nx, ny: 다음 위치
            int nx = x + dx[dir];
            int ny = y + dy[dir];
            // (nx, ny)로 이동할 수 없으면 pass
            if (nx > 0 && nx < 9 &&
                ny > 0 && ny < 19) {
                map[x][y] = ' '; // 지우고
                x = nx; y = ny; // 이동
                map[x][y] = '*'; // 새 위치에 출력
            }
        }

        draw();
        Sleep(10);
        tick += 10;
    }
}
```

연습문제

- 1) 혼자서 무작위로 움직이는 적 '@'를 추가한다.
- 2) 플레이어 '*'와 적 '@'이 충돌하면 게임이 종료되도록 만들어 보자.



과제1: 쭈꾸미 게임(1)



과제1: 주꾸미 게임(1)

- 팀프로젝트(3인)

- 과제2도 과제1 결과물로 계속 작성



Skeleton code

- 자료실에서 jjuggumi.zip 다운로드

- **jjuggumi.h**

- 공용 설정

- **jjuggumi.c**

- 메인함수

- **canvas.h, canvas.c**

- 화면 출력과 관계된 코드

- **keyin.h, keyin.c**

- 키 입력과 관계된 코드

- **sample.c**

- 간단한 예시

```
...  
// 기본값 true, 탈락하면 false  
bool player[PLAYER_MAX];  
int n_player, n_alive;  
int tick; // 시계
```

```
// 미니게임  
void sample(void);  
//void mugunghwa(void);  
//void nightgame(void);  
//void juldarigi(void);  
//void jebi(void);  
...
```

```
...  
int main(void) {  
    jjuggumi_init();  
    sample();  
    //mugunghwa();  
    //nightgame();  
    //juldarigi();  
    //jebi();  
    return 0;  
}
```

미니게임
4개 구현

Skeleton code

- 자료실에서 jjuggumi.zip 다운로드
 - **jjuggumi.h**
 - 공용 설정
 - **jjuggumi.c**
 - 메인함수
 - **canvas.h, canvas.c**
 - 화면 출력과 관계된 코드
 - **keyin.h, keyin.c**
 - 키 입력과 관계된 코드
 - **sample.c**
 - 간단한 예시

```
// 화면 크기(맵 크기x. 맵 + 상태창)
```

```
#define ROW_MAX    40
```

```
#define COL_MAX    80
```

```
char front_buf[ROW_MAX][COL_MAX];
```

```
char back_buf[ROW_MAX][COL_MAX];
```

```
...
```

```
void map_init(int n_row, int n_col);
```

```
void dialog(char message[]);
```

```
bool placable(int row, int col);
```

```
void display(void);
```

```
void gotoxy(int x, int y);
```

```
void printxy(char ch, int row, int col);
```

```
...
```

```
// 입력 받는 키 종류
```

```
#define K_ARROW    224
```

```
#define K_UP        72
```



```
...
```

```
#define K_UNDEFINED '\0'
```

```
typedef int key_t;
```

```
key_t get_key(void);
```

Skeleton code

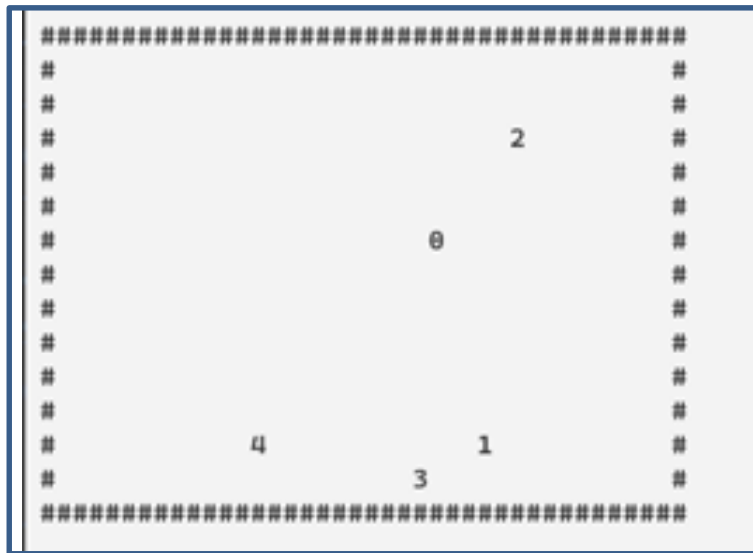
- **jjuggumi.zip**
 - **jjuggumi.h**
 - 공용 설정
 - **jjuggumi.c** 호출
 - 메인함수 
 - **canvas.h, canvas.c**
 - 화면 출력과 관계된 코드
 - **keyin.h, keyin.c**
 - 키 입력과 관계된 코드
 - **sample.c** 
 - 간단한 예제

