

○ 2025.10 ○

GAME PROGRAMMING

“보물 찾기” 게임 업그레이드

발표자 : 양동민

| 학번 : 2023564058

| E-mail: ydm135@naver.com

• INDEX

목차

01

기존 코드 실행 화면

02

기존 코드 문제점

03

개선 코드 실행 화면

04

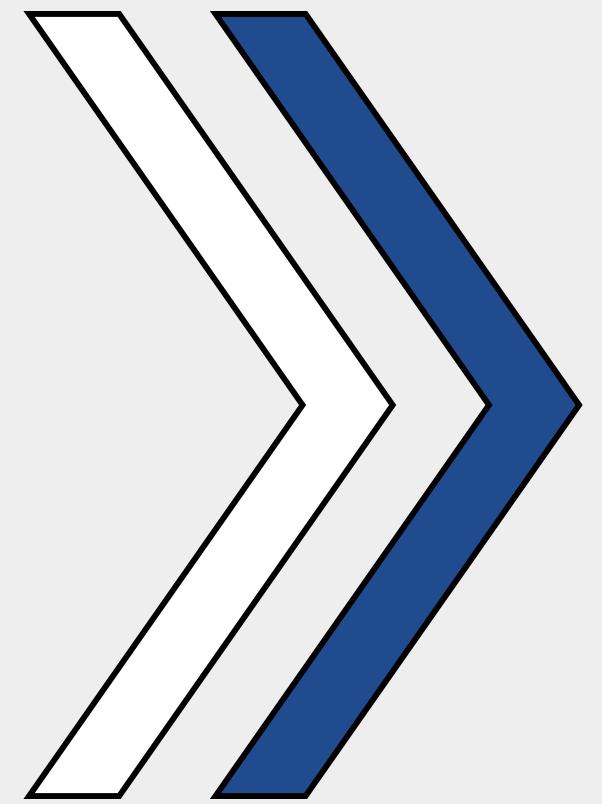
개선 및 추가 기능

05

마무리

01

기존 코드 실행 화면



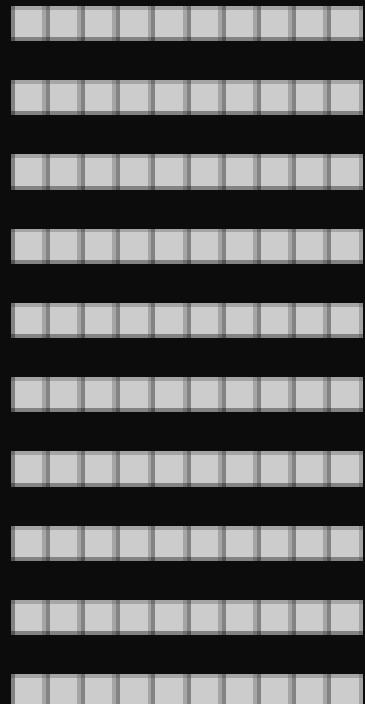
02 기존 코드 실행화면

보물찾기

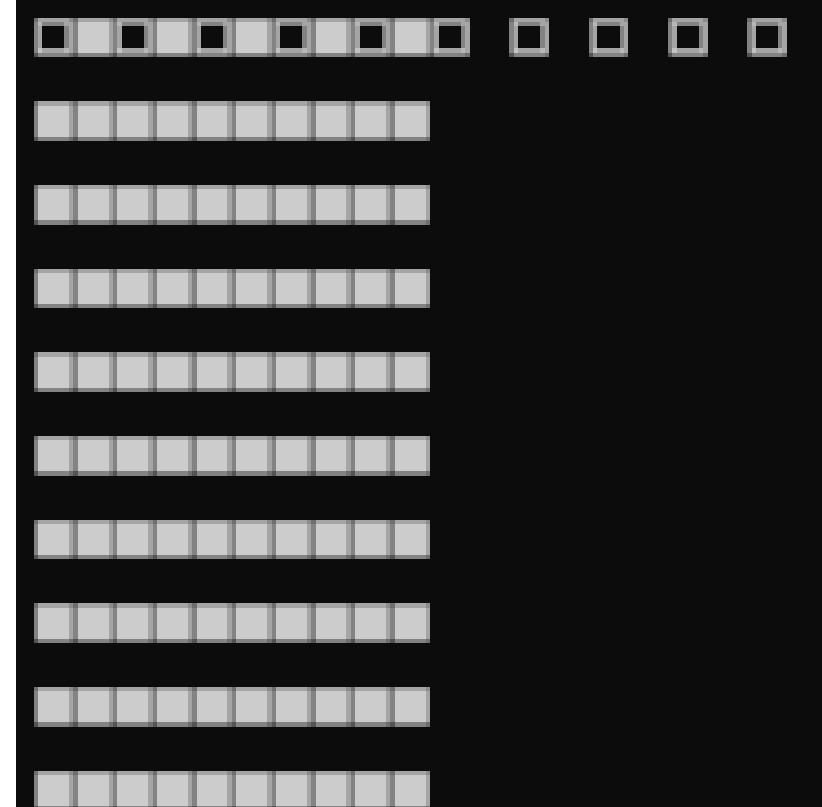
C:\Users\User\Desktop\GP\Gameprogramming\1010\C>app8_7_1.exe

아래의 바둑판 모양의 격자에는 보물이
숨겨져 있습니다.

화살표(↑↓↔)키를 움직여서 찾습니다.



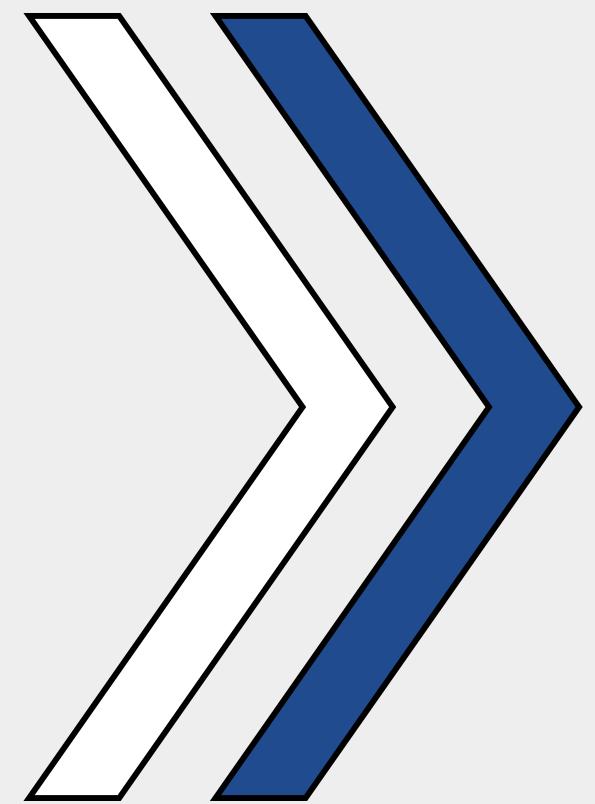
아무키나 누르면 시작합니다.