```
void sample(void) {
    sample_init();
    system("cls");
    display();
    while (1) {
        // player 0만 손으로 움직임(4방향)
        key_t key = get_key();
        if (key == K_QUIT) {
            break;
        } else if (key != K_UNDEFINED) {
            move_manual(key);
        }

        // player 1 부터는 랜덤으로 움직임(8방향)
        for (int i = 1; i < n_player; i++) {
            if (tick % period[i] == 0) {
                move_random(i, -1);
            }
        }

        display();
        Sleep(10);
        tick += 10;
    }
}
```

게임 화면 구성



맵 (게임마다 크기 다름)

```
no. of players left: 5
player 0: alive
player 1: alive
player 2: alive
player 3: alive
player 4: alive
|
```

상태창은 항상 출력
(뒤에 있는 슬라이드에서는 생략. 화면이 작아서...)

게임에 맞춰서
추가 정보를 출력하는 공간

과제 1-1) 준비

- jjuggumi.c:
- intro() 작성
 - 게임 시작할 때 적당한 ascii art 구현
 - 5초 내로 끝내기
- ending() 작성
 - 우승자 출력

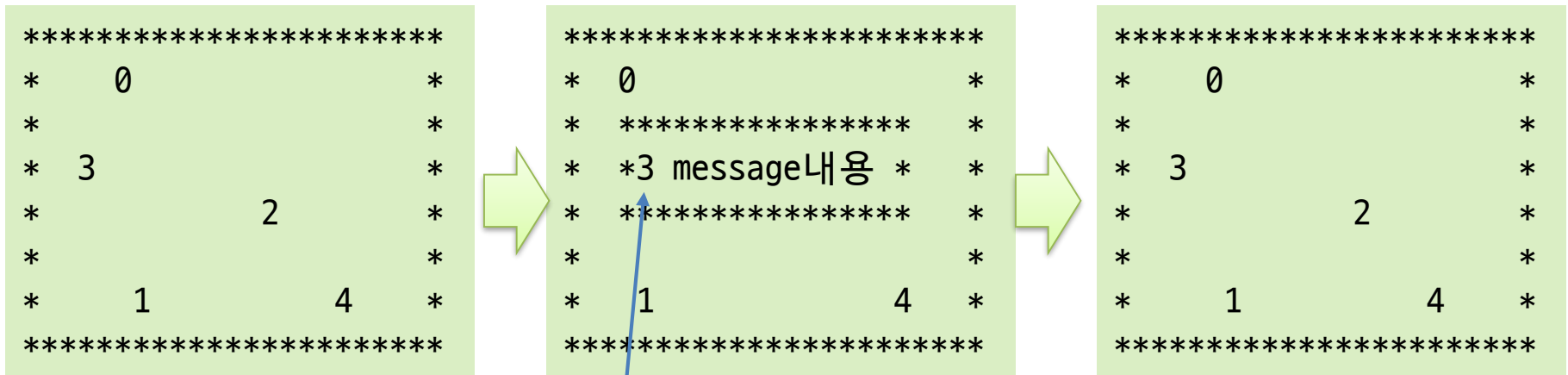
```
...  
int main(void) {  
    jjuggumi_init();  
    intro();  
    sample();  
    //mugunghwa();  
    //nightgame();  
    //juldarigi();  
    //jebi();  
    ending();  
    return 0;  
}
```

과제 1-1) 준비

- **canvas.c: dialog(char message[]) 구현**

- 게임이 잠시 멈추고
 - DIALOG_DURATION_SEC 초 동안
 - 기존 화면 위에 남은 시간(초)과 메시지 출력
 - 남은 시간은 1초마다 4 -> 3 -> 2 -> 1 -> 0으로 바뀜
 - 0초가 되면 dialog 이전 화면을 복구
- sample()에서 테스트해 보기
 - 예)