찾은 보물(★)의 개수 : 0

02

기존 코드 문제점



02 기존 코드 문제점

보물찾기

C:\Users\User\Desktop\GP\Gameprogramming\1010\C>app8_7_1.exe

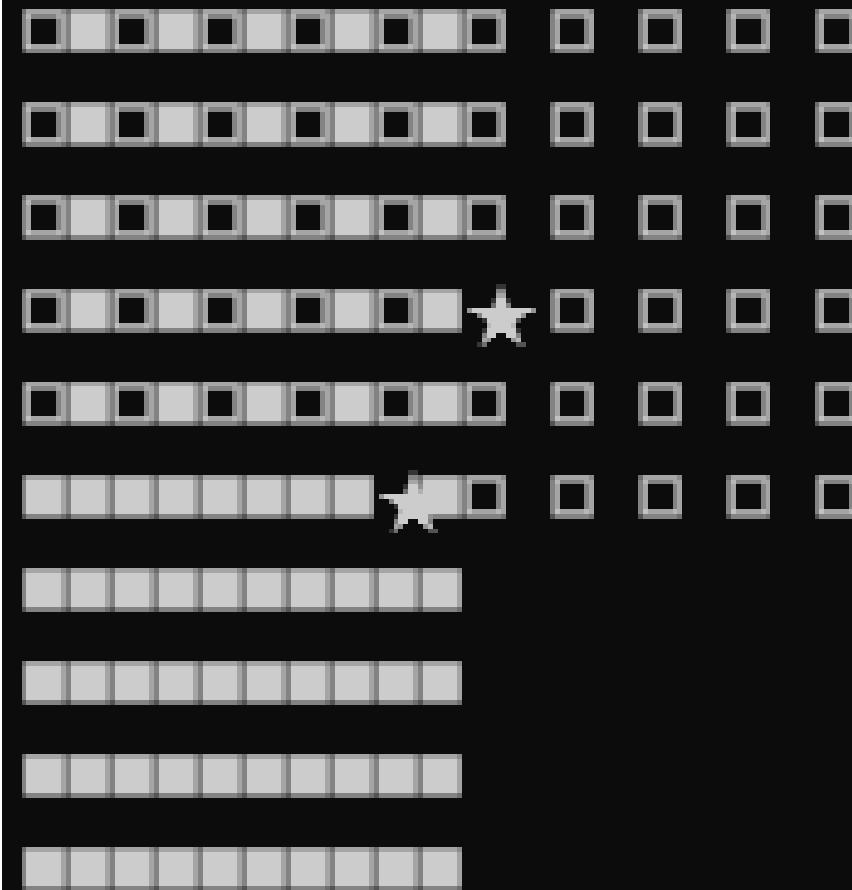
아래의 바둑판 모양의 격자에는 보물이
숨겨져 있습니다.

화살표(↑↓↔)키를 움직여서 찾습니다.

- CMD에서 실행 시 화면이 초기화되지 않아 경로와 문구가 함께 출력

02

기존 코드 문제점



찾은 보물(★)의 개수 : 2

경과 시간 : 9.1 초

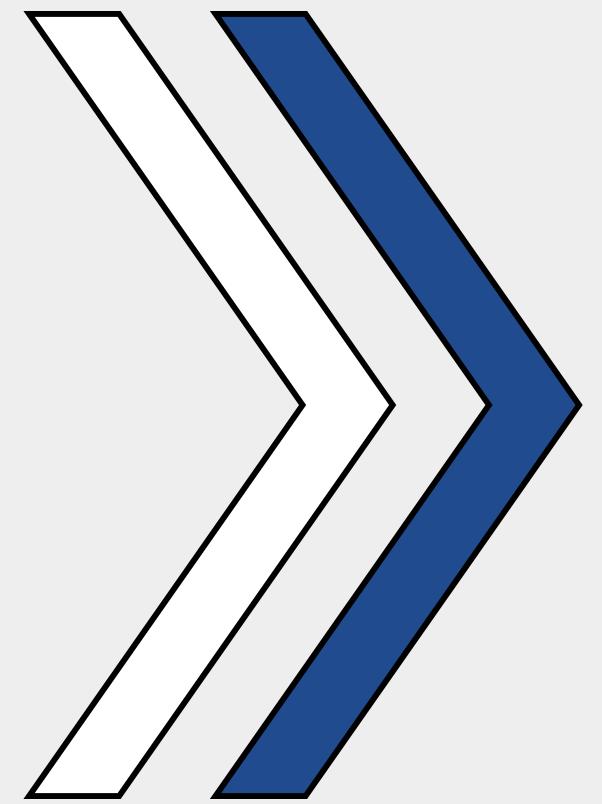
모두 찾았습니다. 게임을 종료합니다.

- 보물 개수 (2), 맵 크기 (10×10) 고정

- 맵과 이동좌표 불일치

03

개선 코드 실행 화면



03

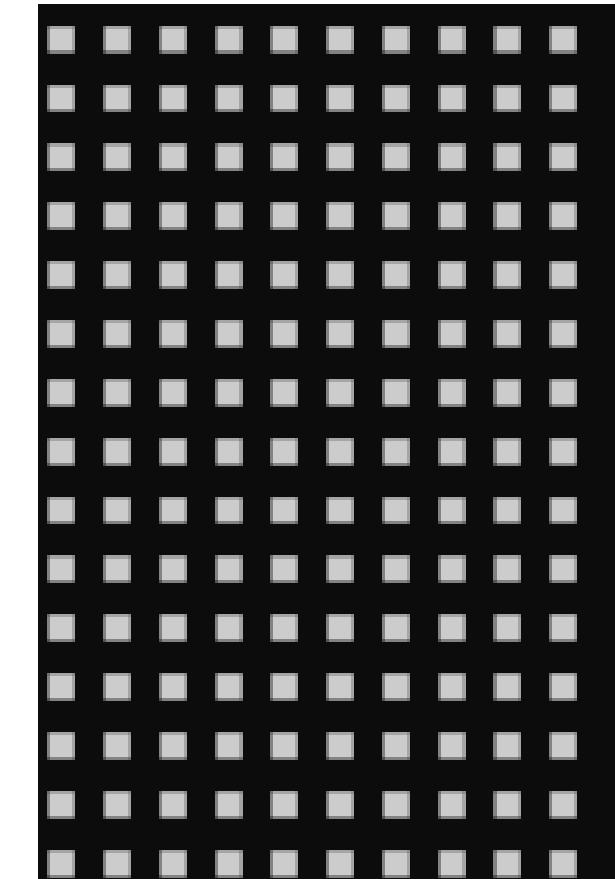
개선 코드 실행화면



03

개선 코드 실행화면

```
열을 입력하세요 (1~20): 10  
행을 입력하세요 (1~20): 15  
보물 개수를 입력하세요 (1~150): 30  
제한시간(초)을 입력하세요 : 60  
최대 이동 회수를 설정하시겠습니까? (Y/N) : Y  
최대 이동 회수를 입력하세요: 100
```



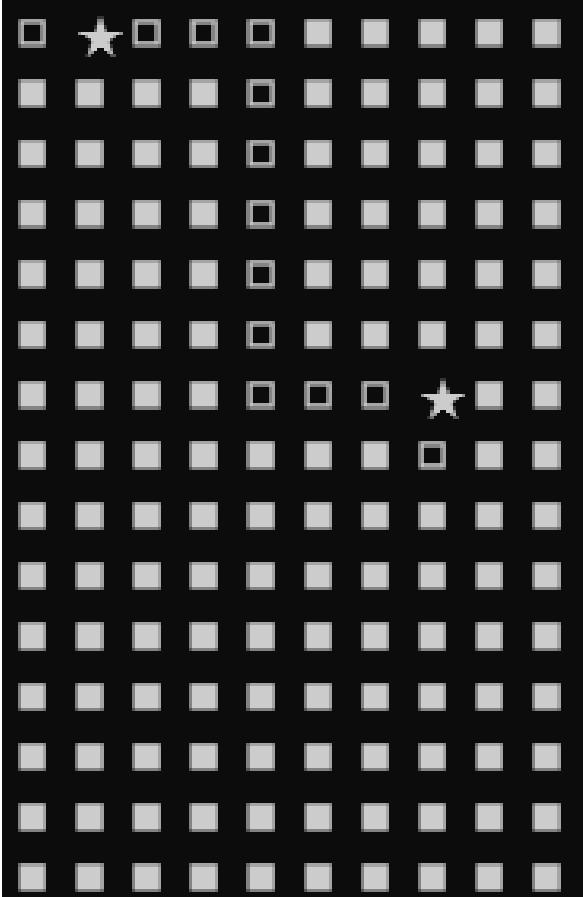
보물찾기

아래의 바둑판 모양의 격자에는 보물이 숨겨져 있습니다.
화살표($\uparrow\downarrow\leftarrow\rightarrow$)키를 움직여서 찾습니다.
ESC를 누르면 종료됩니다.

아무키나 누르면 시작합니다.