```
#define DIALOG_DURATION_SEC 4
...
void dialog(char message[]) {
}
```



1초마다 3->2->1->0 또는 4->3->2->1

1-2) jebi.c

- **jebi.c 파일을 만들고 jebi() 작성**
 - sample.c 참고
 - 남아 있는 참가자들을 1라운드에 1명씩 떨어뜨린다.
 - 제비뽑기는 마지막 게임이므로, 우승자 1명만 남을 때까지 반복한다.
 - 맵 아래에 round 번호와 제비를 뽑는 참가자 번호를 출력
- **라운드마다**
 - 남은 참가자 수만큼 제비를 준비하고 섞는다. 1개만 fail, 나머지는 pass.
 - **참가자 순서대로 각자 제비를 1개씩 뽑는다:**
 - 좌/우 화살표 키로 제비를 고르고
 - 스페이스 바(' ')로 제비 뽑기
 - 통과 / 탈락 dialog를 출력하고
 - fail 제비가 아니면 다음 사람이 뽑기
 - fail 제비를 뽑았으면 탈락하고 다음 라운드 진행

제약 조건:

사용자에게는 보이지 않더라도,
프로그램 내부에서는
라운드 시작할 때 pass/fail이
결정되어 있어야 함
제비를 뽑는 시점에 결정하지 말 것

```
*****
*                               *
* @ ? ? ?                      *
*                               *
*****
round 1, turn: player 0
```

'@': 뽑으려고 하는 제비

→ 입력

```
*****
*                               *
* ? @ ? ?                      *
*                               *
*****
round 1, turn: player 0
```

입력

```
*****
*****
** 3 player 0 pass! *
*****
*                               *
*****
round 1, turn: player 0
```

dialog

```
*****
*                               *
* @ ? ?                      *
*                               *
*****
round 1, turn: player 1
```

```
*****
*****
** 3 player 0 fail! *
*****
*                               *
*****
round 1, turn: player 0
```

dialog

```
*****
*                               *
* @ ? ?                      *
*                               *
*****
round 2, turn: player 1
```

fail 1개, pass 2개 제비를 섞어서
다음 라운드 진행

남은 3명이 1라운드 계속 진행

- 게임 종료

- 어떤 (미니)게임에서든

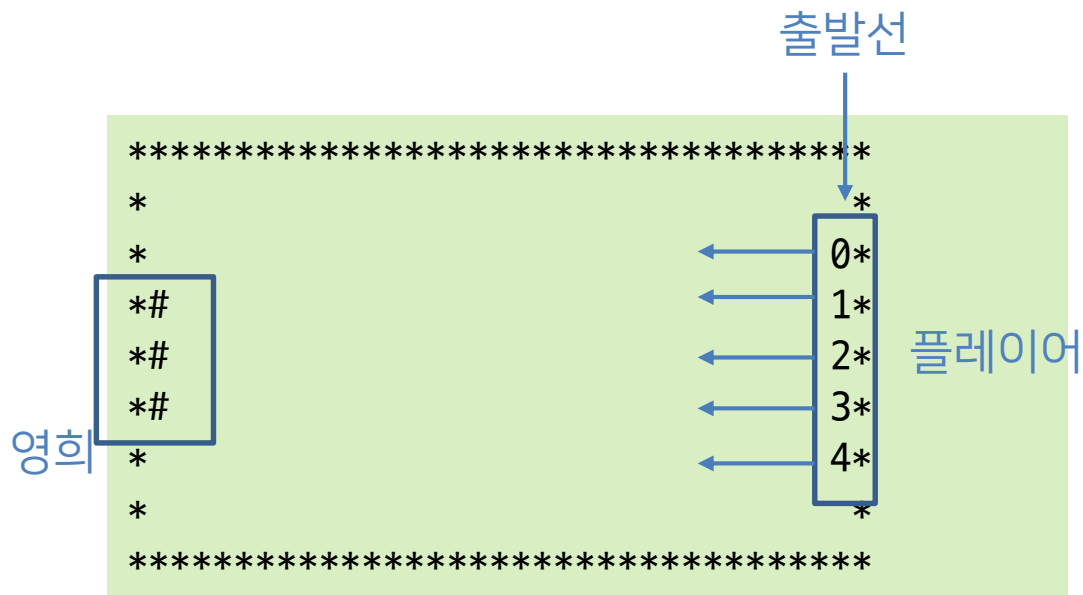
- 1명만 남으면 즉시 엔딩으로 넘어가서 쭈꾸미게임을 종료한다.

- 모든 (미니)게임에서,

- 게임마스터는 게임을 강제로 종료시킬 수 있다.
 - 'q'가 입력 되면 즉시 게임을 끝내고,
 - 그때까지 살아 남은 참가자들만 다음 게임을 진행한다.
- 마지막 제비뽑기에서 2명 이상이 남아 있을 때 'q'를 누르면
 - "우승자를 가리지 못했습니다." 메시지와 함께 살아 남은 플레이어들을 출력하고 끝낸다.

1-3) mugunghwa.c

- mugunghwa.c 파일을 만들고 mugunghwa() 작성
 - 맵 크기, 밸런스 등은 각자 알아서 맞추기
 - 2~30초 정도면 끝나도록 속도 조절
 - 플레이어는 겹칠 수 없음(앞에 다른 플레이어가 있으면 이동x)
 - 사용자는 0번 플레이어만 조종(방향키)
 - 나머지는 일정 시간마다 무작위로 움직임. 단, 각자 이동 주기가 달라야 함
 - 왼쪽 70%, 위 10%, 아래 10%, 제자리 10%




```

*****
*                               *
*                               0*
*#                              1*
*#                              2*
*#                              3*
*                               4*
*                               *
*****
무 궁 화

```



```

*****
*                               *
*                               0*
*#                              1 *
*#                              2 *
*#                              3 *
*                               4 *
*                               *
*****
무 궁 화 꽃

```



```

*****
*                               *
*                               0*
*#                              1 2*
*#                              *
*#                              43 *
*                               *
*                               *
*****
무 궁 화 꽃 이

```

• 영화는

- “무궁화꽃이피었습니다”를
- 한 글자씩 말하고,
- 3초간 멈췄다가 다시 ‘무’ 부터 시작한다.
 - 말하는 간격을
 - “무궁화 꽃이” 는 점점 느리게,
 - “피었습니다.” 는 점점 빠르게 만든다.

- '무궁화꽃이피었습니다'를 말한 후 멈춘 동안은

- 카메라가 돌아가고('#' -> '@'로 변경)
- 그동안 움직인 플레이어들은 탈락한다.
 - 사용자가 조종하지 않는 플레이어들은 10%의 확률로 움직임
 - 탈락자가 있으면 dialog로 표시, 없으면 그냥 계속
- 다시 '#'로 돌아와서 '무궁화...'
- 0번이 탈락하면 그 이후에 게임은 자동 진행