03

개선 코드 실행화면



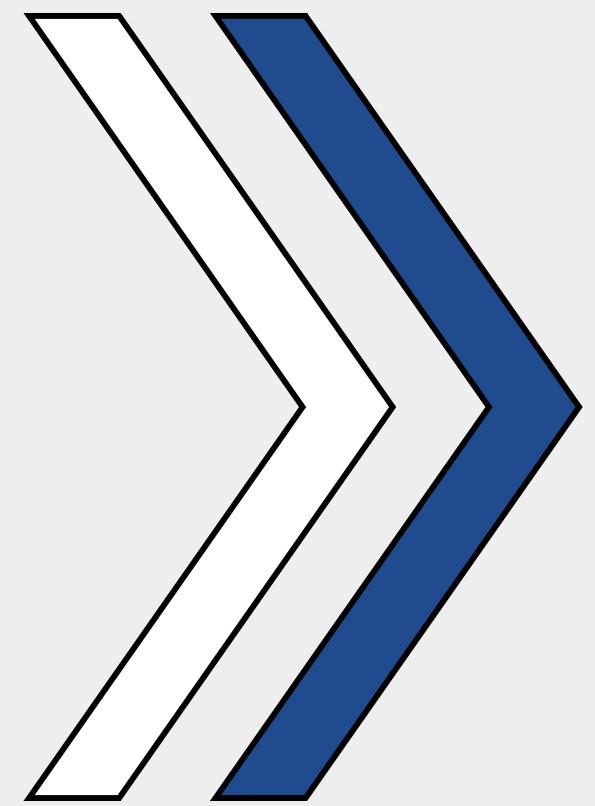
보물찾기

아래의 바둑판 모양의 격자에는 보물이
숨겨져 있습니다.
화살표($\uparrow\downarrow\leftarrow\rightarrow$)키를 움직여서 찾습니다.
ESC를 누르면 종료됩니다.

이동 회수 : 14 / 100
찾은 보물(★)의 개수 : 2 / 30
경과 시간 : 37.4초 / 제한 : 60초

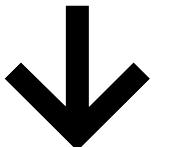
04

개선 및 추가 기능



04 개선 및 추가 기능

```
void display_rule(void);
void gotoxy(int x, int y);
void make_treasure(int tx[], int ty[]);
void display_map(int matrix[][][10], int tx[], int ty[]);
void basic_map(void);
void move_arrow_key(char key, int *x1, int *y1, int x_b, int y_b);
void game_control(int tx[], int ty[]);
```



```
void display_rule(int map_x, int map_y);
void gotoxy(int x, int y);
void make_treasure(int tx[], int ty[], int map_x, int map_y, int star);
void display_map(int matrix[][][41], int tx[], int ty[], int map_x, int map_y);
void basic_map(int map_x, int map_y);
void move_arrow_key(char key, int *x1, int *y1, int x_b, int y_b, int *move_count);
void game_control(int tx[], int ty[], boolean *gameover, int map_x, int map_y, int star, int limit_time, int limit_move);
void intro(void);
```

04

개선 및 추가 기능 (main)

```
int tx[400], ty[400]; // 보물 개수 최대 400
int matrix[41][41]={0}; // 이동 좌표
int map_x, map_y, star; // 보물, 맵 크기 지정
boolean gameover = FALSE; // 게임 오버
clock_t start, end; // 시작, 종료 시간
double pst; // 경과 시간
int limit_time; // 제한 시간
int limit_move = -1; // 이동 회수 제한
char answer; // 이동 회수 제한 여부
```

04

개선 및 추가 기능 (main)

```
do
{
    printf("열을 입력하세요 (1~20): ");
    if (scanf("%d", &map_x) != 1 || map_x < 1 || map_x > 20)
    {
        printf("잘못된 입력입니다. 1~20 사이의 숫자를 입력하세요.\n");
        while (getchar() != '\n')
            ;
        continue;
    }
    break;
} while (1);
```

* 행, 보물개수, 제한시간, 이동회수도 동일한 흐름

04 개선 및 추가 기능 (display_rule)

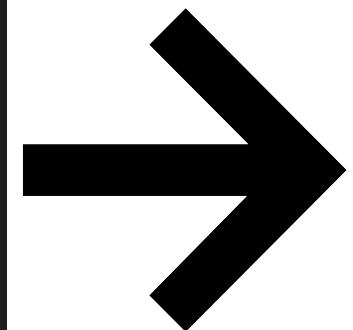
```
void display_rule(int map_x, int map_y)
{
    gotoxy(map_x * 2 + 3, 1);
    printf("보물찾기");
    gotoxy(map_x * 2 + 3, 3);
    printf("아래의 바둑판 모양의 격자에는 보물이 \n");
    gotoxy(map_x * 2 + 3, 4);
    printf("숨겨져 있습니다. \n");
    gotoxy(map_x * 2 + 3, 5);
    printf("화살표(↑↓↔)키를 움직여서 찾습니다. \n");
    gotoxy(map_x * 2 + 3, 6);
    printf("ESC를 누르면 종료됩니다. \n");
    basic_map(map_x, map_y);
    gotoxy(1, map_y + 2);
    if (!kbhit())
    {
        printf("\n 아무키나 누르면 시작합니다. \n");
        getch();
    }
    else
    {
        printf("");
    }
}
```

보물찾기

아래의 바둑판 모양의 격자에는 보물이
숨겨져 있습니다.
화살표(↑↓↔)키를 움직여서 찾습니다.
ESC를 누르면 종료됩니다.