```
*****
*                                     *
*      4                             0*
*@                                     *
*@      2                             *
*@                                     *
*                                     *
*                                     *
*****
무 궁 화 꽃 이 피 었 습 니 다
```

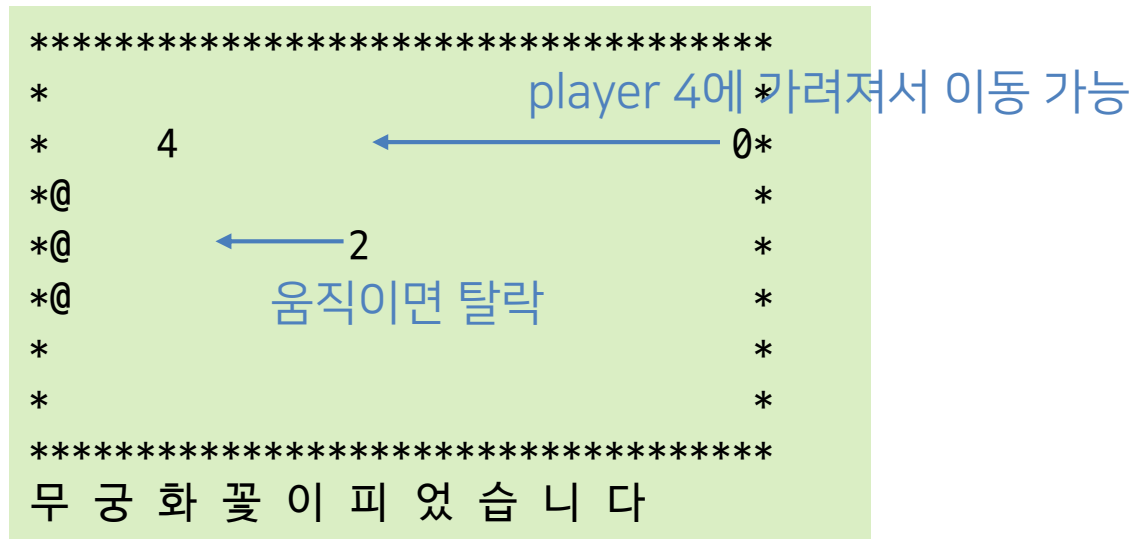
0번, 2번 탈락

```
*****
*                                     *
*      ***** 0*
*@      *         *
*@      * 3 player 0, 2 dead! *
*@      *         *
*      *****
*                                     *
*****
무 궁 화 꽃 이 피 었 습 니 다
```

```
*****
*                                     *
*      4                             *
*#                                     *
*#                                     *
*#                                     *
*                                     *
*                                     *
*****
무
```

- 영화의 시선에서

- 다른 플레이어에게 가려진 플레이어는
- 카메라가 돌아가는 동안에 움직여도 탈락하지 않는다.



- 게임 종료

- 어떤 (미니)게임에서든

- 1명만 남으면 즉시 엔딩으로 넘어가서 쭈꾸미게임을 종료한다.

- 모든 (미니)게임에서,

- 게임마스터는 게임을 강제로 종료시킬 수 있다.
 - 'q'가 입력 되면 즉시 게임을 끝내고,
 - 그때까지 살아 남은 참가자들만 다음 게임을 진행한다.
- 마지막 제비뽑기에서 2명 이상이 남아 있을 때 'q'를 누르면
 - "우승자를 가리지 못했습니다." 메시지와 함께 살아 남은 플레이어들을 출력하고 끝낸다.

과제 제출 방법

- **10월 21일(토요일)까지**

- delay: 1일 10%, 최대 5일

- **hi-class 고급프로그래밍(실) → 시험 및 설문 → 과제1 꾸꾸미게임(1) 제출**

- 팀 이름, 팀원 이름, github 주소

- **제출 시점: 최종수정일 기준**

- 제출 전 윈도우+VS에서 잘 실행되는지 반드시 확인

- **이후에 과제2는 repo 새로 만들어서 작업하기