04 개선 및 추가 기능 (basic_map)

```
void basic_map(void)
{
    int i, j;
    for(i=0;i<10;i++)
    {
        for(j=0;j<10;j++)
            printf("■");
        printf("\n");
    }
}
```



```
void basic_map(int map_x, int map_y)
{
    int i, j;
    for (i = 1; i <= map_x; i++)
    {
        for (j = 1; j <= map_y; j++)
        {
            gotoxy(i * 2 - 1, j);
            printf("■");
        }
    }
}
```

04 개선 및 추가 기능 (make_treasure)

```
void make_treasure(int tx[], int ty[], int map_x, int map_y, int star)
{
    int total = map_x * map_y;
    int allx[400], ally[400];
    int i, j, idx = 0;
```

1. 입력한 맵 크기만큼 좌표 채우기
2. 랜덤 함수로 좌표 섞기
3. 입력한 보물 개수만큼 앞쪽 좌표를 보물 위치로 선택

```
// 모든 좌표 채우기
for (i = 1; i <= map_x; i++)
    for (j = 1; j <= map_y; j++)
    {
        allx[idx] = i;
        ally[idx] = j;
        idx++;
    }

// 셔플
for (i = total - 1; i > 0; i--)
{
    int r = rand() % (i + 1);
    int tmpx = allx[i];
    int tmpy = ally[i];
    allx[i] = allx[r];
    ally[i] = ally[r];
    allx[r] = tmpx;
    ally[r] = tmpy;
}
```

```
// 앞쪽 star개만 복사
for (i = 0; i < star; i++)
{
    tx[i] = allx[i];
    ty[i] = ally[i];
```

04 개선 및 추가 기능 (display_map)

```
void display_map(int matrix[][41], int tx[], int ty[], int map_x, int map_y)
{
    int i, j;
    basic_map(map_x, map_y);
    for (i = 1; i <= map_x; i++)
        for (j = 1; j <= map_y; j++)
            if (matrix[i][j] == 1)
            {
                gotoxy(i * 2 - 1, j);
                printf("□");
            }
            else if (matrix[i][j] == 2)
            {
                gotoxy(i * 2 - 1, j);
                printf("★");
            }
}
```

맵 좌표 위에 현재 플레이어 위치와 보물 표시

- 현재 위치 → □

- 보물 위치 → ★

04 개선 및 추가 기능 (move_arrow_key)

```
void move_arrow_key(char key, int *x1, int *y1, int x_b, int y_b, int *move_count)
{
    switch (key)
    {
        case 72: // 위쪽(상) 방향의 화살표 키 입력
            *y1 = *y1 - 1;
            if (*y1 < 1)
            {
                *y1 = 1;
                *move_count = *move_count;
            } // y좌표의 최소값
            else
            {
                (*move_count)++;
            }
            break;
    }
}
```

- 이동 회수 기록 기능 추가
- 플레이어가 맵 경계 밖으로 나가려 하면
이동 횟수 증가하지 않음

* 상, 하, 좌, 우 동일한 흐름

04

개선 및 추가 기능 (game_control)

```
void game_control(int tx[], int ty[], boolean *gameover, int map_x, int map_y, int star, int limit_time, int limit_move)
{
    char key;
    int i, j, count = 0, move_count = 0;
    int x = 1, y = 1;
    int matrix[41][41] = {0};
    clock_t start, now;
    double elapsed;

    start = clock(); // ◆ 시작 시각 기록
    display_rule(map_x, map_y);
```

- 화살표 키로 플레이어 이동, 보물 탐색 및 경과 시간/이동 횟수 관리
- 제한 시간·이동 횟수 초과 시 자동 종료

04

개선 및 추가 기능 (game_control)

```
// ◆ 키가 눌렸을 때만 이동 처리
if (_kbhit())
{
    key = getch();
    if (key == 27) // ESC 종료
    {
        gotoxy(1, map_y + 7);
        printf("게임을 종료합니다.....\n");
        *gameover = TRUE;
        return;
    }
    move_arrow_key(key, &x, &y, map_x * 2 - 1, map_y, &move_count);
}
```

- 키 값 받아서 플레이어 위치 이동
- ESC 눌렀을 경우 게임 종료

04

개선 및 추가 기능 (game_control)

```
for (i = 0; i < star; i++)
    if (((x + 1) / 2 == tx[i]) && (y == ty[i]))
        matrix[(x + 1) / 2][y] = 2;

gotoxy(1, 1);
display_map(matrix, tx, ty, map_x, map_y);

count = 0;
for (i = 1; i <= map_x; i++)
    for (j = 1; j <= map_y; j++)
        if (matrix[i][j] == 2)
            count++;
```

- 플레이어가 이동한 위치에 보물이 있으면 matrix에 표시
- 현재 찾은 보물 개수 계산

04

개선 및 추가 기능 (game_control)

```
now = clock();
elapsed = (double)(now - start) / CLK_TCK; // ◆ 경과 시간 계산

// ◆ 제한시간 초과 체크 (항상 수행)
if (elapsed >= limit_time)
{
    gotoxy(1, map_y + 7);
    printf("시간초과! 게임을 종료합니다.\n");
    getch();
    *gameover = TRUE;
    return;
}
```

- 제한 시간 초과 시 자동 종료
- elapsed → 게임 시작부터 경과 시간

04

개선 및 추가 기능 (game_control)

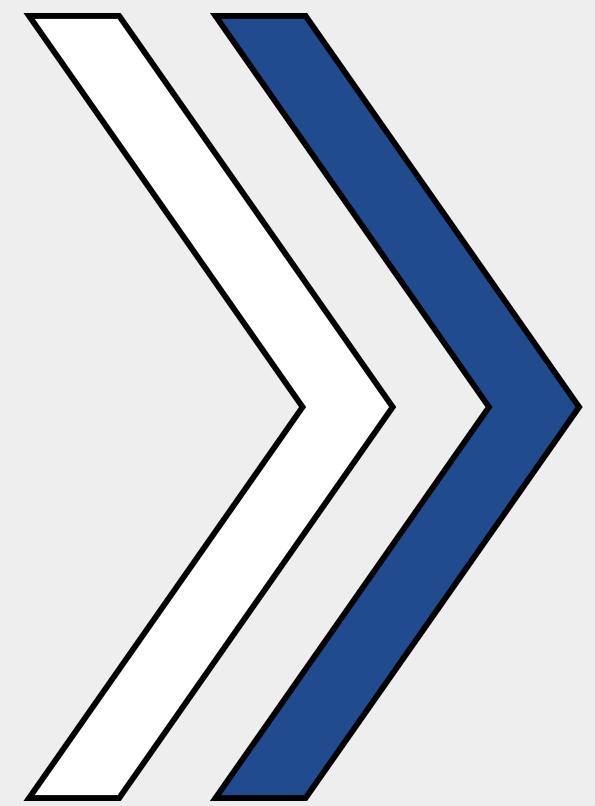
```
if (move_count > limit_move && limit_move != -1)
{
    gotoxy(1, map_y + 7);
    printf("이동 회수 초과! 게임을 종료합니다. \n");
    getch();
    *gameover = TRUE;
    return;
}
```

- 플레이어 이동 횟수가 제한을 초과하면 자동 종료
- 회수 지정을 안했을 경우 무시

04 | 개선 및 추가 가능 (intro)

05

마무리



05 마무리

정리

개선 기능

- 맵과 이동 좌표 혼동 개선
- 화면 지우기를 통해 가독성 향상

추가 기능

- 맵 크기, 이동 회수, 제한 시간, 보물 개수 설정
- 인트로 화면 추가

힘든 점

- 타인의 기존 코드 수정

○ 2025.10 ○

GAME PROGRAMMING

“감사합니다”

발표자 : 양동민

| 학번 : 2023564058

| E-mail: ydm135@naver.com